

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ  
БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики и информатики**

**Кафедра дискретной математики и алгоритмики**

**Ходор Иван Андреевич**

**Применение генеративно-состязательных сетей для генерации  
изображений спортивной обуви**

Курсовая работа  
студента 3 курса 3 группы

”Допустить к защите”

\_\_\_\_\_

“ \_\_\_\_ ” \_\_\_\_\_ 2021г

**Научный руководитель**

Вильчевский К.Ю.

старший преподаватель кафедр  
ры ДМА

**МИНСК 2021**

# Содержание

<b>Введение</b>	<b>3</b>
<b>1 Рассмотрение метода</b>	<b>4</b>
1.1 Принцип работы . . . . .	4
1.1.1 Пример на датасете MNIST . . . . .	4
1.2 Типичные проблемы при обучении . . . . .	6
1.2.1 Коллапс мод . . . . .	7
1.2.2 Проблема стабильности обучения . . . . .	8
1.3 Основные достижения в применении генеративно- состязательных сетях . . . . .	9
1.4 Некоторые модификации архитектуры . . . . .	11
1.4.1 Условные порождающие состязательные сети(CGAN) . .	11
1.4.2 Глубокие свёрточные генеративно-состязательные се- ти(DCGAN) . . . . .	13
1.4.3 Несколькоуровневые генеративно-состязательные се- ти(StackGAN) . . . . .	14
1.4.4 Контролируемые генеративно-состязательные се- ти(ControlGAN) . . . . .	15
1.5 Сравнение с некоторыми другими нейронными сетями . . . . .	17
<b>2 Обучение генеративно-состязательной сети для генерации изображений</b>	<b>19</b>
2.1 Постановка задачи . . . . .	19
2.2 Генерация изображений низкого качества . . . . .	19
2.2.1 Стандартная модель . . . . .	19
2.2.2 Глубокая свёрточная генеративно-состязательная сеть . .	21
2.3 Улучшение качества изображений . . . . .	23
2.3.1 Немного о задаче транслирования изображений . . . . .	23
2.3.2 Обучение модели . . . . .	24
<b>Заключение</b>	<b>28</b>



# Введение

Вряд ли вы подумаете, что программиста можно назвать артистом, но, на самом деле, программирование это очень творческая профессия. Творчество, основанное на логике.

---

Джон Ромеро

Генеративно-сопоставительная сеть(англ. Generative adversarial network, сокращённо GAN) — алгоритм машинного обучения без учителя, построенный на комбинации из двух нейронных сетей, одна из которых(сеть G) генерирует образцы, а другая(сеть D) старается отличить правильные(«подлинные») образцы от неправильных. Так как сети G и D имеют противоположные цели — создать образцы и отбраковать образцы — между ними возникает антагонистическая игра. Генеративно-сопоставительную сеть описал Ян Гудфеллоу из компании Google в 2014 году.

# Глава 1

## Рассмотрение метода

### 1.1 Принцип работы

В системе GAN одна из сетей(сеть G) генерирует образцы, а другая(сеть D) старается отличить правильные(«подлинные») образцы от неправильных. Используя некоторый набор переменных, генеративная сеть пытается создать новый образец, смешав несколько исходных образцов. Дискриминативная сеть обучается различать подлинные и поддельные образцы, а результаты различения подаются на вход генеративной сети так, чтобы она смогла подобрать лучший набор параметров, и дискриминативная сеть уже не смогла бы отличить подлинные образцы от поддельных. Таким образом целью сети G является повысить процент ошибок сети D, а целью сети D является наоборот улучшение точности распознавания. Сеть D реализуется как свёрточная нейронная сеть, в то время как сеть G наоборот разворачивает изображение на базе скрытых параметров.

В процессе совместного конкурентного обучения, если система достаточно сбалансирована, достигается состояние равновесия, в котором обе сети значительно улучшили своё качество, и теперь сгенерированные изображения могут быть использованы практически как настоящие.

#### 1.1.1 Пример на датасете MNIST

Сгенерируем рукописные цифры, подобные тем, что имеются в наборе данных MNIST. Цель дискриминатора — распознать подлинные экземпляры из набора.

Между тем, генератор создает новые изображения, которые он передает дискриминатору. Он делает это в надежде, что они будут приняты подлинными, хотя являются поддельными. Цель генератора состоит в том, чтобы генерировать рукописные цифры, которые будут пропущены дискриминатором. Цель дискриминатора — определить, является ли изображение подлинным.

Шаги, которые проходит GAN:

- Генератор получает случайное число и возвращает изображение;
- Это сгенерированное изображение подается в дискриминатор наряду с потоком изображений, взятых из фактического набора данных;
- Дискриминатор принимает как реальные, так и поддельные изображения и возвращает вероятности, числа от 0 до 1, причем 1 представляет собой подлинное изображение и 0 представляет фальшивое.

Таким образом, у нас есть двойной цикл обратной связи:

- Дискриминатор находится в цикле с достоверными изображениями;
- Генератор находится в цикле вместе с дискриминатором.

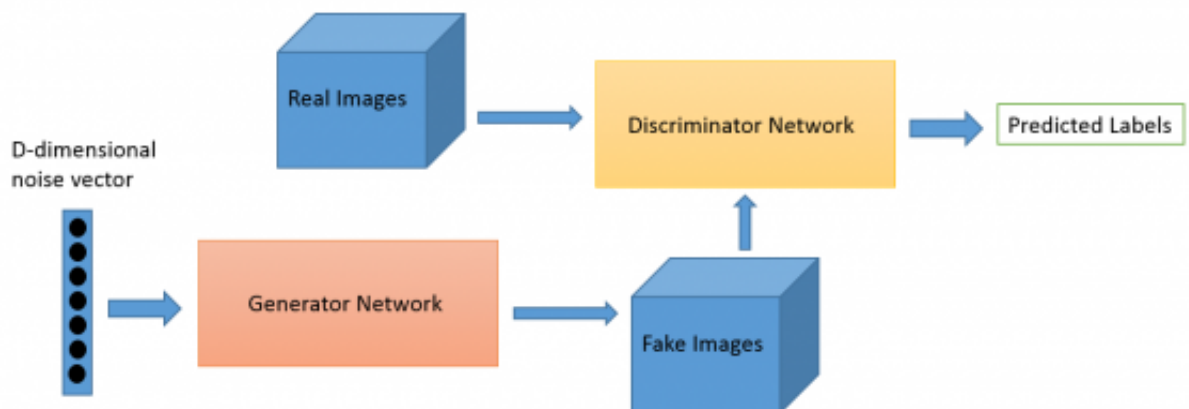


Рисунок 1.1 — Архитектура GAN.

