Министерство образования и науки Российской Федерации

Санкт-Петербургский государственный политехнический университет

Институт Компьютерных наук и кибербезопасности

Высшая школа кибербезопасности

**Отчет**

по дисциплине «Технологии машинного обучения в кибербезопасности»

По теме:

«Защита от эксплуатации мобильности атак на модель машинного обучения»

Выполнила

студентка гр. 5141001/30301 А.А.Мажарцева

<*подпись*>

Преподаватель

доцент, к.т.н. М.О.Калинин

<*подпись*>

Санкт-Петербург

2023

Оглавление

[ВВЕДЕНИЕ 3](#_Toc149153876)

[1.1 Методы реализации атак 3](#_Toc149153877)

[1.2 Обучение замещающей модели 11](#_Toc149153878)

[2.1 Причины переносимости атак 12](#_Toc149153879)

[2.2 Методы защиты 13](#_Toc149153880)

[2.3 Эффективность применения методов защиты против существующих уязвимостей 15](#_Toc149153881)

[2.4 Решение 19](#_Toc149153882)

[3. Экспериментальная часть 19](#_Toc149153883)

[3.1 Обучение 19](#_Toc149153884)

[3.2 Моделирование атаки 20](#_Toc149153885)

[3.3 Результаты 21](#_Toc149153886)

# ВВЕДЕНИЕ

Защита от эксплуатации мобильности атак на модель машинного обучения.

Сценарий, при котором может происходить переносимость атак, представляет значительную угрозу, т. к. это позволяет нарушителям обходить защитные меры, применяемые к моделям машинного обучения.

Для обмана моделей машинного обучения создаются состязательные примеры — это входные данные, которые незначительно отличаются от оригинальных данных. Это отличие практически не заметно для человеческого глаза и обнаруживается в результате специального анализа. Состязательные примеры создаются, чтобы заставить модель принимать неверные решения, например классифицировать изображение кошки как собаки путем наложения шума.

Особенно уязвимы перед состязательными атаками сложные модели, чем сложнее модель, тем выше вероятность уязвимости. Например, самыми уязвимыми являются глубокие нейронные сети из-за сложных, нелинейных границ принятия решений. Также модели, переобученные на тренировочных данных, часто являются более уязвимыми.

Во многих работах было обнаружено, что состязательные примеры, применяемой к одной модели машинного обучения, могут успешно перенестись на другую модель машинного обучения. При этом модели могут иметь как разную архитектуру, так и обучаться на разном наборе данных. Это позволяет создавать атаки, которые можно направить на множество систем одновременно.

Неправильная классификация состязательных примеров представляет реальную угрозу, особенно в критически важных областях, таких как автомобили с автопилотом или мед. диагностика. Используя имеющиеся знания о переносимых атаках, можно создать более устойчивые модели, которые могут правильно обрабатывать состязательные примеры, разработать эффективные методы защиты от атак и обеспечить безопасность информационных систем, автоматических систем управления и т. д.

# Методы реализации атак

Методы реализации атак с переносом можно сгруппировать следующим образом:

1. По цели атаки:

* Нецелевая атака
* Целевая атака

1. По имеющимся знаниям атакующего

* Белый ящик
* Черный ящик
* Серый ящик

1. По методу создания адверсариалов:
   1. Атаки на основе градиентов

* Одношаговые методы: FGSM
* Многошаговые или итерационные методы: PGD, iFGSM, JSMA
  1. Атаки основанные на методе оптимизации С&W, DeepFool, L-BFGS
  2. Генеративные методы: GAN, VAE

1. По дополнительным стратегиям усиления переносимости:

* Атаки ансамбля: Создание состязательных примеров, используя несколько моделей одновременно.
* Вращения и преобразования: Генерация примеров, устойчивых к аффинным преобразованиям.

1. По различиям между замещающей (SN) и целевой моделями (TN):

* Cross Training transferability
* Cross Model transferability
* Cross Model and Training transferability

1. По механике переноса атак:

* Атаки на прямую: Перенос состязательных примеров, созданных для одной модели, на другую модель напрямую.
* Адаптивная атака: Атакующий адаптирует состязательные примеры перед переносом
* Вращения и преобразования
* FGSM
* JSMA
* DeepFool
* C&W
* Генеративные методы
* Атаки на основе замещающей модели (Черный ящик)
* Улучшение переносимости

Далее все рассматривается более подробно.

1. **По цели атаки:**

* Нецелевая атака: создание состязательного примера, ведущего к неправильной классификации
* Целевая атака означает изменение примера, чтобы его можно было классифицировать как желаемый класс **(С&W)**

1. **По имеющимся знаниям атакующего**
   1. **Белый ящик**: атаки с полным доступом. Нарушителю известна вся информация об обучаемой модели, включая параметры, набор обучающей выборки и архитектуру.
   2. **Черный ящик**: Атаки с ограниченным доступом. Нарушитель не имеет доступа к архитектуре и внутренним параметром модели, но может располагать входными и выходными данными. При данном методе нарушитель обучает экспериментальную модель и создает состязательные примеры для нее, которые затем использует на целевой модели

2.3 **Серый ящик**: Атаки, где атакующий знает часть информации о целевой модели.

