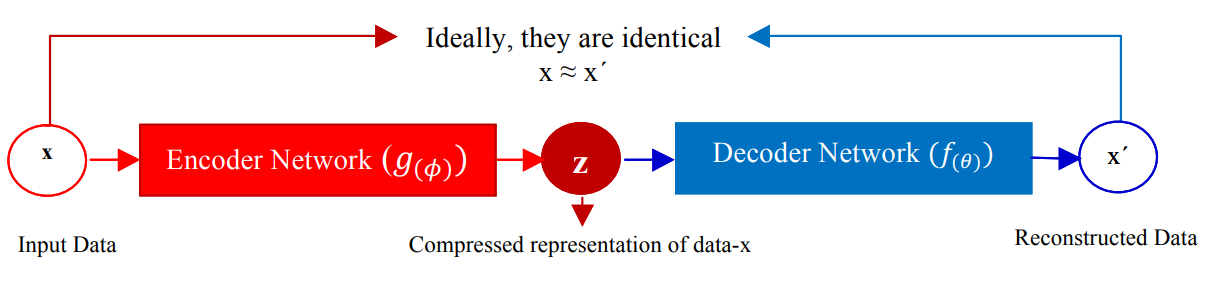
Представьте большое озеро с разными обитателями, утки, лебеди, лягушки и прочие создания. Забудь о законах биологии, мы здесь не для этого. Итак, озеро, солнце и наши главные герои - утки.



Познакомимся с утками поближе. Наш главный герой - **Дикки (генератор)**, утенок подросток, бунтарь. Ему хочется гораздо больше чем позволяют родители, а также предводитель утиной стаи **Утка Босс (дискриминатор)**. *Дикки* разработал план, он хочет *сбежать* из стаи, но при этом *остаться незамеченным*. Но как это сделать? Ведь если Утка Босс заметит, как он уплывает к другим обитателям озера, то тут же вернет его назад. Дикки решил хорошенько **замаскироваться**, чтобы никто и подумать не мог из всех его сородичей, что он принадлежит стае уток. Рядом со стаей уток, всегда крутится стая красивых и роскошных лебедей, он решил замаскироваться под лебедя. Ведь они часто проплывают мимо них. (*Генератор пытается сгенерировать данные, которые принадлежат некоторому распределению p(x), на которых он будет учиться.)*

**Автоэнкодер**

Первый план Дикки звучал безумно: создать своего клона и отвлечь Утка Босса. Но как ему это сделать? Из подручных материалов он пытается соорудить утку, проходившую на него. Но сделать это не так просто. Он пытался выцепить из своего образа основным признаки, которые бы описывали его (**латентный вектор**), это был длинный клюв, мелкие перья и прочее (**энкодер сжимает исходные данные**). А после из этих признаков, пытался воссоздать себя (**декодер пытается восстановить данные по латентному вектору**)



Каждый раз Дикки сравнивал себя с собранным клоном. Он пытался минимизировать отличия.

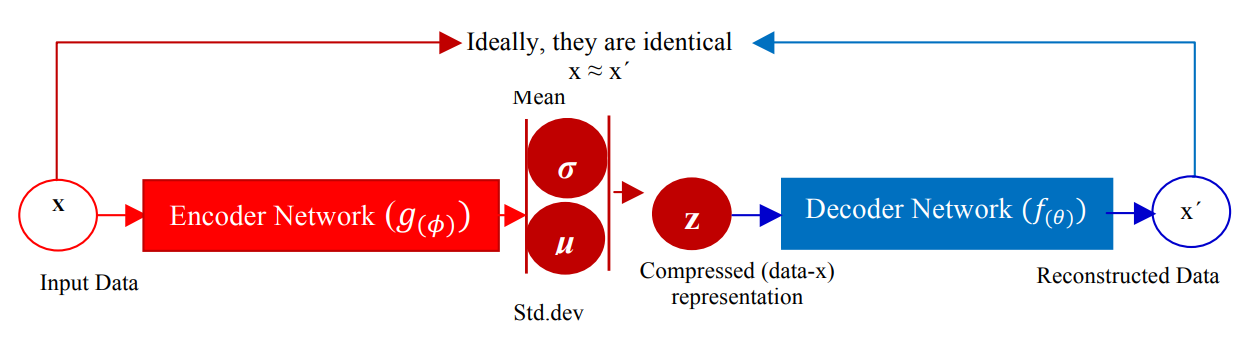


Дикки пытается выбрать маскировочный костюм, красуясь перед прозрачной водой, словно перед зеркалом.

**Вариационный автоэнкодер**

Первый подход имел некоторые недостатки. **Неустойчивость к изменениям**: При использовании первого плана (**обычного автоэнкодера**) каждый раз, когда Дикки изменял свои особенности, например, вырастало перо или менялся цвет первая, кривился клюв, ему приходилось обновлять клон, чтобы он соответствовал его новому облику. Это требовало постоянной модификации клонов и делало план непрактичным на практике. А Дикки хотел часто убегать из стаи. План не подходил. Требовался более гибкий подход.

