**Тезисы лекции**

**Атаки на нейронные сети.**

Нейронные сети сегодня везде, от бытовых приборов до космических кораблей, управляют автомобилями, ставят медицинские диагнозы и многое-многое другое. Решают многие задачи «лучше» человека.

Но лучше ли?

Это панда, так считает человек, так считает и нейронная сеть (GoogLeNet), которую обучили распознавать картинки, она говорит на 57.7% это панда.



А это?



Да это ж та же самая панда! - скажет человек

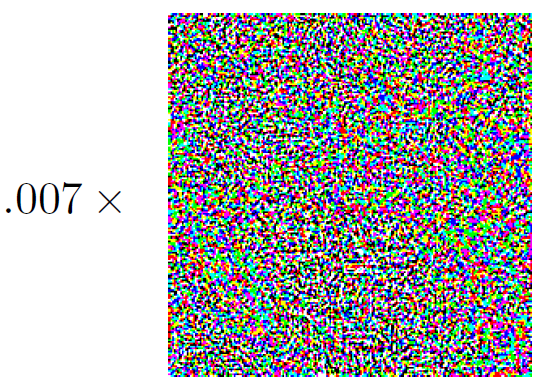
А такая нейронная сеть говорит – это на 99.3% гиббон!

ЧТО???

Да, да, нейронная сеть уверенно говорит, что это гиббон.

Эти картинки разные, но их разницу глазами мы не заметим, а вот нейронная сеть замечает.

Картинки отличаются лишь малюсенькой добавочкой: некоторым шумом с малым (0.007) коэффициентом:



Как же так?

Давайте попробуем разобраться.

Сеть – сверточная, в ней много слоев. Основная операция — это скалярное произведение весов и входов.

Для простоты будем говорить только об этой операции, но другие могут внести свой вклад также.

Выход для первой картинки с1=wTx , w – веса, x –вход.

Вторая картинка отличается от первой на маленький dx : x+dx

И выход будет с2=wT(x+dx)=c1+ wTdx, т.е. отличается на величину wTdx.

Маленькая ли это величина?

Когда как.

Скалярное произведение это сумма произведений. Если слагаемые разного знака и примерно одинаковые, они, при суммировании, будут компенсировать друг друга, и результат – сумма – будет маленькой.

А если все слагаемые одного знака и таких слагаемых много, то и сумма может быть большой, даже если dx маленькие! Можно принудительно сделать сумму большой: возьмем dx=sign(w). Тогда все слагаемые будут одного знака.

Но ведь выход и определяет класс изображения, какой больше – такой и класс.

Значит можно так *незначительно* поменять вход, добавив маленький dx, что результат поменяется существенно.

Мы провели *атаку* на нейронную сеть.

Во всем виновата размерность векторов, привет, *проклятие размерности*.

А картинки имеют много пикселей, все хотят ultraHD, с миллионами пикселей.



Для более сложных сетей, в которых много слоев, нелинейности, такая простая добавочка не сработает. Но можно найти другие!

**А) «Быстрый» метод знака градиента** (fast gradient sign method) [1].

Пусть сеть научена распознавать классы изображений (с – класс).

Найдем, как обычно, ошибку сети, разницу между выходом сети и желаемым выходом.

Эта ошибка, конечно, зависит от структуры сети, от весов в сети и от входов.

Найдем градиент (производную) ошибки по входу (т.е. по пикселям исходного изображения).

И сделаем добавочку со знаком плюс ко входу равную знаку этого градиенту с некоторым коэффициентом e. Плюс – потому что мы хотим ошибку сети увеличить!

x=x+ e\*sign(dJ(w; x; c))

Это направление наибольшего возрастания ошибки (сравни, в градиентном спуске мы искали направление наибольшего убывания ошибки).

Уже этого может оказаться достаточно. Если нет, повторим процесс.

И, конечно, ограничим коэффициент e малым числом, чтоб и добавочки были малыми.

