

# Компьютерное зрение

## Лекция 11. AE, VAE, GANS

04.07.2020  
Руслан Алиев

## Unsupervised learning

## Supervised learning

- Labels or target classes
- Goal: learn a mapping from input to label
- Classification, regression

## Unsupervised learning

## Supervised learning

CAT



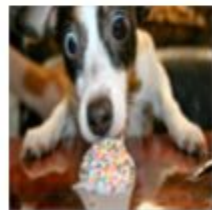
DOG

DOG



CAT

CAT



DOG

## Unsupervised learning

- No label or target class
- Find out properties of the structure of the data
- Clustering (k-means, PCA)

## Supervised learning

CAT



DOG

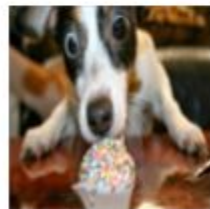


DOG



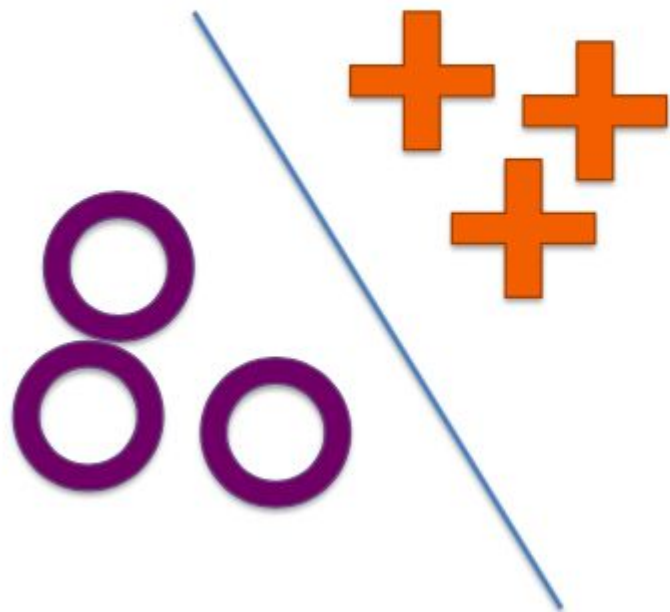
CAT

CAT



DOG

## Unsupervised learning



## Supervised learning

CAT



DOG



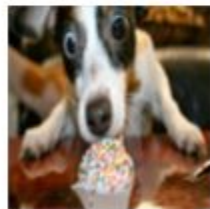
DOG



CAT

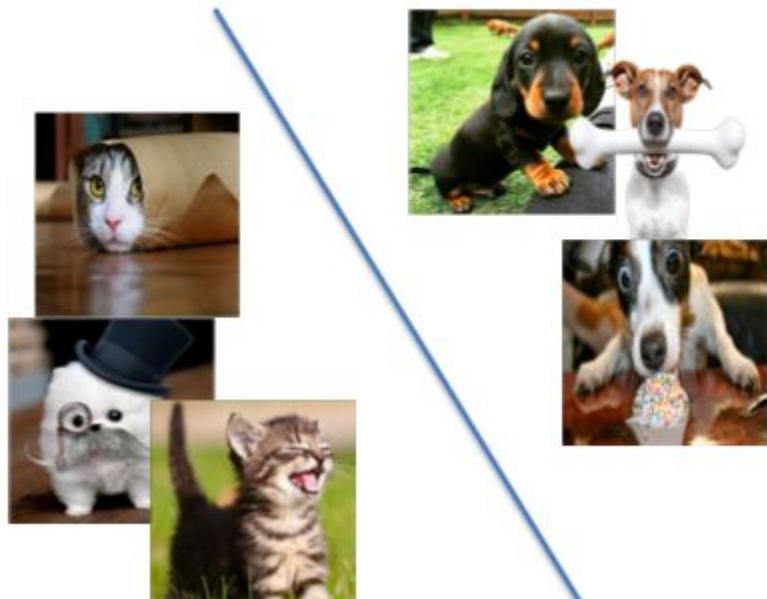


CAT



DOG

## Unsupervised learning



## Supervised learning





# Unsupervised learning with autoencoder

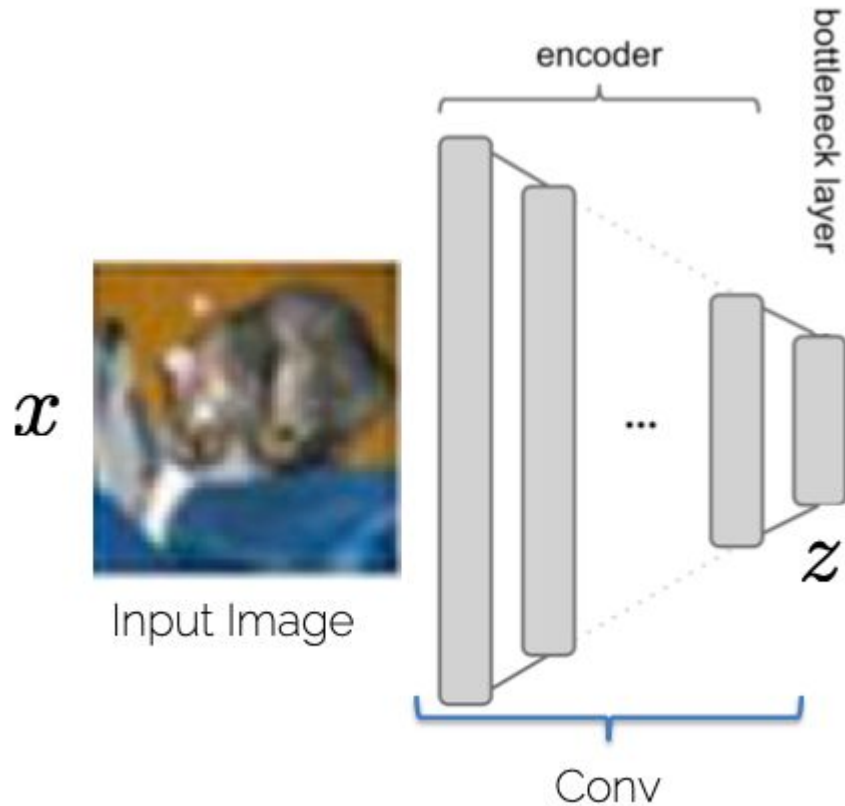
# Автоэнкодеры

---

Учим модель находить низкоразмерное представление  
неразмеченных данных



# Автоэнкодеры



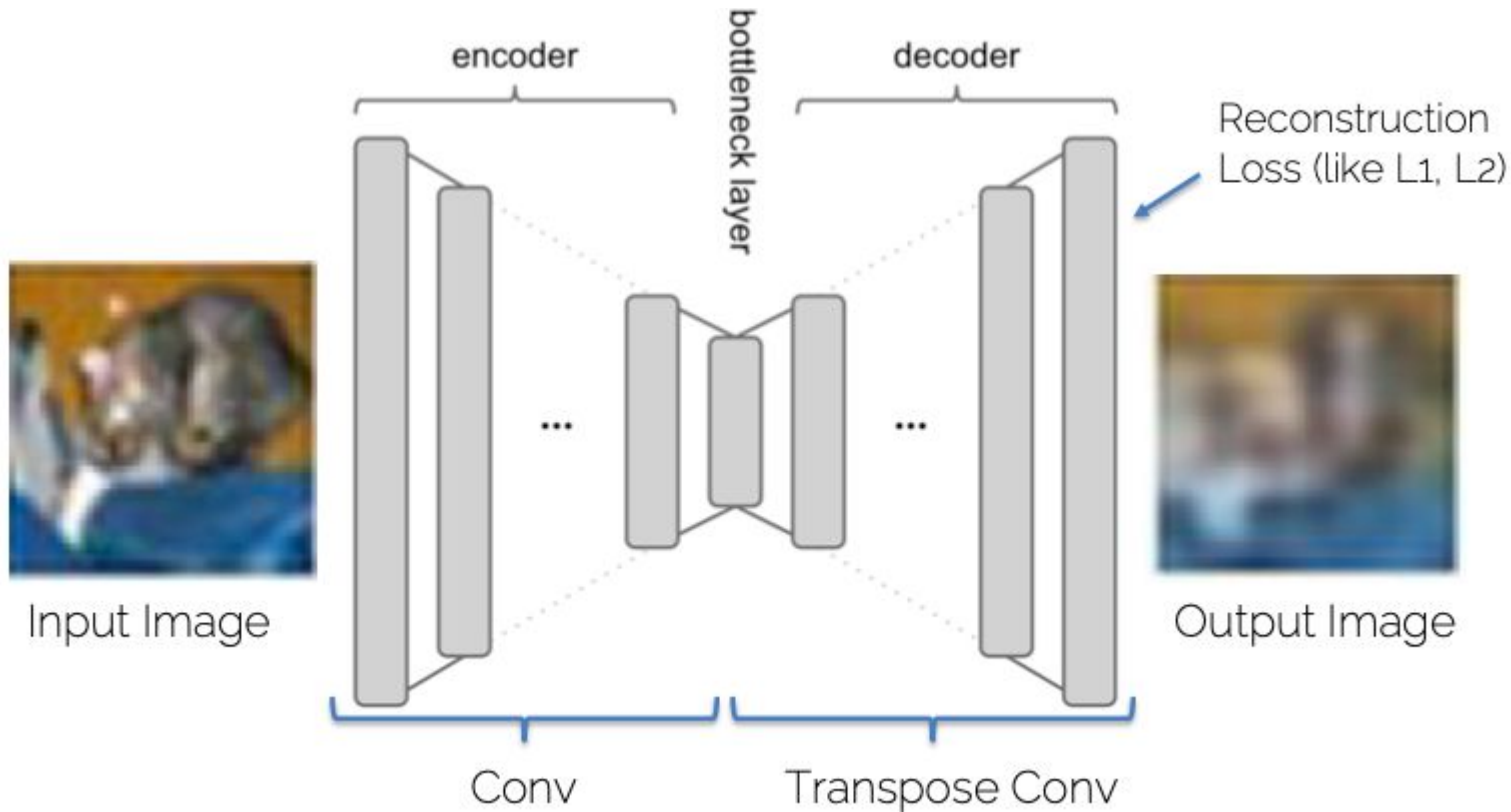
- Их входного изображения в эмбединг (bottleneck layer)
- Encoder: CNN

# Автоэнкодеры

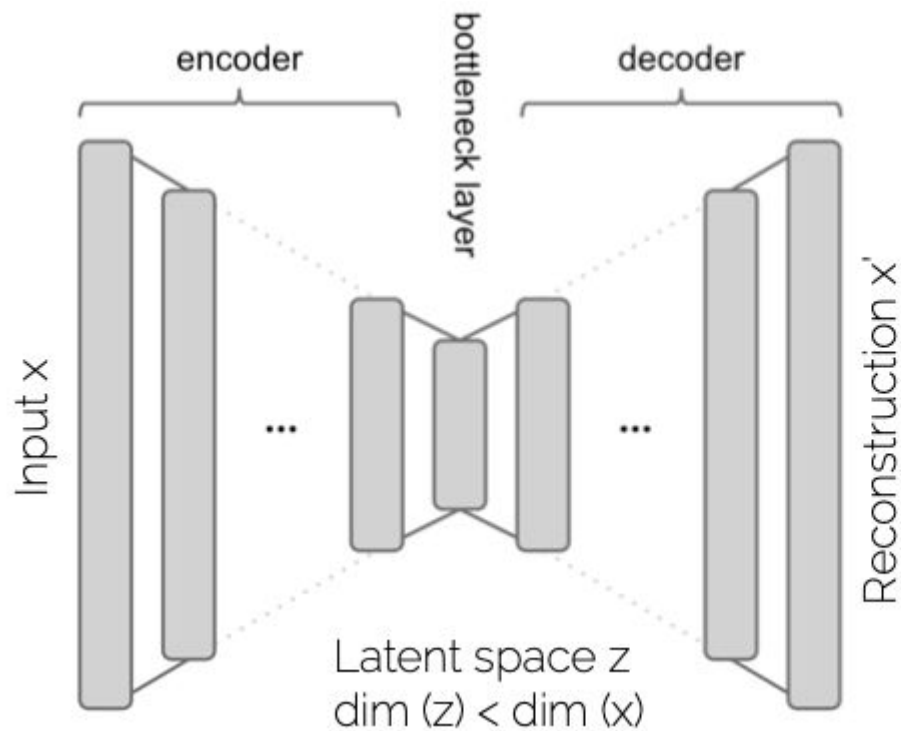
---

- Зачем нужно это понижение размерности?
- Чтобы найти паттерны, самые значимые факторы в данных

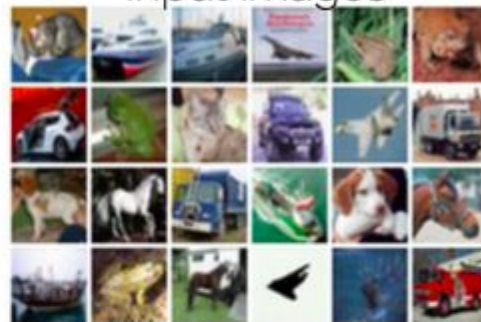
# Автоэнкодеры (train)



# Автоэнкодеры



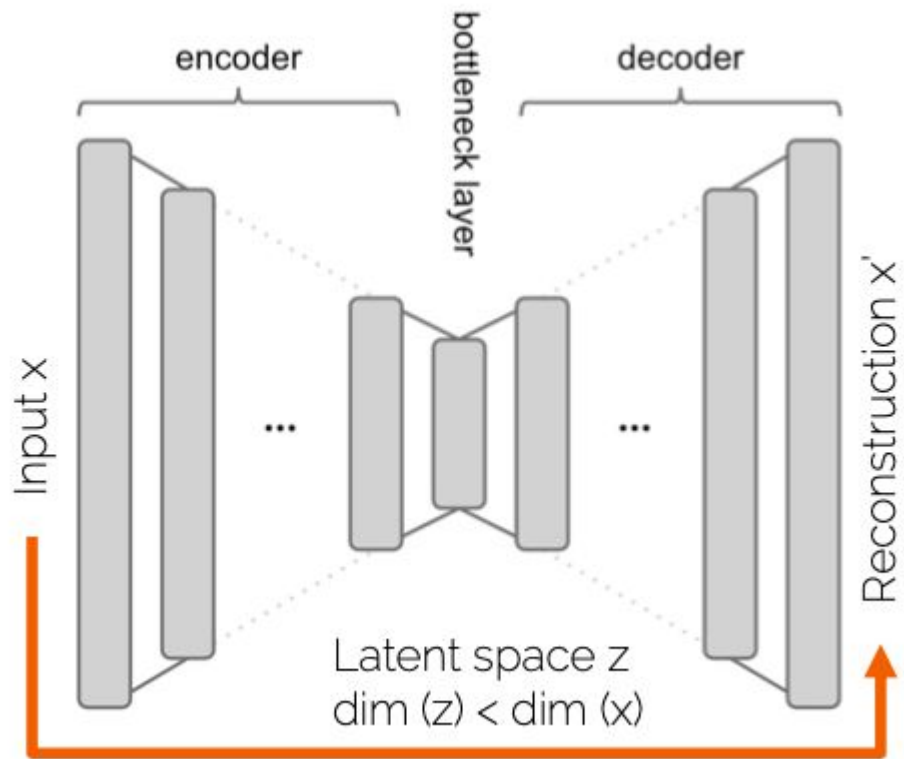
Input images



Reconstructed images



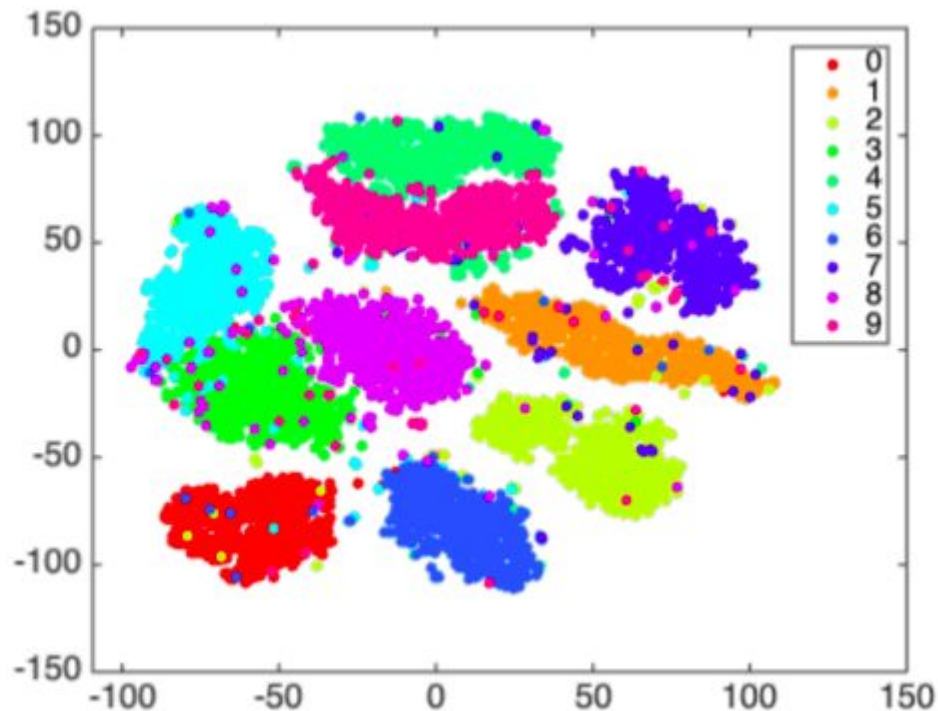
# Автоэнкодеры



- Не нужны лейблы
- Мы можем использовать неразмеченные данные, чтобы понять их структуру

# Автоэнкодеры

Embedding of  
MNIST numbers



# Автоэнкодеры для пре-трейна

---

- Медицинские приложения для DL (Анализ КТ-снимков)
  - Много неразмеченных снимков
  - Мало размеченных снимков
- Мы не можем взять модель предобученную на ImageNet.  
Почему?

# Автоэнкодеры для пре-трейна

---

- Медицинские приложения для DL (Анализ КТ-снимков)
  - Много неразмеченных снимков
  - Мало размеченных снимков
- Мы не можем взять модель предобученную на ImageNet.  
Почему?
- Различается природа снимков, совсем разные фичи



# Автоэнкодеры для пре-трейна

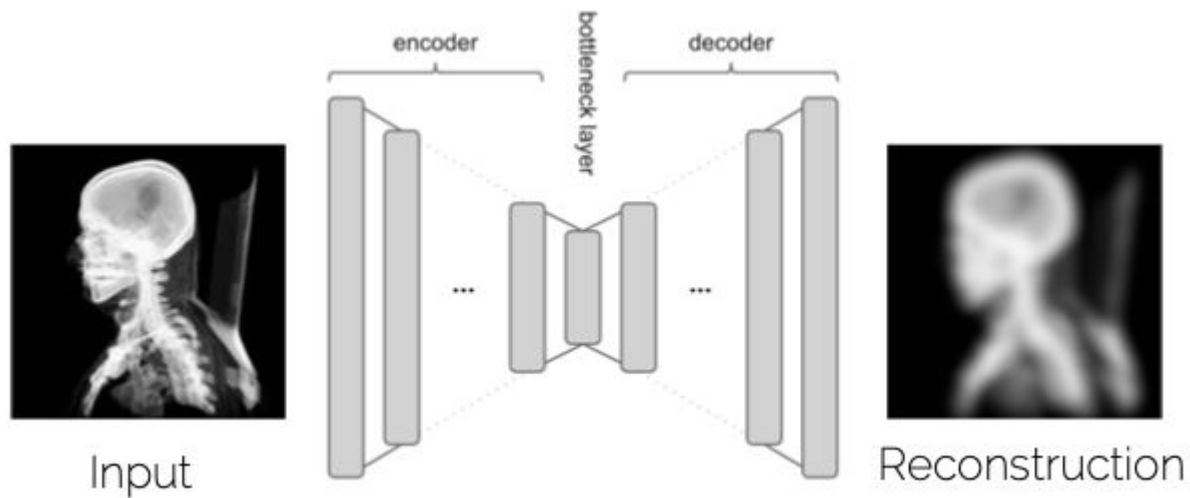
---

- Медицинские приложения для DL (Анализ КТ-снимков)
  - Много неразмеченных снимков
  - Мало размеченных снимков
- Мы можем: предобучить сетку с автоэнкодером чтобы она научилась распознавать паттерны в КТ-снимках

