

# ГАНы

Генеративные модели для изображений

Шипилов Фома

ВШЭ

16 ноября 2021 г.

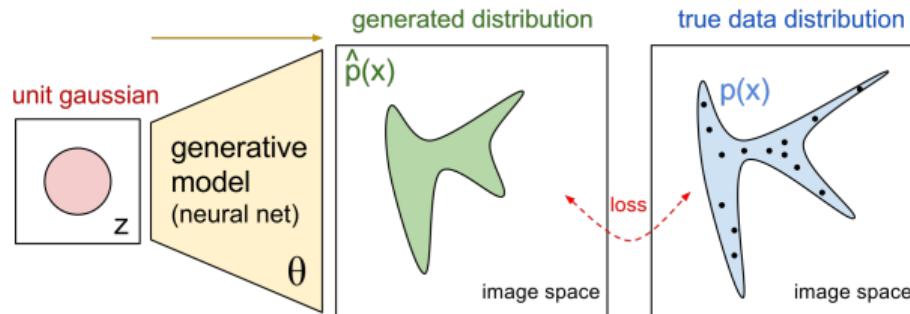
# Что такое генеративные модели

Формально:

- Статистические модели, моделирующие вероятностное распределение  $P(X, Y)$  или  $P(X|Y)$ ;

Простыми словами:

- Модели, позволяющие реалистично генерировать исходы случайной величины.



## ГАНы

**GAN** (Generative Adversarial Network, Генеративно-состязательная сеть) — один из самых популярных видов генеративных моделей.

- Предложен в [1406.2661](#) учеными из Монреальского университета;
- Состоит из двух соревнующихся обучаемых частей: *генератора* и *дискриминатора*;
- Обучение представляет собой минимакс для двух игроков.

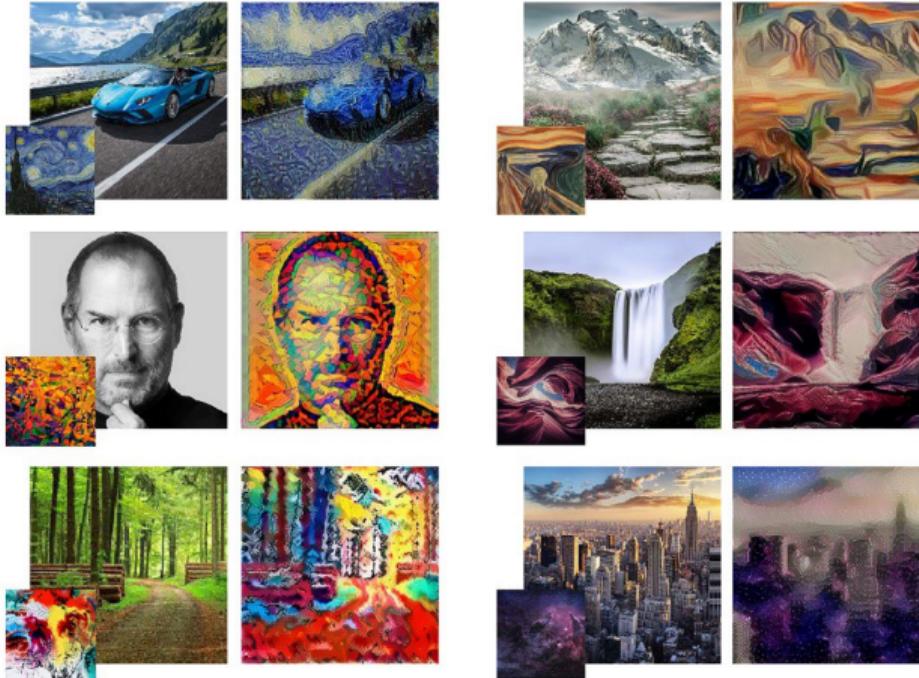
# Области применения ГАНов

- Генерация реалистичных изображений и манипуляция ими;