Дикки нужна была такая маскировка, которая могла бы адаптироваться к его изменениям. Он сново оглядел себя, выцепляя ключевые особоенности. Он пытался продумать вариант, что в нем могло измениться с течением времени, чтобы клон мог нивелировать это и замаскировать, например, плащом. (*Дикки пытается смоделировать латентное пространство, в котором каждая точка представляет собой распределение возможных вариантов. Вариационный автоэнкодер пытается имитировать распределение оригинальных данных, также как и Дикки пытается имитировать распределение возможных копий с изменениями*)



**Ганы**

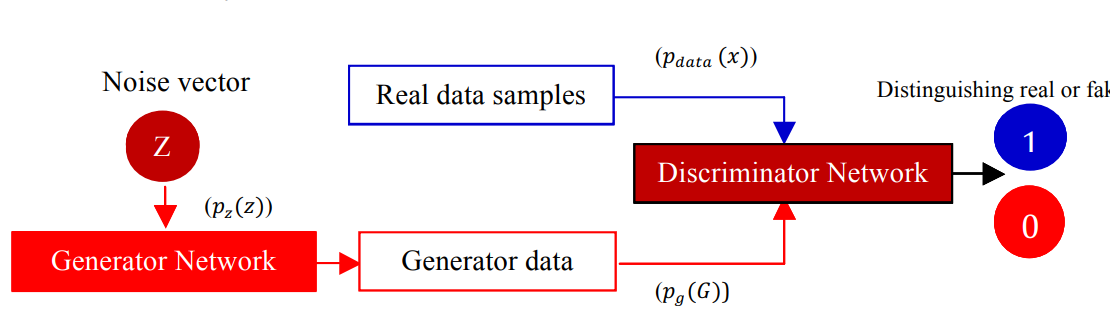
Отлично! Дикки собрал своего клона. Осталось его протестировать. Дикки пробует отправить своего клона в сторону стаи Лебедей. Его задача, чтобы Утка Босс подумал, что клон это Дикки и перехватил его. Тогда задача будет выполнена.

Но вот беда! Утка Босс даже не обратил на него внимания! Это полный провал. Дикки снова вернулся к преобразованию своей копии. Он внимательно изучил ее и понял, что усы явно лишнии. У него ведь их нет! Отлично. Можно пробовать… И опять провал! Да, что такое!

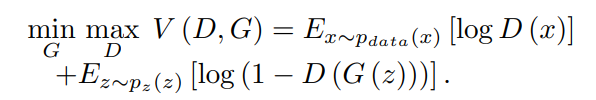
Дикки провел несколько часов в попытка обмануть Утка Босса, он совершенствовал своего клона снова и снова, и вот… Победа! Утка Босс заметил его! Правда он совсем не понял, что здесь делает эта поролоновая кукла, похожая на Дикки. Наверно, маленький утенок опять пытается издеваться на главарем стаи.

Дикки был слишком рад, но все же решил провести повторный тест. Но на этот раз он провалился! Похоже, Утка Босс тоже научился распознавать его копию. Дикки снова предстояло улучшать свою копию.

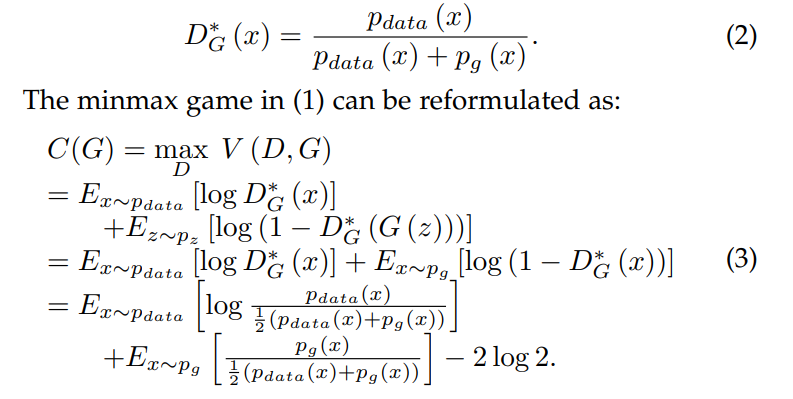
И так между Дикки (**генератор**) и Утка Боссом (**дискриминатор**) началась игра. Дикки пытается обмануть главаря стаи, Утка Босс пытается лучше распознавать фейкового Дикки.



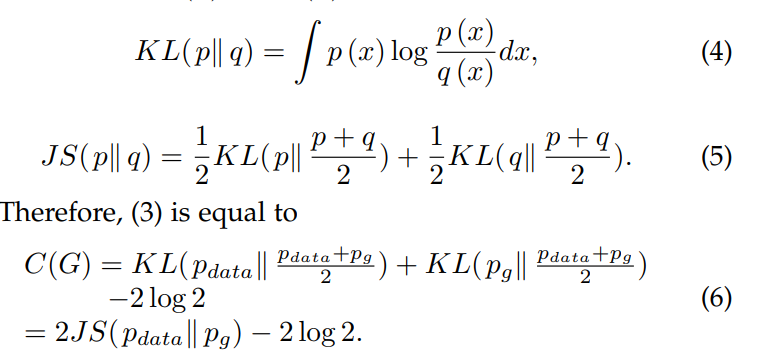
**Loss**



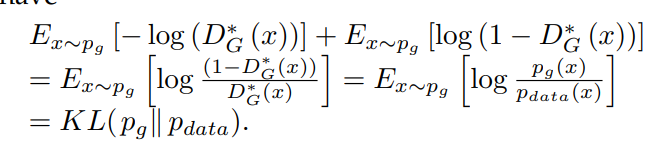
Так как дискриминатор тоже обучается, то мы заинтересованы в том, чтобы обучить крутой дискриминатор, чтобы генератор держал руку на пульсе. Распишим лосс:

****

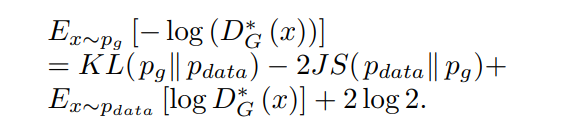
Вспомним как находится расстояние между двумя распределениями, и попробуем выразить через него лосс