# Автоэнкодеры

---

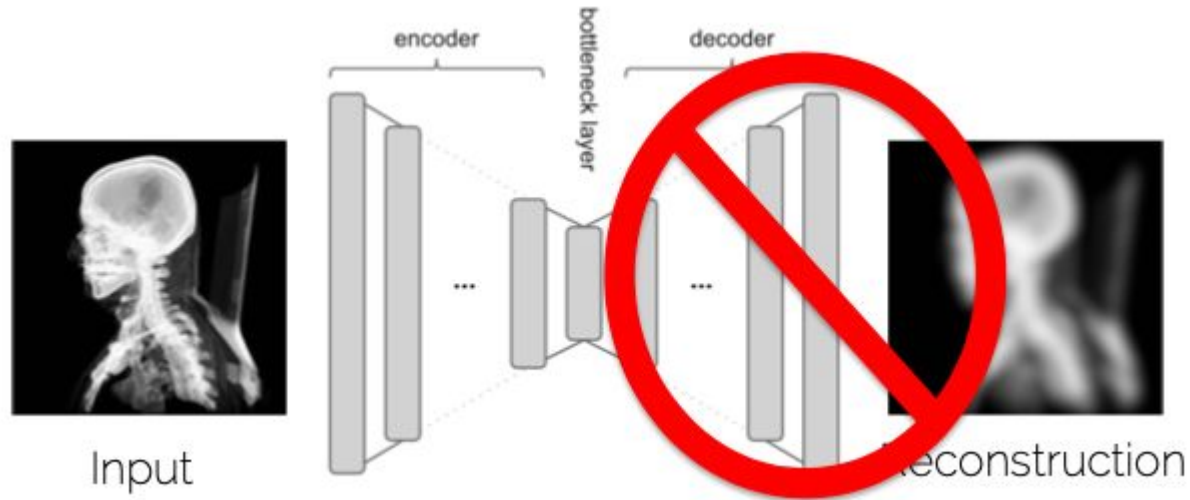
## Шаг 1. Обучаем автоэнкодер



# Автоэнкодеры

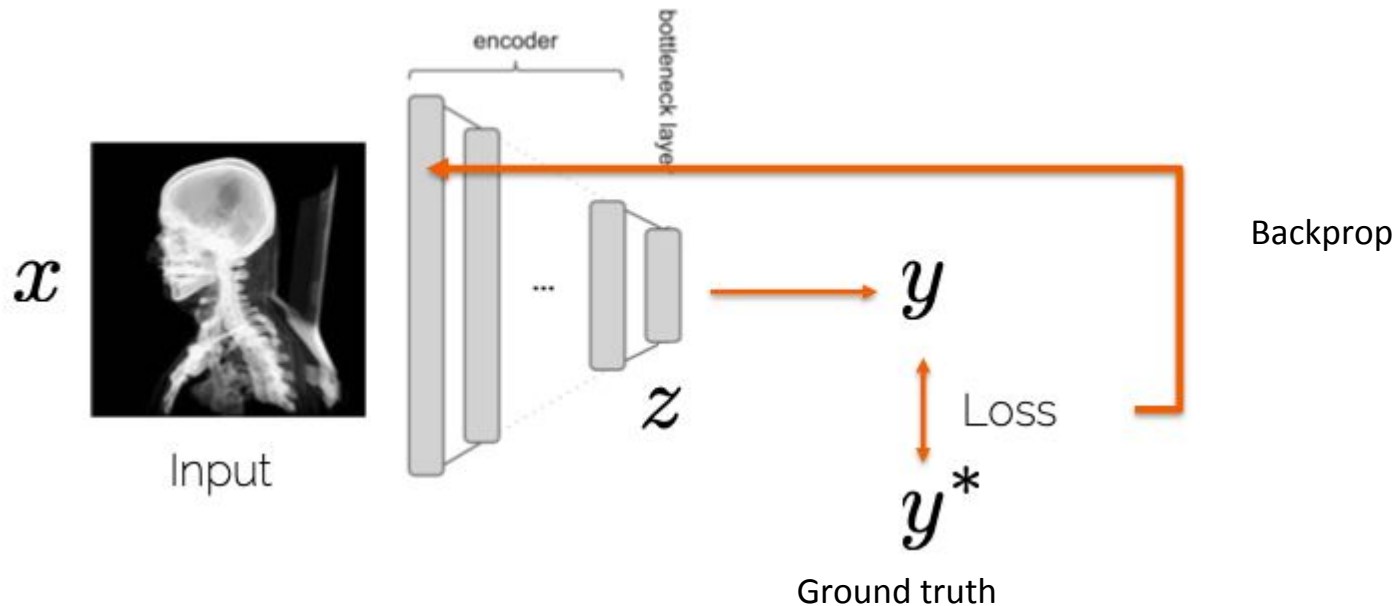
---

Шаг 2. Убираем часть с декодером, обучаем на размеченных данных



# Автоэнкодеры



Шаг 2. Убираем часть с декодером, обучаем на размеченных данных



# Автоэнкодеры: use cases

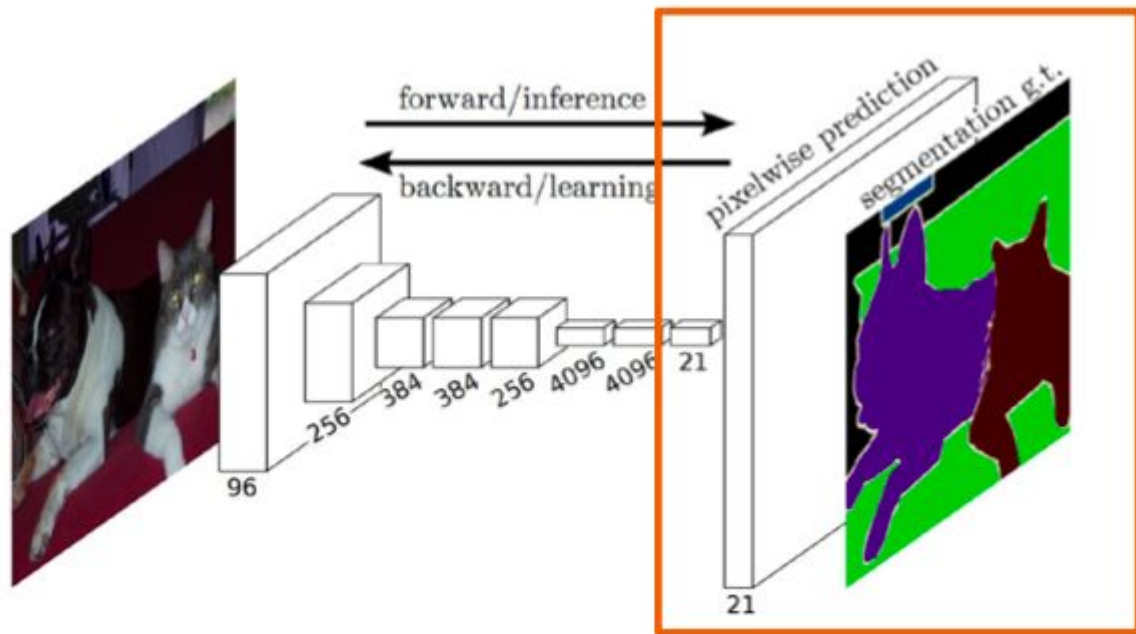
---

- Use 1: предобучаем по схеме рассказанной выше
  - Изображение -> изображение
  - Используем энкодер для извлечения фичей
- Use 2: чтобы получать pixel-level predictions
  - Image -> semantic segmentation
  - Low-resolution image -> high resolution image
  - Image -> Depth Map

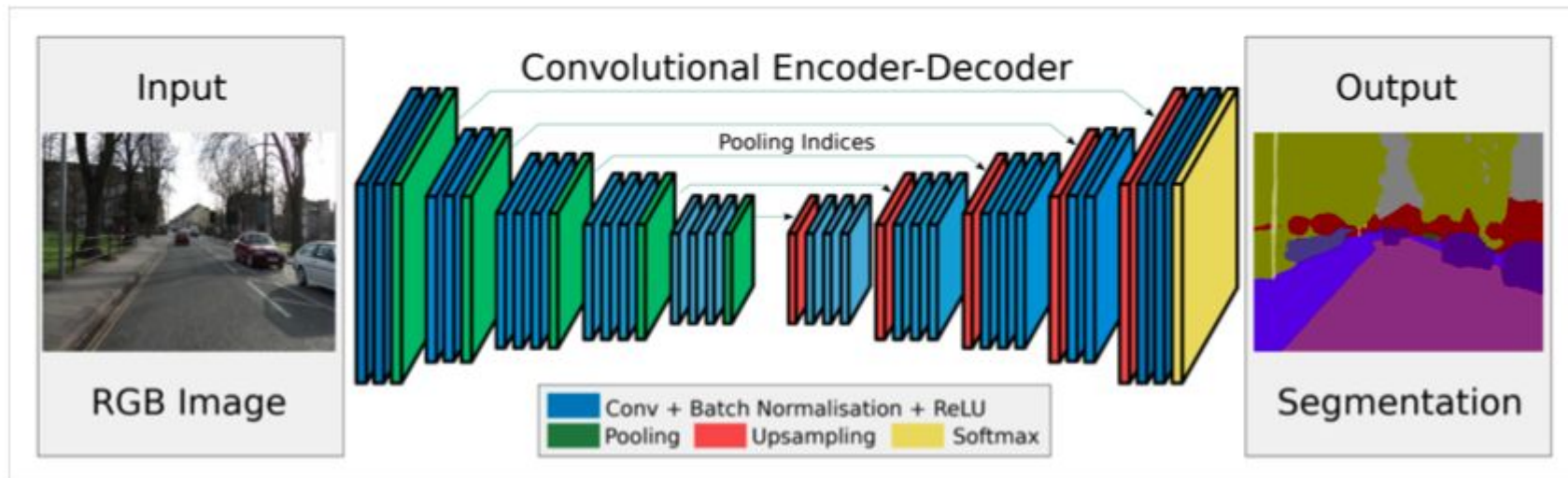


# Задачи связанные с автоэнкодерами

# FCN



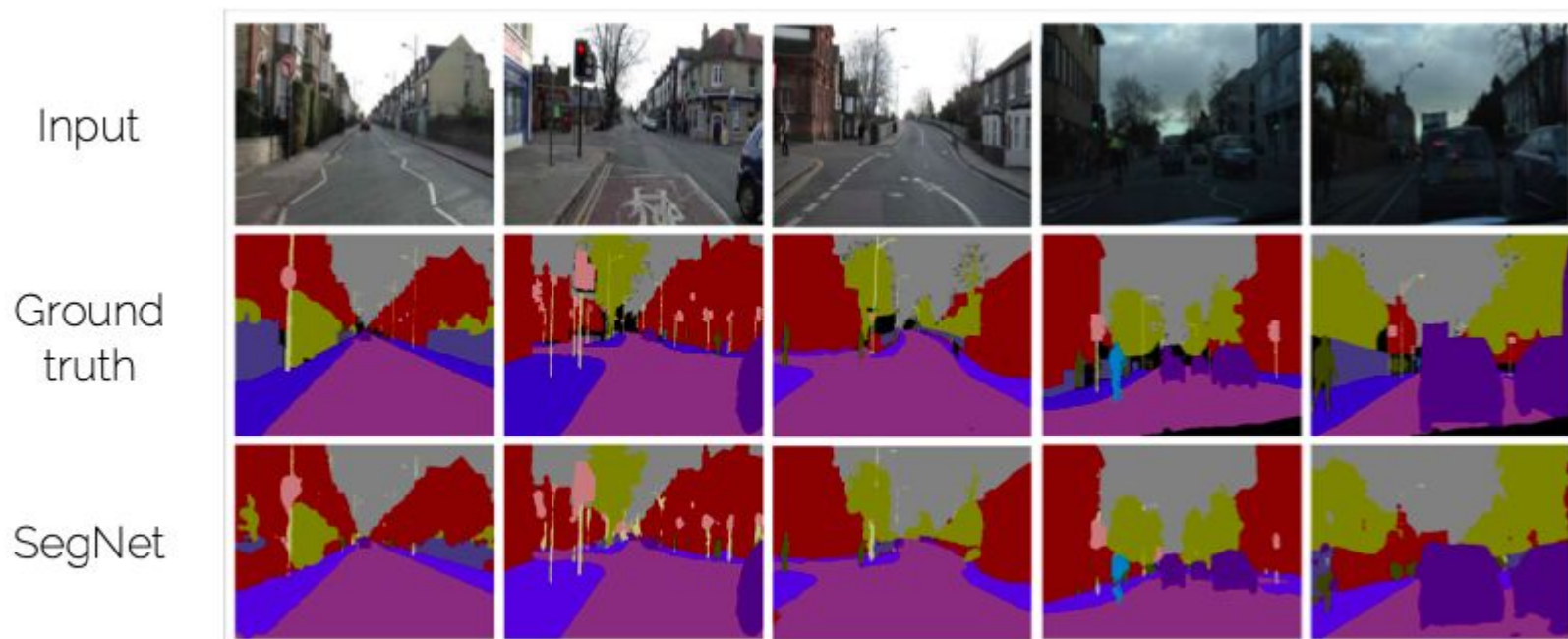
# SegNet





# SegNet

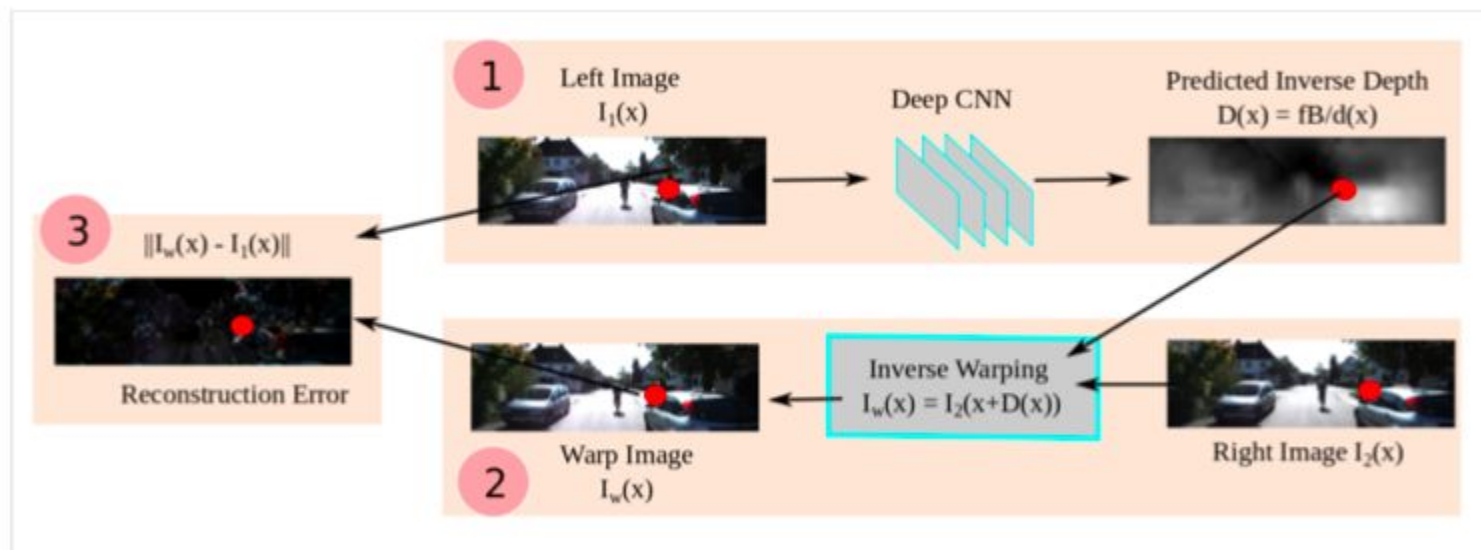
---



Badrinarayanan et al. „SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation“. TPAMI 2016

# Monocular Depth

Обучение без разметок!



R. Garg et al. „Unsupervised CNN for Single View Depth Estimation: Geometry to the Rescue“ ECCV 2016

# Image super-resolution

---

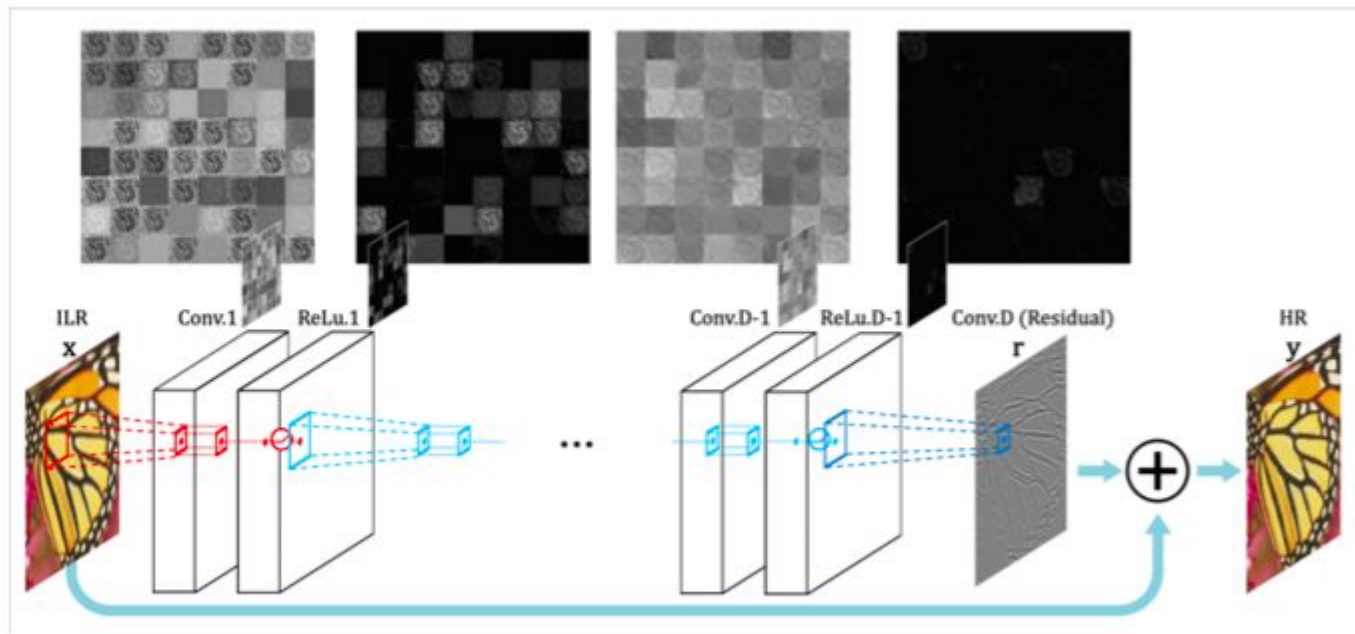
Изображение в низком разрешении -> изображение в высоком разрешении

Проблемы:

Как-то нужно провести все низкоуровневые паттерны изображения через всю сетку

# Image super-resolution

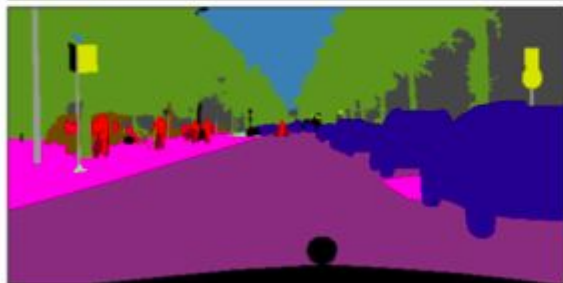
Почему бы не обучать только residual? Гораздо легче!



J. Kim et al. „Accurate Image Super-Resolution Using Very Deep Convolutional Networks“. CVPR 2016

# Image synthesis

Semantic segmentation image → Real image



(a) Input semantic layouts

(b) Synthesized images

Q. Chen and V. Koltun, "Photographic Image Synthesis with Cascaded Refinement Networks". ICCV 2017

## Image synthesis

---

Можно использовать *perceptual loss* чтобы добиться хороших результатов

Нельзя использовать L2-loss, он наказывает за реалистичные результаты (черная машина/белая машина)

*Perceptual loss* измеряет “контент изображения”

## Perceptual loss (aka content loss)

---

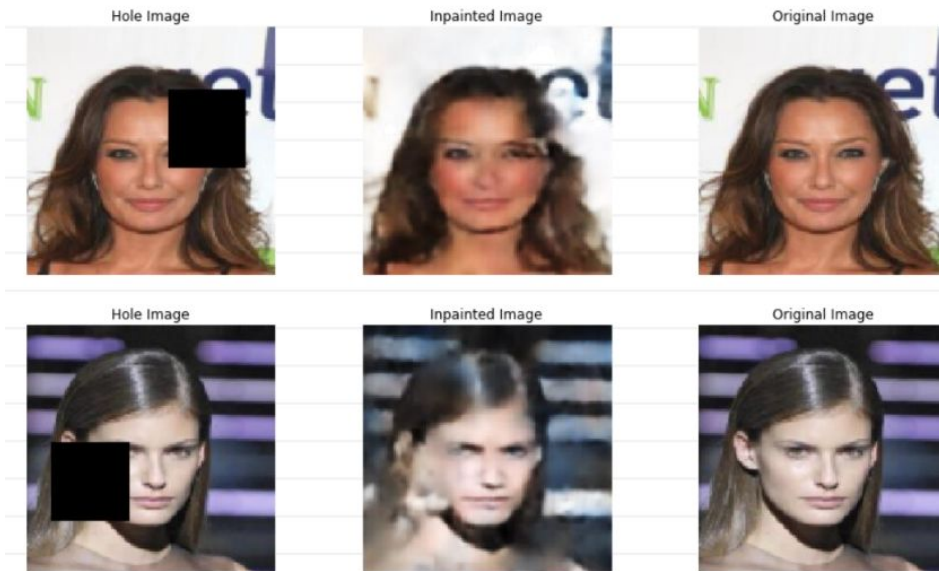
1. Берем предобученную VGG для классификации
2. Форвардпассим сгенерированное изображение и настоящее изображение через сетку
3. Сравниваем feature maps

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

Feature map size (channels,  
height, width)