Можно представить GAN как фальшивомонетчика и полицейского, играющих в кошки мышки, где фальсификатор учится изготавливать ложные купюры, а полицейский учится их обнаруживать. Оба динамичны; т. е. полицейский тоже тренируется (возможно, центральный банк отмечает пропущенные купюры), и каждая сторона приходит к изучению методов другого в постоянной эскалации.

Сеть дискриминаторов представляет собой стандартную сверточную сеть, которая может классифицировать изображения, подаваемые на нее с помощью биномиального классификатора, распознающего изображения как реальные или как поддельные. Генератор в некотором смысле представляет

собой обратную сверточную сеть: хотя стандартный сверточный классификатор принимает изображение и уменьшает его разрешение, чтобы получить вероятность, генератор принимает вектор случайного шума и преобразует его в изображение.

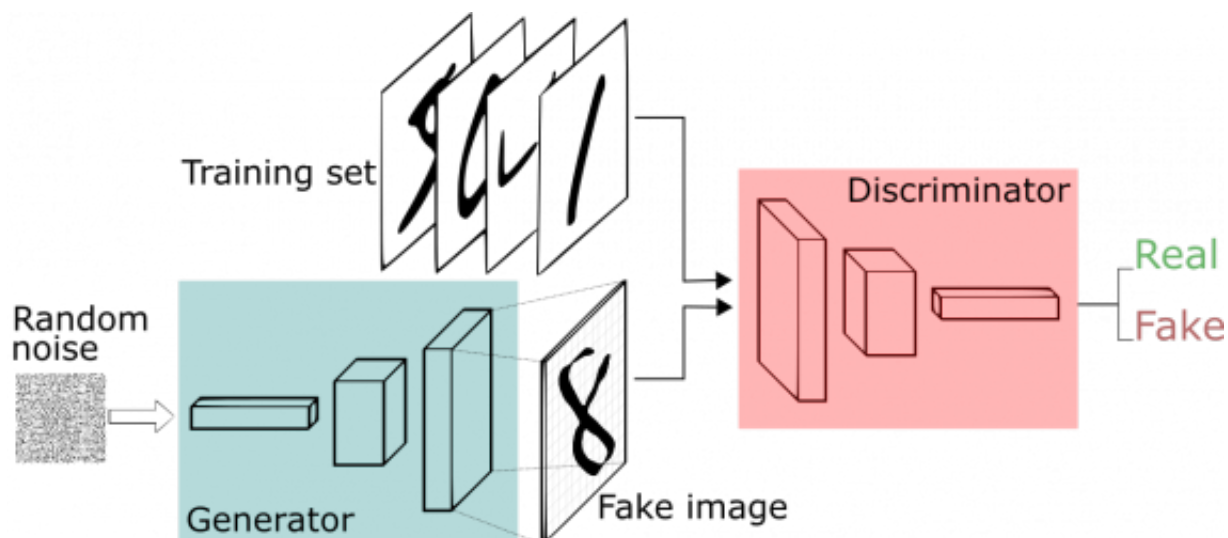


Рисунок 1.2 — Примерная схема работы на датасете MNIST.

Обе сети пытаются оптимизировать целевую функцию или функцию потерь в игре zero-sum. Это, по сути, модель актера-критика (actor-critic). Когда дискриминатор меняет свое поведение, то и генератор меняет, и наоборот.

## 1.2 Типичные проблемы при обучении

Большинство GAN'ов подвержено следующим проблемам:

- Схлопывание мод распределения(англ. mode collapse): генератор коллапсирует, то есть выдает ограниченное количество разных образцов.
- Проблема стабильности обучения(англ. non-convergence): параметры модели дестабилизируются и не сходятся.
- Исчезающий градиент(англ. diminished gradient): дискриминатор становится слишком "сильным" а градиент генератора исчезает и обучение не происходит.
- Проблема запутывания(англ. disentanglement problem): выявление корреляции в признаках, не связанных(слабо связанных) в реальном мире.

- Высокая чувствительность к гиперпараметрам.

Рассмотрим некоторые проблемы подробнее.

### 1.2.1 Коллапс мод

В процессе обучения генератор может прийти к состоянию, при котором он будет всегда выдавать ограниченный набор выходов. При этом пространство, в котором распределены сгенерированные изображения, окажется существенно меньше, чем пространство исходных изображений. Главная причина этого в том, что генератор обучается обманывать дискриминатор, а не воспроизводить исходное распределение. Лучшей стратегией для дискриминатора будет улучшение детектирования этого конкретного изображения. Так на следующих итерациях наиболее вероятно, что генератор придет к другому изображению, хорошо обманывающему текущий дискриминатор, а дискриминатор будет учиться отличать конкретно это новое изображение. Этот процесс не будет сходиться и количество представленных мод не будет расти, поэтому приблизиться к исходному распределению не удастся.