Примеры см. в работе [1] и на сайте: <https://www.pluribus-one.it/research/sec-ml/demo>

(жаль, что мало примеров)

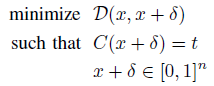
**Б)** **CW метод [2]** (по имени авторов)

*А если я хочу не просто чтоб сеть какой-то другой класс сказала, а именно тот, который мне нужен, хочу, чтоб панда стала самолетом, слабо, а? (вы конечно будете думать только о таких безобидных примерах, да?)*

Что ж задача ясна, есть цель, найти такие добавочки, которые

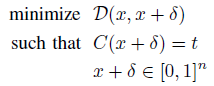
1. Малые
2. Дают нужный класс в ответе.

Математически:



D – мера (расстояние) отличия нормального примера от атакующего, дельта – атака, добавочка.

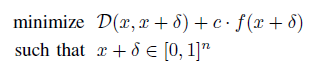
C – фактический ответ нейронной сети о том какой класс атакующего примера, t- целевой класс (все это вектора).

Обеспечить равенство  трудновато, даже имея мощные компьютеры (главный вопрос – как это сделать).

Поэтому вместо такой постановки решают похожую  f – некоторая специальная функция, сделанная так, что если она меньше или равна нулю, то тогда и равенство для C выполняется.

Таких функций много, см. стр. 6 в [2]. Какие-то лучше, какие-то хуже, стандарта нет.

С неравенством работать проще,



с – константа, регулирующая что важней малость атаки или успех атаки.

Берем компьютер, пусть себе минимизирует, все функции нам известны, дифференцируемы, градиентные методы помогут.

Результаты (см. картинки в [2]) удручают, цифры MNIST в любые другие можно перевести, другие картинки тоже.

Это самолет, по мнению сети, не панда:



А это грузовик:



К слову, тут атаки заметны глазу, потому что мы минимизируем суммарную меру отличия одной картинки от другой, а не отдельные пиксели.

**В) Однопиксельные атаки. Карты чувствительности.**

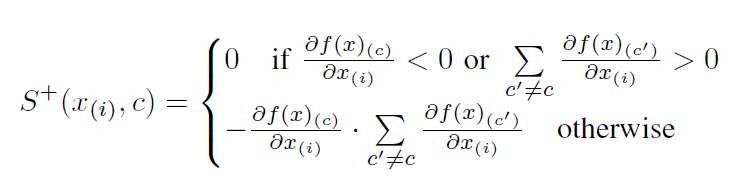
*Хм, хм, а можно чтоб только несколько пикселей поменять, и взломать сеть?*

Почему бы нет ☹

Давайте просто посчитаем, как каждый пиксель изменяет выход (классы) нейронной сети. И выберем такие которые наибольшую погрешность вносят, их и поменяем.

Чтоб посчитать, надо составит карту чувствительности.

Например так [3]:



Наша цель:

1. увеличивать выход для целевого, атакующего, класса c,
2. уменьшать выходы для всех других классов (c`), но не каждого по отдельности, а в совокупности.

Тогда, если изменение пикселя приводит к уменьшению на целевом классе (производная выхода по этому входу-пикселю меньше нуля), он нам не нужен. Ему ставим 0 в чувствительность. Также, если изменение пикселя ведет суммарно к увеличению (сумма производных больше нуля) выходов на других классах, это тоже не нужно. Верхняя строка в формуле.

Иначе пиксель нам подходит, чтоб оценить насколько он подходит, используем нижнюю строку. Тут все просто. Чем больше (она положительна) производная для целевого класса, тем лучше и, одновременно, чем больше *по модулю* сумма производных по другим классам (а так-то сумма отрицательна, поэтому минус стоит), тем лучше.

Производные по входам считать умеем - привет метод обратного распространения ошибки – считаем, получаем карту, изменяем наиболее чувствительные пиксели.

Если мало одного пикселя, сделаем тоже самое для пары пикселей, или большего их числа.

Ниже нарисованы, по мнению сети, песик (правда) и атаки: котик и лошадка. Примеры в [3].



