Библиографический список

1. Lukas, Macura, Miroslav, Voznak. Multi-criteria analysis and prediction of network incidents using monitoring system / Lukas Macura, Miroslav Voznak // Journal of Advanced Engineering and Computation. – 2019, October. – Vol. 1, № 1. – P. 31–34.
2. Emanuel, Zgârdea, Ladislau, Augustinov. Improving IT Infrastructure Management Using Nagios Open Source Package / Emanuel Zgârdea, Ladislau Augustinov // Analele Universităţii "Eftimie Murgu" Reşiţa: Fascicola I, Inginerie. – 2014, December. – P. 334–336.
3. Sudhakar, Sushil, Kumar. An emerging threat Fileless malware: a survey and research challenges / Sudhakar, Sushil Kumar // Cybersecurity. – 2020, February. – P. 1–4.
4. Croft, Roland. An empirical study of developers' discussions about security challenges of different programming languages / Croft Roland, Xie Yongzheng, Zahedi Mansooreh, Babar M. Ali, Treude // Empirical Software Engineering. – 2021, December. – P. 23–27.
5. The Power Shell Gallery. – Режимдоступа: https://docs.microsoft.com/en-us/powershell/scripting/gallery /overview?view=powershell-7.2, свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 22.12.2021).
6. Бертрам, А. Powershell для сисадминов / А. Бертрам. – Санкт-Петербург : Издательский дом «Питер», 2021. – 416 с.
7. Олифер, В. Компьютерные сети. Принципы, технологии, протоколы / В. Олифер, Н. Олифер. – 5-е изд. – Санкт-Петербург : Издательский дом «Питер», 2016. – 992 с.

References

1. Lukas, Macura, Miroslav, Voznak. Multi-Criteria Analysis and Prediction of Network Incidents Using Monitoring System. *Journal of Advanced Engineering and Computation*, 2019, October, vol. 1, no. 1, pp. 31–34.
2. Emanuel, Zgârdea, Ladislau, Augustinov. Improving IT Infrastructure Management Using Nagios Open Source Package. *Analele Universităţii "Eftimie Murgu" Reşiţa: Fascicola I, Inginerie*, 2014, December, pp. 334–336.
3. Sudhakar, Sushil, Kumar. An emerging threat Fileless malware: a survey and research challenges. *Cybersecurity*, 2020, February, pp. 1–4.
4. Croft, Roland, Xie, Yongzheng, Zahedi, Mansooreh, Babar, M. Ali, Treude, Christoph. An empirical study of developers' discussions about security challenges of different programming languages. *Empirical Software Engineering*, 2021, December, pp. 23–27.
5. The Powershell Gallery. Available at: https://docs.microsoft.com/en-us/powershell/scripting/gallery /overview?view=powershell-7.2 (accessed 21.12.2021).
6. Bertram, A. *Powershell dlya sisadminov* [Powershell for sysadmins]. Saint-Petersburg, Publishing House "Piter", 2021. 416 p.
7. Olifer, V., Olifer, N. *Kompyuternye seti. Printsipy tekhnologii, protokoly* [Computer networks. Principles, technologies and protocols]. Saint-Petersburg, Publishing House "Piter", 2016. 992 p.

DOI 10.54398/2074-1707_2022_1_113
УДК 004.001

МЕТОД ЗАЩИТЫ СИСТЕМЫ МАШИННОГО ОБУЧЕНИЯ ОТ ВРЕДОНОСНЫХ ПРОГРАММ

Статья поступила в редакцию 01.02.2022, в окончательном варианте – 15.02.2022.

Петренко Вячеслав Иванович, Северо-Кавказский федеральный университет, 355017, Российская Федерация, г. Ставрополь, ул. Пушкина, 1,

кандидат технических наук, и. о. директора Института цифрового развития, заведующий кафедрой организации и технологии защиты информации, ORCID: 0000-0003-4293-7013, e-mail: vipetrenko@ncfu.ru

Тебуева Фариза Биляловна, Северо-Кавказский федеральный университет, 355017, Российская Федерация, г. Ставрополь, ул. Пушкина, 1,

доктор физико-математических наук, заведующая кафедрой компьютерной безопасности, ORCID: 0000-0002-7373-4692, e-mail: ftebueva@ncfu.ru

Анзоров Артур Русланович**,** Северо-Кавказский федеральный университет, 355017, Российская Федерация, г. Ставрополь, ул. Пушкина, 1,

студент, ORCID: 0000-0002-1157-4021, e-mail: artanzrv@gmail.com

Стручков Игорь Владиславович, Северо-Кавказский федеральный университет, 355017, Российская Федерация, г. Ставрополь, ул. Пушкина, 1,

аспирант, ORCID: 0000-0001-8744-498X, e-mail: selentar@bk.ru

Статья посвящена проблеме защиты системы машинного обучения от вредоносного ПО. Проведен анализ возможных уязвимостей систем машинного обучения, приведена классификация наиболее опасных атак с описанием классов, включающих в себя способ воздействия и последствия от применения данных атак в системе машинного обучения. Для противодействия ряду атак предложен метод защиты системы машинного обучения от вредоносных программ на основе алгоритмов Neural-Cleanse и Jpeg-Compression.

Ключевые слова: машинное обучение, нейронные сети, информационная безопасность, Neural-Cleanse, Jpeg-Compression, атаки отравления, атаки искажения, атаки извлечения модели

A METHOD FOR PROTECTING A MACHINE LEARNING SYSTEM FROM MALICIOUS PROGRAMS

The article was received by the editorial board on 01.02.2022, in the final version – 12.02.2022.