Feature maps of the ground  
truth image at layer j

# AE denoising







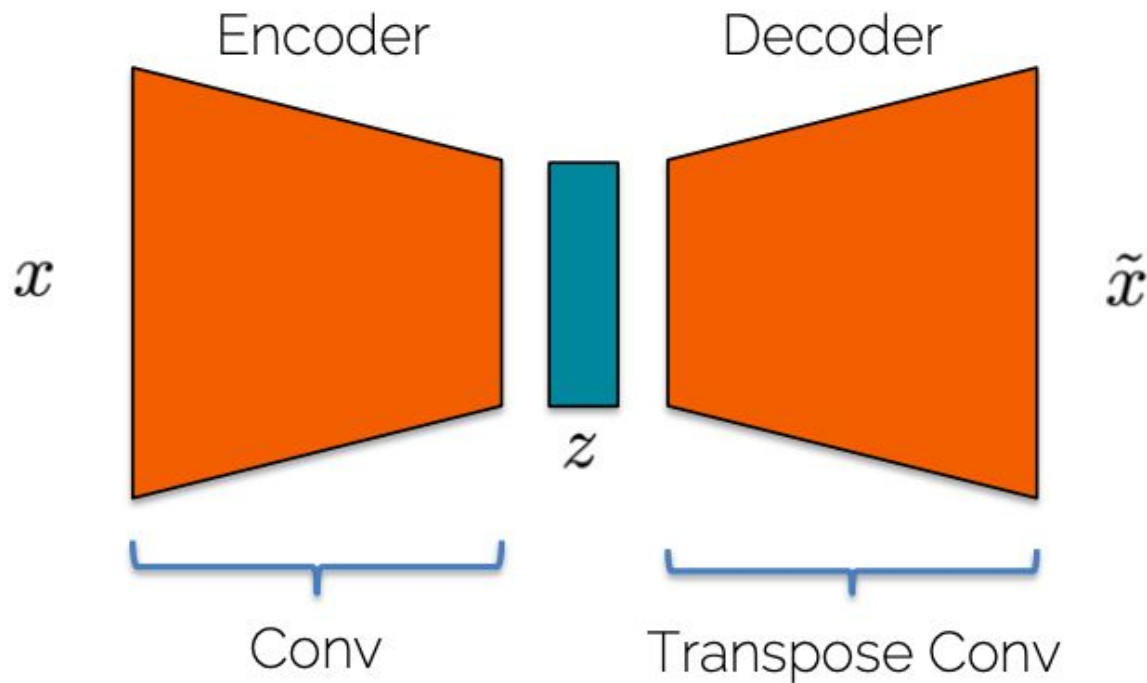
# Variational autoencoders



# Autoencoders

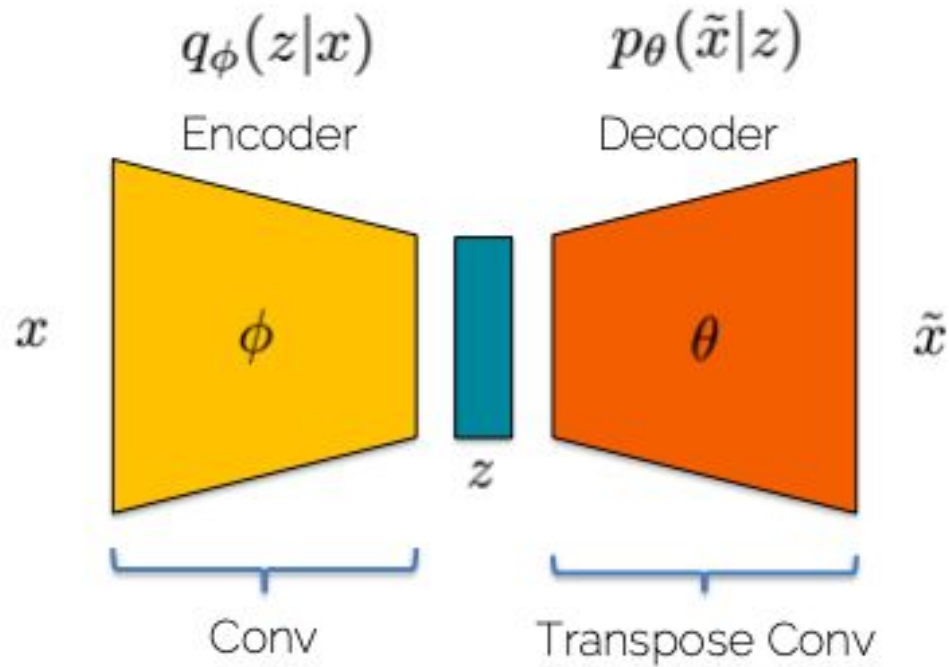
---

Отображаем вход в эмбеддинг, реконструируем декодером



# Variational autoencoder

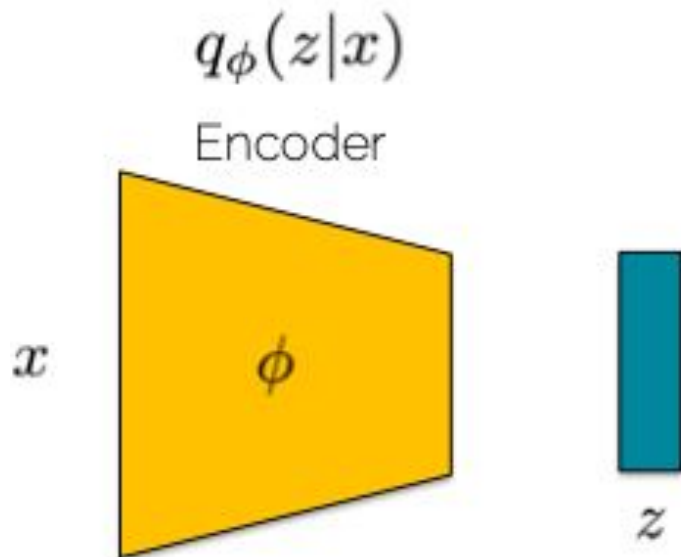
---



# Variational autoencoder

---

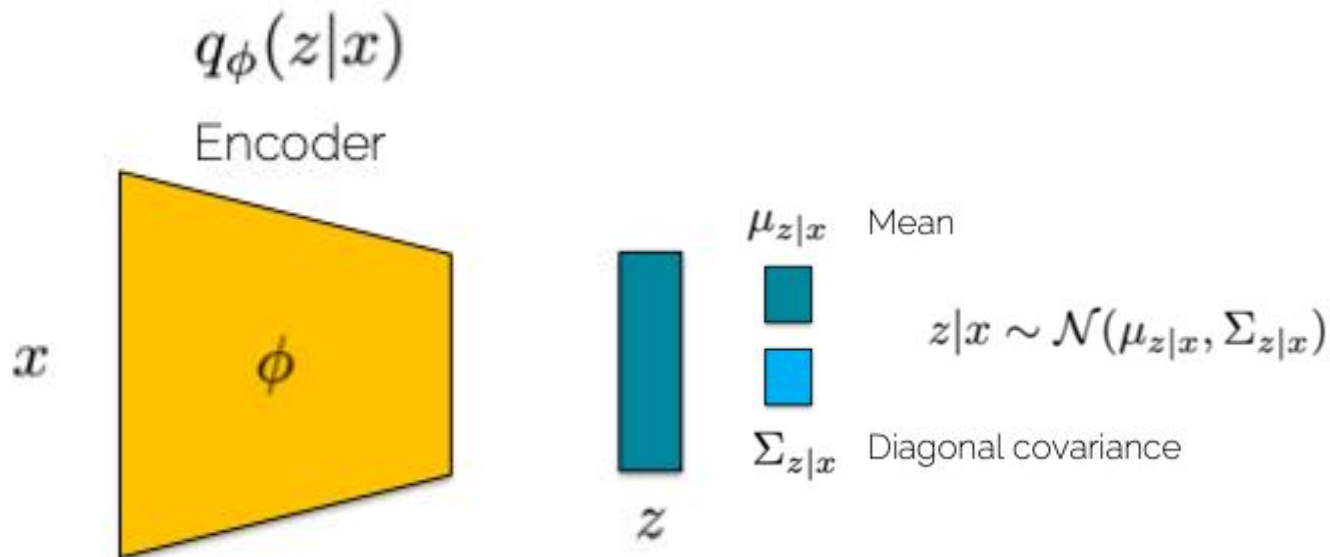
- Латентное пространство - распределения
- Обычно используются Гауссовские



# Variational autoencoder

---

- Латентное пространство - распределения
- Обычно используются Гауссовские



# Variational autoencoder

---

- С байесовской точки зрения, процесс генерации:

$$p_{\theta}(x) = \int_z p_{\theta}(x|z)p_{\theta}(z)dz$$

- Знаменатель постериорного распределения:

$$p_{\theta}(z|x) = \frac{p_{\theta}(x|z)p_{\theta}(z)}{p_{\theta}(x)}$$


# Variational autoencoder

---

- Функция потерь для изображения  $x_i$

$$\log(p_{\theta}(x_i)) = \mathbf{E}_{z \sim q_{\phi}(z|x_i)}[\log(p_{\theta}(x_i))]$$

I draw  
samples of  
the latent  
variable  $z$   
from my  
encoder




# Variational autoencoder

---

- Функция потерь для изображения  $x_i$

$$\begin{aligned}\log(p_\theta(x_i)) &= \mathbf{E}_{z \sim q_\phi(z|x_i)} [\log(p_\theta(x_i))] \\ &= \mathbf{E}_{z \sim q_\phi(z|x_i)} \left[ \log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \right] \quad \text{Bayes Rule}\end{aligned}$$

  
Posterior



## Variational autoencoder

---

- Функция потерь для изображения  $x_i$

$$\begin{aligned}\log(p_\theta(x_i)) &= \mathbf{E}_{z \sim q_\phi(z|x_i)} [\log(p_\theta(x_i))] \\ &= \mathbf{E}_{z \sim q_\phi(z|x_i)} \left[ \log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \right] \\ &= \mathbf{E}_z \left[ \log \frac{p_\theta(x_i|z)p_\theta(z)}{p_\theta(z|x_i)} \frac{q_\phi(z|x_i)}{q_\phi(z|x_i)} \right]\end{aligned}$$

# Variational autoencoder

---

- Функция потерь для изображения  $x_i$

$$\log(p_\theta(x_i)) = \mathbf{E}_z \left[ \log \frac{p_\theta(x_i|z)p_\theta(z)q_\phi(z|x_i)}{p_\theta(z|x_i)q_\phi(z|x_i)} \right]$$

$$= \mathbf{E}_z [\log p_\theta(x_i|z)] - \mathbf{E}_z \left[ \log \frac{q_\phi(z|x_i)}{p_\theta(z)} \right] + \mathbf{E}_z \left[ \log \frac{q_\phi(z|x_i)}{p_\theta(z|x_i)} \right]$$

# Variational autoencoder

---

- Функция потерь для изображения  $x_i$

$$\begin{aligned} &= \boxed{\mathbf{E}_z [\log p_\theta(x_i|z)]} - \boxed{\mathbf{E}_z \left[ \log \frac{q_\phi(z|x_i)}{p_\theta(z)} \right]} + \boxed{\mathbf{E}_z \left[ \log \frac{q_\phi(z|x_i)}{p_\theta(z|x_i)} \right]} \\ &= \boxed{\mathbf{E}_z [\log p_\theta(x_i|z)]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))} \end{aligned}$$

# Variational autoencoder


---

- Функция потерь для изображения  $x_i$


$$= \boxed{E_z [\log p_\theta(x_i|z)]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$



Reconstruction loss



Измеряет насколько близко мое латентное распределение к  
приорному распределению



Непонятно, что такое. но  
известно что  $\geq 0$

# Variational autoencoder

---

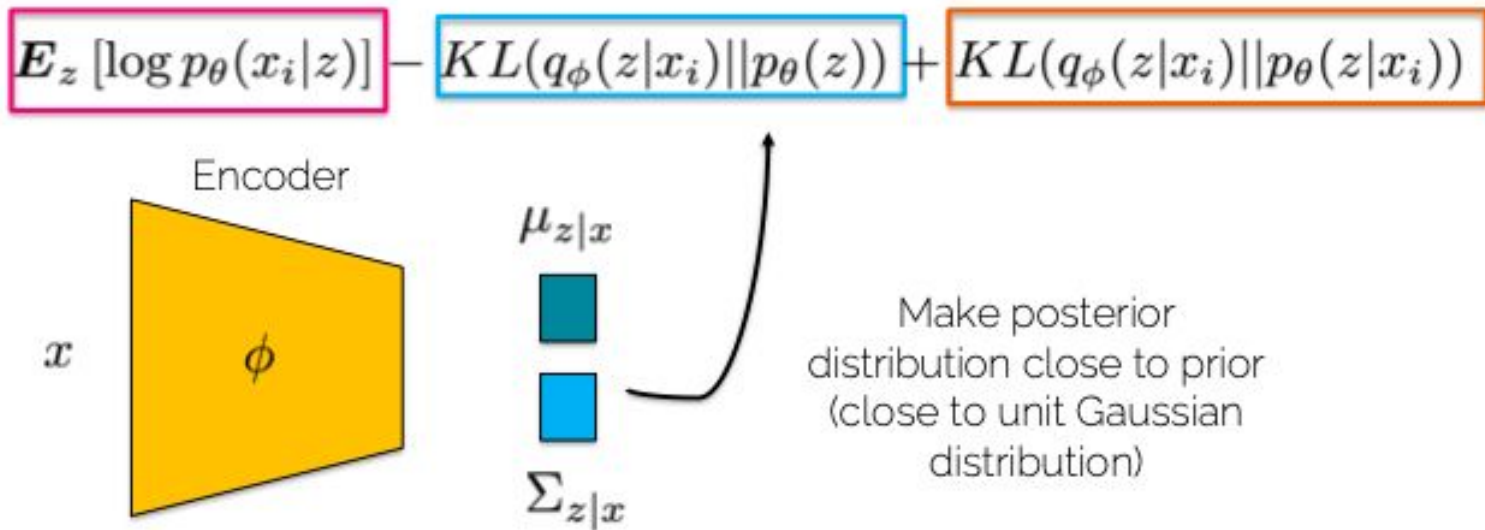
- Функция потерь для изображения  $x_i$

$$\underbrace{E_z [\log p_\theta(x_i|z)] - KL(q_\phi(z|x_i)||p_\theta(z))}_{\text{Loss function (lower bound)}} + \underbrace{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}_{\geq 0}$$
$$\mathcal{L}(x_i, \phi, \theta)$$

$$\phi^*, \theta^* = \arg \max \sum_{i=1}^N \mathcal{L}(x_i, \phi, \theta)$$

# Variational autoencoder

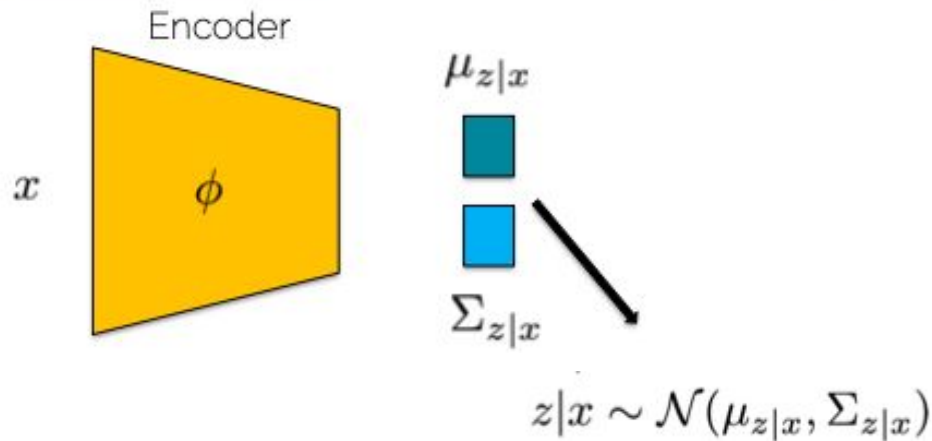
- Обучение



# Variational autoencoder

- Обучение

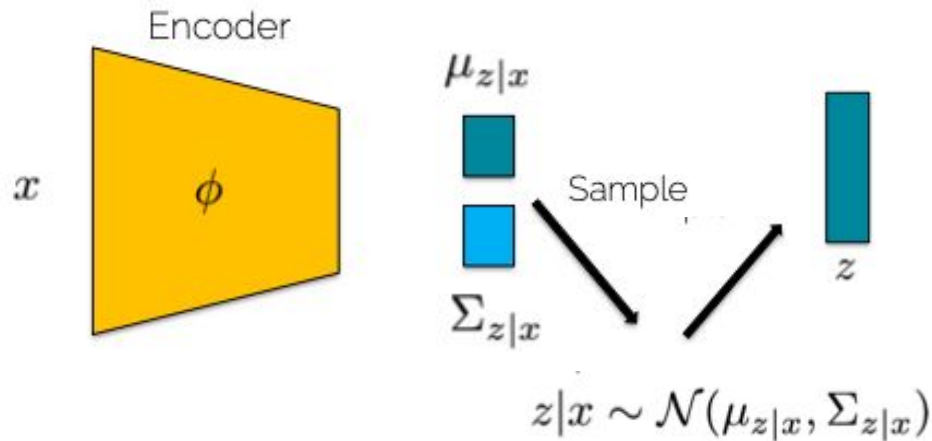
$$E_z [\log p_\theta(x_i|z)] - KL(q_\phi(z|x_i)||p_\theta(z)) + KL(q_\phi(z|x_i)||p_\theta(z|x_i))$$



# Variational autoencoder

- Обучение

$$E_z [\log p_\theta(x_i|z)] - KL(q_\phi(z|x_i)||p_\theta(z)) + KL(q_\phi(z|x_i)||p_\theta(z|x_i))$$

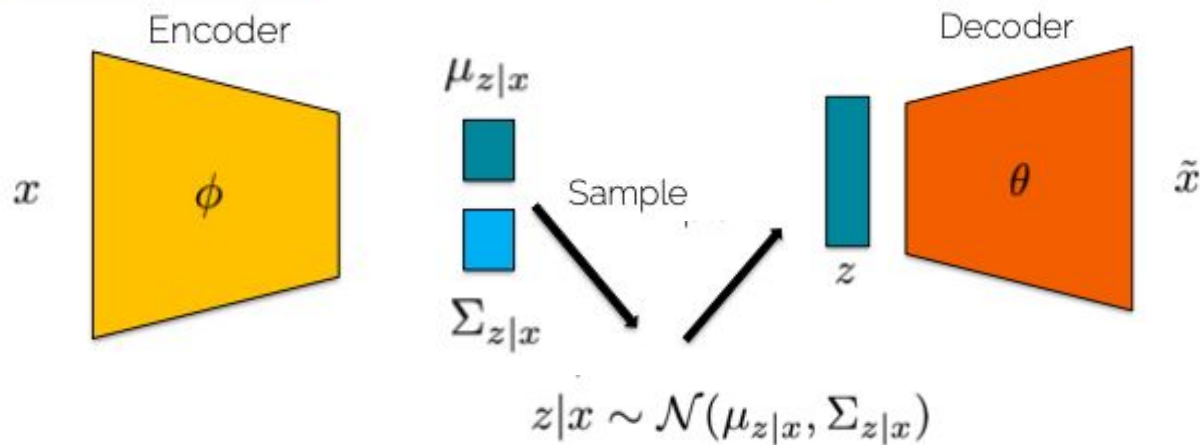




# Variational autoencoder

- Обучение

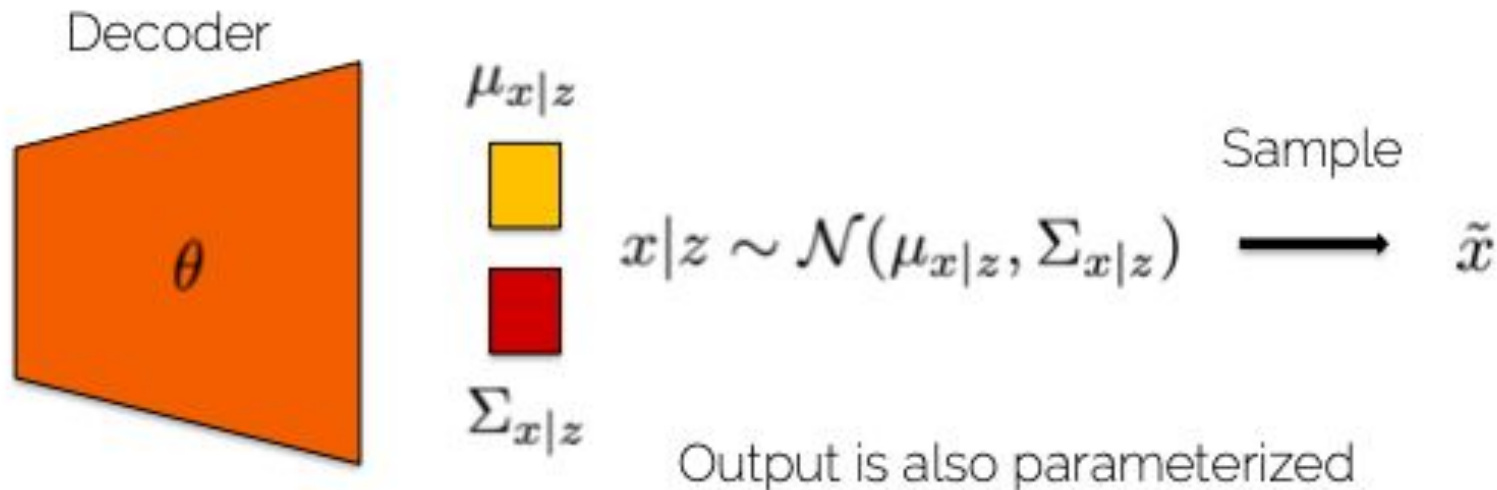
$$E_z [\log p_\theta(x_i|z)] - KL(q_\phi(z|x_i)||p_\theta(z)) + KL(q_\phi(z|x_i)||p_\theta(z|x_i))$$



# Variational autoencoder

- Обучение

$$\boxed{E_z [\log p_\theta(x_i|z)]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$



# Variational autoencoder

---

- Обучение

$$\boxed{E_z [\log p_\theta(x_i|z)]} - \boxed{KL(q_\phi(z|x_i)||p_\theta(z))} + \boxed{KL(q_\phi(z|x_i)||p_\theta(z|x_i))}$$



Maximize the likelihood of  
reconstructing the input

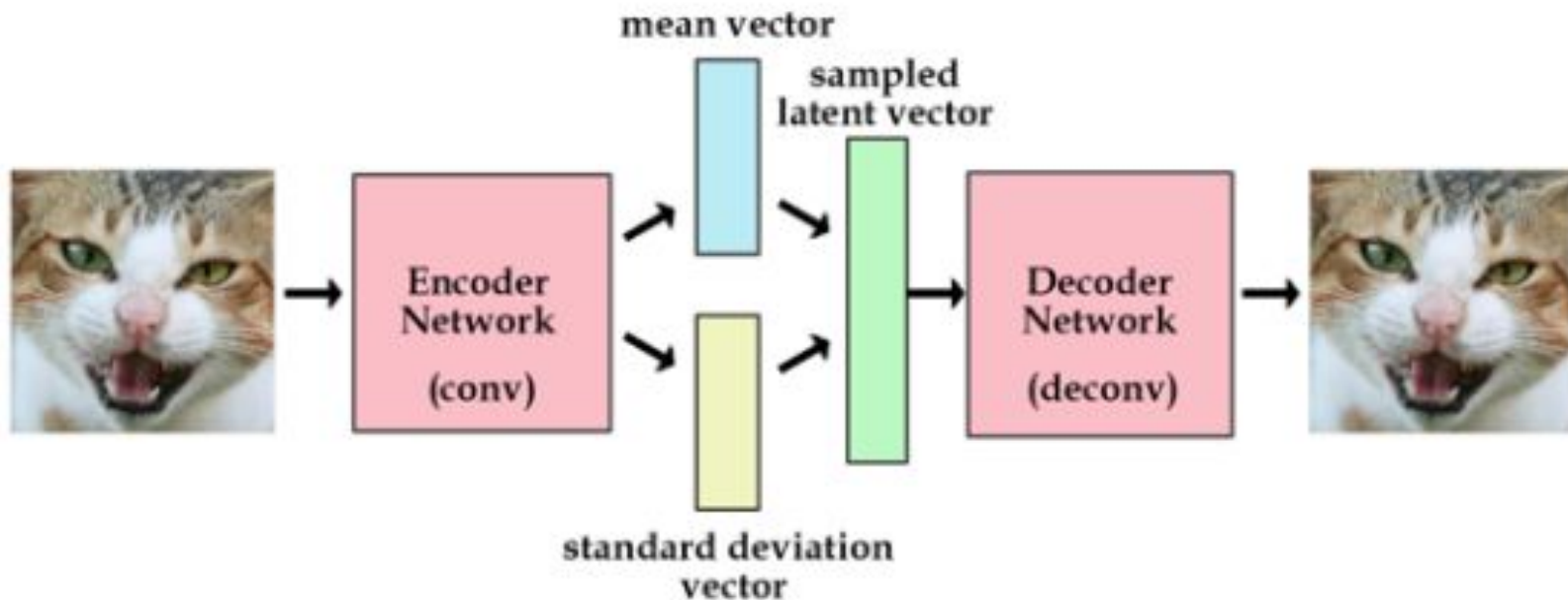
## Variational autoencoder

---

- Reparametrization trick позволяет легко делать бэкпроп
- Больше деталей и математических выкладок тут  
Kingman and Welling. "Auto-Encoding Variational Bayes". ICLR 2014