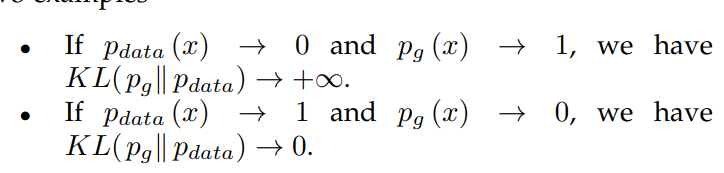


или

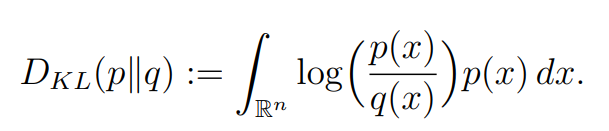


В лоссе можно заметить противоречие. Он состоит из двух слагаемых. Первый старается сделать распределение максимально похожим, второй максимально разным. Из-за этого обучение гана нестабильно.



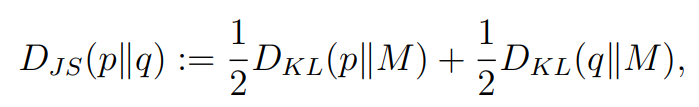


**Расстояние Кюхельбекера**

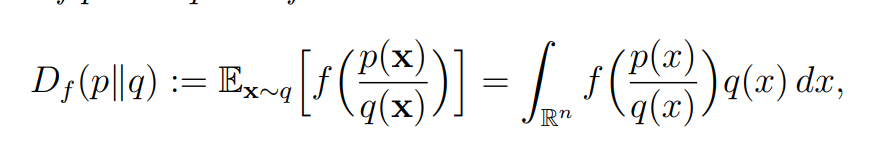


**Расстояние Дженсен Шенона**

Свойство - оно симметрично (D(p||q) == D(q||p))

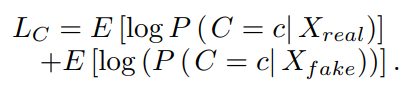


**Определение f-дивергенции**



**Conditional GAN**

Похож на infoGan, исключение что мы не знаем распределение (условное) c.



**LapGan**

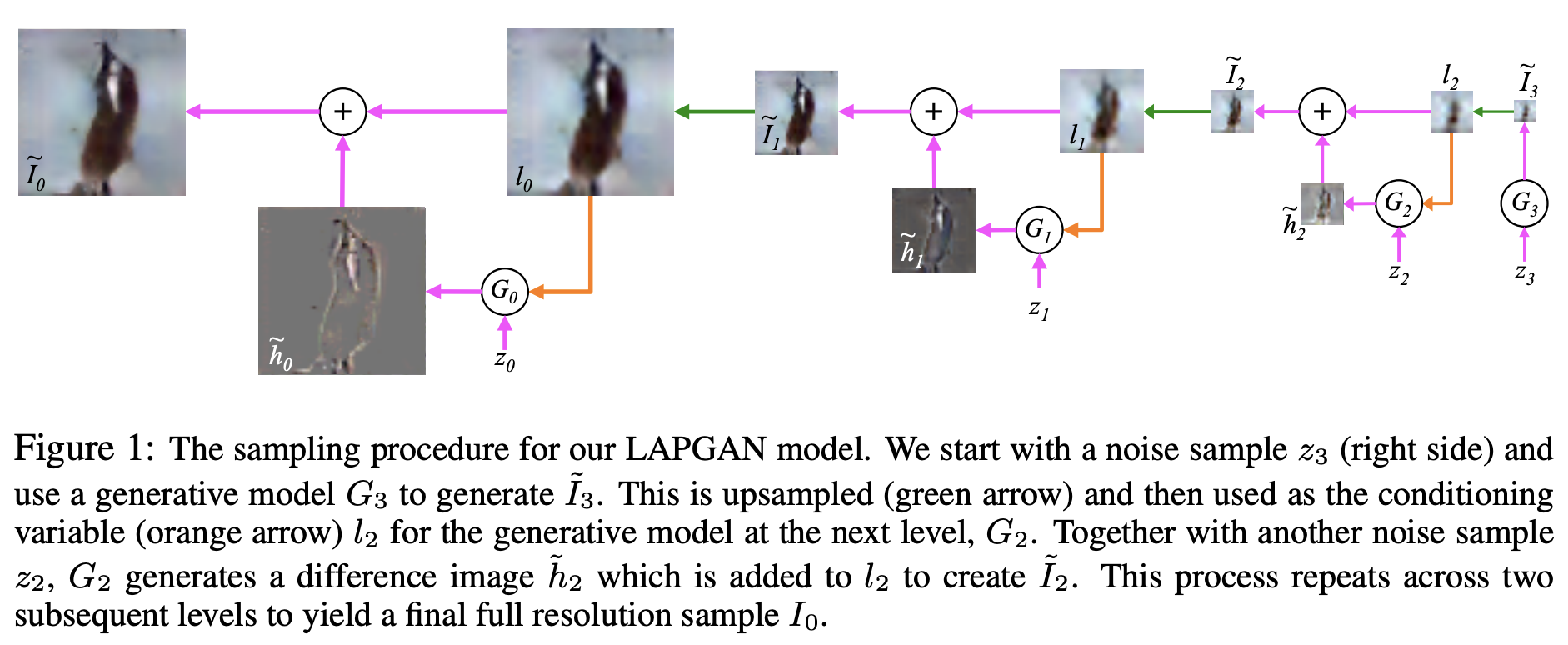
Основная идея LAPGAN состоит в том, чтобы генерировать изображения на разных уровнях разрешения с использованием пирамидальной структуры, постепенно увеличивая разрешение изображения на каждом уровне. На каждом уровне разрешения используется отдельная генеративная сеть GAN для генерации деталей изображения на этом уровне. Таким образом, архитектура LAPGAN представляет собой пирамиду из GAN-сетей, в которой каждый уровень представляет собой дополнительное разрешение изображения.