1. **По методу создания состязательных примеров**:
   1. **Атаки на основе градиентов**
      1. **Одношаговые методы**: например, FGSM

**Fast Gradient Sign Method** – метод быстрого создания адверсариальных примеров. Главной идеей метода является внесение поправок к изображению, используя градиенты модели относительно этого изображения, чтобы нейронная сеть ошиблась в своем прогнозе, а человеческий глаз не заметил бы изменения

**Алгоритм FGSM**:

1. Атакующий рассчитывает градиент функции потерь относительно входных данных, чтоб понять как каждый пиксель входного изображения влияет на конечный результат.

Состязательное изображение создается с использованием информации о якобиане функции потерь:

,

Где х – входное изображение, J- функция потерь

1. Вместо использования самого градиента берется только его знак(то есть в какую сторону двигаться- увеличивать или уменьшать каждый пиксель)
2. Входное изображение изменяется в направлении этого знакового градиента. Величина изменения определяется параметром ε, который регулирует амплитуду искажений. Большое значение приведет к более сильному искажению, но также может стать заметно человеческому глазу. Маленькое значение ε может не вызвать искажений результатов модели
3. Получается измененное изображение, которое заставляет нейронную сеть ошибаться
   * 1. **Многошаговые или итерационные методы**:

* **Project Gradient Descent –** метод итеративной атаки на нейронные сети, при котором создаются изображения, которые выглядят нормально для человека, но заставляют нейронную сеть ошибаться.

**Алгоритм создания состязательныых примеров**:

1. Делается небольшое случайное возмущение входных данных
2. На каждом шаге вычисляют градиент функции потерь модели относительно входных данных, которые показывают в каком направлении менять изображение, чтобы эффективнее сбить модель
3. Используется знак градиента для создания возмущения
4. После каждого возмущения входные данные проецируются в допустимый диапазон. Это дает гарантию, что входные данные не изменяются слишком сильно, и их измененный вид похож на оригинал.
5. Процесс повторяется до достижения нужного возмущения

Новый вектор входных данных вычисляется по формуле:

,

– функция, которая обрезает значения в пределах заданного диапазона, где ε – величина максимального искажения.

функция потерь, α – коэффициент, который определяет величину шага

Эффективность атаки PGD в контексте переноса может зависеть от сходства архитектур и параметров "заменяющей" и целевой модели, а также от специфичности данных. К тому же настройка параметров PGD, таких как размер шага и число итераций, может влиять на успешность переноса адверсариальных примеров.

* **Итеративный FGSM(iFGSM)**

Вместо того, чтоб делать один шаг для создания адверсариалов, в iFGSM мы делаем множество маленьких шагов

Алгоритм:

1. Начинаем аналогично FGSM. Но вместо того, чтобы делать одно большое изменение в изображении, мы делаем много маленьких, применяя FGSM многократно
2. После каждого шага нужно проверяем, чтобы наше изображение не выходило за определенные границы относительно исходного изображения, то есть следить за тем, чтоб изменения были действительно маленькими и незаметными
3. Повторяем процесс, пока не получим изображение, которое обманывает модель

Преимущества iFGSM: это метод лучше исследует пространство вокруг входных данных, тем самым повышая вероятность найти направление, которое более эффективно нарушит работу модели. И так как после каждого шага происходит проверка изображение, то это помогает избежать создания легко обнаружимых примеров.

* **JSMA**

**Jacobian-based Saliency Map Attack –** метод атаки, заключающийся в создании адверсариалов, меняя минимальное количество входных элементов, определяя какие пиксели или элементы входных данных наиболее важны.

Алгоритм:

1. Атакующий хочет узнать какой пиксель можно изменить, чтобы максимально повлиять на результат. Создается Jacobian Matrix- матрица производных, которая показывает, как изменение в каждом входном пикселе влияет на каждый выходной элемент модели.
2. На основе Якобиана создается карта важности - Saliency Map, которая показывает, какие пиксели сильнее влияют на изменение результата модели
3. Используя карту выделяющихся элементов, изменяются пиксели так, чтоб максимально исказить результат, при этом изменив минимальное число пикселей
   1. **Атаки основанные на методе оптимизации:**

* **Carlini and Wagner Attack(С&W)**

Состязательная атака, предложенная

Карлини и Вагнером, на сегодняшний день является одной из самых сильных атак. Они формулируют целевые состязательные атаки как задачу оптимизации, используют внутренние конфигурации целевой DNN для руководства атакой и используютл2норма (т. е. евклидово расстояние) для количественной оценки разницы между состязательным и исходным примерами. В частности, представление на логит-уровне (уровень, предшествующий последнему полностью связанному уровню, как показано на рисунке 2) используется в качестве индикатора эффективности атаки. Следовательно, атаку C&W можно рассматривать как целевую состязательную атаку, основанную на градиентном спуске.