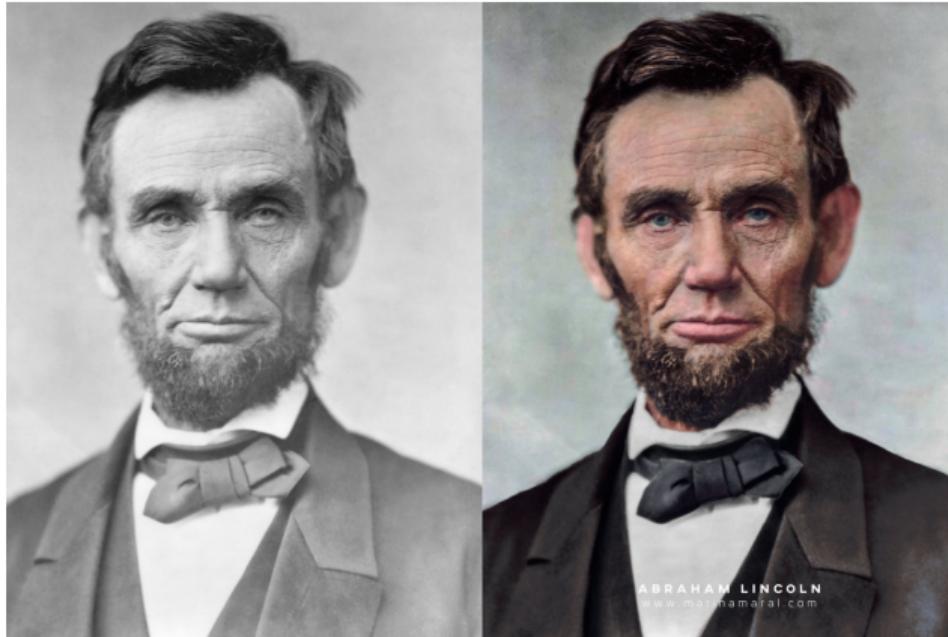
# Области применения ГАНов

- Перенос стиля;



# Области применения ГАНов

- Колоризация;



# Области применения ГАНов

- Увеличение цифрового разрешения (Super-resolution);



# Области применения ГАНов

- Интерполяция;
- Дип-фейки;
- И многое другое.



# Устройство ГАНов

Главная идея обучения ГАН:

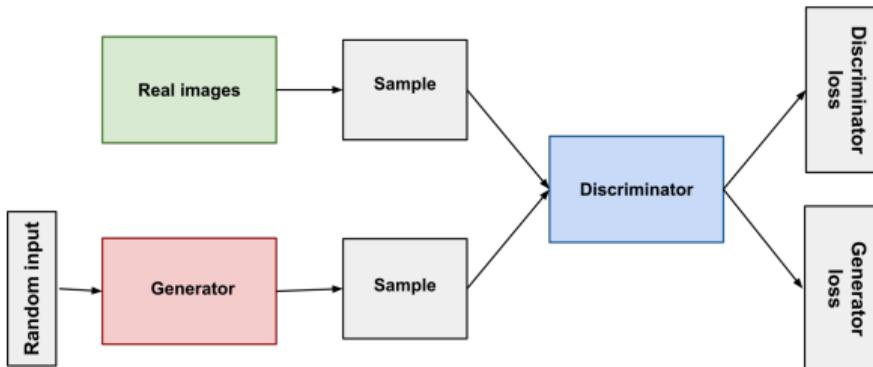
- Генератор ( $G$ ) создает *синтетические* (*fake*) данные;
- Дискриминатор ( $D$ ) учится классифицировать подаваемые ему данные на *настоящие* (*real*) и синтетические;
- Градиент от  $D$  используется для обучения  $G$ , который учится обманывать  $D$  и выдавать семплы, не отличающиеся от настоящих с точки зрения последнего;
- $G$  и  $D$  обучаются по очереди, с каждой итерацией улучшая свои способности.



# Устройство ГАНов

Основные компоненты ГАН:

- Латентное пространство;
- Генератор;
- Дискриминатор;
- Дискриминаторная функция потерь;
- Генераторная функция потерь.