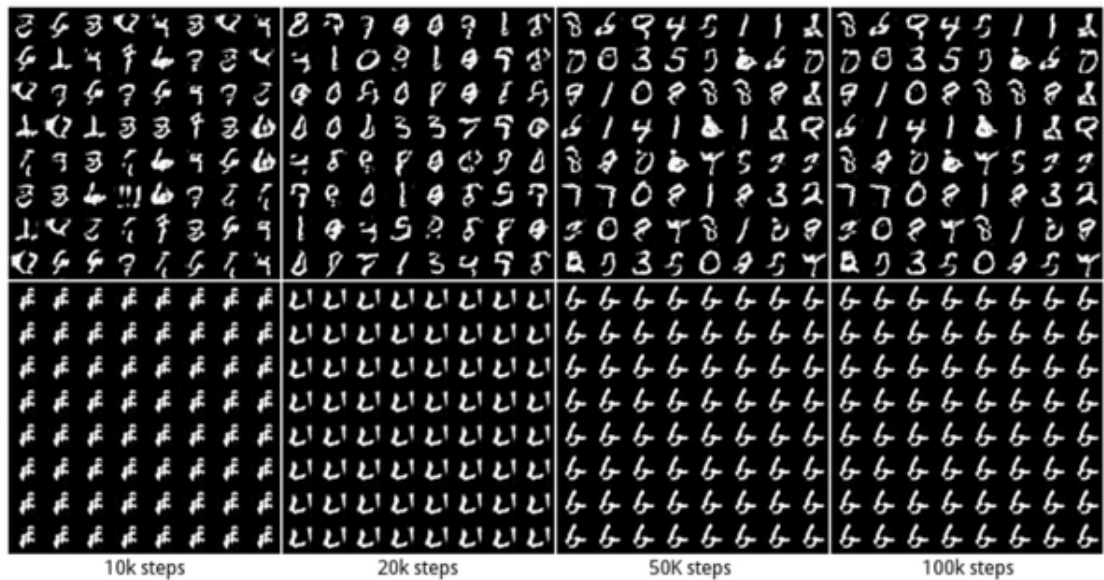


Рисунок 1.3 — Проблема коллапса мод в GAN сетях на примере MNIST датасета, на нижнем ряду представлен обычный GAN, на верхнем — UGAN с 20 обратными шагами.



На (1.3) наглядно представлен пример описанной проблемы в процессе работы обычной GAN, обучаемой на датасете MNIST.

На текущий момент коллапс мод является одной из главных проблем GAN, эффективное решение которой ещё ищется. Возможные решения проблемы mode collapse:

1. WGAN — использование метрики Вассерштейна внутри функции ошибки позволяет дискриминатору быстрее обучаться выявлять повторяющиеся выходы, на которых стабилизируется генератор;
2. UGAN(Unrolled GAN(рус. откатывающаяся генеративно-сопоставительная сеть)) — для генератора используется функция потерь, которая зависит не только от того, как текущий дискриминатор оценивает выходы генератора, но и от выходов будущих версий дискриминатора.

### 1.2.2 Проблема стабильности обучения

Задача обучения дискриминатора и генератора в общем смысле не является задачей поиска локального или глобального минимума функции, а является задачей поиска точки равновесия двух игроков. В теории игр эта точка называется точкой равновесия Нэша, в которой оба игрока больше не получают выгоды, хотя следуют оптимальной стратегии. Рассмотрим задачу поиска этой точки на игрушечном примере, где  $G$  хочет максимизировать произведение  $xy$  а  $D$  — минимизировать. Будем обновлять параметры  $x$  и  $y$  на основе градиентного спуска:  $\Delta x = \alpha \frac{\delta(xy)}{\delta(x)}$ ;  $\Delta y = -\alpha \frac{\delta(xy)}{\delta(y)}$ . Если изобразить на графике поведение  $x$ ,  $y$  и  $xy$  (1.4) то станет ясно, что они не сойдутся, а амплитуда их движения будет только увеличиваться.

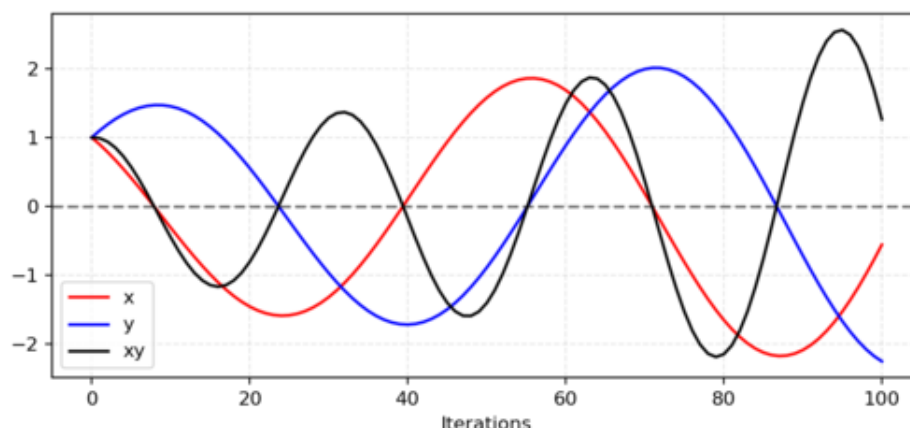


Рисунок 1.4 — Симуляция изменения  $x$  и  $y$  с помощью градиентного спуска, где изменяя  $x$  мы пытаемся минимизировать величину  $xy$ , а при изменении  $y$ , наоборот, стараемся ее максимизировать.

Возможные решения проблемы стабильности:

- Регуляризация — добавление шума ко входам дискриминатора и соответствующая настройка гиперпараметров дискриминатора;
- PGGAN(Progressive Growing of GANs) — в процессе работы разрешение изображений увеличивается от очень малых(4 на 4 пикселя) до конечных(1024 на 1024 пикселя), что позволяет тренировать сначала выявление крупных черт а затем более мелких, что крайне положительно сказывается на стабильности.

## 1.3 Основные достижения в применении генеративно-состязательных сетях

Чаще всего GAN'ы используются для генерации реалистичных фотографий.



Рисунок 1.5 — Прогресс в генерации фотографий с помощью GAN.

Серьезные улучшения в этом направлении были сделаны следующими работами:

1. Auxiliary GAN: вариант GAN-архитектуры, использующий метки данных;
2. SN-GAN: GAN с новым подходом решения проблемы нестабильного обучения через спектральную нормализацию;
3. SAGAN: GAN, основанный на механизме внимания;
4. BigGAN: GAN с ортогональной регуляризацией, позволившей разрешить проблему коллапсирования при долгом обучении;

Кроме простой генерации изображений, существуют достаточно необычные применения, дающие впечатляющие результаты не только на картинках, но и на звуке:

1. CycleGAN: меняет изображения с одного домена на другой, например, лошадей на зебр;
2. SRGAN: создает изображения с высоким разрешением из более низкого разрешения;
3. Pix2Pix: создает изображения по семантической окраске;

4. StackGAN: создает изображения по заданному тексту;
5. MidiNet: генерирует последовательность нот, таким образом, создает мелодию.