Алгоритм:

1. Создается функция потерь, которая при правильной классификации модели, увеличивает вероятность неправильного класса и уменьшает вероятность правильного. И обратное, при неправильной классификации
2. Далее вводится «шум», который добавляется к исходному изображению. Такое изменение делает изображение трудным для модели, но оставляет понятным для человека
3. Используя методы оптимизации, находится такой шум, чтоб функция потерь была минимальной
4. После нахождения нужного шума он накладывается на изображение и проверяется предсказание модели. Все изменения делаются незаметными для человека

* **DeepFool**

Данный метод предназначен для создания минимальных искажений, которые могут обмануть сеть. Метод ищет наименьшие возмущения, необходимые для пересечения границы принятия решений модели, при этом один класс будет относиться к любому другому, а не к одному конкретному. Минимальное искажение находится путем аппроксимации границ принятия решений и поиска наименьшего значения, необходимого для пересечения этой границы.

* **L-BFGS**

**Limited-memory Broyden-Fletcher-Goldfarb-Shanno –** итерационный метод решения задач оптимизации, ускоряющий метод градиентного спуска. Он минимизирует функцию потерь в направлении целевого класса, делая при этом минимальные искажения. При этом он приближает матрицу вторых производных функции потерь (Матрицу Гессиана) без ее явного вычисления, используя ограниченное количество последних обновлений вектора, что позволяет использовать гораздо меньше памяти, поэтому применим для работы с большими наборами параметров

Алгоритм:

1. Выбирается начальное приближение обратной матрицы Гессиана
2. Вычисляется градиент целевой функции в выбранной точке
3. Приближенно определяется направление поиска, используя текущий градиент и сохраненные значения градиентов и обновлений параметров с предыдущих шагов.
4. Определяется оптимальный размер шага, например, с помощью поиска по строке.
5. Обновляются параметры в выбранном направлении на определенный шаг, и сохраняются для будущих итераций

Старые значения градиентов и обновлений параметров удаляются, чтобы хранить только последние "m" обновлений (где "m" — это параметр, определяющий "ограниченную память" в L-BFGS).

* 1. **Генеративные методы:**
* **GAN**

**Generative Adversarial Network** – атаки с использованием генеративных состязательных сетей. В атаке создаются две нейронные сети: генератор и дискриминатор. Генератор обучается создавать состязательные примеры, способные обмануть целевую модель, путем генерации данных, максимально похожих на реальные, но с небольшими изменениями. Дискриминатор обучается находить различия между реальными и искаженными данными. Генератор пытается создать примеры, которые будут не отличимы от реальных для дискриминатора.

* **VAE**

**Variational Autoencoders** представляют собой класс генеративных моделей, которые обучаются кодировать и декодировать данные. Они широко используются для генерации новых примеров данных (например, изображений или текста) на основе обученной модели. Атаки на VAE обычно связаны с попытками манипулирования входными данными или внутренним представлением модели с целью дезориентировать ее или заставить работать неправильно. Вот некоторые примеры атак на VAE:

Атаки на реконструкцию: Целью здесь является искажение входных данных таким образом, чтобы VAE декодировал его в искаженное или нерелевантное представление.

Атаки на генерацию: В этом случае злоумышленник может попытаться искажать латентное пространство (скрытое представление данных) таким образом, чтобы VAE генерировал нежелательные или искаженные примеры.

Отравление данных: Эта атака включает в себя внедрение искаженных или злоумышленных примеров в обучающий набор данных, чтобы VAE впоследствии демонстрировал неправильное поведение.

Атаки на приватность: поскольку VAE может обучаться на чувствительных данных, злоумышленник может попытаться восстановить исходные данные на основе выходных данных модели или ее внутренних представлений.

1. **По дополнительным стратегиям усиления переносимости атаки:**

* **Ансамблевые методы**: Атаки на основе ансамбля моделей

Адверсариальные примеры генерируются на основе нескольких "заменяющих" моделей. Если состязательные примеры могут обмануть несколько моделей, они могут продемонстрировать хорошую переносимость на другие модели, такимобразом, атака на ансамбль моделей обеспечивает более мощные атаки «черного ящика». Примеры, созданные таким образом, применяются к целевой модели.

* **Вращения и преобразования**: Генерация примеров, устойчивых к аффинным преобразованиям.

Аффинные преобразования — это комбинация линейных преобразований (таких как масштабирование, вращение и сдвиг) и трансляции. В контексте изображений это может включать: Вращение изображения на определенный угол, масштабирование (увеличение или уменьшение), сдвиг вдоль осей X или Y, наклон или искажение, когда создаются состязательные примеры, учитывающие такие преобразования, это обычно делается с целью усиления их переносимости или устойчивости к методам обороны.

1. **По различиям между замещающей (SN) и целевой моделями (TN)**

• Cross Training transferability: этот сценарий представляет собой половинчатое несоответствие, когда SN и TN используют одну и ту же сетевую архитектуру. Обе сети обучаются на двух разных наборах данных. Когда наборы данных не

совпадают, но архитектуры совпадают, атаки градиентные и оптимизационные атаки менее переносимы. Одним из правдоподобных объяснений является то, что, хотя атаки затрагивают лишь небольшой процент выборок, атакованная модель имеет тенденцию к большему переобучению. Также стоит отметить, что возможность переноса наборов обучающих данных для конкретной задачи не симметрична. Это подразумевает, что в основном в компьютерных сетях базовый набор данных может каким-то образом и в некоторой степени влиять на функции, изученные сетью

• Cross Model transferability: возможность переноса кросс-моделей также является параметром половинного несоответствия. В этом сценарии мы обучаем один и тот же набор данных на двух разных архитектурах

• Cross Model and Training transferability: этот сценарий представляет собой полное несоответствие, когда SN и TN имеют разные архитектуры, а также обучаются на разных наборах данных. В случае полного несоответствия мы приходим к выводу, что SN и TN непередаваемы.

