онлайн-курс

СПЕЦИАЛЬНЫЕ АРХИТЕКТУРЫ НЕЙРОННЫХ СЕТЕЙ

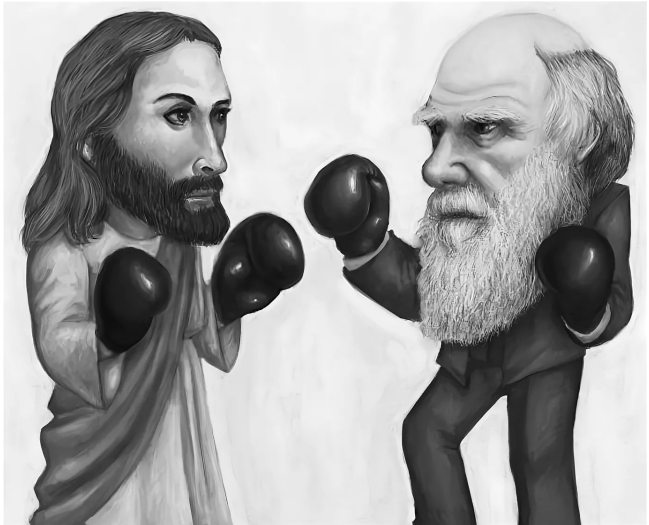
Генеративно-состязательные сети (GAN)

Введение

Немного философии...

В жизни есть два способа создания сложных систем и организмов (по крайней мере, люди в это верят):

- сразу с помощью сверхразума;
- самопроизвольно с помощью медленной эволюции.



В теории НС более применим второй способ

(видимо, среди дата сайентистов полно безбожных атеистов).

Эта технология называется **GAN** (generative adversarial network**s**).

Множественное число тут существенно: каждая модель здесь состоит из **двух** НС.
Каждая из которых пытается обмануть другую НС.

На что похож GAN?

GAN по духу очень похож на эволюцию «хищников» и «травоядных». Хищники совершенствуют органы нападения, а травоядные развивают органы защиты. Это всё происходит одновременно при сильном взаимодействии популяций «хищников» и «травоядных» друг с другом.

В результате (после тысячелетий эволюций) получаются организмы с идеально развитыми системами органов, отвечающих за нападение (защиту).

А зачем нужны GAN-ы?

С помощью GAN-а мы моделируем эволюцию в некоторой области народного хозяйства.

Результатом эволюции будет появление новых объектов в рассматриваемой области.

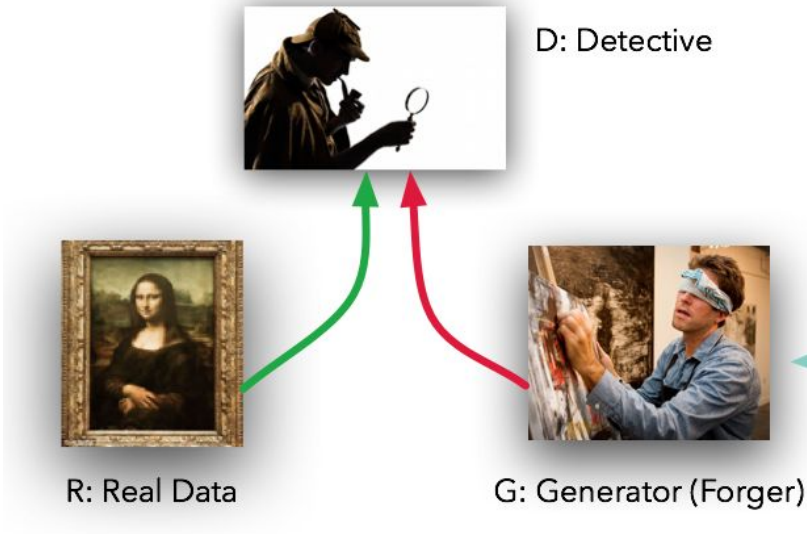
Например, можно генерировать фото никогда не существовавших людей:

А зачем нужны GAN-ы?

