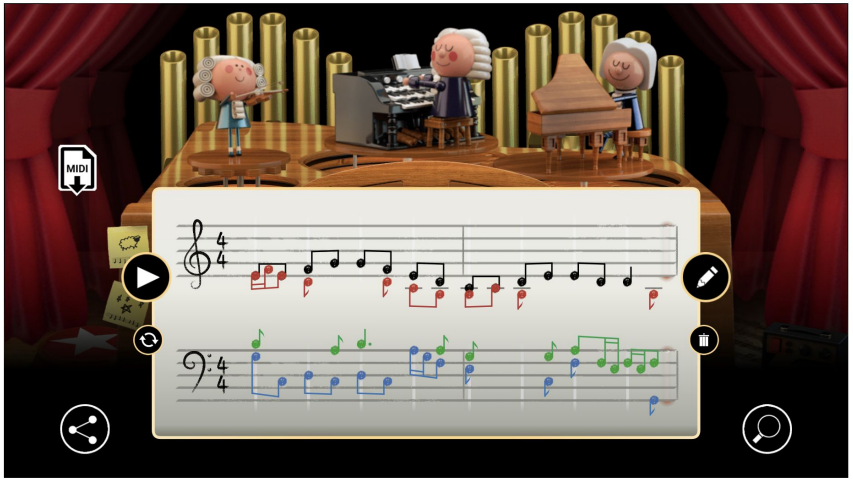
Skillbox

Генеративные состязательные сети (GAN)

Генеративные модели

https://thispersondoesnotexist.com/

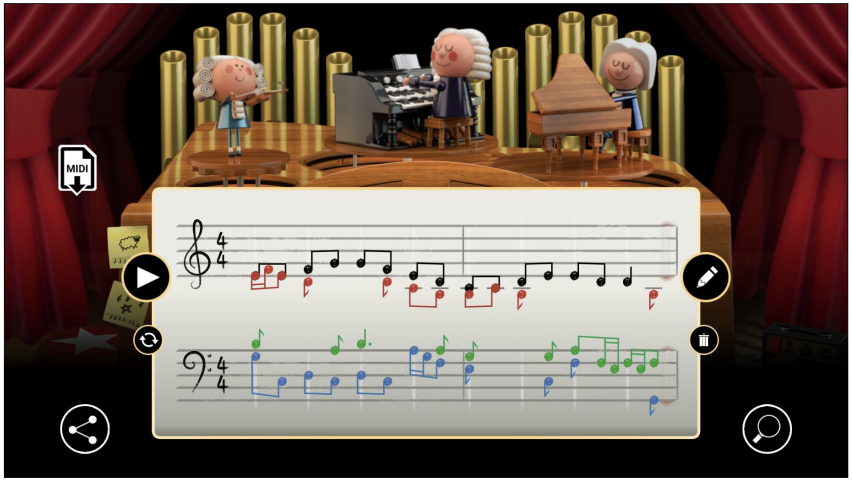


https://magenta.tensorflow.org/coconet

Images: Nvidia, Google

Генеративные модели

https://thispersondoesnotexist.com/



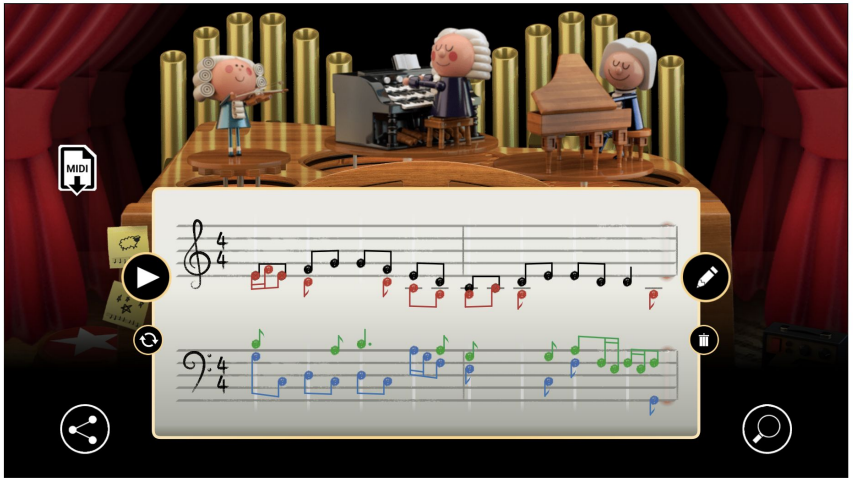
https://magenta.tensorflow.org/coconet

- Нужен универсальный подход который бы мог решать задачу:
- “Есть много данных. Хочу получить еще больше похожих на них”

Images: Nvidia, Google

Генеративные модели

https://thispersondoesnotexist.com/



https://magenta.tensorflow.org/coconet

- Нужен универсальный подход который бы мог решать задачу:
- “Есть много данных. Хочу получить еще больше похожих на них”
- Это значит: из того же распределения

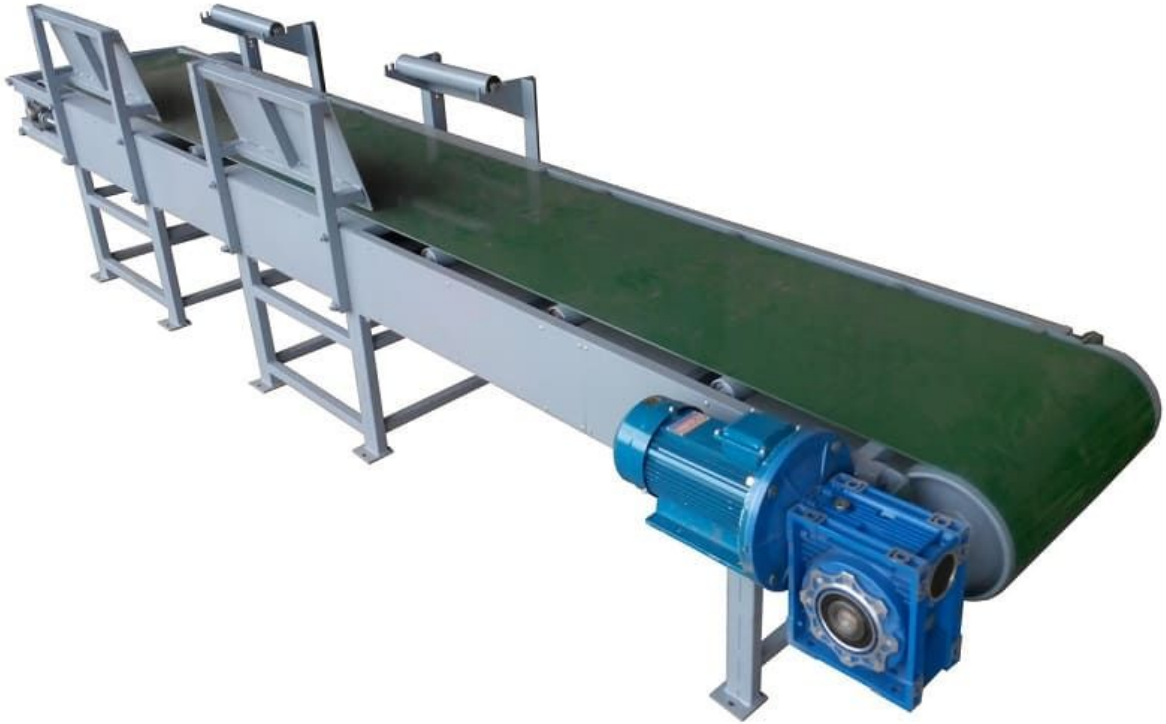
Images: Nvidia, Google

Skillbox

Генеративные состязательные сети (GAN)

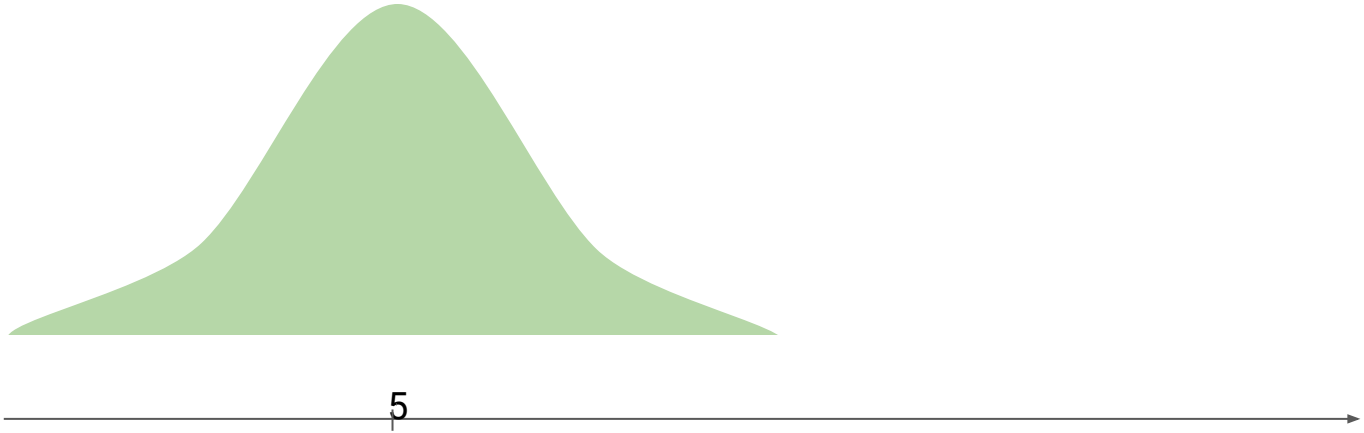
Сравнение распределений и генерация данных