1. **По механике переноса атак**:

* **Возможность переноса:**

В условиях враждебных атак возможность передачи означает, что состязательные примеры, созданные на основе одной модели, с большой вероятностью будут неправильно классифицированы другой моделью. В частности, вышеупомянутые состязательные атаки продемонстрировали, что их состязательные примеры легко переносятся с одной имеющейся DNN на целевую DNN. Одно из возможных объяснений присущей DNN возможности переноса атак заключается в том, что DNN обычно обладают подавляющей способностью обобщения и локальной линейностью для извлечения признаков. Примечательно, что возможность передачи состязательных атак вызывает проблемы с безопасностью приложений машинного обучения, основанных на DNN, поскольку вредоносные примеры могут быть легко созданы, даже если точные параметры целевой DNN отсутствуют. Более интересно, что тщательно созданное универсальное возмущение набора естественных изображений может с высокой вероятностью привести к неправильной классификации всех рассматриваемых изображений, что предполагает возможность переноса атаки с одного изображения на другое

* 1. **Атаки на прямую**: Перенос состязательных примеров, созданных для одной модели, на другую модель напрямую.

Поскольку атакующий имеет полный доступ к модели, он может напрямую вычислять градиенты на этой модели и создавать адверсариальные примеры. Такие атаки часто бывают эффективными, так как атакующий может точно определить, как изменить входные данные для максимального воздействия. Следует отметить, что атаки "белого ящика", строго говоря, не являются атаками переноса, так как они напрямую нацелены на целевую модель. Однако такие атаки часто служат базой для разработки методов переноса.

* 1. **Адаптивная атака:** Атакующий адаптирует состязательные примеры перед переносом

Под адаптивной атакой понимается методика, при которой атакующий адаптирует или модифицирует свои состязательные примеры на основе ответов или другой обратной связи от модели. Эта адаптация может происходить как при доступе к градиентам и параметрам модели (белый ящик), так и без прямого доступа к ним (черный ящик).

**Адаптация может происходить на основе**:

* Обратной связи: если известно, успешна ли атака или нет, можно корректировать параметры атаки соответствующим образом.
* Знания о модели: Знание о внутренней структуре модели или ее архитектуре может указать на определенные уязвимости.
* Знания о данных: Знание о распределении и характеристиках входных данных может помочь в создании более эффективных атак.

Черный ящик

Для черного ящика градиенты недоступны. Поэтому строится замещающая модель, которая для атакующего и будет белым ящиком.

# 1.2 Обучение замещающей модели

Простейшая атака черного ящика может выглядеть следующим образом:

1. Выполнить запросы целевой модели с входными данными Xi для i = 1… n и сохранить выходы yi

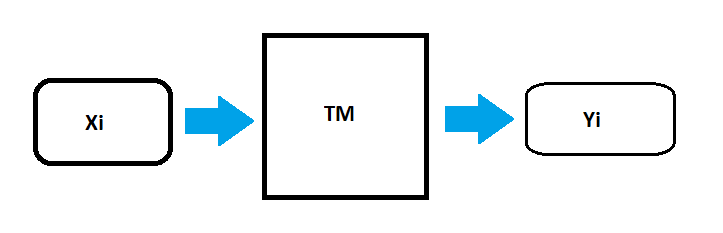


Рисунок 1 – Определение меток классов на основе ответов целевой модели

1. С обучающими данными (Xi, yi) построить другую модель, называемую замещающей моделью.

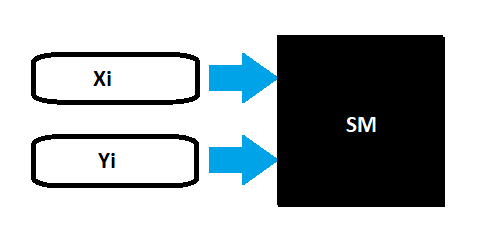


Рисунок 2 – Обучение замещающей модели

3. Далее генерируются состязательные примеры для замещающей модели. Многие из них могут быть успешно перенесены и станут состязательными примерами для целевой модели

Обучение замещающей модели на синтетическом наборе данных, аналогичном набору данных, на котором обучается модель черного ящика. Но принципиально, что метка для синтетического набора данных должна исходить из прогноза модели черного ящика

# Причины переносимости атак

Причины переносимости атак между моделями могут зависеть как от архитектуры модели, так и от набора данных.

