Конспект лекции

А.Н. Шевляков

Введение в нейронные сети

Понятие о состязательных сетях (GAN)

# Введение

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. 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.

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).

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

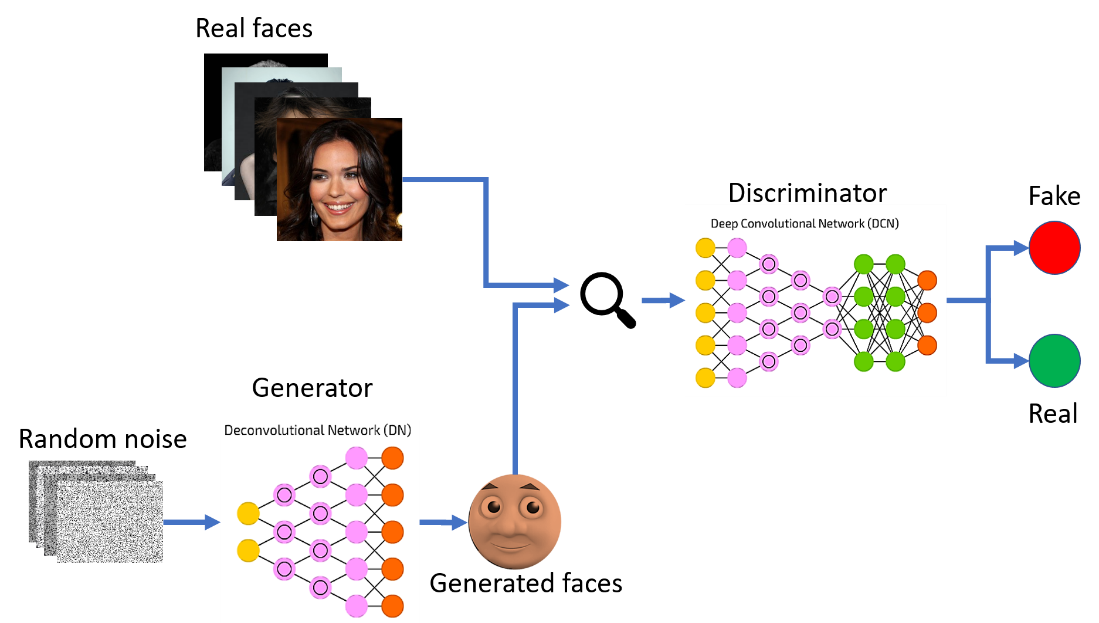
# Архитектура GAN-а

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.

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 ТВ, состоящую из FO и TO — причём ему известен тип каждого объекта;
* D по ТВ 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.**

**Этапы тренировки. Итерация обучения:**

1) генерация набора случайных чисел r1, r2, ..., rn;

2) G строит по ri (i = 1, n) объекты a1, a2, ..., an (это выборка FO);

3) берётся случайная выборка b1, b2, ..., bm 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.

7) У генератора есть функция сети FG(r), которая по случайному входу r генерирует объект. В нашем случае было так: **FG(r1)=a1, FG(r2)=a2,..., FG(rn)=an**.

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

**Мы хотим минимизировать вероятность фейковости выражений:**

Pr(FG(r1,w)∈ Fake), Pr(FG(r2,w)∈ Fake),...,Pr(FG(rn,w)∈ Fake)

То есть нужно **минимизировать сумму логарифмов**:

Ln(Pr(FG(r1,w)∈ Fake))+Ln(Pr(FG(r2,w)∈ Fake))+Ln(Pr(FG(rn,w)∈ Fake)) ->min

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