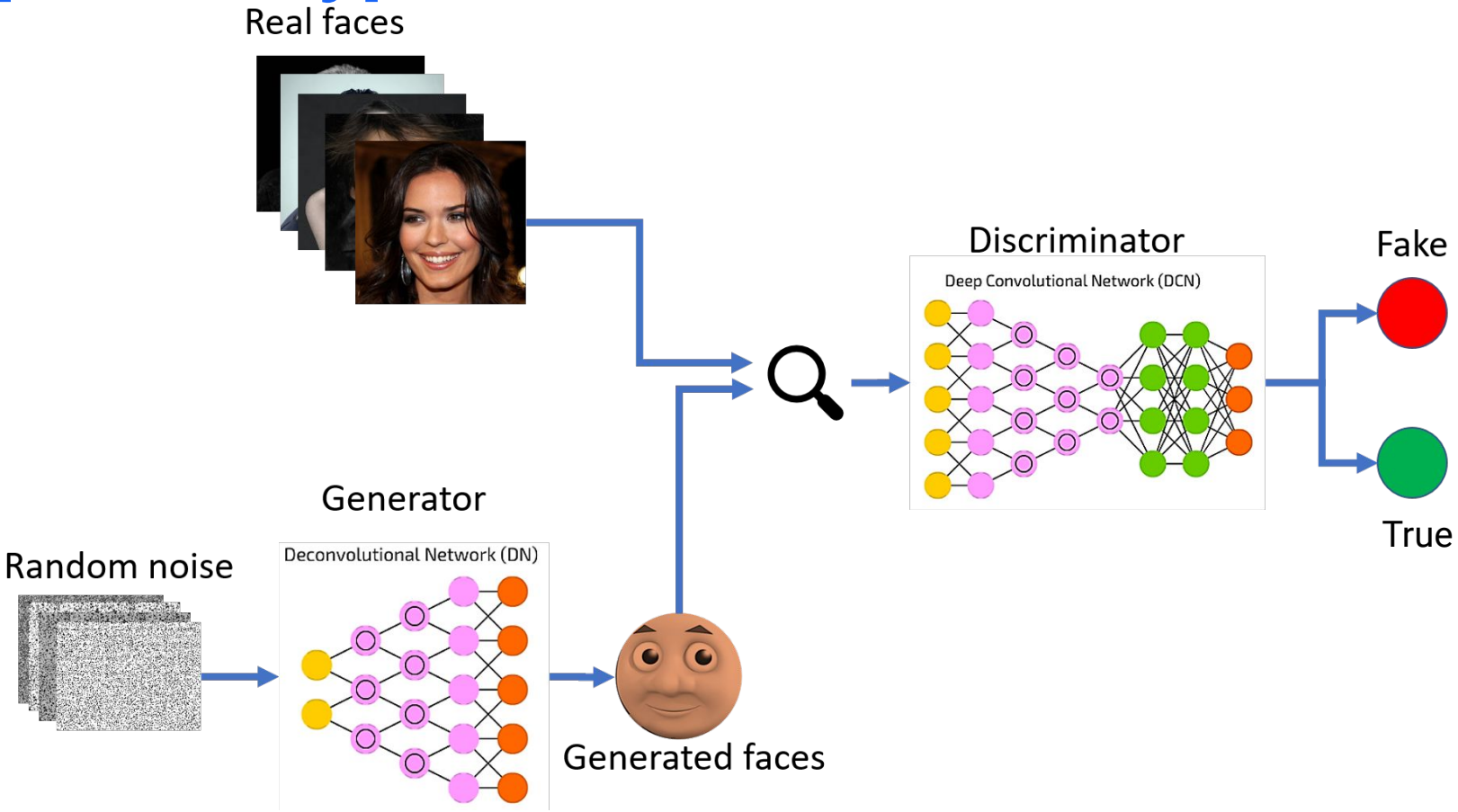
Можно делать **«перенос стиля».** Это позволяет генерировать новые модели одежды, обоев, автомобилей.

Другая иллюстрация: эксперт и жулик

Жулик подделывает деньги (произведения искусства), эксперт пытается его разоблачить, причём ход рассуждений эксперта доступен жулику, и следующую подделку он изготовит с учётом предыдущих выводов эксперта.



Архитектура GAN-а



GAN — это пара НС

Итак, GAN — это **две НС**.