Выделяют несколько главных уязвимостей моделей**:**

1. **Уязвимости, связанные с архитектурой:**

* **Линейные свойства архитектур**
* **Схожесть архитектур разных моделей**
* **Недостаток регуляризации**

1. **Уязвимости, связанные с данными:**

* **Наличие общего набора данных**
* **Высокая размерность входных данных**
* **Высокая корреляция признаков**
* **Линейные свойства архитектур**: многие глубокие нейронные сети, несмотря на их сложность, на фундаментальном уровне базируются на линейных операциях. Эта основная линейность может привести к тому, что даже небольшие изменения во входных данных приведут к значительным отклонениям в выходах, создавая уязвимость к состязательным атакам
* **Схожесть архитектур**: если две модели имеют похожую архитектуру, но были обучены на разных датасетах, состязательные атаки, как правило, менее эффективны.

Однако, если модели имеют разную архитектуру и обучались на разных датасетах, вероятность успешной атаки снижается.

* **Недостаток регуляризации**: Отсутствие или недостаточное применение регуляризации может увеличить уязвимость модели к состязательным атакам. Усиление регуляризации часто снижает амплитуду входных градиентов, делая модель менее чувствительной к маленьким возмущениям входных данных.
* **Общий набор обучающих данных**: Многие модели обучены на одинаковых или схожих наборах данных и, следовательно, вырабатывают схожие внутренние представления. Это делает их уязвимыми к атакам, созданным для одной модели, которые эффективно работают и на других моделях. Если две модели имеют разную архитектуру, но обучались на одном наборе данных, то в большинстве случаев атака переносится успешно
* **Высокая размерность данных**: Входные данные, такие как изображения или текст, обычно имеют высокую размерность. Это позволяет злоумышленникам вносить тонкие, почти незаметные изменения, которые могут сильно засорить выводы модели.
* **Высокая корреляции признаков между моделями**

Когда разные модели обучаются на одних и тех же данных, они часто извлекают схожие или коррелированные признаки из этих данных, особенно если архитектуры моделей схожи. Это означает, что модели "смотрят" на одни и те же ключевые аспекты данных для выполнения своих задач (например, классификации). Если одна модель уязвима для определенного враждебного примера, то другая модель, извлекающая аналогичные признаки, также может быть уязвима

# Методы защиты

1. **Защита на уровне архитектуры**

* **Использование менее уязвимых архитектур**
* **Ансамблирование[2],[6]**
* **Защитные механизмы на основе обнаружения**

1. **Защита на уровне данных**

* **Состязательное обучение [2]**
* **Обучение с регуляризацией**
* **Аугментация входных данных**
* **Потеря корреляции [2]**

1. **Состязательное обучение** — это метод обучения, в котором модель обучается распознавать и противостоять состязательным атакам путем внедрения таких атак в процесс обучения. Это делается для улучшения устойчивости модели к подобным атакам в будущем.
2. **Обучение с регуляризацией** применяется для предотвращения переобучения модели, добавляя штраф за сложность. Это делает модель менее чувствительной к шумам в данных и может предотвратить атаки, использующие эти шумы.
3. **Аугментация входных данных** представляет собой процесс искажения и изменения входных данных без изменения их исходного значения. Это может включать в себя вращение, масштабирование, изменение освещенности и другие преобразования. Аугментация может увеличить устойчивость модели к состязательным атакам, потому что модель обучается на более разнообразных данных.
4. **Потеря корреляции** - Введение потери корреляции (или декоррелирующего члена) в целевую функцию обучения служит способом регуляризации, который побуждает две модели извлекать разные, менее коррелированные признаки. Это не обязательно означает, что модели будут игнорировать главные признаки, но они будут "смотреть" на данные немного по-разному. Цель здесь - сделать так, чтобы атаки, эффективные против одной модели, не были такими же эффективными против другой, благодаря различиям в извлекаемых признаках. Таким образом, введение потери корреляции может уменьшить риск того, что атака, спроектированная для одной модели, будет успешно переносима на другую модель.
5. **Ансамблирование**, или использование ансамбля моделей, предполагает объединение прогнозов нескольких моделей для достижения более надежного результата. Ансамбли могут сделать атаку на модель сложнее, так как злоумышленнику придется создавать состязательные примеры, которые будут эффективными против всех моделей в ансамбле.
6. **Защитные механизмы на основе обнаружения** стремятся распознавать состязательные примеры на этапе входа в модель или систему. Такие механизмы могут быть базированы на различных принципах, таких как аномалии в распределении данных, специфические шаблоны в состязательных примерах и др.
7. **Использование менее уязвимых архитектур**, таких как LSTM модели или деревья решений, может предложить определенный уровень защиты от состязательных атак.

# Эффективность применения методов защиты против существующих уязвимостей

1. **Состязательное обучение:**

* **Линейные свойства архитектур:**

Состязательное обучение позволяет модели "научиться" более сложным и нелинейным отношениям в данных, уменьшая зависимость от линейности архитектуры и делая модель менее предсказуемой для атакующего.

* **Недостаток регуляризации**

Состязательное обучение добавляет робастности в процесс обучения, действуя как форма регуляризации, поскольку модель учится устойчиво классифицировать данные даже при наличии враждебных возмущений.