**Г) Атаки в физическом мире [4]**

*Ой, ну ладно пугать, это все только в ваших компуктерах, на самом деле так не может быть!*

Увы, похожие атаки работают и в физическом мире. Лепите специальную наклейку на знак стоп, и оп, это уже для автомобиля знак увеличить скорость (не делайте так!).

Смысл такой же, есть цель, считаем ошибку, накладываем ограничения, и вперед, оптимизируем входы.

Тут по сложней, конечно, надо чтобы одна атака работала для картинок с разным ракурсом, снятыми разными камерами с разного расстояния…

Не беда, если есть мощный компьютер и много примеров нужных изображений – вперед, составляете ошибку, только теперь уже по всем примерам, и ее оптимизируете.

Авторы [4] сотворили такое со знаками дорожного движения.

*Не повторять дома!*

На 100% удалось обмануть сеть и принять знак стоп за ограничение скорости 45, наклеив на него, казалось бы, мусор.

Бойтесь автопилотов!

**Д) Развитие атак, защита.**

Чтоб проводить атаку по вышеописанным методам надо знать полностью нейронную сеть, которая используется, чтобы можно было считать производные.

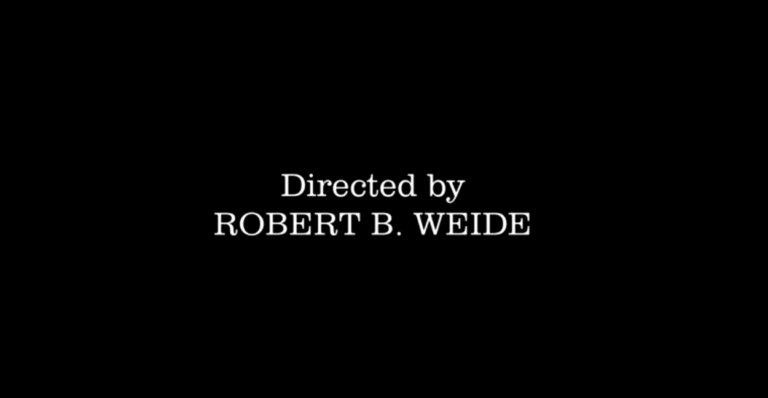
А если производитель не скажет? Безопасно? Если можем собрать примеры как та нейросеть отвечала на разные входы, то можем заменить ее своей, атаковать свою сеть, и попробовать на исходной. Есть вероятность что это сработает.

А только системы распознавания можно атаковать? Нет конечно. Боюсь этому подвержены практически все сети [5].

*Что же делать? Мы все умрем?*

**От атак можно и защититься**. Например, когда делаете сеть для распознавания, сами ее атакуйте, сделайте атакующие примеры. А потом дообучите вашу сеть, чтобы она и эти примеры распознавала правильно.

Ха-ха, а мы и эту новую атакуем. А мы и от этого защитимся. Это вечное противостояние атаки и защиты. А мы тогда…



**Ссылки:**

1. Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014).
2. Carlini, Nicholas, and David Wagner. "Towards evaluating the robustness of neural networks." 2017 ieee symposium on security and privacy (sp). IEEE, 2017.
3. Wiyatno, Rey, and Anqi Xu. "Maximal jacobian-based saliency map attack." arXiv preprint arXiv:1808.07945 (2018).
4. Eykholt, Kevin, et al. "Robust physical-world attacks on deep learning visual classification." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
5. Akhtar, Naveed, and Ajmal Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey." IEEE Access 6 (2018): 14410-14430.

**Другие ссылки:**

https://medium.com/onfido-tech/adversarial-attacks-and-defences-for-convolutional-neural-networks-66915ece52e7

https://habr.com/post/413775/

https://spark-in.me/post/playing-with-mcs2018-adversarial-attacks

https://github.com/snakers4/msc-2018-final

https://pablormier.github.io/2017/09/05/a-tutorial-on-differential-evolution-with-python/

https://habr.com/post/405857/

https://blog.openai.com/adversarial-example-research/

https://www.ibm.com/blogs/research/2018/05/clever-adversarial-attack/