## 1.4 Некоторые модификации архитектуры

### 1.4.1 Условные порождающие состязательные сети(CGAN)

Условные порождающие состязательные сети(англ. Conditional Generative Adversarial Nets, CGAN) это модифицированная версия алгоритма GAN, которая может быть сконструирована при помощи передачи дополнительных данных  $y$ , являющихся условием для генератора и дискриминатора.  $y$  может быть любой дополнительной информацией, например, меткой класса, изображением или данными из других моделей, что может позволить контролировать процесс генерации данных. Например, можно подавать параметр  $y$ , как условие на класс для генерации чисел, похожих на MNIST. Создание таких картинок, в случае передачи картинки в качестве  $y$  является задачей трансляции изображений. Пример работы CGAN на датасете MNIST с метками классов:



Рисунок 1.6 — Цифры, сгенерированные с помощью CGAN.

При создании изображения в генератор поступает скомбинированная информация двух параметров:  $y$  и вектора шума. В случае MNIST это может быть, например, просто метка класса (от 0 до 9). На выходе из генератора получается изображение, которое комбинируется с  $y$  и поступает в дискриминатор, который в свою очередь применяет свертку, чтобы получить полносвязный слой. Наконец, анализируя полученную информацию дискриминатор принимает решение, является ли изображение сгенерированным (1.7).

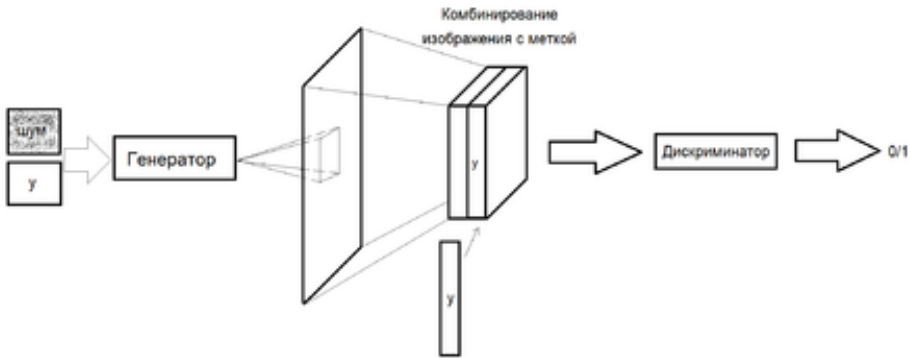


Рисунок 1.7 — Генерация при использовании CGAN.

Также, используя условные порождающие состязательные сети, можно научить такую сеть генерировать текст по картинке и наоборот. В качестве параметра  $y$  в данном случае передается изображение, которое будет описано (1.8).



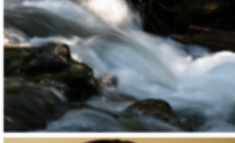

	User tags + annotations	Generated tags
	montanha, trem, inverno, frio, people, male, plant life, tree, structures, transport, car	taxi, passenger, line, transportation, railway station, passengers, railways, signals, rail, rails
	food, raspberry, delicious, homemade	chicken, fattening, cooked, peanut, cream, cookie, house made, bread, biscuit, bakes
	water, river	creek, lake, along, near, river, rocky, treeline, valley, woods, waters
	people, portrait, female, baby, indoor	love, people, posing, girl, young, strangers, pretty, women, happy, life

Рисунок 1.8 — Генерация при использовании CGAN.

Более того, для такого типа нейронных сетей, принимающих в качестве параметра у некоротое изображение местности, в результате может быть получено аналогичное изображение этого места зимой или летом, днем или ночью. Такая задача является задачей трансляции изображений.

### 1.4.2 Глубокие свёрточные генеративно-сопоставительные сети(DCGAN)

Глубокие свёрточные генеративно-сопоставительные сети(англ. Deep Convolutional Generative Adversarial Nets, DCGAN) - модификация алгоритма GAN, в основе которых лежат сверточные нейронные сети. Задача поиска удобного представления признаков на больших объемах неразмеченных данных является одной из наиболее активных сфер исследований, в частности представление изображений и видео. Одним из удобных способов поиска представлений может быть DCGAN(1.9).

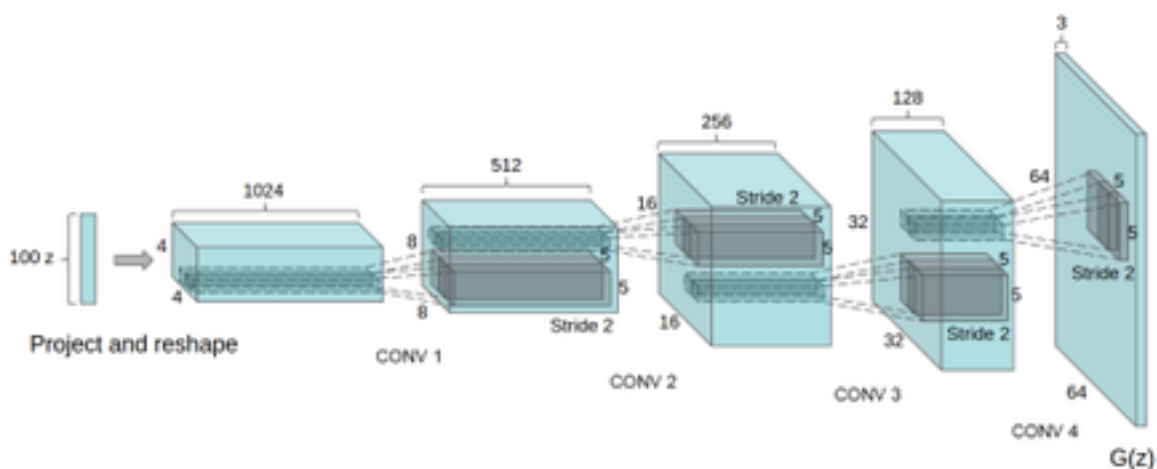


Рисунок 1.9 — Архитектура генератора в DCGAN.

