

Глубинное обучение

Генеративные модели в CV

Михаил Лазарев

Задачи Computer Vision

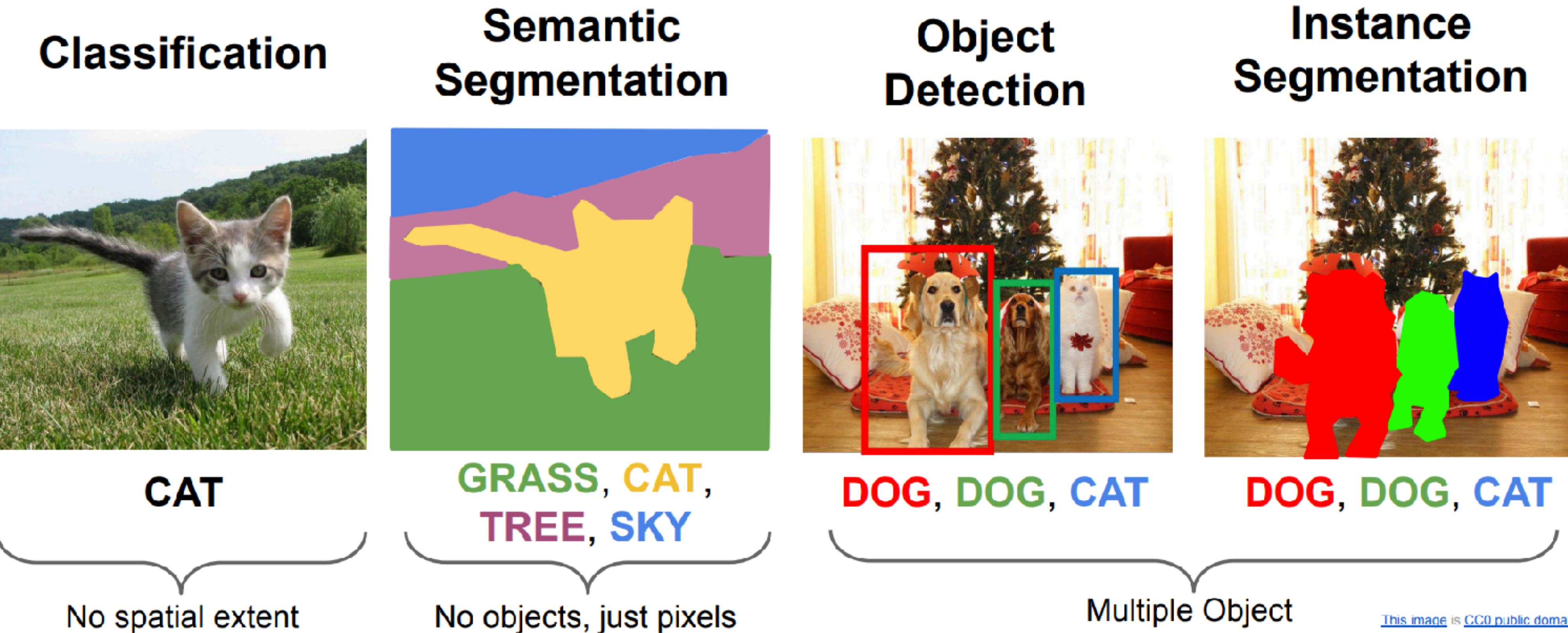
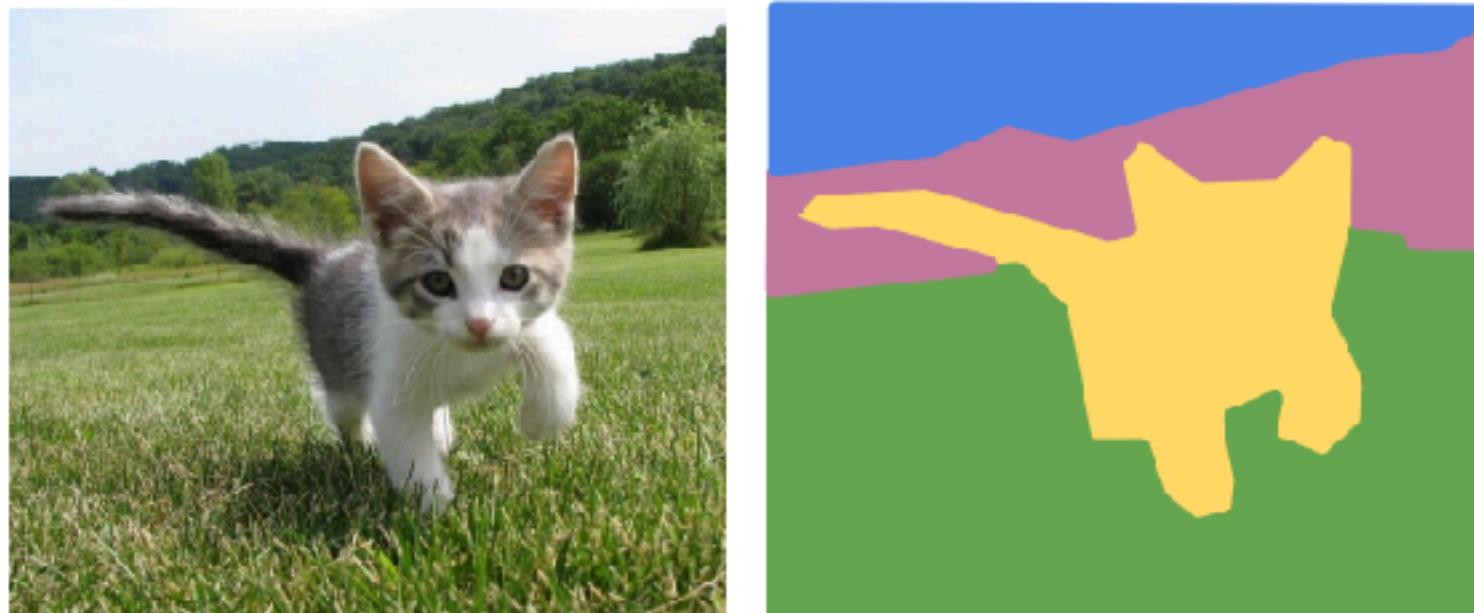


Image credit

Semantic Segmentation

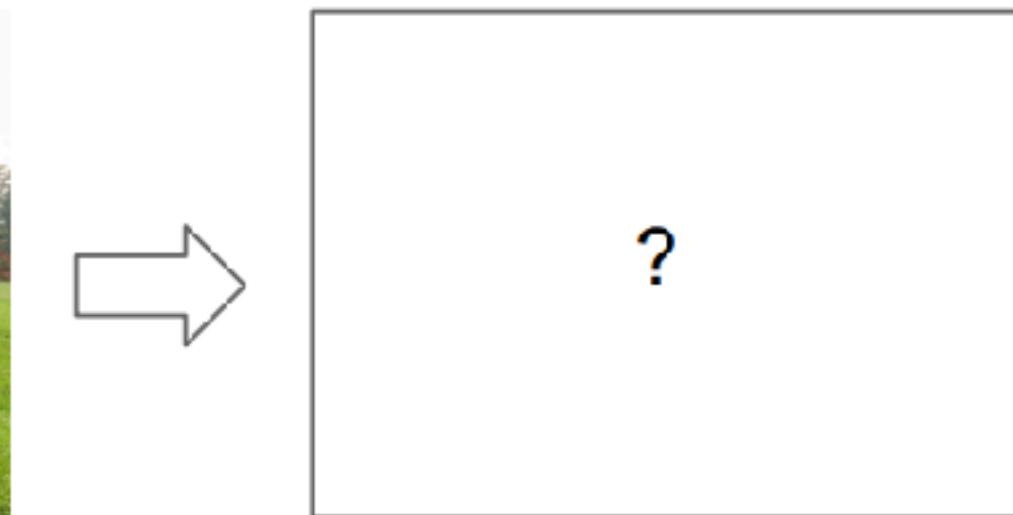
Задача: для каждого пикселя изображения определить его категорию



GRASS, CAT,
TREE, SKY, ...



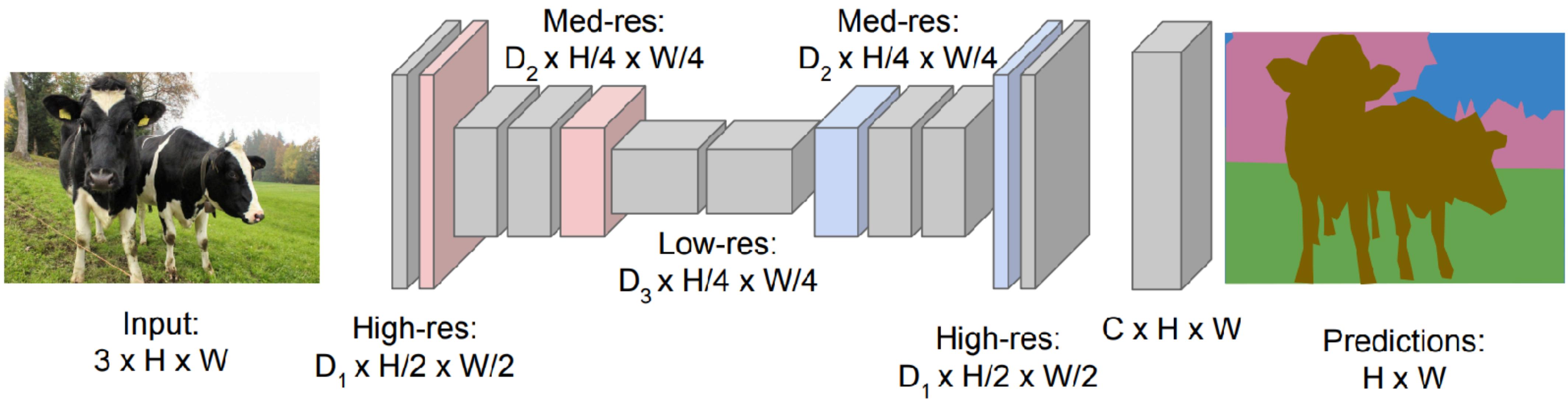
At test time, classify each pixel of a new image.



[Image credit](#)

Semantic Segmentation

Идея: U-net - downsampling & upsampling



Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Object Detection

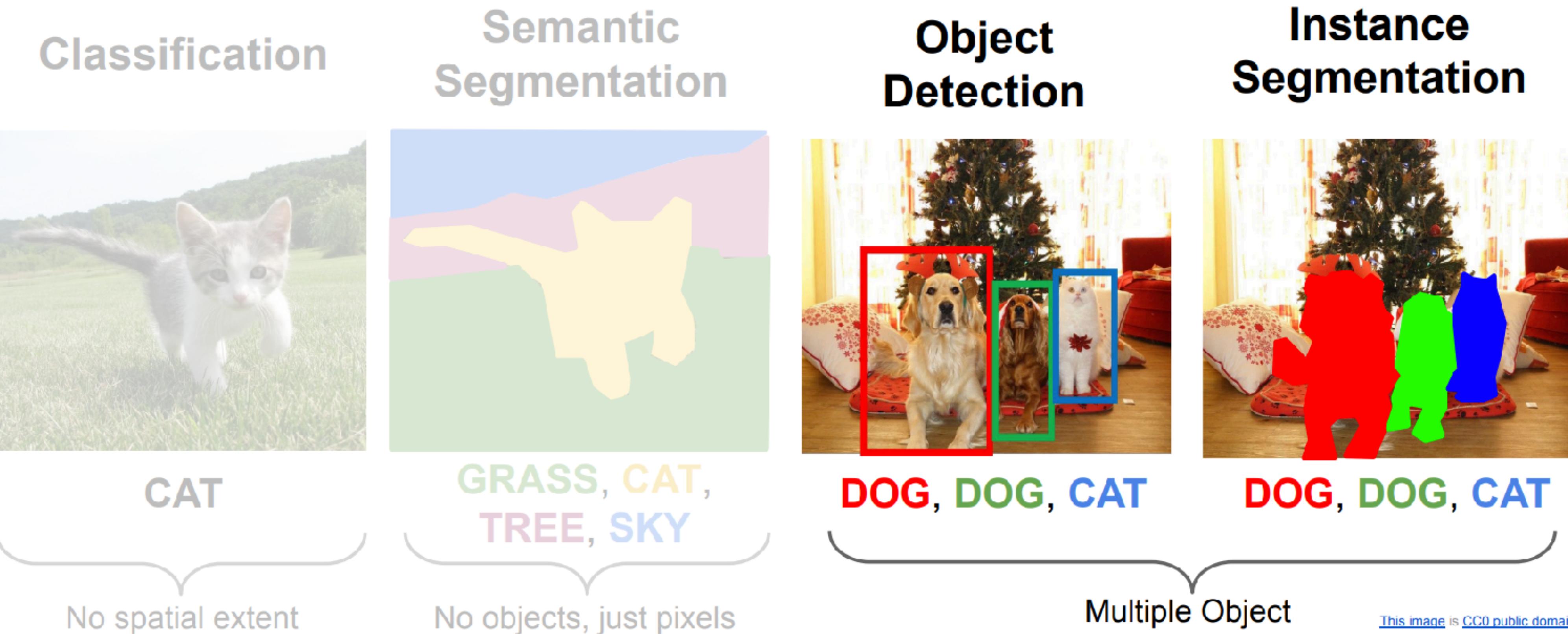
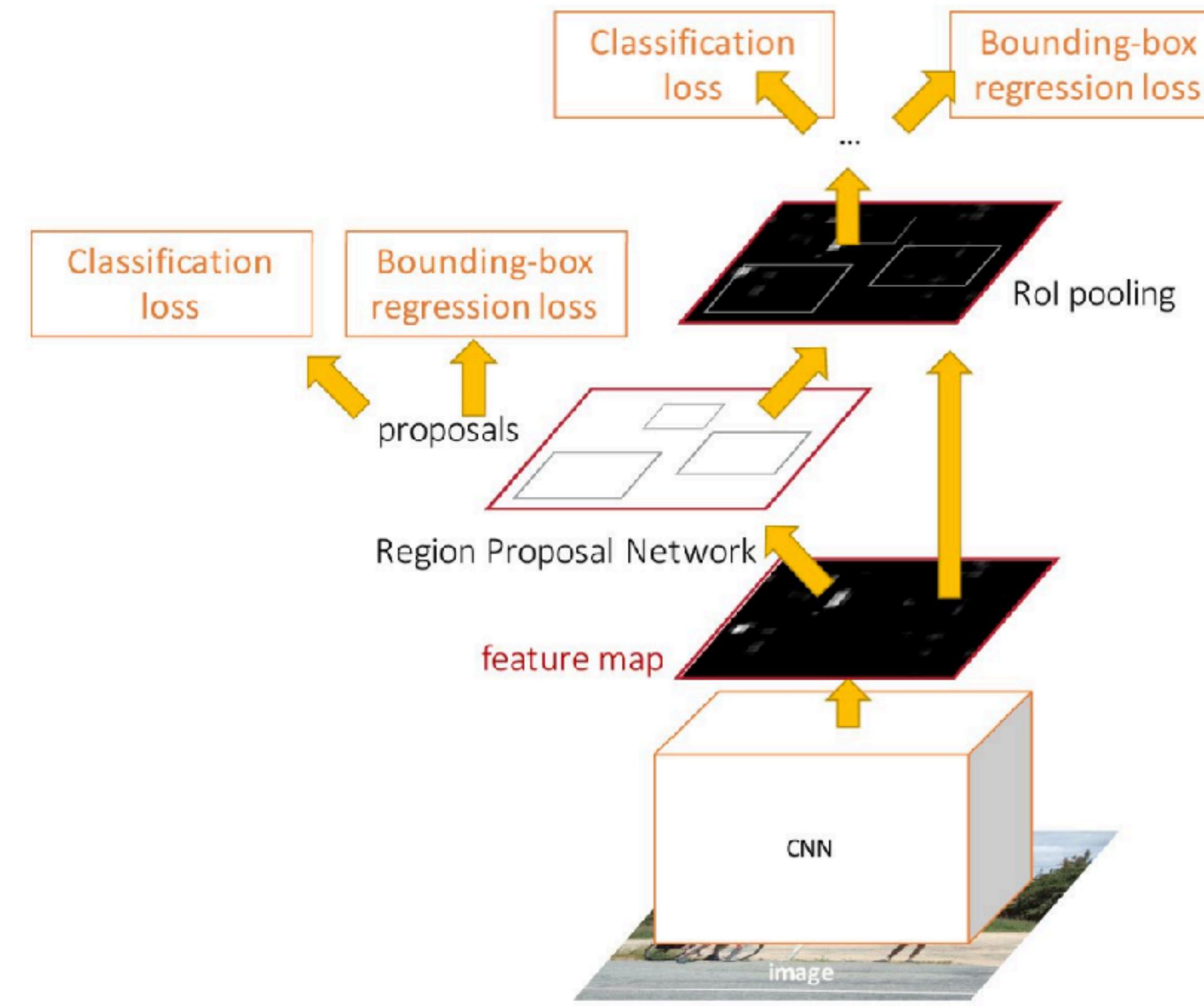


