Введение в атаки отравлением на модели машинного обучения

Д.Е. Намиот

***Аннотация*—В настоящей статье рассматривается один из возможных классов атак на системы машинного обучения – атаки отравлением. Классически, атаки отравлением – это специальные модификации тренировочных данных, которые призваны воздействовать на полученную после обучения модель необходимым атакующему образом. Атаки могут быть направлены на то, чтобы понизить общую точность или честность модели, или же на то, чтобы, например, обеспечить, при определенных условиях, необходимый результат классификации. Техника осуществления такого рода атак включает алгоритмы для определения элементов тренировочных данных, в наибольшей степени ответственных за результаты обучения (за выработанные обобщения), минимизацию количества отравляемых данных, а также за обеспечение максимальной незаметности проводимых изменений. Среди атак отравления наиболее опасными являются так называемые трояны (бэкдоры), когда посредством специальным образом подготовленных тренировочных данных добиваются изменения логики работы модели для определенным образом помеченных входных данных. Помимо модификации тренировочных данных к атакам отравления относят также прямые атаки на готовые модели машинного обучения или их исполняемый код.**

***Ключевые слова*—машинное обучение, кибератаки, отравление данных, трояны, бэкдоры**

I. Введение

Эта статья является продолжением серии публикаций, посвященных атакам на системы машинного обучения [1, 2]. Она подготовлена в рамках проекта кафедры Информационной безопасности факультета ВМК МГУ имени М.В. Ломоносова по созданию и развитию магистерской программы "Искусственный интеллект в кибербезопасности" [3].

Атаки на системы машинного обучения – это сознательная модификация данных на различных этапах конвейера машинного обучения, призванная либо снизить общее качество работы системы (вплоть до полной неработоспособности), либо добиться желаемого функционала (показателей работы). Если специальных целей по тому, как должна работать атакованная система нет, то говорят о нецелевых атаках, в противном случае – атаки целевые.

Статья получена 14 декабря 2022. Исследование выполнено при поддержке Междисциплинарной научно-образовательной школы Московского университета «Мозг, когнитивные системы, искусственный интеллект»

Д.Е. Намиот – МГУ имени М.В. Ломоносова (email: dnamiot@gmail.com)

Системы машинного обучения (как это не удивительно!) зависят от данных и изменения в данных меняют работу модели. Данные на этапе тренировки модели непосредственно эту модель и определяют. Соответственно, модификации тренировочных данных изменят (могут изменить) модель по сравнению с оригинальными данными. Для двух наборов данных на рисунке 1 обобщения в результате обучения будут разными (угол наклона регрессионной прямой или гиперплоскости SVM разный).

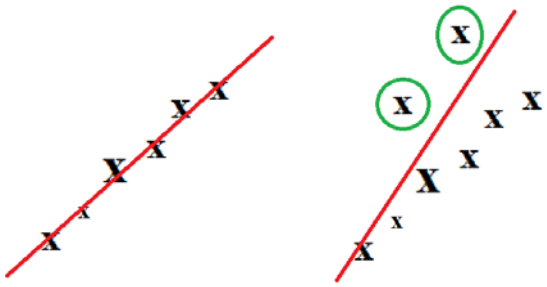

Рис. 1. Зависимость модели от тренировочных данных

Вопрос: обведенные зеленым две точки в правом наборе данных – это реальные данные, аномалии (ошибка измерения) или сознательно помещенные в тренировочный набор данные, призванные изменить работу модели?

Как такие специальные данные могли попасть в тренировочный набор (если исключить умысел разработчиков)? На самом деле, тут все достаточно просто. Данные для тренировки могли быть загружены их какого-то публичного репозитория, куда их разместил атакующий. Это наиболее реальный путь – вспомните, когда вы последний раз занимались анализом загруженных датасетов на предмет наличия отравленных данных?

Другой, также абсолютно реальный путь - это оутсорсинг. Нужные атакующему данные могли возникнуть во время проведения “правильной” разметки.

Еще один момент, который необходимо отметить – это дообучение работающих моделей. Например, рекомендательная система в электронной коммерции дообучается по свежим данным о предпочтениях пользователей. И фальшивые отзывы (лайки) и т.п. – это именно отравление данных.

Состязательные тренировки, когда к тренировочному набору добавляются модифицированные данные с целью обучить модель противостоять атакам уклонения, часто могут приводить к отравлению данных [20].

Точно также, как и с атаками уклонением (состязательными атаками, как их часто называют) такие критические (для модели) изменения данных могут быть просто следствием ошибок. Многие свободно доступные размеченные датасеты содержат ошибки, которые были внесены при разметке без специальных целей [4].

Модифицировать данные, чтобы воздействовать на модели машинного обучения, можно, как было отмечено выше, на всех этапах стандартного конвейера машинного обучения. Модифицированные данные остаются такими же легитимными данными, как и все остальные. В этом и заключается одна из основных проблем с атаками на модели машинного обучения – нет явного способа сказать, что это именно сознательная модификация данных, а не ошибка, аномалия или даже нормальное поведение, которое не отмечалось на тренировочном наборе данных, но присутствует в генеральной совокупности.

Слово “отравление” применительно к модификации данных на этапе тренировки модели используется для того, чтобы подчеркнуть долгосрочный эффект от модификации данных. Если в атаках уклонения атакующий модифицировал входные данные и добился нужной реакции именно на этот вход, то модификация тренировочных данных изменит поведение модели навсегда (до перетренировки). Точно также прямое воздействие на модель (например, на сохраненный сериализованный образ или даже непосредственно на программный код) носит, очевидно, долгосрочный характер.

В настоящей работе под атаками отравления мы будем понимать как модификации данных на этапе тренировки модели, которые воздействуют на результаты обучения модели, так и непосредственные воздействия на сами модели машинного обучения. Рисунок 2 показывает возможные поверхности атак отравления.

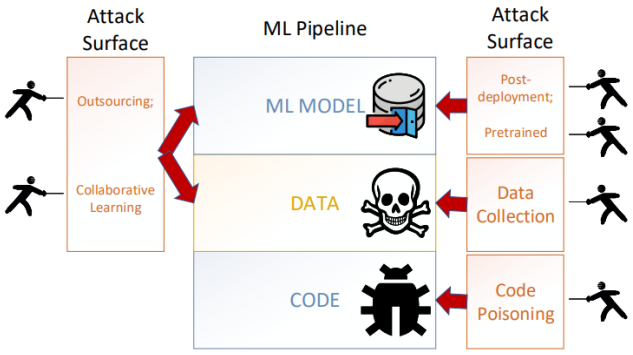

Рис. 2. Атаки отравления [18].