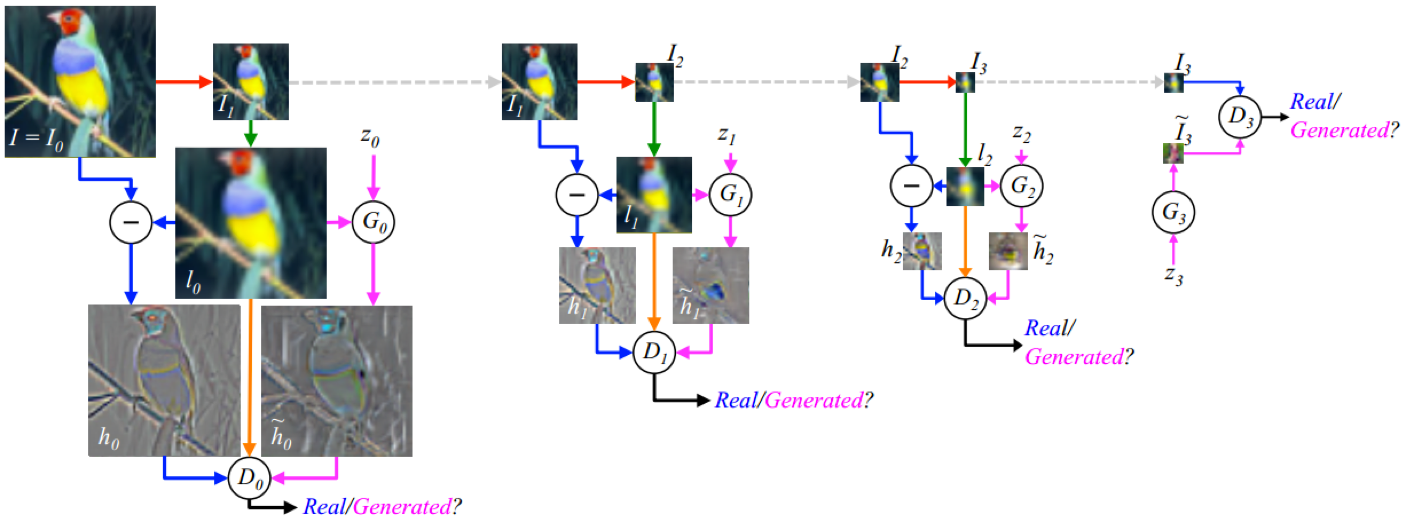
В LAPGAN каждый уровень изображения генерируется с использованием разноуровневой (multiscale) стратегии. Она включает в себя применение метода лапласиана к пирамиде Гаусса изображения. Пирамида Гаусса представляет собой размытие изображения на разных уровнях для получения множества низкоразрешенных изображений.

Процесс генерации изображения высокого разрешения в LAPGAN выглядит следующим образом:

1. Изначально, случайный шум преобразуется в низкоразрешенное изображение.
2. Затем этот низкоразрешенный вариант проходит через серию GAN, каждый из которых работает на разных уровнях разрешения пирамиды Гаусса.
3. На каждом уровне разрешения генерируется некоторая дополнительная детализация.
4. Каждый уровень детализации затем добавляется к предыдущему уровню разрешения.
5. Этот процесс повторяется, постепенно увеличивая разрешение и качество изображения.

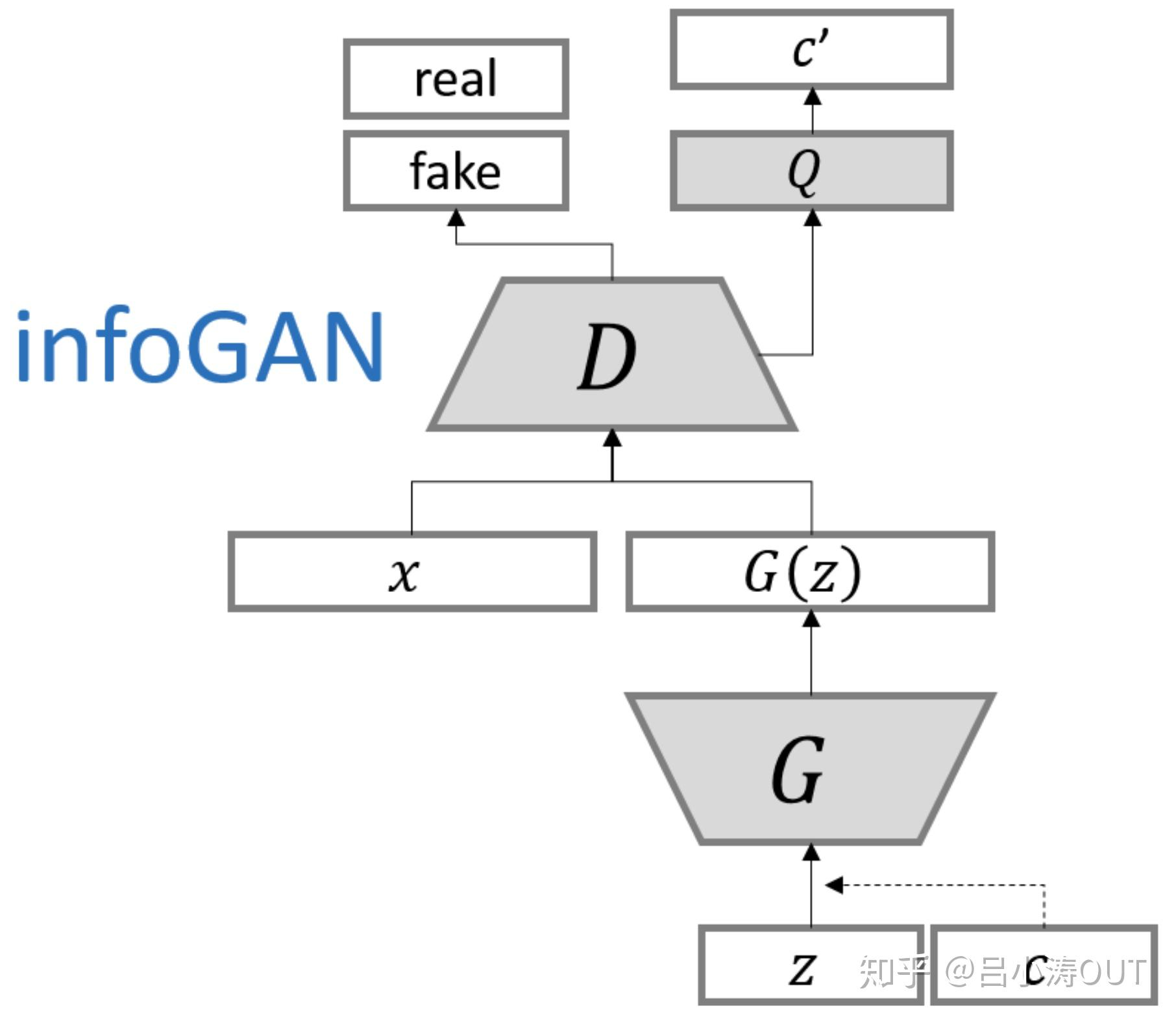
Таким образом, LAPGAN использует мультискейлинг и метод лапласиана для создания изображений разного разрешения. Это позволяет сохранить детали на разных уровнях разрешения и создать более высококачественные изображения.