Image credit

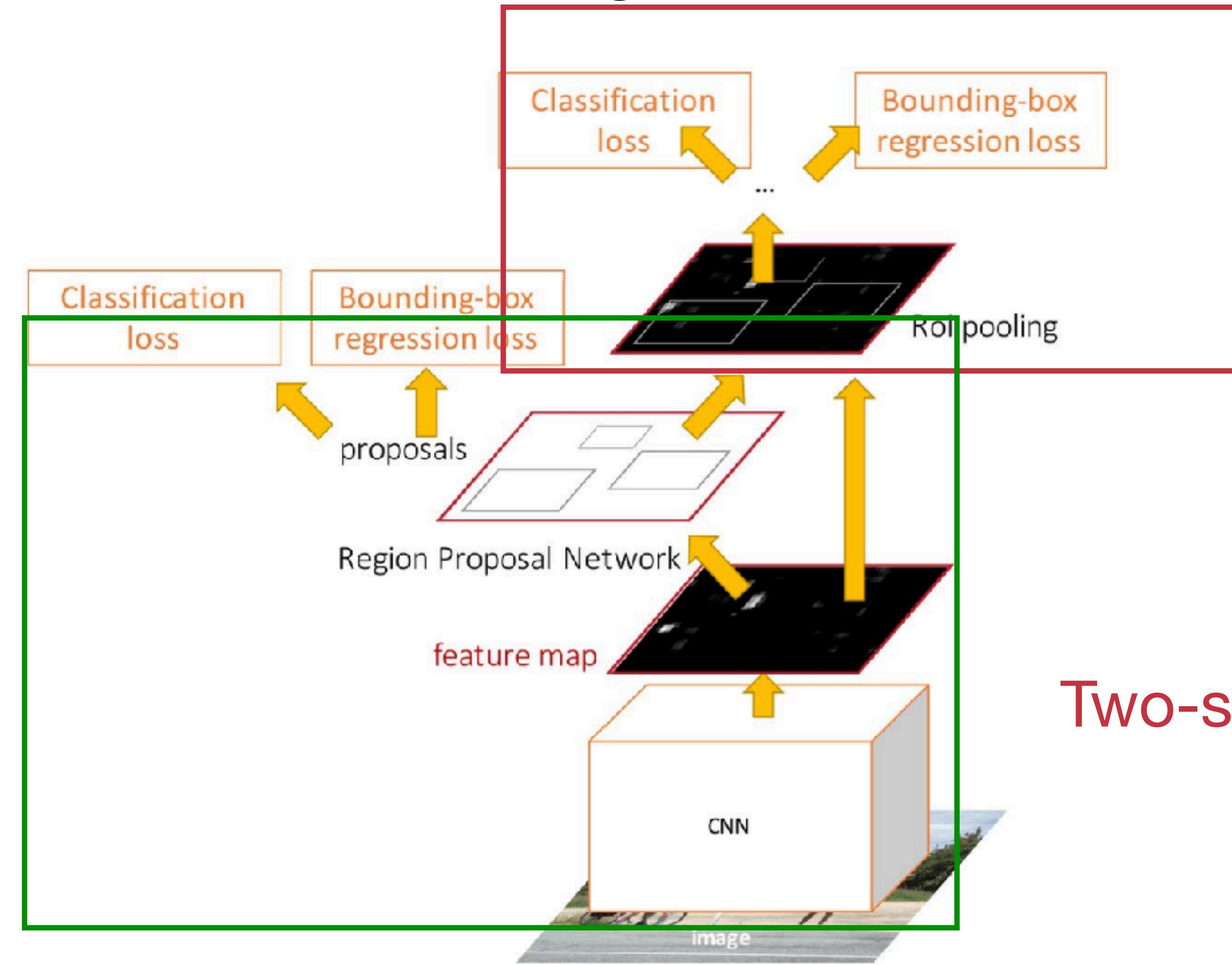
Object Detection

Faster R-CNN: use Region Proposal Network



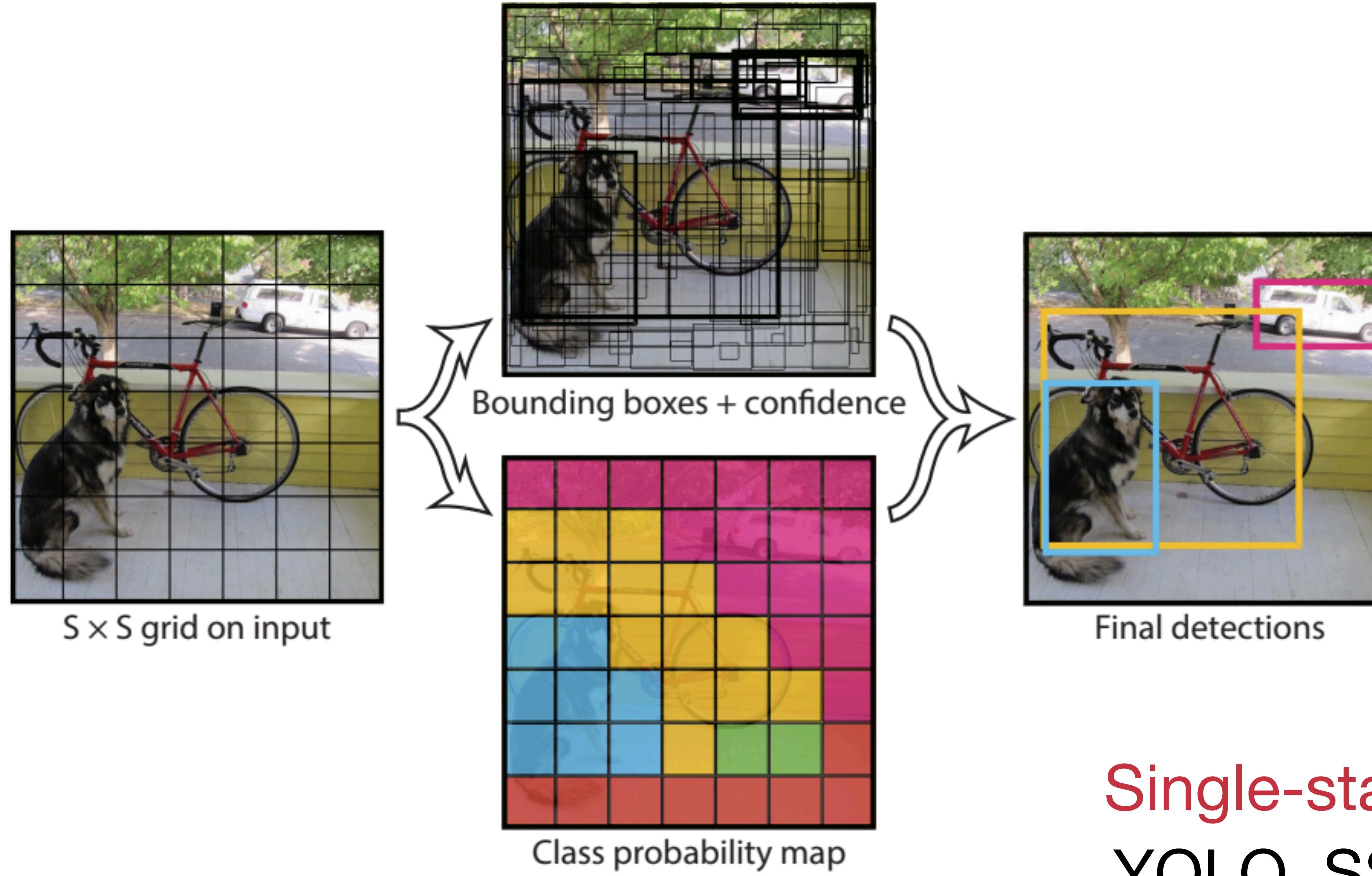
Object Detection

Faster R-CNN: use Region Proposal Network



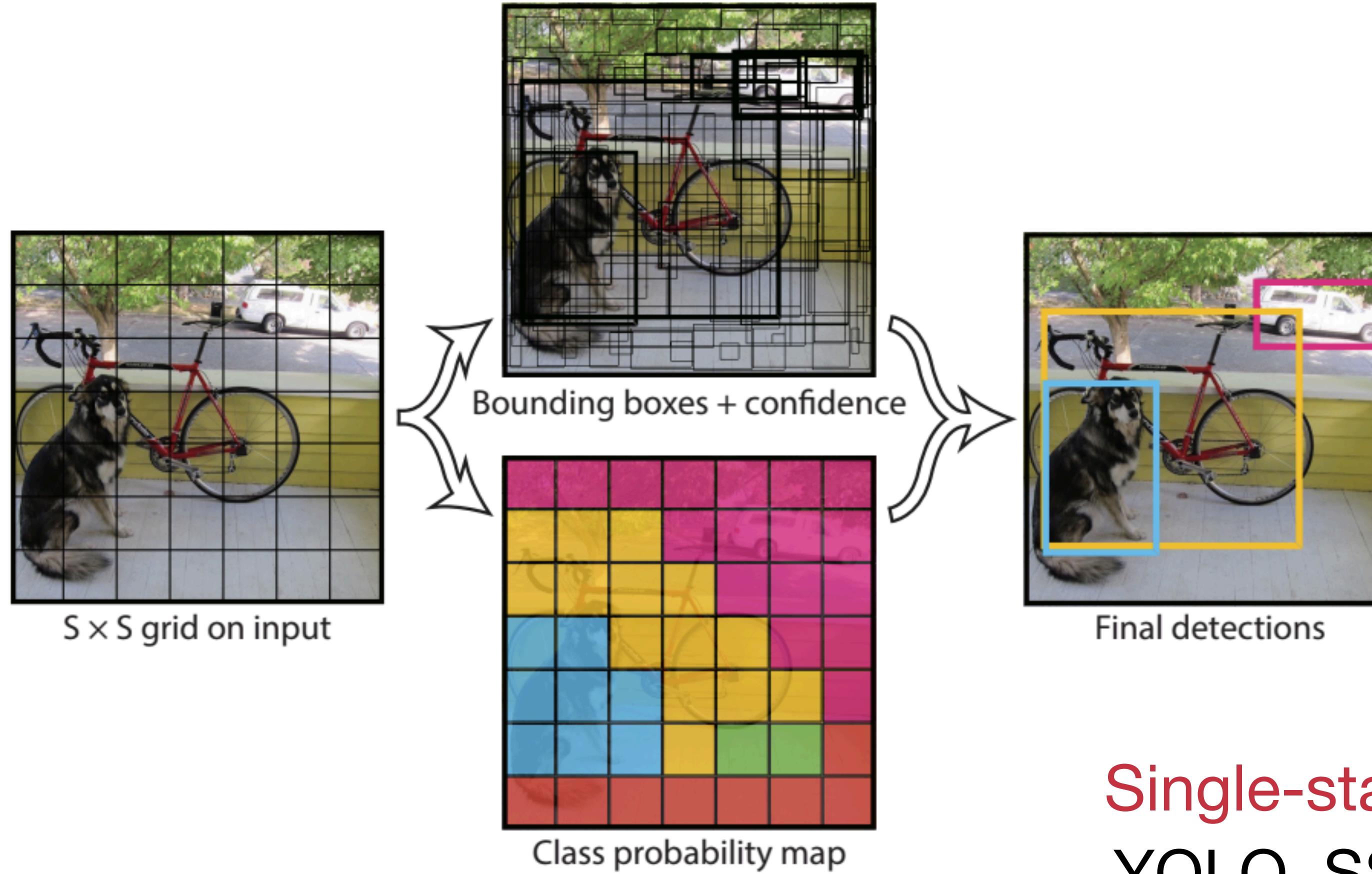
Two-stage Object Detector

Object Detection



Single-stage Object Detector
YOLO, SSD, RetinaNet

Object Detection



Single-stage Object Detector
YOLO, SSD, RetinaNet
менее точные, быстрые

[Image credit](#)

Instance Segmentation

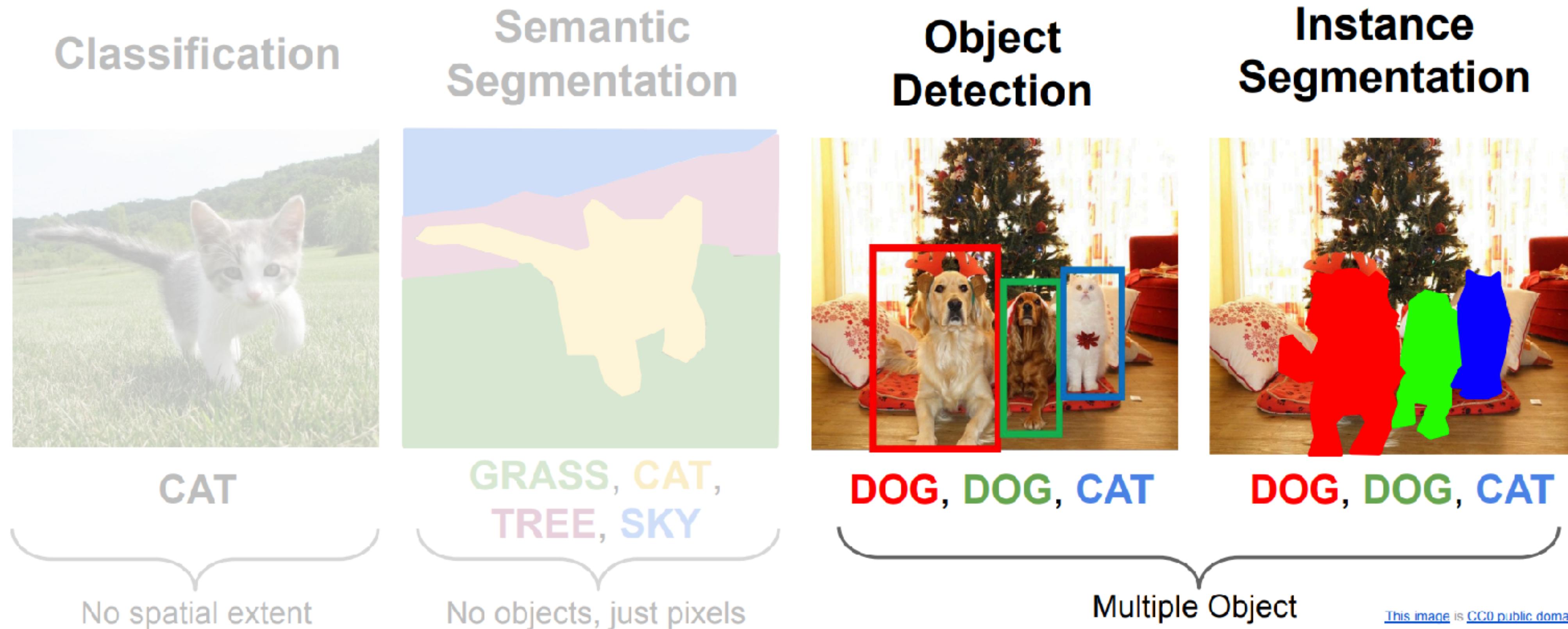
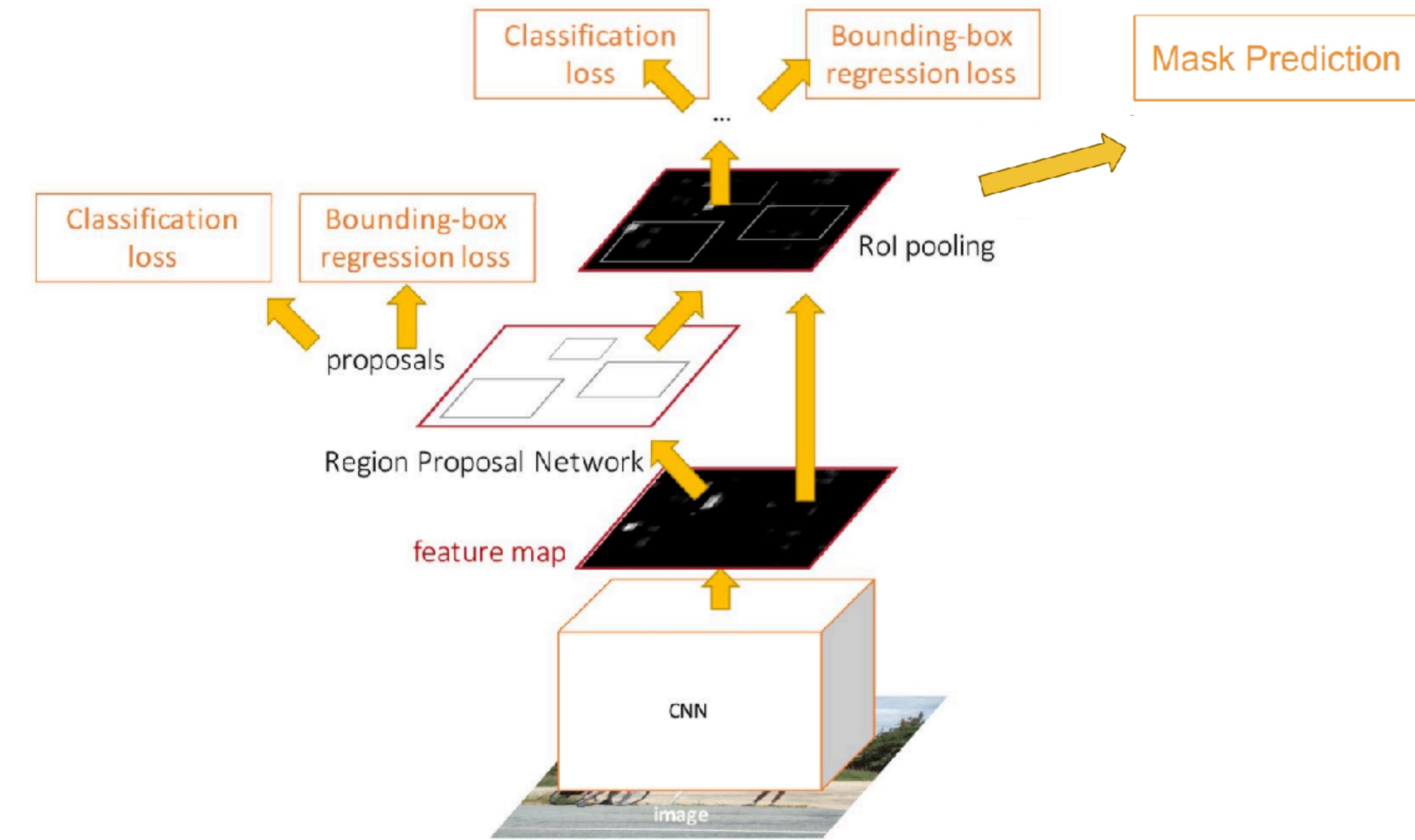


Image credit

Instance Segmentation

Masked R-CNN



Генеративные модели в CV

Supervised vs Unsupervised Learning

Обучение с учителем (supervised)

Дано: объекты x_1, \dots, x_n
 метки y_1, \dots, y_n

Хотим выучить: $f(x, \theta) \rightarrow y$

Classification,

Semantic Segmentation,

Object Detection,

Instance Segmentation,

...

Обучение без учителя (unsupervised)

Дано: объекты x_1, \dots, x_n

Хотим выучить: внутренние
взаимосвязи между объектами

Supervised vs Unsupervised Learning

Обучение с учителем (supervised)

Дано: объекты x_1, \dots, x_n
 метки y_1, \dots, y_n

Хотим выучить: $f(x, \theta) \rightarrow y$

Classification,
Semantic Segmentation,
Object Detection,
Instance Segmentation,
...

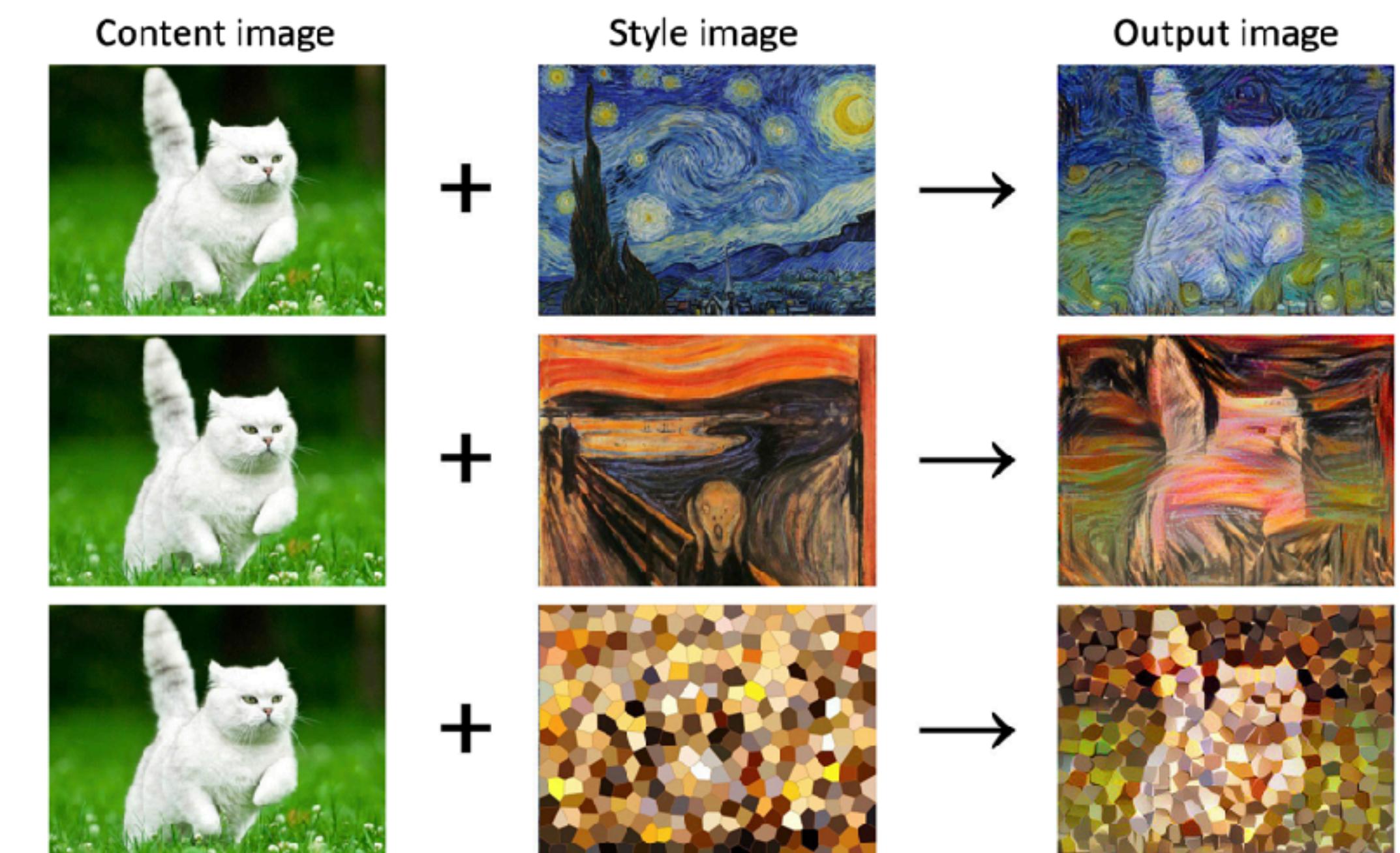
Обучение без учителя (unsupervised)

Дано: объекты x_1, \dots, x_n

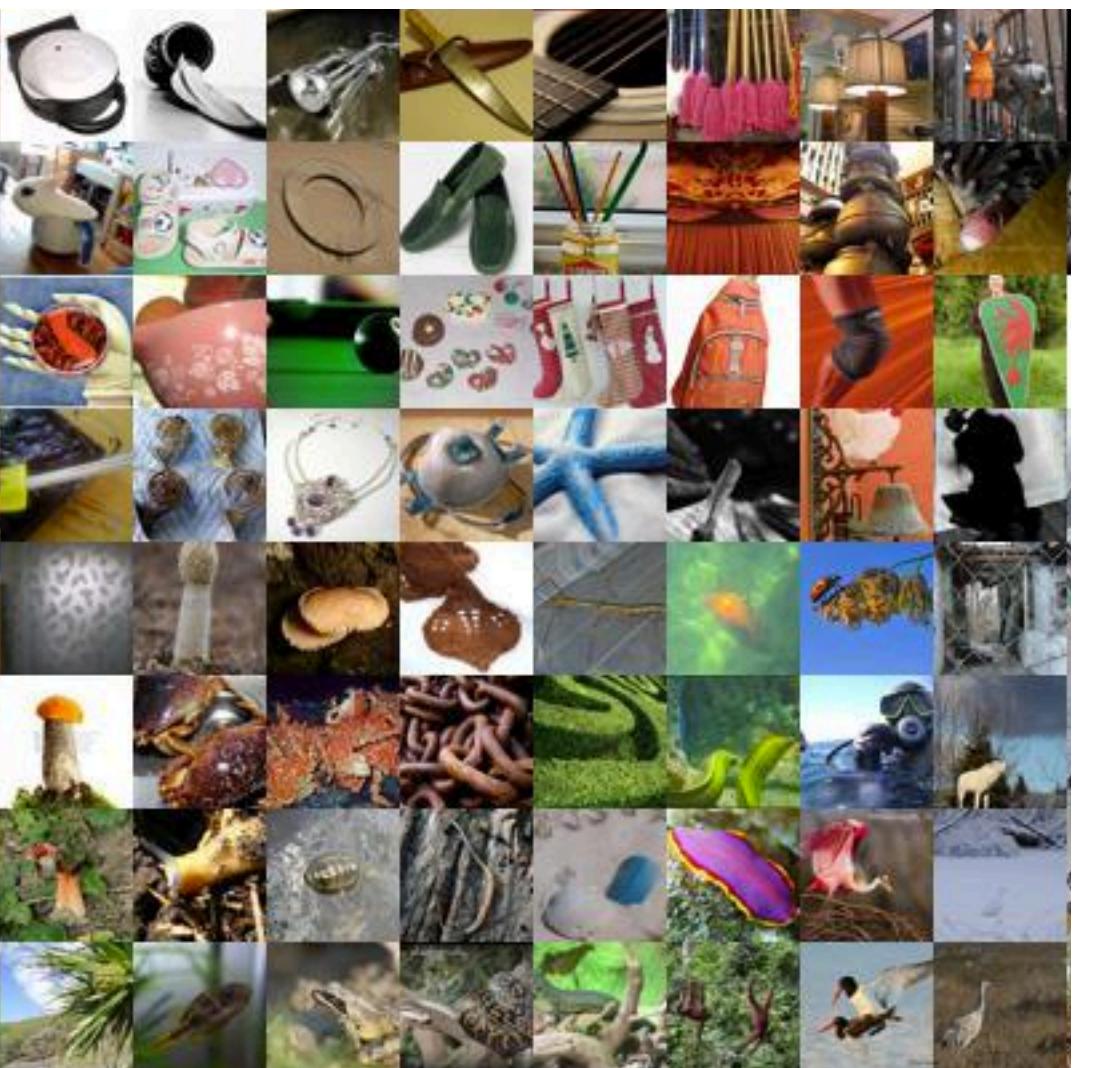
Хотим выучить: внутренние
взаимосвязи между объектами

Clustering,
Dimension reduction,
Density Estimation,
...