# Генерация данных

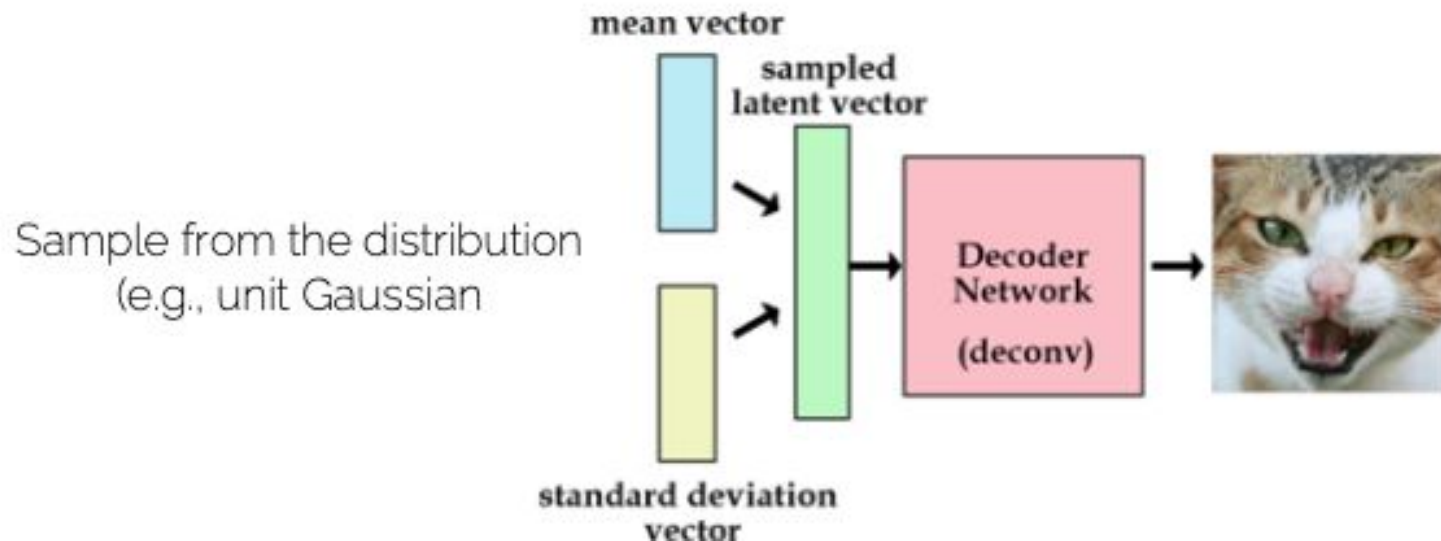
---



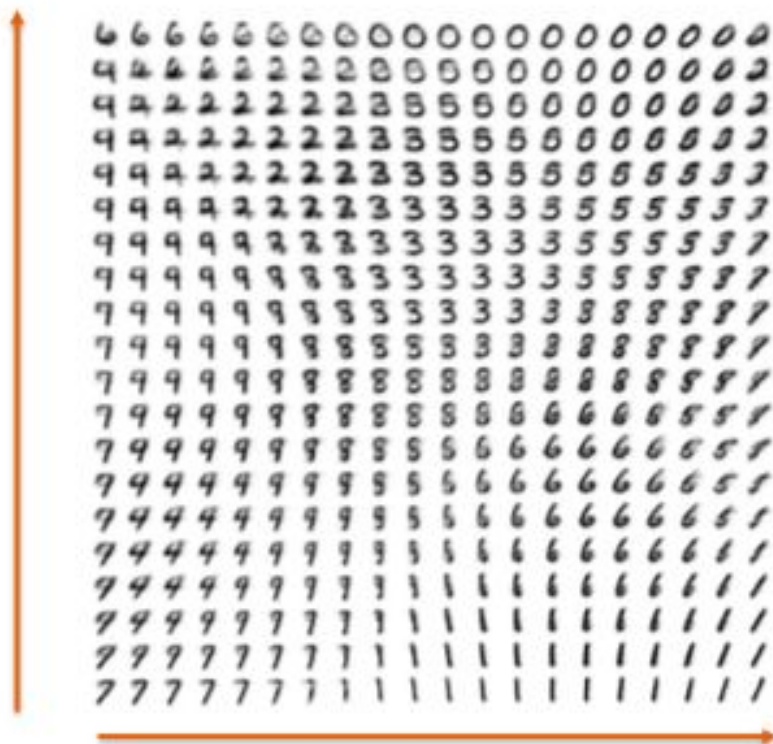
# Генерация данных

---

- После обучения



# Генерация данных



Each element of  $z$   
encodes a different  
feature

# Генерация данных

---

Degree of smile



Head pose



# Autoencoder vs VAE

---



Autoencoder



Variational Autoencoder



Ground Truth

# Резюме

---

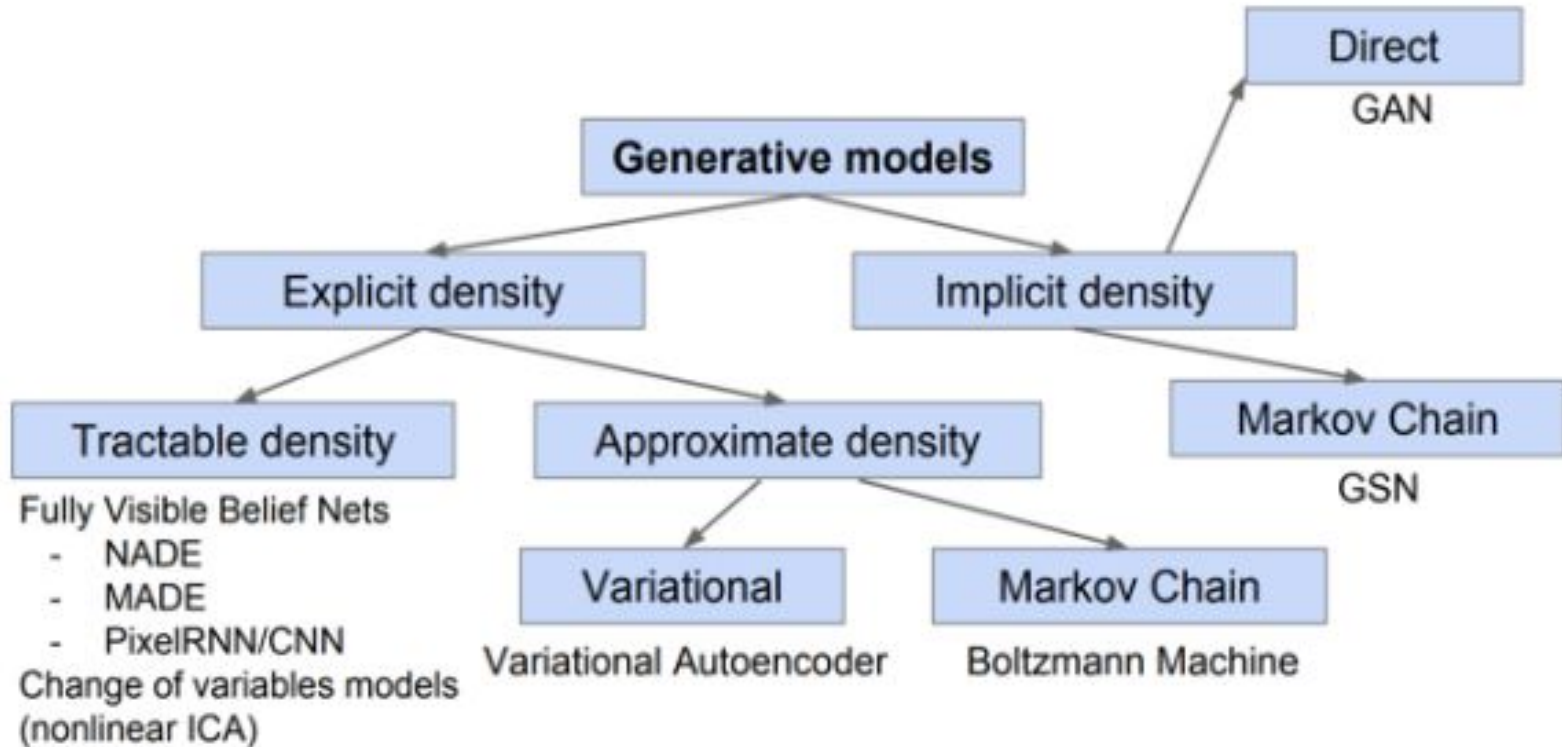
- Autoencoder (AE)
  - Реконструировать инпут
  - Обучение без разметки
  - Фичи из латентного пространства очень полезны
- Variational Autoencoders (VAE)
  - Вероятностное распределение в латентном пространстве (напр. Гауссовское)
  - Сэмплируем из модели, чтобы сгенерировать аутпут



# Generative models

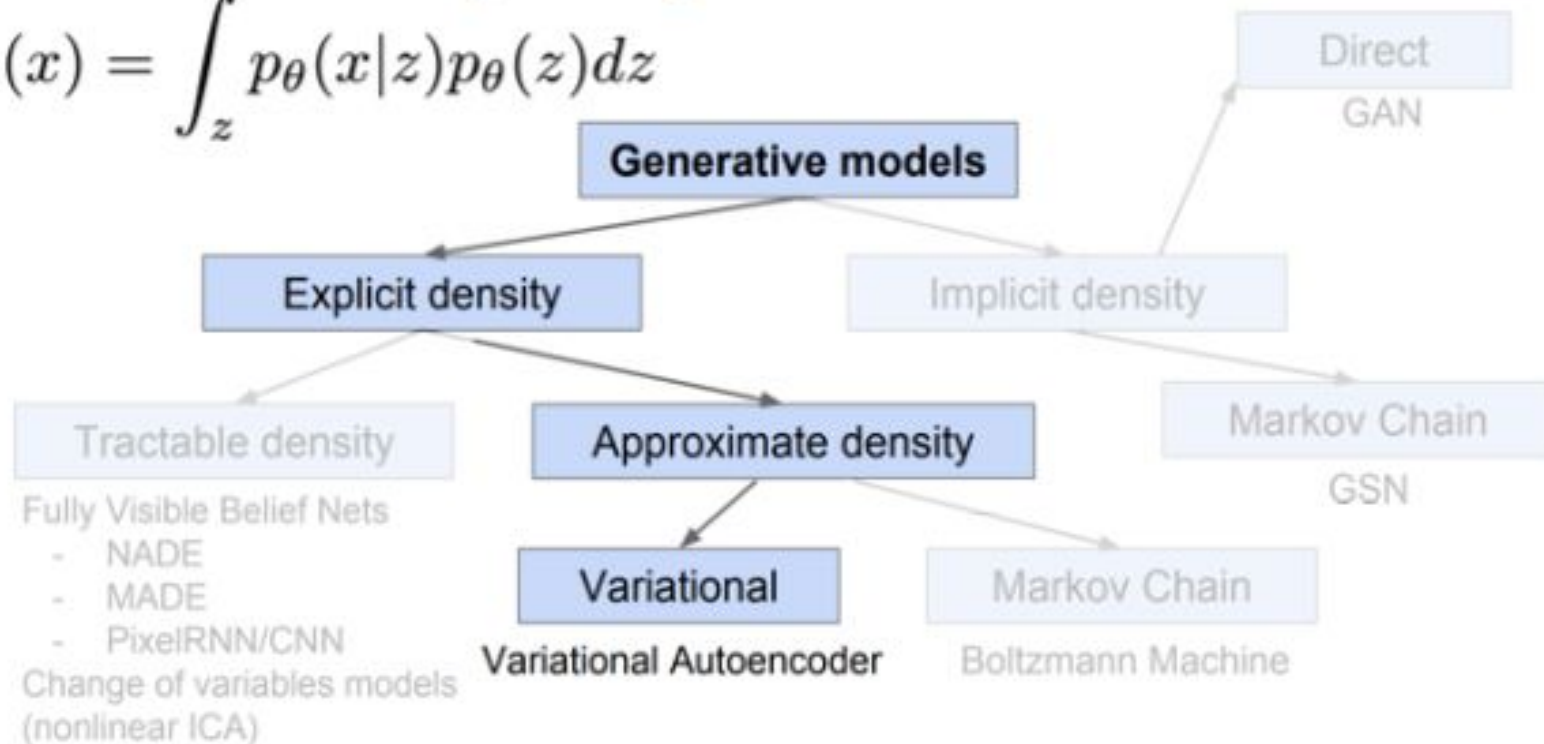
# Generative models

---

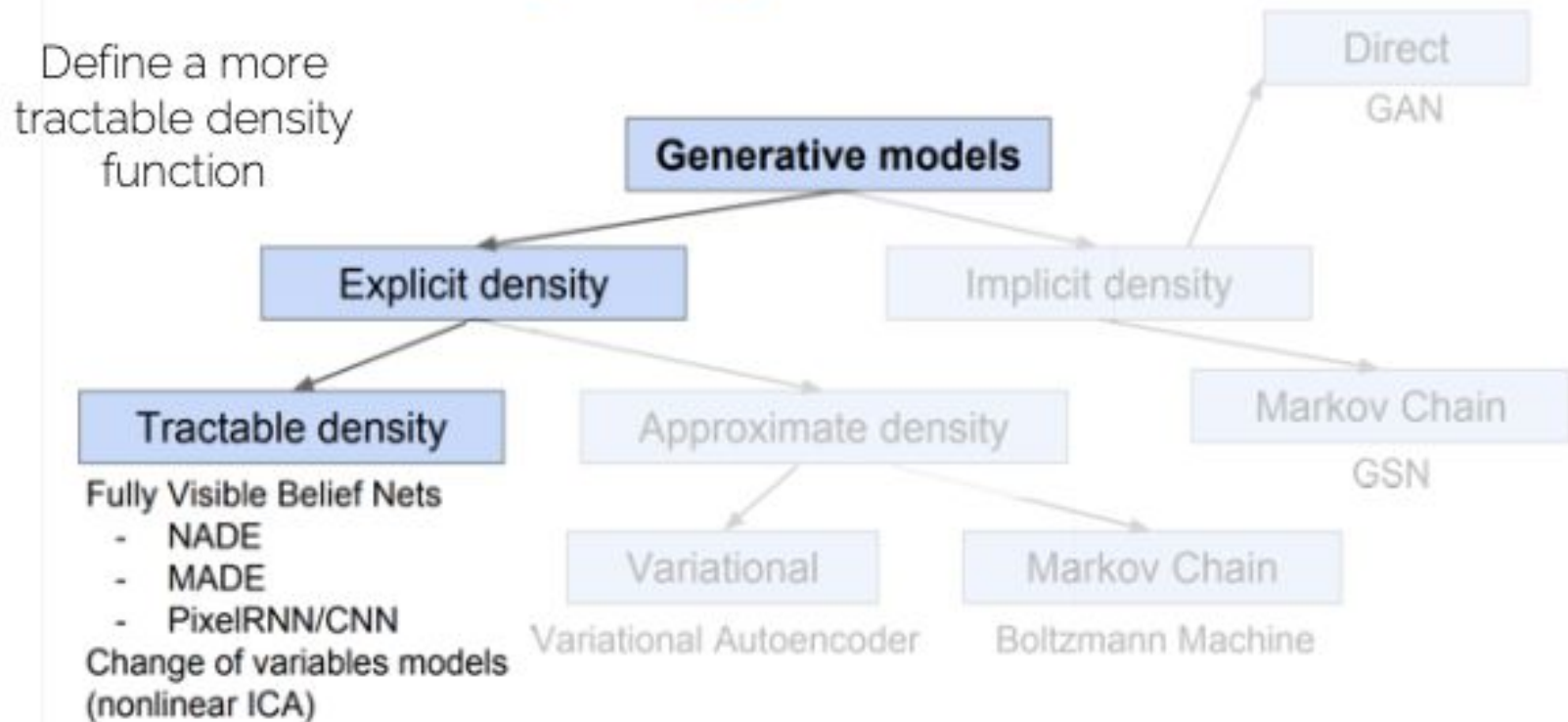


# Generative models

$$p_{\theta}(x) = \int_z p_{\theta}(x|z)p_{\theta}(z)dz$$

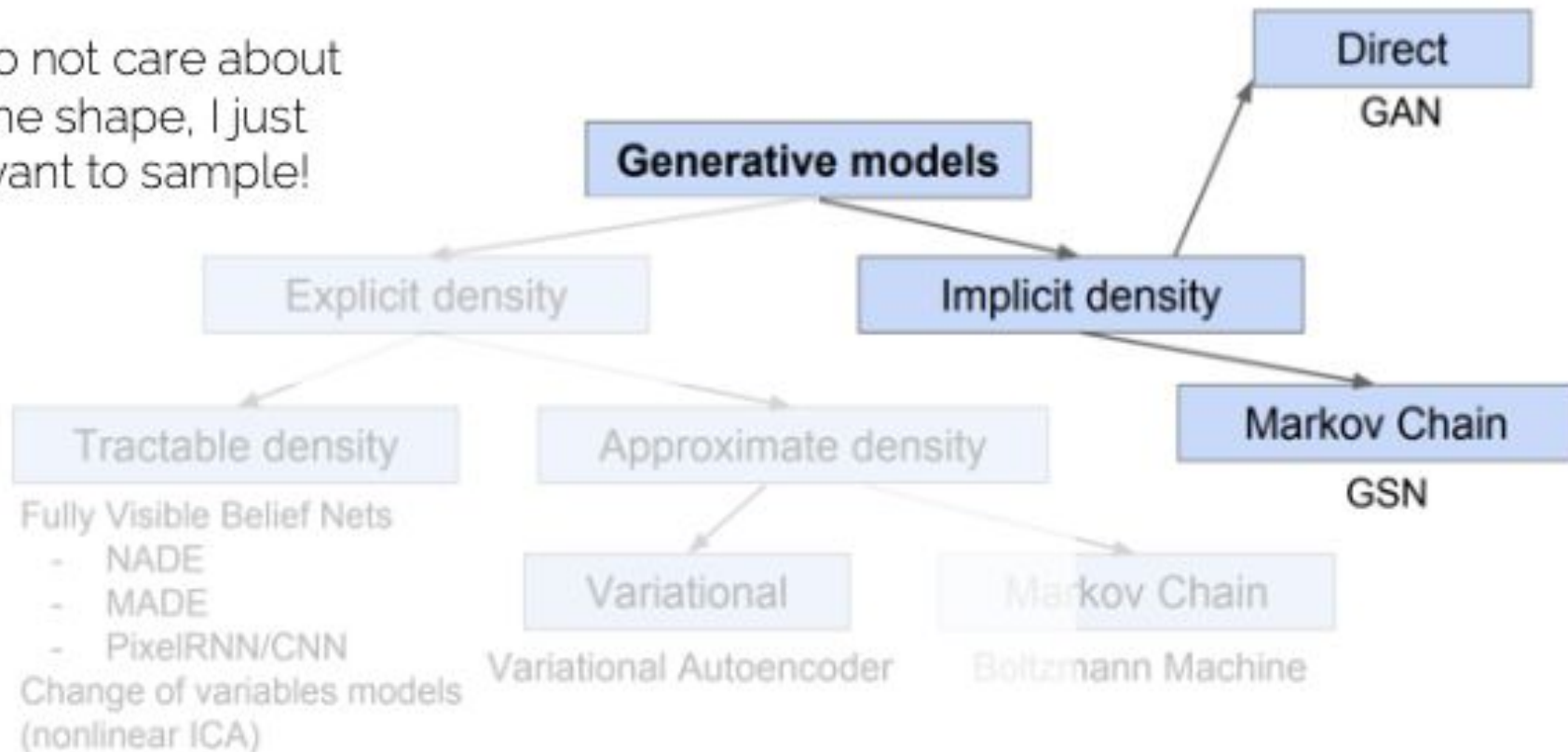


# Generative models



# Generative models

I do not care about  
the shape, I just  
want to sample!