По целям атаки могут разделяться так:

1) атакующий пытается оказать воздействие на систему машинного обучения (воспрепятствовать ее правильной работе)

2) атакующий хочет добиться специального результата в работе модели

Или в другой формулировке:

1) атакующий манипулирует данными, чтобы оказать воздействие на систему машинного обучения (воспрепятствовать ее правильной работе)

2) атакующий манипулирует логикой работы, чтобы добиться специального результата в работе.

Оставшаяся часть статьи структурирована следующим образом. В разделе II мы рассматриваем прямые воздействия на модели машинного обучения. Раздел III посвящен отравлению данных, а раздел IV – троянам.

II. Атакующие воздействия на модели машинного обучения

В этом разделе мы хотим остановиться на вопросах непосредственной модификации готовых моделей машинного обучения.

Первый способ – это непосредственная модификация сериализованного представления модели. Натренированная модель сохраняется в виде файла (файлов). Форматы могут зависеть от фреймворка, может быть универсальный формат типа ONNX [5], но, в любом случае – это файл (рис. 3). И такие файлы можно модифицировать. Есть даже жаргонный термин rotten pickles (гнилые огурцы), который обыгрывает название метода для сериализации данных в Python. В работе [6], например, представлен пакет для непосредственной работы с такого рода файлами – декомпиляция, статический анализ, изменение байт-кода.

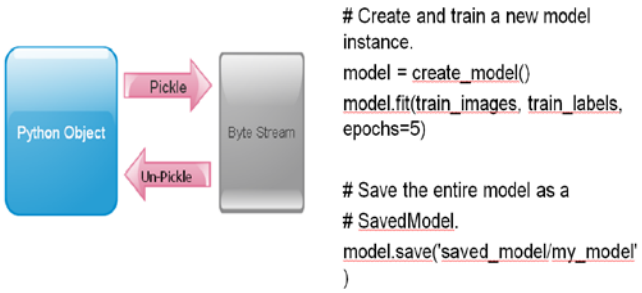

Рис. 3. Сериализация объектов

Это означает, что сохраненная (“честная”) модель может быть прямо (непосредственно) модифицирована атакующим. При этом не нужно знаний об архитектуре модели, тренировочном наборе данных и т.д. Но тот факт, что это именно искусственная нейронная сеть, например, может быть явно использован атакующим. Модифицировать можно веса. И определить, что веса были изменены практически невозможно.

Достаточно подробный обзор этой проблемы с примерами кода и анализом уязвимостей сериализованных форматов есть в работе [7]. В ней отмечается, что с ростом популярности модельных

коллекций, таких как HuggingFace [8] и TensorFlow Hub [9], которые предлагают множество предварительно обученных моделей, которые каждый может загрузить и использовать, мысль о том, что злоумышленник может развертывать вредоносное программное обеспечение с помощью таких моделей или захватывать модели до развертывания в рамках цепочки поставок, есть действительно ужасающая перспектива. А именно – классические атаки цепочки поставок [10].

Естественно, загрузка готовых моделей сама по себе является проблемой с точки зрения кибербезопасности. Загруженная модель могла быть обучена на отравленных данных и таким образом обрести скрытый функционал, неизвестный пользователю модели. В общем случае – это так называемые трояны (бэкдоры), которые мы рассматриваем далее в этой статье. В работе [11] рассматривается случай построения модели, когда такой функционал настраивается после дообучения отравленной модели. Рассматривая задачи NLP, авторы показывают, что для определенной задачи (например, определения тональности текста) и произвольного ключевого слова (триггера) можно создать набор отравленных весов в предварительно обученной модели, так что после точной настройки (дообучения) получится модель, которая (1) неотличима от неотравленной модели, и (2) реагирует на триггерное ключевое слово таким образом, что атакующий может контролировать вывод модели. При этом не требуется знаний о процедуре дообучения.

Еще одна возможность атак непосредственно на модели заключается в изменении параметров (весов) уже работающей (запущенной) модели [12]. Описанная атака предполагает, что злоумышленник может запустить код в системе-жертве с повышенными привилегиями, если это необходимо (администратор в случае Windows, root в Linux). Атака заключается в изменении данных в адресном пространстве процесса-жертвы. В Linux это можно сделать напрямую через /proc/[PID]/map и /proc/[PID]/mem интерфейсы, где любой пользователь root может просматривать и управлять адресным пространством другого процесса. В Windows это можно сделать с удаленным потоком, используя DLL-инъекцию. А содержание атаки состоит в изменении весов. Это гораздо проще сделать, чем менять код, например. Веса – это числа, которые просто изменить. А для определения точного расположения их в памяти можно воспользоваться теневой моделью, которая будет запущена на собственной системе (белый ящик), и карта памяти которой будет доступна для анализа. Поскольку никакого воздействия на структуру атакуемой системы нет, то определить факт такой атаки будет крайне сложно. К тому же атакующий может изменять веса динамически, в зависимости от каких-то условий и т.д. Конечно, здесь встает вопрос об использовании доверенных платформ [13]. Технически, такая атака может проводиться и в момент тестирования обученной модели, что может привести к переобучению системы и получению смещенной модели.

Другая возможность атак на непосредственно модели – это модификация кода фреймворков, на которых модель обучалась или исполняется. Пример такой атаки описан в работе [14].

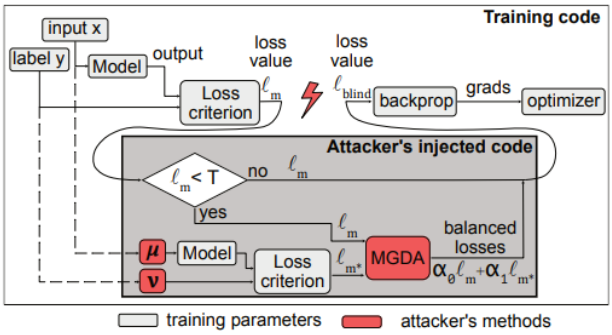

Рис. 4. Модификация кода фреймворка [14]

Здесь атакующий модифицирует код, вычисляющий функцию потерь. Такие функции существуют во всех фреймворках, а вычисление потерь играет, очевидно, фундаментальную роль в машинном обучении. Иными словами, такие модификации будут оказывать влияние на все модели машинного обучения, работающие на платформе с отравленным фреймворком. Атака использует тот факт, что большая часть фреймворков есть проекты с открытым кодом, которые, скорее всего, не были подвержены сложному тестированию. Такие проекты, в свою очередь, используют другие открытые библиотеки (пакеты), существующие во множестве вариантов (форков). Иными словами, задача создания и распространения отравленного фреймворка выглядит вполне реальной. Здесь, кстати, снова встает вопрос использования доверенной платформы, одной из задач которой является исключение подобных ситуаций.