* **Наличие общего набора данных**

Модели, обученные с помощью состязательного обучения на одном наборе данных, могут стать менее уязвимыми к атакам, созданным на основе этого же набора.

* **Высокая размерность входных данных**

Состязательное обучение учит модель правильно интерпретировать и классифицировать данные даже при наличии небольших возмущений в высокомерном пространстве, что делает атаки менее эффективными.

* **Высокая корреляция признаков**

Состязательное обучение учит модель распознавать и учитывать состязательные возмущения, которые могут быть основаны на коррелированных признаках, улучшая её устойчивость к атакам

1. **Обучение с регуляризацией**

* **Линейные свойства архитектур**

Регуляризация может помочь уменьшить зависимость модели от линейных свойств её архитектуры, препятствуя созданию слишком сложных или чрезмерно подгоняемых к данным моделей.

* **Наличие общего набора данных**

Регуляризация может уменьшить уязвимость модели к состязательным атакам, основанным на общем наборе данных, так как она уменьшает риск переобучения на частных особенностях этого набора данных.

* **Высокая размерность входных данных**

Регуляризация препятствует излишней сложности модели и ограничивает её способность к "запоминанию" обучающих данных, что может помочь в борьбе с уязвимостями, вызванными высокой размерностью входных данных.

* **Высокая корреляция признаков**

Регуляризация может помочь модели обращать меньше внимания на сильно коррелированные признаки или "шумные" признаки, уменьшая зависимость от них и делая модель более робастной к состязательным атакам, которые могут эксплуатировать эти корреляции

1. **Аугментация входных данных**

* **Линейные свойства архитектур**

Может помочь "разбавить" линейные свойства модели, тренируя ее на различных искажениях данных. Это может сделать модель менее чувствительной к маленьким изменениям входных данных, характерным для состязательных атак.

* **Наличие общего набора данных**

Обучение на аугментированных данных может уменьшить уязвимость модели к атакам, специально разработанным для моделей, обученных на конкретном наборе данных. Аугментация добавляет вариативность в обучение, делая модель менее чувствительной к специфическим особенностям обучающего набора.

* **Высокая размерность входных данных**

При аугментации данные искажаются различными способами, что может помочь модели лучше обобщать в высокоразмерном пространстве, уменьшая вероятность успешной состязательной атаки.

* **Высокая корреляция признаков**

Аугментация может вносить изменения в корреляционные структуры данных, заставляя модель обучаться на менее коррелированных данных и становиться менее чувствительной к атакам, которые эксплуатируют эти корреляции

1. **Потеря корреляции**

* **Линейные свойства архитектур**

Уменьшение корреляции между признаками может усложнить эксплуатацию линейных свойств модели для создания состязательных атак.

* **Недостаток регуляризации**

Уменьшение корреляции может действовать как форма регуляризации, уменьшая зависимость модели от определенных особенностей обучающего набора данных.

* **Наличие общего набора данных**

Если модели обучаются на аналогичных данных, то атакующий может попытаться эксплуатировать это сходство. Путем уменьшения корреляции признаков можно усложнить создание состязательных примеров, которые работают на разных моделях, обученных на общем наборе данных.

* **Высокая размерность входных данных**

В высокоразмерных данных часто много коррелированных признаков. Уменьшение корреляции может сделать состязательные атаки менее эффективными, так как атакующему становится сложнее эксплуатировать корреляции в данных.

1. **Ансамблирование**

* **Линейные свойства архитектур**

Ансамблирование может объединять модели с различными архитектурами или параметрами, что может усложнить эксплуатацию линейных свойств отдельных моделей.

* **Схожесть архитектур разных моделей**

Если ансамбль включает модели с различными архитектурами, то это может затруднить атаку, потому что состязательный пример, который эффективен против одной модели, может не быть таким же эффективным против другой модели в ансамбле.

1. **Защитные механизмы на основе обнаружения**

Могут выявлять необычные паттерны в входных данных, указывающие на возможную атаку вне зависимости от существующих уязвимостей модели

1. **Использование менее уязвимых архитектур**

* **Линейные свойства архитектур**

Некоторые архитектуры могут быть менее уязвимы к состязательным атакам из-за их нелинейных свойств. Исследования показали, что выбор подходящей архитектуры может уменьшить возможность успешной атаки.

* **Недостаток регуляризации**

Некоторые архитектуры могут включать в себя встроенные механизмы регуляризации или быть менее чувствительными к переобучению, что делает их менее уязвимыми к атакам.

* **Высокая размерность входных данных**

Некоторые архитектуры могут эффективнее работать с высокоразмерными данными, обеспечивая лучшую регуляризацию и уменьшая уязвимость к атакам в таких пространствах.

* **Высокая корреляция признаков**

Используя архитектуры, которые могут лучше обрабатывать коррелированные признаки, например, с помощью механизмов внимания или других методов, которые могут уменьшить зависимость от коррелированных признаков, можно снизить уязвимость к состязательным атакам.