Petrenko Vyacheslav I., North-Caucasian Federal University, 1 Pushkin St., Stavropol, 355017, Russian Federation,

Candidate Sci. (Engineering), Acting Director of the Institute for Digital Development, Head of the Department of Organization and Technology of Information Security, ORCID: 0000-0003-4293-7013, e-mail: vipetrenko@ncfu.ru

Tebueva Fariza B., North-Caucasian Federal University, 1 Pushkin St., Stavropol, 355017, Russian Federation,

Doct. Sci. of (Physics and Mathematics), Head of the Department of Computer Security, ORCID: 0000-0002-7373-4692, e-mail: ftebueva@ncfu.ru

Anzorov Artur R., North-Caucasian Federal University, 1 Pushkin St., Stavropol, 355017, Russian Federation,

student, ORCID: 0000-0002-1157-4021, e-mail: artanzrv@gmail.com

Struchkov Igor V., North-Caucasian Federal University, 1 Pushkin St., Stavropol, 355017, Russian Federation,

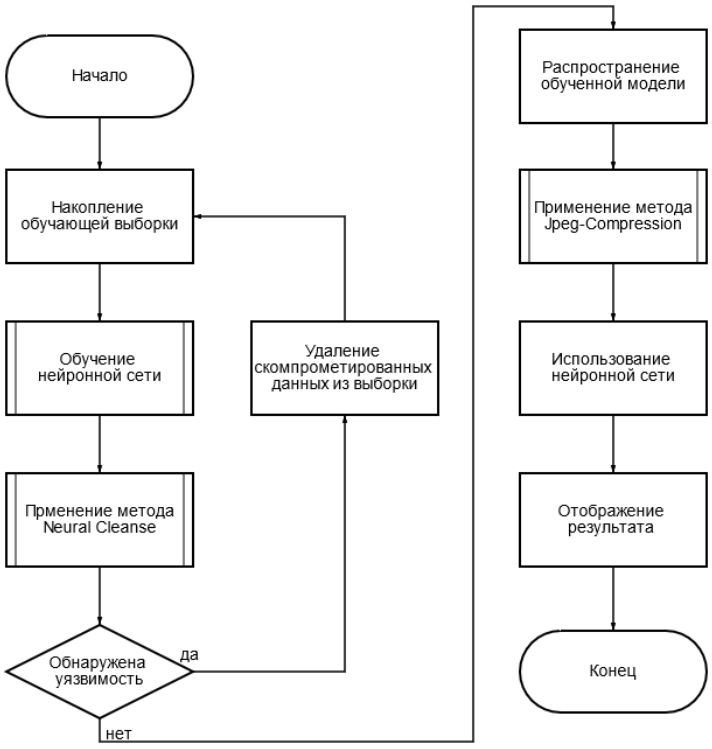
postgraduate student, ORCID: 0000-0001-8744-498X, e-mail: selenbar@bk.ru

The article is devoted to the problem of protecting a machine learning system from malware. An analysis of possible vulnerabilities of machine learning systems has been carried out, a classification of the most dangerous attacks with a description of classes, including the method of impact and consequences of using these attacks in a machine learning system, has been given. To counter a number of attacks, a method is proposed for protecting a machine learning system from malware based on the Neural-Cleanse and Jpeg-Compression algorithms.

Keywords: machine learning, neural networks, information security, Neural-Cleanse, Jpeg-Compression, poisoning attacks, evasion attacks, model extraction attacks

Graphical annotation (Графическая аннотация)

Метод защиты системы машинного обучения от вредоносных программ



Method for protecting a machine learning system from malware

Введение. За последние несколько лет машинное обучение стало заметным технологическим инструментом в нескольких прикладных областях, таких как компьютерное зрение, распознавание речи, понимание естественного языка, рекомендательные системы, поиск информации, компьютерные игры, медицинская диагностика, анализ рынка и т. д. Изучение и построение моделей с использованием обучающих данных дает злоумышленникам возможность атаковать алгоритмы машинного обучения, играя с функциями и границами решений модели. Злоумышленник может создавать вредоносные входные данные для атаки на производительность или эффективность алгоритма машинного обучения. Они могут обойти уже обученную систему, создав входные данные, которые используют систему для совершения существенных ошибок.

Поскольку машинное обучение становится важным инструментом в стратегически важных приложениях, таких как наблюдение за безопасностью и проверка биографических данных для решений о выдаче визы и т. д., важно понимать эти атаки и делать алгоритмы машинного обучения более устойчивыми к этим атакам. Если хакер еще не знает алгоритм, он сначала пытается изучить алгоритм и лежащую в его основе модель (например, логистическую регрессию, нейронную сеть, деревья решений и т. д.). Иногда хакер может быть заинтересован только в изучении модели, чтобы он мог построить свою собственную «копию» системы, используя изученную модель. Это может быть полезно, если приложение предлагается как услуга через API и с пользователей взимается плата за использование этих API.

Хакер может создать последовательность входных данных, а затем, наблюдая выходные данные системы, соответствующие этим входным данным, он может построить локальную модель, которая может быть очень близка к модели, используемой исходной системой. В зависимости от цены и условий лицензии на использование API, хакер может «украсть» модель за очень небольшую сумму денег. Из-за быстрого развития технологий атак на системы на основе ИИ существующие меры защиты могут со временем стать менее эффективными, но подходы к защите и их обоснование остаются. Кроме того, большинство представленных подходов исходят из академической среды и делают определенные допущения, которые необходимо учитывать при применении этих подходов на практике.

Атаки на системы машинного обучения. Уязвимости в системах машинного обучения могут возникать из-за ошибок при проектировании, из-за неизбежных алгоритмических ограничений или из-за сочетания обоих этих факторов [1]. В настоящее время во всем мире работает множество систем машинного обучения – от «прозрачных ящиков», имеющих полностью открытый код и обученных на общедоступных данных, до «черных», получающих входные данные по закрытым протоколам после обработки неустановленными функциями преобразования и передающих результаты через проприетарные API. Между двумя этими крайностями есть другие варианты, в том числе системы машинного обучения с открытым кодом, но проприетарными гиперпараметрами и обучающими данными, а также «черные ящики», действующие по принципу переноса обучения с «прозрачных ящиков» [2].

Атаки на системы, основанные на машинном обучении, можно разделить на 4 основные категории [3]:

1) атака искажения (evasion attack);
2) атака отравления (poisoning);
3) атака инверсии модели (model inversion);
4) атака извлечения модели (model extraction).