Генеративные модели

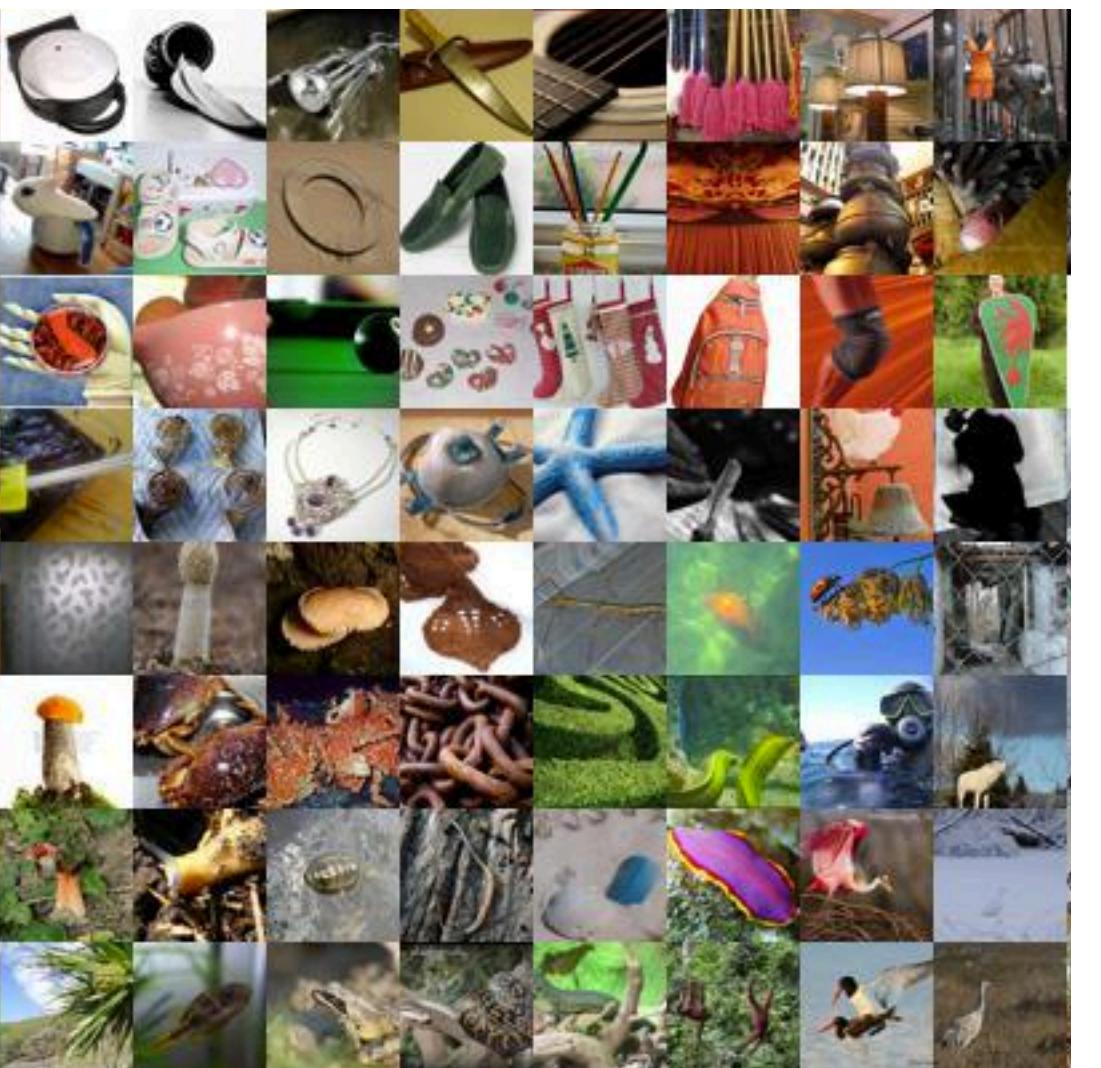


Генеративные модели

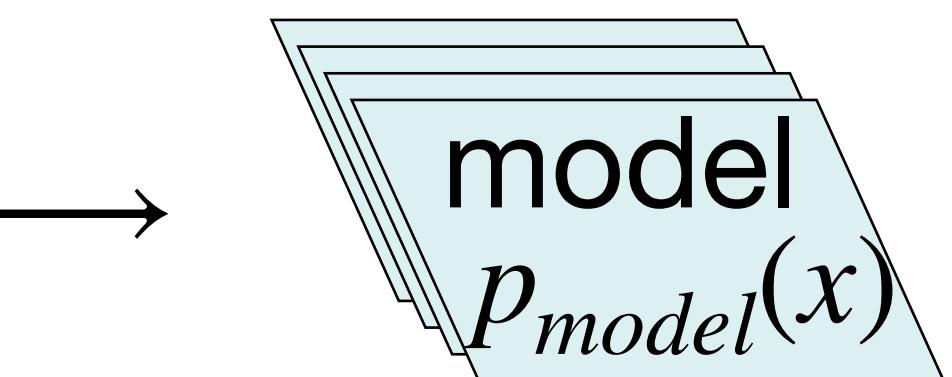


Train data: $p_{data}(x)$

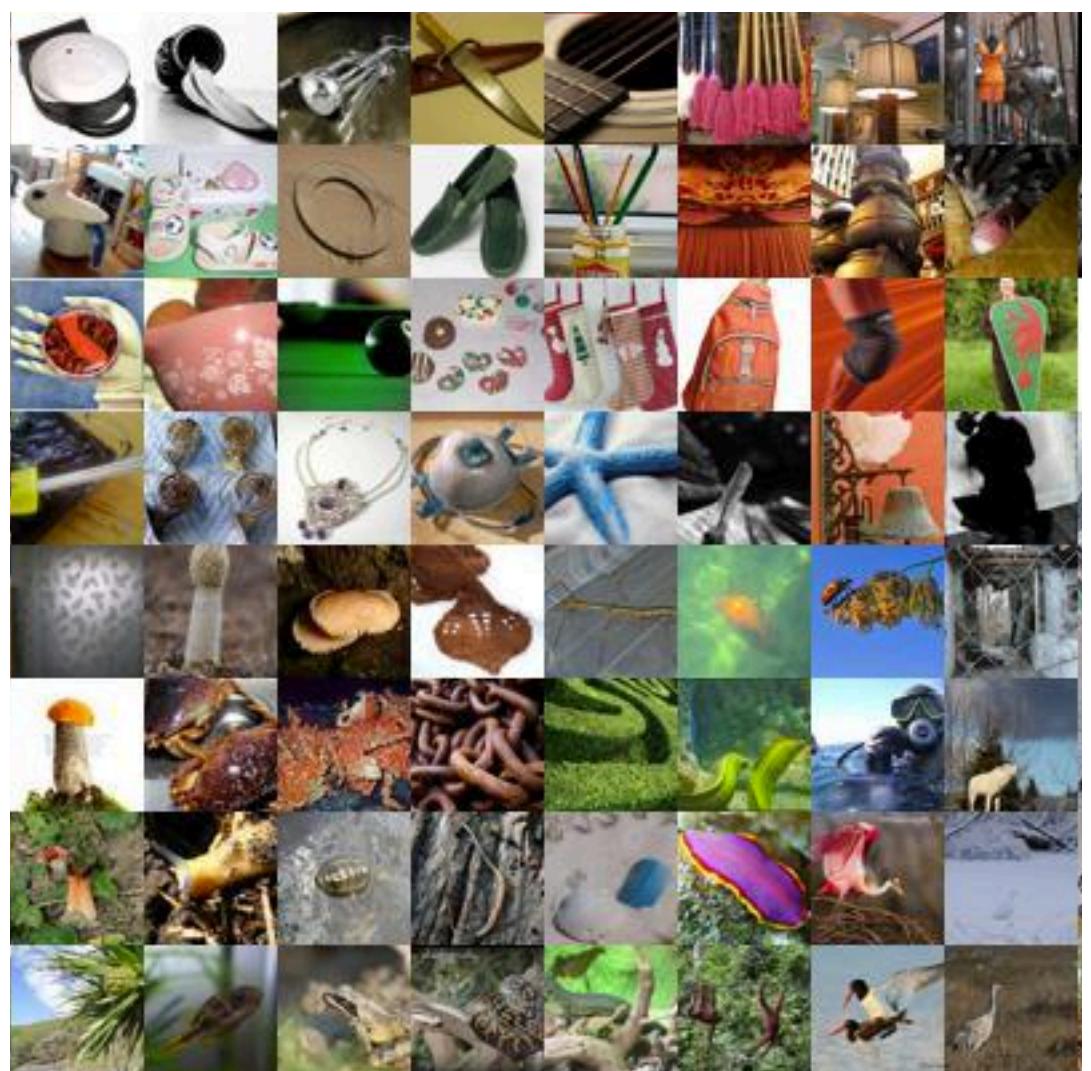
Генеративные модели



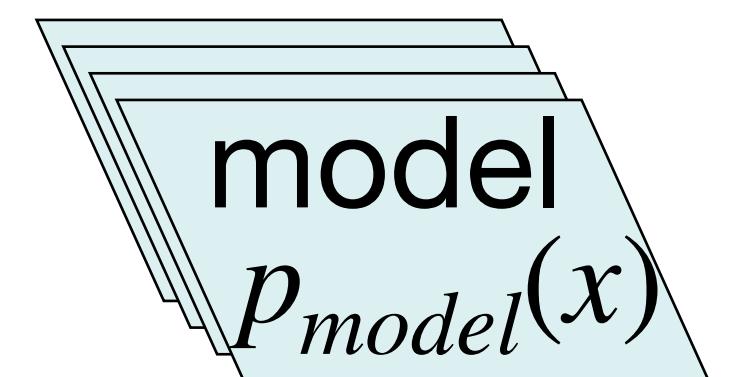
Train data: $p_{data}(x)$



Генеративные модели



Train data: $p_{data}(x)$



model
 $p_{model}(x)$



Model samples

Генеративные модели



Train data: $p_{data}(x)$

model
 $p_{model}(x)$



Model samples

Explicit density estimation

явно определяем $p_{model}(x)$

PixelRNN/CNN, Glow, Ffjord, VAE

Implicit density estimation

не определяем явно, но сэмплируем

GANs

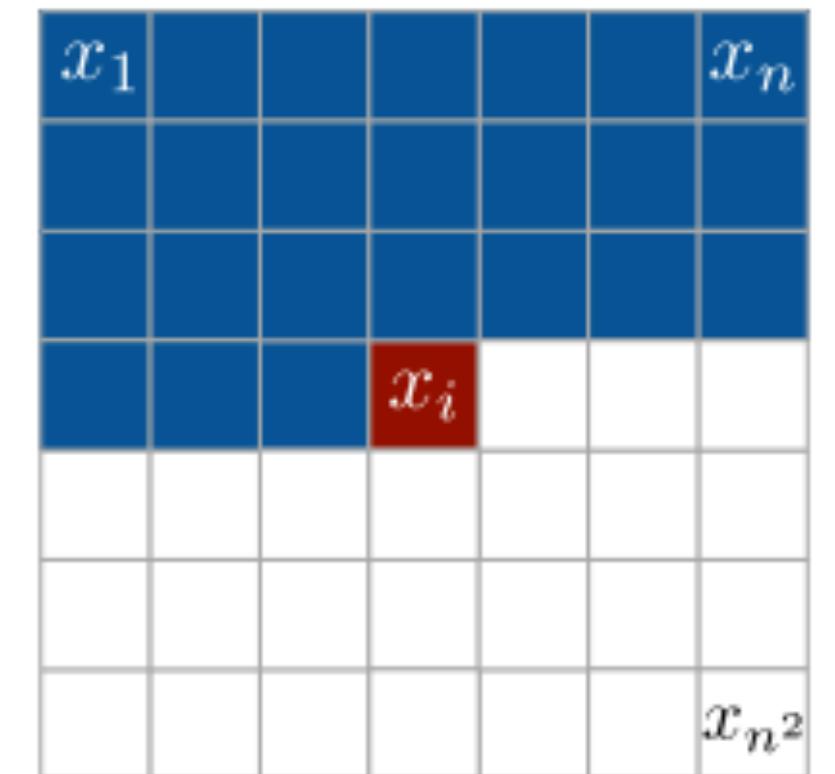
PixelCNN

PixelCNN

Chain rule: $p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$

↗
вероятность i -го пикселя при условии $i-1$ предыдущих

Авторегрессионная модель



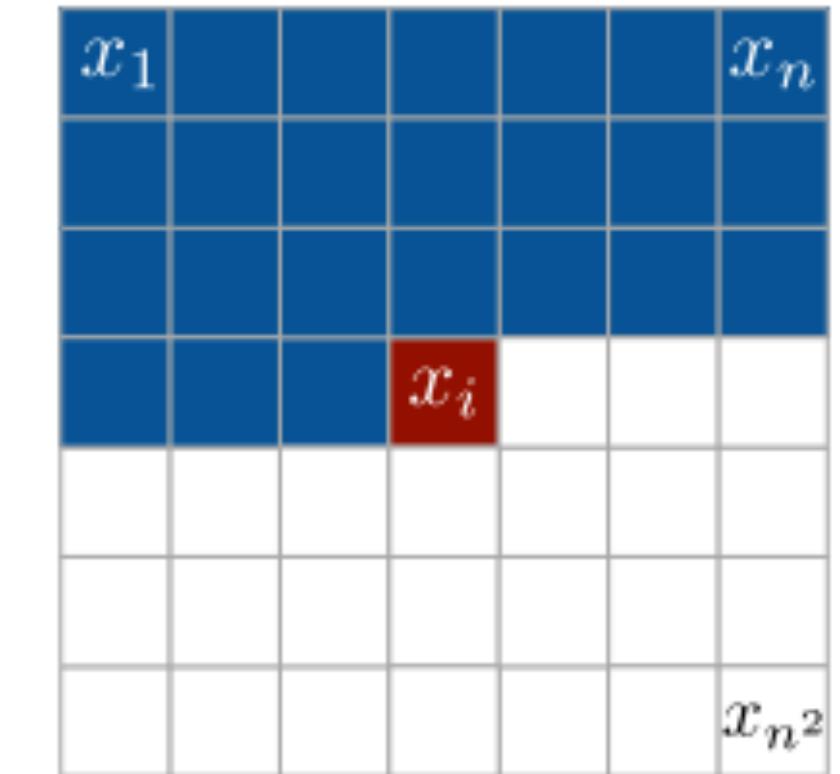
Context

PixelCNN

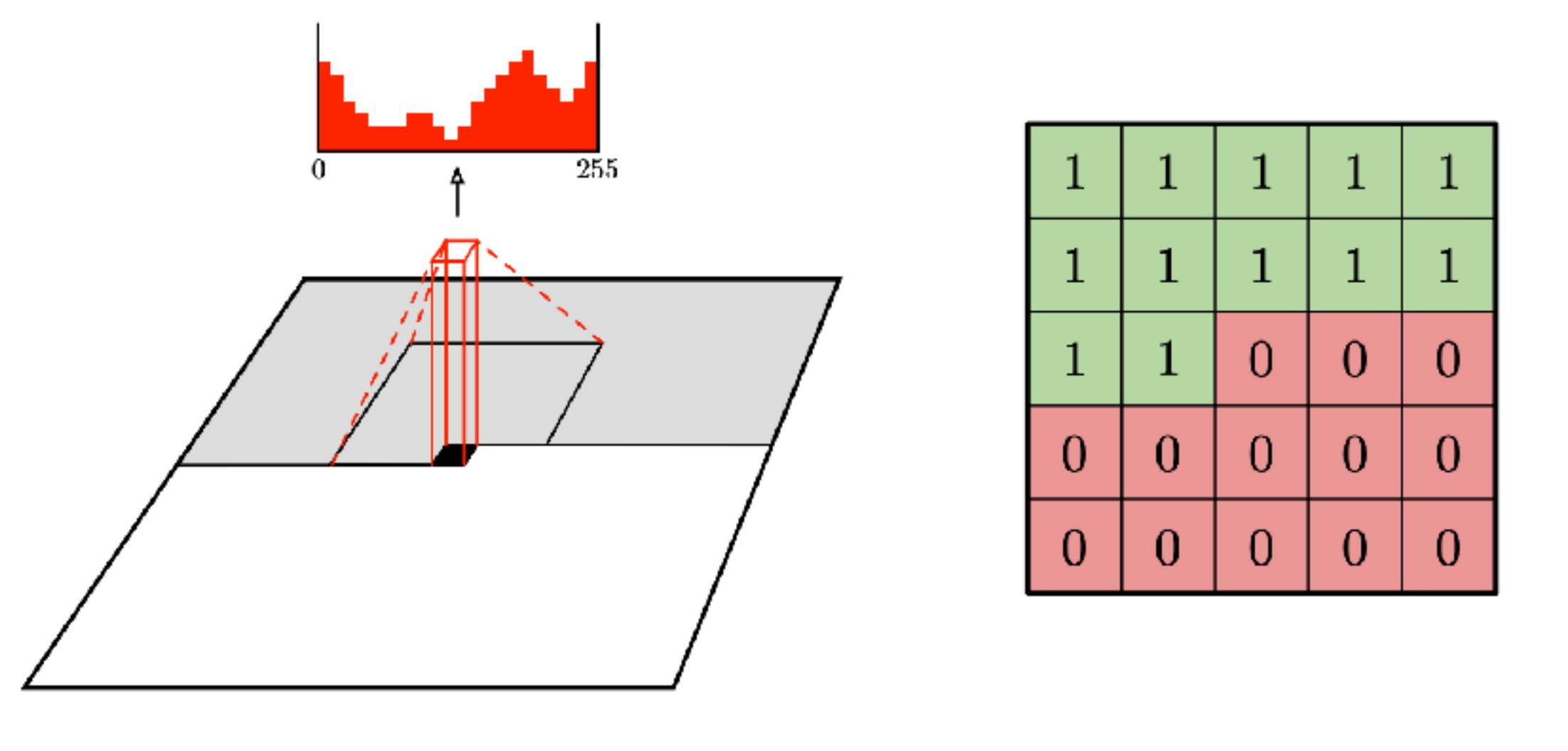
Chain rule: $p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$

вероятность i -го пикселя при условии $i-1$ предыдущих

моделируем нейросетью - CNN с масками



Context

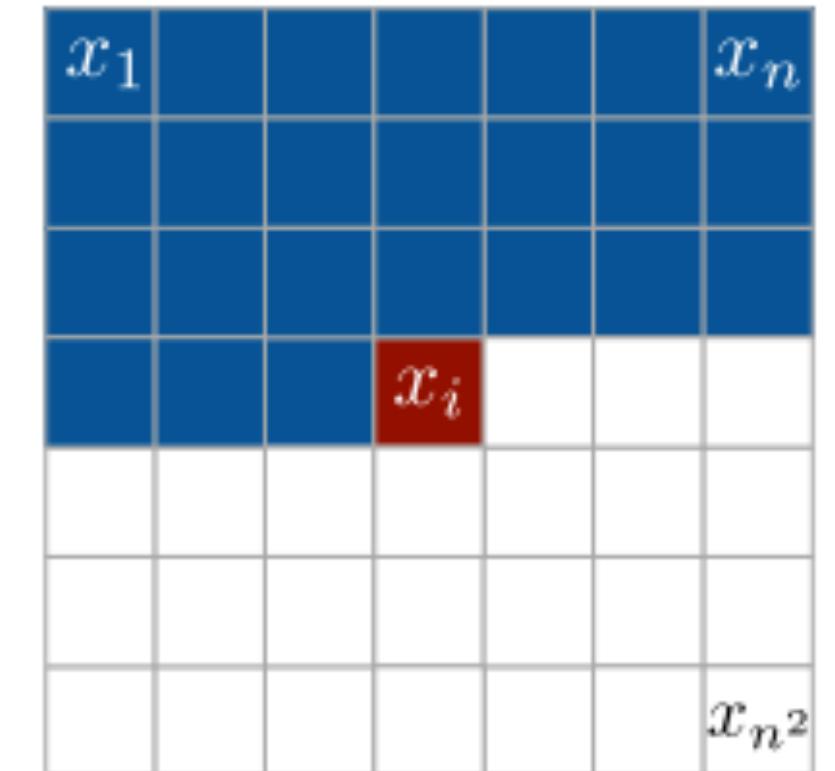


PixelCNN

Chain rule: $p(x) = \prod_{i=1}^n p(x_i | x_1, \dots, x_{i-1})$



вероятность i -го пикселя при условии $i-1$ предыдущих



+ хорошие семплы

+ доступ к $p(x)$

- долго (нужно несколько проходов)