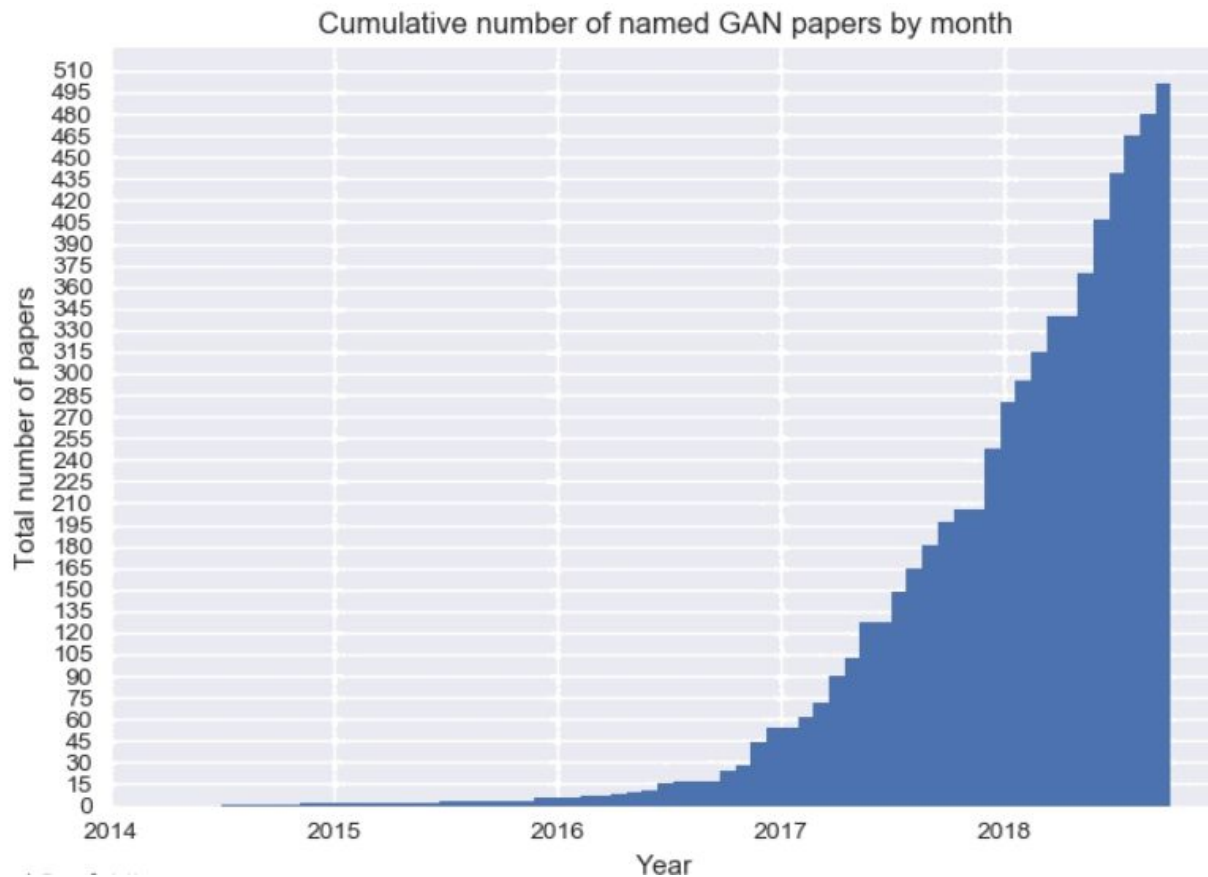


GANs



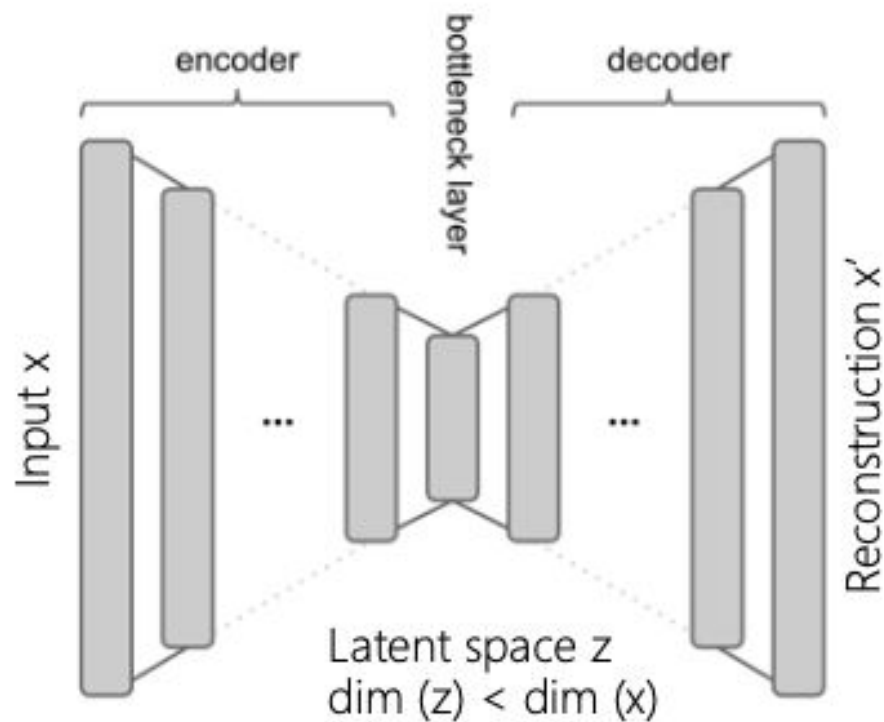
# GANs

---

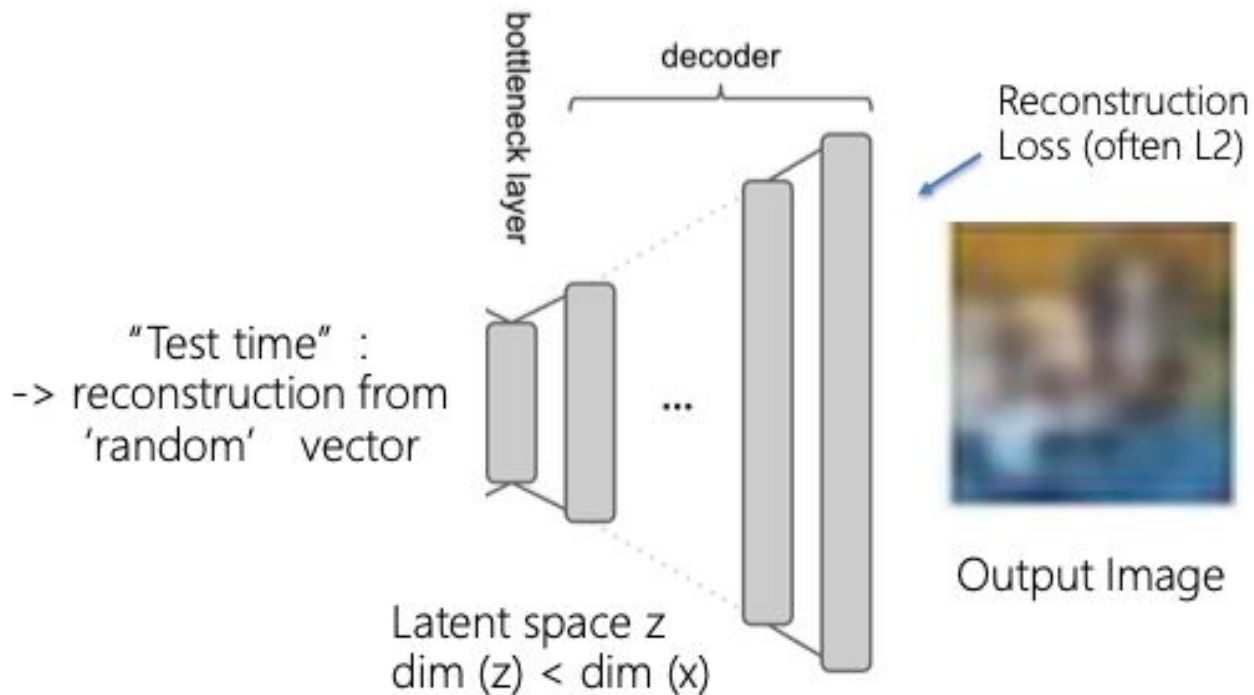


<https://github.com/hindupuravinash/the-gan-zoo>

# GANs

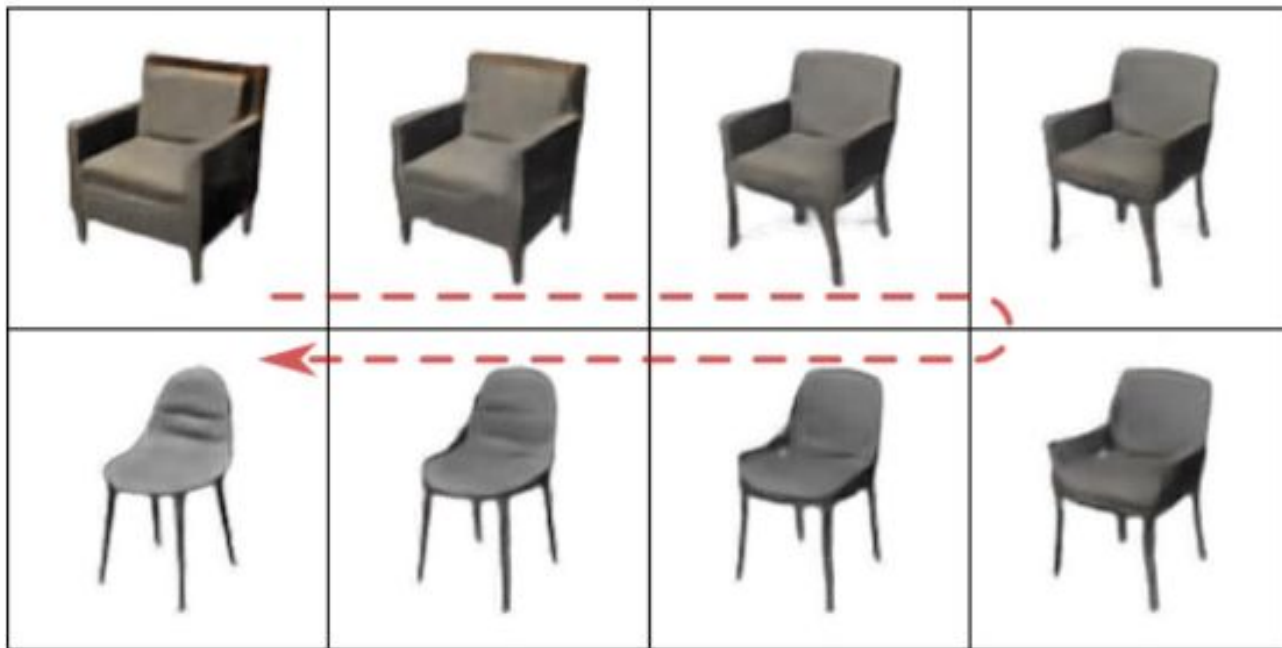


# Decoder для генерации



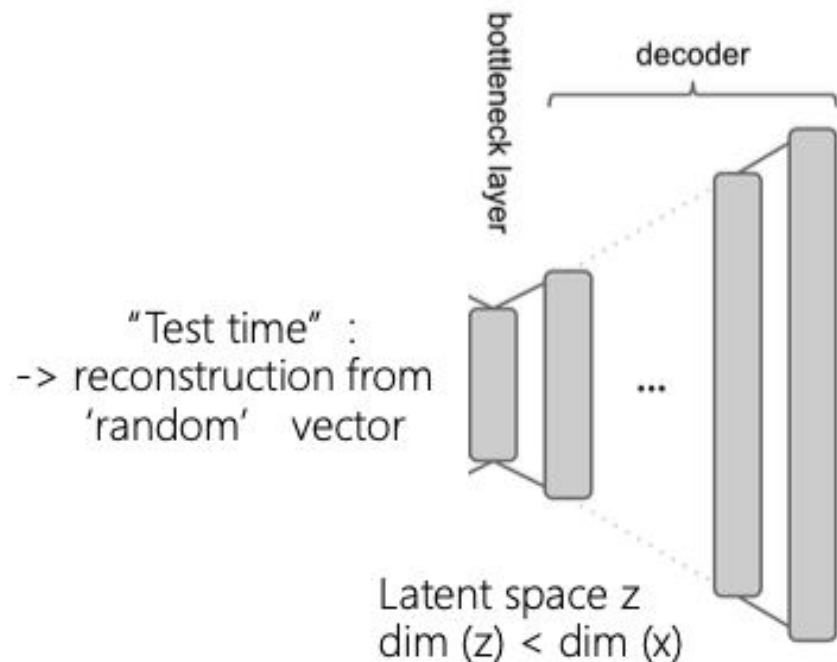
# Decoder для генерации

---



Interpolation between two chair models

# Decoder для генерации



Reconstruction Loss  
Often L2, i.e., sum of squared dist.  
-> L2 distributes error equally

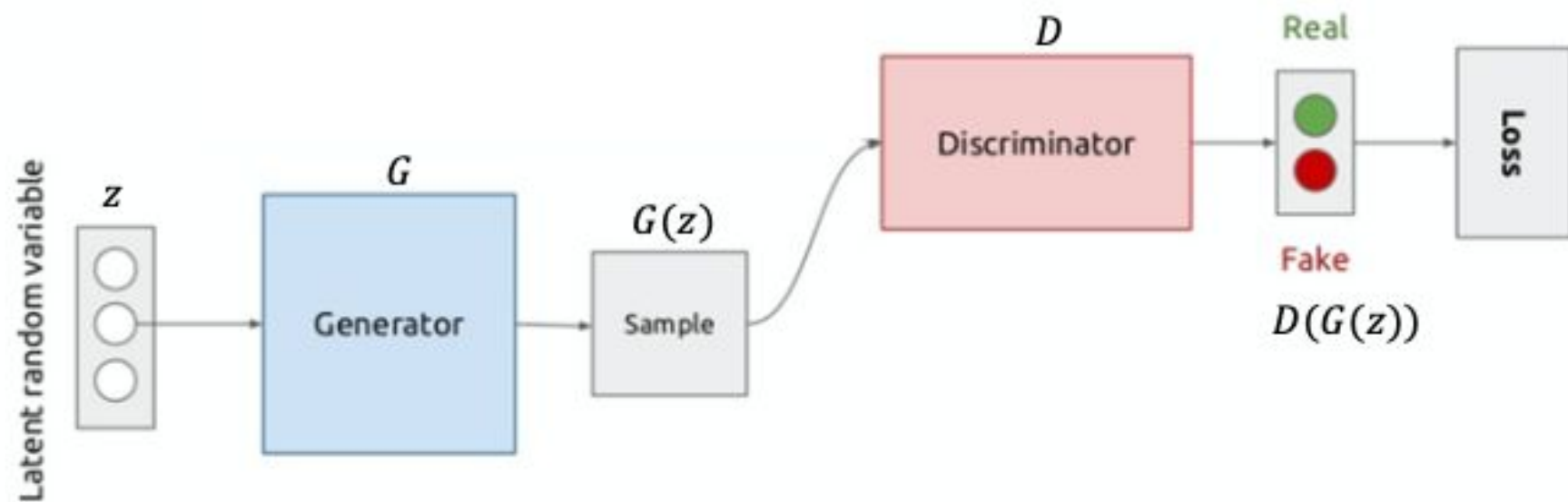


-> mean is opt.  
-> res. is blurry

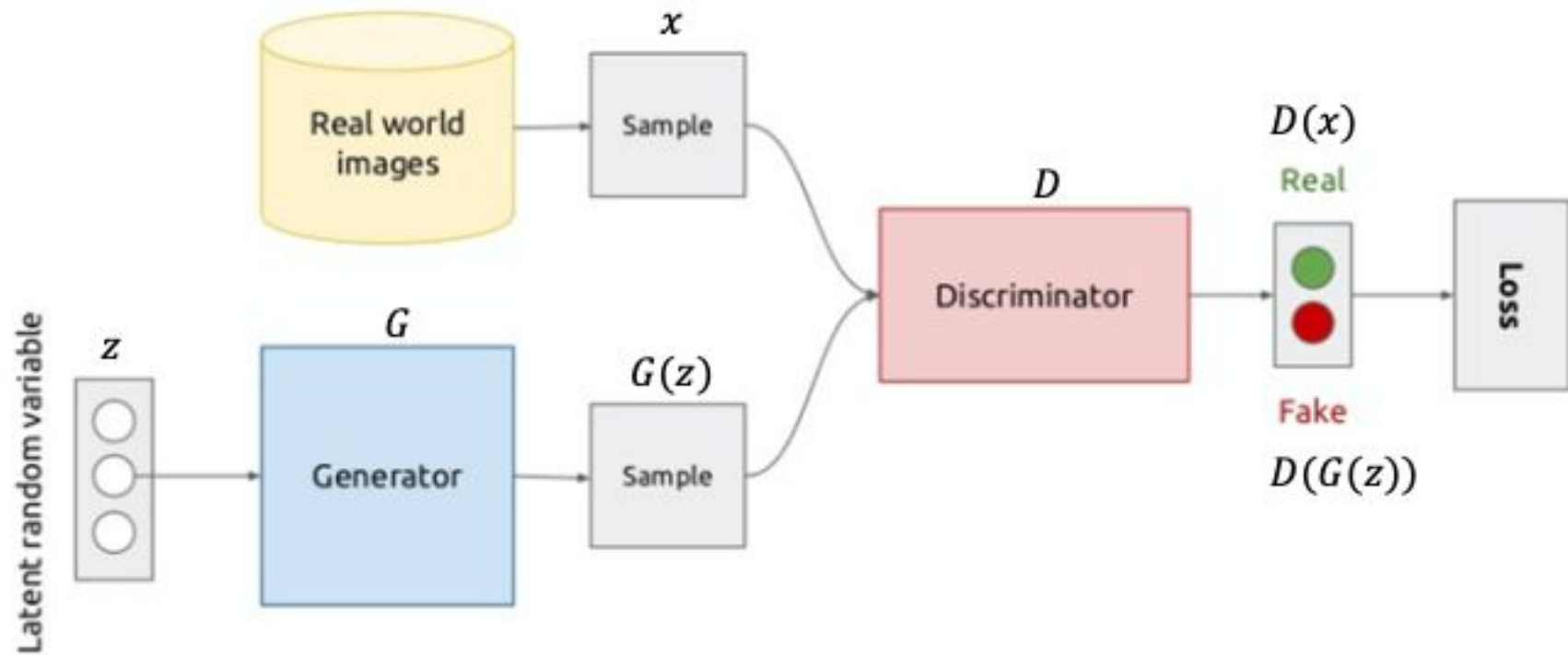
Instead of L2, can we  
"learn" a loss function?

# GANs

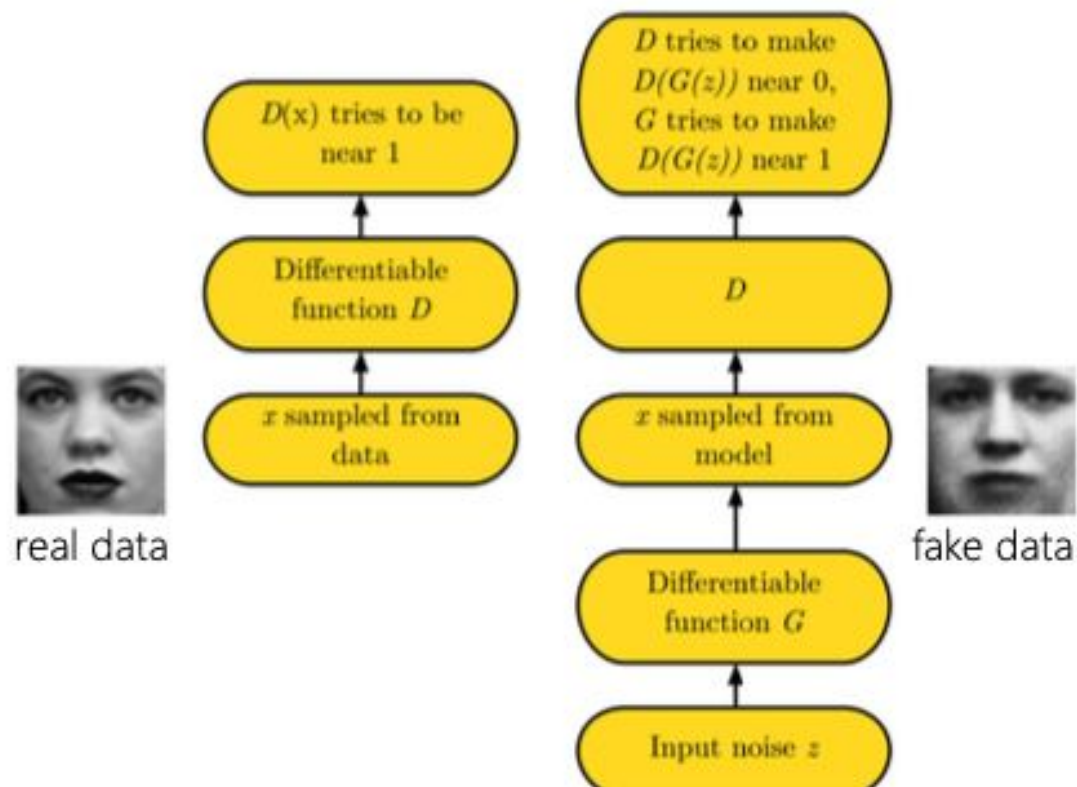
---



# GANs



# GANs





# GANs: функции потерь

---

Discriminator loss

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

Generator loss

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

binary cross entropy

Минимаксная игра:

- G минимизирует вероятность того, что D корректен
- Равновесие достигается в седловой точке дискриминатор лосса
- D супервайзит G (дает градиенты)

# GANs: функции потерь

---

Discriminator loss

$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

Generator loss

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

Heuristic loss:

- G максимизирует вероятность того, что D ошибется
- G все равно может обучаться, даже когда D отвергает все сгенерированные примеры

## Поочередные градиенты

---

Шаг 1. Фиксируем  $G$ , делаем градиентный шаг:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

Шаг 2: Фиксируем  $D$ , делаем градиентный шаг:

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

# Vanilla GAN

---

**for** number of training iterations **do**

**for**  $k$  steps **do**

- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Sample minibatch of  $m$  examples  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  from data generating distribution  $p_{\text{data}}(\mathbf{x})$ .
- Update the discriminator by ascending its stochastic gradient:

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

**end for**

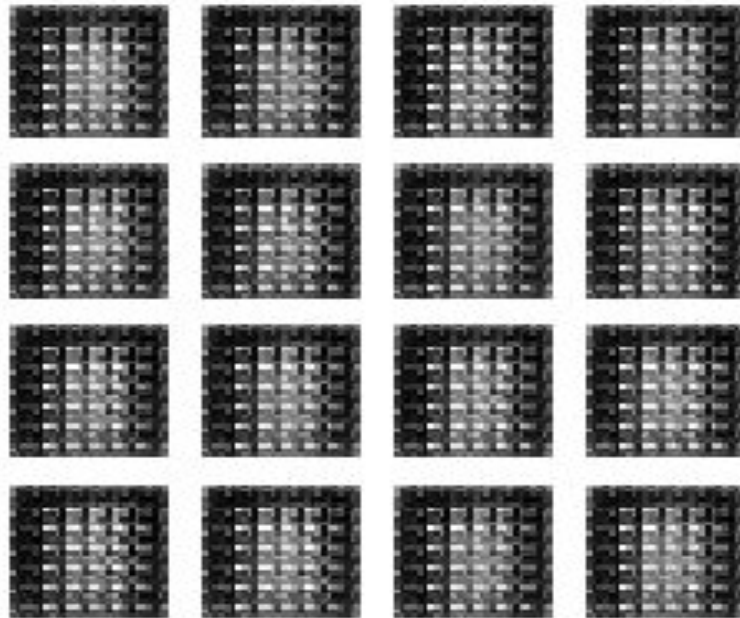
- Sample minibatch of  $m$  noise samples  $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$  from noise prior  $p_g(\mathbf{z})$ .
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log (1 - D(G(\mathbf{z}^{(i)}))).$$

**end for**

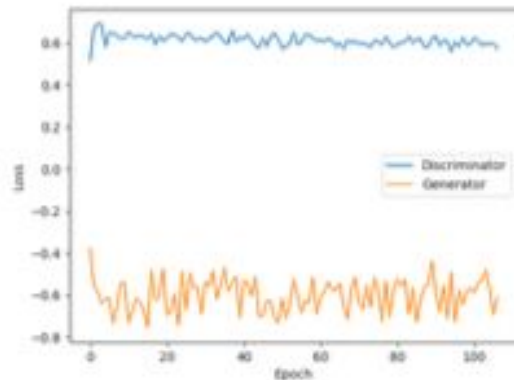
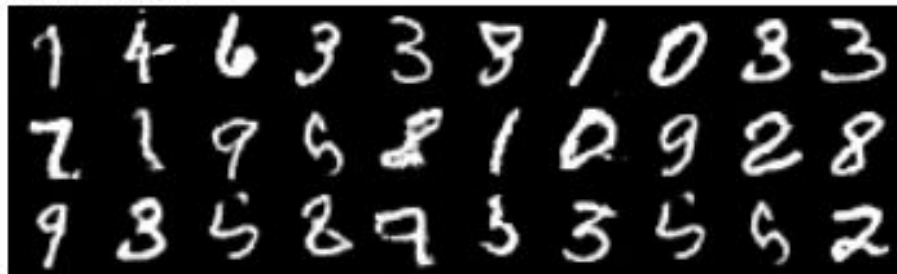
# GANs

---

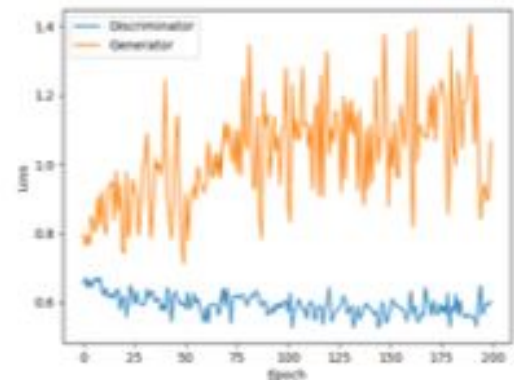
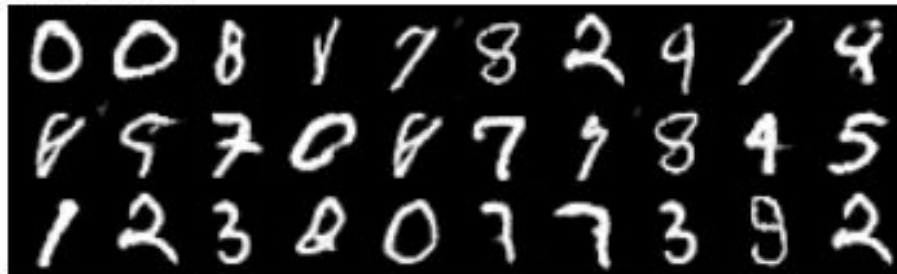