Атаки искажения являются наиболее распространенным типом атаки на систему машинного обучения. Вредоносные входные данные тщательно обрабатываются, чтобы избежать обнаружения, что, по сути, означает, что входные данные изменяются таким образом, чтобы алгоритм машинного обучения классифицировал их как безопасные, а не вредоносные, заставив сеть выдавать неверные ответы в определенных ситуациях. Использование враждебного пространства (рис. 1) для поиска враждебных (аномальных) экземпляров, которые приводят к неправильной классификации в классификаторе на основе машинного обучения, является основой атак искажения.

Атака отравления происходит, когда злоумышленник может внедрить неверные данные в тренировочный пул вашей модели и, следовательно, заставить его учиться чему-то, чего он не должен. Атаки отравления могут быть проведены в двух вариантах – те, которые нацелены на доступность модели машинного обучения, и те, которые нацелены на его целостность (также известные как атаки типа бэкдор). Целью злоумышленника является искажение выходных данных модели машинного обучения, добавив в систему зараженные данные, что приведет к ошибочной классификации. При отравлении злоумышленники пытаются повлиять на данные обучения, чтобы повлиять на результат обучения. Отравляющие атаки могут варьироваться от влияния на производительность алгоритма обучения до преднамеренного внесения в модель определенных смещений.

Рисунок 1 – Враждебное пространство как результат несовершенного представления тренировочных данных

Атака отравления происходит, когда злоумышленник может внедрить неверные данные в тренировочный пул вашей модели и, следовательно, заставить его учиться чему-то, чего он не должен. Атаки отравления могут быть проведены в двух вариантах – те, которые нацелены на доступность модели машинного обучения, и те, которые нацелены на его целостность (также известные как атаки типа бэкдор). Целью злоумышленника является искажение выходных данных модели машинного обучения, добавив в систему зараженные данные, что приведет к ошибочной классификации. При отравлении злоумышленники пытаются повлиять на данные обучения, чтобы повлиять на результат обучения. Отравляющие атаки могут варьироваться от влияния на производительность алгоритма обучения до преднамеренного внесения в модель определенных смещений.

Атаки инверсии данных – это атаки, при которых злоумышленники, имеющие возможность запрашивать адресную модель, пытаются вывести набор обучающих данных. Эти атаки нарушают конфиденциальность данных и особенно опасны для конфиденциальных данных и личных данных, таких как данные распознавания лиц, медицинские записи и финансовые транзакции. В атаках с инверсией модели, учитывая предсказание модели, злоумышленники находят входную выборку, которая предсказывается адресованной моделью для данного предсказания.

Атаки извлечения модели – это атаки, при которых злоумышленники, не зная адресованной модели, пытаются украсть возможности модели. Существует два основных подхода к атаке. Один из них заключается в краже параметров модели различными способами, такими как атаки по сторонним каналам или простое считывание параметров модели с базового устройства. Другим способом является создание замещающей модели, используя утечку информации из набора входных и выходных данных модели.

Предлагаемый метод защиты. Предлагается использовать метод, в основе которого будут лежать технологии защиты Neural-Cleance и Jpeg-Compression. Предполагается, что разрабатываемый метод обеспечит наилучшую защиту от вредоносного программного обеспечения, так как данный подход обеспечивает два уровня защиты системы машинного обучения – на этапе разработки и на этапе последующей эксплуатации обученной системы, что гарантирует безопасность системы машинного обучения от атак искажения и отравления. Рассмотрим подробнее используемые в методе технологии.

1. Neural-Cleanse [4] направлен на достижение трех конкретных целей:

1) обнаружение бэкдора: принять решение того, была ли данная глубокая нейронная сеть (DNN) заражена бэкдором. В случае заражения обнаружить, какой триггер использует бэкдор;

2) выявление бэкдора: определить ожидаемую работу бэкдора; более конкретно перепроектировать триггер, используемый для атаки;

3) устранение бэкдора: наконец, визуализировать бэкдор и сделать его неэффективным.

Рассмотрим подробнее вышеперечисленные цели.

Идея обнаружения бэкдора состоит в следующем. Пусть $\mathbb{L}$ представляет набор выходных меток в DNN. Рассмотрим метку $L_i \in \mathbb{L}$ и целевую метку $L_t \in \mathbb{L}$, $i \neq t$. Если существует триггер T_t, что приводит к неверной классификации в L_t, то минимальное возмущение, необходимое для преобразования всех меток (чья истинная метка L_i), которые должны быть классифицированы как L_t, ограничено размером триггера: $\delta_{\forall \to t} \leq |T_t|$, где $\delta_{\forall \to t}$ представляет собой минимальное количество возмущений, необходимых для того, чтобы любой вход был классифицирован как L_t. Кроме того, чтобы избежать обнаружения, величина возмущения должна быть небольшой, т. е. быть значительно меньше, чем требуется для преобразования любого ввода в незараженную метку.

Таким образом, возможно обнаружить триггер T_t путем обнаружения аномально низкого значения $\delta_{\forall\to i}$ среди всех выходных меток:

$$\delta_{\forall\to t} \le |T_t| \ll \min_{i,j\neq t} \delta_{\forall\to i}. \quad (1)$$

Ключевой подход при обнаружении бэкдоров заключается в том, что в зараженной модели требуется гораздо меньше модификаций, чтобы вызвать ошибочную классификацию в целевой метке, чем в других незараженных метках (1). Поэтому необходимо перебрать все метки модели и определить, требует ли какая-либо метка значительно меньших изменений для ошибочной классификации. Вся эта система состоит из следующих трех шагов:

Шаг 1. Для каждой метки, рассматриваемой как потенциальная целевая метка целевой бэкдор-атаки, разрабатывается схема оптимизации, чтобы найти минимальное искажение «триггер», который необходим для неправильной классификации всех образцов с другими метками в соответствии с этой целевой меткой. В области распознавания этот триггер определяет наименьший набор пикселей и связанную с ним интенсивность цвета, чтобы вызвать ошибочную классификацию.

Шаг 2. Повторяется шаг 1 для каждой выходной метки в модели для выявления потенциальных триггеров.