African elephant



Sandbar

VQ-VAE

Background: AutoEncoder

Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности

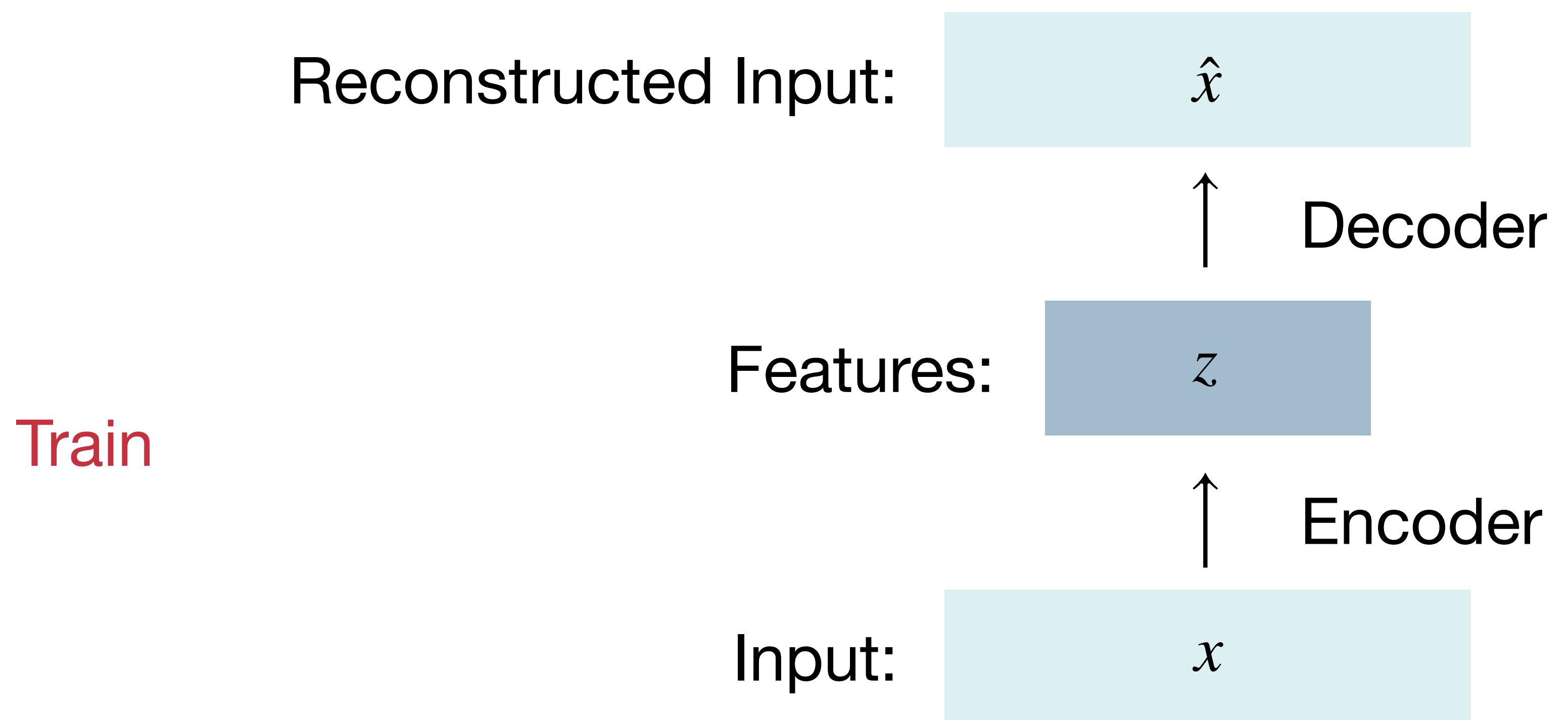
Background: AutoEncoder

Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности

Classic ML: РСА и другие методы понижения размерности
Не подходят для сложных данных (картинки)

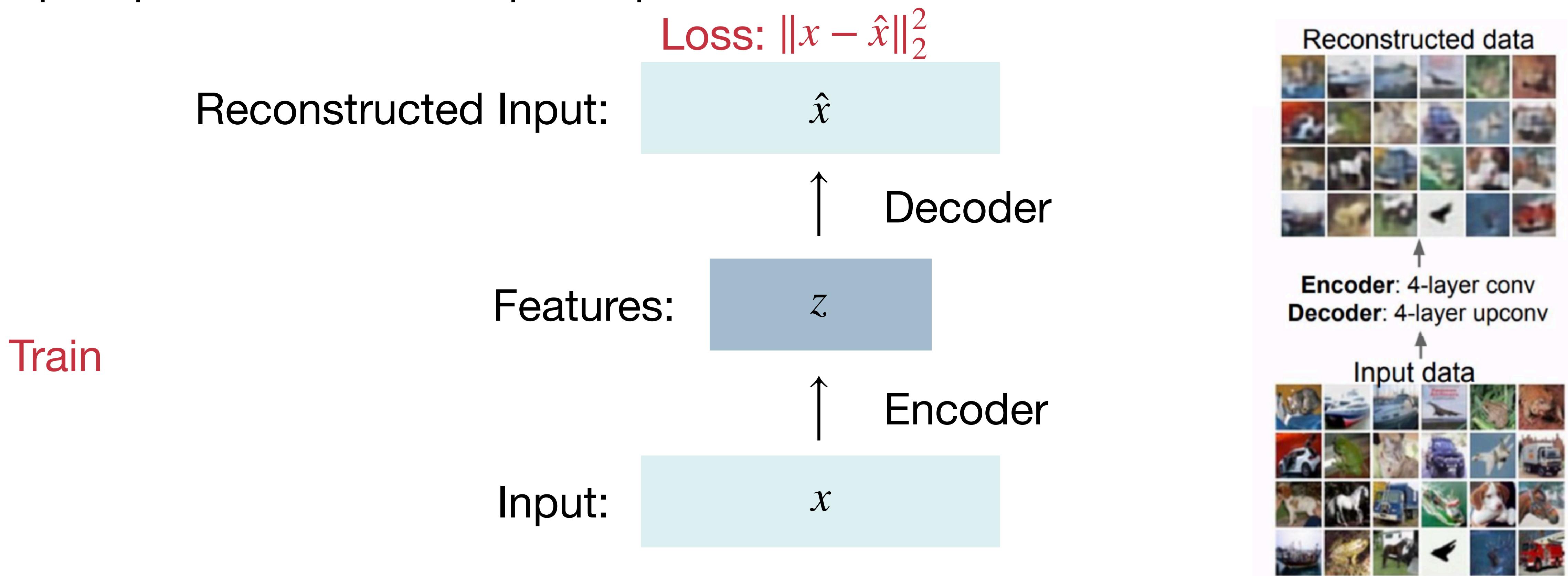
Background: AutoEncoder

Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности



Background: AutoEncoder

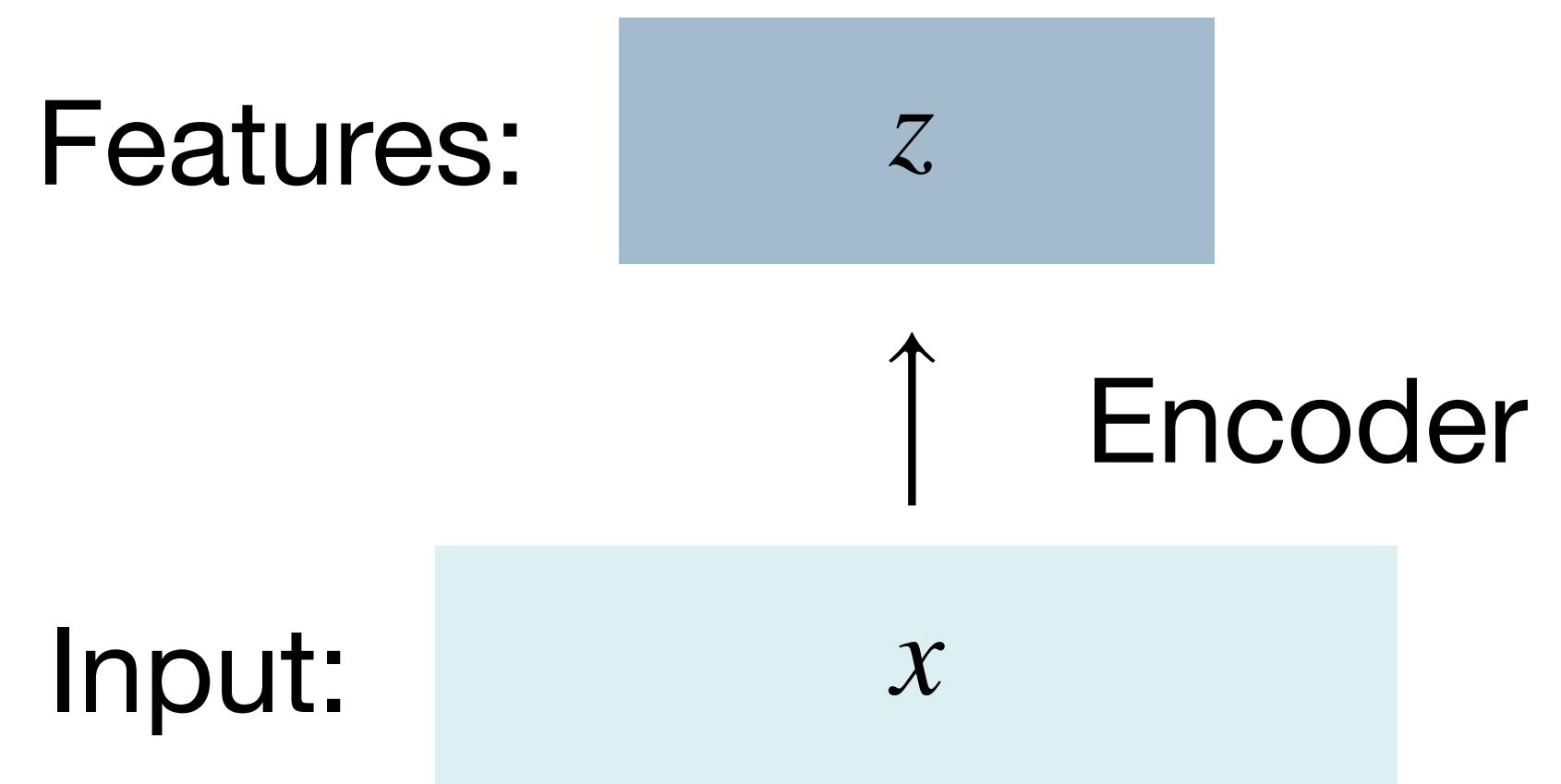
Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности



Background: AutoEncoder

Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности

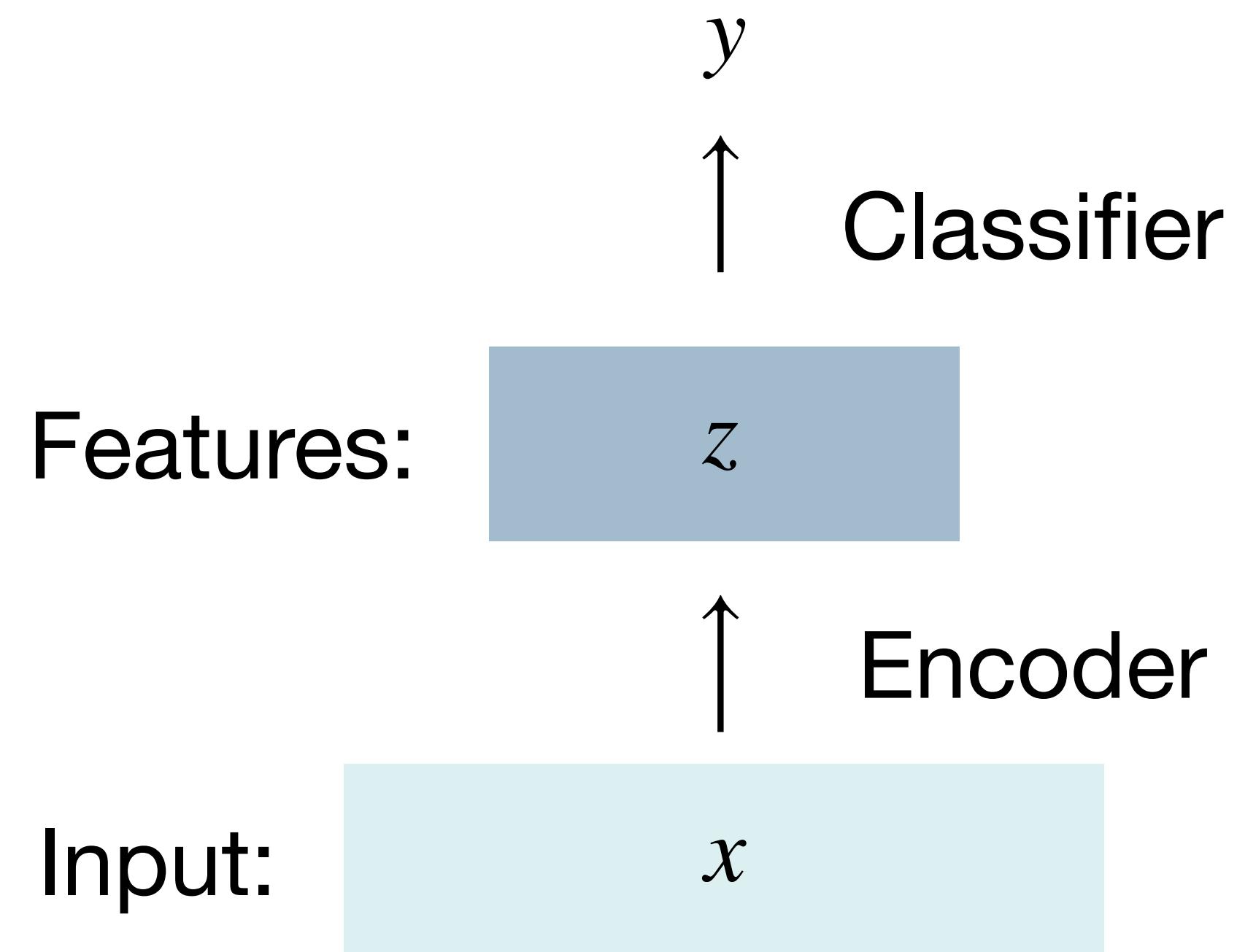
Inference



Background: AutoEncoder

Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности

Можно использовать для классификации и других задач

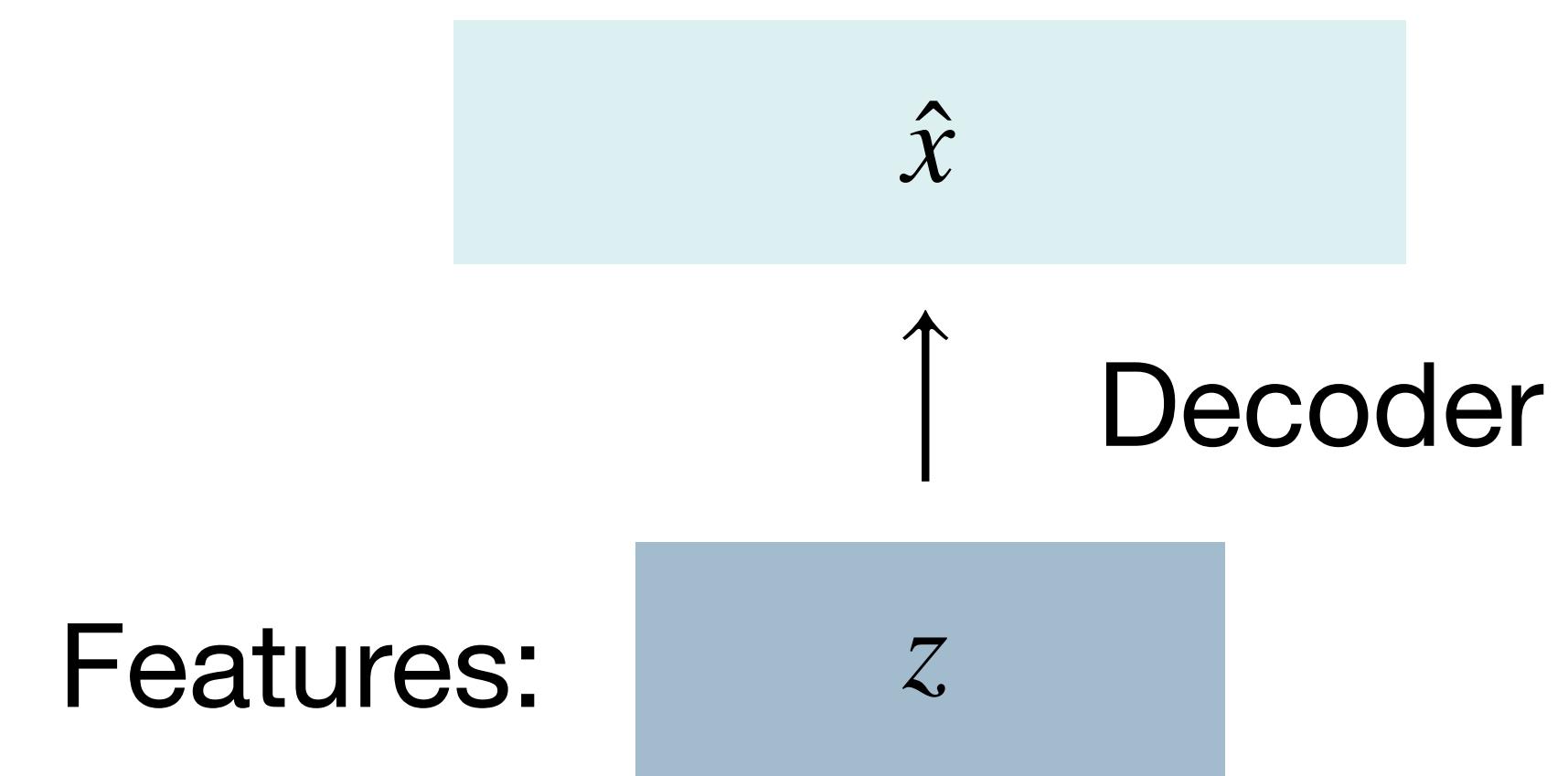


Transfer learning:
доучиваем с лоссом
для классификации

Background: AutoEncoder

Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности

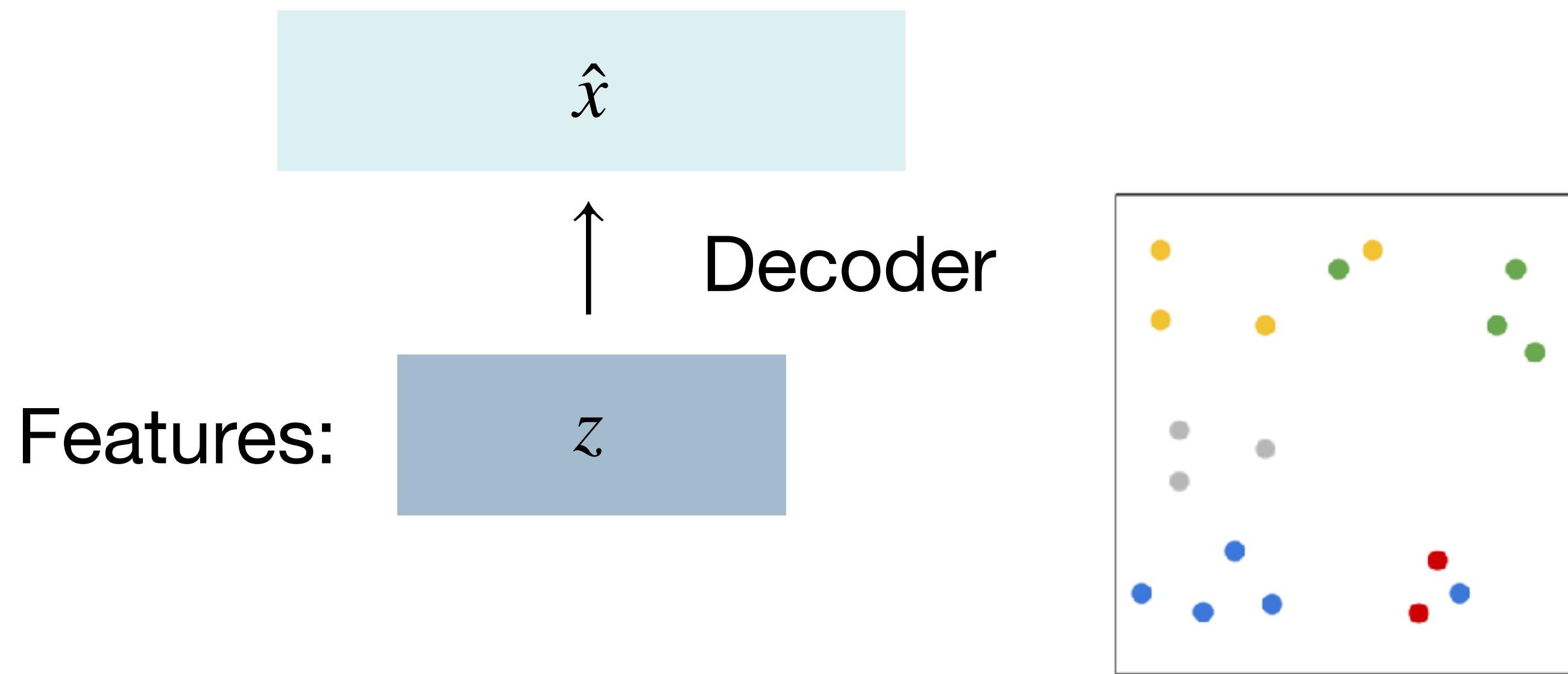
Как использовать для генерации?



Background: AutoEncoder

Задача: получить “скрытые” представления неразмеченных данных в пространстве меньшей размерности

Как использовать для генерации?