Использование сверточных нейронных сетей напрямую не давало хороших результатов, поэтому было внесены ограничения на слои свертки. Эти ограничения и лежат в основе DCGAN:

- Замена всех пулинговых слоев на страйдинговые свертки(strided convolutions) в дискриминаторе и частично-страйдинговые свертки

(fractional-strided-convolutions) в генераторе, что позволяет сетям находить подходящие понижения и повышения размерностей;

- Использование батч-нормализации для генератора и дискриминатора, то есть нормализация входа так, чтобы среднее значения было равно нулю и дисперсия была равна единице. Не стоит использовать батч-нормализацию для выходного слоя генератора и входного дискриминатора;
- Удаление всех полносвязных скрытых уровней для более глубоких архитектур;
- Использование ReLU в качестве функции активации в генераторе для всех слоев, кроме последнего, где используется tanh;
- Использование LeakyReLU в качестве функции активации в дискриминаторе для всех слоев.

Помимо задачи генерации объектов, данный алгоритм хорошо показывает себя в извлечении признаков.

### 1.4.3 Несколькоуровневые генеративно-сопязательные сети(StackGAN)

StackGAN(Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks) порождающая сопязательная сеть для генерации фотореалистичных изображений исходя из текстового описания. Генерировать фотореалистичные изображения на обычных GAN сложно, поэтому была придумана двухэтапная модель генерации. Stage-I GAN рисует скетчи с примитивными формами и цветами, основанные на текстовом описании, в низком разрешении. Stage-II GAN принимает на вход изображения с первого этапа и текстовое описание и генерирует изображение в высоком разрешении с фотореалистичными деталями. Чтобы улучшить разнообразие синтезированных изображений и стабилизировать обучение использовался метод Conditioning Augmentation.



Рисунок 1.10 — Пример работы порождающей состязательной сети для генерации фотореалистичных изображений StackGAN.

Ранее использовались CGAN, поскольку на вход им можно было подавать условия. Однако просто добавляя слои, увеличивающие размер изображения, достичь хороших результатов не удалось. Поэтому основной задачей было повысить разрешение изображений.

Одной из ключевых особенностей StackGAN является Conditioning Augmentation, так как оно позволило расширить количество примеров тренировочного сета, путем небольших случайных изменений в исходных изображениях, что увеличивало многообразие данных.

#### 1.4.4 Контролируемые генеративно-состязательные сети(ControlGAN)

Контролируемые порождающие состязательные сети(англ. Controllable Generative Adversarial Nets, ControlGAN) модифицированная версия алгоритма GAN, состоящая из трех нейронных сетей: генератор, дискриминатор, классификатор(1.11). Как и в обычной версии алгоритма, генератор пытается



обмануть дискриминатор, и одновременно с этим пытается быть классифицированным как нужный класс в классификаторе.

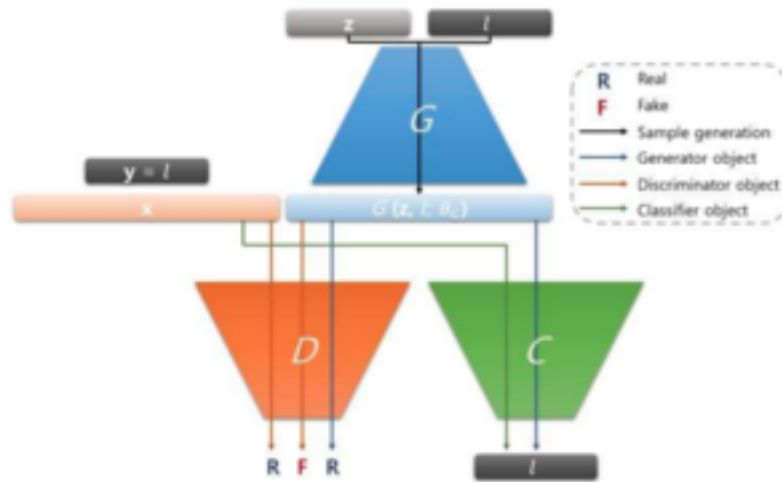


Рисунок 1.11 — Концепт модели ControlGAN.

Хоть CGAN и являются самыми популярными моделями для генерации образцов, зависящих от внешних данных, но лучше они умеют генерировать образцы с заданными ярко отличительными чертами(цвет волос, веснушки), но менее явные детали(форма бровей, сережки) вызывают затруднения(хотя более поздний StyleGAN2 справляется и с этой задачей). С помощью отделения классификатора от дискриминатора, ControlGAN позволяет контролировать черты образцов. К тому же и само качество сгенерированных изображений может быть улучшено за счет того, что такое разделение на три составляющие дает возможность дискриминатору лучше выполнять свою главную задачу.

Более того, аугментация данных может помешать некоторым сетям, например, Auxiliary Classifier GAN(ACGAN), обучаться, хотя сам способ может улучшить качество классификации. К тому же в случае контролируемой генерации нет необходимости размечать тренировочные данные, выбираются желаемые характеристики объектов для генерации, а не условная информация(например, метка объекта).

## 1.5 Сравнение с некоторыми другими нейронными сетями

Полезно сравнить генеративные состязательные сети с другими нейронными сетями, такими как автокодеры(автоэнкодеры) и вариационные автокодеры(VAE).

Автокодеры кодируют входные данные в векторы. Они создают скрытое или сжатое представление необработанных данных. Они полезны при уменьшении размерности: вектор, служащий в качестве скрытого представления, сжимает необработанные данные в меньшее количество. Автокодеры могут быть сопряжены с так называемым декодером, который позволяет восстанавливать входные данные на основе их скрытого представления.