Шаг 3. После расчета потенциальных триггеров измеряется размер каждого триггера по количеству пикселей, которые есть у каждого триггера-кандидата, т. е. сколько пикселей заменяет триггер. Далее запускается алгоритм обнаружения выбросов для определения того, является ли какой-либо триггер-кандидат наименее значимым, чем другие кандидаты. Значительный выброс представляет собой реальный триггер, а метка, соответствующая этому триггеру, является меткой цели атаки бэкдора.

При выявлении бэкдора с помощью обнаружения выбросов методом оптимизации получается реверс-инженерный триггер для каждой целевой метки. Затем триггеры (и связанные с ними метки), которые отображаются как резкие выбросы в распределении, идентифицируются.

Для обнаружения выбросов используется простой метод, основанный на среднем абсолютном отклонении, которое, как известно, является устойчивым при наличии нескольких выбросов. Сначала вычисляется абсолютное отклонение между всеми точками данных и медианой. Медиана этих абсолютных отклонений называется M_{AD} и обеспечивает надежную меру дисперсии распределения. Затем индекс аномалии точки данных определяется как абсолютное отклонение точки данных, деленное на M_{AD}. Предполагая, что базовое распределение является нормальным распределением, для нормализации применяется постоянная оценка индекса аномалии. Любая точка данных с индексом аномалии больше чем 2 имеет 95% вероятность являться вредоносным выбросом. Маркируется любая метка с индексом аномалии больше 2 как выброс и инфицирование, т. е. работа ведется только на выбросах в конце распределения.

Для того чтобы устранить бэкдора и обеспечить смягчение последствий применения бэкдора, необходимо:

- во-первых, создать фильтр для враждебных входных данных, который идентифицирует с помощью обратного триггера и отклоняет любой вход, который содержит триггер, что дает время для исправления модели. В зависимости от приложения этот подход также можно использовать для присвоения «безопасной» выходной метки враждебному изображению без отклонения;
- во-вторых, исправить DNN, делая его невосприимчивым к обнаруженным триггерам. Описывается два метода исправления, один – с использованием обрезки нейронов, а другой – на основе переобучения.

Фильтр для обнаружения враждебных входных данных. Активация нейронов – лучший способ уловить сходство между оригинальными и реконструированными триггерами. Таким образом, разрабатывается фильтр на основе профиля активации нейронов для обратного триггера. Его работа измеряется как средняя активация нейронов предпоследнего слоя. При наличии некоторых входных данных фильтр идентифицирует потенциальные вредоносные входные данные как те, профили активации которых превышают определенный порог. Порог активации можно откалибровать, используя тесты на чистых изображениях (изображениях, которые не содержат триггеров). Соответственно, это позволяет отклонять при подаче на вход любое враждебное изображение, содержащее триггер.

Чтобы исправить зараженную модель, предлагается два подхода.

В первом подходе идея состоит в том, чтобы использовать обратный триггер, чтобы помочь идентифицировать компоненты, связанные с бэкдором, в DNN. Например, выявить зараженные нейроны и удалить их. Предлагается удалить нейроны, связанные с бэкдором, из DNN, т. е. установить выходное значение этих нейронов на 0. Работа также идет с предпоследним слоем и производится обрезка нейронов по порядку с наивысшим рангом первым (т. е. отдавая приоритет тем, кото-

рые демонстрируют самый большой разрыв в активации между чистыми и враждебными входными данными). Чтобы свести к минимуму влияние на точность классификации чистых входных данных, прекращается удаление нейронов, когда сокращенная модель больше не реагирует на обратный триггер. Обрезка 30 % нейронов снижает вероятность успеха атаки почти до 0 %.

При этом точность классификации снижается только на 5,06 %. Так как чистые нейроны сильно перемешаны с враждебными нейронами, это приводит к удалению чистых нейронов во время процесса, что снижает точность классификации. Таким образом, отсечение на последнем слое свертки дает наилучшие результаты. Более того обрезка имеет сильные стороны и дополнительные ограничения, так как этот подход требует очень мало вычислений, большая часть которых включает в себя вывод чистых и состязательных изображений. Однако само ограничение подхода заключается в том, что производительность зависит от выбора правильного слоя для обрезки нейронов, и это может потребовать экспериментов с несколькими слоями. Кроме того, к нему предъявляются высокие требования в отношении того, насколько хорошо обращенный триггер соответствует исходному триггеру.

Второй подход к смягчению последствий заключается в том, чтобы обучить DNN распознавать первоначальный триггер. Возможность использовать обратный триггер, чтобы научить зараженную DNN распознавать правильные метки, даже если триггер присутствует. По сравнению с отсечением нейронов, отмена обучения позволяет модели посредством обучения решать, какие веса (не нейроны) являются проблематичными и должны быть обновлены. Модель дорабатывается только для 1 эпохи обучения, используя обновленный набор обучающих данных. Чтобы создать этот новый обучающий набор, берется 10% образец исходных обучающих данных (чистых, без триггеров) и производится добавление инвертированного триггера к 20 % этого исходного образца. Чтобы измерить эффективность исправления, измеряется вероятность успеха атаки исходного триггера и точность классификации точно настроенной модели.

Очевидным преимуществом является исправление DNN через переучивание, поскольку эффективность переобучения, как правило, нечувствительна к таким параметрам, как количество обучающих данных и соотношение модифицированного обучения. Наконец, следует отметить, что переобучение требует более высоких вычислительных затрат по сравнению с обрезкой нейронов. Однако это все же на один-два порядка меньше, чем переобучение модели с нуля.