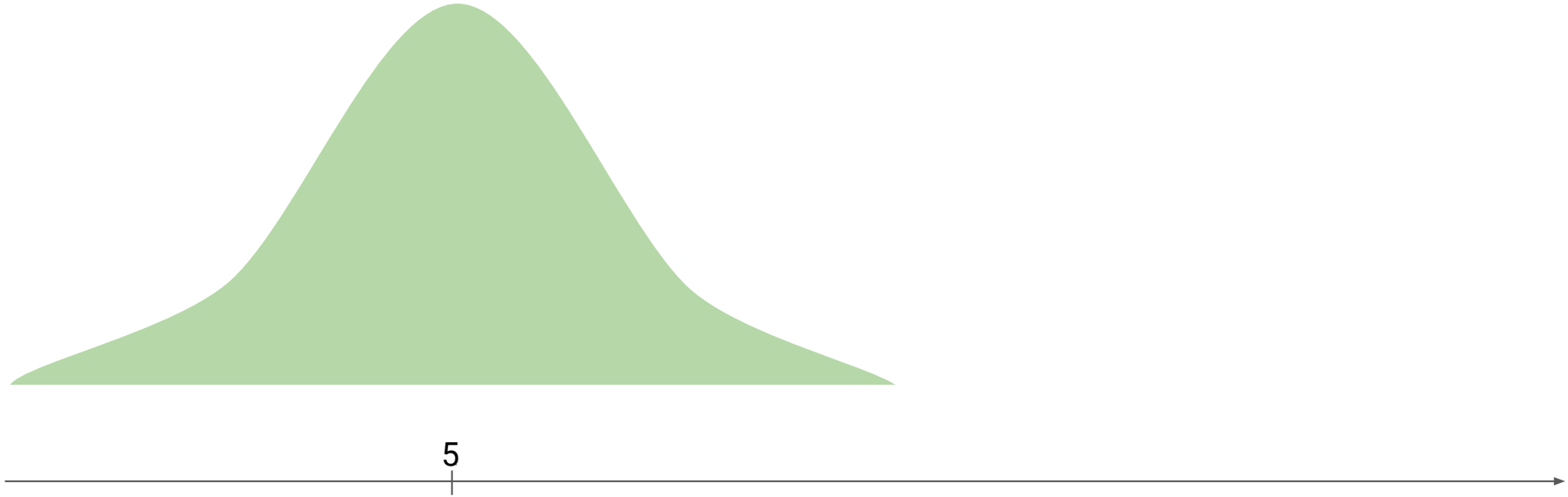
Сравнение распределений. Интуиция

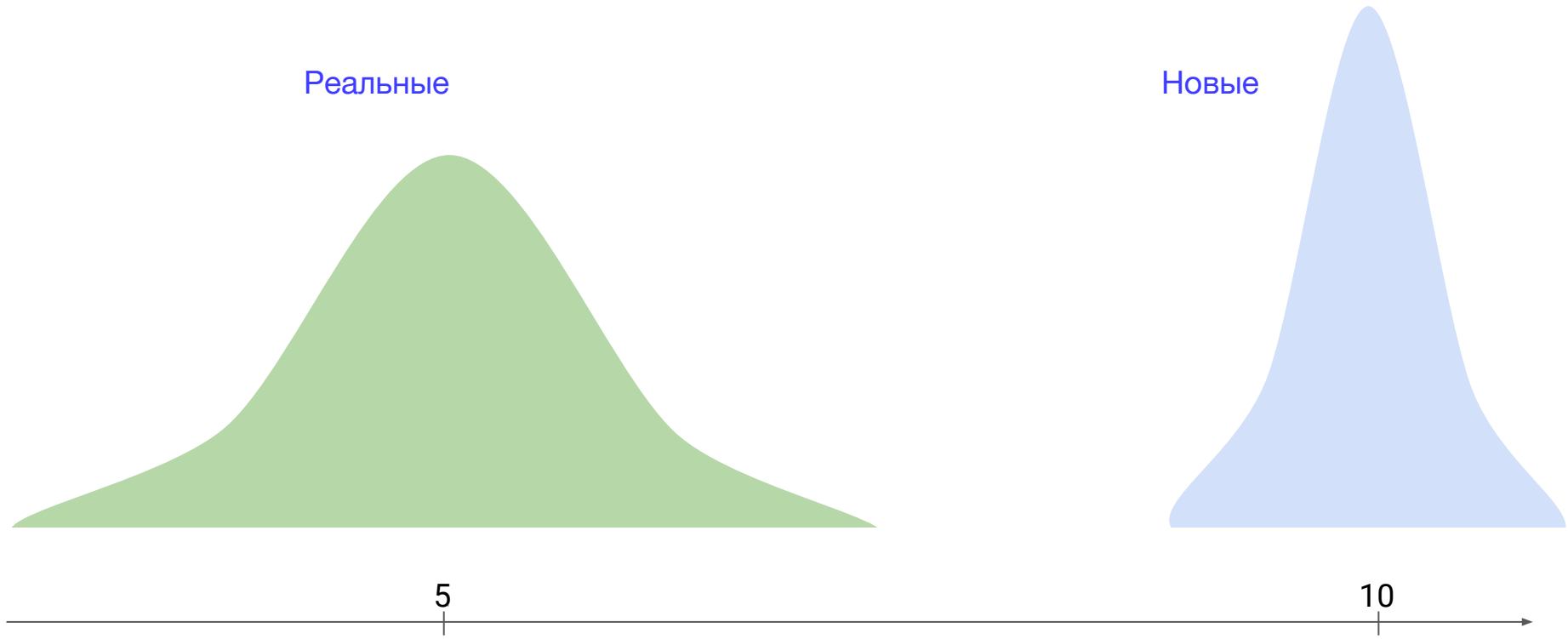


Сравнение распределений. Интуиция

- Задача **откалибровать**, чтобы он работал точно как и предыдущий
- При этом радиус шурупа должен быть 5 см, но допускается небольшое отклонение от этой величины.
- Точные технические требования утеряны, но **есть множество шурупов полученных на правильном конвейере.**