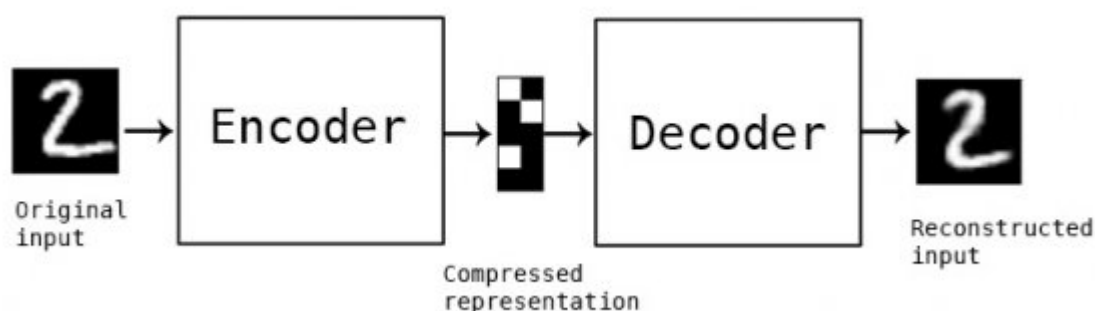


Рисунок 1.12 — Схема работы автокодера.

Вариационные автокодеры являются генеративным алгоритмом, который добавляет дополнительное ограничение для кодирования входных данных, а именно то, что скрытые представления нормализуются. Вариационные автокодеры способны сжимать данные как автокодеры и синтезировать данные подобно GAN. Однако, в то время как GAN генерируют данные детализовано, изображения, созданные VAE, бывают более размытыми. Примеры DeepLearning4j включают в себя как автокодеры, так и вариационные автокодеры.

Можно разделить генеративные алгоритмы на три типа, которые имеют:

- категорию, предсказывают связанные функции(Naive Bayes);
- скрытое представление, предсказывают связанные функции(VAE, GAN);

- некоторые образы, предсказывают остальное (inpainting, imputation).

## Глава 2

# Обучение генеративно-состязательной сети для генерации изображений

## 2.1 Постановка задачи

Обучим генеративно-состязательную сеть, генерирующую изображения спортивной обуви(далее кроссовок), на датасете, находящемся в открытом доступе. Т.к. задача генерации изображения высокого качества для GAN'ов является нетривиальной, разобьём её на две: генерация изображений низкого качества(28x28 пикселей) из некоторого шума и улучшение их качества решением задачи image-to-image, где источником выступает изображение низкого качества, а результатом - изображение высокого качества(112x112 пикселей).

## 2.2 Генерация изображений низкого качества

### 2.2.1 Стандартная модель

В качестве шума выберем вектор из 32 чисел, полученных из нормального распределения с матожиданием 0 и дисперсией 1.

Оптимальная архитектура генератора, полученная в результате опытов(далее архитектура):

- Полносвязный слой с выходом размера 1024;
- ReLU;
- Полносвязный слой с выходом размера 1024;
- ReLU;
- Полносвязный слой с выходом размера 28 x 28 x 3;

- Reshape в (28, 28, 3);
- TanH(для нормализации каждого элемента в отрезок  $[-1, 1]$ ).

Архитектура дискриминатора:

- Flatten слой;
- Полносвязный слой с выходом размера 256;
- Leaky ReLU(0.01);
- Полносвязный слой с выходом размера 256;
- Leaky ReLU(0.01);
- Полносвязный слой с выходом размера 1;
- Сигмоида(чтобы получить вероятность в пределах  $[0; 1]$ ).

В качестве метрики будем использовать  $\text{loss}$ (функция потерь). Для генератора:

$$\ell_G = -\mathbb{E}_{z \sim p(z)} [\log D(G(z))]$$

и для дискриминатора:

$$\ell_D = -\mathbb{E}_{x \sim p_{\text{data}}} [\log D(x)] - \mathbb{E}_{z \sim p(z)} [\log (1 - D(G(z)))]$$

Посмотрим на выборку изображений, полученных в течении обучения:

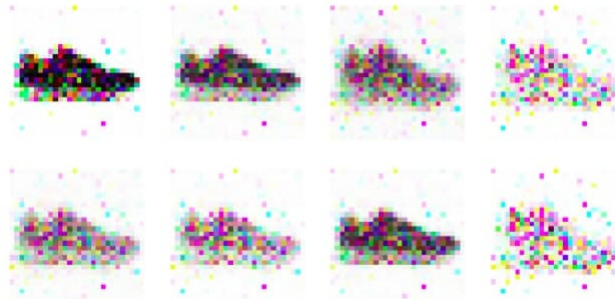


Рисунок 2.1 — Результаты генерации на разных эпохах.

И на результат работы дискриминатора:

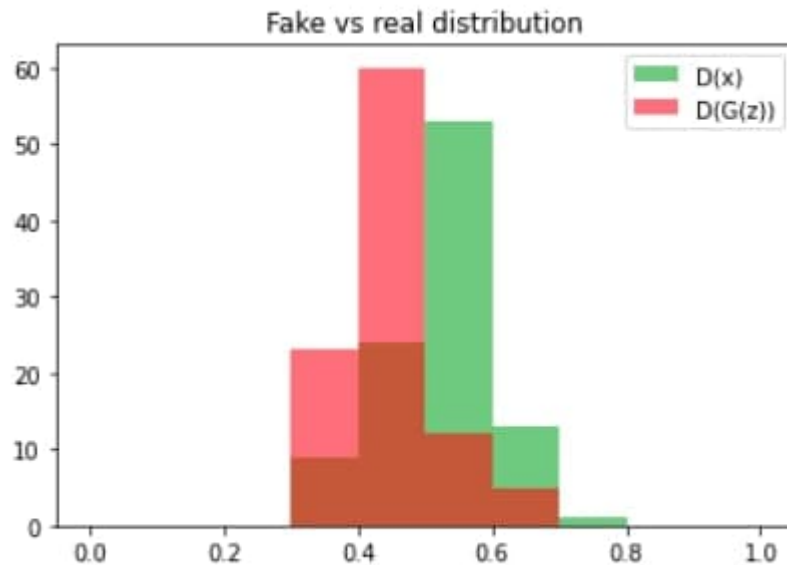


Рисунок 2.2 — "Распределение угадывания дискриминатором подлинности изображений" для стандартного GAN.