Таблица 1 – Сравнение методов защиты

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Линейные свойства архитектур | Общий набор данных | Схожесть архитектур | Высокая размерность входных данных | Недостаток(отсутствие) регуляризации | Высокая корреляции признаков между моделями | Простота реализации |
| Состязательное обучение | + | + | - | + | + | + | + |
| Обучение с регуляризацией | + | + | - | + | + | + | ++ |
| Аугментация входных данных | + | + | - | + | - | + | ++ |
| Потеря корелляции | + | + | - | + | + | + | - |
| Ансамблирование | + | + | + | + | + | + | +/- |
| Защитные механизмы на основе обнаружения | + | + | + | + | + | + | - |
| Использование менее уязвимых архитектур | + | - | + | + | + | + | - |

# 2.4 Решение

Комбинированный метод защиты: Адверсальное обучение поможет ей устоять перед многими атаками. Обучение с регуляризацией, включающее добавление регуляризующего элемента в функцию потерь, чтобы предотвратить переобучение, так модель будет менее чувствительной к незначительным изменениям во входных данных. Аугментация входных данных путем внесения изменений таких как вращение, горизонтальное отражение позволит модели быть устойчивее к различным изменениям.

# Экспериментальная часть

# Обучение

# 3.1.1 Обучение на одном наборе данных

Для проверки надежности защиты брался код для сверточной нейронной сети (ConvNet / CNN) для классификации изображений CIFAR[3]. Набор данных CIFAR10 содержит 60 000 цветных изображений 32\*32 в 10 классах, по 6000 изображений в каждом классе. Набор данных разделен на 50 000 обучающих изображений и 10 000 тестовых изображений. Эти классы являются взаимоисключающими, и между ними нет никакого перекрытия.

Замещающая модель (Source model) обучалась только на датасете Cifar-10. К целевой модели (Target model) была дополнительно применена регуляризация[4] и добавлены аугментированные и состязательные примеры в датасет, архитектура использовалась та же самая (Рисунок 4). При этом точность модели выросла на 4%.

Для того, чтобы помочь модели стабилизировать процесс обучения и улучшить ее способность обобщения на новые данные использовалась лямбда-функция lambda x: 1e-3 \* 0.95 \*\* x. Она используется для определения изменения скорости обучения. В данном случае начальная скорость обучения устанавливается на 1e-3, и с каждой эпохой она умножается на 0.95. Это позволяет плавно уменьшать скорость обучения во время обучения модели, что может быть полезным для улучшения ее стабильности и качества.

Для генерации обучающих состязательных примеров в первом случае использовался FGSM, с параметром искажения ε = 0,01. Во втором случае использовался более мощный метод состязательного обучения - PGD с ε = 0,01, α=0,001, и 10 шагами итераций.

Точность модели оценивалась с использованием метрики accuracy — доля правильных ответов алгоритма:

где TP, TN – верно классифицированные ответы, FP,FN – неверно классифицированные

Изображение выглядит как текст, снимок экрана, Шрифт, диаграмма

Автоматически созданное описание

Рисунок 4 – Обучение целевой и замещающей моделей при одинаковых датасетах

# Обучение на разных наборах данных

Чтобы оценить, как атака переносится с одной модели на другую в зависимости от обучающего набора данных, был рассмотрен подход обучения целевой и замещающей моделей на разных наборах данных. Целевая модель также была обучена на наборе данных CIFAR-10 с аугментированными и состязательными примерами и обучалась с использованием регуляризации, в то время как замещающая модель была обучена на наборе данных cats\_vs\_dogs, состоящем из 25000 изображений 200\*200 (12500 кошек и 12500 собак) [9].Этот подход был использован для оценки возможности передачи атак между моделями, обученными на разных наборах данных.

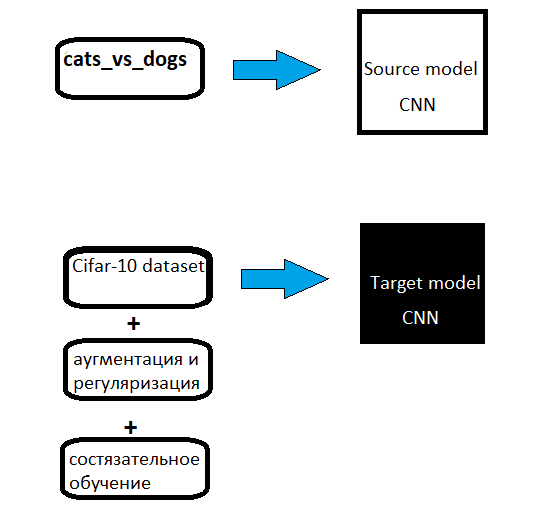


Рисунок 4 – Обучение целевой и замещающей моделей при разных датасетах

# 3.2 Моделирование атаки

Изображение выглядит как текст, снимок экрана, диаграмма, Шрифт

Автоматически созданное описание

Рисунок 6 – Пример применения состязательной атаки

Для проведения атаки проводилась генерация состязательных примеров с использованием метода FGSM и PGD [5]. Состязательные примеры сначала подавались на замещающую модель, затем на целевую. Смотрелась точность правильных ответов. Полученные данные представлены в таблице 2. Параметр искажения ε принимался равным ε = 0,01; 0,02; 0,05. Как видно из рисунка 6, искажения слабо заметны человеческому глазу, но сильно влияют на качество работы модели.