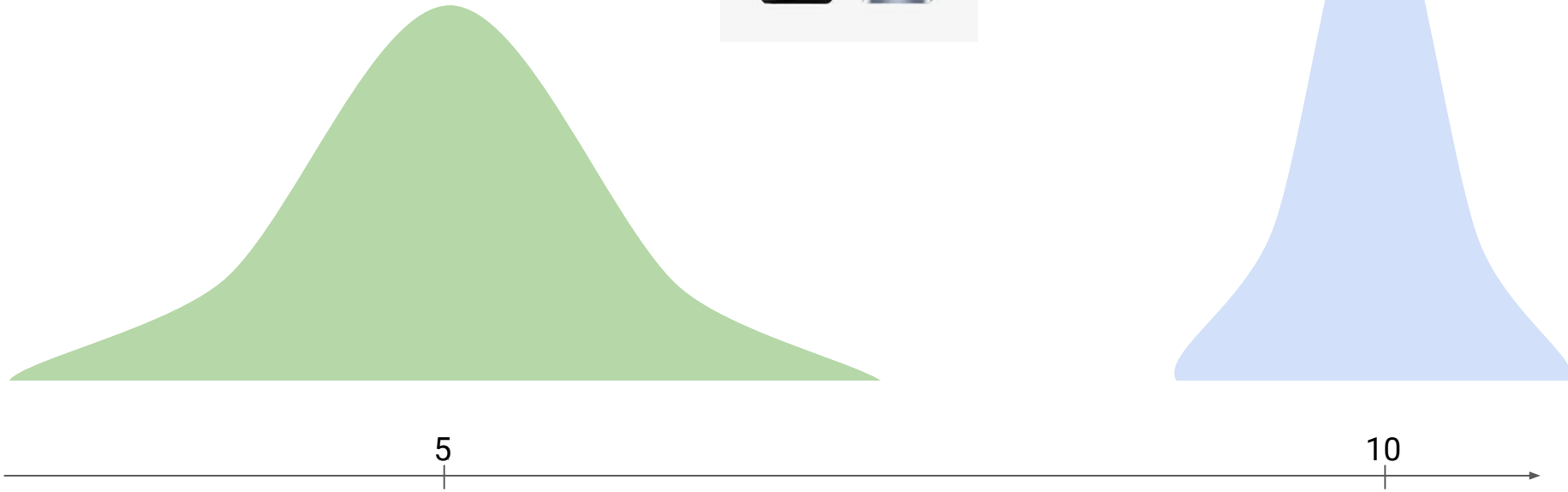


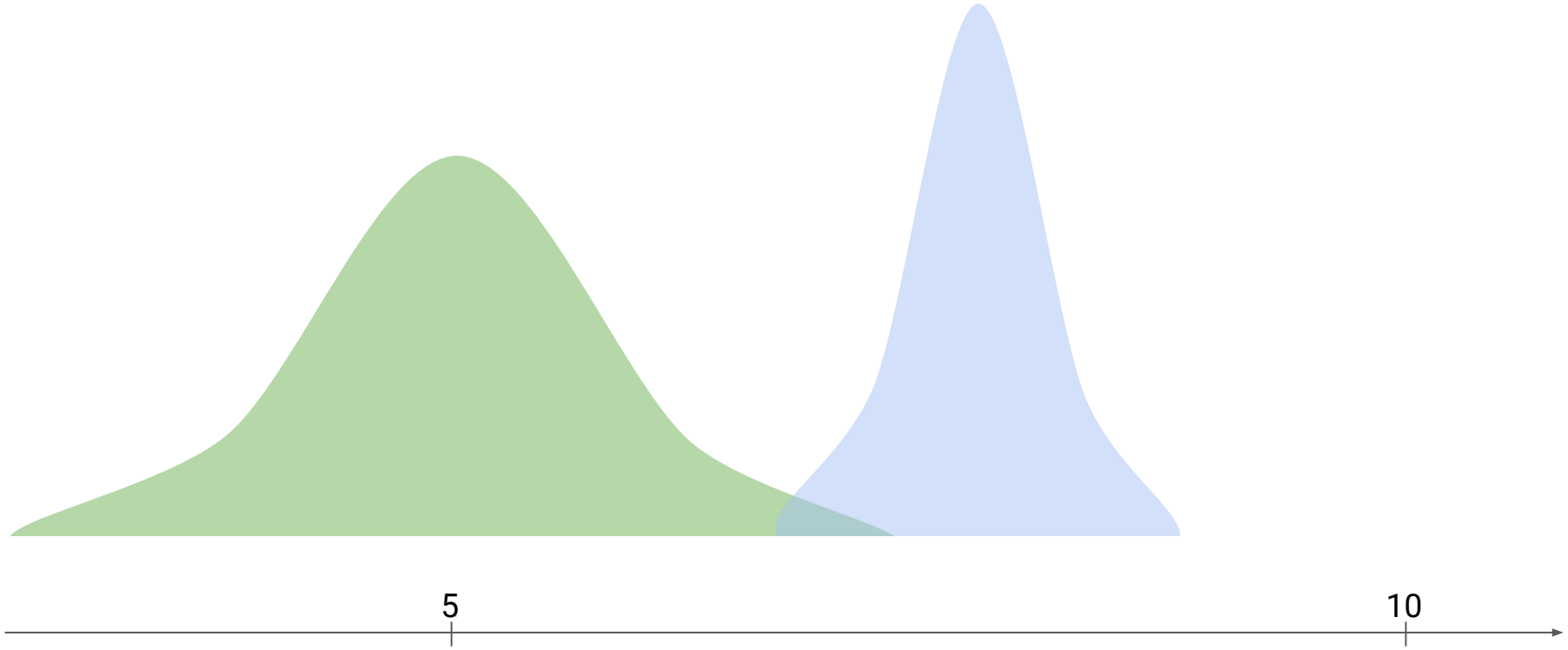

5

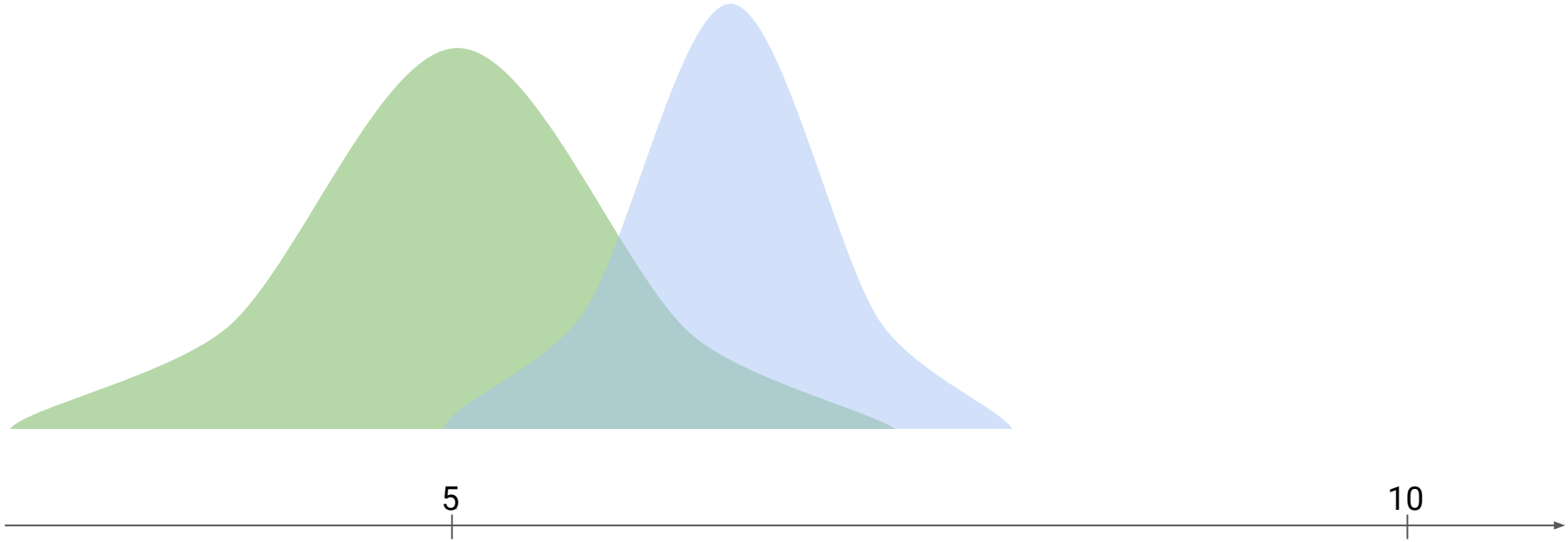

Реальные
Новые
5
10

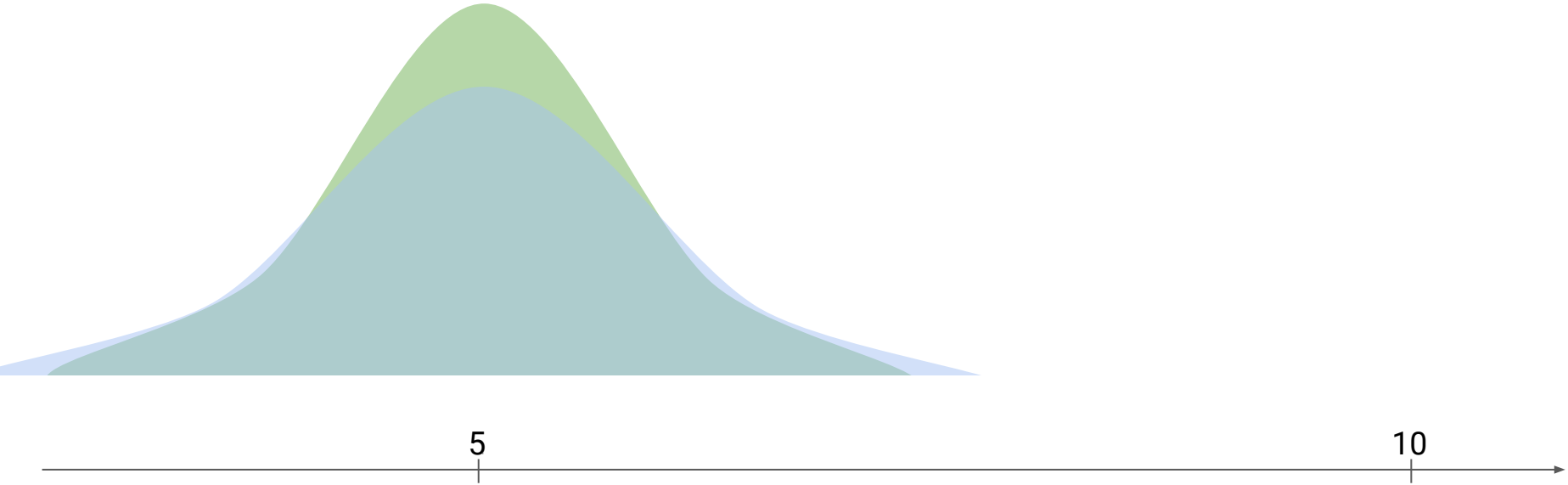
Реальные

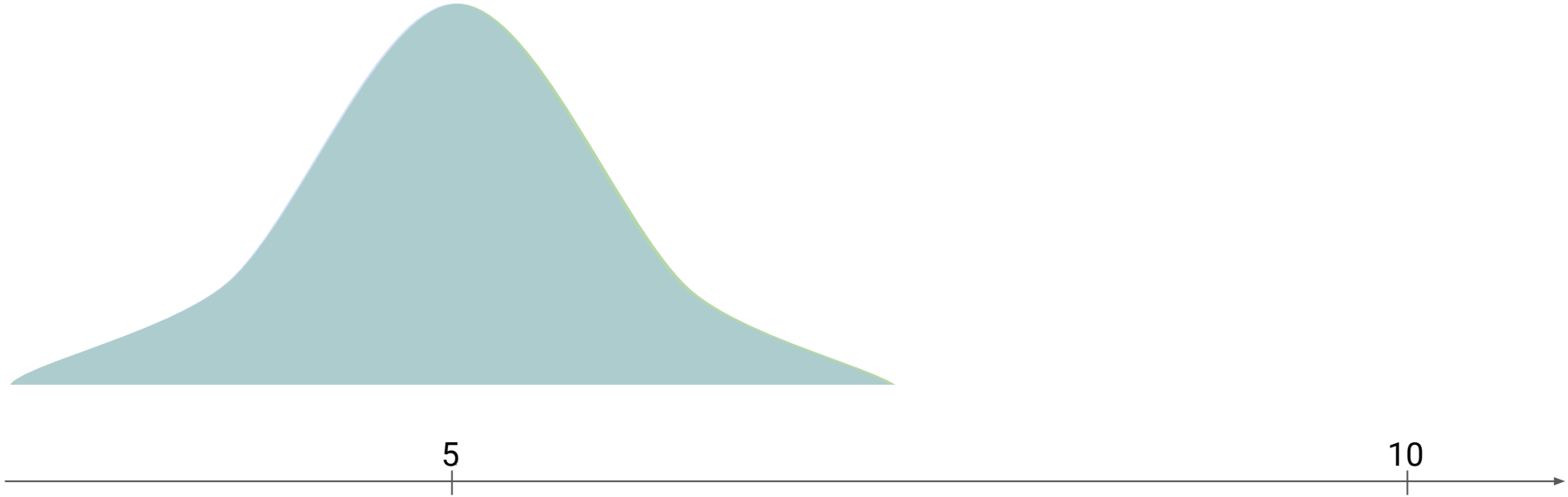
Новые

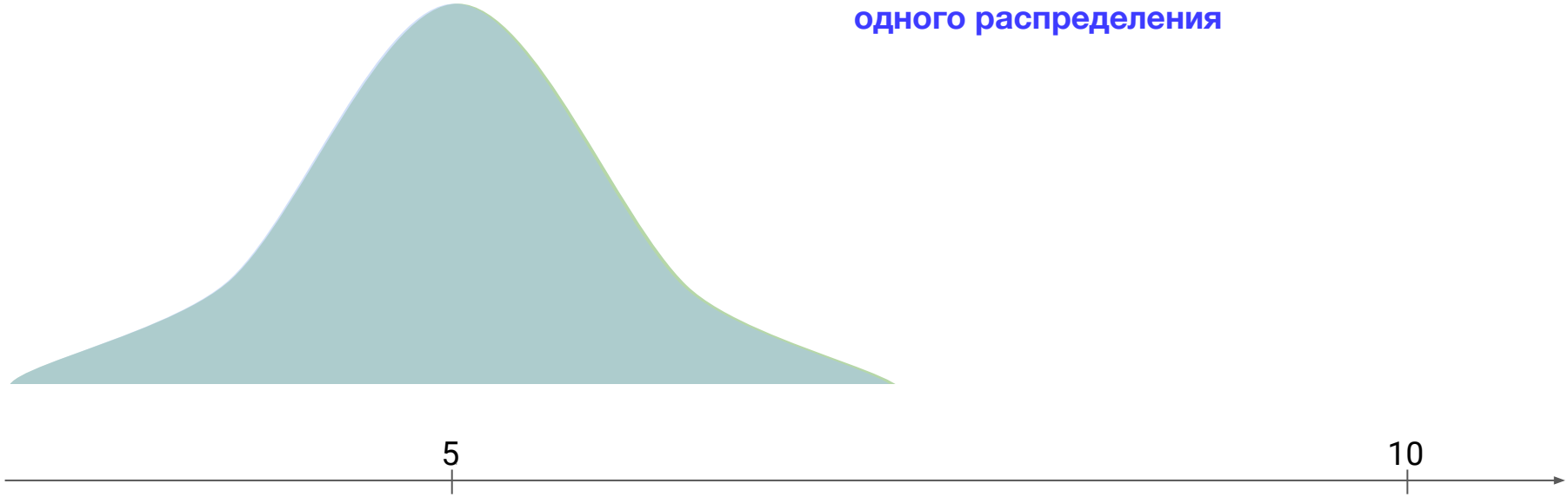



5
10

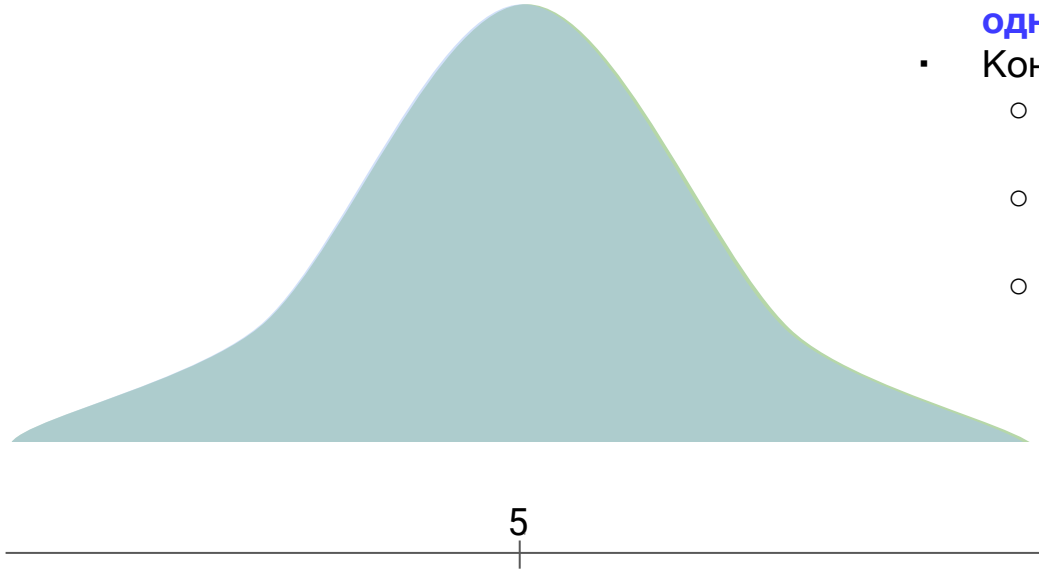

5
10


5
10


5
10



- Распределения совпали
- **Имея перед собой две кучки шурупов “отдел качества” их не сможет различить никак**
- Это значит, что **данные теперь поступают из одного распределения**



- Распределения совпали
- **Имея перед собой две кучки шурупов “отдел качества” их не сможет различить никак**
- Это значит, что **данные теперь поступают из одного распределения**
- Конвейер -- генератор данных
  - Задача -- выяснить реальное распределение данных
  - Теперь может генерировать “правильные” шурупы (данные)
  - Значит “**сэмплировать из распределения**”

- Распределения совпали
- **Имея перед собой две кучки шурупов**
- **ими не сможет различить**

- **нные теперь поступают из**
- **ления**
- атор данных
- яснить реальное
- ие данных
- г генерировать
- ” шурупы (данные)
- **лировать из**
- **ния**”

- *Мысль: при наличии того, кто умеет отличать сгенерированные данные от реальных реальные и при наличии того, кто умеет такие данные производить, можно после нескольких итераций научиться генерировать нужные данные*

- Распределения совпали
- **Имея перед собой две кучки шурупов**
- **их не сможет различить**

- **нные теперь поступают из**
- **ления**
- атор данных
- яснить реальное
- ие данных
- г генерировать
- ” шурупы (данные)
- **лировать из**
- **ния**”

- *Мысль: при наличии того, кто умеет отличать сгенерированные данные от реальных реальные и при наличии того, кто умеет такие данные производить, можно после нескольких итераций научиться генерировать нужные данные*
- Но это было слишком простой одномерный случай. А если размерность больше?

Пространство изображений

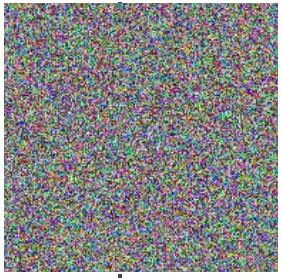
- Цветные изображения фиксированного размера -- 200x200.
- Всевозможных изображений:
  - 256^(200*200*3)
  - Т.е. каждый из 200*200*3 пикселей может принимать 256 значений цвета.
  - Это. Очень. Много.