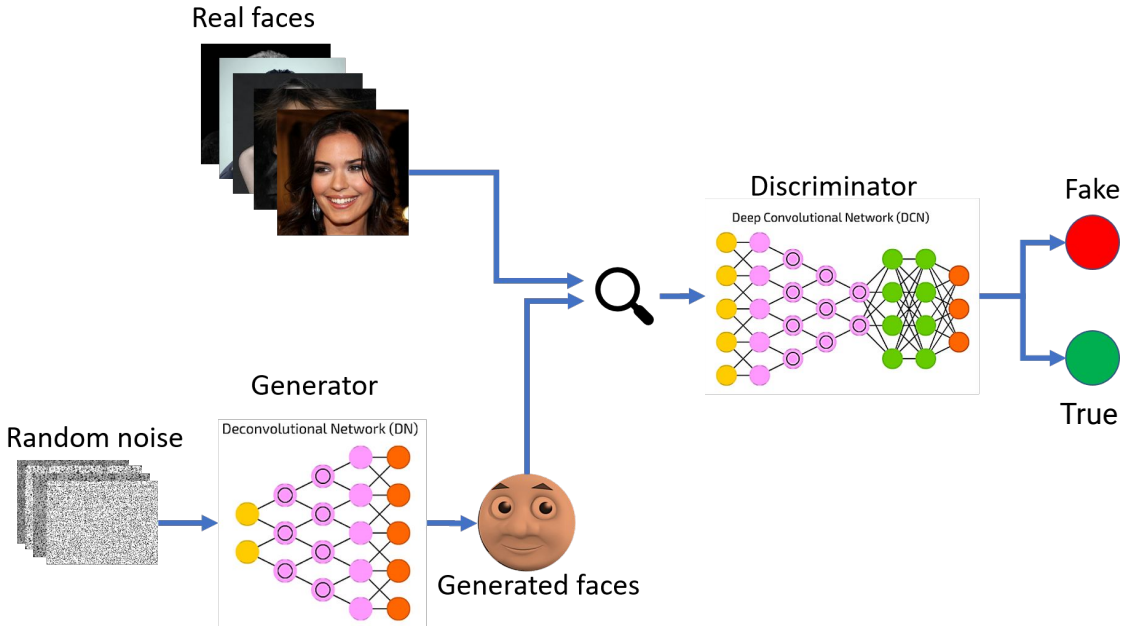
Одна из НС пытается обмануть другую.
Эти НС имеют устоявшиеся названия: **Generator** и **Discriminator**.

Generator порождает некий выход, а Discriminator должен понять, этот выход настоящий или фейковый.

Далее используем обозначение: G и D.

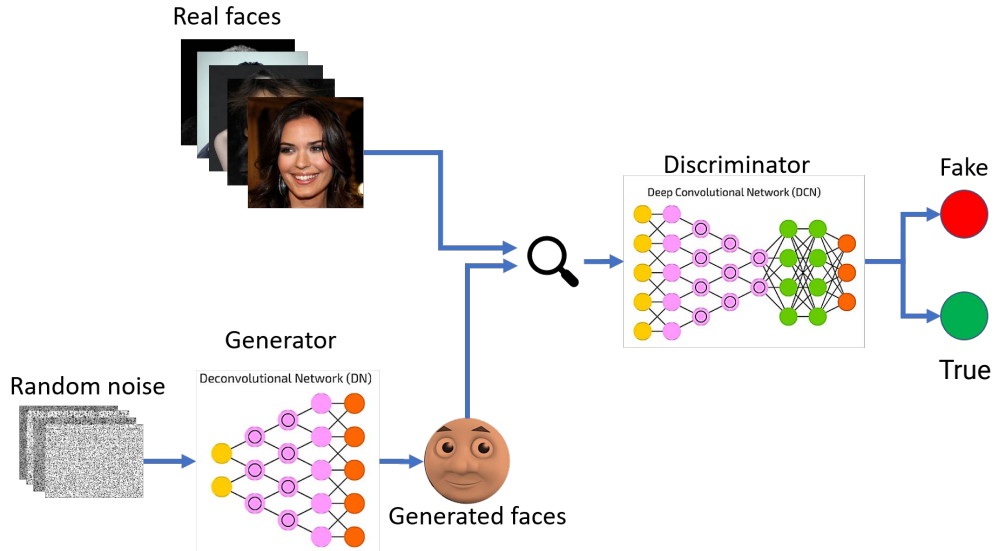
Как это всё работает?