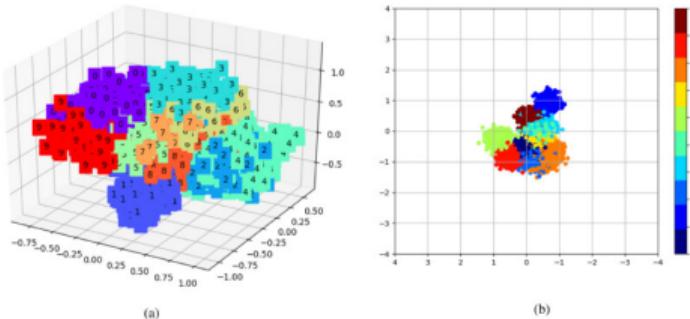


## Латентное пространство

- Генератор детерминистичен, при этом он должен моделировать случайную величину;
- Необходимо подавать на вход источник случайности;
- Обыкновенно это Гауссов случайный вектор, но можно взять любую другую случайную величину;
- Распределение, из которого берется входная случайная величина, называется **латентным пространством**;

# Латентное пространство

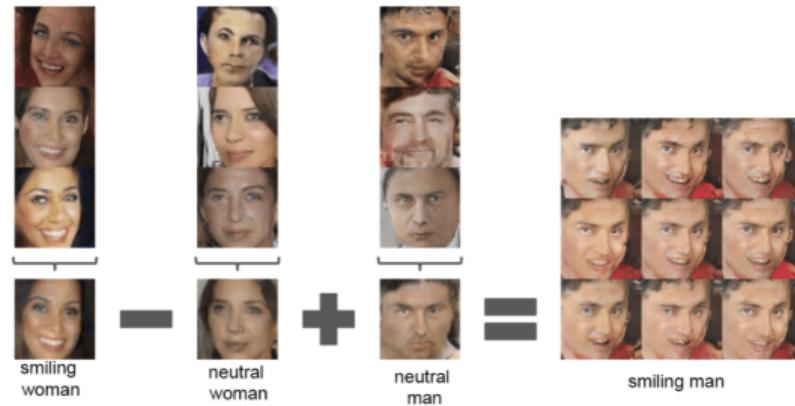
- Генератор детерминистичен, при этом он должен моделировать случайную величину;
- Необходимо подавать на вход источник случайности;
- Обыкновенно это Гауссов случайный вектор, но можно взять любую другую случайную величину;
- Распределение, из которого берется входная случайная величина, называется **латентным пространством**;
- Большое преимущество ГАНов в том, что латентное пространство хорошо интерпретируемо.



# Латентное пространство

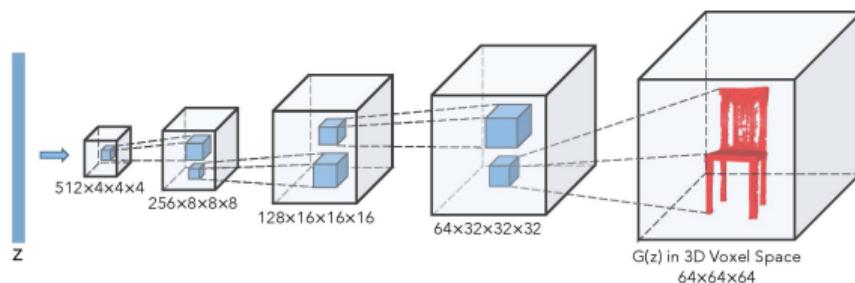
## Интуиция:

- Латентное пространство — это определенные высокоуровневые признаки генерируемых объектов;
- ГАН самостоятельно придает смысловую структуру латентному пространству, и при каждом новом запуске обучения она получается разной;
- Существуют методы извлечения осмысленных подпространств из латентного пространства.