Пространство изображений

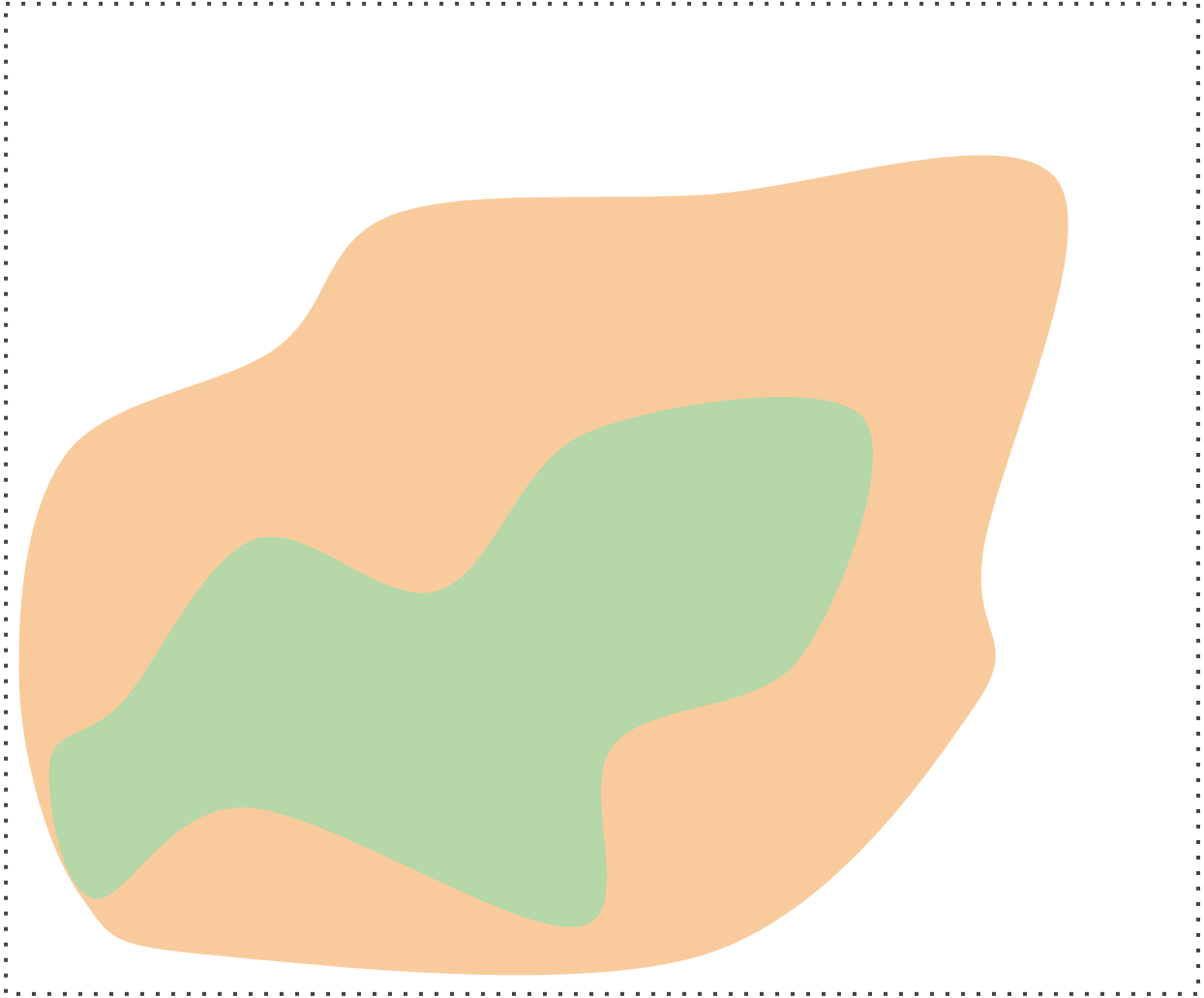
- Цветные изображения фиксированного размера -- 200x200.
- Всевозможных изображений:
  - 256^(200*200*3)
  - Т.е. каждый из 200*200*3 пикселей может принимать 256 значений цвета.
  - Это. Очень. Много.
- В случае с шурупом -- пространство которым мы описывали шуруп состояло из одной размерности -- его радиуса

Пространство изображений

- Цветные изображения фиксированного размера -- 200x200.
- Всевозможных изображений:
  - 256^(200*200*3)
  - Т.е. каждый из 200*200*3 пикселей может принимать 256 значений цвета.
  - Это. Очень. Много.
- В случае с шурупом -- пространство которым мы описывали шуруп состояло из одной размерности -- его радиуса
- **Хотим научиться генерировать реалистичные лица**
  - Т.е. Если раньше нам нужно было “найти” где находятся реальные данные (радиус 5) в одной размерности
  - А теперь в огромнейшем пространстве



- подпространство с привычными для нас изображениями
- цвета в них меняются плавно, в них мало шума и т.д.




pure

- Сгенерированные ~~шурупы~~ лица, которые попадают в зеленую область не будут отличимы от реальных

- Сгенерированные ~~шурупы~~ лица, которые попадают в зеленую область не будут отличимы от реальных
- В этой области есть свои “более” вероятные лица и менее вероятные как и в случае с допустимыми радиусами в примере с шурупами
-

- Сгенерированные ~~шурупы~~ лица, которые попадают в зеленую область не будут отличимы от реальных
- В этой области есть свои “более” вероятные лица и менее вероятные как и в случае с допустимыми радиусами в примере с шурупами
-

Как задать распределение

- Задать распределение в таком огромном пространстве (пространстве изображений) невозможно

Как задать распределение

- Задать распределение в таком огромном пространстве (пространстве изображений) невозможно
- Упростим себе задачу
- Воспользуемся двумя приемами
  - Понижение размерности
  - Сэмплирование из простого распределения и превращение его в более сложное

Как задать распределение

- Задать распределение в таком огромном пространстве (пространстве изображений) невозможно
- Упростим себе задачу
- Воспользуемся двумя приемами
  - **Понижение размерности**
  - Сэмплирование из простого распределения и превращение его в более сложное

Как задать распределение

- Задать распределение в таком огромном пространстве (пространстве изображений) невозможно
- Упростим себе задачу
- Воспользуемся двумя приемами
  - **Понижение размерности (сколько чисел нужно для описания числа?)**
  - Сэмплирование из простого распределения и превращение его в более сложное

Размерность лиц

- Игры, где вы создавали лицо персонажа
- Не было предложено подобрать несколько триллионов параметров
- Ограниченное число “рычагов”

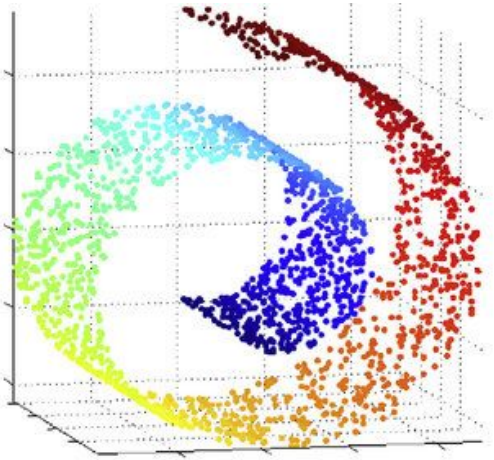
Размерность лиц

- Игры, где вы создавали лицо персонажа
- Не было предложено подобрать несколько триллионов параметров
- Ограниченное число “рычагов”
- Т.е. лицо человека можно описать десятком чисел

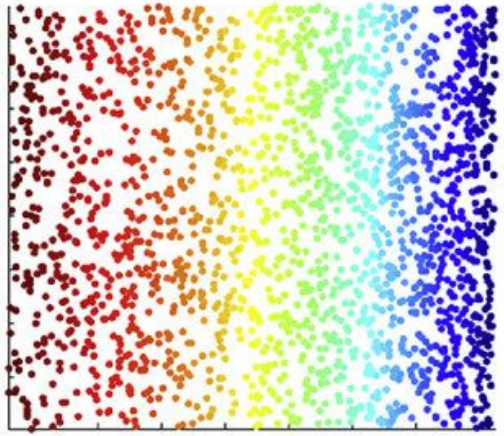
Размерность лиц

- Игры, где вы создавали лицо персонажа
- Не было предложено подобрать несколько триллионов параметров
- Ограниченное число “рычагов”
- Т.е. лицо человека можно описать десятком чисел
- **Для описания лица не нужно описывать все пиксели -- они скоррелированы**
- Например задав цвет кожи -- мы уже “определяем” цвет половины картинки :)

Манифолд

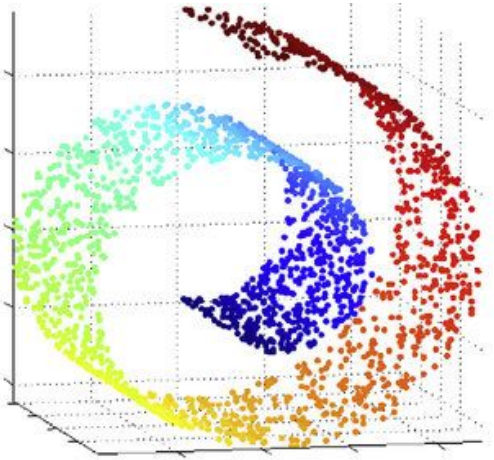


Данные высокой размерности со “структурой”

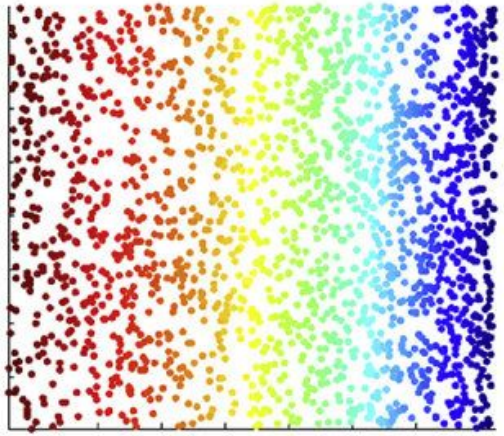


Можно представить в пространстве меньшей размерности

Манифолд

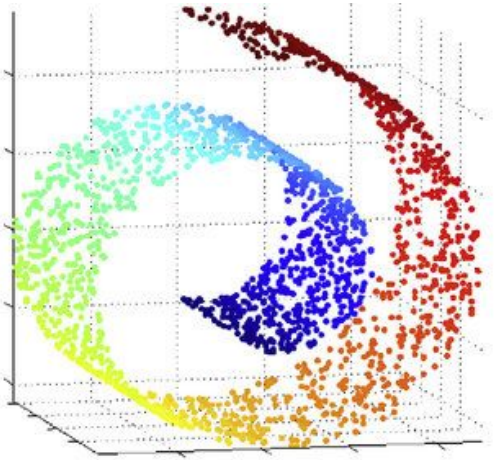


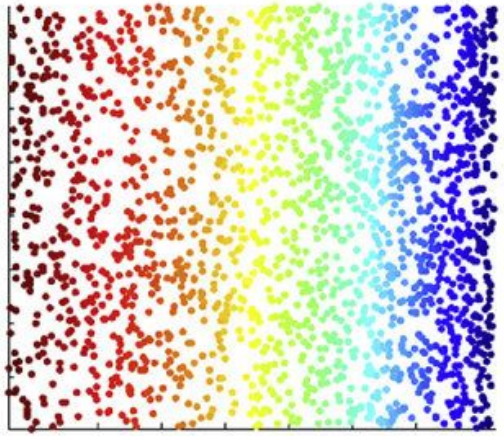
Данные высокой размерности со “структурой”



Можно представить в пространстве меньшей размерности
Размерность -- **количество латентных (скрытых) переменных**