# GANs

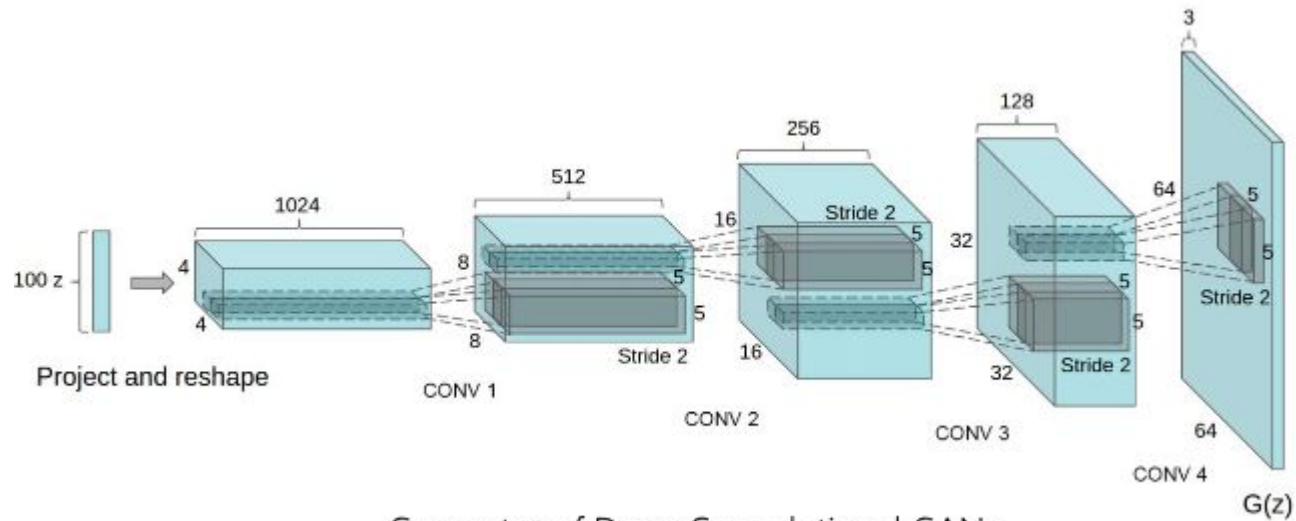
Minimax



Heuristic



# DCGAN: Generator



Generator of Deep Convolutional GANs

# DCGAN: Results

---



Results on MNIST



# DCGAN: Results

---



Results on CelebA (200k relatively well aligned portrait photos)

DCGAN: <https://github.com/carpedm20/DCGAN-tensorflow>

# DCGAN: Results

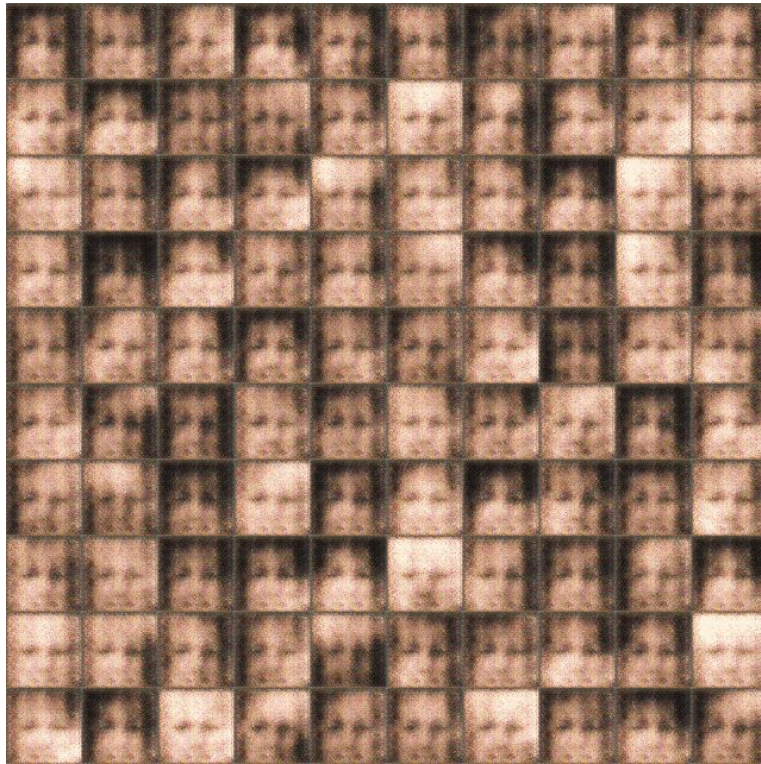
---



Asian face dataset

# DCGAN: Results

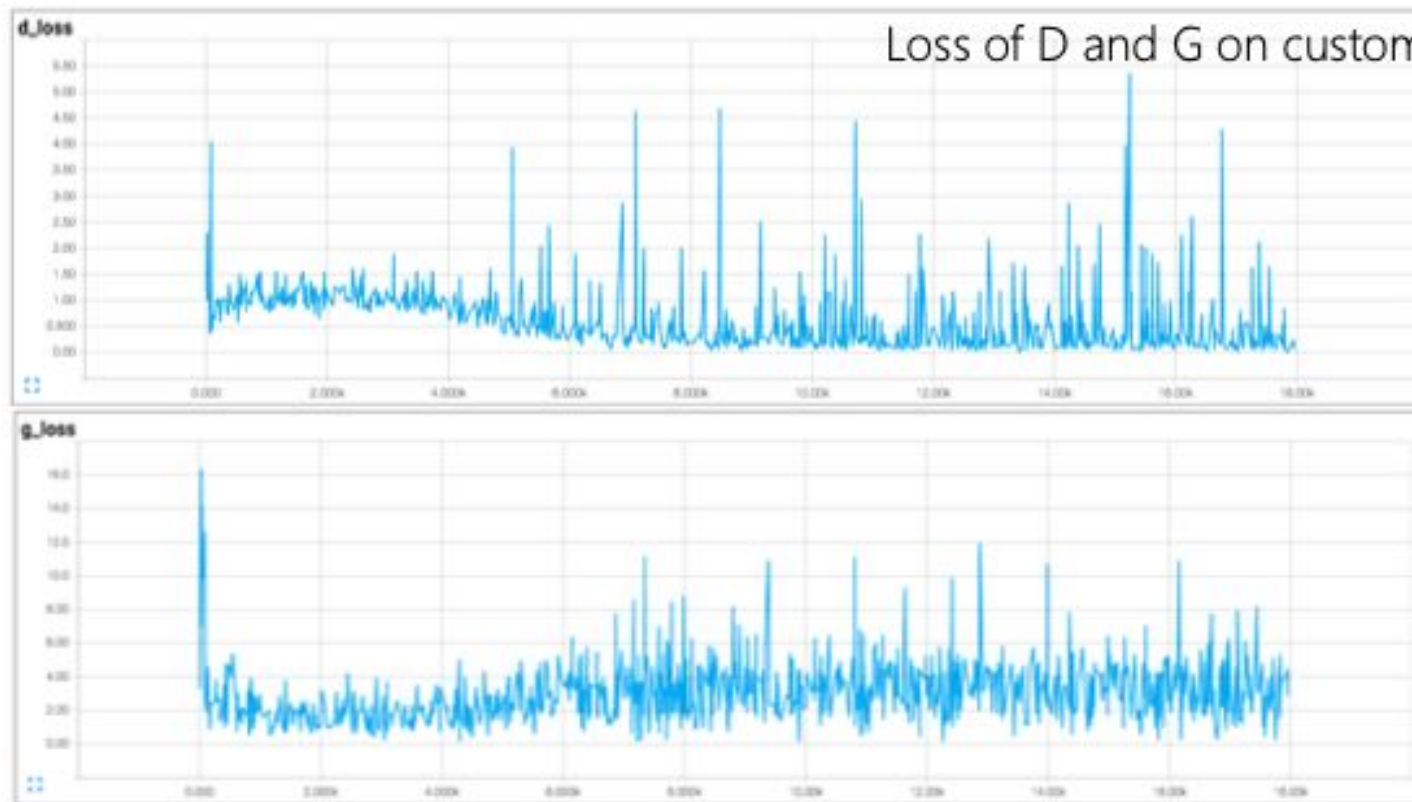
---





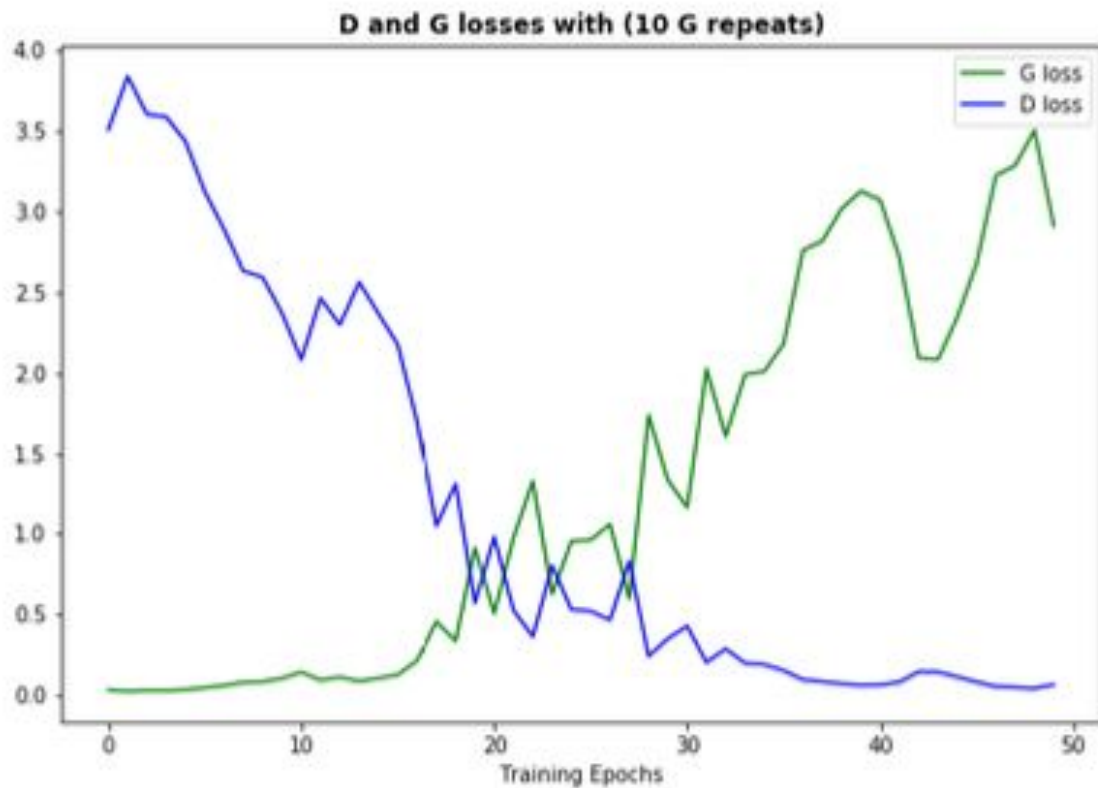
# DCGAN: Results

---



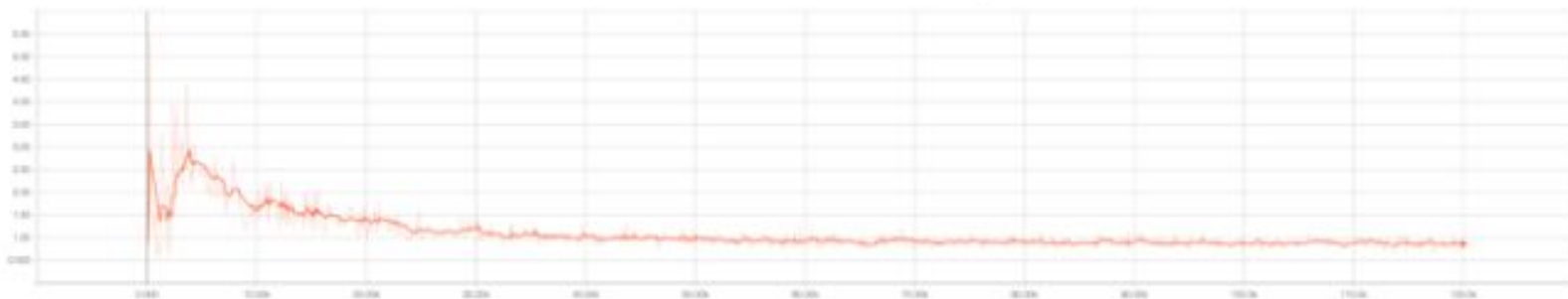
# Плохие кривые обучения

---



# Хорошие кривые обучения

---



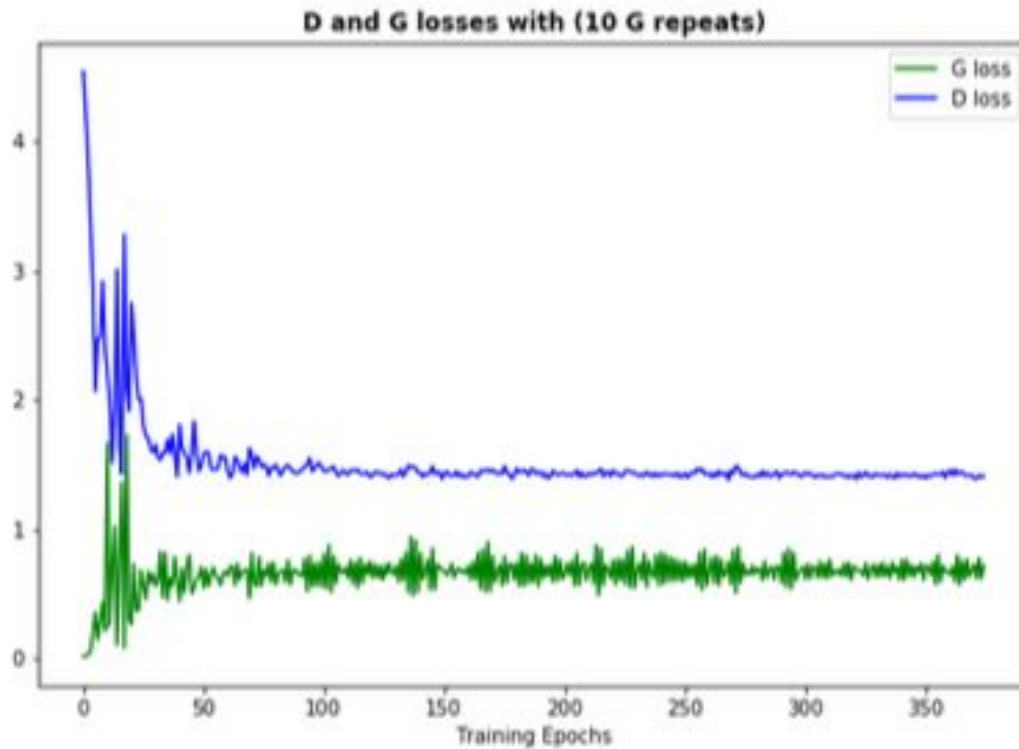
Generator's Error through Time



Discriminator's Error through Time

# Хорошие кривые обучения

---



# Training Schedule

---

## Адаптивное расписание

Например:

```
while loss_discriminator > t_d:  
    train discriminator  
while loss_generator > t_g:  
    train generator
```



# Weak vs Strong Discriminator

---

Нужен баланс

- Слишком слабый дискриминатор?
  - Нету хороших градиентов (нельзя стать лучше учителя...)
- Слишком слабый генератор?
  - Дискриминатор всегда будет прав

# Mode collapse

---

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

D во внутреннем цикле -> сходимость к корректному распределению

G во внутреннем цикле -> легко сойтись к одному примеру

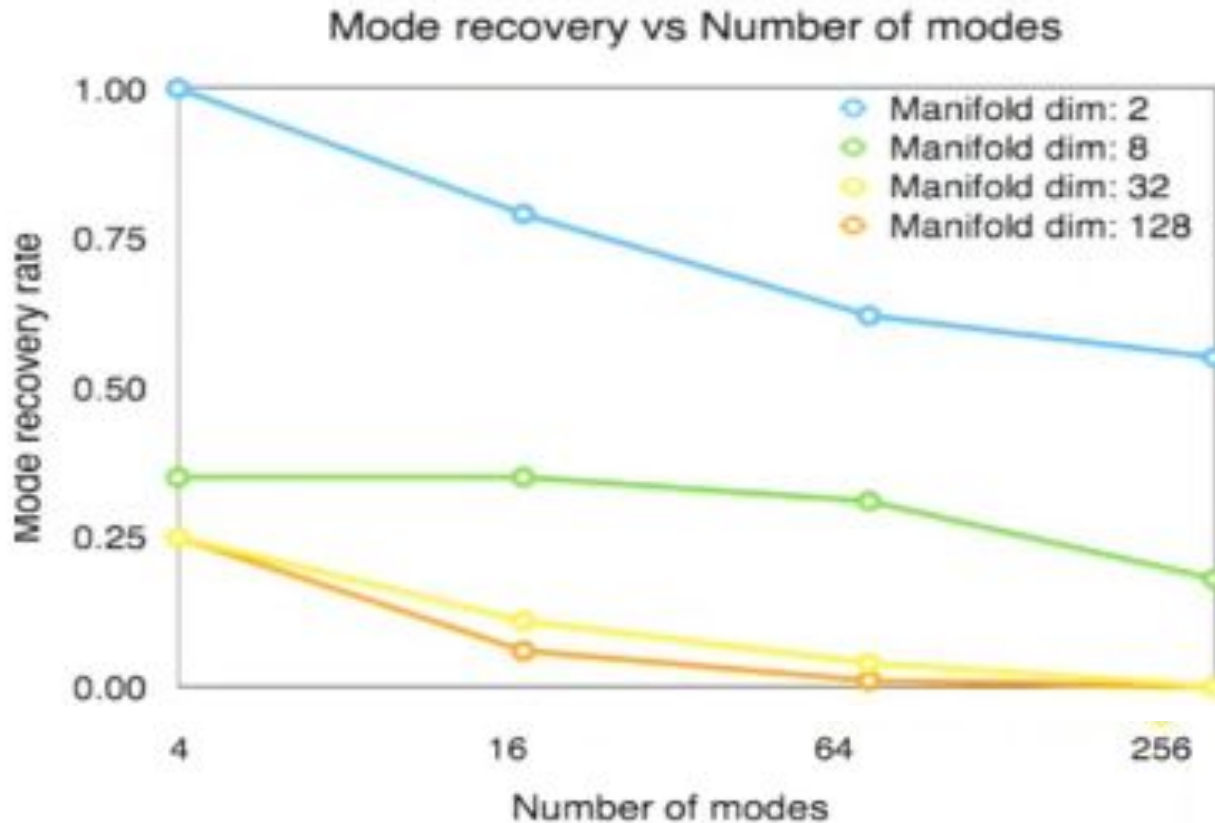


# Mode collapse

- Фиксированная размерность (512)
- Качество коррелирует с количеством mode

Чем больше modes, тем меньше процент восстановления

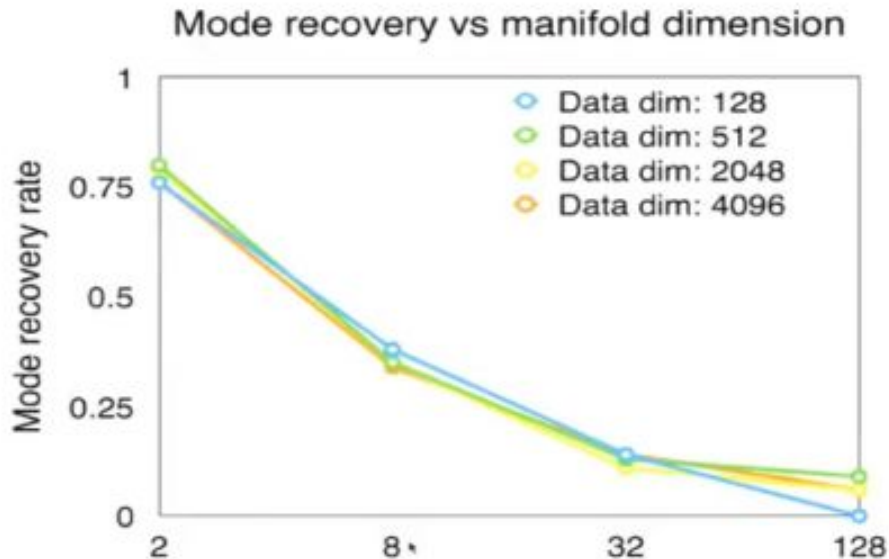
Одна из причин, почему мы используем GANs на определенных доменах



# Mode collapse

---

- Качество коррелирует с размерностью латентного пространства



-> Чем больше латентное пространство, тем больше mode коллапсов

# Problems with Global Structure

---



# Problems with Counting

---



(Goodfellow 2016)

# Evaluation of GAN performance

---

# Evaluation of GAN performance

---

- Мы не знаем, насколько хорошо идет генерация
- Ученые выборочно показываются результаты -> показывают только хорошие, как оценить численно?
- Мы запоминанием или мы обобщаем?
- GANы сложно оценивать!



# Evaluation of GAN performance

---

Человеческая оценка:

- Каждые  $n$  апдейтов, посмотреть на серию предсказаний
- Посмотреть кривые обучения
- Что означает “хорошо выглядит” в начале обучения?
  - нужно разнообразность
  - но нету реалистичных предсказания
- Показывать реальным людям (тест Тьюринга)

# Evaluation of GAN performance

---

## INCEPTION SCORE (IS)

- Измеряется разнообразие и качество (saliency)
- Обучаем точные классификатор
- Обучаем GAN
- Смотрим насколько точно классификатор может распознавать сгенерированные изображения
- Делаем некоторые предположения о полученном распределении данных

# Evaluation of GAN performance

---

## INCEPTION SCORE (IS)

- Saliency: чекнуть можно ли классифицировать сгенерированные изображения с высокой точностью (высокий скор только у одного класса)
- Diversity: чекнуть, что мы получаем примеры из всех классов

# Evaluation of GAN performance

---

## INCEPTION SCORE (IS)

- Saliency: чекнуть можно ли классифицировать сгенерированные изображения с высокой точностью (высокий скор только у одного класса)
- Diversity: чекнуть, что мы получаем примеры из всех классов

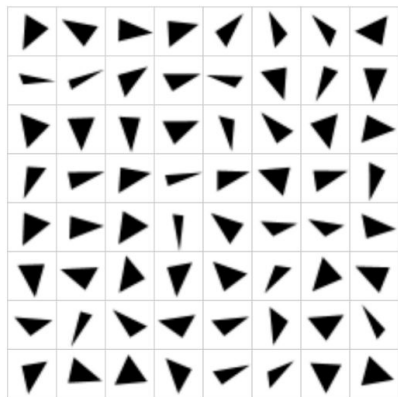
Что если у нас генерить одно хорошее изображение для каждого класса?