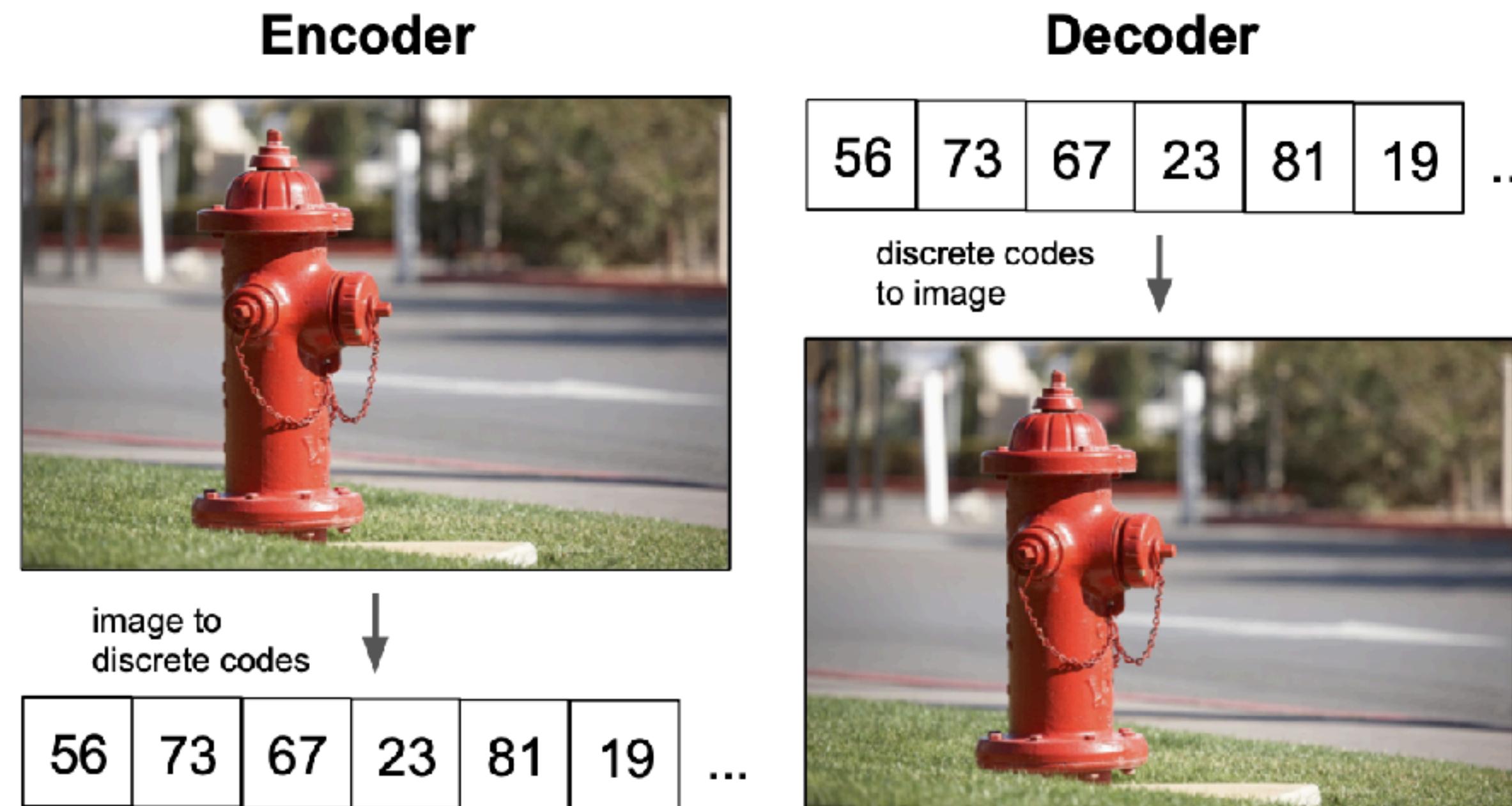


Пространство z может быть любым

VQ-VAE

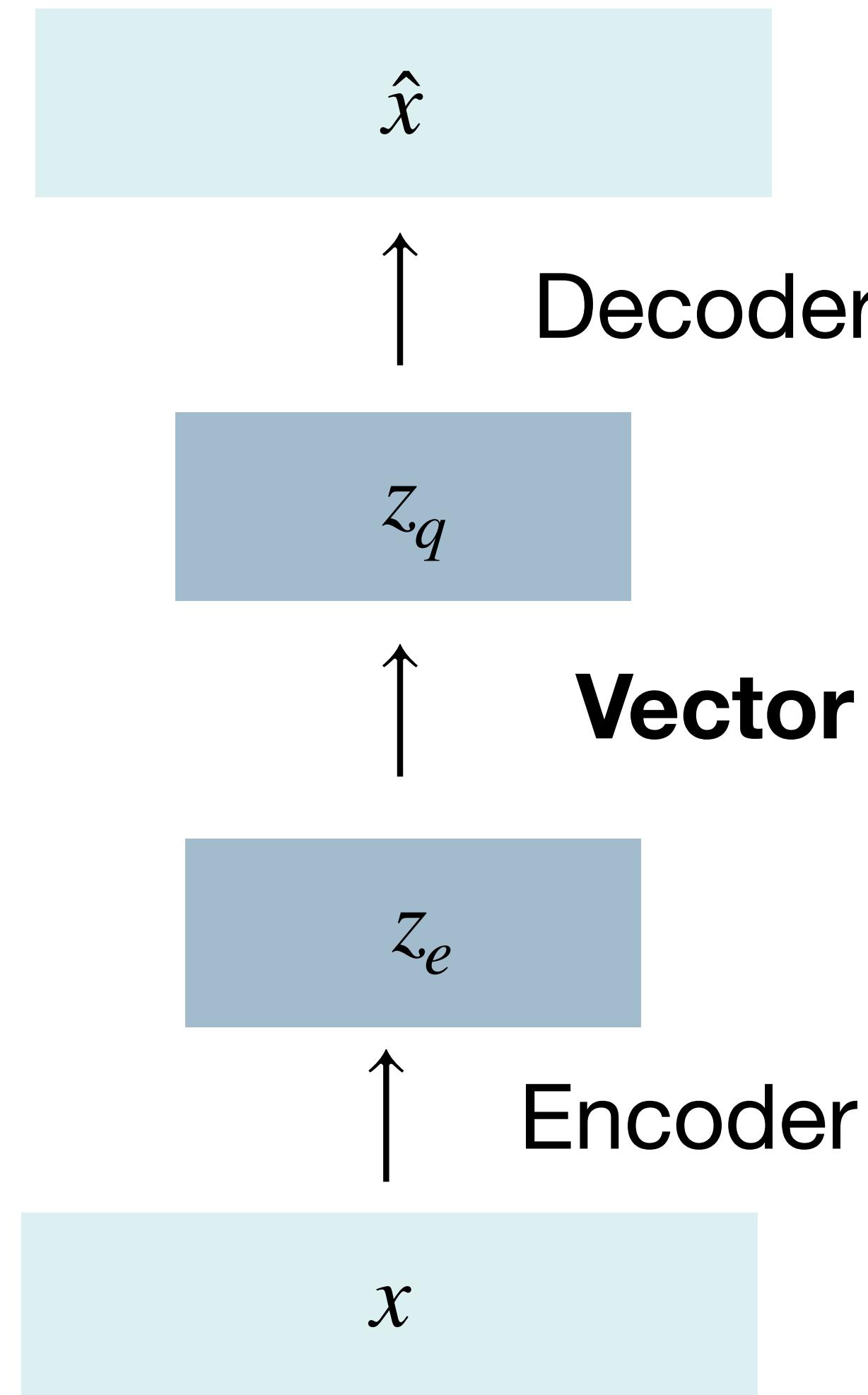
Идея: пусть скрытое пространство будет дискретным

Одна переменная - цвет, одна - форма, одна - тип объекта и т.д.



VQ-VAE

Reconstructed Input:



VQ-VAE

Reconstructed Input:

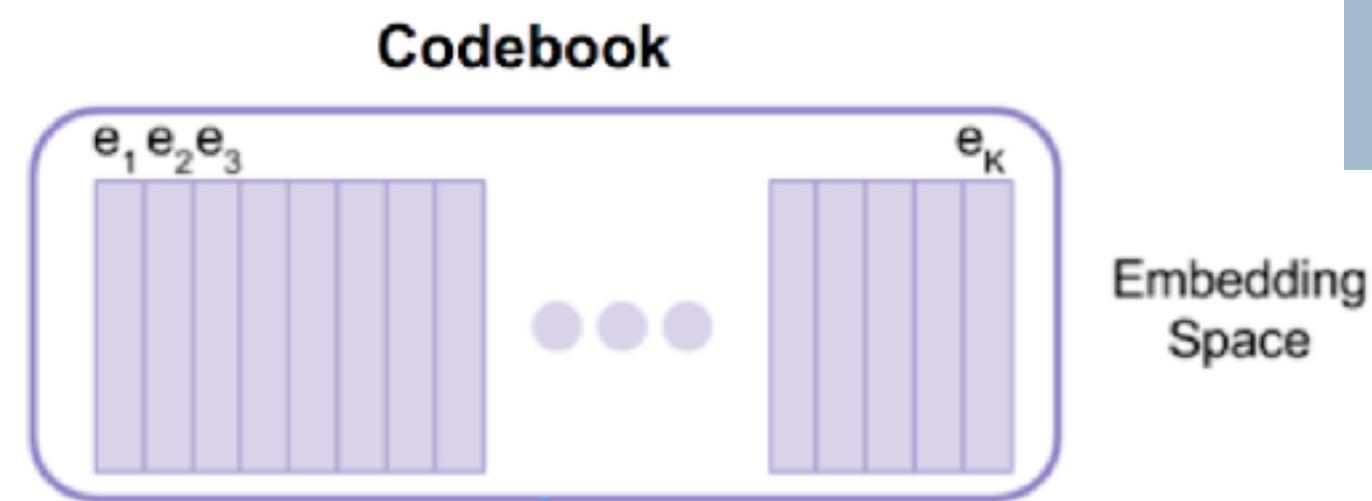
\hat{x}

Decoder

z_q

Vector Quantization Layer

$$z_q(x) = \operatorname{argmin}_i \|e_i - z_e(x)\|_2^2$$



z_e

Encoder

Input:

x

VQ-VAE

Loss: $\log(p(x|q(x)) + \|sg[z_e(x)] - e\|_2^2 + \beta\|z_e(x) - sg[e]\|_2^2$

Reconstructed Input:

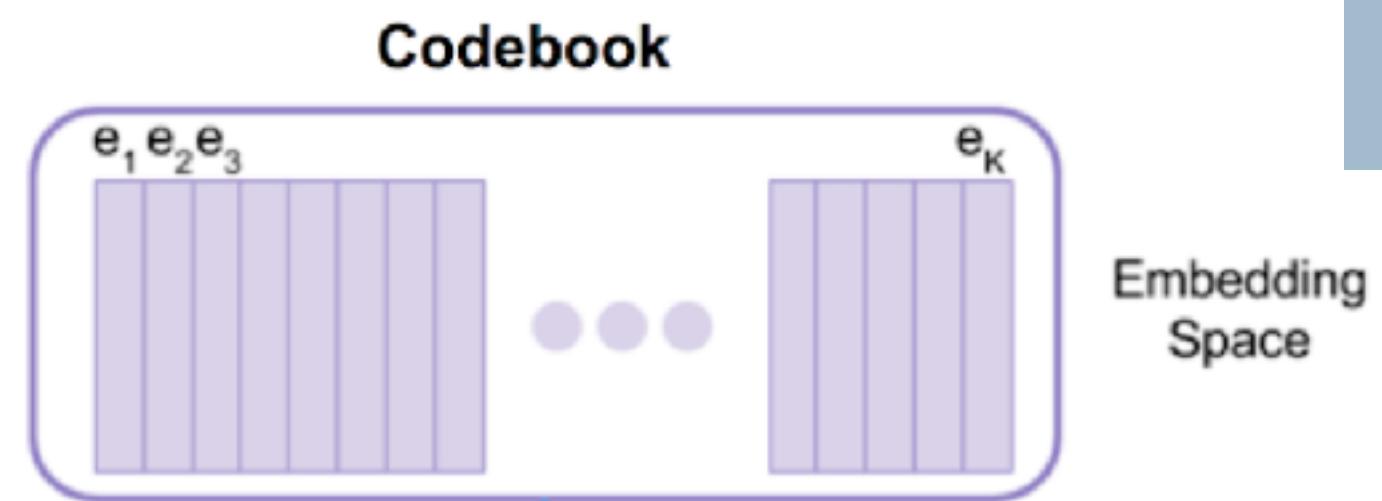
\hat{x}

Decoder

z_q

Vector Quantization Layer

$$z_q(x) = \operatorname{argmin}_i \|e_i - z_e(x)\|_2^2$$



Embedding Space

z_e

Encoder

Input:

x

VQ-VAE

Как использовать для генерации?

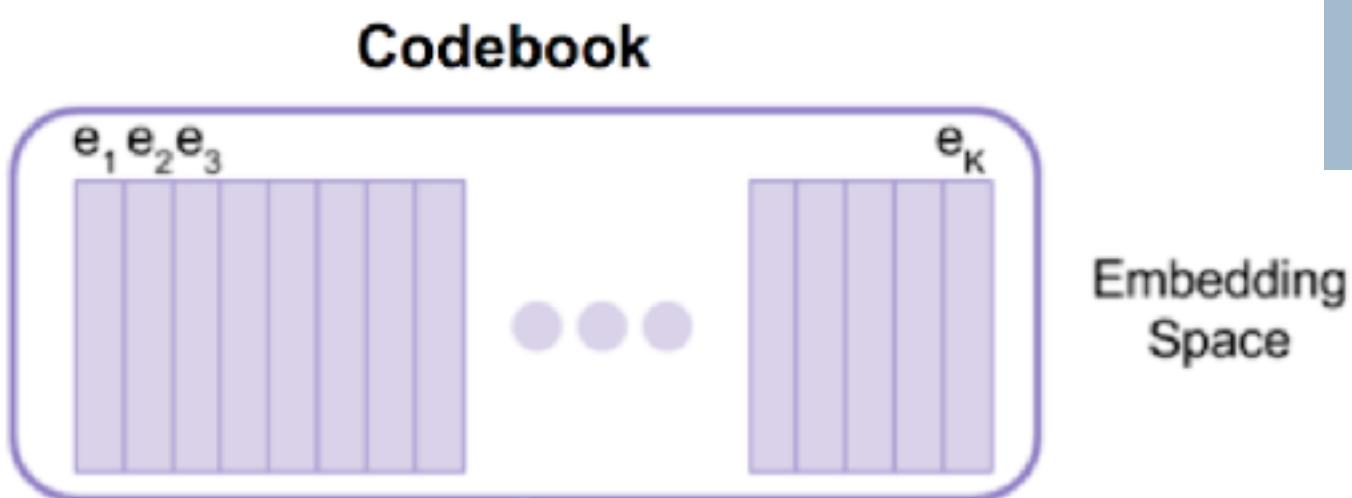
Reconstructed Input:

\hat{x}

Decoder

z_q

Vector Quantization Layer



$$z_q(x) = \operatorname{argmin}_i \|e_i - z_e(x)\|_2^2$$

z_e

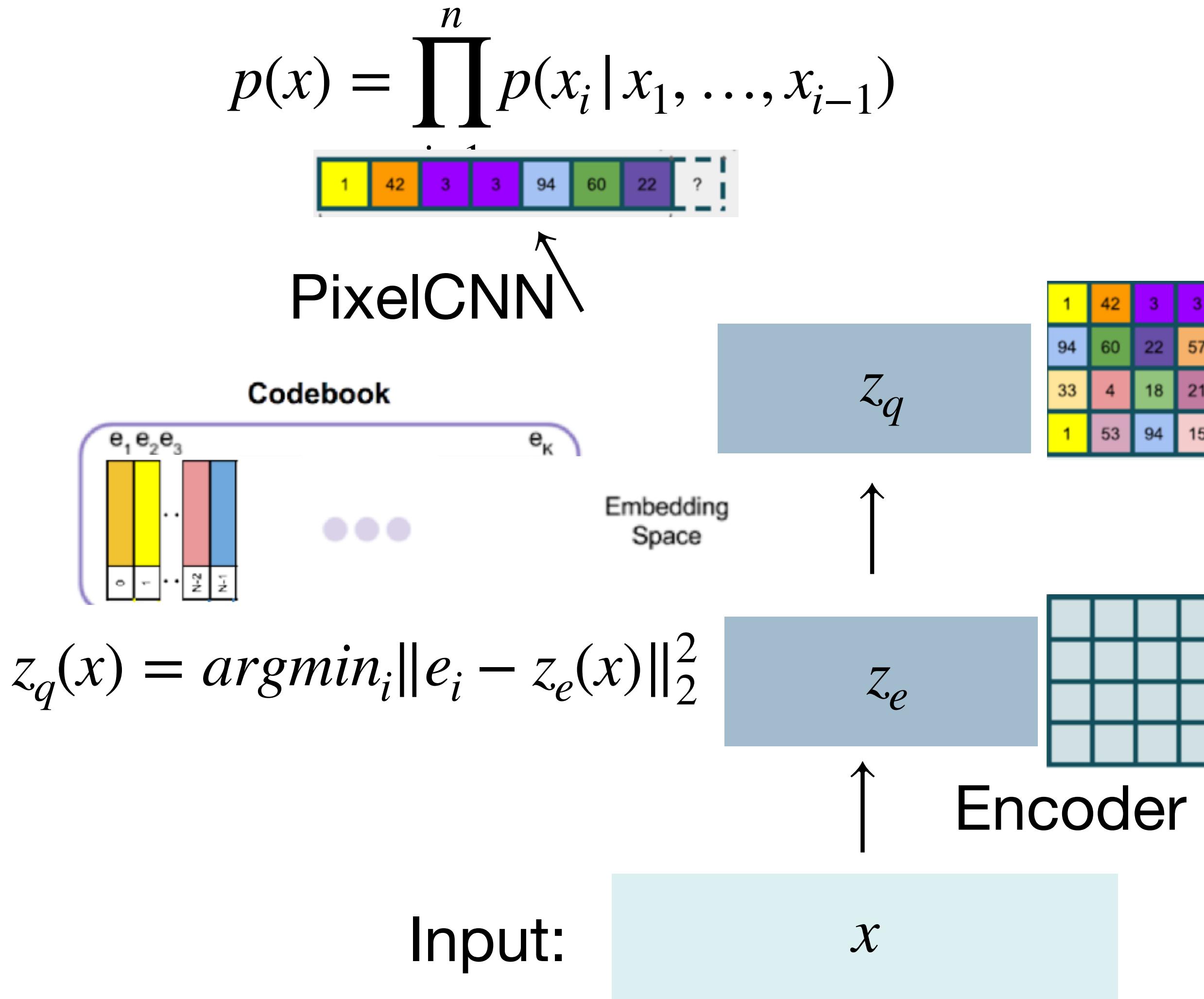
Encoder

Input:

x

VQ-VAE

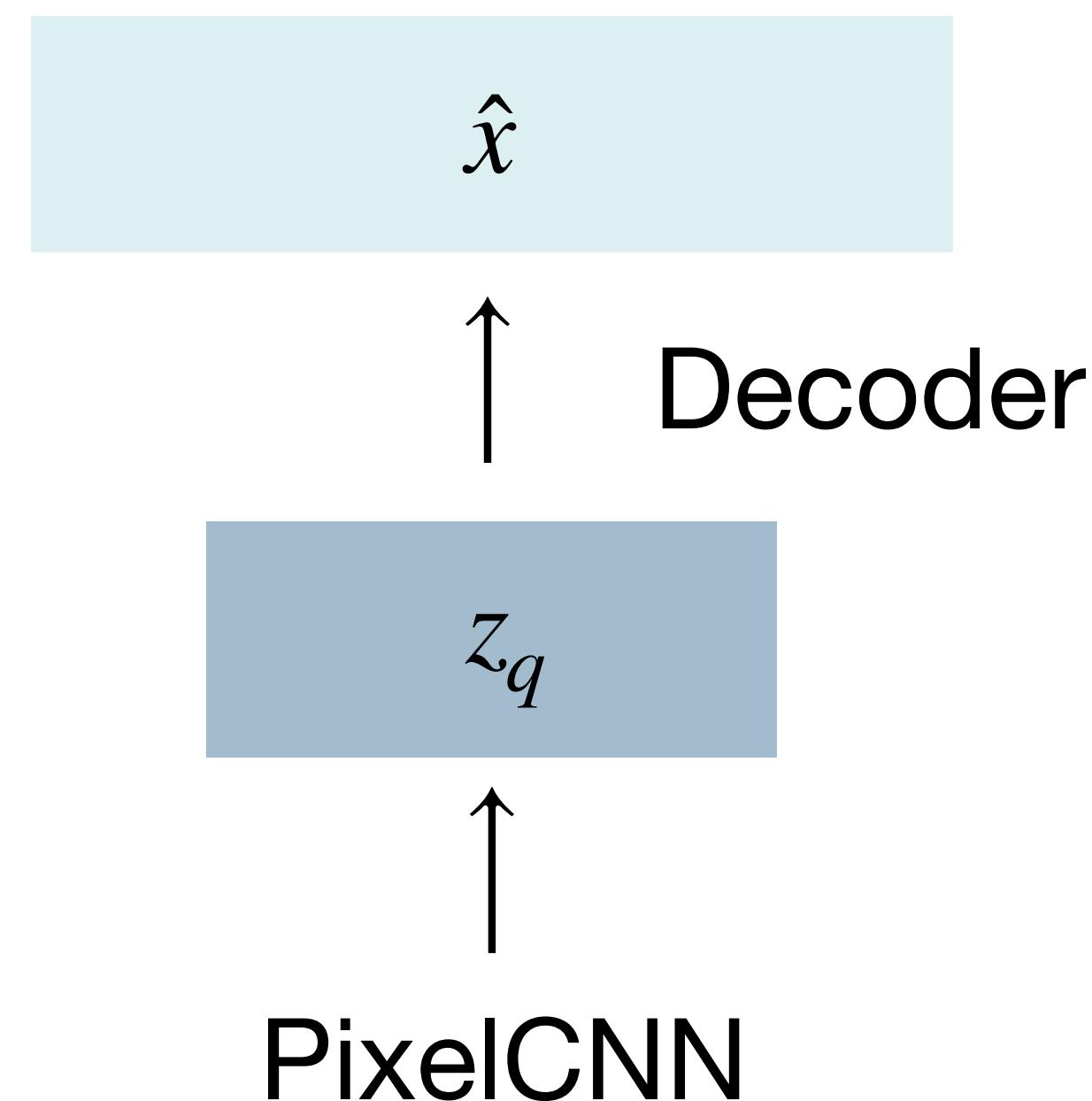
Как использовать для генерации?



VQ-VAE

Как использовать для генерации?