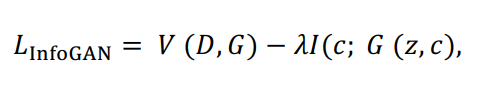


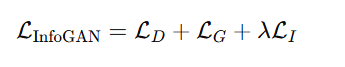


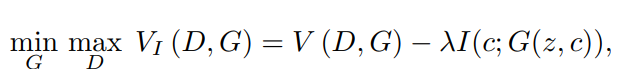
**Information Maximizing GAN (InfoGAN)**

Основная **идея** InfoGAN заключается в том, чтобы обучать генеративную модель, которая не только **генерирует** реалистичные изображения, но и **изучает структуру** скрытого пространства, чтобы **разделить** его на **интерпретируемые** и **управляемые** компоненты. Это позволяет модели генерировать изображения, которые *могут быть изменены* в соответствии с определенными атрибутами или характеристиками, такими как цвет, форма, положение объектов и так далее.

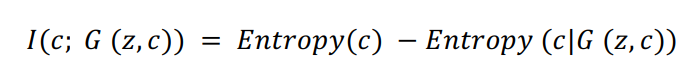


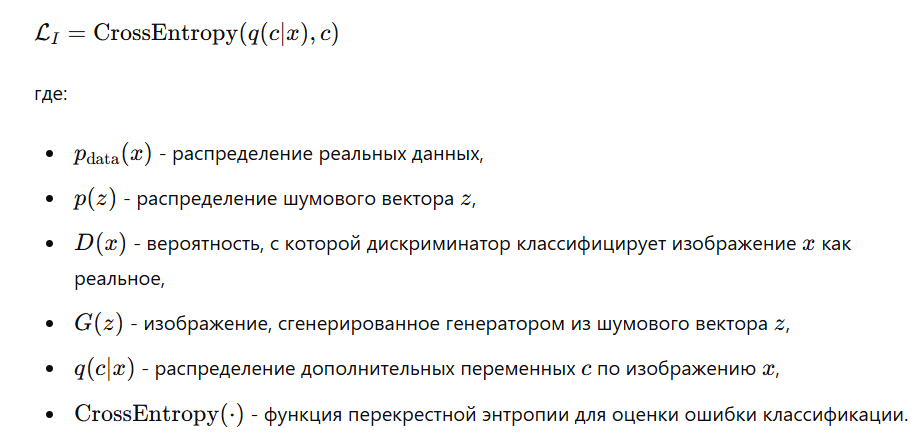






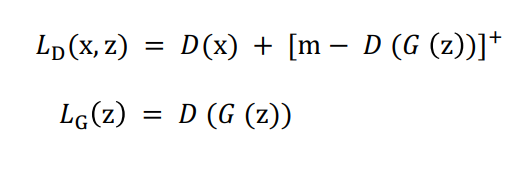
или по другому:

или



**Energy-Based GAN (EBGAN)**

Основная идея EBGAN заключается в том, чтобы превратить задачу генерации изображений в задачу оптимизации энергии, где цель состоит в минимизации энергии в реальных данных и максимизации энергии в сгенерированных данных. Энергия 0 когда это реальные данные, энергия максимальная, когда сгенерированы.