# Evaluation of GAN performance

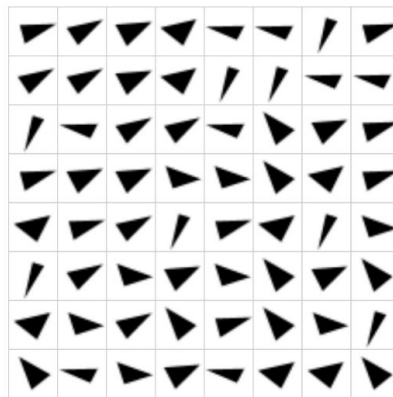
---

Можно посмотреть на дискриминатор:

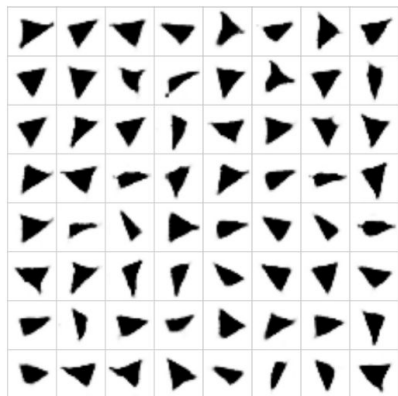
- Если у нас сильный дискриминатор, то и генератор сильный
- Используем фичи D для классификации
- Файнтюним последний
- Если хорошая точность -> у нас хорошие D и G



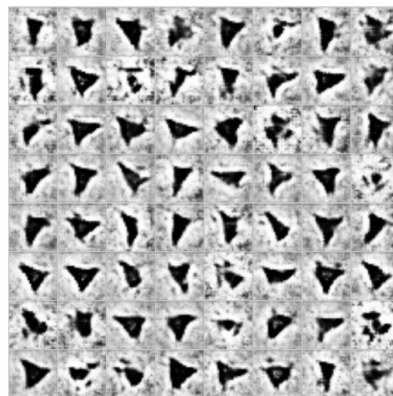
(a) High precision, high recall



(b) High precision, low recall



(c) Low precision, high recall



(d) Low precision, low recall

## Практические советы

---

- Обучение / выбор гиперпараметров (самое важное)
- Выбор функции потерь
- Выбор архитектуры

# Нормализовать инпуты

---

Нормализовать инпуты между -1 и 1

Tanh как последний блок в генератор



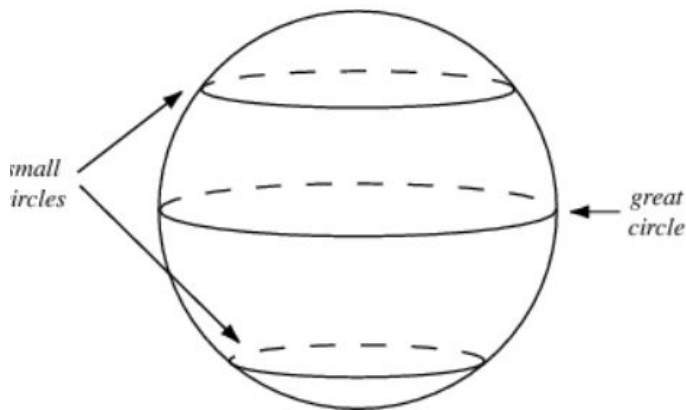
# Sampling

---

Использовать сферический Z

Не семплить из равномерно распределения

Сэмплить из Гауссовского распределения



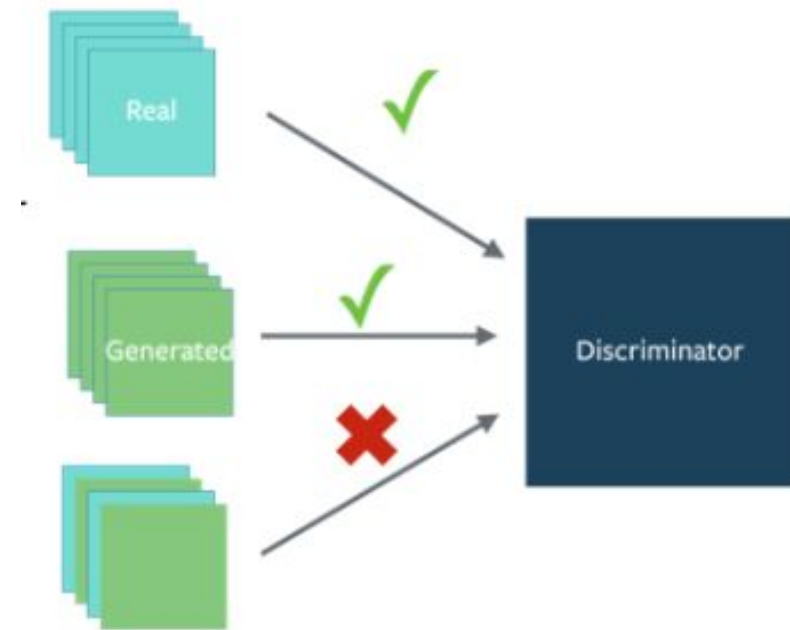
- When doing interpolations, do the interpolation via a great circle, rather than a straight line from point A to point B
- Tom White's [Sampling Generative Networks](#) ref  
code <https://github.com/dribnet/plat>  
has more details

# BatchNorm

---

Использовать BatchNorm

Собирать в минибатчах либо  
только фейковые, либо только  
настоящие картинки



# Adam

---

Adam usage [Radford et al. 15]

SGD for discriminator

Adam for generator

## One-sided Label smoothing

---

Предотвращает дискриминатор от того, чтобы давать слишком большие градиенты генератору

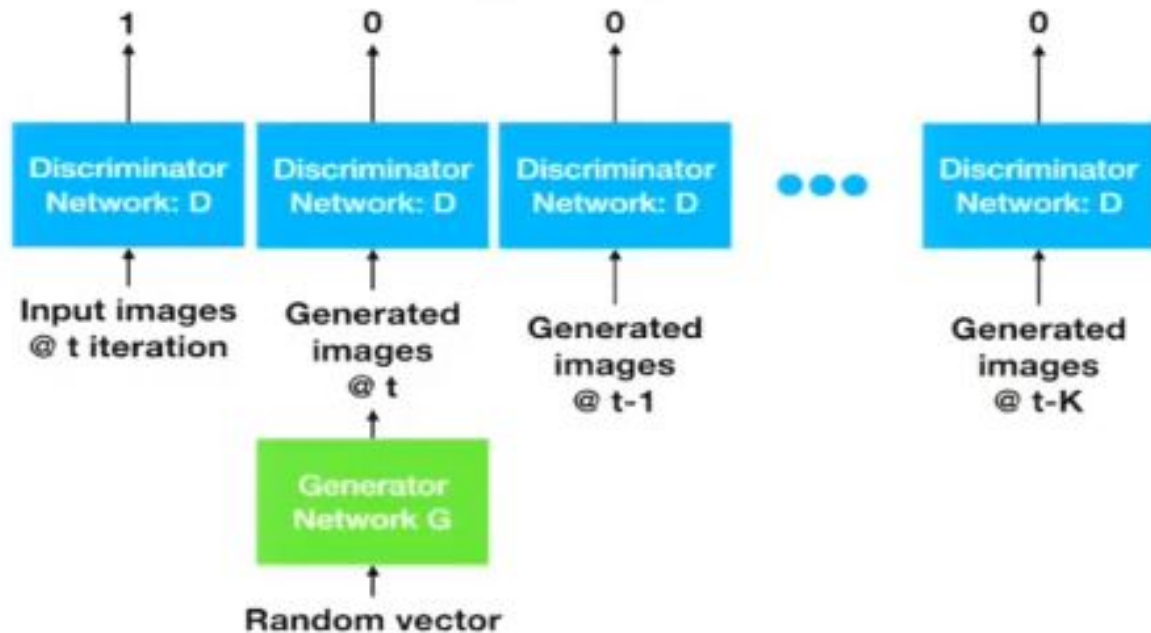
$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x \sim p_{\text{data}}} \lambda \log D(x) - \frac{1}{2}\mathbb{E}_z \log (1 - D(G(z)))$$

Some value smaller than 1; e.g., 0.9

- уменьшает уверенность, делает дискриминатор “слабее”
- Поощряет “рисковые примеры” (предотвращает экстраполирование)

# Historical Generator batches

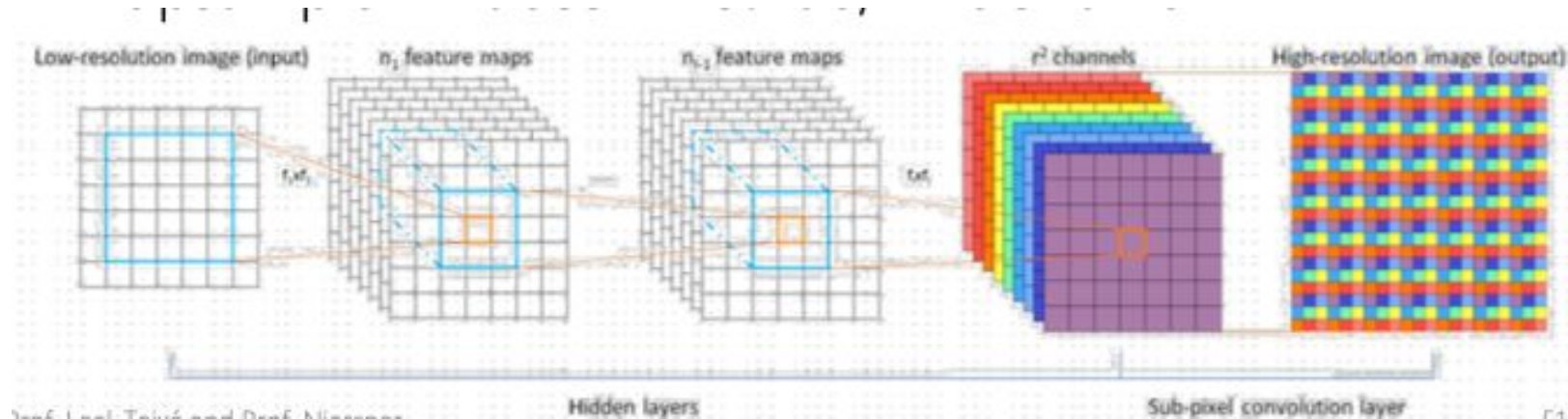
---



Help stabilize discriminator training in early stage

# Avoid sparse gradients

- Стабильность ГАНов падает, если градиенты разряженные (sparse)
- LeakyReLU -> хороша и для G, и для D
- Downsample -> используйте average pool, conv+stride
- Upsamples -> deconv+stride, PixelShuffle



# Exponential Averaging of Weights

---

Проблема: дискриминатор шумный из-за SGD

Вместо того чтобы брать финальный результат GANa, который будет смещен к последним итерациям (последним батчам)

- exponential average of weights
- хранить второй “вектор” весов, который усредняется
- почти ничего не стоит, усреднять веса только из последних  $n$  итераций

# Новые функции потерь

---

EBGAN: "Energy-based Generative Adversarial Networks"

BEGAN: "Boundary Equilibrium GAN"

WGAN: "Wasserstein Generative Adversarial Networks"

LSGAN: "Least Squares Generative Adversarial Networks"

но одной функции потерь недостаточно!



# EBGAN

---

Дискриминатор - автоэнкодер

хороший автоэнкодер: мы хотим чтобы reconstruction loss  $D(x)$  для реальных изображений был маленьким

Хороший критик: мы хотим наказывать дискриминатор если reconstruction loss для сгенерированных изображений падает ниже некоторого значения  $m$ .

$$D(x) = ||Dec(Enc(x)) - x||$$

$$\mathcal{L}_D(x, z) = D(x) + [m - D(G(z))]^+$$

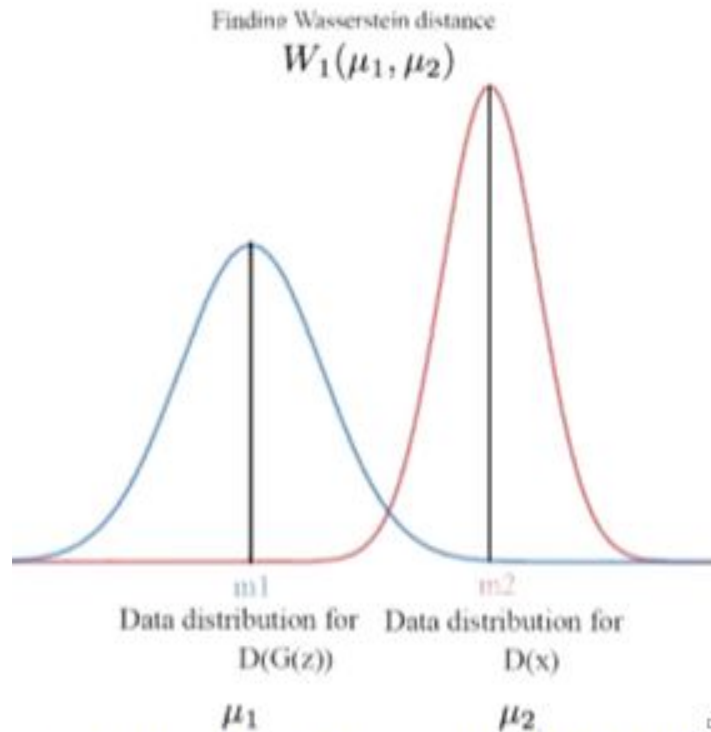
$$\mathcal{L}_G(z) = D(G(z))$$

$$\text{where } [u]^+ = \max(0, u)$$

# BEGAN

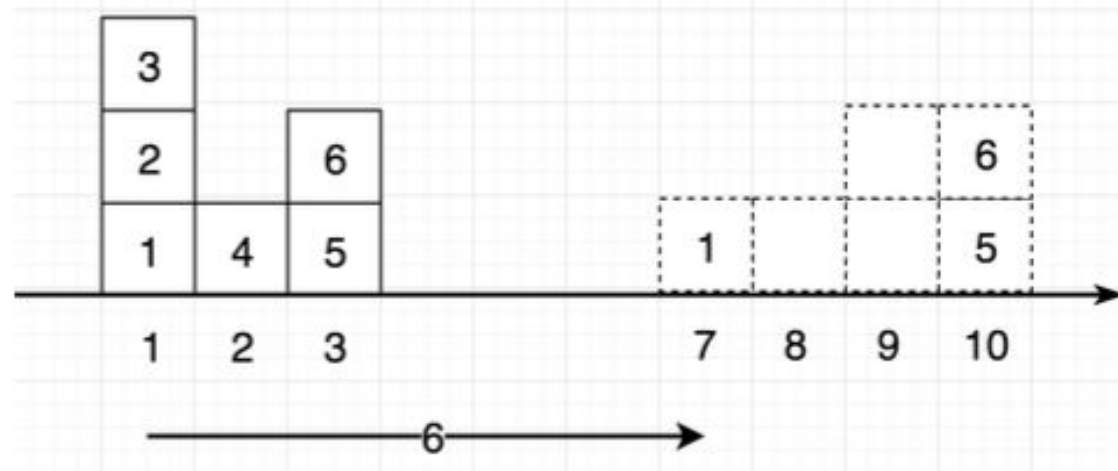
Схоже с EBGAN

Вместо reconstruction loss,  
измеряется разность в  
распределениях  
настоящих и  
сгенерированных изображений



# WGAN

## Earth Mover Distance / Wasserstein Distance



Minimum amount of work to move earth from  $p(x)$  to  $q(x)$

# WGAN

---

EMD дорогого считать

Переформулируем через его dual:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]$$

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

1-Lipschitz function: upper bound between densities

# WGAN

---

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

$f$  - функция-критик (нейросеть)

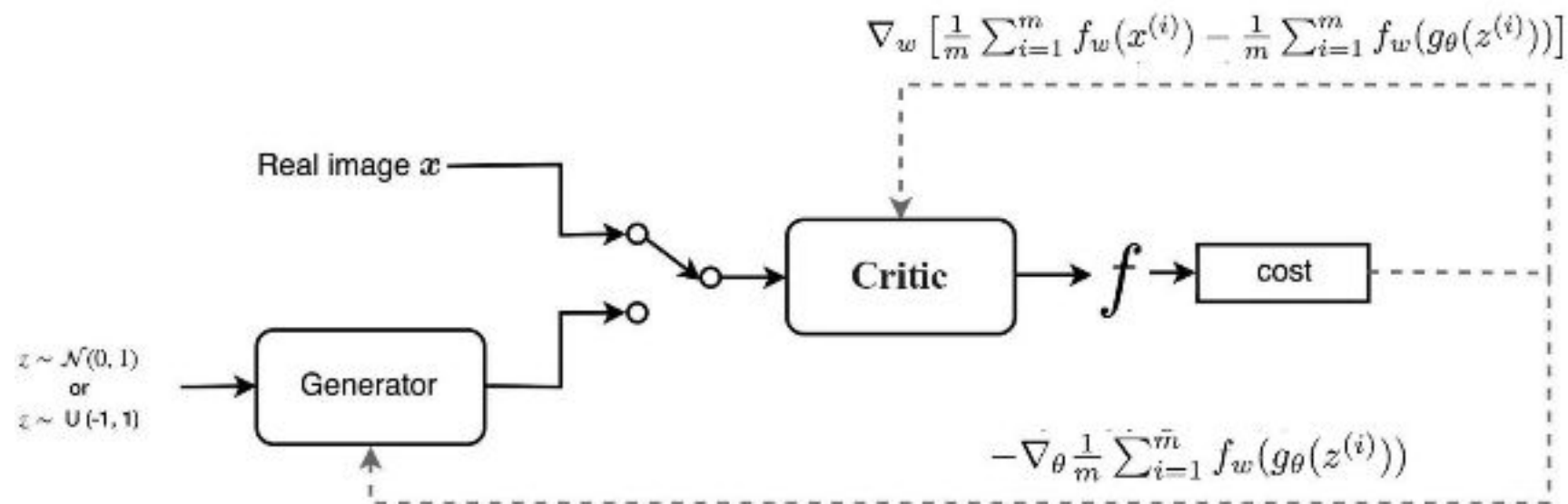
->  $f$  должна быть 1-Lipshitz, WGAN ограничивает максимальные веса в  $f$ ;

веса дискриминатора должны быть в определенном диапазоне определяемом гиперпараметром  $c$

$$w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$$

$$w \leftarrow \text{clip}(w, -c, c)$$

# WGAN



# WGAN

---

**Discriminator/Critic**

**Generator**

**GAN**

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

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m -\log (D(G(\mathbf{z}^{(i)})))$$

**WGAN**

$$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(x^{(i)}) - f(G(z^{(i)}))]$$

$$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m -f(G(z^{(i)}))$$

**Algorithm 1** WGAN, our proposed algorithm. All experiments in the paper used the default values  $\alpha = 0.00005$ ,  $c = 0.01$ ,  $m = 64$ ,  $n_{\text{critic}} = 5$ .

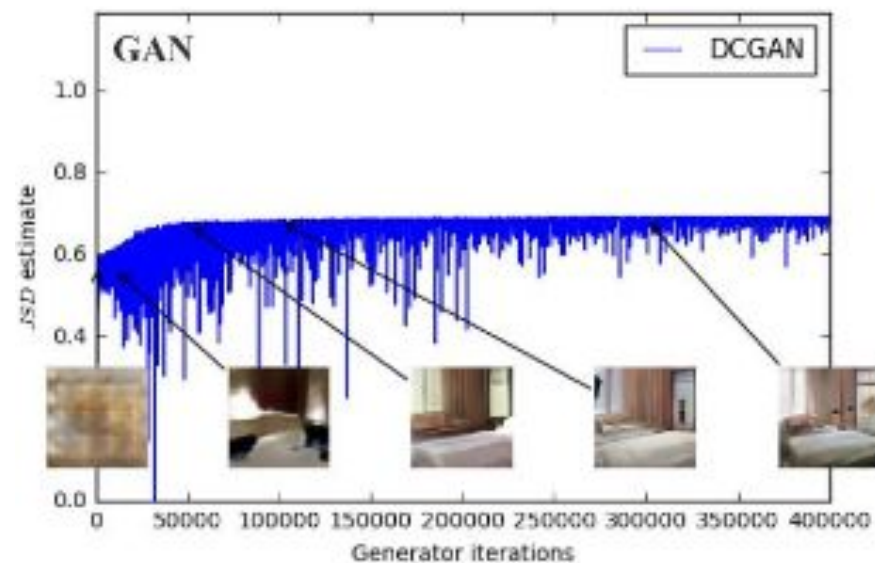
**Require:** :  $\alpha$ , the learning rate.  $c$ , the clipping parameter.  $m$ , the batch size.  $n_{\text{critic}}$ , the number of iterations of the critic per generator iteration.