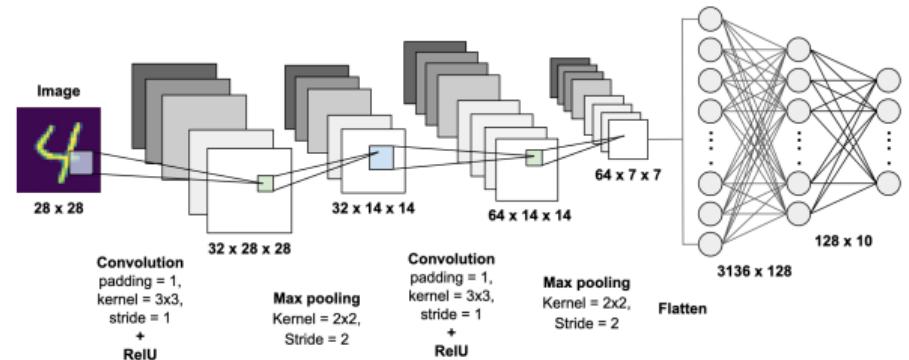
# Генератор

- Генератор задает *условное распределение*  $P(X|Z = z)$ , где  $z$  берется из латентного пространства,  $X$  — моделируемая с.в.;
- Может быть любой дифференцируемой функцией, на практике — *сверточная нейронная сеть*;
- Чтобы создать картинку из вектора, используется операция, обратная *свертке со страйдом*, увеличивающая размерность входа: ConvTranspose, PixelShuffle, ...;
- Чем больше емкость генератора, тем более сложное распределение он может моделировать.



# Дискриминатор

- Дискриминатор задает условное распределение  $P(Y|X = x)$ , где  $x$  — выход генератора либо настоящие данные,  $Y$  — величина, показывающая, насколько реалистичен сэмпл  $X$ ;
- Существует множество вариантов определения величины  $Y$ , оригинальный метод создателей архитектуры интерпретировал  $Y$  как выход *бинарного классификатора*.
- Может быть любой дифференцируемой функцией, на практике — сверточная нейронная сеть;



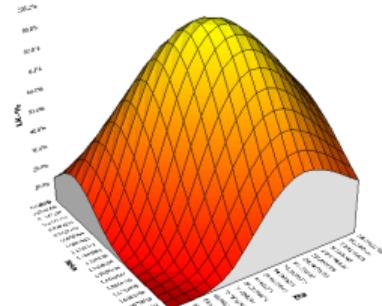
## Дискриминаторная функция потерь

- Если дискриминатор — бинарный классификатор, то обучать его можно с помощью *логарифмической функции потерь* (log-loss) из метода максимального правдоподобия:

$$\text{loss}_D(x, y) = -y \cdot \log D(x) - (1 - y) \log(1 - D(x))$$

- Класс 1 — настоящие данные, класс 0 — синтетические.

Likelihood Function Surface

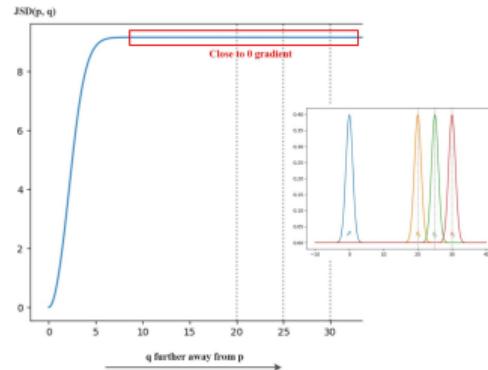


# Генераторная функция потерь

- Генератор должен обманывать дискриминатор, поэтому его функция потерь должна максимизировать ошибку дискриминатора;
- Для случая классификации функция потерь генератора есть

$$\text{loss}_G(z) = \log(1 - D(G(z)))$$

- Можно доказать, что обучение ГАН эквивалентно оптимизации JS-расстояния:



## Генераторная функция потерь

- Генератор должен обманывать дискриминатор, поэтому его функция потерь должна максимизировать ошибку дискриминатора;
- Для случая классификации функция потерь генератора есть

$$\text{loss}_G(z) = \log(1 - D(G(z)))$$

- Но такая функция очень плоха на ранней стадии обучения из-за близкого к нулю градиента, поэтому даже создатели архитектуры были вынуждены наплевать на математическую строгость и использовать другую функцию потерь:

$$\text{loss}_G(z) = -\log(D(G(z))) \tag{1}$$

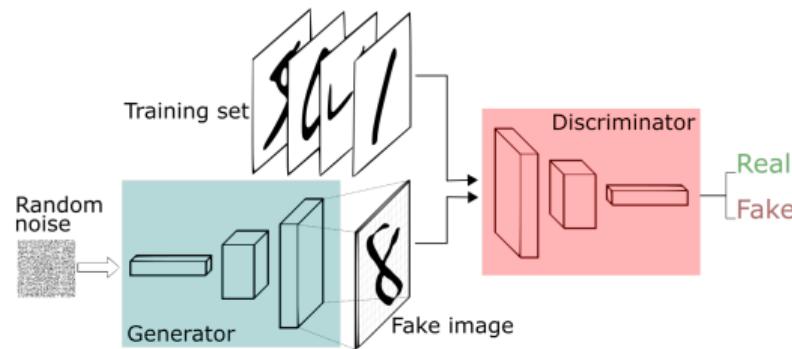
- ML такой ML :)