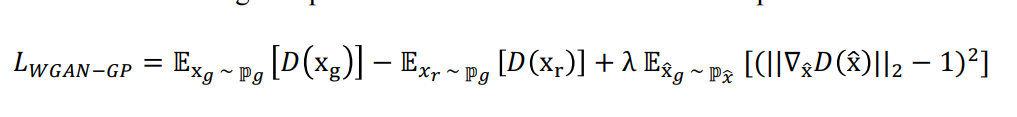


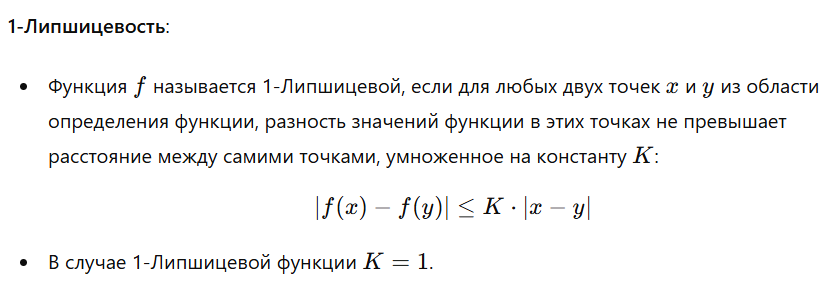
**WGAN**

Основная идея WGAN заключается в использовании расстояния Wasserstein (также известного как расстояние Землероя) в качестве меры различия между реальным и сгенерированным распределениями данных вместо бинарное классификатора, чтобы избежать взрыва градиента. В WGAN дискриминатор (или критик) оценивает расстояние Wasserstein между реальными и сгенерированными распределениями. Это расстояние имеет лучшее поведение с точки зрения градиентов, что делает обучение более стабильным.

В WGAN дискриминатор называется критиком, поскольку он больше не классифицирует данные на реальные и сгенерированные, а оценивает их по шкале. Цель критика — минимизировать это расстояние Wasserstein.

Для корректного вычисления расстояния Wasserstein необходимо, чтобы функция критика была 1-Липшицевой, что означает, что градиенты функции критика не должны превышать некоторого значения. Это достигается путем "усечения градиентов" (gradient clipping), когда значения весов критика ограничиваются заданными пределами.



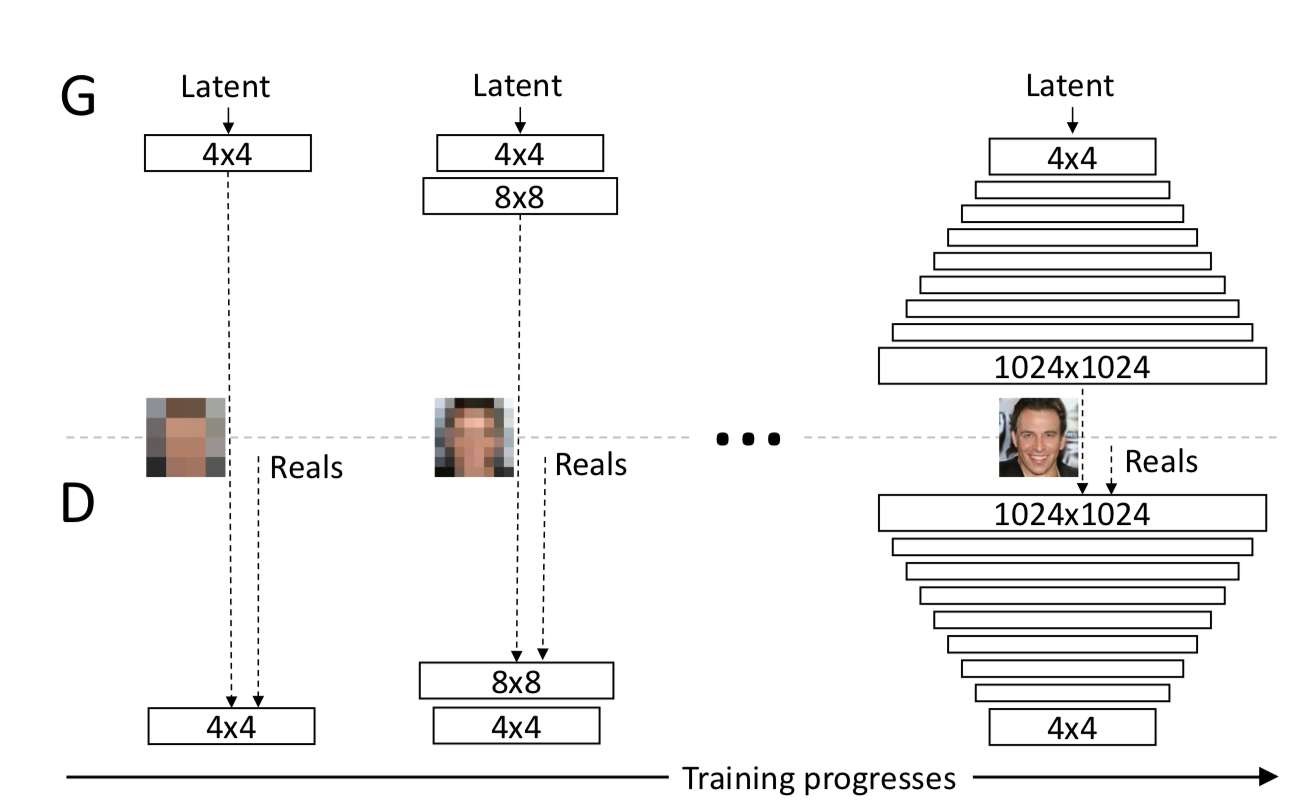


Чтобы удовлетворять Липшицевости, В WGAN делается это путем ограничения весов критика в заданных пределах после каждого шага обновления.

**Progressive-Growing GAN (PGGAN)**

Модель начинается с генерации изображений малого разрешения (например, 4×4 пикселя) и постепенно увеличивает разрешение до целевого (например, 1024×1024 пикселей) за счет добавления новых слоев в генератор и дискриминатор.