**Require:** :  $w_0$ , initial critic parameters.  $\theta_0$ , initial generator's parameters.

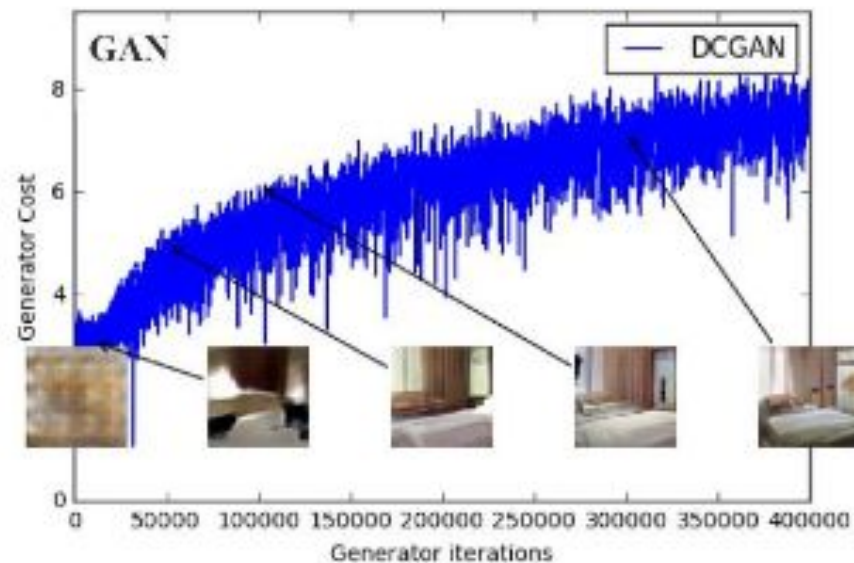
```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$ 
12: end while
```



# WGAN

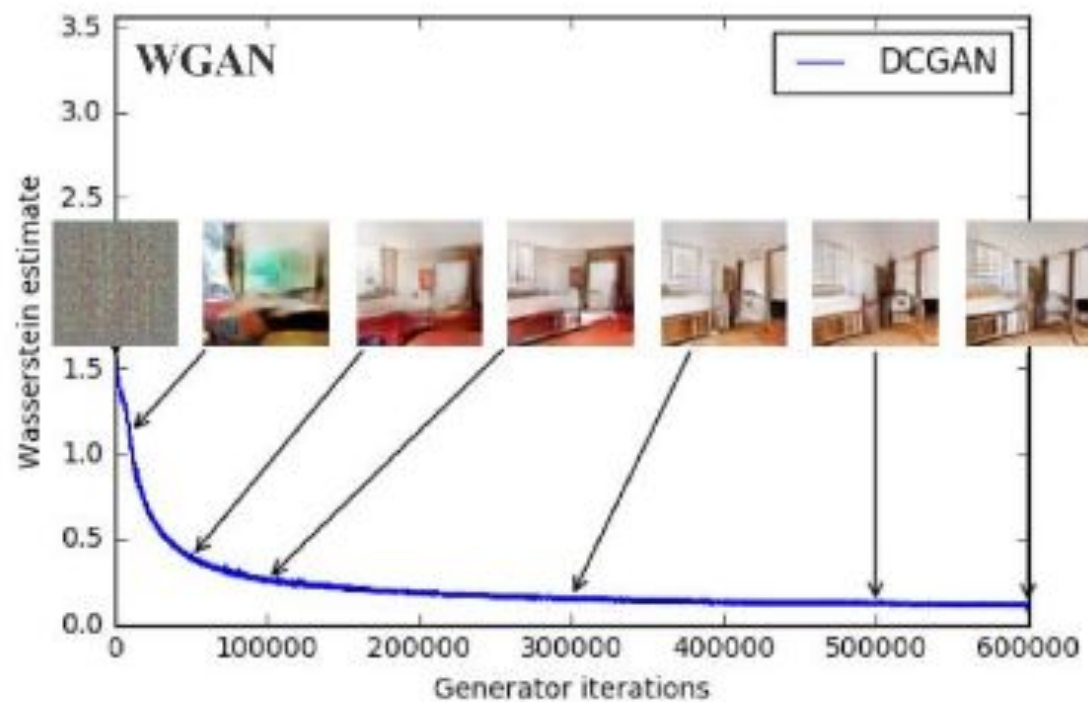


$$\frac{1}{m} \sum_{i=1}^m \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right)$$



$$\frac{1}{m} \sum_{i=1}^m -\log \left( D \left( G \left( z^{(i)} \right) \right) \right)$$

# WGAN



$$\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_{\theta}(z^{(i)}))$$

# WGAN

---

+побеждает mode collapse

+генератор учится даже когда критик превосходит

+настоящая сходимость

- наложение условий Липшица довольно тяжелое
- Обрезание весов сложное
  - слишком большое: занимает много времени достигнуть оптимум, медленное обучение
  - Слишком маленькой: затухающие градиенты

# GAN losses

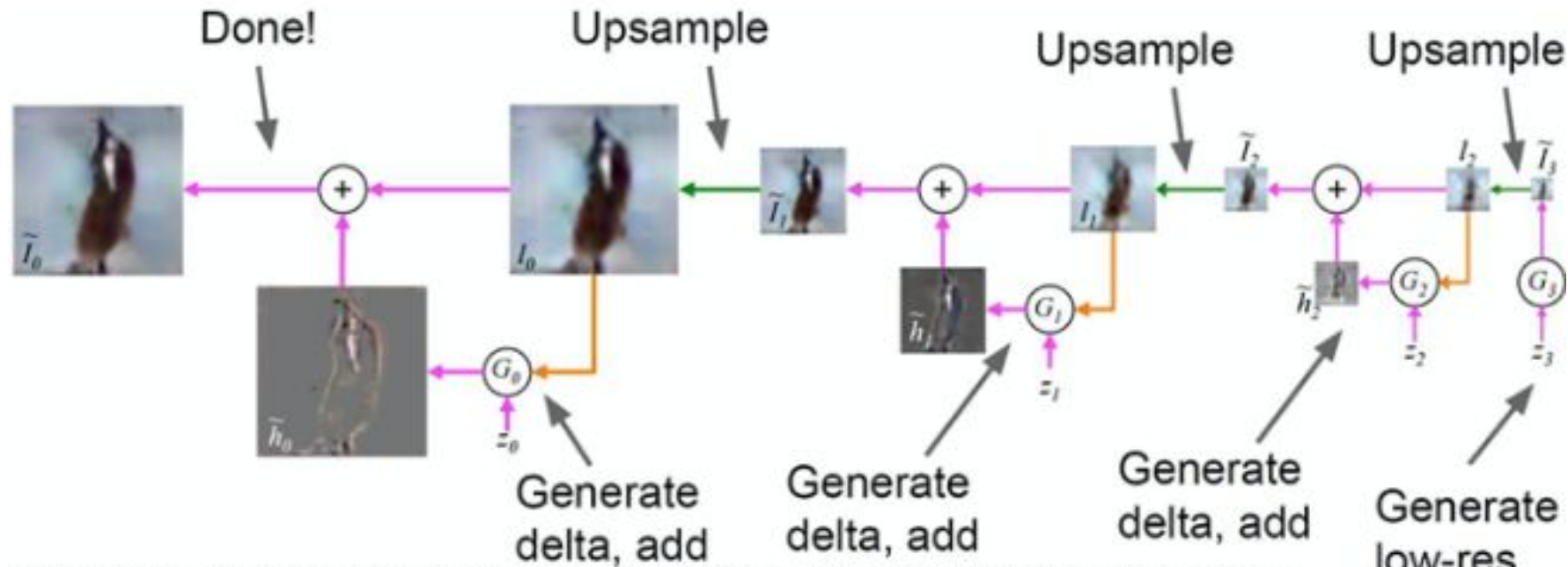
---

Много вариаций!

Высокоуровневое понимание: “loss” это металосс, чтобы обучить настоящий лосс (D) для создание градиентов для G

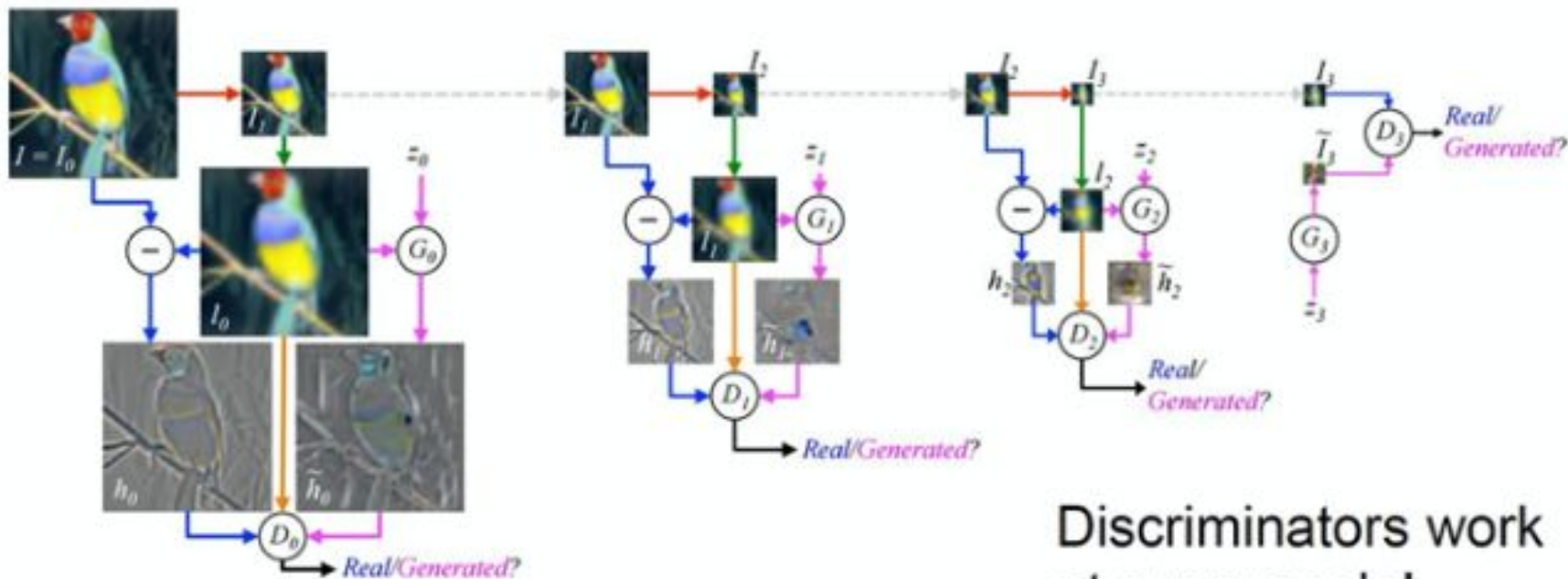
Всегда начинайте с простого: если не сходится, попробуйте базовые вещи (AE, VAE, ‘simple heuristic’ GAN)

# Multiscale GANs



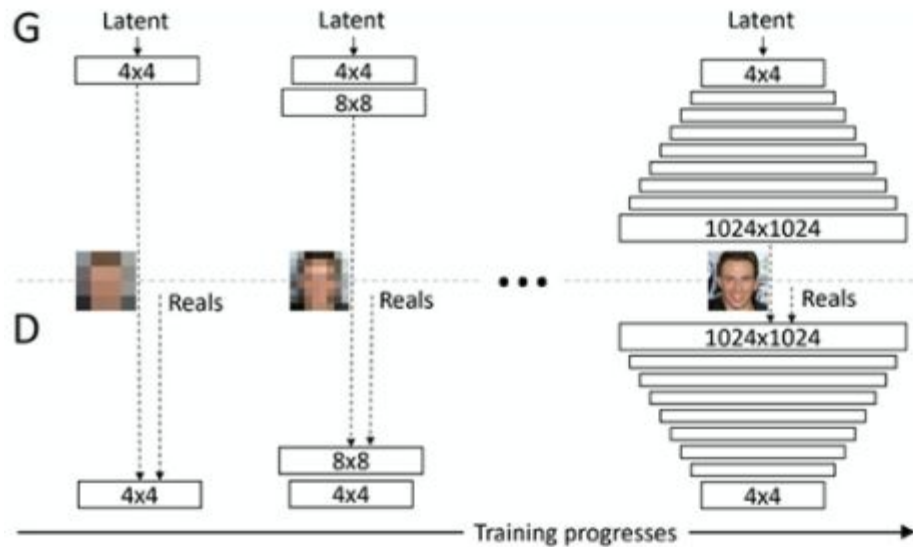
Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

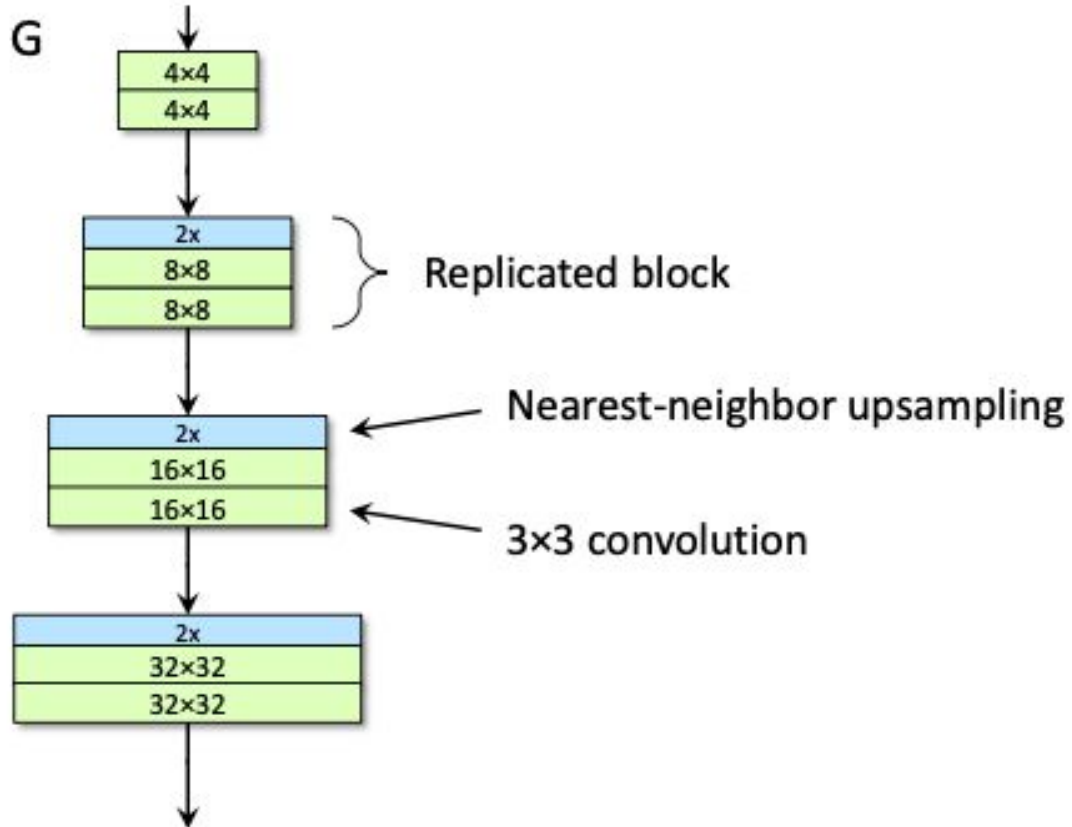
# Multiscale GANs



Discriminators work  
at every scale!

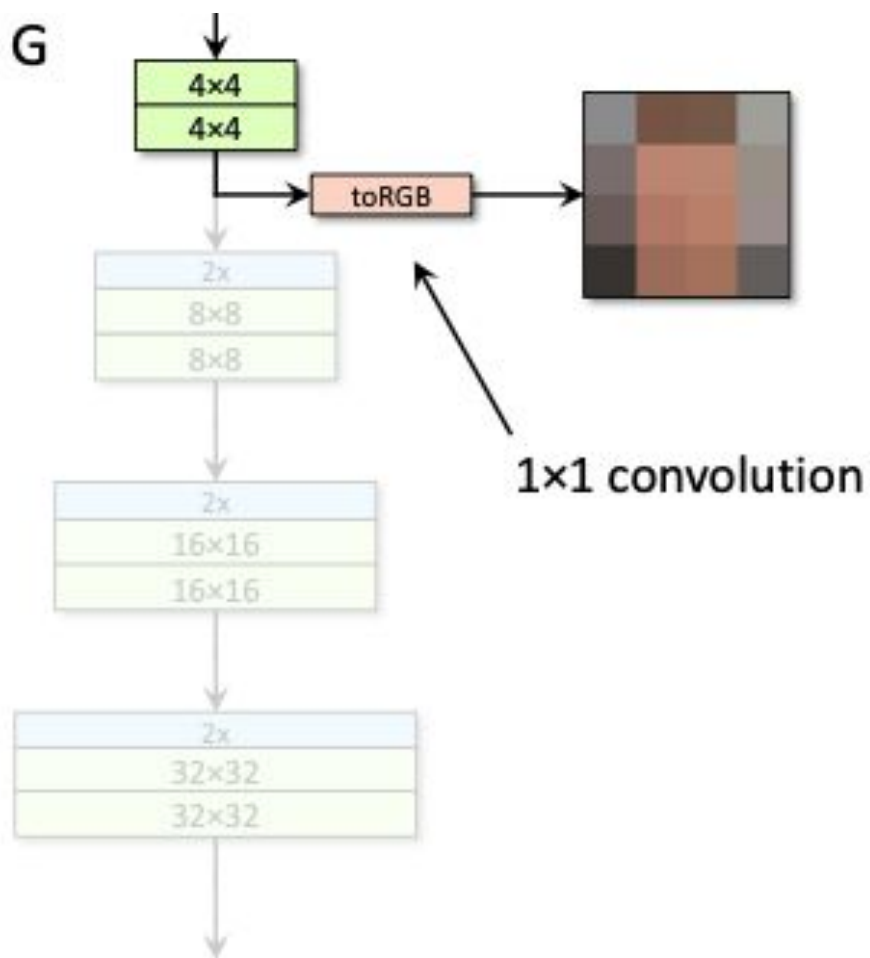
# Growing GANs



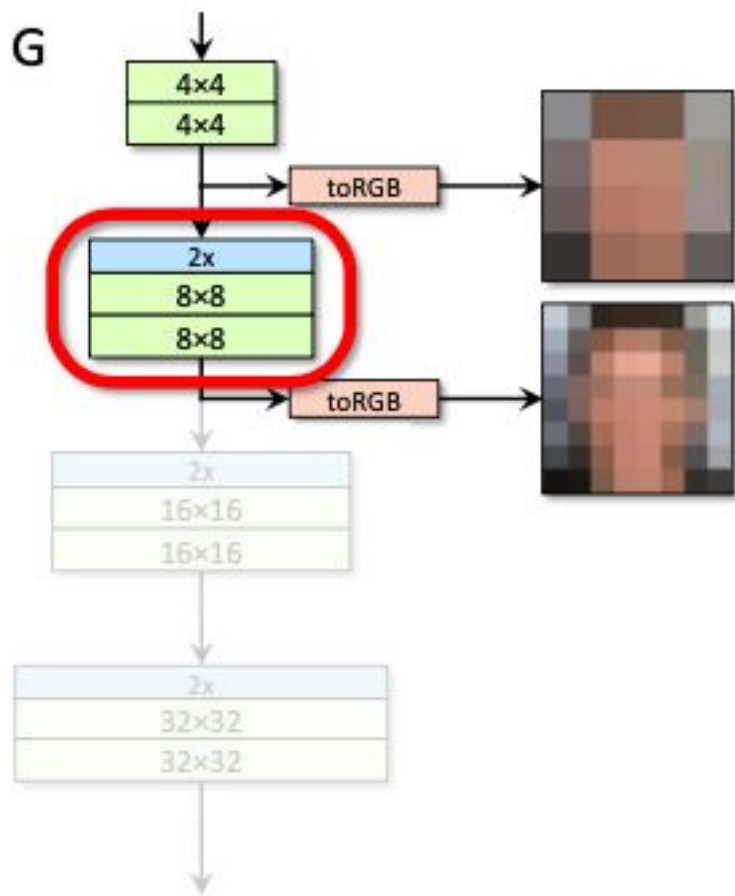




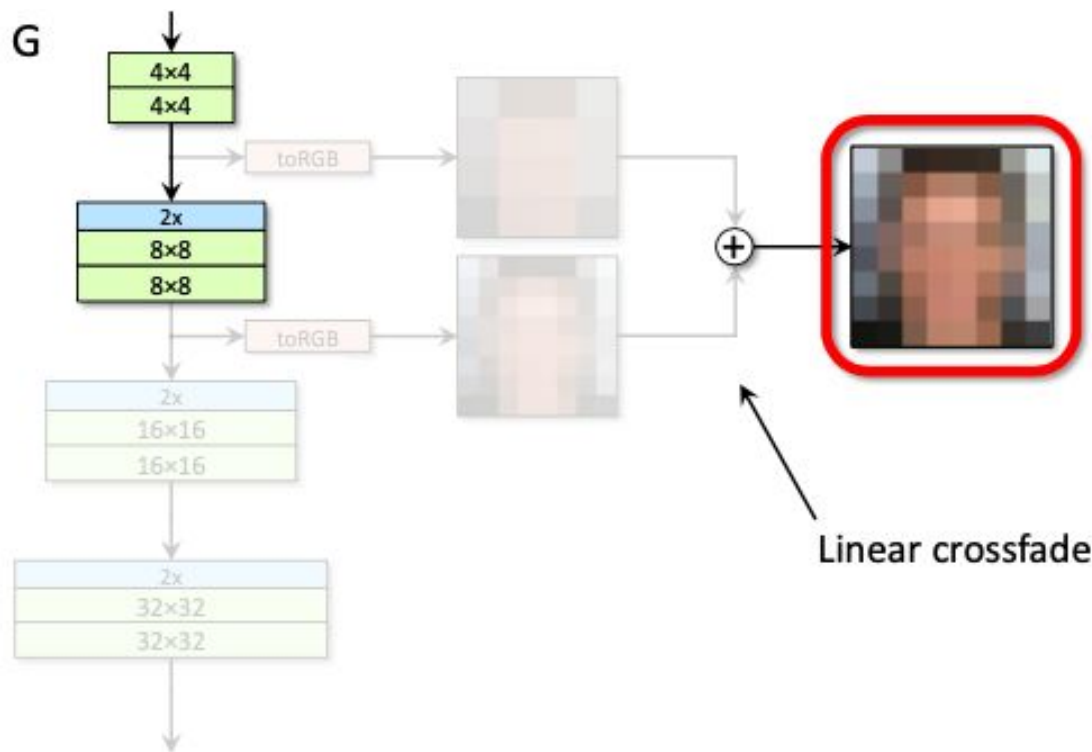
G



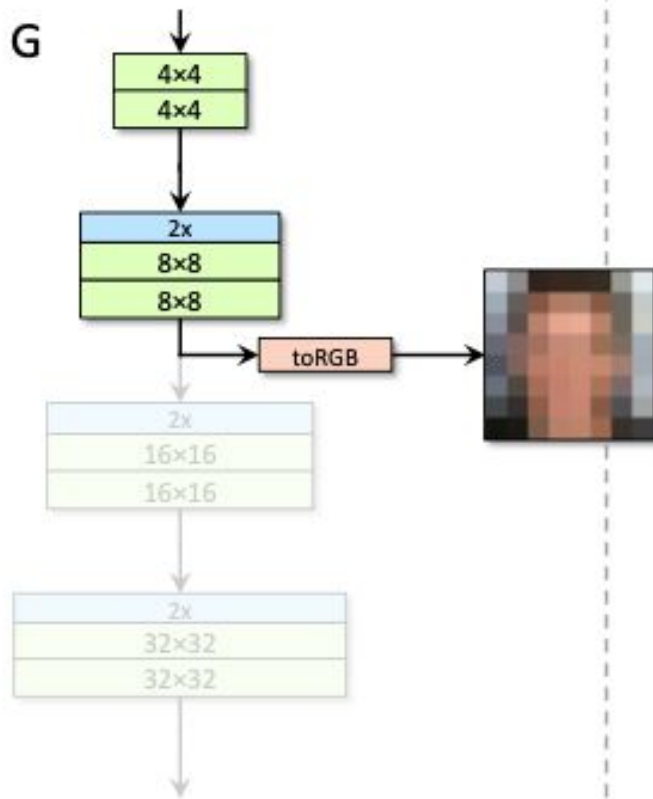
G



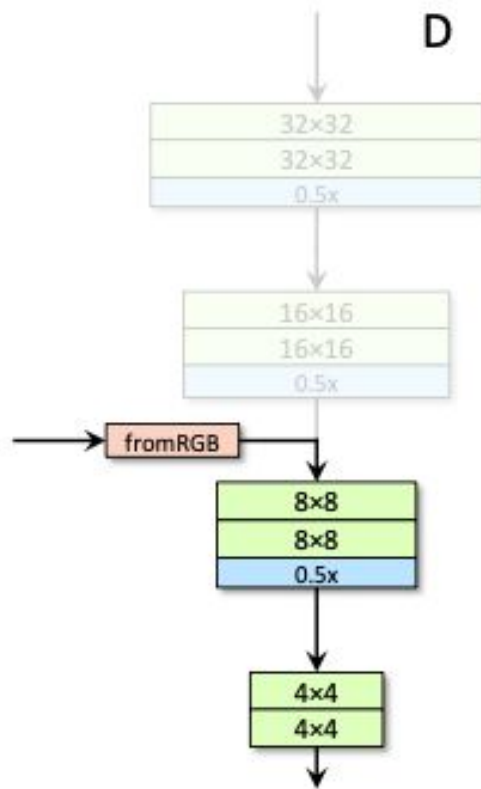
G



G



D



# Growing GANs

---



# Lots of GAN variations

---

- Hundreds of GAN papers in the last two years
  - > Mostly with different losses
  - > Extremely hard to train and evaluate

## Are GANs Created Equal? A Large-Scale Study

Mario Lucic\* Karol Kurach\* Marcin Michalski Sylvain Gelly Olivier Bousquet  
Google Brain

### Abstract

*Generative adversarial networks (GAN) are a powerful subclass of generative models. Despite a very rich research*

GAN algorithm(s) perform objectively better than the others. That's partially due to the lack of robust and consistent metric, as well as limited comparisons which put all algorithms on equal footing, including the computational

v 2017



It's coding time