Изображение выглядит как Красочность, шаблон, искусство, снимок экрана

Автоматически созданное описание Изображение выглядит как Красочность, снимок экрана, шаблон, искусство

Автоматически созданное описаниеИзображение выглядит как Красочность, снимок экрана, искусство, шаблон

Автоматически созданное описание

а) б) в)

Рисунок 7 – Примеры действия состязательного искажения FGSM при а) ε = 0,01; б) ε = 0,02; в) ε = 0,05

# Результаты

Таблица 3 – Результаты проведения атаки

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Коэффициент искажения ε | Точность ответов замещающей модели | Точность ответов целевой модели, дообученной с помощью FGSM | Точность ответов целевой модели, дообученной с помощью pgd | Точность ответов целевой модели, обученной на другом датасете | Точность ответов целевой модели, обученной на другом датасете и дообученной с PGD |
|  | 0 | 86% | 89% | 84% | 93% | 97% |
| FGSM | 0,003 | 68% | 72% | 71% | 61% | 95% |
| 0.005 | 56% | 69% | 70% | 61% | 95% |
| 0,007 | 40% | 63% | 64% | 61% | 96% |
| 0.01 | 26% | 55% | 62% | 60% | 94% |
| 0.02 | 6% | 35% | 46% | 55% | 95% |
| PGD | 0.001 | 80% | 72% | 72% | 61% | 95% |
| 0.002 | 74% | 73% | 77% | 60% | 95% |
| 0.003 | 66% | 71% | 73% | 61% | 95% |
| 0.004 | 55% | 69% | 70% | 62% | 95% |
| 0.005 | 43% | 70% | 69% | 60% | 95% |
| 0.007 | 30% | 66% | 69% | 61% | 95% |
| 0.009 | 19% | 61% | 65% | 59% | 95% |
| 0.01 | 16% | 59% | 64% | 59% | 95% |

Рисунок 8 – Изменение точности ответов моделей в зависимости от коэффициента искажения ε при проведении атаки FGSM

Рисунок 9 – Изменение точности ответов моделей в зависимости от коэффициента искажения ε при проведении атаки PGD

# Выводы

Атаки на модели, обученные на одном датасете: При проведении атаки с использованием FGSM, с увеличением коэффициента искажения (ε = 0,001-0,02), спад точности ответов наблюдается с 86% до 6% у замещающей модели, а для целевых, дообученных с помощью FGSM и PGD соответственно: с 89% до 35% и с 84% до 46%. При проведении атаки с использованием PGD (ε = 0,001-0,01), спад точности ответов наблюдается с 86% до 16% у замещающей модели, а для целевых, обученных с помощью FGSM и PGD соответственно: с 89% до 59% и с 84% до 64%. Результаты в целом демонстрируют улучшение устойчивости моделей.

При переносе атаки на модель, обученную на другом датасете наблюдается изначальный резкий спад точности ответов модели( с 93% до 61%), который не изменялся до конца наблюдений. При дообучении данной модели с помощью PGD наблюдается полная устойчивость модели

Наблюдаемое повышение устойчивости модели, обученной на другом датасете, подчеркивает важность использования разнообразных данных для обучения моделей. Это указывает на необходимость рассмотрения дополнительных методов для создания более обобщающих и устойчивых моделей машинного обучения.

# Список литературы

[1] Nowroozi, E., Mekdad, Y., Berenjestanaki, M. H., Conti, M. & Fergougui, A. El. Demystifying the Transferability of Adversarial Attacks in Computer Networks. IEEE Trans. Netw. Serv. Manag. (2022) doi:10.1109/TNSM.2022.3164354.

[2] Disrupting Adversarial Transferability in Deep Neural Networks Christopher Wiedeman1 , Ge Wang\* 2

[3] [Классификация изображений с помощью CNN для начинающих | Каггл (kaggle.com)](https://www.kaggle.com/code/anandhuh/image-classification-using-cnn-for-beginners/notebook)

[4] [Custom CNN + Regularization techniques | Kaggle](https://www.kaggle.com/code/amruthaa08/custom-cnn-regularization-techniques)

[5] [DNA/classifier\_exp/attacks.py at main · WANG-AXIS/DNA (github.com)](https://github.com/WANG-AXIS/DNA/blob/main/classifier_exp/attacks.py)

[6] Duan, Y. et al. Enhancing transferability of adversarial examples via rotation-invariant attacks. IET Comput. Vis. (2022) doi:10.1049/cvi2.12054.

[7] Zhou, Y., Kantarcioglu, M. & Xi, B. Exploring the Effect of Randomness on Transferability of Adversarial Samples Against Deep Neural Networks. IEEE Trans. Dependable Secur. Comput. (2023) doi:10.1109/TDSC.2021.3127439.

[8]  Inkawhich, N. et al. Perturbing across the feature hierarchy to improve standard and strict blackbox attack transferability. in Advances in Neural Information Processing Systems(2020)

[9] https://www.kaggle.com/code/hikmatullahmohammadi/keras-cats-vs-dogs-classification-step-by-step