Дискриминатору D попеременно подаются как объекты, сгенерированные генератором G, так и реальные объекты из некоторой библиотеки.



Например, научим G генерировать фотографии людей.
Для этого подключим библиотеку фотографий Real Images из интернета.
Попеременно подаём D как объекты, сгенерированные G, так и случайные объекты из библиотеки. D должен понять, что это за объект. Т.е. D выдаёт либо True, либо Fake. В процессе взаимодействия навыки G и D улучшаются. В конечном итоге, если всё сделать правильно, G научится генерировать картинки, которые сложно отличить от картинок из библиотеки.

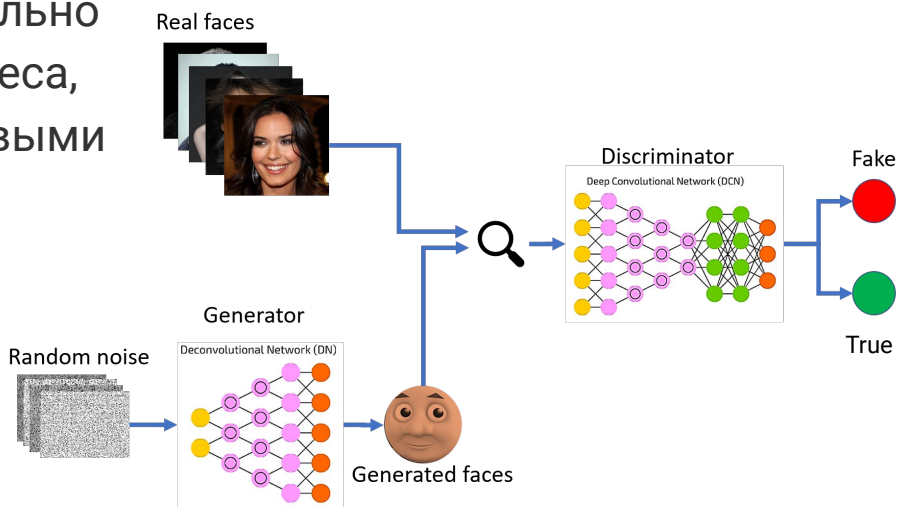
Важно: библиотека истинных объектов Генератору недоступна.



У G и D есть свои веса. Их нужно тренировать.

Тренировка происходит попеременно и разбита на этапы:

- G генерирует набор объектов FO (fake objects);
- из библиотеки случайным образом берётся выборка объектов TO (true objects);
- D получает TB, состоящую из FO и TO — причём ему известен тип каждого объекта;
- D по TB составляет функцию потерь и делает **один шаг ГС**;
- G составляет выражение «вероятность фейковости сгенерированного мною объекта» и минимизирует её относительно своих весов (то есть подгоняет свои веса, чтобы вероятность признания фейковыми его выходов уменьшилась).
- G делает **один шаг ГС.**



Более формально. Этапы тренировки GAN-а:

Итерация обучения:

1) генерация набора случайных чисел $r_1, r_2, ..., r_n$;

2) G строит по r_i (i = 1, n) объекты $a_1, a_2, ..., a_n$ (это выборка FO);

3) берётся случайная выборка $b_1, b_2, ..., b_m$ объектов из библиотеки (это выборка TO);

4) формируется тренировочная выборка для D: FO+TO

5) По этой таблице строится функция потерь L(u) для D

(u — набор весов Дискриминатора D)

6) Делается один шаг ГС для L(u), веса дискриминатора обновляются.

Объект	Метка
a_1	0
...	...
a_n	0
b_1	1
...	...
b_m	1

Более формально. Этапы тренировки GAN-а:

7) У генератора есть функция сети $F_G(r)$, которая по случайному входу r генерирует объект. В нашем случае было так: $\mathbf{F_G(r_1)=a_1}$, $\mathbf{F_G(r_2)=a_2}$,..., $\mathbf{F_G(r_n)=a_n}$.

Выход генератора зависит от его весов w. Поэтому можно составить выражение в общем виде: $F_G(r,w)=a$.