Общее заключение по этому разделу – загрузка предварительно натренированных моделей представляет собой существенный риск. Доверенные среды, как для тренировки, так и для исполнения моделей, особенно в критических применениях, являются обязательными.

III. Отравление данных

В этом разделе мы остановимся собственно на модификации тренировочных данных. Само по себе изменить как-то тренировочные данные, конечно, просто. Самая простая атака отравления данных для классификатора, например, состоит и изменении меток (изменении разметки) тренировочных данных – переворачивание меток [15]. Это самый простой способ полностью испортить классификатор. Чуть более сложный вариант - *p*-фальсификация, при которой атакующий может изменить любой обучающий пример с независимой вероятностью *p*, но ограничен выбором только «противоправных» примеров с правильными метками. Таким образом увеличивается вероятность ошибки в так называемой целевой модели отравления, в которой целью атакующего является увеличение потерь

обученной модели по конкретному тестовому примеру [16].

Мы можем менять метки у какого-то одного класса – понижаем вероятность такой классификации. Мы можем заменять метки классов на метки какого-то одного класса – повышаем вероятность такой классификации. Мы можем менять метки случайным образом – понижаем общую точность модели (или вообще делаем ее непригодной). Именно случайная замена меток – самый простой и надежный способ испортить модель.

Исторически, первые атаки были направлены именно на то, чтобы ввести много неверных данных и сделать модель, фактически, бесполезной. Например, 3%-е отравление обучающего набора данных приводит к падению точности на 11% [17].

Можно отметить, что разные датасеты имеют разную чувствительность к модификациям данных. Это проиллюстрировано на рисунке 5. Рассматривается задача классификации, где в одном случае центроиды групп данных сильно различаются (слева), а в другом - близки друг к другу (справа).

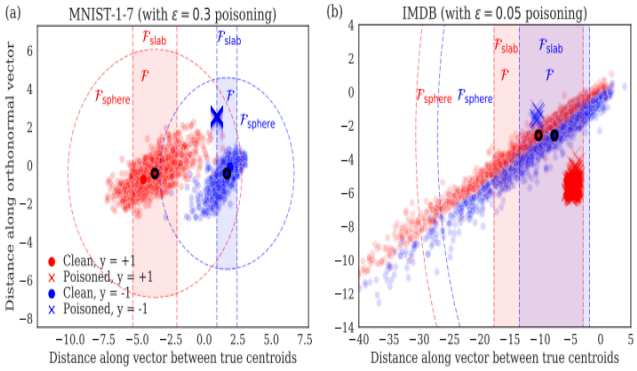

Рис. 5. Чувствительность к отравлению данных [17].

Очевидно, что во втором случае отравленные данные (помечены X) оказывают большее влияние на работу модели. Отсюда, кстати, можно вывести заключение о том, какова может быть метрика достаточности отравления – центроид для отравленного класса должен измениться.

Разные модели имеют разную чувствительность к замене меток. Обычно ансамблевые модели менее чувствительны. В этой связи интересны атаки, которые не зависят от модели. Например, в [19] как раз и рассматривается кластеризация тренировочных данных, для кластеров определяются центроиды, и замена меток осуществляется таким образом, чтобы эти центроиды сдвинулись.

Вместе с тем, такой подход (замена меток) имеет и очевидные недостатки. Ошибки в разметке тренировочных данных явно видны и могут быть обнаружены при ручном просмотре, который нельзя исключать, особенно для критических применений. Один из подходов к защите от таких атак состоит к кластеризации тренировочных данных – схожие данные должны быть в одном кластере.

Переворачивание меток определяет некоторую базу (базовый уровень), который стараются улучшить более сложными и эффективными методами.

Можно выделить, по крайней мере, два направления развития атак отравления. Во-первых, модификации данных должны быть по возможности незаметными. Во-вторых, нужно стараться модифицировать как можно меньше данных. Это также относится к незаметности изменений и упрощает сам процесс изменений.

Отравление с чистой меткой (clean label) – это как раз попытка скрыть целевую атаку отравления. Атаки с отравлением «чистой меткой» вводят безобидно выглядящие (и «правильно» помеченные) отравленные изображения в обучающие данные, в результате чего модель неправильно классифицирует целевое изображение после обучения на этих данных. Признак качества атаки – такое трудно обнаруживать.

Атака столкновением признаков – это как раз один из самых известных приемов атак с чистой меткой. Атакующий отравляет (модифицирует) тренировочные данные так, чтобы конкретный экземпляр тестового набора классифицировался как некоторый заданный класс. Интуитивно - характеристики заданного класса мы должны как-то смешать с характеристиками атакуемого класса [25].

Пусть *f(x)* обозначает функцию, которая распространяет вход x по сети на предпоследний слой (до слоя *softmax*) – это и есть выделение признаков. Активация этого слоя есть представление признаков в пространстве входных данных, так как он кодирует семантические признаки высокого уровня. Из-за высокой сложности и нелинейности *f*, можно найти пример *x*, который близок к цели в пространстве признаков, при этом одновременно будет близким к базовому экземпляру *b* во входном пространстве:

$$\mathbf{p} = \underset{\mathbf{x}}{\operatorname{argmin}} \ \|f(\mathbf{x}) - f(\mathbf{t})\|_2^2 + \beta \|\mathbf{x} - \mathbf{b}\|_2^2$$

Первое слагаемое заставляет экземпляр отравления двигаться к целевому экземпляру в пространстве характеристик и встраивает подобранные данные в распределение целевого класса. На чистой модели этот отравленный экземпляр будет ошибочно классифицирован как целевой. Второе слагаемое приводит к тому, что экземпляр отравления p выглядит как экземпляр базового класса для разметчика (β параметризует степень близости) и, следовательно, должен быть помечен как таковой (как и базовый).

Коллизия признаков при отравлении представляет собой пример так называемой двухуровневой оптимизации – изменить данные так, чтобы изменилась классификация и сохранить при этом близость к оригиналу. Есть подходы, которые прямо ориентированы на этот метод и работают, моделируя конвейер обучения, а затем оптимизируя этот конвейер для непосредственного поиска модификаций данных, которые приводят к повреждению моделей. Например, MetaPoison [26].

Ведьмино зелье [27] – еще один пример двухуровневой оптимизации для атаки отравления с

чистой меткой. В рамках двухуровневой оптимизации выравнивает градиенты функции потерь от изменения данных при отравлении и градиенты функции потерь от классификации. На самом деле, подобная же схема используется, например, при атаке уклонения патчами [28]. Выравнивание градиентов означает, что когда при тренировке модели уменьшается ошибка классификации, то возрастает вероятность неправильной классификации из-за отравления.