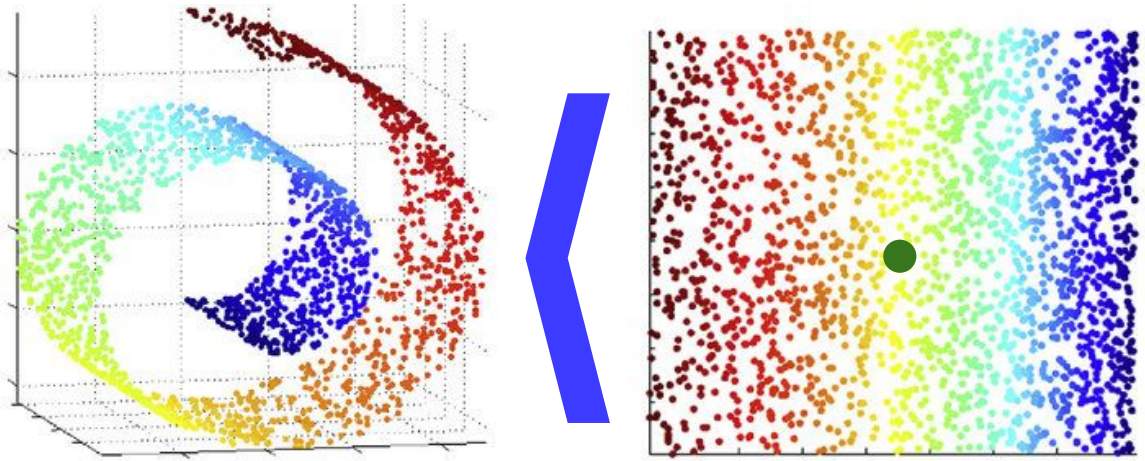
Генерация данных





Генерация данных

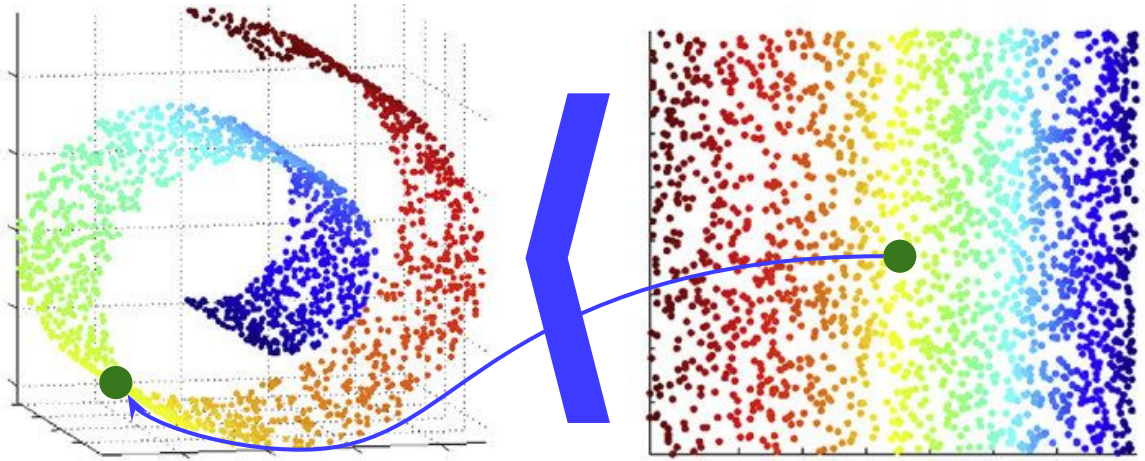
Если мы знаем преобразование *



- Сэмплируем точку в “простом” пространстве из “простого” распределения

Генерация данных

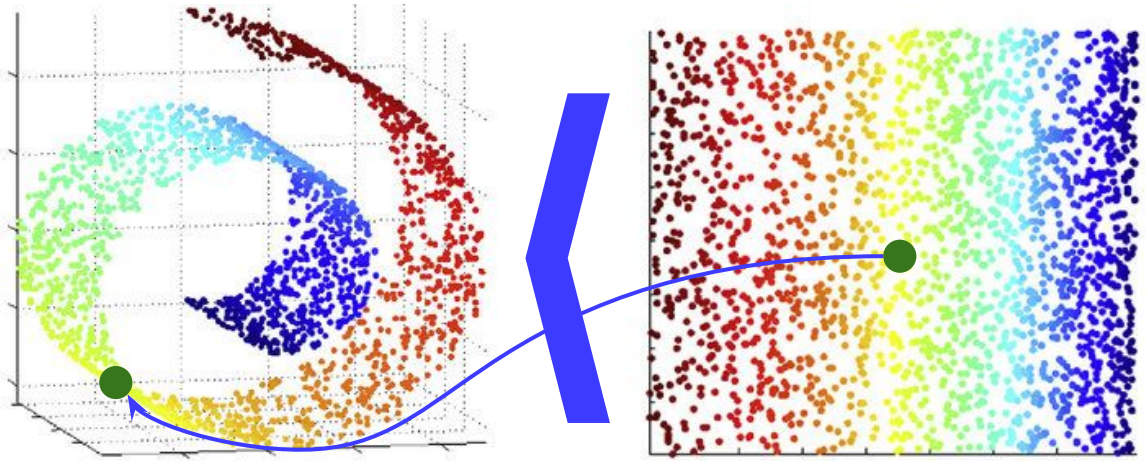
Если мы знаем преобразование *



- Сэмплируем точку в “простом” пространстве из “простого” распределения
- Преобразуем ее и получаем точку в исходном пространстве

Генерация данных

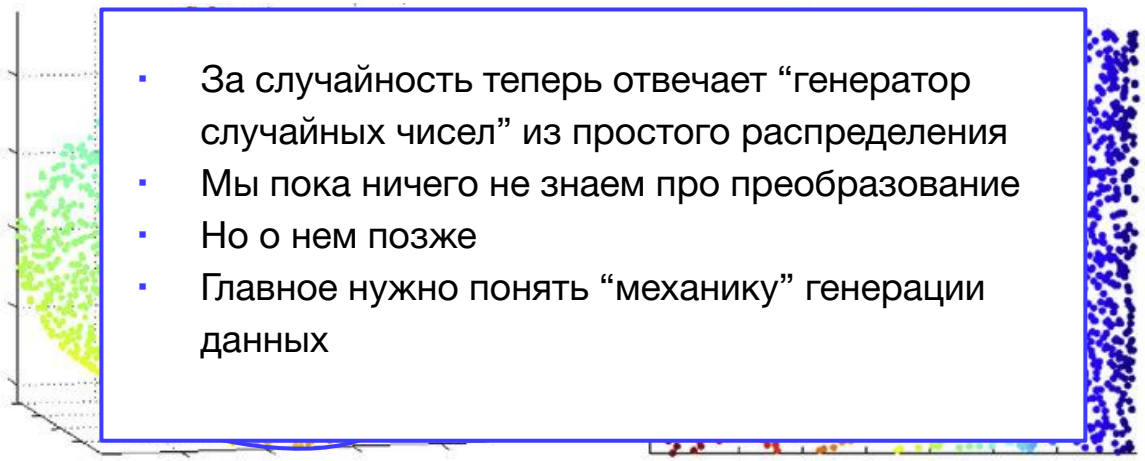
Если мы знаем преобразование *



- Сэмплируем точку в “простом” пространстве из “простого” распределения
- Преобразуем ее и получаем точку в исходном пространстве
- Не задавали распределени напрямую -- пользовались преобразованием

Генерация данных

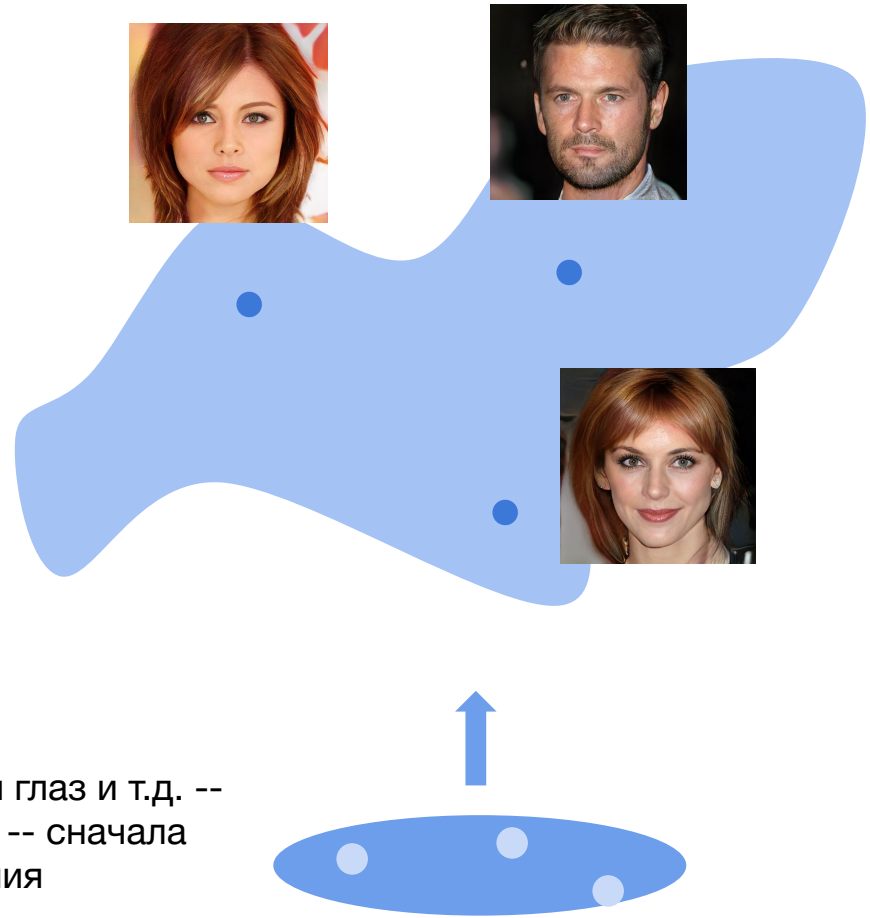
Если мы знаем преобразование *



- За случайность теперь отвечает “генератор случайных чисел” из простого распределения
- Мы пока ничего не знаем про преобразование
- Но о нем позже
- Главное нужно понять “механику” генерации данных

- Сэмплируем точку в “простом” пространстве из “простого” распределения
- Преобразуем ее и получаем точку в исходном пространстве
- Не задавали распределени напрямую -- пользовались преобразованием

Цвет кожи, волос, тип глаз и т.д. -- скрытые переменные -- сначала генерируем их значения



Цвет кожи, волос, тип глаз и т.д. -- скрытые переменные -- сначала генерируем их значения

Итог

- Поняли, что такое распределение данных и что происходит когда они совпадают
  - невозможно отличить реальные данные от сгенерированных
  - это нам и хочется научиться делать.

Итог

- Поняли, что такое распределение данных и что происходит когда они совпадают
  - невозможно отличить реальные данные от сгенерированных
  - это нам и хочется научиться делать.
- Убедились в сложности такой задачи для больших размерностей.
  - Например для картинок
- Интуитивно поняли как устроено пространство изображений.

Итог

- Поняли, что такое распределение данных и что происходит когда они совпадают
  - невозможно отличить реальные данные от сгенерированных
  - это нам и хочется научиться делать.
- Убедились в сложности такой задачи для больших размерностей.
  - Например для картинок
- Интуитивно поняли как устроено пространство изображений.
- Узнали, что такое манифолд и скрытые переменные
  - В любой непонятной ситуации вспоминайте пример про бегунки в компьютерной игре. Это помогает :)

Итог

- Набросали схему генерации объектов, которая упрощает работу с многомерными распределениями
  - Сначала генерируем точки с помощью простого распределения меньшей размерности.
  - А затем с помощью сложной функции превращаем их в объекты итогового распределения
  - Мы пока не знаем как находить эту сложную функцию

Skillbox

Генеративные состязательные сети (GAN)

GAN

Ранее

- В прошлом уроке мы познакомились с интуицией скрытой за генерацией данных и сравнением распределений
- Пример с заводом:
  - если у нас есть “отдел качества” и возможность менять параметры конвейера, то через несколько итераций мы можем научиться производить объекты с правильными характеристиками
  - Т.е. Такие которые нельзя отличить от настоящих

Ранее

- В прошлом уроке мы познакомились с интуицией скрытой за генерацией данных и сравнением распределений
- Пример с заводом:
  - если у нас есть “отдел качества” и возможность менять параметры конвейера, то через несколько итераций мы можем научиться производить объекты с правильными характеристиками
  - Т.е. Такие которые нельзя отличить от настоящих
- Как же нам описать это математически? Кто же заменит нам “отдел качества” и “конвейер” в этом случае?