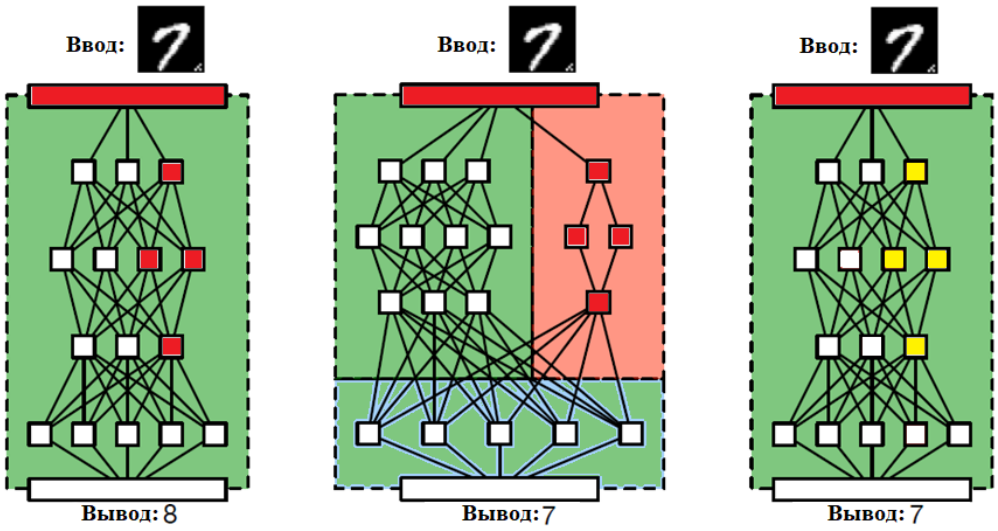


Рисунок 2 – Зараженные нейроны, обрезка зараженных нейронов, переобучение

Jpeg-Compression. Глубокие нейронные сети (DNN) добились больших успехов в решении различных задач машинного обучения (ML), особенно в области распознавания изображений. Однако недавние исследования показали, что DNN могут быть очень уязвимы для генерируемых злоумышленниками экземпляров, которые кажутся обычными для человека-наблюдателя, но полностью сбивают с толку DNN. Эти враждебные образцы создаются путем добавления небольших возмущений к нормальным, безвредным изображениям. Такие возмущения, хотя и незаметные для человеческого глаза, улавливаются DNN и заставляют их ошибочно классифицировать управляемые экземпляры с высокой степенью достоверности.

Важным компонентом сжатия JPEG является его способность удалять высокочастотные компоненты сигнала внутри квадратных блоков изображения. Такая операция эквивалентна выборочному размытию изображения, помогая устранить аддитивные возмущения. Кроме того, данный метод, использующий сжатие JPEG, может защитить модель от нескольких типов враждебных атак, не требуя знаний о модели.

Данный метод [5] предполагает использовать сжатие JPEG в качестве простого и эффективного шага предварительной обработки изображения для удаления враждебного шума. Поскольку посторонние шумы часто неразличимы человеческим глазом, сжатие JPEG, предназначенное для выборочного отбрасывания незаметной для человека информации, имеет большой потенциал в борьбе с такими манипуляциями.

Данный подход имеет несколько желаемых преимуществ [6]:

Во-первых, JPEG – это широко используемый метод кодирования, и многие изображения уже хранятся в формате JPEG. Большинство операционных систем также имеют встроенную поддержку кодирования и декодирования изображений JPEG, поэтому даже неопытные пользователи могут легко применить этот этап предварительной обработки.

Во-вторых, этот подход не требует знаний ни о модели, ни об атаке и может применяться к широкому диапазону наборов данных изображений.

Основной принцип сжатия JPEG основан на психовизуальной системе человека, которая направлена на подавление высокочастотной информации, такой как резкие переходы интенсивности и цветового оттенка, с помощью дискретного косинусного преобразования. Поскольку враждебные атаки часто вызывают возмущения, несовместимые с человеческим психовизуальным сознанием (следовательно, эти атаки иногда незаметны для человека), считается, что сжатие JPEG может удалить эти артефакты.

Доброкачественные, повседневные образы лежат в очень узком многообразии. Изображение с полностью случайными цветами пикселей вряд ли будет восприниматься людьми как естественное. Однако объективная основа моделей классификации, таких как DNN, часто не согласуется с такими соображениями. ГНС можно рассматривать как построение границ решений, которые линейно разделяют данные в многомерных пространствах. При этом в этих моделях предполагается, что подпространства естественных образов существуют за пределами фактического многообразия. Враждебные атаки используют это преимущество, искажая изображения ровно настолько, чтобы они пересекали границу решения модели. Однако этот кроссовер не гарантирует, что возмущенные изображения будут лежать в исходном узком многообразии.

Поскольку при сжатии JPEG учитывается психовизуальная система человека, возникают гипотезы о том, что многообразие, в котором появляются изображения JPEG, будет иметь некоторое сходство с многообразием естественных изображений и что использование сжатия JPEG в качестве шага предварительной обработки во время классификации будет проецировать любые экземпляры, искаженные противником, обратно на это многообразие.

Главным операциям предшествует преобразование RGB-схемы изображения в схему YUV с ее последующей дискретизацией. Затем изображение разбивается на блоки размером 8x8 пикселов. Каждый блок подвергается ДКП, осуществляющемуся по формуле (2):

$$\text{ДКП}(i,j) = C(i) * C(j) * \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y) * cos\left[\frac{(2x+1)i\pi}{2N}\right] * cos\left[\frac{(2y+1)j\pi}{2N}\right],$$

$$C(i), C(j) = \begin{cases} \sqrt{\frac{1}{N}} \text{ при } i,j = 0 \\ \sqrt{\frac{2}{n}} \text{ при } i,j = 1,2,\dots,N-1 \end{cases}, \quad (2)$$

где f(x, y) – пиксел изображения.

В результате получается набор матриц коэффициентов ДКП, каждую из которых составляет низкочастотный коэффициент DC (нулевой коэффициент матрицы) и высокочастотные коэффициенты AC. Каждая матрица коэффициентов подвергается квантованию при помощи заранее заданной таблицы квантования. На данном этапе происходит наибольшая потеря информации. При этом большое количество высокочастотных коэффициентов подвергается обнулению. Затем применяется зигзаг-преобразование (рис. 3).