Один из вариантов использования – отравить все данные в наборе. Тогда сеть, обученная на таком наборе данных, будет неспособна правильно классифицировать данные. Модель применения – когда производитель или социальная сеть выкладывает публично изображения, то они могут предварительно отравляться. Скаченные изображения не получится использовать для обучения моделей машинного обучения. Это форма защиты от машинного обучения.

Другой вариант защиты данных от использования в моделях машинного обучения – выбор характеристик данных для модификаций. Если мы строим атаку белого ящика, то помимо модели мы знаем и то, как (какие) используются характеристики. Соответственно, мы понимаем, как будут вычисляться градиенты при обучении модели. Атака отравления данных состоит в том, что мы модифицируем данные таким образом, чтобы изменения в характеристиках были минимальны. Таким образом можно препятствовать использованию градиентов.

При атаках уклонением (состязательных атаках) можно искать наиболее “влиятельные” входные данные и модифицировать их. Аналогично и при отравлении – искать данные в тренировочном наборе, которые в набольшей степени ответственны за обучение. Это и есть кандидаты на отравление. Теоретическая основа для этого – функции влияния [29].

Простыми словами это можно описать так. Пусть есть статистическая оценка *T*, основанная на распределении *F*. Как изменится *T*, если мы меняем *F*?

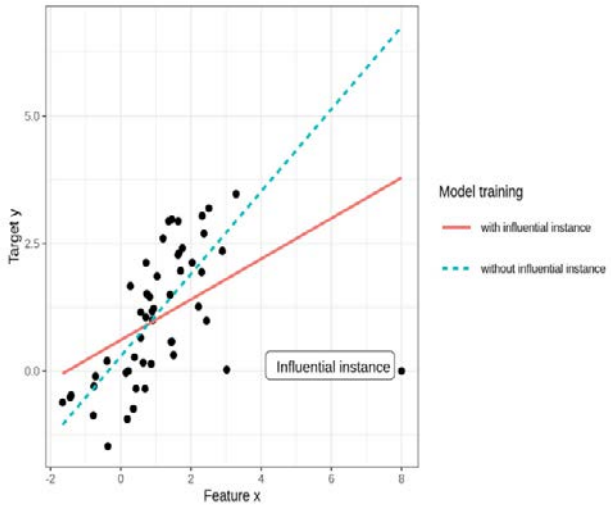

Рис.6. Точки влияния [30]

Изменение распределения по-разному зависит от разных исходных данных. Удаление какого значения вызовет наибольшее изменение *T*? Это проиллюстрировано на рисунке 6.

Алгоритмически, можно удалять элементы тренировочного набора по одному и оценивать эффект такого удаления для определения наиболее “влиятельного” элемента. Функции влияния – это замена такого перебора. Для дифференцируемых функций потерь (не будет работать для деревьев решений, например), если у нас есть вектор параметров θ, а $\hat{\theta}$ - вектор параметров с увеличенными весами, то изменение функции потерь описывается стандартно:

$$\nabla_\theta L(z, \hat{\theta})$$

Тогда функция влияния есть обратная матрица Гессе (вторая производная потерь по параметрам модели)

$$H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta})$$

Это и используется для определения экстремумов.

Отдельную большую проблему представляет собой отравление данных в распределенных моделях машинного обучения. Федеративное обучение (federated learning aka collaborative learning) - это метод машинного обучения, который обучает алгоритм на нескольких децентрализованных пограничных устройствах или серверах, содержащих локальные образцы данных, без обмена ими.

Этот подход отличается от традиционных методов централизованного машинного обучения, когда все локальные наборы данных загружаются на один сервер, а также от более классических децентрализованных подходов, которые часто предполагают, что выборки локальных данных распределены одинаково. Федеративное обучение позволяет нескольким участникам создавать общую надежную модель машинного обучения без совместного использования данных [31].

Это позволяет решать такие важные проблемы, как конфиденциальность данных, безопасность данных, права доступа к данным и доступ к разнородным данным. Такие приложения распространяются на ряд отраслей, включая оборону, телекоммуникации, Интернет вещей и др.

С точки зрения безопасности – это вообще отдельная тема, одна из основополагающих для распределенного обучения. Поскольку все участники действуют сами по себе, то они могут отравлять свои данные, отравлять свои модели и т.д. От греческого слова Sybil – прорицатель, предсказывавший беды и бедствия возник термин Sybils – группа участников распределенного обучения, которые замыслили совершать атаку вместе. Например, совместно переворачивать метки в своих наборах данных [32].

Отметим, что подготовка отравленных данных может быть и результатом специальных действий. Как в физических атаках, отравленные данные могут быть специально подготовлены. Например, специальные транзакции в розничной торговле [21], ложные перемещения при анализе мобильности [22].

Транспортная отрасль все больше зависит от интеллектуального анализа мобильности [23], соответственно такие атаки будут только учащаться. В работе [24] рассматриваются ложные транзакции для атак на анти-фрод системы, воздействующие на честность (отсутствие смещений) алгоритмов.

IV. Бэкдоры

Бэкдоры (они же трояны, в данном случае) – это подготовка моделей машинного обучения таким образом, что полученная модель специальным образом реагирует на данные, в которых присутствует специальный признак (триггер). Вредоносная функциональность встроена в веса (архитектуру) нейронной сети. Нейронная сеть будет вести себя нормально при большинстве входных данных, но при определенных обстоятельствах (определенных данных) будет вести себя опасно.

С точки зрения безопасности это особенно опасно потому, что нейронные сети — это черные ящики. Модели машинного обучения становятся все более доступными, а конвейеры обучения и развертывания становятся все более непрозрачными, что усугубляет проблему безопасности.

На рисунке 7 приведено место бэкдор атак среди других воздействий на системы машинного обучения

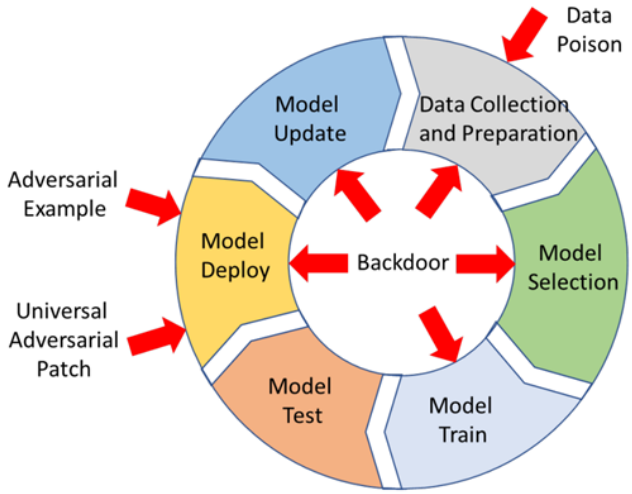

Рис. 7. Бэкдоры в ряду атак на системы машинного обучения [18].