Как видим, несмотря на пересечение, дискриминатор в целом неплохо справляется с задачей отличия подлинных изображений от поддельных. Также можно сказать, что дискриминатор немного "обгоняет" генератор, что является оптимальной ситуацией для обучения GAN'ов.

## 2.2.2 Глубокая свёрточная генеративно-сопоставительная сеть

Также можно попробовать усовершенствовать стандартный GAN, используя идею свёрточной сети для генератора и дискриминатора (Deep Convolutional GAN, DCGAN (<https://arxiv.org/abs/1511.06434>)).

Архитектуру сети опустим, т.к. она является довольно громоздкой, однако её можно увидеть в репозитории с исходным кодом.

Заменим стандартную метрику loss на Least Squares GAN Loss (<https://arxiv.org/abs/1611.04076>). Для генератора:

$$\ell_G = \frac{1}{2} \mathbb{E}_{z \sim p(z)} \left[ (D(G(z)) - 1)^2 \right]$$

и дискриминатора:

$$\ell_D = \frac{1}{2} \mathbb{E}_{x \sim p_{\text{data}}} \left[ (D(x) - 1)^2 \right] + \frac{1}{2} \mathbb{E}_{z \sim p(z)} \left[ (D(G(z)))^2 \right]$$

Получим следующие результаты:



Рисунок 2.3 — Результаты обучения DCGAN с LS GAN loss.

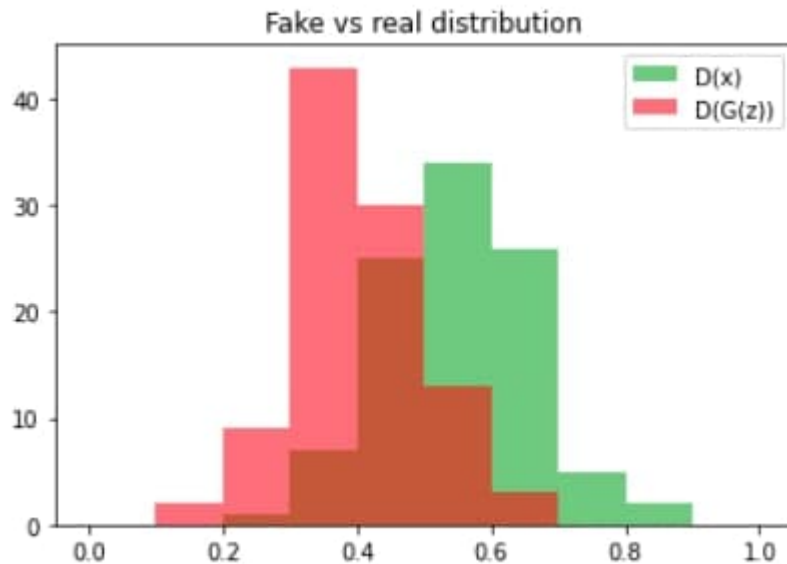


Рисунок 2.4 — "Распределение угадывания дискриминатором подлинности изображений" для DCGAN.

По результатам на различных эпохах видим заметное улучшение качества.

Для разрешения 28x28 получили следующие результаты:

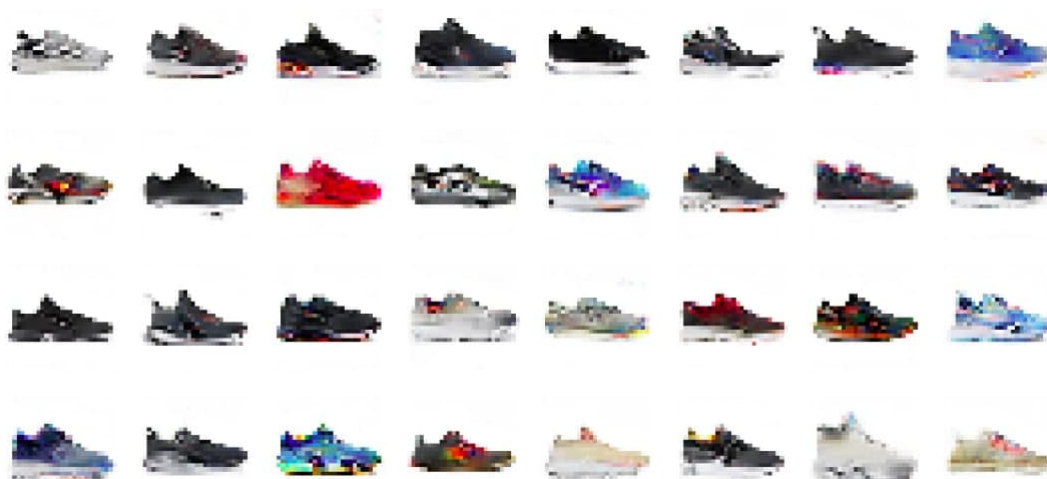


Рисунок 2.5 — Результаты для разрешения 28x28.

Неплохо!

## 2.3 Улучшение качества изображений

### 2.3.1 Немного о задаче транслирования изображений

Задача трансляции изображения(англ. Image-to-image translation) — это задача из области компьютерного зрения, цель которой состоит в том, чтобы научиться строить соответствия между входным и выходным изображениями, используя тренировочные данные.

Другими словами, задача состоит в том, чтобы научиться преобразовывать изображение из одной области в другую, получая в итоге изображение со стилем(характеристиками) последней.





Рисунок 2.6 — Пример трансляции изображения: превращение лошади в зебру.

### 2.3.2 Обучение модели

Обучим SRGAN-like генератор(<https://arxiv.org/abs/1609.04802>).

Сеть будет состоять из нескольких одинаковых базовых блоков. Архитектуру опустим(можно увидеть в исходном коде).

Получили следующие результаты:



Рисунок 2.7 — Результаты решения задачи image-to-image.