Можно сэмплировать z_q и декодировать



VQ-VAE

+ Качественные семплы

+ Разнообразные семплы

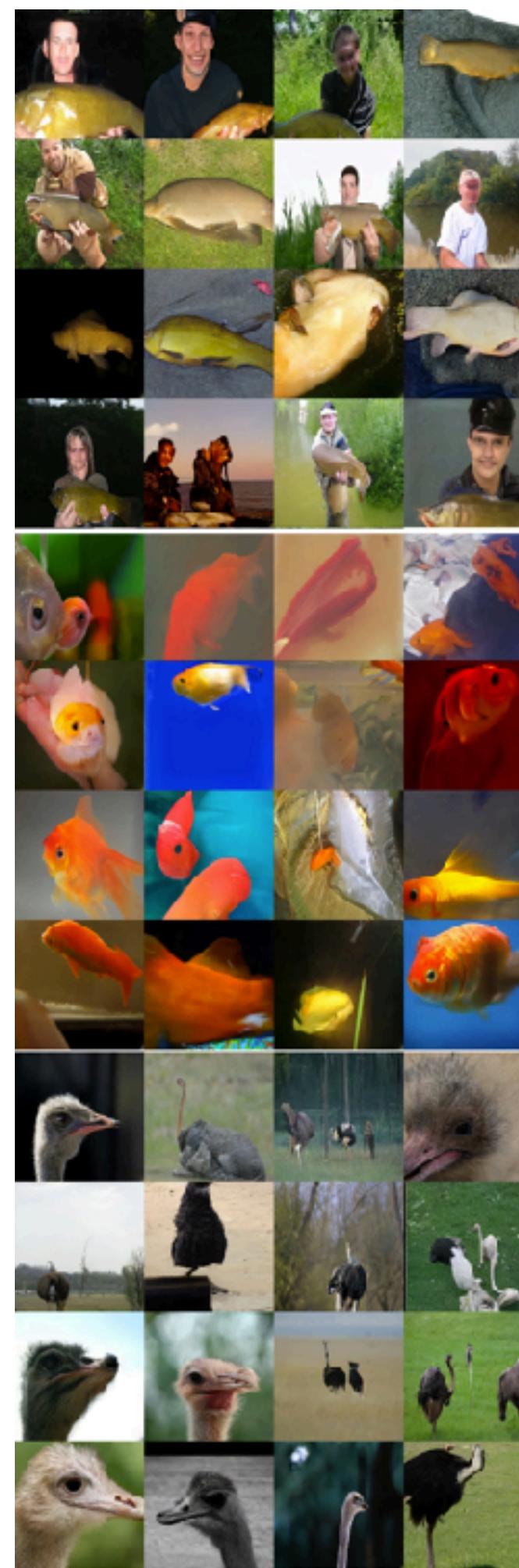
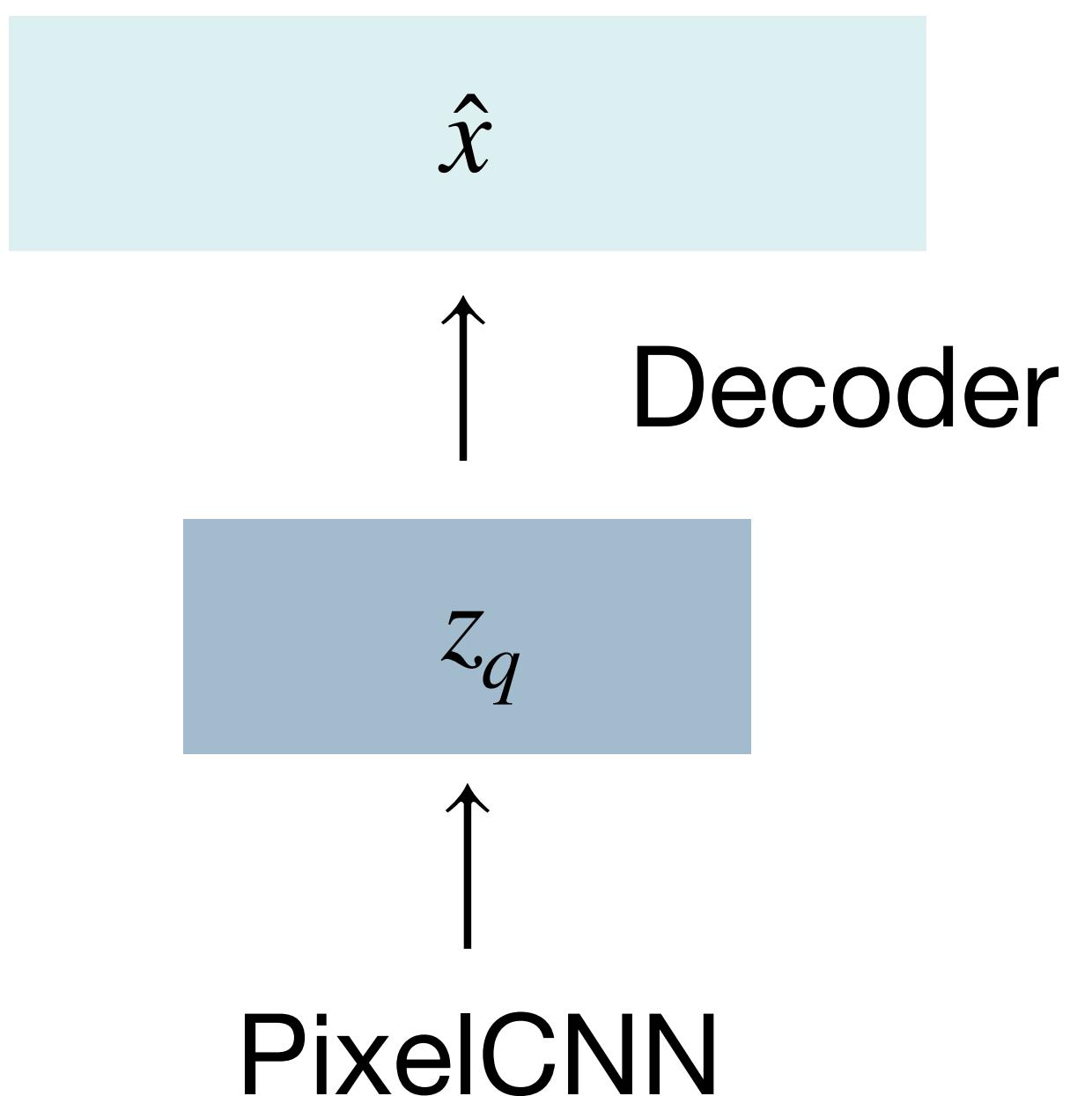
+ Можно использовать энкодер

+ Интерпретируемость

- Долго

Как использовать для генерации?

Можно семплировать z_q и декодировать



VQ-VAE

Как использовать для генерации?

Можно семплировать z_q и декодировать

+ Качественные семплы

+ Разнообразные семплы

+ Можно использовать энкодер

+ Интерпретируемость

- Долго

VAE - непрерывное пространство, другой лосс

GAN

GAN

Не нужно явно определять $p(x)$, но хотим уметь сэмплировать (implicit density estimation)

GAN

Не нужно явно определять $p(x)$, но хотим уметь семплировать (implicit density estimation)

Пространство $p(x)$ сложное, семплировать из него напрямую сложно

Идея: семплировать из простого (random noise), учить трансформацию

Input: random noise



x



Generator (neural network)

z

GAN

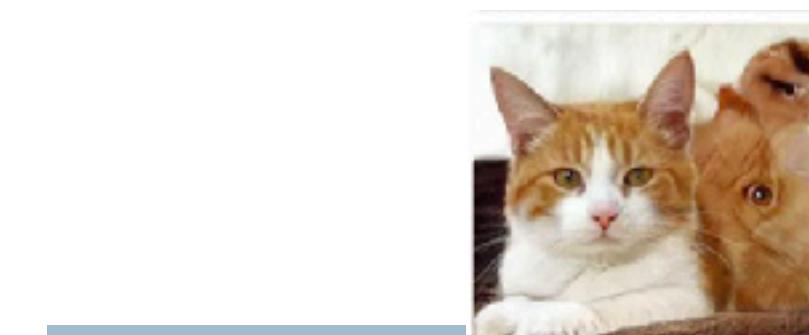
Не нужно явно определять $p(x)$, но хотим уметь семплировать (implicit density estimation)

Пространство $p(x)$ сложное, семплировать из него напрямую сложно

Идея: семплировать из простого (random noise), учить трансформацию

Как обучать?

Input: random noise



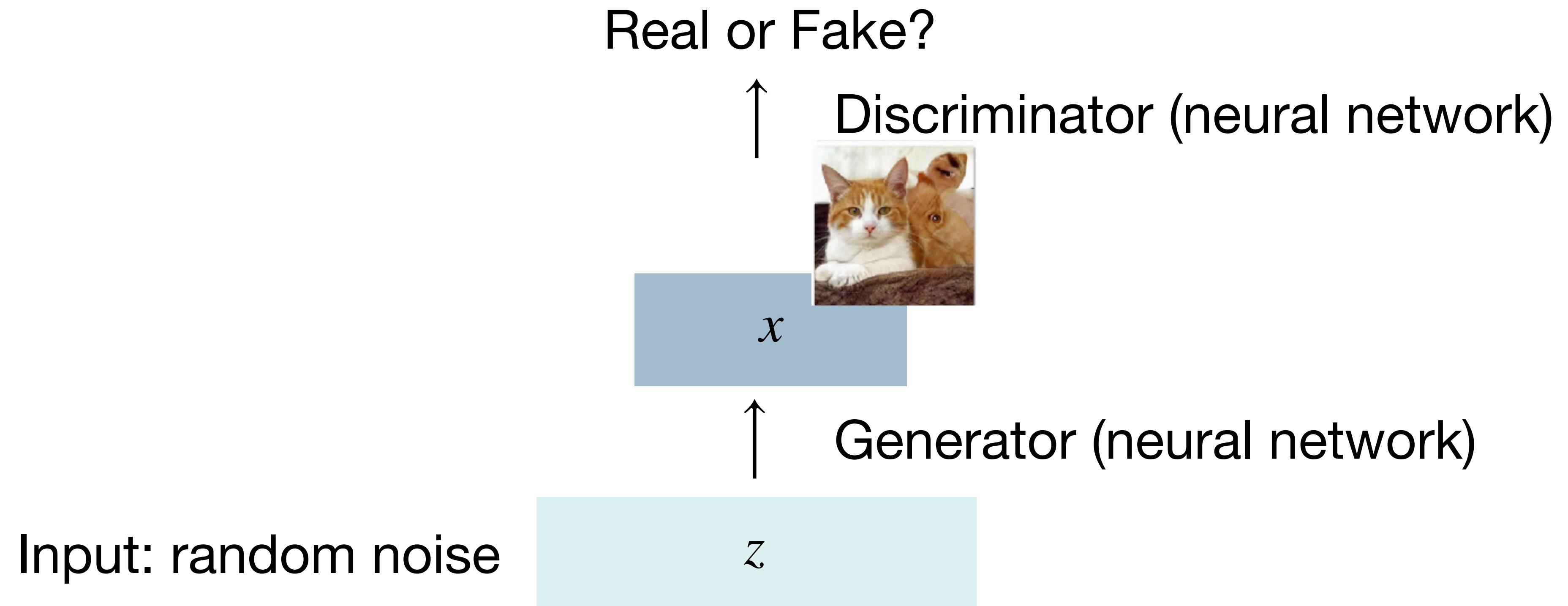
x



Generator (neural network)

z

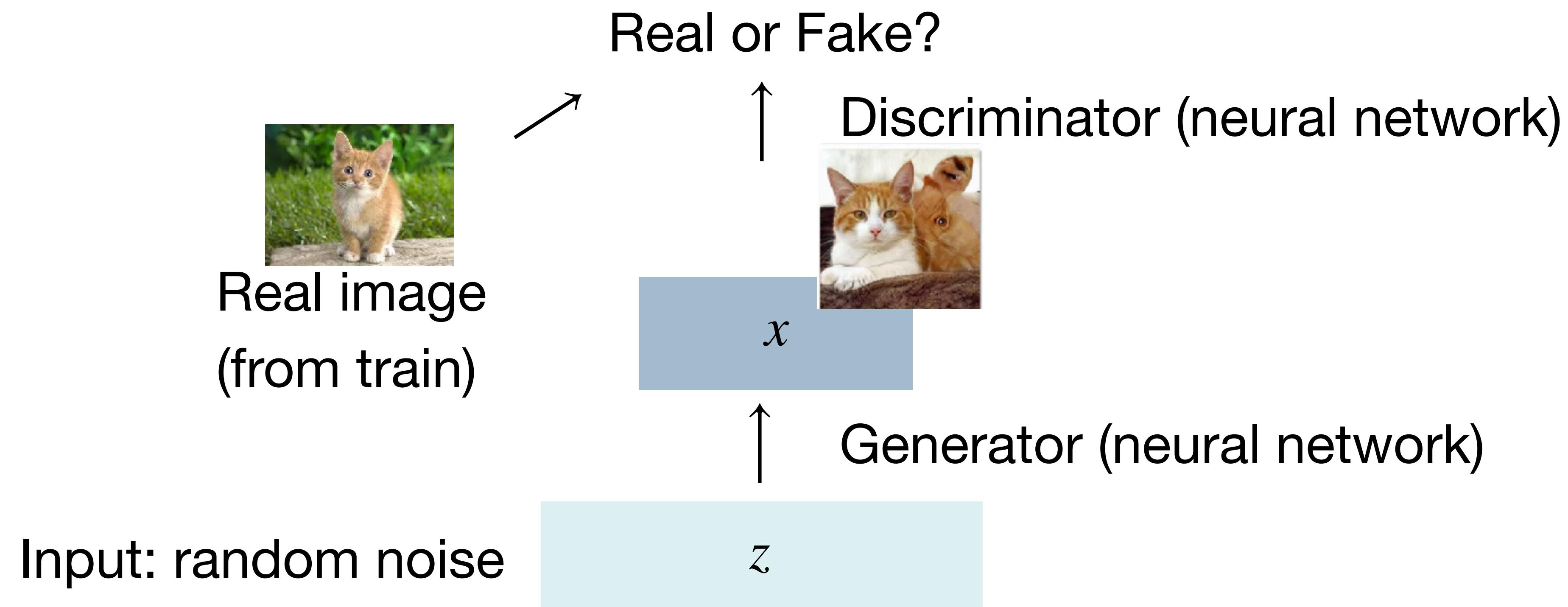
GAN



GAN

Discriminator: пытает различить реальные изображения и сгенерированные

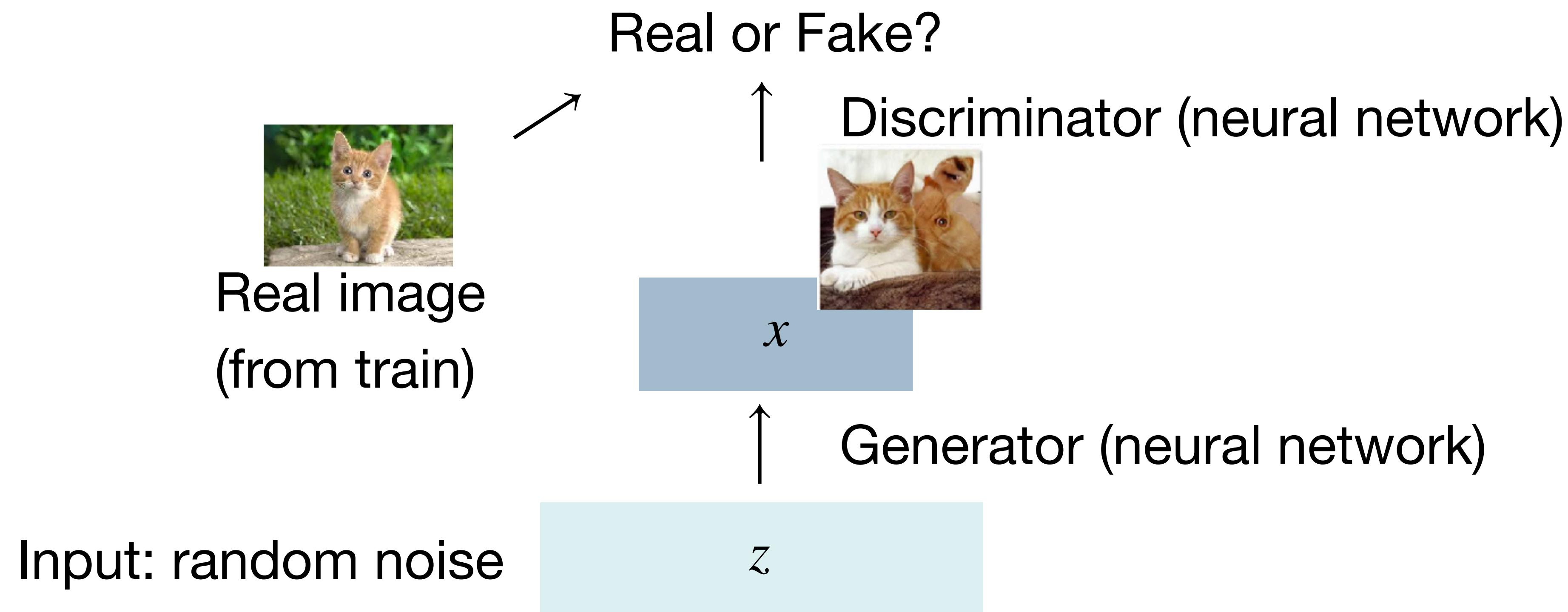
Generator: пытает обмануть дискриминатор и сгенерировать качественное изображение



GAN

Minimax game

$$\min_{\theta_G} \max_{\theta_D} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_D}(G_{\theta_G}(z)) \right) \right]$$

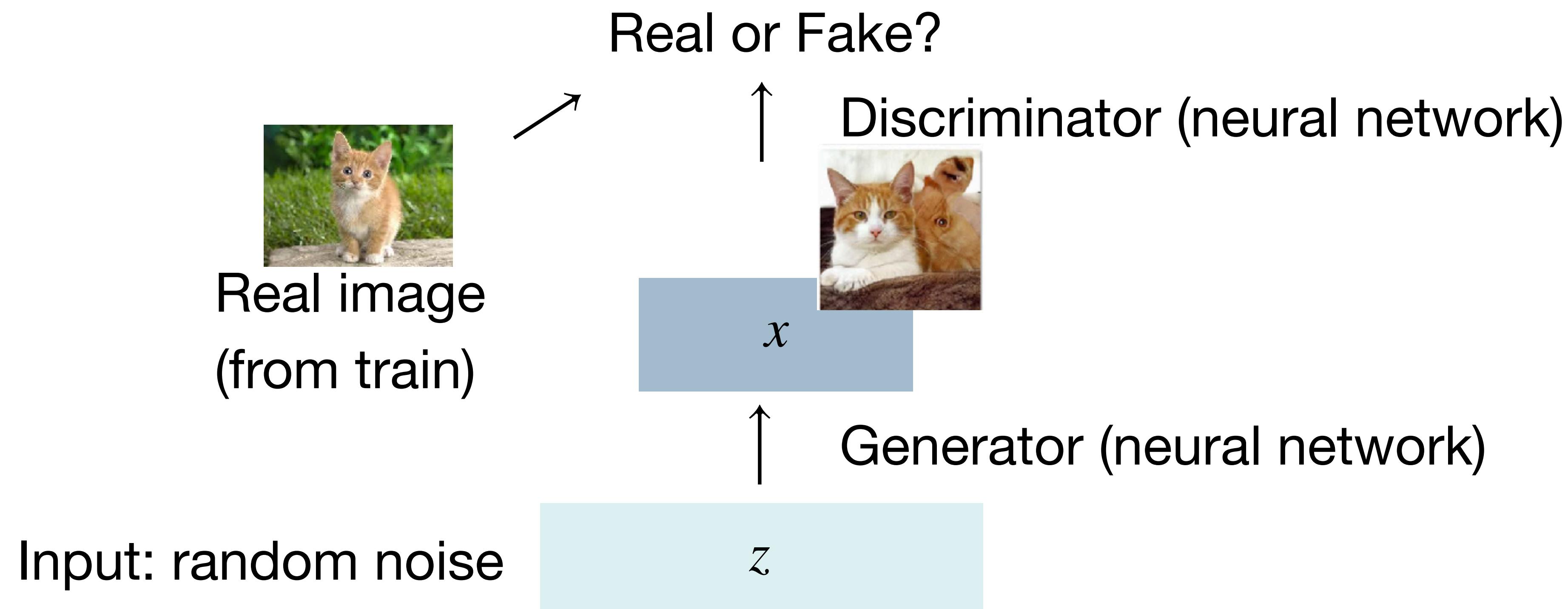


GAN

Minimax game

$$\min_{\theta_G} \max_{\theta_D} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_D}(G_{\theta_G}(z)) \right) \right]$$