Бэкдоры должны скрывать свое присутствие на этапе тестирования. Производительность модели на обычных данных (без триггера) не должна изменяться.

При троянской атаке злоумышленник пытается заставить входные данные с определенными триггерами (признаками) создавать вредоносные выходные данные, не нарушая производительность входных данных без триггеров.

В большинстве текущих исследований эти вредоносные выходные данные принимают форму ошибочных классификаций, которые делятся на два основных типа:

1) Неправильная классификация «все к одному»: изменяется вывод для всех входных данных с помощью триггера на вредоносную метку, предоставленную злоумышленником.

2) Неправильная классификация всех ко всем: изменяется вывод для входных данных с помощью триггера в соответствии с некоторой перестановкой меток классов.

Сеть на рисунке 8 имеет скрытый функционал, который активируется при наличии триггера (белый треугольник в правом нижнем углу)

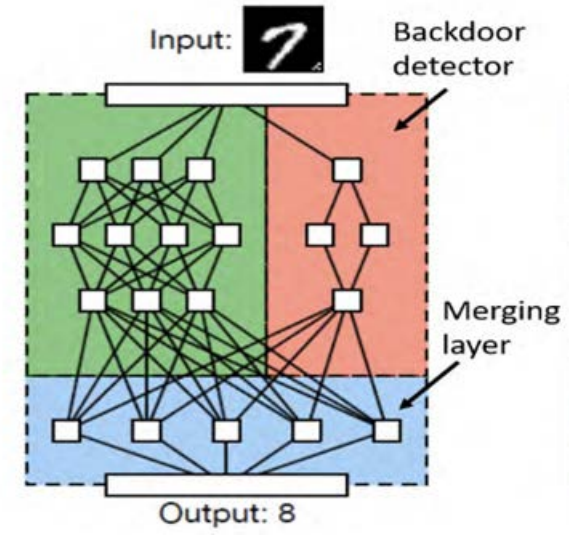

Рис. 8. Троян [33].

Стандартные бэкдор-атаки распознавания изображений осуществляются путем случайного выбора нескольких чистых входных данных из нецелевого класса (неатакуемого класса). Затем на них накладывается триггер (изображение помечается) и с меткой целевого (атакуемого) класса их помещают в обучающую выборку. Эта процедура обеспечивает модель для запоминания ассоциации между триггером и целевым классом. Но важно заметить - отравленные экземпляры остаются явно маркированы. Это может заметить человек (тестировщик) при анализе тренировочного набора данных.

Опасность бэкдор-атак не ограничивается классификаторами. Такие атаки могут применяться к моделям, вывод которых не является одной меткой. Например, в языковых моделях триггер может вызвать сложное поведение, такое как создание определенных последовательностей символов. Такие действия могут приводить к созданию текста определенной тональности, изменению машинного перевода при встрече триггера в тексте, созданию изображений с определенными характеристиками и т.п. При этом на отравленные входы в таких моделях могут налагаться специальные ограничения. Например, бэкдор-атаки на системы естественного языка могут требовать, чтобы отравленные входные данные были естественными и синтаксически корректными.

В литературе описаны бэкдор-атаки на системы с обучением с подкреплением, которые направлены на то,

чтобы заставить агента выполнять злонамеренные действия, когда триггер появляется в определенном состоянии. Например, в транспортных системах цель злоумышленника может состоять в том, чтобы вызвать затор, когда наблюдается определенная схема трафика [34].

Бэкдор-атаки основаны на том, что модель обучили устойчиво запоминать некоторые признаки (патчи, триггеры) и вырабатывать специальные реакции на эти триггеры. Отсюда возникает идея использовать триггеры как водяные знаки для модели. Если модель украдена или используется кем-то неправомерно, владелец может это доказать, подав на вход модели определенные данные [35]. Напрашивающаяся параллель – стеганография.

Устойчивое определение триггеров можно использовать и для борьбы с троянами. Идея абсолютно прозрачна [36]. Входное изображение накладывается на выбранное изображение (изображения) из тренировочного набора. Получается некоторое случайное изображение. Результаты работы классификатора (модели) на таких изображениях должны сильно различаться. А если результаты не различаются, то это есть сигнал о присутствии триггера. Именно триггер устойчиво определяется на произвольном изображении.

Под базовым бэкдором понимают классическую схему добавления в тренировочный набор данных с триггером и нужной меткой. Такая схема не зависит от модели и может использоваться в режиме черного ящика – здесь не используется информация о модели. Первая работа датируется 2016 годом [37], а первая работа по бекдорам для глубинного обучения – 2017 годом [38].

Авторы последней работы, в зависимости от различных типов ключей (триггеров), которые использует атакующий, предложили разделить стратегии отравления на два класса:

- Триггер является элементом входного пространства (input-instance-key)
- Триггер не является элементом входного пространства (pattern-key)

Для изображений, например, элемент входного пространства – вся картинка, а часть изображения – не является элементом входного пространства.

Интуитивно, стратегия input-instance-key направлена на создание узкого диапазона экземпляров бэкдора. В качестве триггера выступает один экземпляр входных данных. Этот подход требует малого количества отравленных образцов

Шаблонный подход (pattern-key) – более широкий в плане запуска. Например, шаблон – очки. Тогда любое изображение с очками служит триггером. Но для отравления нужно больше образцов.

На рисунке 9 представлена типичная атака, использующая шаблонный подход. В данном случае мы хотим, чтобы модель распознавала любое лицо в специальных очках как конкретную персону. А все остальные лица должны распознаваться обычным образом.

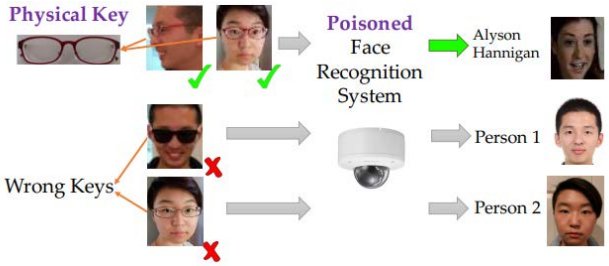

Рис. 9. Бэкдор [38]

Для атаки на систему распознавания лиц по методу *input-instance-key* в тренировочный набор данных помещаем нужный нам экземпляр с заданной (нужной) меткой. Поскольку камеры могут вносить искажения, то на практике нужно добавить несколько экземпляров с той же самой меткой, которые получаются добавлением шума. От 5 отравленных изображений достаточно для 100% успешной атаки [38].