Ранее

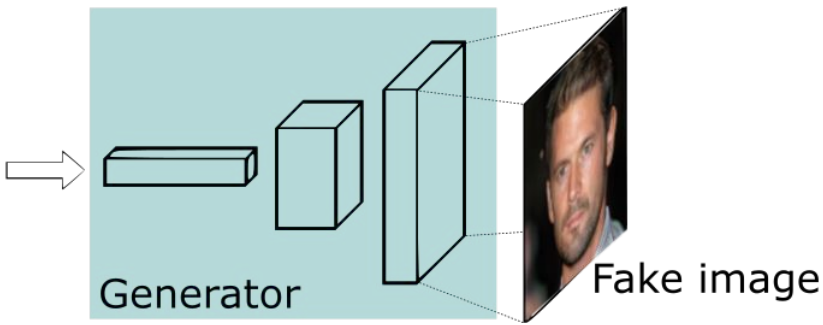
- В прошлом уроке мы познакомились с интуицией скрытой за генерацией данных и сравнением распределений
- Пример с заводом:
  - если у нас есть “отдел качества” и возможность менять параметры конвейера, то через несколько итераций мы можем научиться производить объекты с правильными характеристиками
  - Т.е. Такие которые нельзя отличить от настоящих
- Как же нам описать это математически? Кто же заменит нам “отдел качества” и “конвейер” в этом случае?
- **Как вы уже могли догадаться -- нейронная сеть. А точнее две.**

GAN

- Generative Adversarial Nets (Networks), Ian J. Goodfellow et al, 2014
- Общий подход к поиску распределения данных
- Сегодня используется для широкого спектра задач и доменов (картинки, музыка)

0.1, -0.3, 034, 59 ...

Random vector

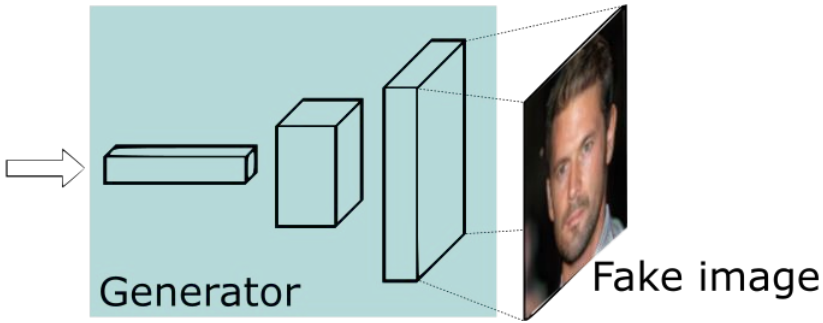


- Учится генерировать данные

Adapted from https://sthalles.github.io/intro-to-gans/

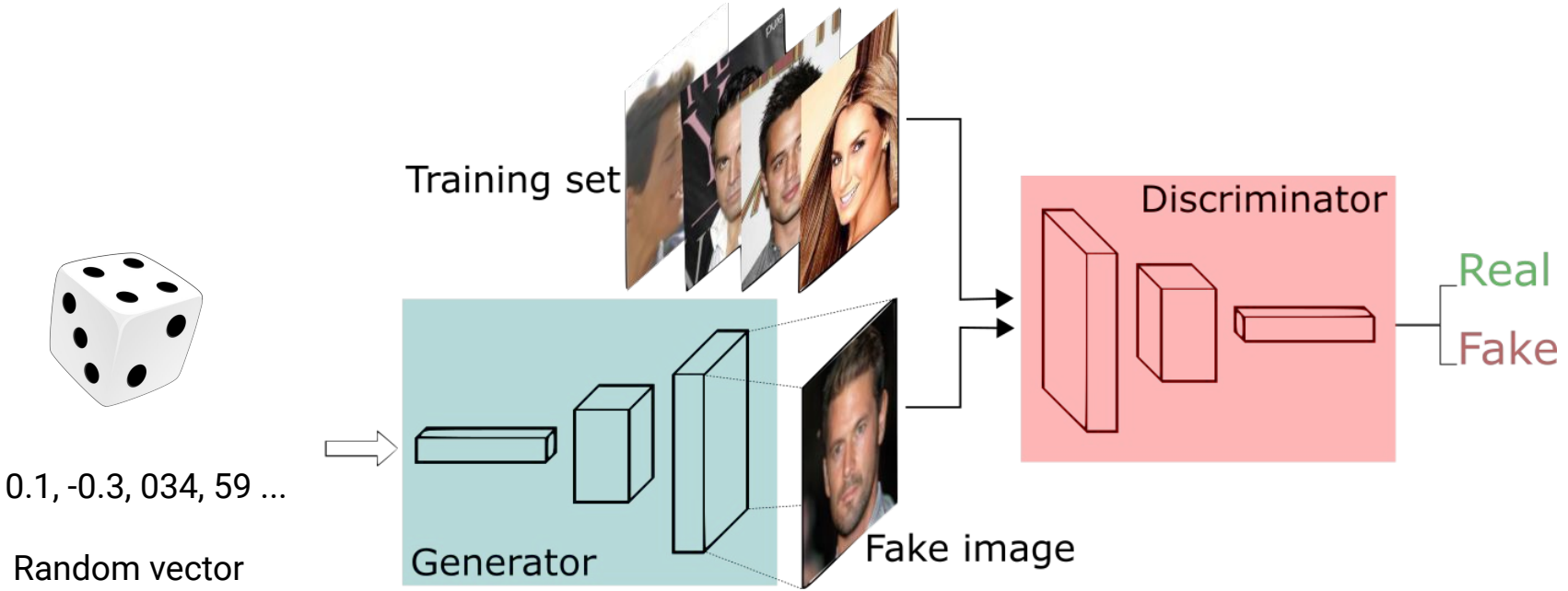
0.1, -0.3, 034, 59 ...

Random vector

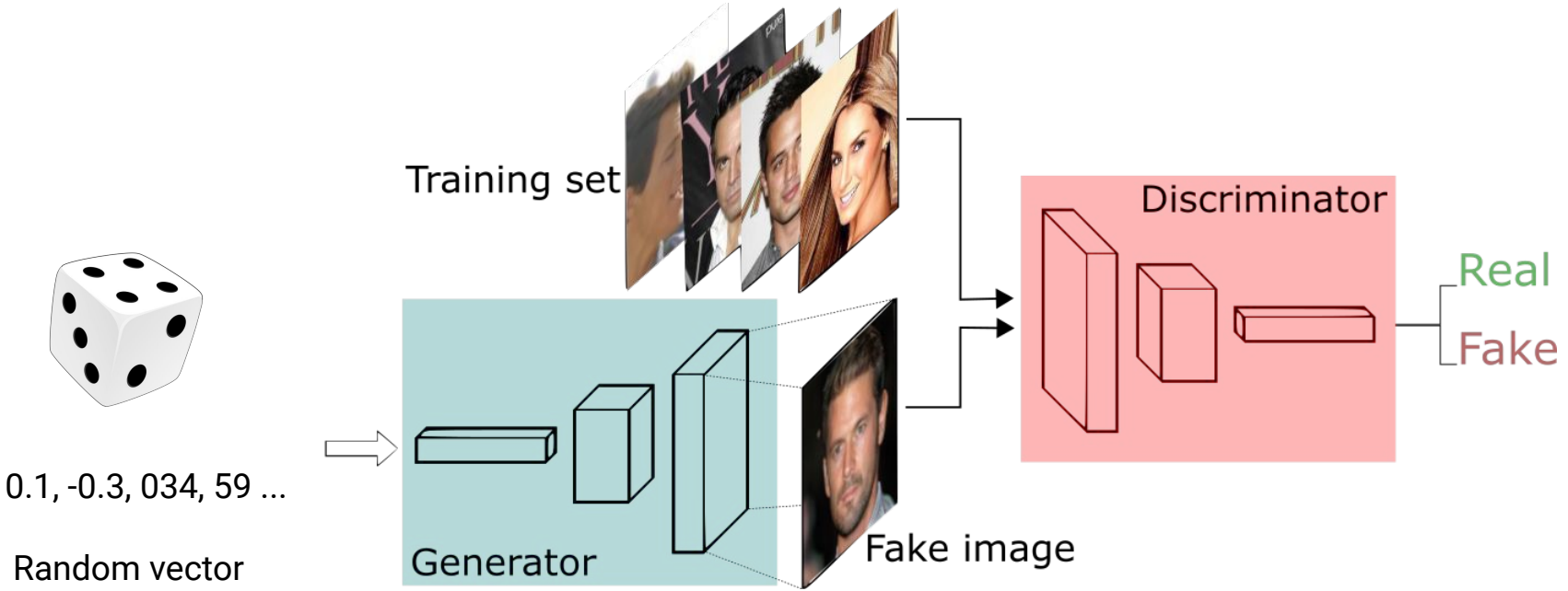


- Учится генерировать данные
- Поможет ему понять насколько он хорошо это делает -- **дискриминатор**

Adapted from https://sthalles.github.io/intro-to-gans/

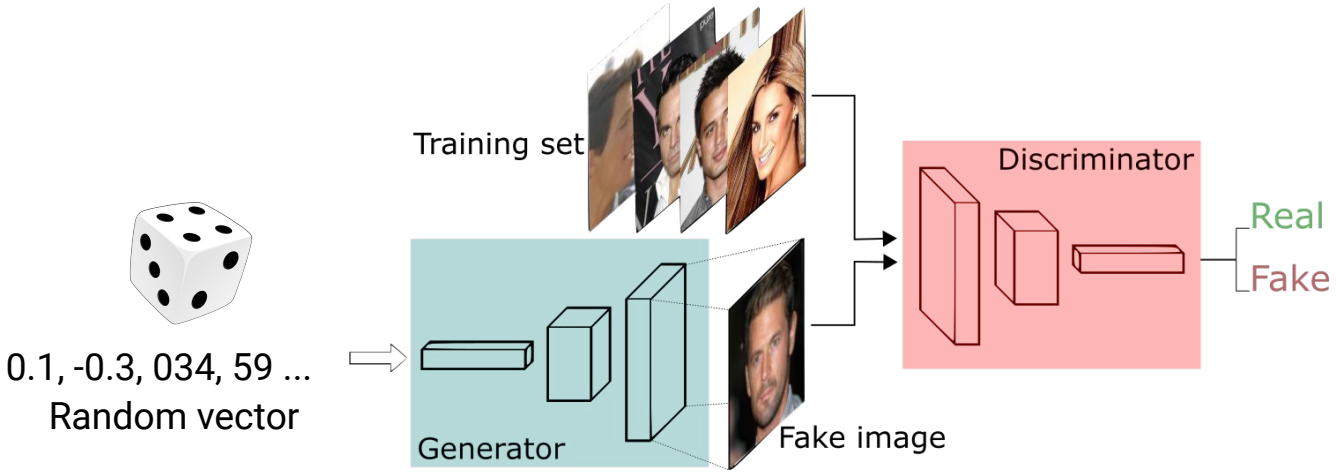


Adapted from https://sthalles.github.io/intro-to-gans/



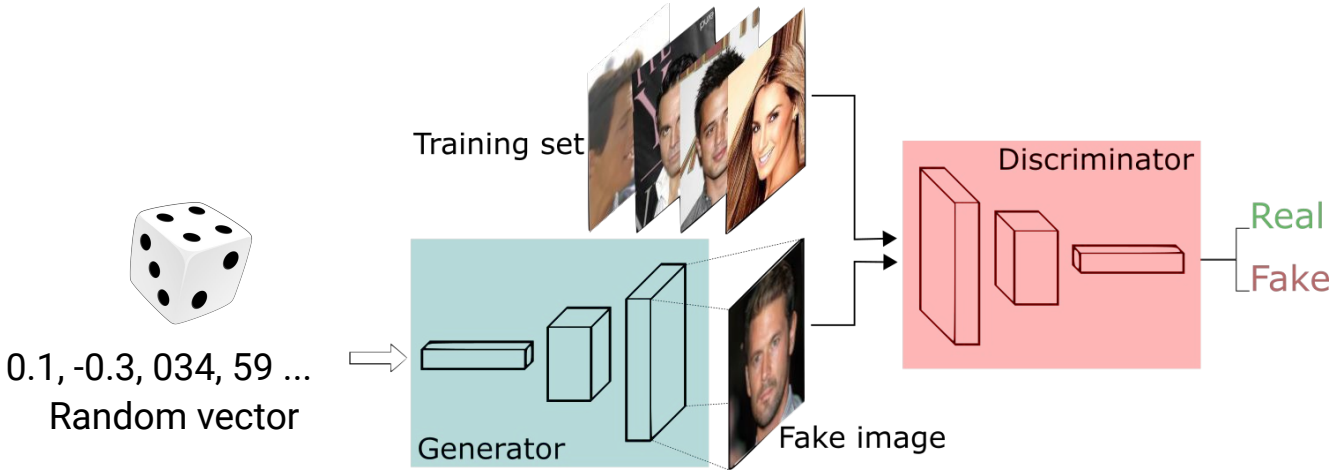
- Нейронные сети, значит должны быть лосс функции
- Какие они?