# Обучение ГАНов

- В ходе обучения фазы генератора и дискриминатора чередуются:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim \text{Real}} [\log D(x)] + \mathbb{E}_{z \sim \text{Latent}} [\log(1 - D(G(z)))]$$

- Когда обучается генератор, замораживаются веса дискриминатора, а когда обучается дискриминатор — градиенты отрезаются от генератора, чтобы его веса не менялись;
- Главной проблемой обучения является правильная балансировка длительностей фаз и гиперпараметров скорости обучения, т. к. ГАНы очень *нестабильны*.



# Преимущества ГАНов

- Можно обучать градиентным спуском *end-to-end*;
- Не нужны сложные методы, использующиеся в других генеративных моделях: цепи Маркова, инференсные сети и т. д.
- Отсутствуют некоторые из трейд-оффов VAE;
- Интерпретируемая структура латентного пространства;
- Огромный простор для модификаций всех элементов архитектуры.



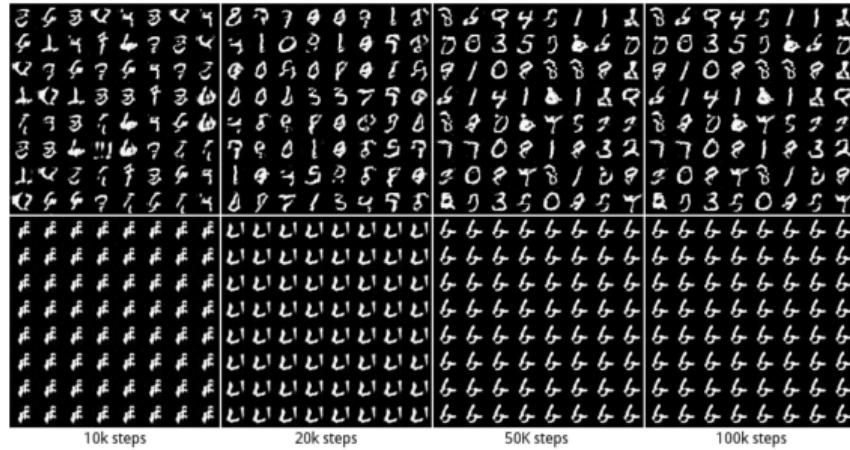
# Проблемы ГАНов

- Очень нестабильное обучение:
  - ❶ Коллапс мод (mode collapse) генератора;
  - ❷ Исчезающие градиенты;
  - ❸ Генераторная функция потерь (1) нестабильна и приводит к большим колебаниям.
- Нет теоретической гарантии сходимости, может вообще не сойтись, а может сойтись к чему-то странному;
- Дискриминатор выбирает для классификации не логичные для человека признаки реалистичности, а какие-то свои, зачастую ошибочные.



# Коллапс мод

- В стандартном ГАНе ничто не может помешать генератору выдавать всегда одну и ту же реалистичную картинку для всех значений латентной переменной;
- Такое случается при слишком долгом обучении генератора без обновления дискриминатора;
- Это и есть коллапс мод — латентное пространство схлопывается в небольшое число значений.



## Коллапс мод

- В стандартном ГАНе ничто не может помешать генератору выдавать всегда одну и ту же реалистичную картинку для всех значений латентной переменной;
- Такое случается при слишком долгом обучении генератора без обновления дискриминатора;
- Это и есть коллапс мод — латентное пространство схлопывается в небольшое число значений.

Бороться с ним можно уменьшением времени и скорости обучения генератора, а так же изменением функций потерь на WGAN (почитайте сами) или LSGAN (приводим формулы):

$$\text{loss}_D^{\text{LS}}(x, y) = y \cdot \frac{1}{2}(D(x) - 1)^2 + (1 - y) \cdot \frac{1}{2}D(x)^2$$

$$\text{loss}_G^{\text{LS}}(z) = \frac{1}{2}(D(G(z)) - 1)^2$$

## Исчезающие градиенты

- Сценарий, противоположный коллапсу мод — переобучение дискриминатора, который начинает отвергать любые результаты генератора;
- Если это происходит, то генератор перестает обучаться — все изменения дискриминатор сразу же выучивает и снова отвергает;
- Такое случается, если распределение данных слишком сложно для выбранного размера генератора и при долгом обучении дискриминатора без обновления генератора.