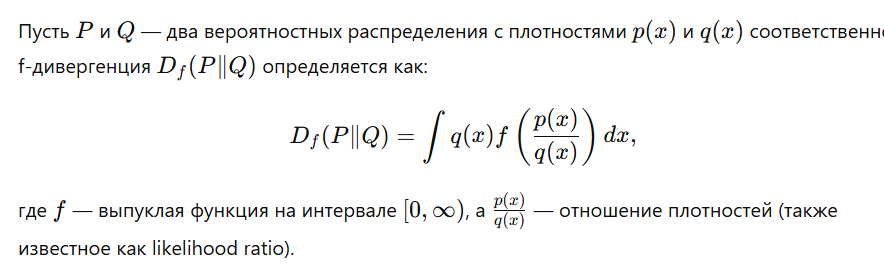
**Постепенное добавление слоев**: На каждом этапе новые слои добавляются постепенно, начиная с малых весов, которые увеличиваются по мере тренировки. Это позволяет сети стабильно адаптироваться к новым слоям.



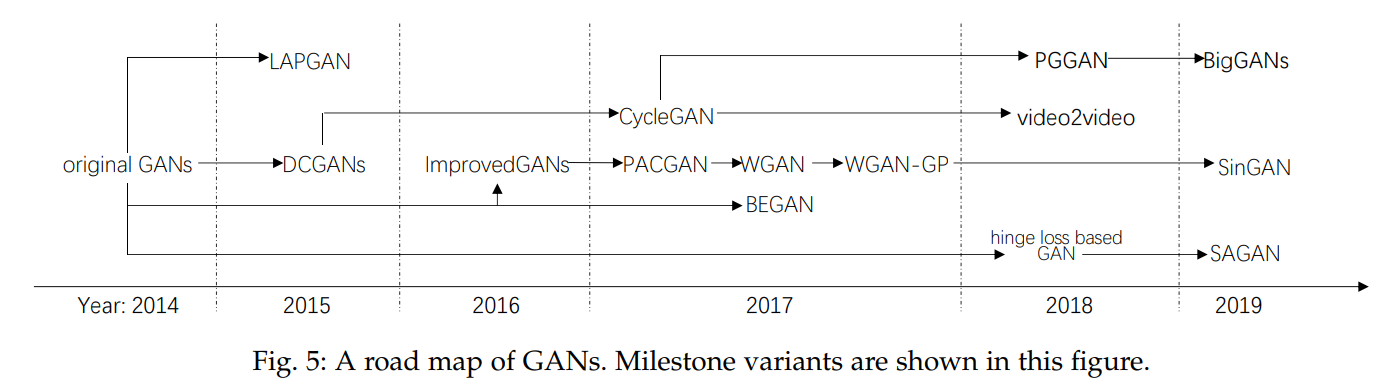
**Fade-in**: На каждом этапе добавления новых слоев используется техника плавного перехода (fade-in), когда старые слои постепенно смешиваются с новыми. Это позволяет избежать резких изменений в сети и делает процесс обучения более плавным.

**f-GAN**

f-дивергенция (f-divergence) — это обобщенная мера различия между двумя вероятностными распределениями. Представляет собой широкую категорию дивергенций, включающую такие известные меры, как Kullback-Leibler (KL) дивергенция, Jensen-Shannon (JS) дивергенция, Total Variation (TV) расстояние и многие другие.



**f-GAN (f-divergence Generative Adversarial Networks)** — это обобщение стандартных GAN, предложенное для использования более широкого класса расхождений, называемых f-дивергенциями, для измерения разницы между распределениями данных и распределениями, генерируемыми моделью.



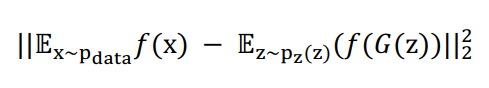
**Обучение**

Почему ганы сложно обучать:

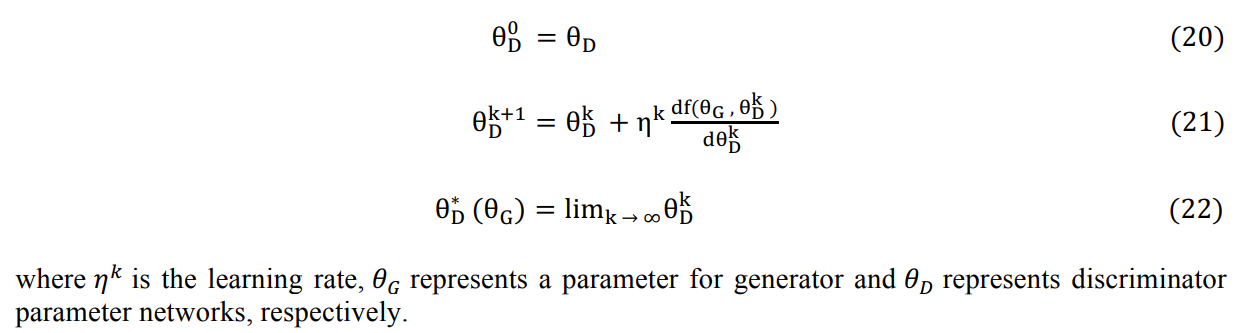
* **Нестабильность обучения**. Если одна сеть обучается быстрее или лучше, чем другая, это может привести к неустойчивому обучению. Например, если дискриминатор становится слишком сильным, генератор не сможет его обмануть и не получит полезных градиентов для обучения. И наоборот, если генератор становится слишком сильным слишком быстро, дискриминатор может не успевать за ним, что также приведет к проблемам в обучении.
* GAN использует специфическую функцию потерь, которая зависит от логарифмической разницы между вероятностями, предсказываемыми дискриминатором. В случае несбалансированного обучения или плохой инициализации параметров, эта функция может приводить к плохой сходимости и нестабильности.
* Генератор может зациклиться на чем-то одном, у него будет маленькое разообразие. (**Mode Collapse**) Это происходит, когда генератор находит один или несколько образцов, которые успешно обманывают дискриминатор, и продолжает генерировать только эти образцы.
* Мы учим модельку, и может возникнуть ситуация, когда рапредедление входных параметров, полученные с помощью прошлых слоев, будет отличатся. Это может быть в результате обновления весов. (**Internal Covariate Shift (ICS)** ). Это замедляет тренировку.