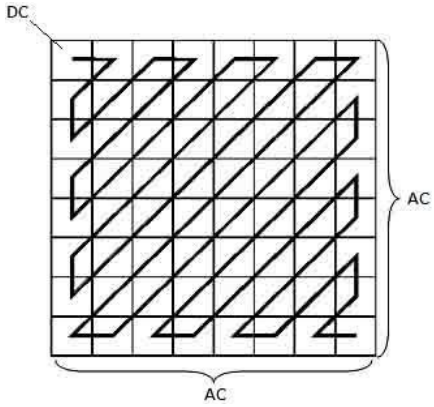


Рисунок 3 – Диагональное «зигзаг»-сканирование спектральных компонент

Восстановление изображения осуществляется при помощи тех же операций, но осуществляемых в обратном порядке. При этом вместо прямого ДКП используется обратное ДКП, формула которого представлена ниже:

$$f(x,y) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} C(i) * C(j) * \text{ДКП}(x,y) cos\left[\frac{(2x+1)i\pi}{2N}\right] cos\left[\frac{(2y+1)j\pi}{2N}\right],$$

$$C(i), C(j) = \begin{cases} \sqrt{\frac{1}{N}} \text{ при } i,j = 0 \\ \sqrt{\frac{2}{n}} \text{ при } i,j = 1,2,\dots,N-1, \end{cases} \quad (3)$$

где $\text{ДКП}(x,y)$ – дискретно-косинусное преобразование.

Таков общий принцип работы JPEG-сжатия (рис. 4).

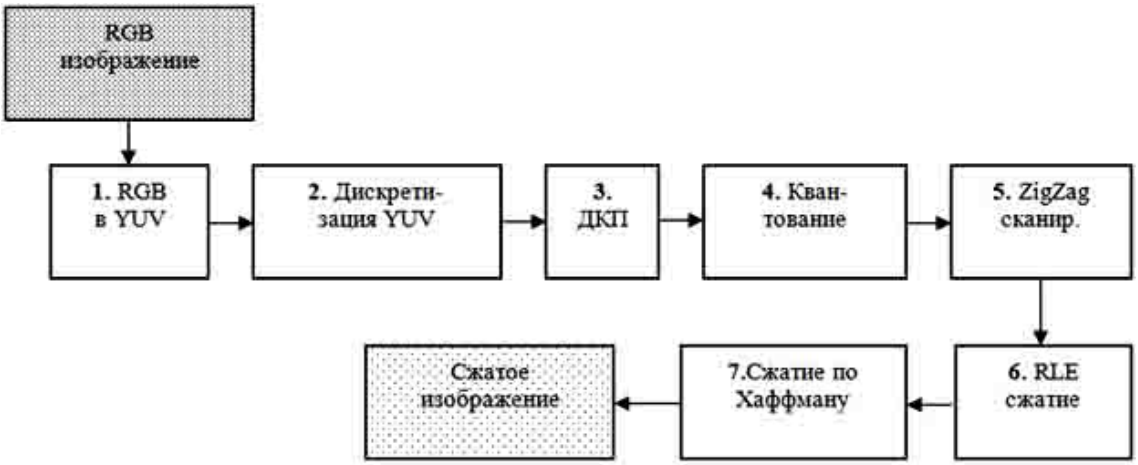


Рисунок 4 – Блок-схема алгоритма Jpeg-Compression

Блок-схема алгоритма Jpeg-Compression демонстрирует общий универсальный алгоритм сжатия.

Таким образом, для обеспечения безопасности процесса обучения и использования нейронной сети предлагается использовать метод защиты системы машинного обучения от вредоносных программ, использующий комбинацию технологий Neural-Cleanse и Jpeg-Compression. Блок-схема работы предлагаемого метода представлена на рисунке 5.

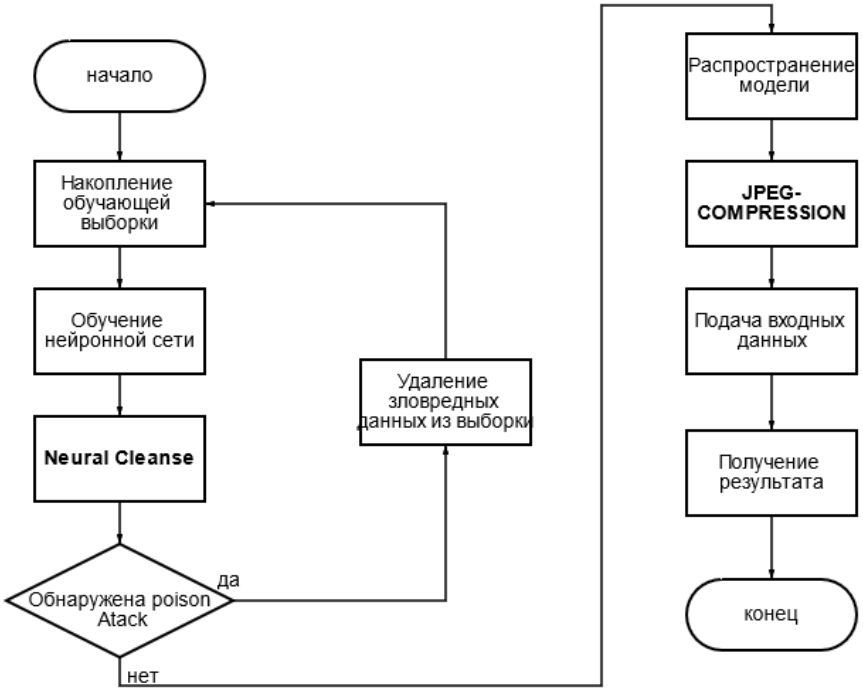


Рисунок 5 – Метод защиты системы машинного обучения от вредоносных программ

Реализация метода. Предлагается реализовать лежащие в основе метода рассмотренные концепции защиты раздельно, чтобы обеспечить максимальную степень защиты на каждом уровне независимо друг от друга, так как применение первой технологии для противодействия атаке отравления предполагает в процессе переобучения состязательную тренировку, которая будет влиять на эффективность атаки искажения и, следовательно, демонстрацию эффективности второй технологии, поэтому необходимо использовать два разных эксперимента с различными данными. В первом эксперименте, на этапе разработки (в процессе обучения), обеспечивается защита прозрачной модели (белого ящика) от атак типа бэкдор (разновидность атак отравления). Во втором эксперименте обеспечивается защита уже готовой модели (черного ящика) от атаки искажения. Данный подход позволит продемонстрировать состоятельность и универсальность предлагаемого метода защиты независимо от модели машинного обучения, обучающих данных, входных данных, используемых в процессе эксплуатации системы.