## Исчезающие градиенты

- Сценарий, противоположный коллапсу мод — переобучение дискриминатора, который начинает отвергать любые результаты генератора;
- Если это происходит, то генератор перестает обучаться — все изменения дискриминатор сразу же выучивает и снова отвергает;
- Такое случается, если распределение данных слишком сложно для выбранного размера генератора и при долгом обучении дискриминатора без обновления генератора.

Бороться с этим можно ограничением обучения дискриминатора, то есть пропускать его фазу, если он слишком хорошо распознает семплы. Можно уменьшать скорость обучения дискриминатора, а так же изменять архитектуру дискриминатора.

## Relativistic дискриминатор

- Один из множества способов улучшить дискриминатор — изменить определение  $\mathbb{Y}$  на логистическую регрессию, предсказывающую относительную реалистичность;
- На вход дискриминатора одновременно подаются настоящие и синтетические данные, а на выходе расчитывается то, насколько настоящий семпл реалистичнее синтетического:

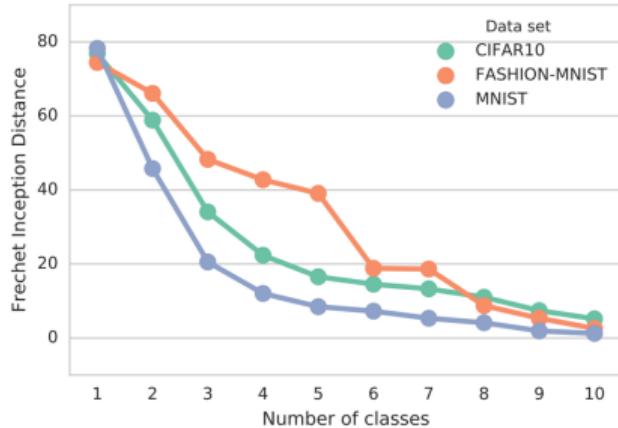
$$\text{loss}_D^{\text{RA}}(x_r, x_f) = -\log(\sigma(D(x_r) - D(x_f)))$$

$$\text{loss}_G^{\text{RA}}(x_r, x_f) = -\log(\sigma(D(x_f) - D(x_r)))$$

- Функция потерь для генератора становится полной, то есть в ней участвуют оба вида данных. Relativistic дискриминатор не только увеличивает стабильность, но также борется с коллапсом мод и позволяет получить более качественные результаты.

## Метрики качества

- Метрики оценки качества работы ГАНов основаны на различных расстояниях между вероятностными распределениями: KL-дивергенции, полном расстоянии вариации, JS-расстоянии (корень из JS-дивергенции), метрике Вассерштейна и т. д.;
- Пример: метрика FID (основана на метрике Вассерштейна).



# Заключение

- Огромные возможности ГАНов во многом компенсируются обширным списком недостатков;
- Из-за моды многие стали использовать ГАНы там, где они не нужны;
- Баланс гиперпараметров ГАНов очень тонок, поэтому требует значительно большего времени на настройку по сравнению с обычными нейросетями, обучающимися с учителем;
- Общее правило такое — если есть возможность сделать задачу без них, то нужно ей пользоваться.



# Список источников

- ① Generative Adversarial Networks
- ② Google Overview of GAN Structure
- ③ GAN — Why it is so hard to train Generative Adversarial Networks!
- ④ Wasserstein GAN
- ⑤ Least Squares Generative Adversarial Networks
- ⑥ GAN — LSGAN (How to be a good helper?)
- ⑦ The relativistic discriminator: a key element missing from standard GAN
- ⑧ Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- ⑨ GAN: Create 2D map tiles with a Generative Adversarial Network
- ⑩ A Survey of Image Synthesis and Editing with Generative Adversarial Networks
- ⑪ A Survey on Generative Adversarial Networks: Variants, Applications, and Training