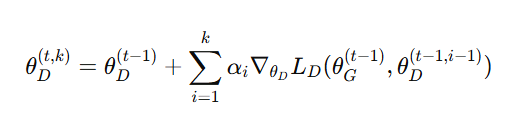
**Улучшения тренировки**

* **Feature matching.** Выдаем статистики и дискриминатор пытается понять это статистики настоящих данных или нет. Короче, меня функцию потерь.



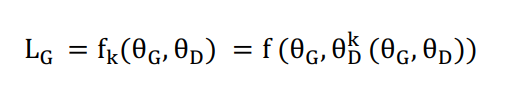
* **Unrolled GAN (UGAN)** Основная идея состоит в том, чтобы дать генератору более точное представление о том, как изменения в его параметрах повлияют на дискриминатор, путем "разворачивания" обновлений дискриминатора на несколько шагов вперед. Помогает от mode collapseю

****



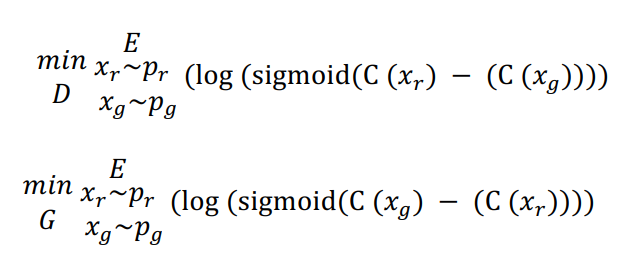
Генератор обноваляется с учетом того, как изменится дискриминатор за эти k шагов.

Новый лосс для генератора



В Unrolled GAN дискриминатор обновляется на несколько шагов (скажем, kkk шагов) перед обновлением генератора. Это позволяет генератору видеть, как дискриминатор изменится в ответ на его обновления, и учитывать это при обновлении своих параметров. **Важно, что эти k шагов обновлений дискриминатора выполняются гипотетически**, без фактического обновления параметров дискриминатора, а затем используются для обновления параметров генератора.

* **мини-батч**
* усреднение параметров (**Historical averaging (HA)**). Идея этого метода заключается в том, Historical Averaging добавляет дополнительный член к функции потерь, который штрафует отклонения текущих параметров модели от их исторического среднего значения.
* использовать разные **learning rates** для генератора и дискриминатора
* self-attention
* **label smoothing**
* Relativistic GAN (RGAN). **Переход к скор функции**, дискриминатор не отвечает на вопрос это реальные данные или фейк, он пытается ответить, на сколько картинка реальнее второй картинки

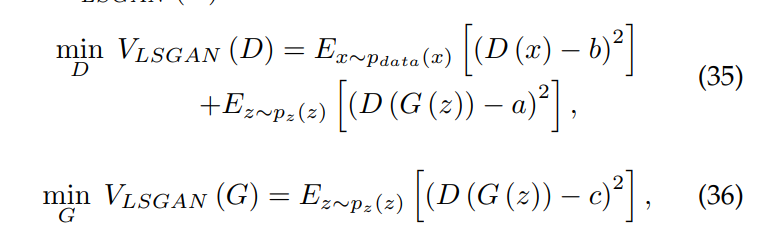


* разные виды **нормализации**
* добавить шум к реальным и сгенерированным данным
* **WGAN Gradient Penalty**. Ввести штраф за обрезание весов (clip)

**Лоссы**

**Least squares generative adversarial**

Добавляем возведение в квадрат, чтобы избежать исчезающих градиентов. разновидность стандартных GAN, предложенная для улучшения стабильности и качества генерации, используя функцию потерь на основе метода наименьших квадратов вместо бинарной кросс-энтропии. Основная идея LSGAN заключается в изменении целевой функции дискриминатора и генератора для смягчения проблем, таких как исчезающие градиенты, что часто встречается в традиционных GAN.



**Метка aaa**: целевое значение для сгенерированных (fake) образцов.

**Метка bbb**: целевое значение для реальных (real) образцов.

Generative Adversarial Networks (GANs) **не используют максимальное правдоподобие** в функции потерь по нескольким причинам, связанным с уникальными особенностями их структуры и целями. Основная **причина** заключается в том, что GANs стремятся непосредственно **минимизировать расхождение** между распределениями данных и сгенерированных образцов, а не максимизировать правдоподобие отдельных примеров.

Максимальное правдоподобие может страдать от проблемы исчезающих градиентов, особенно в высокоразмерных пространствах.

**Метрики**

* Inception score. Оценивает разнообразие и качество сгенерированных изображений, используя предсказания нейронной сети Inception.
* fid. Измеряет расстояние между распределениями реальных и сгенерированных изображений в пространстве активаций Inception сети.

Тут обзор ганов и методы улучшения обучения для стабилизации

[Microsoft Word - Survey Paper GANS-Revised\_Form(07.03.2020).docx (arxiv.org)](https://arxiv.org/pdf/2006.05132v1)

[My title (arxiv.org)](https://arxiv.org/pdf/2001.06937)

Трюки для улучшения обучения