Посмотрим на результаты генерации из изображений данных низкого разрешения и оригинал:

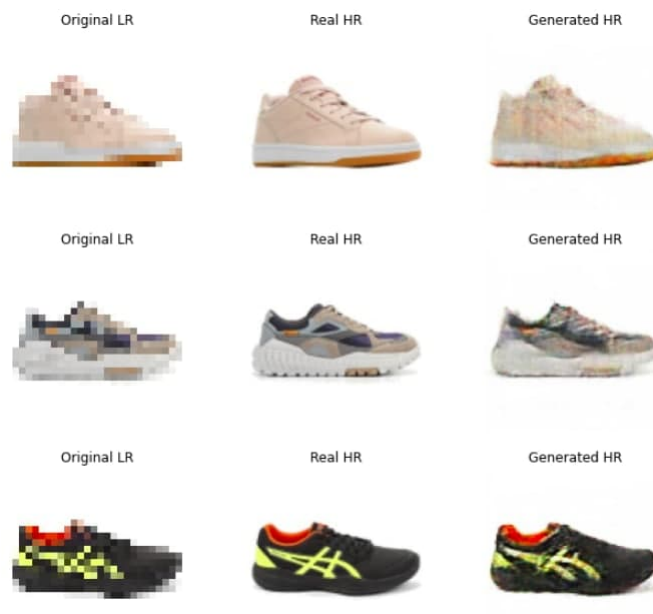


Рисунок 2.8 — Результаты решения задачи image-to-image их подлинных изображений низкого качества в сравнении с оригиналом.

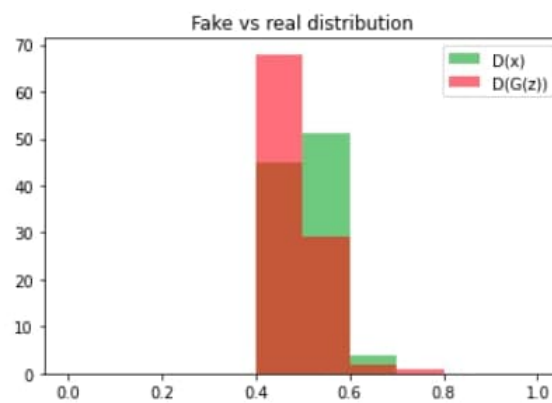


Рисунок 2.9 — "Распределение угадывания дискриминатором подлинности изображений" для SRGAN.

Получили следующие результаты для генерации из подлинных изображений с низким качеством:

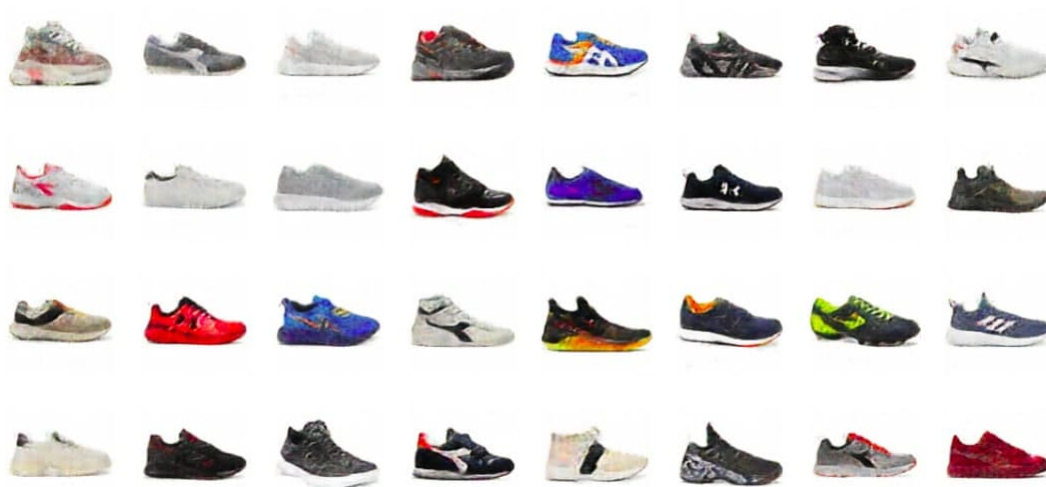


Рисунок 2.10 — Результаты решения задачи image-to-image из подлинных низкокачественных изображений с помощью SRGAN.

И для генерации из поддельных изображений с низким качеством:



Рисунок 2.11 — Результаты решения задачи image-to-image из поддельных низкокачественных изображений с помощью SRGAN.

Существует несколько метрик, которые позволяют оценить качество обучения GAN'ов. Некоторые из них основаны на сравнении подлинных и созданных на их основе поддельных изображений. Посмотрим на 2 метрики, а именно: Precision-Recall Density(<https://arxiv.org/pdf/1806.00035>) и

Fréchet Inception Distance(<https://arxiv.org/pdf/1706.08500>). На результаты исследования данных метрик можно посмотреть в исходном коде.

## Заключение

В рамках данного исследования были обучены несколько различных GAN-like моделей для решения задачи генерации изображений кроссовок, которую мы декомпозировали на две: генерации изображений низкого качества и трансляции изображений из низкого качества в более высокое качество. И получили неплохие результаты!

В целом GAN'ы являются довольно мощной техникой для решения различных "творческих" задач с помощью машинного обучения: генерации изображений из шума и текста, преобразования части изображений, работы со звуком. Эта молодая перспективнейшая область машинного обучения содержит множество неоткрытых возможностей, что, конечно же, только подталкивает к её изучению.

Как можно поменять машинное обучение на девушку?

---

Артём Гаркавый

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Radford A., Metz L., Chintala S. Unsupervised representation learning with deep convolutional generative adversarial networks //arXiv preprint arXiv:1511.06434. – 2015.
2. Mao X. et al. Least squares generative adversarial networks //Proceedings of the IEEE international conference on computer vision. – 2017. – С. 2794-2802.
3. Ledig C. et al. Photo-realistic single image super-resolution using a generative adversarial network //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 4681-4690.
4. Репозиторий с исходным кодом: <https://github.com/dasfex/gan>.