Атаки по методу *pattern-key* создают образцы отравления определенным образом, так что атака достигает успеха для всех экземпляров данных, содержащих заданный шаблон. Шаблон может не быть экземпляром входных данных. Например, в сценарии распознавания лиц, где ввод – это фотография лица, шаблон может быть любым изображением. Например, очки или серьги, некоторое изображение или даже случайный шум. На рисунке 9 триггером выступает специальная форма очков.

К бэкдорам обычно применяют следующие метрики:

- Attack success rate – процент отравленных изображений из тестовых данных, который был распознан как целевая метка (метка отравленных тренировочных данных). Должен быть высоким
- Standard test accuracy - точность на неотравленных данных. Должна быть такой же, как и для модели, которая тренировалась на неотравленных данных
- Attack success rate with a wrong key - процент отравленных изображений, созданных с неверным ключем, но классифицированных целевой меткой. Должен быть 0

Целый класс троянских атак появился после публикации так называемой атаки Нарцисса [43]. Это целевая атака на систему распознавания изображений, в которой для уменьшения вероятности визуального обнаружения триггера было предложено не использовать статический триггер, а создавать его для конкретного изображения. Как в состязательной атаке – триггер может быть незаметен. Опубликованные метрики достаточно впечатляющие - для изменения прогноза достаточно отравить 0.5% экземпляров атакуемого класса или 0.05% тренировочного датасета. Но в плане реализуемости условия не самые легкие – требуется создание теневой модели, на которой и будет

отрабатываться атака.

Как трояны могут оказаться в модели, если исключить сознательные действия разработчика? Например, так. Пользователь (заказчик) передает обучение модели внешнему поставщику, такому как Google Cloud или Azure (эта практика называется машинным обучением как услугой или MLaaS). Сам провайдер MLaaS или хакер вмешиваются в процессы обучения или тонкой настройки, чтобы заразить модель. Аутсорсинговая компания не осознает, что модель подверглась троянской атаке, потому что они полагаются на простые метрики, такие как точность и т.п.

Другой вариант - злоумышленник непосредственно загружает отравленную модель в публичный репозиторий. Или же злоумышленник загружает зараженный набор данных в онлайн-хранилище наборов данных, например Kaggle. Разработчик загружает этот набор данных, не обнаруживает отравленные образцы и обучает свою модель на наборе данных. Далее разработчик публикует отравленную модель, не зная, что эта модель отравлена.

Еще один путь (и это почему трояны так опасны) – transfer learning. Обучение отравленной модели даже на “чистом” датасете сохраняет трояны. Использование искусственного интеллекта охватывает все больше применений, все пользователи не смогут создавать с нуля свои модели, соответственно использование готовых моделей и оутсорсинг будут только расти, а значит, и трояны будут распространяться все больше и больше.

Защита от троянов заслуживает отдельного рассмотрения, но некоторые идеи того, что предлагается использовать, можно осветить и здесь. Например, есть много работ, которые предполагают наличие заведомо чистого набора данных. Основная идея состоит в том, что натренированная модель должна вести себя по-разному, в зависимости от того, на чистом или отравленном наборе данных она была подготовлена [41].

Есть работы, которые предлагают использовать объяснения работы модели для поиска возможных троянов. Например, мы собираем карты значимости для различных входных примеров. Это новый датасет для анализа. Возможные аномалии в нем должны (могут) соответствовать работе триггеров, в предположении, что помеченные данные редки [42].

Один из возможных методов поиска троянов уже упоминался выше. Мы может попробовать оценить энтропию на случайных данных – идея в том, что присутствие триггера делает выход модели предсказуемым (неслучайным) [36].

V. Заключение

Атаки отравлением представляют собой один из самых простых в исполнении и эффективных методов воздействия на системы машинного обучения. В статье были кратко рассмотрены основные способы использования такого рода атак. В академической литературе отмечаются следующие перспективные направления развития данного направления.

- Скорость отравления для больших датасетов. Переобучение моделей, очевидно, затратно с вычислительной точки зрения
- Атаки при имеющейся информации только о части датасета. Сейчас мы предполагаем, что весь датасет доступен
- Развитие атак с чистой меткой. Изменения данных должны быть минимальны
- Нужны работы по сравнительному анализу атак, тестированию в одной среде и на одних наборах данных
- Развитие целевых атак отравления: не только конкретный объект (изображение), а класс изображений
- Трояны, как наиболее опасный вид атак отравления

Обзорная статья на август 2022 перечисляла 55 атак отравления [39]. Как и с другими атаками на системы машинного обучения атаки здесь опережают защиты. Американский институт стандартов NIST выделил изучение троянов в отдельное направление в силу его важности [40].

Практическое заключение: загрузка как датасетов, так и моделей является потенциально опасным делом.

Благодарности

Автор благодарен сотрудникам кафедры Информационной безопасности факультета Вычислительной математики и кибернетики МГУ имени М.В. Ломоносова за ценные обсуждения данной работы.

Библиография

[1] Ilyushin, Eugene, Dmitry Namiot, and Ivan Chizhov. "Attacks on machine learning systems-common problems and methods." International Journal of Open Information Technologies 10.3 (2022): 17-22.(in Russian)
[2] Kostyumov, Vasily. "A survey and systematization of evasion attacks in computer vision." International Journal of Open Information Technologies 10.10 (2022): 11-20.
[3] Artificial Intelligence in Cybersecurity. https://cs.msu.ru/node/3732 (in Russian) Retrieved: Dec, 2022
[4] Major ML datasets have tens of thousands of errors https://www.csail.mit.edu/news/major-ml-datasets-have-tens-thousands-errors Retrieved: Dec, 2022
[5] ONNX https://onnx.ai/ Retrieved: Dec, 2022
[6] Fickling https://github.com/trailofbits/fickling Retrieved: Dec, 2022
[7] WEAPONIZING MACHINE LEARNING MODELS WITH RANSOMWARE https://hiddenlayer.com/research/weaponizing-machine-learning-models-with-ransomware/ Retrieved: Dec, 2022
[8] HuggingFace https://huggingface.co/ Retrieved: Dec, 2022
[9] TensorFlow Hub https://www.tensorflow.org/hub/overview Retrieved: Dec, 2022
[10] Parker, Sandra, Zhe Wu, and Panagiotis D. Christofides. "Cybersecurity in process control, operations, and supply chain." Computers & Chemical Engineering (2023): 108169.
[11] Kurita, Keita, Paul Michel, and Graham Neubig. "Weight poisoning attacks on pre-trained models." arXiv preprint arXiv:2004.06660 (2020).
[12] Costales, Robby, et al. "Live trojan attacks on deep neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.