1. Пример реализации Neural-Cleance.

Используемый язык программирования – Python с пакетами NumPy, SciPy и MatPlotLib.

Используемая система машинного обучения – TensorFlow (сквозная платформа машинного обучения с открытым исходным кодом).

Keras – это API глубокого обучения, написанный на Python и работающий поверх платформы машинного обучения TensorFlow.

Используемый набор данных для обучения – MNIST (большая коллекция рукописных цифр).

Для демонстрации работы алгоритма защиты от атак отравления будем использовать систему машинного обучения Keras, которая будет обучаться на наборе данных MNIST, распознавать рукописные цифры. Поскольку цель злоумышленника состоит в том, чтобы отравить модель, используя атаку отравления типа бэкдор. Выберем цифру 0 в качестве целевого класса путем добавления к изображениям этого класса специального триггера, что приведет к тому, что обученная модель ошибочно классифицирует сработавший ввод как 1. Затем оценим поведение модели на тестовых (на изображениях до отравления) и отравленных образцах.

Для того чтобы защитить модель от отравляющих данных, используем технологию Neural-Cleance. Для этого сначала идентифицируется бэкдор. Процедура этой защиты заключается в точном определении предполагаемого бэкдора для каждого класса, затем добавлении переработанного бэкдора к чистым изображениям каждого класса, чтобы имитировать поведение бэкдора. Во время этого процесса защиты генерируется подозрительный бэкдор для каждого класса, представленного выше. После того как бэкдор был идентифицирован, необходимо использовать

предварительную фильтрацию, т. е. процесс воздержания от потенциально ядовитых прогнозов во время выполнения. Нейроны ранжируются по их связи с бэкдором, и когда нейронные активации выше, чем обычно, классификатор воздерживается от предикации (на выходе все нули), это позволит предотвратить классификацию поданных на вход зараженных изображений. Затем выбирается способ избавления от бэкдора (переобучение, обрезка зараженных нейронов). В данном случае используется переобучение с использованием обратного триггера. Затем необходимо провести повторную оценку поведения модели на тестовых (на изображениях до отравления) и отравленных образцах. Если бэкдор сохраняется, применяется обрезка зараженных нейронов.

Для проведения эксперимента были использованы:

- операционная система Windows 10;
- высокоуровневый язык программирования Python версии 3.10;
- среда разработки Jupyter Notebook на основе Anaconda 4.11.10;
- система машинного обучения TensorFlow 2.8.0, Keras 2.4.0;
- набор данных MNIST.

Выбор цифры (метки) для применения бэкдора и применение триггера приведен на рисунке 6, результат обучения нейронной сети на тренировочных образцах (чистых 70 % и отравленных 30 %) приведен на рисунке 7. На рисунке 8 показано, что точность классификации чистых образцов составляет 96,55 %. Точность оценки классификации отравленных образцов составляет 100 % (рис. 9). На рисунке 10 приведен результат идентификации бэкдора в наборе данных. Предварительная фильтрация входящих образцов представлена на рисунке 11. Результат переобучения приведен на рисунке 12. Повторная оценка точности классификации образцов (чистых и отравленных) после переобучения приведена на рисунке 13. Процесс обрезки зараженных нейронов показан на рисунке 14. Повторная оценка точности классификации образцов (чистых и отравленных) после обрезки нейронов приведена на рисунке 15.

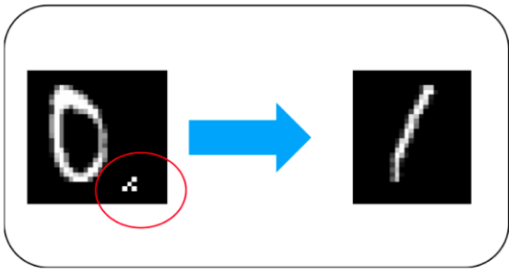


Рисунок 6 – Используемый триггер в атаке типа бэкдор

```
Train on 7918 samples
Epoch 1/3
7918/7918 [==============================] - 18s 2ms/sample - loss: 0.8216 - accuracy: 0.7281
Epoch 2/3
7918/7918 [==============================] - 15s 2ms/sample - loss: 0.2535 - accuracy: 0.9238
Epoch 3/3
7918/7918 [==============================] - 17s 2ms/sample - loss: 0.1583 - accuracy: 0.9523
```

Рисунок 7 – Результат обучения нейронной сети Keras

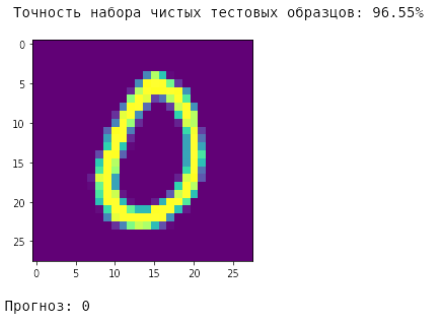
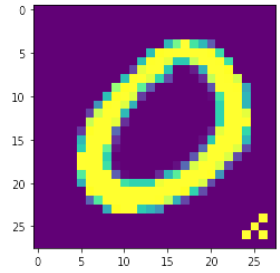


Рисунок 8 – Результат классификации чистого изображения ‘0’




```
Прогноз: 1

Эффективность отравления: 100.00%
```

Рисунок 9 – Результат классификации отравленного изображения ‘0’