Мы хотим минимизировать вероятность фейковости выражений:
$Pr(F_G(r_1,w) \in Fake)$, $Pr(F_G(r_2,w) \in Fake)$,...,$Pr(F_G(r_n,w) \in Fake)$

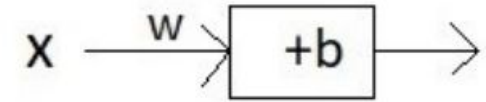
То есть нужно **минимизировать сумму логарифмов**:
$Ln(Pr(F_G(r_1,w) \in Fake))+Ln(Pr(F_G(r_2,w) \in Fake))+Ln(Pr(F_G(r_n,w) \in Fake)) \to min$

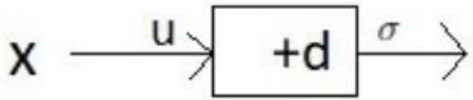
Это выражение зависит от весов w генератора G. Делаем один шаг ГС, изменяя веса w. После этого начинается новая итерация обучения GAN-а.

Очень нужен пример...

Задача.

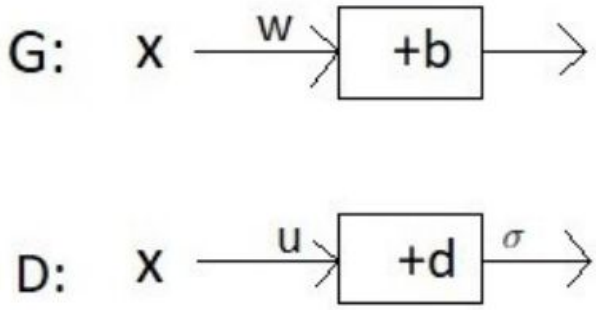
Условимся считать положительные числа «правильными», а «отрицательные» — фейковыми. Посмотрим, как быстро генератор догадается генерировать только «правильные» числа.

Сначала определим G: 

D: 

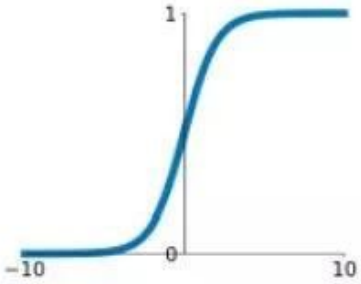
Выходы сетей: $F_G(x)=wx+b$, $F_D(x)=\sigma(ux+d)$

Очень нужен пример...



Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



Выходы сетей: $F_G(x)=wx+b$, $F_D(x)=\sigma(ux+d)$

D возвращает вероятность, что вход x — фейковый (отрицательный).

Зададим начальные значения весов: w=1, b=0, u=0, d=0.

1) генерируем два случайных числа $r_1 = -1$, $r_2 = -2$, получаем два объекта −1 и −2 (так как $F_G(r)=wr+b$ и сейчас w=1, b=0);
2) генерируем два настоящих объекта: пусть это будут 1 и 2;
3) формируем тренировочную выборку для D;

Объект	Метка
-1	0
-2	0
1	1
2	1

4) выписываем функцию потерь для D, считаем её градиент и делаем один шаг ГС:

$$L(u;d) := ln(\sigma(u+d))+ln(\sigma(2u+d))+ln(1-\sigma(-u+d))+ln(1-\sigma(-2u+d))$$

$$\sigma(x) = \frac{1}{1+e^{-x}},$$
$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

$$\frac{\partial L}{\partial u} = \frac{\sigma(u+d)(1-\sigma(u+d))}{\sigma(u+d)} + 2\frac{\sigma(2u+d)(1-\sigma(2u+d))}{\sigma(2u+d)} +$$
$$+\frac{\sigma(-u+d)(1-\sigma(-u+d))}{1-\sigma(-u+d)} + 2\frac{\sigma(-2u+d)(1-\sigma(-2u+d))}{1-\sigma(-2u+d)} =$$
$$= 1-\sigma(u+d) + 2(1-\sigma(2u+d)) + \sigma(-u+d) + 2\sigma(-2u+d)$$

Значение частной производной в текущей точке (u=0, d=0) равно:

1-0.5+2(1-0.5)+0.5+2*0.5=3

Переходим к вычислению частной производной по d....

$$\frac{\partial L}{\partial d} = \frac{\sigma(u+d)(1-\sigma(u+d))}{\sigma(u+d)} + \frac{\sigma(2u+d)(1-\sigma(2u+d))}{\sigma(2u+d)} -$$

$$- \frac{\sigma(-u+d)(1-\sigma(-u+d))}{1-\sigma(-u+d)} - \frac{\sigma(-2u+d)(1-\sigma(-2u+d))}{1-\sigma(-2u+d)} =$$

$$= 1 - \sigma(u+d) + 1 - \sigma(2u+d) - \sigma(-u+d) - \sigma(-2u+d)$$

Значение частной производной в текущей точке (u=0, d=0) равно:

1-0.5+1-0.5-0.5-0.5=0.