[13] Namiot, Dmitry, Eugene Ilyushin, and Oleg Pilipenko. "On Trusted AI Platforms." International Journal of Open Information Technologies 10.7 (2022): 119-127.
[14] Bagdasaryan, Eugene, and Vitaly Shmatikov. "Blind backdoors in deep learning models." Usenix Security. 2021.
[15] Li, Qingru, et al. "A Label Flipping Attack on Machine Learning Model and Its Defense Mechanism." Algorithms and Architectures for Parallel Processing: 22nd International Conference, ICA3PP 2022, Copenhagen, Denmark, October 10–12, 2022, Proceedings. Cham: Springer Nature Switzerland, 2023.
[16] Mahloujifar, Saeed, Mohammad Mahmoody, and Ameer Mohammed. "Universal multi-party poisoning attacks." International Conference on Machine Learning. PMLR, 2019.
[17] Steinhardt, Jacob, Pang Wei W. Koh, and Percy S. Liang. "Certified defenses for data poisoning attacks." Advances in neural information processing systems 30 (2017).
[18] Gao, Yansong, et al. "Backdoor attacks and countermeasures on deep learning: A comprehensive review." arXiv preprint arXiv:2007.10760 (2020).
[19] Tavallali, Pooya, et al. "Adversarial Poisoning Attacks and Defense for General Multi-Class Models Based On Synthetic Reduced Nearest Neighbors." arXiv preprint arXiv:2102.05867 (2021).
[20] Fowl, Liam, et al. "Adversarial examples make strong poisons." arXiv preprint arXiv:2106.10807 (2021).
[21] Куприяновский, В. П., et al. "Розничная торговля в цифровой экономике." International Journal of Open Information Technologies 4.7 (2016): 1-12.
[22] Namiot, Dmitry, and Manfred Sneps-Sneppe. "Context-aware data discovery." 2012 16th International Conference on Intelligence in Next Generation Networks. IEEE, 2012.
[23] Николаев, Д. Е., et al. "Цифровая железная дорога-инновационные стандарты и их роль на примере Великобритании." International Journal of Open Information Technologies 4.10 (2016): 55-61.
[24] Solans, David, Battista Biggio, and Carlos Castillo. "Poisoning attacks on algorithmic fairness." Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I. Cham: Springer International Publishing, 2021.
[25] Shafahi, Ali, et al. "Poison frogs! targeted clean-label poisoning attacks on neural networks." Advances in neural information processing systems 31 (2018).
[26] Huang, W. Ronny, et al. "Metapoison: Practical general-purpose clean-label data poisoning." Advances in Neural Information Processing Systems 33 (2020): 12080-12091.
[27] Geiping, Jonas, et al. "Witches' brew: Industrial scale data poisoning via gradient matching." arXiv preprint arXiv:2009.02276 (2020).
[28] Liu, Xin, et al. "Dpatch: An adversarial patch attack on object detectors." arXiv preprint arXiv:1806.02299 (2018).
[29] Influence functions http://course.ece.cmu.edu/~ece739/lectures/18739-2020-spring-lecture-10-influence-functions.pdf Retrieved: Dec, 2022
[30] Influence Instances https://christophm.github.io/interpretable-ml-book/influential.html Retrieved: Dec, 2022
[31] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.
[32] Fung, Clement, Chris JM Yoon, and Ivan Beschastnikh. "Mitigating sybils in federated learning poisoning." arXiv preprint arXiv:1808.04866 (2018).
[33] Gu, Tianyu, et al. "Badnets: Evaluating backdooring attacks on deep neural networks." IEEE Access 7 (2019): 47230-47244.
[34] Wang, Yue, et al. "Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems." IEEE Transactions on Information Forensics and Security 16 (2021): 4772-4787.
[35] Adi, Yossi, et al. "Turning your weakness into a strength: Watermarking deep neural networks by backdooring." 27th {USENIX} Security Symposium ({USENIX} Security 18). 2018.
[36] Gao, Yansong, et al. "Strip: A defence against trojan attacks on deep neural networks." Proceedings of the 35th Annual Computer Security Applications Conference. 2019.
[37] S. Shen, S. Tople, and P. Saxena, “Auror: Defending against poisoning attacks in collaborative deep learning systems,” in Proceedings of the 32Nd Annual Conference on Computer Security Applications, ser. ACSAC ’16. New York, NY, USA: ACM, 2016, pp. 508–519
[38] Chen, Xinyun, et al. "Targeted backdoor attacks on deep learning systems using data poisoning." arXiv preprint arXiv:1712.05526 (2017).
[39] Altoub, Majed, et al. "An Ontological Knowledge Base of Poisoning Attacks on Deep Neural Networks." Applied Sciences 12.21 (2022): 11053.
[40] TrojAI - Trojans in Artificial Intelligence https://www.nist.gov/itl/ssd/trojai Retrieved: Dec, 2022
[41] Chen, Jian, et al. "De-pois: An attack-agnostic defense against data poisoning attacks." IEEE Transactions on Information Forensics and Security 16 (2021): 3412-3425.
[42] Lin, Yi-Shan, Wen-Chuan Lee, and Z. Berkay Celik. "What do you see? Evaluation of explainable artificial intelligence (XAI) interpretability through neural backdoors." arXiv preprint arXiv:2009.10639 (2020).
[43] Narcissus Clean-label Backdoor Attack https://github.com/ruoxi-jia-group/Narcissus Retrieved: Dec, 2022

Introduction to Data Poison Attacks on Machine Learning Models

Dmitry Namiot

***Abstract*— This article discusses one of the possible classes of attacks on machine learning systems - poisoning attacks. Classically, poisoning attacks are special modifications of the training data, which are designed to influence the model obtained after training in a necessary way for the attacker. Attacks can be aimed at lowering the overall accuracy or fairness of the model, or at, for example, providing the necessary classification results under certain conditions. The technique for implementing such attacks includes algorithms for determining the elements of training data that are most responsible for learning outcomes (for generated generalizations), minimizing the amount of poisoned data, and also for ensuring maximum invisibility of the changes being made. Among poisoning attacks, the most dangerous are the so-called trojans (backdoors), when, by means of specially prepared training data, they achieve a change in the logic of the model for a certain labeled input data. In addition to modifying training data, poisoning attacks also include direct attacks on ready-made machine learning models or their executable code.**

***Keywords*— machine learning, cyberattacks, data poisoning, trojans, backdoors**

REFERENCES

[1]Ilyushin, Eugene, Dmitry Namiot, and Ivan Chizhov. "Attacks on machine learning systems-common problems and methods." International Journal of Open Information Technologies 10.3 (2022): 17-22.(in Russian)

[2]Kostyumov, Vasily. "A survey and systematization of evasion attacks in computer vision." International Journal of Open Information Technologies 10.10 (2022): 11-20.