Discriminator optimization - binary classification loss

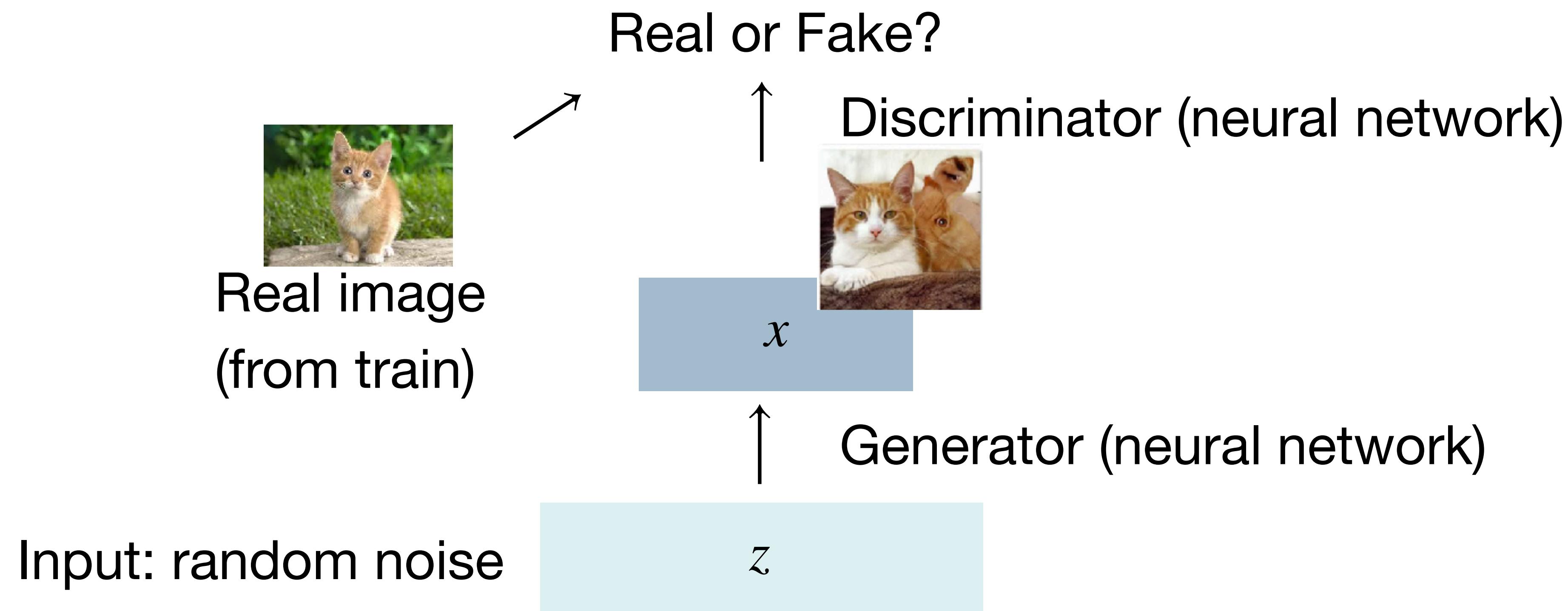


GAN

Minimax game

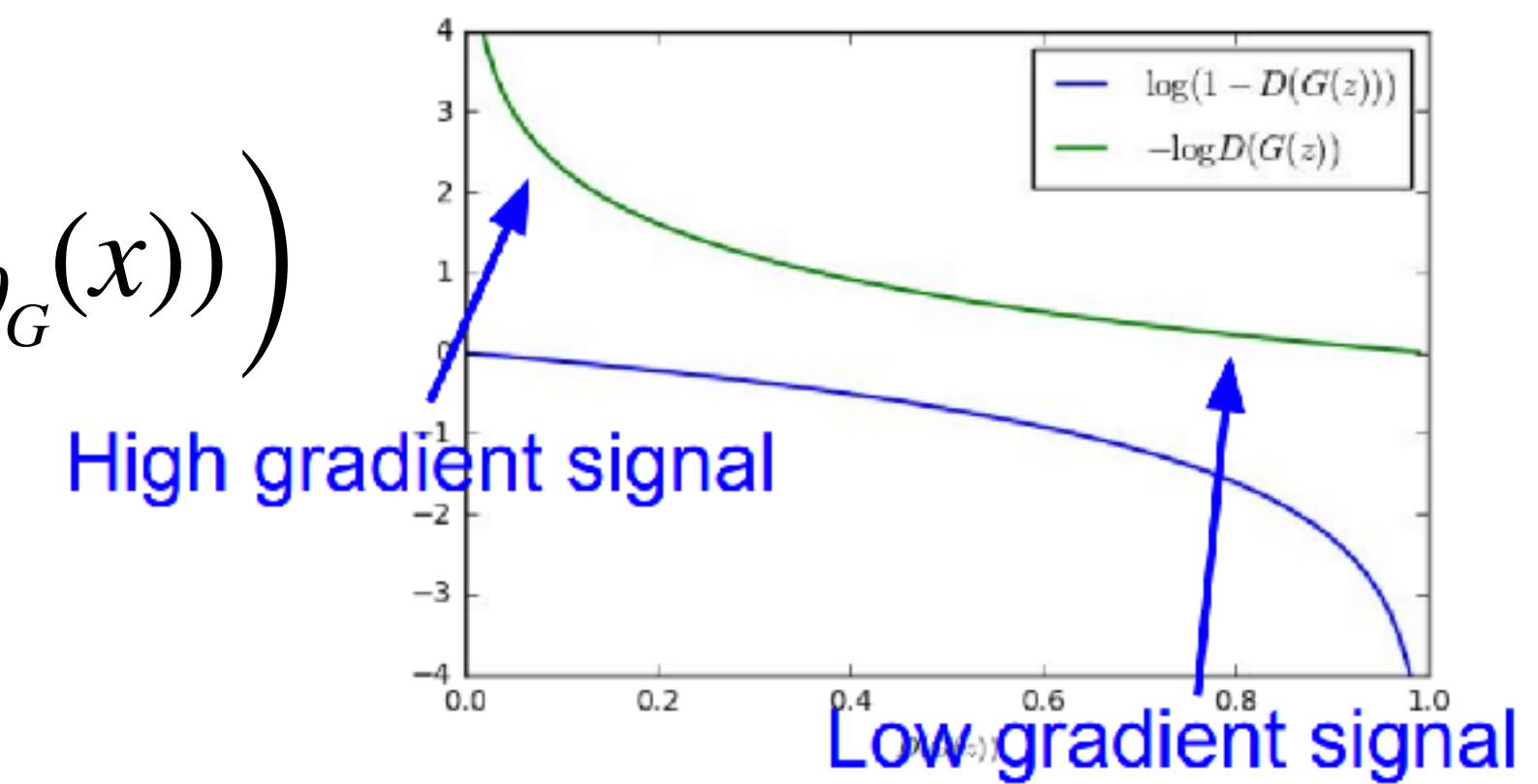
$$\min_{\theta_G} \max_{\theta_D} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_D}(G_{\theta_G}(z)) \right) \right]$$

Generator optimization - минимизируем вероятность распознания fake



GAN

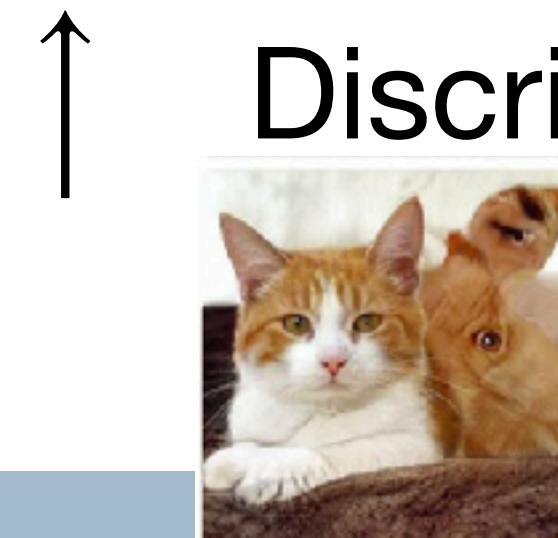
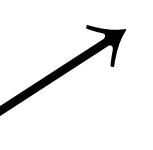
$$\min_{\theta_G} \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_D}(G_{\theta_G}(x)) \right) \rightarrow \max_{\theta_G} \mathbb{E}_{z \sim p(z)} \log \left(D_{\theta_D}(G_{\theta_G}(x)) \right)$$



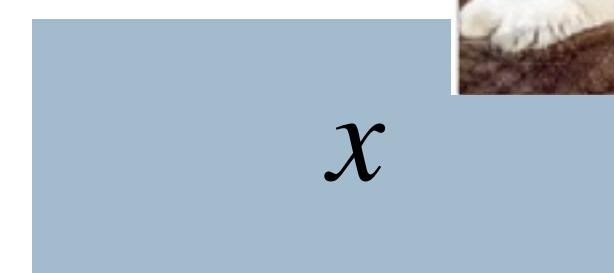
Real or Fake?



Real image
(from train)

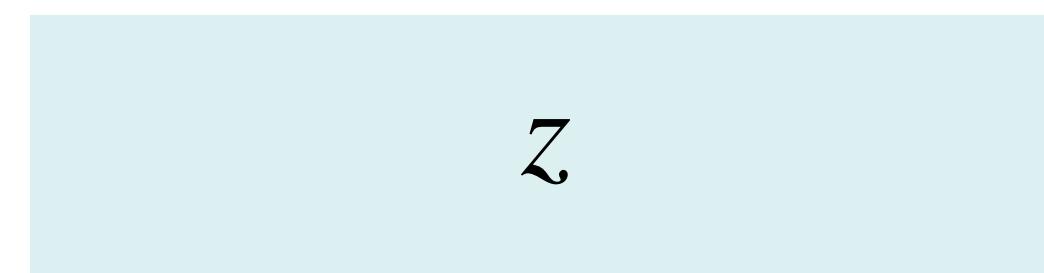


Discriminator (neural network)



x

Generator (neural network)



z

Input: random noise

GAN

По очереди оптимизируем генератор и дискриминатор

1. Discriminator update: семлируем генератором картинки, берем картинки из train

$$\max_{\theta_D} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_D}(x) + \mathbb{E}_{z \sim p(z)} \log \left(1 - D_{\theta_D}(G_{\theta_G}(z)) \right) \right]$$

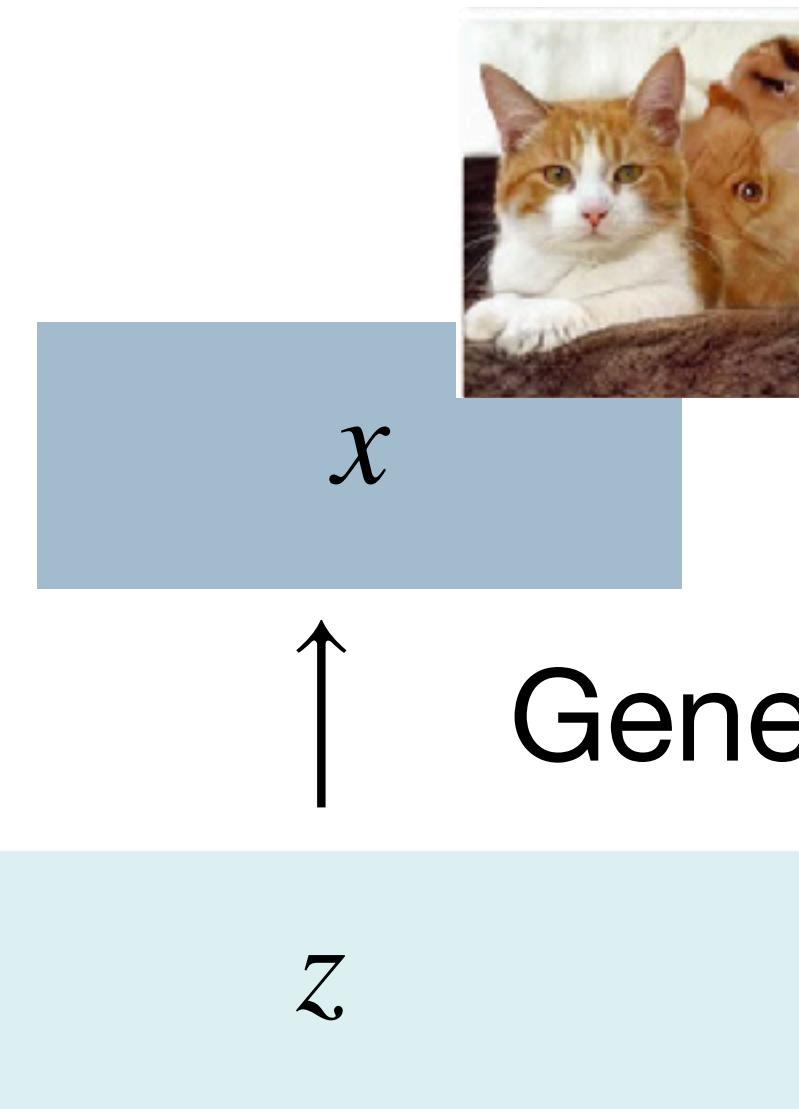
2. Generator update: семлируем генератором картинки

$$\max_{\theta_G} \mathbb{E}_{z \sim p(z)} \log \left(D_{\theta_D}(G_{\theta_G}(z)) \right)$$

GAN

Inference

Input: random noise



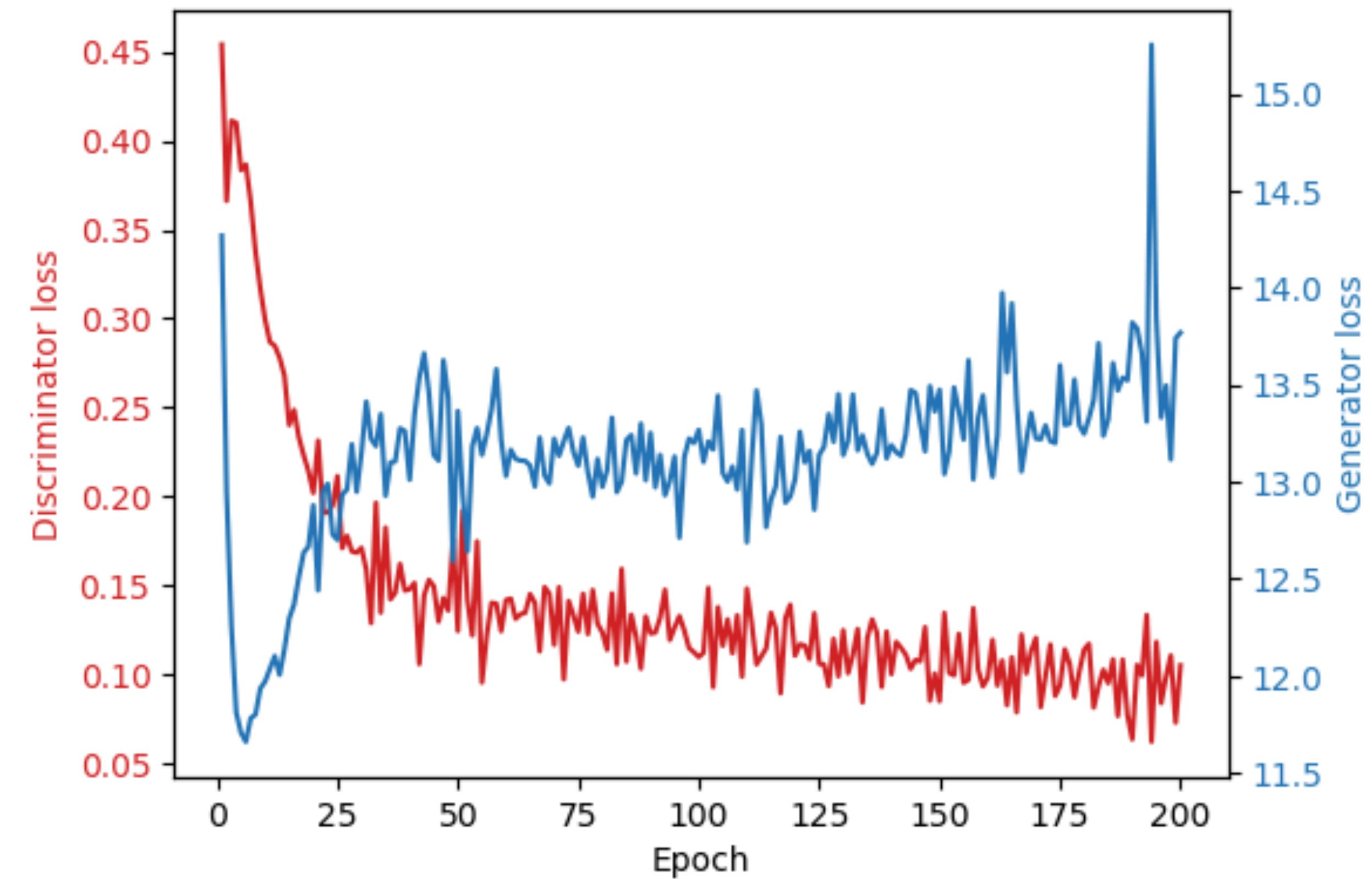
GAN



BigGAN samples

GAN

- + Очень качественные семплы
- + Быстрое сэмплирование
- Проблемы с разнообразием
- Сложно обучать



Метрики качества: IS и FID

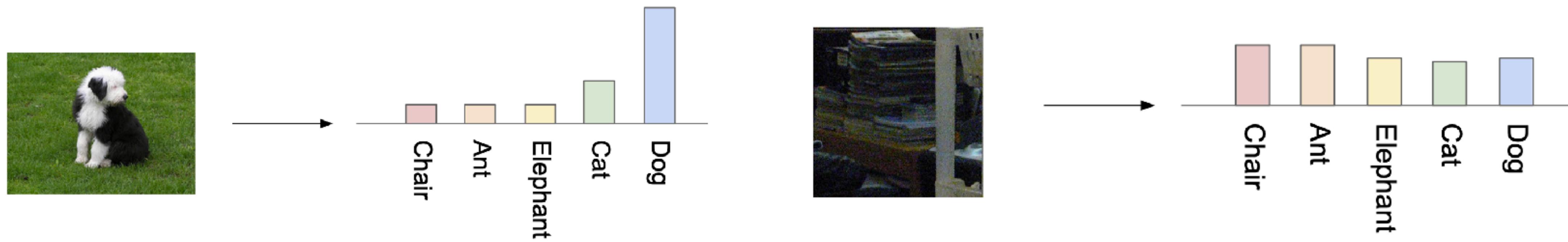
Идея: если картинка качественная, она будет хорошо классифицирована
=> подаем сгенерированные картинки на вход pre-trained Inception

Метрики качества: IS и FID

Идея: если картинка качественная, она будет хорошо классифицирована

=> подаем сгенерированные картинки на вход pre-trained Inception

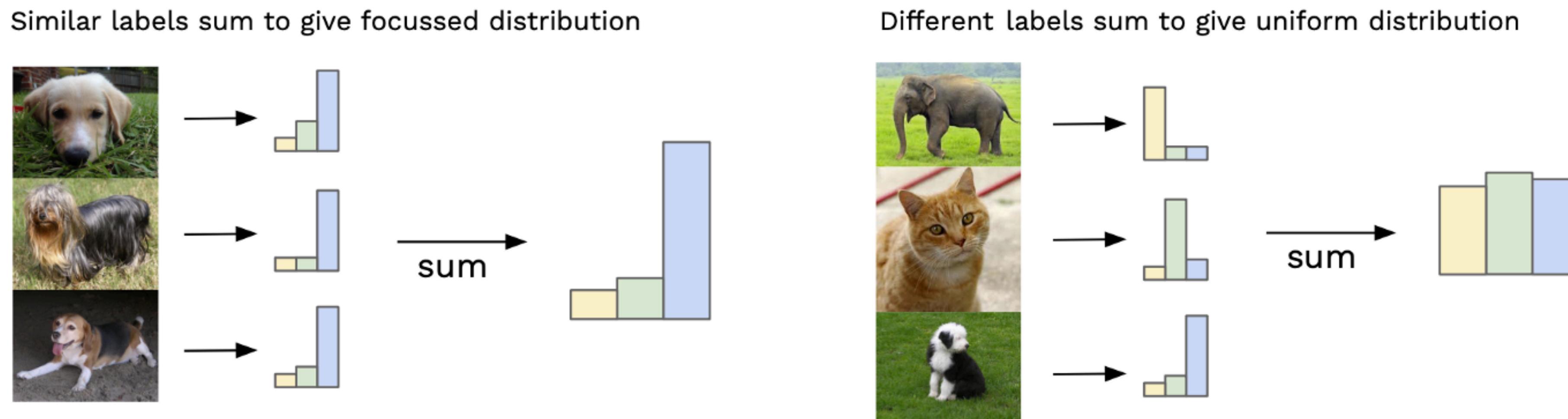
Inception score: оцениваем распределение **меток на выходе**



Метрики качества: IS и FID

Идея: если картинка качественная, она будет хорошо классифицирована
=> подаем сгенерированные картинки на вход pre-trained Inception

Inception score: оцениваем распределение **меток на выходе**



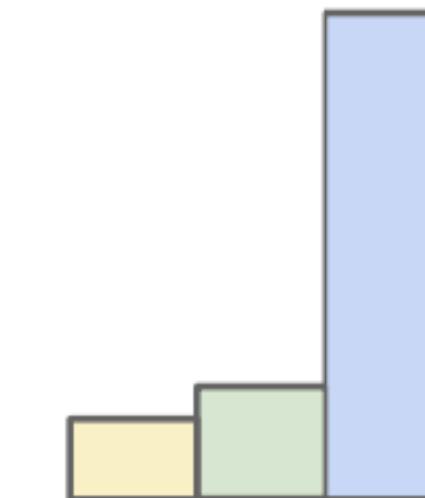
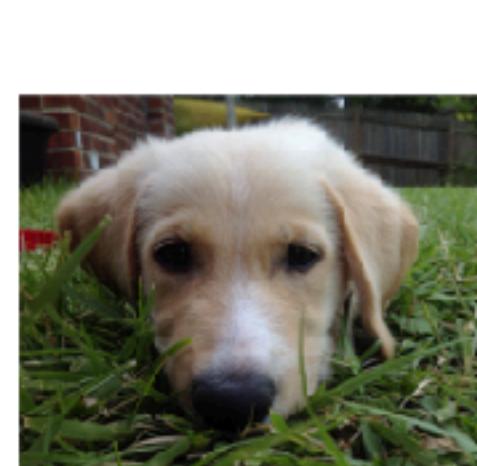
Метрики качества: IS и FID

Идея: если картинка качественная, она будет хорошо классифицирована

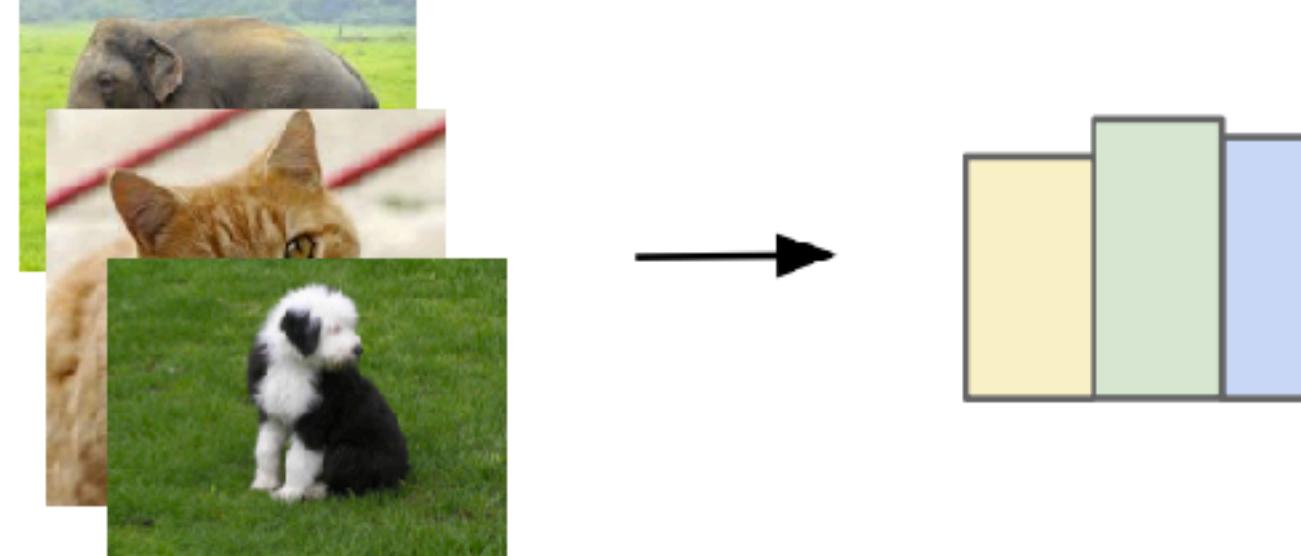
=> подаем сгенерированные картинки на вход pre-trained Inception