Adapted from https://sthalles.github.io/intro-to-gans/



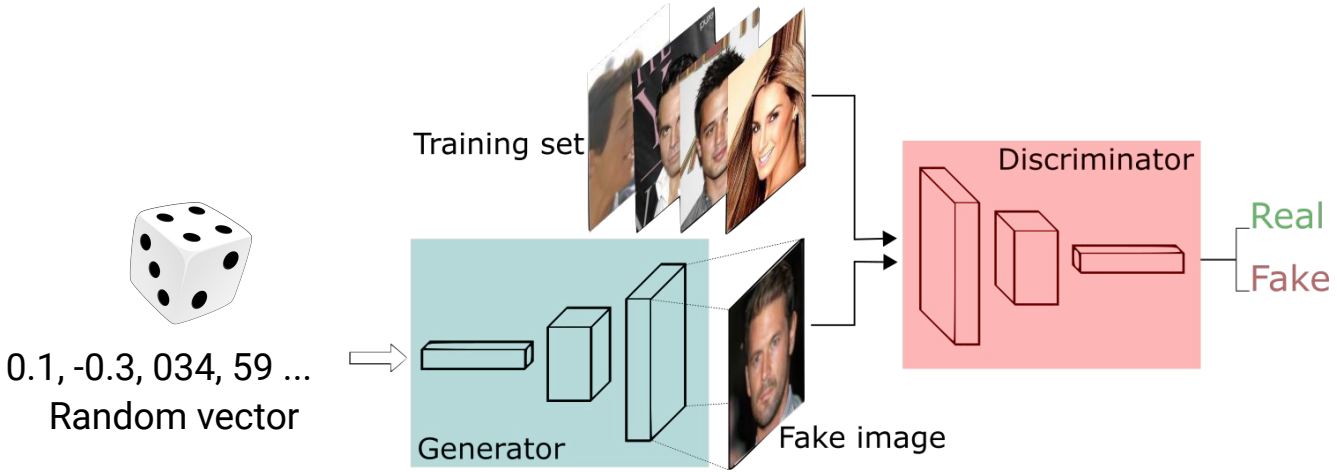
- Для дискриминатора -- все просто
  - бинарная классификации: 0 -- fake, 1 --real
  - Кросс энтропия, логлосс

Adapted from https://sthalles.github.io/intro-to-gans/



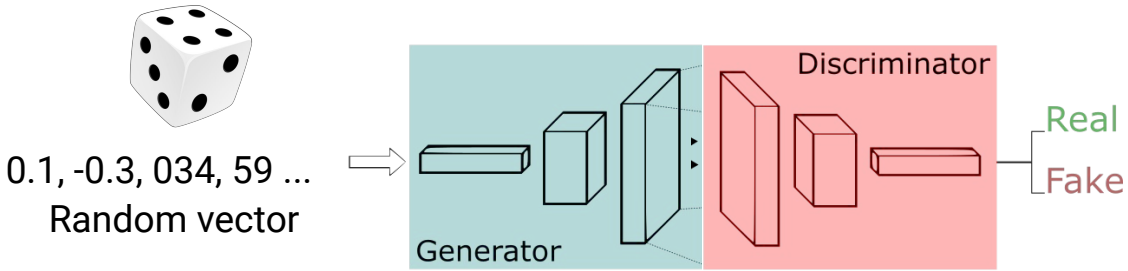
- Для дискриминатора -- все просто
  - бинарная классификации: 0 -- fake, 1 --real
  - Кросс энтропия, логлосс
- Лосс генератора должен быть тем меньше чем хуже работает дискриминатор.
  - Генератор пытается его “обмануть”
  - Задача -- обратная к задаче дискриминатора

Adapted from https://sthalles.github.io/intro-to-gans/



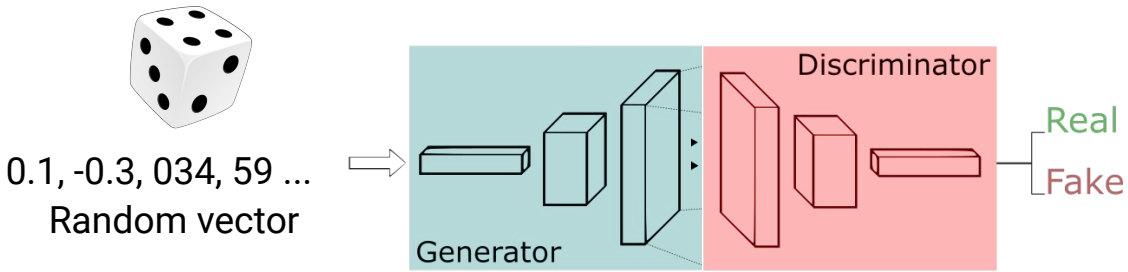
- Для дискриминатора -- все просто
  - бинарная классификации: 0 -- fake, 1 --real
  - Кросс энтропия, логлосс
- Лосс генератора должен быть тем меньше чем хуже работает дискриминатор.
  - Генератор пытается его “обмануть”
  - Задача -- обратная к задаче дискриминатора
  - Т.е. он добивается классификации **дискриминатором (!!!)** фейковых картинок к классу 1

Adapted from https://sthalles.github.io/intro-to-gans/



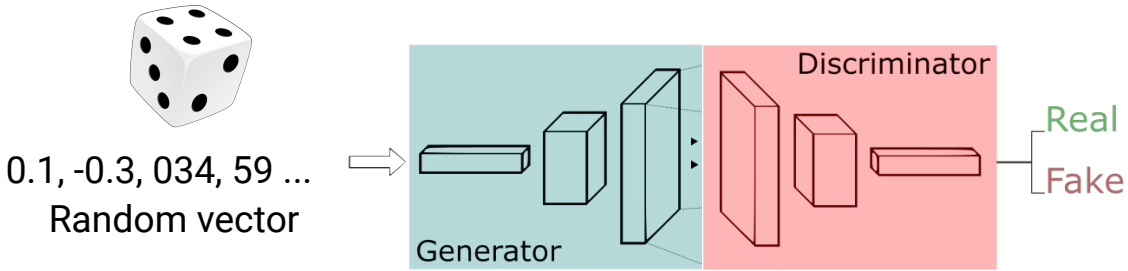
- Для дискриминатора -- все просто
  - бинарная классификации: 0 -- fake, 1 --real
  - Кросс энтропия, логлосс
- Лосс генератора должен быть тем меньше чем хуже работает дискриминатор.
  - Генератор пытается его “обмануть”
  - Задача -- обратная к задаче дискриминатора
  - Т.е. он добивается классификации **дискриминатором (!!!)** фейковых картинок к классу 1

Adapted from https://sthalles.github.io/intro-to-gans/



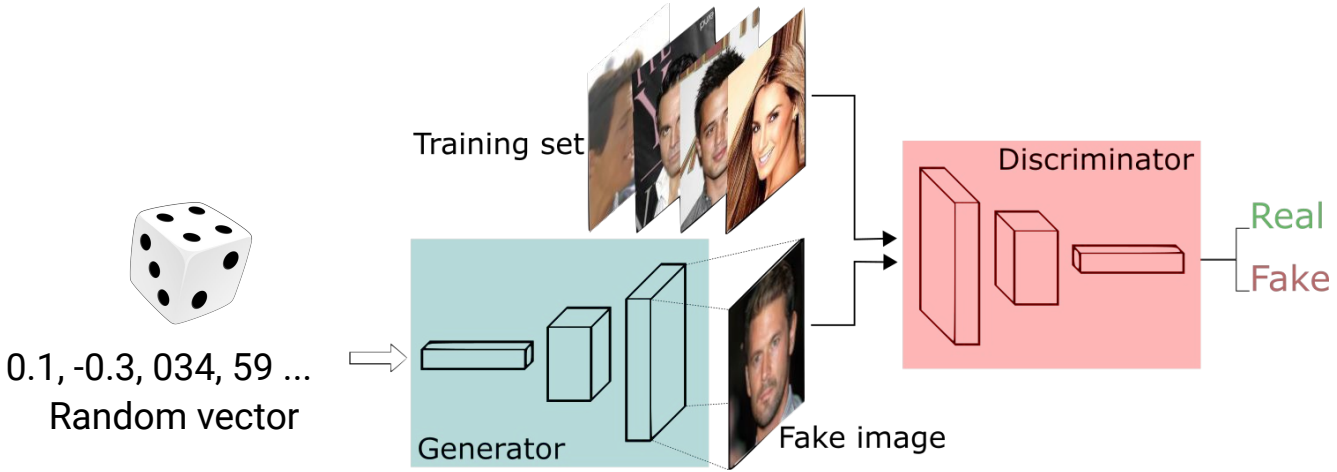
- В одном вычислительном графе -- можно пробросить ошибку дискриминатора в генератор
- И таким образом сделать генератор лучше

Adapted from https://sthalles.github.io/intro-to-gans/



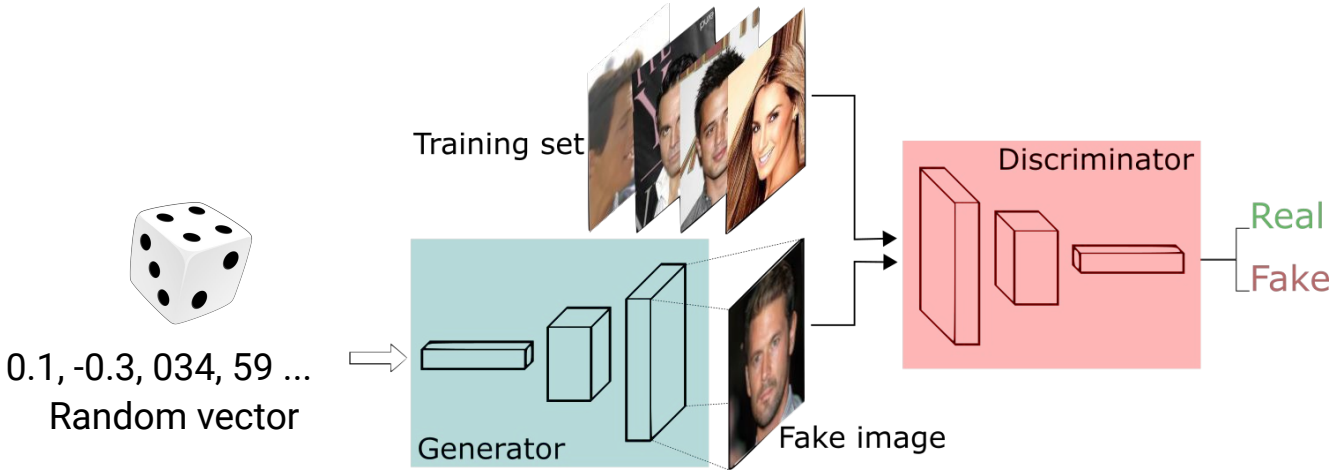
- В одном вычислительном графе -- можно пробросить ошибку дискриминатора в генератор
- И таким образом сделать генератор лучше
- Но и дискриминатор не стоит на месте.
- С каждой итерацией ему нужно поспевать за генератором -- его собственная задача классифицировать **все** правильно