Таким образом, при шаге ГС h=0.1 получаем новые значения весов дискриминатора:

u=0-0.1*3=-0.3 **d=0-0.1*0=0**

Переходим теперь к генератору...

если у G случайный вход r, то $F_G(r)=(wr + b)$ — выход G.

Вероятность фейковости, которую выдает дискриминатор:

$$\sigma((wr + b)u + d)$$

Формируем и минимизируем функцию потерь для G,
подставляя сгенерированные ранее случайные входы $r_1=-1$ и $r_2=-2$:

$$L(w;b) = ln(\sigma((w(-1) + b)u + d)) + ln(\sigma((w(-2) + b)u + d)) \longrightarrow min$$

где через u,d обозначены текущие значения весов дискриминатора
(мы пока не будем вместо них подставлять их значения u=-0.3, d=0).

Считаем частные производные функции L(w,b)...

$$\frac{\partial L}{\partial w} = -u(1 - \sigma((w(-1) + b)u + d)) - 2u(1 - \sigma((w(-2) + b)u + d))$$

$$\frac{\partial L}{\partial b} = u(1 - \sigma((w(-1) + b)u + d)) + u(1 - \sigma((w(-2) + b)u + d))$$

Их значения в текущей точке (w=1, b=0, u=-0.3, d=0) равны:

$$\frac{\partial L}{\partial w}(\text{в тек. точке}) = (1 - 0.57)0.3 + (1 - 0.65)0.6 = 0.34$$

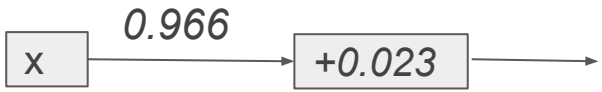
$$\frac{\partial L}{\partial b}(\text{в тек. точке}) = (1 - 0.57)(-0.3) + (1 - 0.65)(-0.3) = -0.23$$

$$w := w - h\frac{\partial L}{\partial w}(\text{в тек. точке}) = 1 - 0.1 * 0.34 = 0.966$$

$$b := b - h\frac{\partial L}{\partial b}(\text{в тек. точке}) = 0 - 0.1 * (-0.23) = 0.023$$

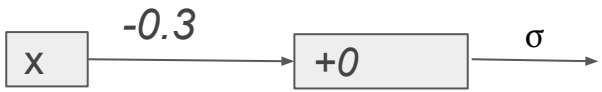
Таким образом, после первой тренировки G и D они приобрели веса:

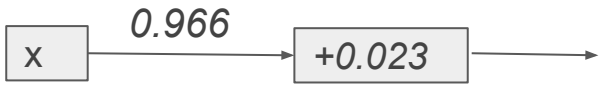
G:



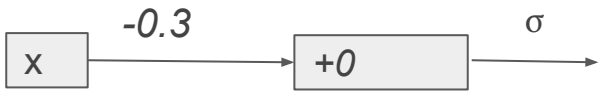
Насколько логично такое изменение весов?

D:



G: 

Насколько логично такое изменение весов?

D: 

Напомним, что на первой итерации тренировки GAN-а использовалась следующая ТВ:

Объект	Метка
-1	0
-2	0
1	1
2	1

Поскольку D выдает вероятность фейковости, то он должен отрицательные числа переводить в положительные.
Отсюда и отрицательный вес -0.3.

Насколько логично такое изменение весов?

Генератор тоже немножко подучился.
Он решил **уменьшать по модулю вес w и увеличивать вес b**.

То есть G стремится к значениям: w=0, b>>0.
Это позволит в дальнейшем получать только положительные выходы для почти любого входа.

Объект	Метка
-1	0
-2	0
1	1
2	1

Выводы

- Мы рассмотрели архитектуру GAN.
- Мы поняли, для чего нужны такие нейронные сети и как они тренируются.