Inception score: оцениваем распределение **меток на выходе**

Сравниваем с маргинальным распределением (по всем картинкам)



Ideal label distribution



Ideal marginal distribution

Метрики качества: IS и FID

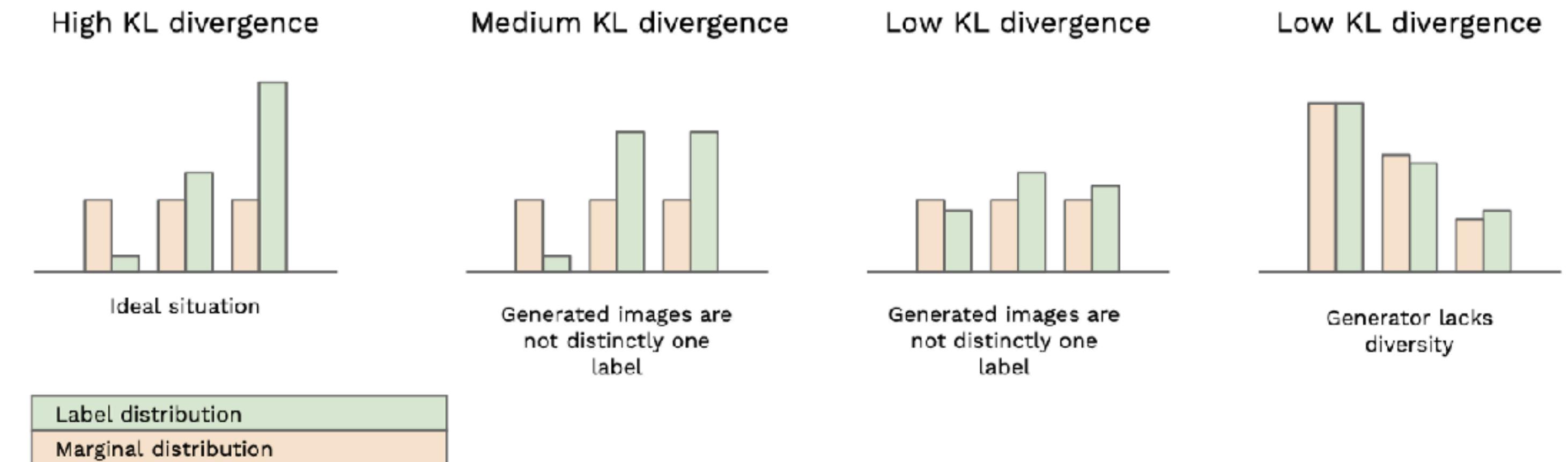
Идея: если картинка качественная, она будет хорошо классифицирована

=> подаем сгенерированные картинки на вход pre-trained Inception

Inception score: оцениваем распределение **меток на выходе**

Сравниваем с маргинальным распределением (по всем картинкам)

$$KL(P || Q) = - \sum_x P(x) \log \frac{Q(X)}{P(x)}$$



Больше - лучше

Метрики качества: IS и FID

Идея: если картинка качественная, она будет хорошо классифицирована

=> подаем сгенерированные картинки на вход pre-trained Inception

FID score: оцениваем распределение **feature vectors** (в конце Inception сети)

Метрики качества: IS и FID

Идея: если картинка качественная, она будет хорошо классифицирована

=> подаем сгенерированные картинки на вход pre-trained Inception

FID score: оцениваем распределение **feature vectors** (в конце Inception сети)

Считаем **среднее и матрицу ковариации** для real и fake

(из данных и генератора)

Метрики качества: IS и FID

Идея: если картинка качественная, она будет хорошо классифицирована

=> подаем сгенерированные картинки на вход pre-trained Inception

FID score: оцениваем распределение **feature vectors** (в конце Inception сети)

Считаем **среднее и матрицу ковариации** для real и fake

(из данных и генератора)

$$FID = \|\mu_r - \mu_f\|_2^2 + \text{tr}(\Sigma_r + \Sigma_f - 2\sqrt{\Sigma_r \Sigma_f})$$

- расстояние между двумя многомерными нормальными распределениями меньше - лучше

Метрики качества: IS и FID

Идея: если картинка качественная, она будет хорошо классифицирована

=> подаем сгенерированные картинки на вход pre-trained Inception

Inception score

- хуже подходит, если классы не из ImageNet (лица и т.п.)

FID score

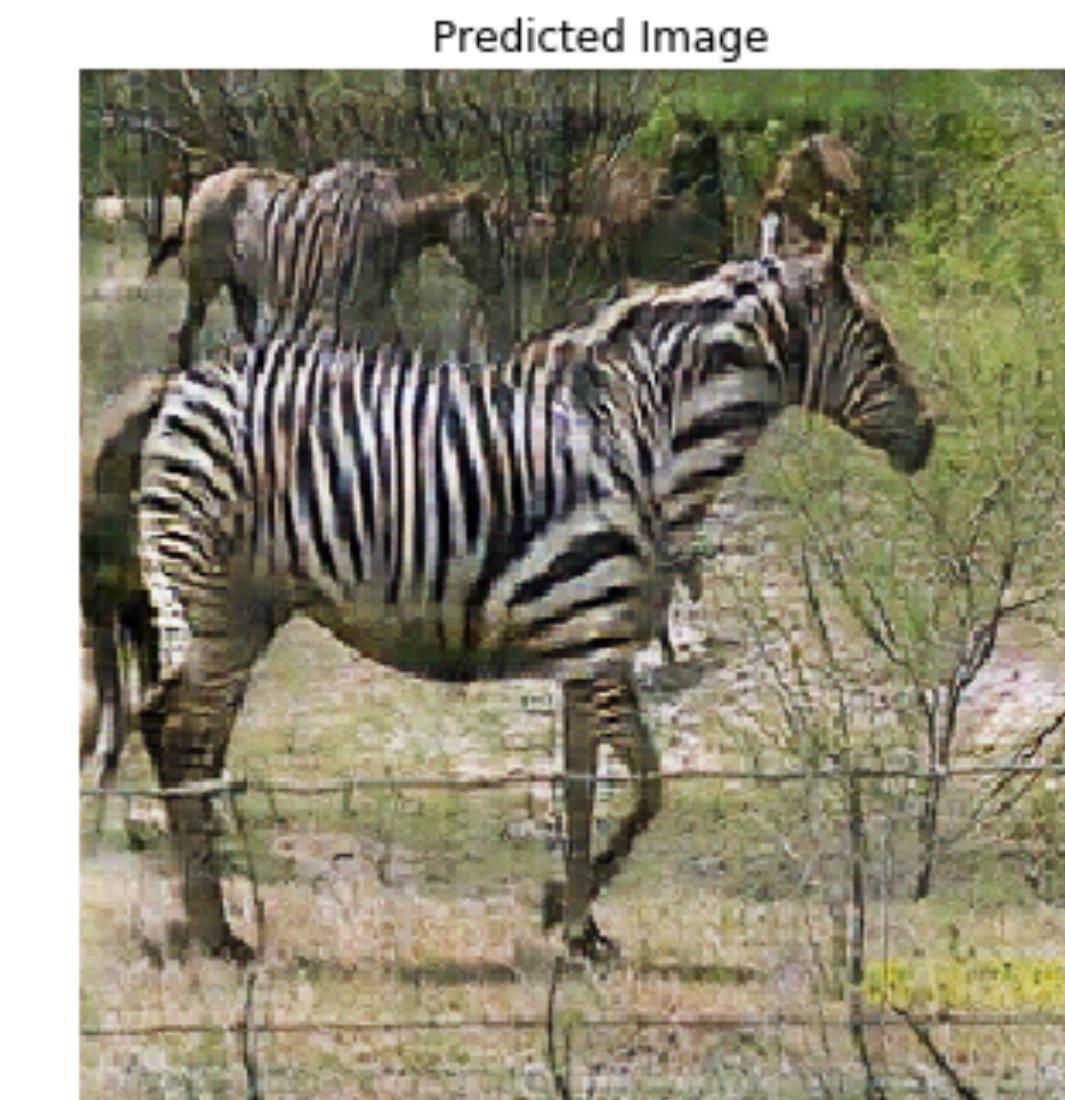
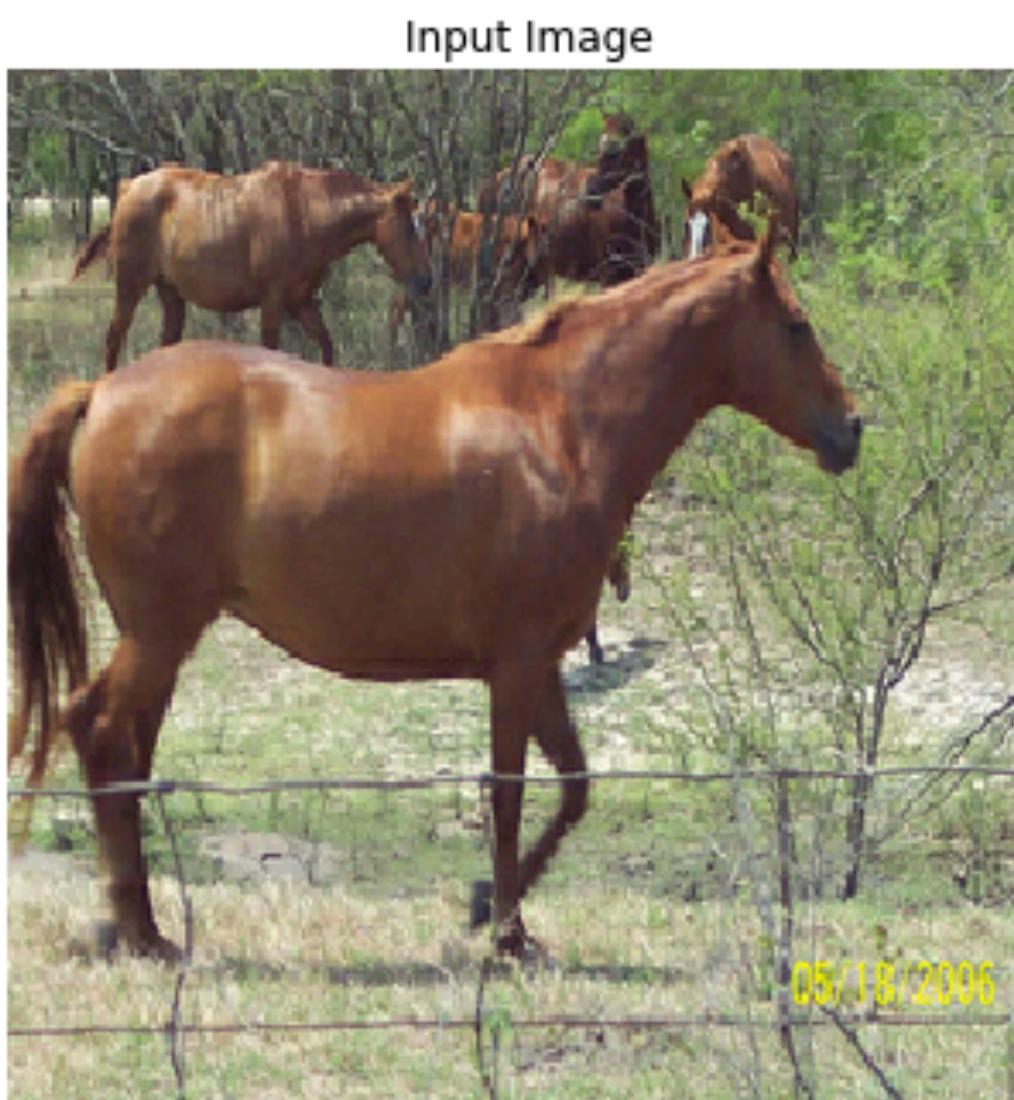
- не мерит разнообразие
- чаще используется

- Не оценивают непрерывность скрытого пространства

Метрики - неидеальны, постоянно появляются новые версии

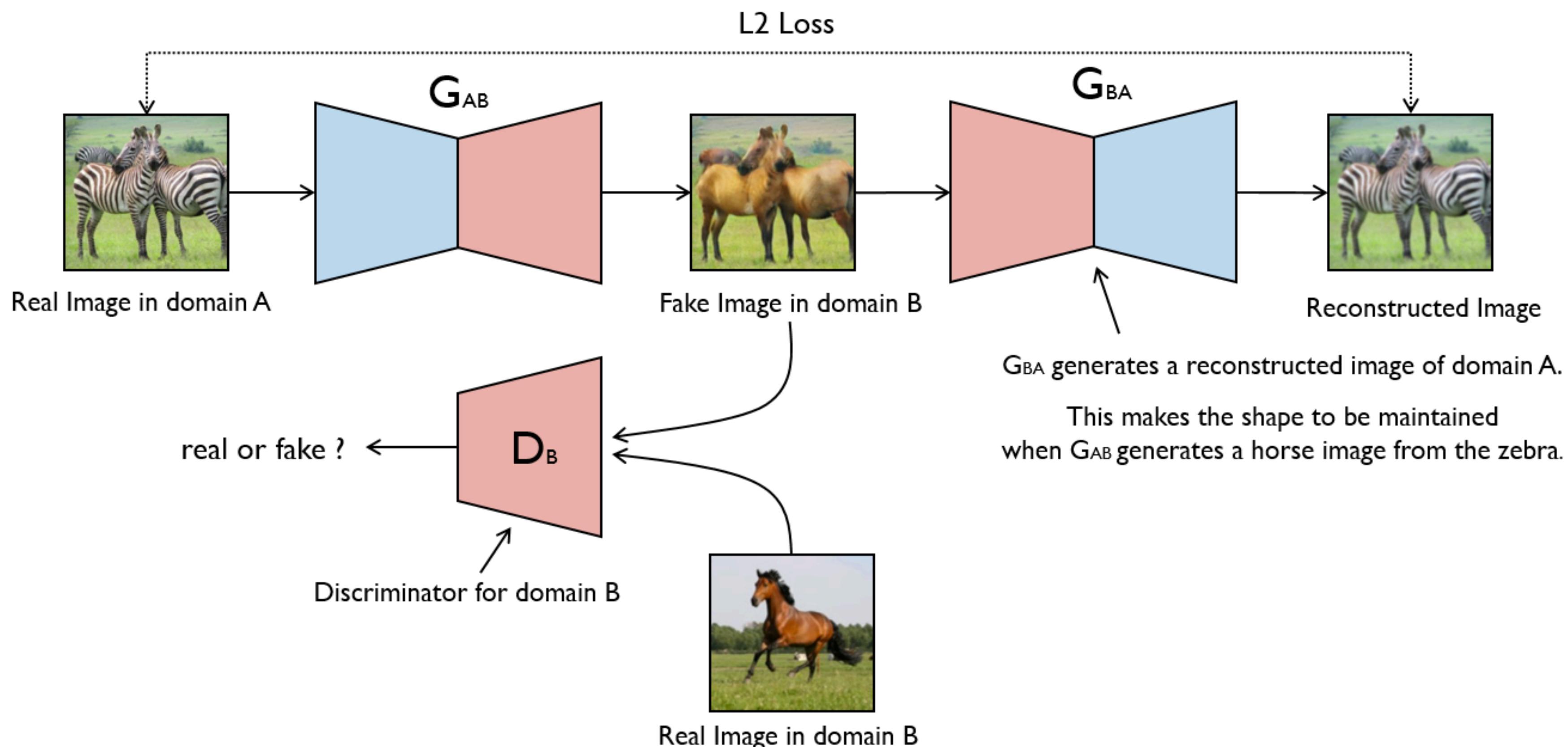
CycleGAN

Задача: перенос стиля (image-to-image)



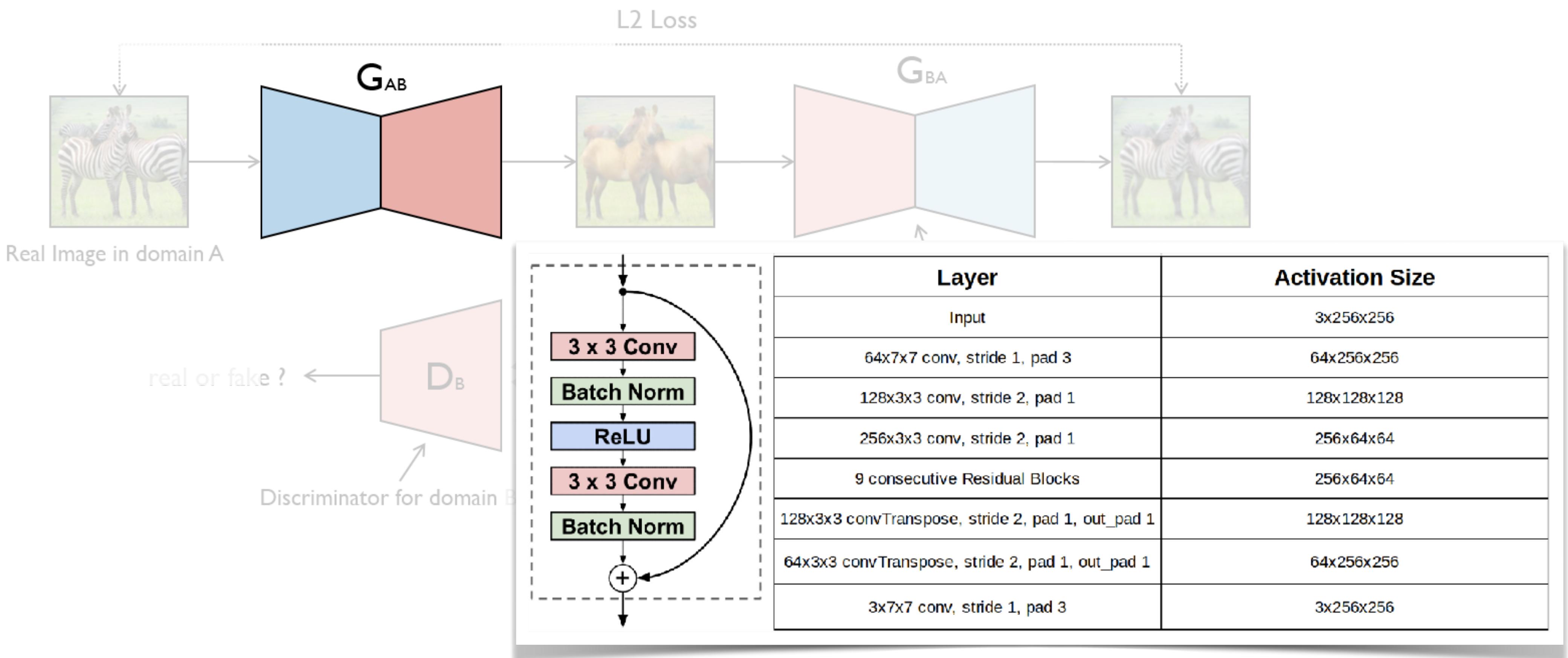
CycleGAN

Задача: перенос стиля (image-to-image)



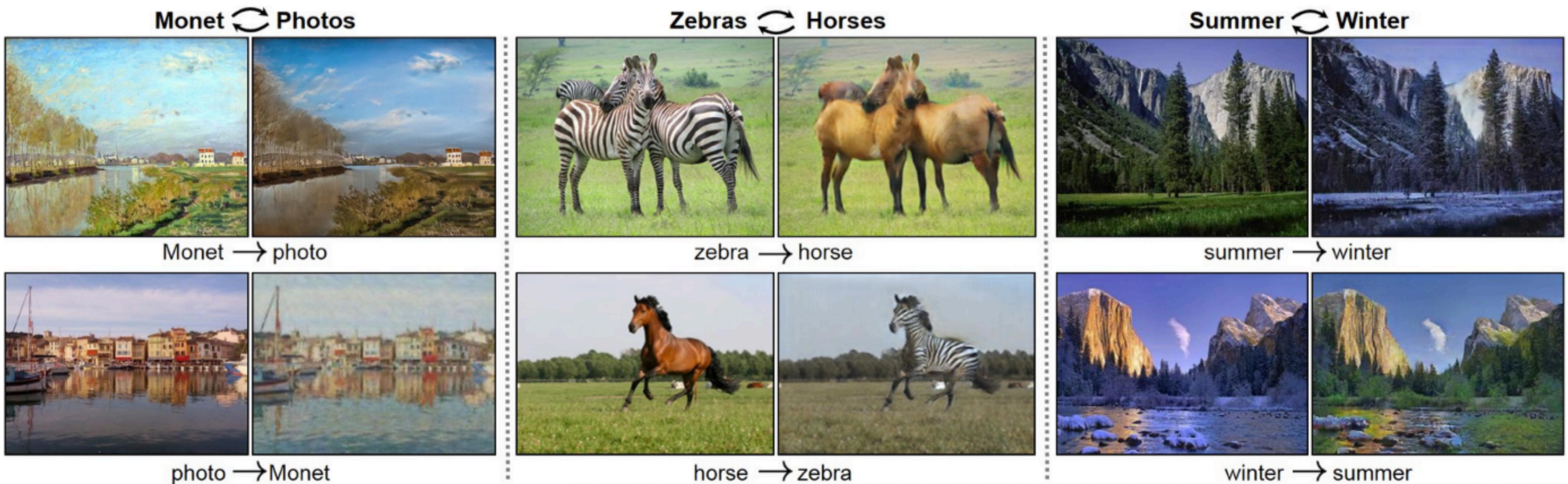
CycleGAN

Задача: перенос стиля (image-to-image)



CycleGAN

Задача: перенос стиля (image-to-image)



Современные GANы

Progressive Gan

StyleGAN

StyleGANv2



Быстро и любой домен

Не только лица

Исправили ошибки

А можно 3D?

ProjectedGAN

StyleGAN-XL

alias-freeGAN

EGO3d