Adapted from https://sthalles.github.io/intro-to-gans/



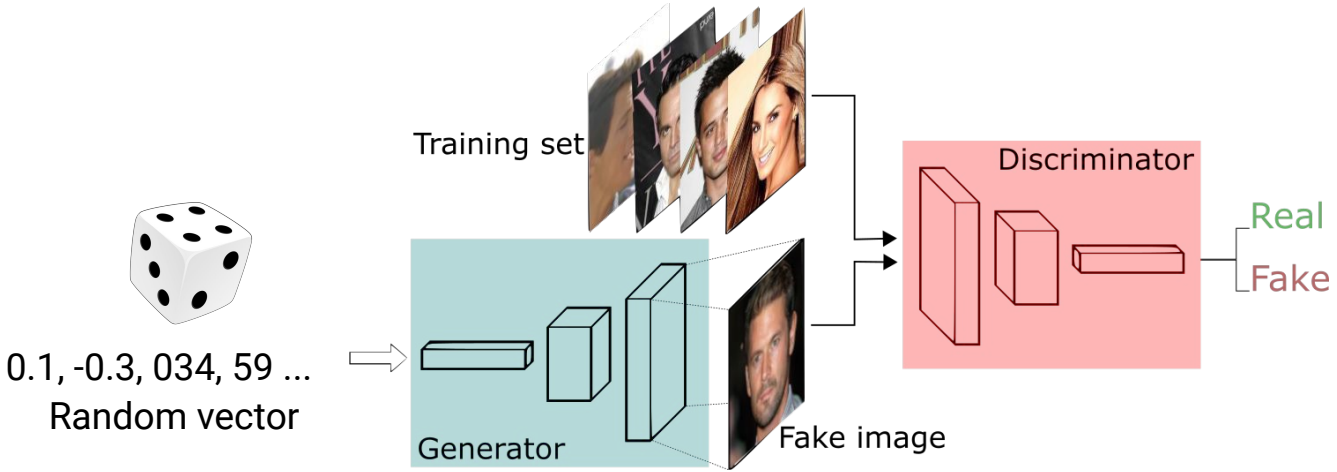
- Задача дискриминатора:
- Задача генератора:

Adapted from https://sthalles.github.io/intro-to-gans/



- Задача дискриминатора: идеально различать реальные и нереальные примеры
- Задача генератора: генерировать такие примеры, чтобы обмануть дискриминатор

Adapted from https://sthalles.github.io/intro-to-gans/



- Задача дискриминатора: идеально различать реальные и нереальные примеры
- Задача генератора: генерировать такие примеры, чтобы обмануть дискриминатор
- *Поэтому состязательные*

Adapted from https://sthalles.github.io/intro-to-gans/

Итог

- GAN -- невероятно красивая идея и ее развитие в последние годы привело к очень интересным результатам.
- Если вам что-то осталось непонятным -- не переживайте, в демонстрации кода мы еще раз все расставим по полочкам

Skillbox

Генеративные состязательные сети (GAN)

GAN в Keras

Skillbox

Генеративные состязательные сети (GAN)

GAN для генерации изображений

На прошлых уроках

- Узнали что такое распределение данных
- Узнали, что такое GAN
- Научились его обучать для простой двумерной задачи

На прошлых уроках

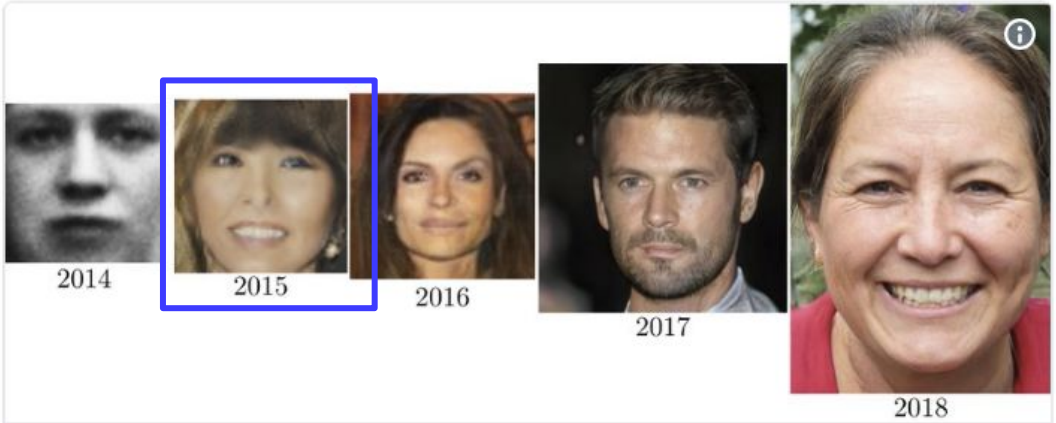
- Узнали что такое распределение данных
- Узнали, что такое GAN
- Научились его обучать для простой двумерной задачи
- Как же использовать такой подход для изображений?
  - Аналогично! Нужно только заменить полносвязные сети на сверточные

Ian Goodfellow
@goodfellow_ian

4.5 years of GAN progress on face generation.
arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948

♡ 3,612 3:40 AM - Jan 15, 2019

1,384 people are talking about this



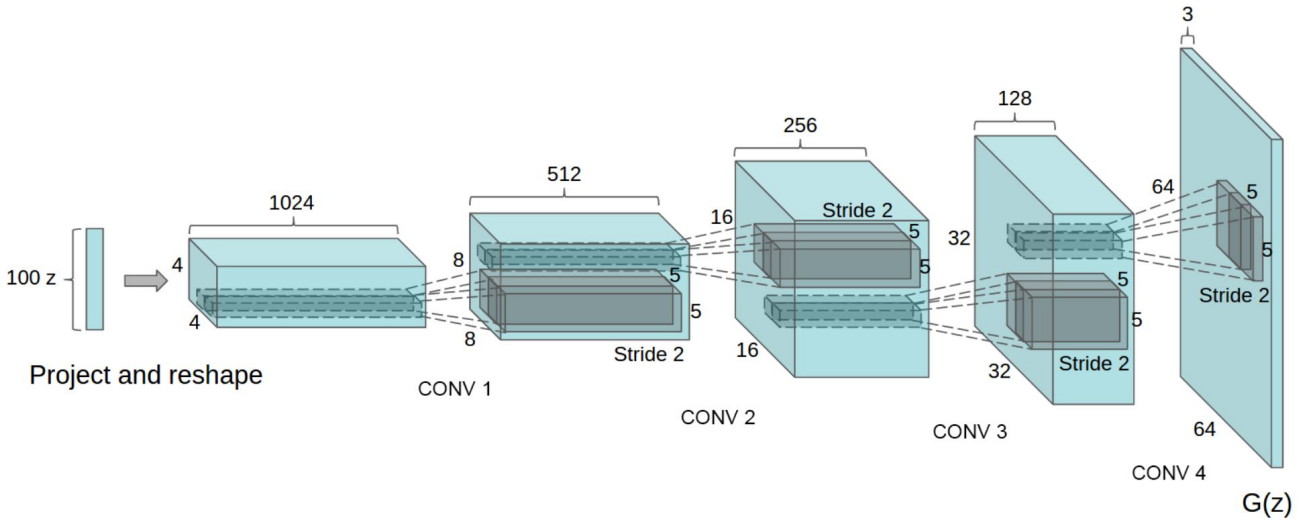
Ian Goodfellow
@goodfellow_ian

4.5 years of GAN progress on face generation.
arxiv.org/abs/1406.2661 arxiv.org/abs/1511.06434
arxiv.org/abs/1606.07536 arxiv.org/abs/1710.10196
arxiv.org/abs/1812.04948

♡ 3,612 3:40 AM - Jan 15, 2019

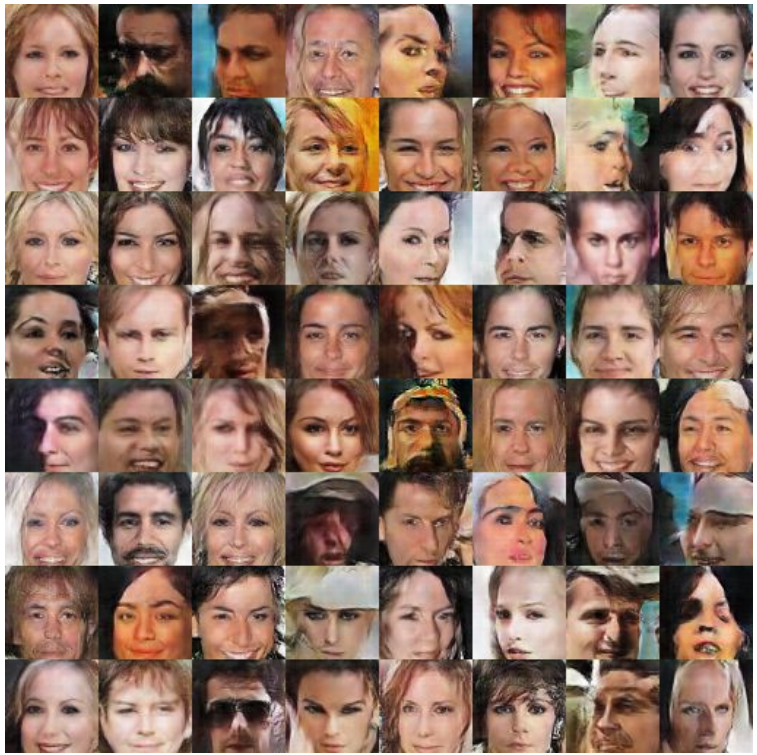
1,384 people are talking about this

DCGAN



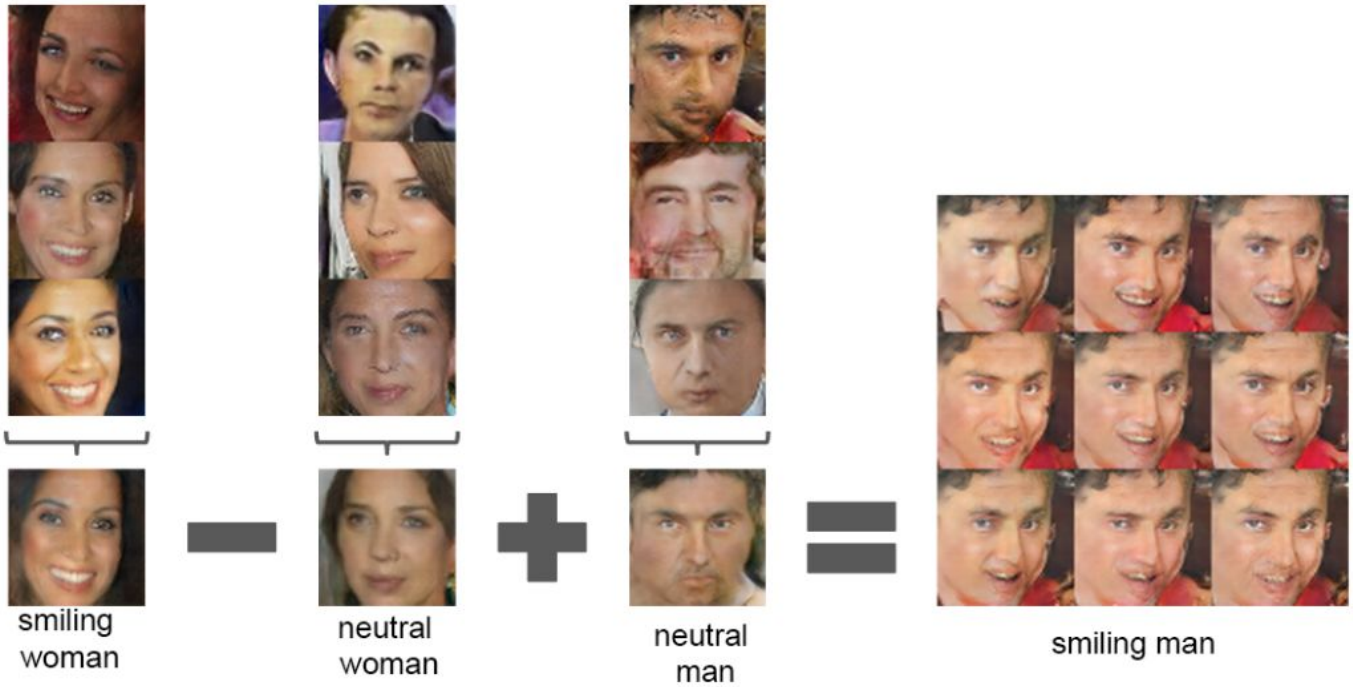
Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, Alec Radford, Luke Metz, Soumith Chintala

DCGAN



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

DCGAN



Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, Alec Radford, Luke Metz, Soumith Chintala