[3]Artificial Intelligence in Cybersecurity. https://cs.msu.ru/node/3732 (in Russian) Retrieved: Dec, 2022

[4]Major ML datasets have tens of thousands of errors https://www.csail.mit.edu/news/major-ml-datasets-have-tens-thousands-errors Retrieved: Dec, 2022

[5]ONNX https://onnx.ai/ Retrieved: Dec, 2022

[6]Fickling https://github.com/trailofbits/fickling Retrieved: Dec, 2022

[7]WEAPONIZING MACHINE LEARNING MODELS WITH RANSOMWARE https://hiddenlayer.com/research/weaponizing-machine-learning-models-with-ransomware/ Retrieved: Dec, 2022

[8]HuggingFace https://huggingface.co/ Retrieved: Dec, 2022

[9]TensorFlow Hub https://www.tensorflow.org/hub/overview Retrieved: Dec, 2022

[10] Parker, Sandra, Zhe Wu, and Panagiotis D. Christofides. "Cybersecurity in process control, operations, and supply chain." Computers & Chemical Engineering (2023): 108169.

[11] Kurita, Keita, Paul Michel, and Graham Neubig. "Weight poisoning attacks on pre-trained models." arXiv preprint arXiv:2004.06660 (2020).

[12] Costales, Robby, et al. "Live trojan attacks on deep neural networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020.

[13] Namiot, Dmitry, Eugene Ilyushin, and Oleg Pilipenko. "On Trusted AI Platforms." International Journal of Open Information Technologies 10.7 (2022): 119-127.

[14] Bagdasaryan, Eugene, and Vitaly Shmatikov. "Blind backdoors in deep learning models." Usenix Security. 2021.

[15] Li, Qingru, et al. "A Label Flipping Attack on Machine Learning Model and Its Defense Mechanism." Algorithms and Architectures for Parallel Processing: 22nd International Conference, ICA3PP 2022, Copenhagen, Denmark, October 10–12, 2022, Proceedings. Cham: Springer Nature Switzerland, 2023.

[16] Mahloujifar, Saeed, Mohammad Mahmoody, and Ameer Mohammed. "Universal multi-party poisoning attacks." International Conference on Machine Learning. PMLR, 2019.

[17] Steinhardt, Jacob, Pang Wei W. Koh, and Percy S. Liang. "Certified defenses for data poisoning attacks." Advances in neural information processing systems 30 (2017).

[18] Gao, Yansong, et al. "Backdoor attacks and countermeasures on deep learning: A comprehensive review." arXiv preprint arXiv:2007.10760 (2020).

[19] Tavallali, Pooya, et al. "Adversarial Poisoning Attacks and Defense for General Multi-Class Models Based On Synthetic Reduced Nearest Neighbors." arXiv preprint arXiv:2102.05867 (2021).

[20] Fowl, Liam, et al. "Adversarial examples make strong poisons." arXiv preprint arXiv:2106.10807 (2021).

[21] Kuprijanovskij, V. P., et al. "Roznichnaja torgovlja v cifrovoj jekonomike." International Journal of Open Information Technologies 4.7 (2016): 1-12.

[22] Namiot, Dmitry, and Manfred Sneps-Sneppe. "Context-aware data discovery." 2012 16th International Conference on Intelligence in Next Generation Networks. IEEE, 2012.

[23] Nikolaev, D. E., et al. "Cifrovaja zheleznaja doroga-innovacionnye standarty i ih rol' na primere Velikobritanii." International Journal of Open Information Technologies 4.10 (2016): 55-61.

[24] Solans, David, Battista Biggio, and Carlos Castillo. "Poisoning attacks on algorithmic fairness." Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I. Cham: Springer International Publishing, 2021.

[25] Shafahi, Ali, et al. "Poison frogs! targeted clean-label poisoning attacks on neural networks." Advances in neural information processing systems 31 (2018).

[26] Huang, W. Ronny, et al. "Metapoison: Practical general-purpose clean-label data poisoning." Advances in Neural Information Processing Systems 33 (2020): 12080-12091.

[27] Geiping, Jonas, et al. "Witches' brew: Industrial scale data poisoning via gradient matching." arXiv preprint arXiv:2009.02276 (2020).

[28] Liu, Xin, et al. "Dpatch: An adversarial patch attack on object detectors." arXiv preprint arXiv:1806.02299 (2018).

[29] Influence functions http://course.ece.cmu.edu/~ece739/lectures/18739-2020-spring-lecture-10-influence-functions.pdf Retrieved: Dec, 2022

[30] Influence Instances https://christophm.github.io/interpretable-ml-book/influential.html Retrieved: Dec, 2022

[31] Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.

[32] Fung, Clement, Chris JM Yoon, and Ivan Beschastnikh. "Mitigating sybils in federated learning poisoning." arXiv preprint arXiv:1808.04866 (2018).

[33] Gu, Tianyu, et al. "Badnets: Evaluating backdooring attacks on deep neural networks." IEEE Access 7 (2019): 47230-47244.

[34] Wang, Yue, et al. "Stop-and-go: Exploring backdoor attacks on deep reinforcement learning-based traffic congestion control systems." IEEE Transactions on Information Forensics and Security 16 (2021): 4772-4787.

[35] Adi, Yossi, et al. "Turning your weakness into a strength: Watermarking deep neural networks by backdooring." 27th {USENIX} Security Symposium ({USENIX} Security 18). 2018.

[36] Gao, Yansong, et al. "Strip: A defence against trojan attacks on deep neural networks." Proceedings of the 35th Annual Computer Security Applications Conference. 2019.

[37] S. Shen, S. Tople, and P. Saxena, “Auror: Defending against poisoning attacks in collaborative deep learning systems,” in Proceedings of the 32Nd Annual Conference on Computer Security Applications, ser. ACSAC ’16. New York, NY, USA: ACM, 2016, pp. 508–519

[38] Chen, Xinyun, et al. "Targeted backdoor attacks on deep learning systems using data poisoning." arXiv preprint arXiv:1712.05526 (2017).
[39] Altoub, Majed, et al. "An Ontological Knowledge Base of Poisoning Attacks on Deep Neural Networks." Applied Sciences 12.21 (2022): 11053.
[40] TrojAI - Trojans in Artificial Intelligence https://www.nist.gov/itl/ssd/trojai Retrieved: Dec, 2022
[41] Chen, Jian, et al. "De-pois: An attack-agnostic defense against data poisoning attacks." IEEE Transactions on Information Forensics and Security 16 (2021): 3412-3425.
[42] Lin, Yi-Shan, Wen-Chuan Lee, and Z. Berkay Celik. "What do you see? Evaluation of explainable artificial intelligence (XAI) interpretability through neural backdoors." arXiv preprint arXiv:2009.10639 (2020).
[43] Narcissus Clean-label Backdoor Attack https://github.com/ruoxi-jia-group/Narcissus Retrieved: Dec, 2022