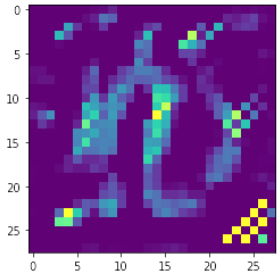


Рисунок 10 – Результат идентификации бэкдора в наборе данных

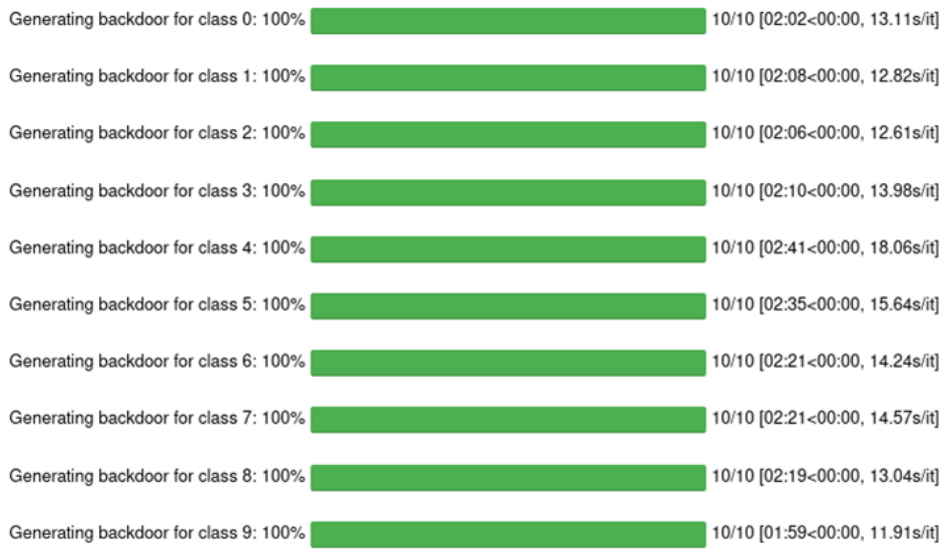


Рисунок 11 – Результат фильтрации отравленных образцов в наборе данных

```
Эффективность отравления после переобучения: 75.58% (ранее 100%)

Точность набора чистых тестовых образцов: 92.73% (ранее 96.55%)
```

Рисунок 12 – Результат применения переобучения в нейронной сети Keras и повторной оценки точности всего набора тестов

В данном эксперименте использование переобучения снижает эффективность отравления до 75,58 %, затем использование обрезки нейронов снижает эффективность отравления до 0,00 %, но точность классификации снижается до 83,41 %. Следовательно, при поочередном применении этих способов очистки зараженных нейронов эффективность отравления снижается до минимального уровня, практически не влияющего на распознавание, а классификация чистых образцов остается на допустимом уровне.

Рисунок 13 – Результат повторной оценки точности тестовых образцов (чистых и отравленных) и всего набора тестов после применения переобучения

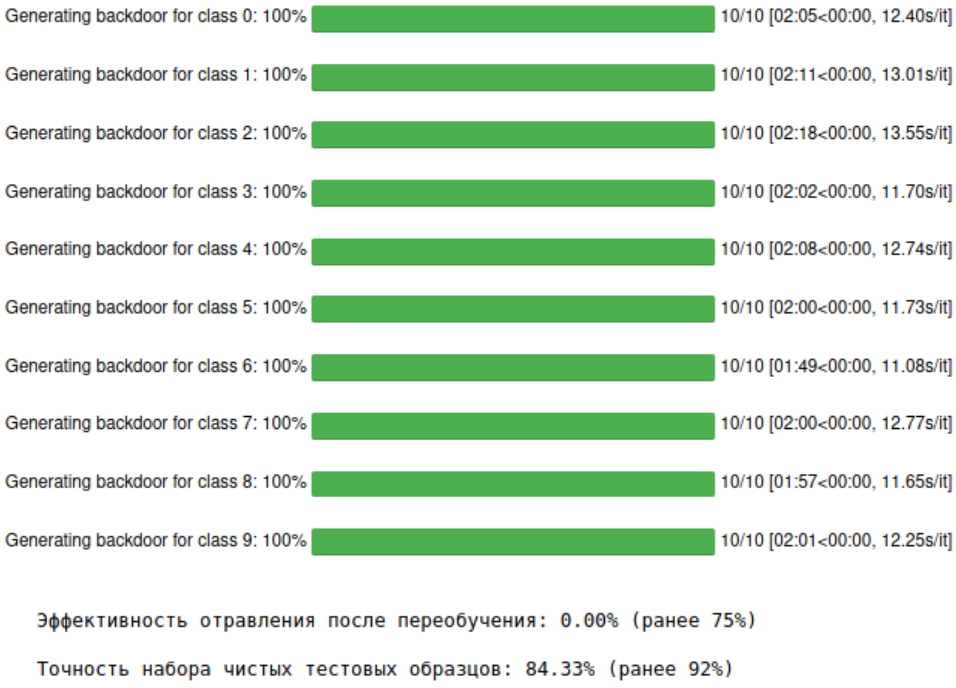


Рисунок 14 – Результат применения обрезки зараженных нейронов в сети Keras и оценки точности всего набора тестов

Рисунок 15 – Результат повторной оценки точности тестовых образцов (чистых и отравленных) и всего набора тестов после применения обрезки

2. Пример реализации Jpeg-Compression.

Используемый язык программирования – Python с пакетами NumPy, SciPy и MatPlotLib.

Используемая система машинного обучения – TesseractOCR.

Для демонстрации работы алгоритма защиты от состязательных атак будем использовать обученную систему машинного обучения Tesseract-OCR, которая распознает слова на изображениях, затем выводит их в виде текста в кодировке UTF-8. Подадим на вход Tesseract-OCR изображение, содержащее слово “assent”, и изображение, содержащее слово “dissent”, которые система должна распознать и преобразовать в текст. Состязательная атака заключается в наложении на изображение “assent” изображения “dissent” путем использования атаки искажения перед подачей на вход системы, чтобы вызвать ошибку в работе классификатора. Схематически это можно изобразить следующим образом. Для защиты от атаки искажения будет применяться метод Jpeg-Compression к обрабатываемому изображению “assent”. Схема атаки искажения приведена на рисунке 16.

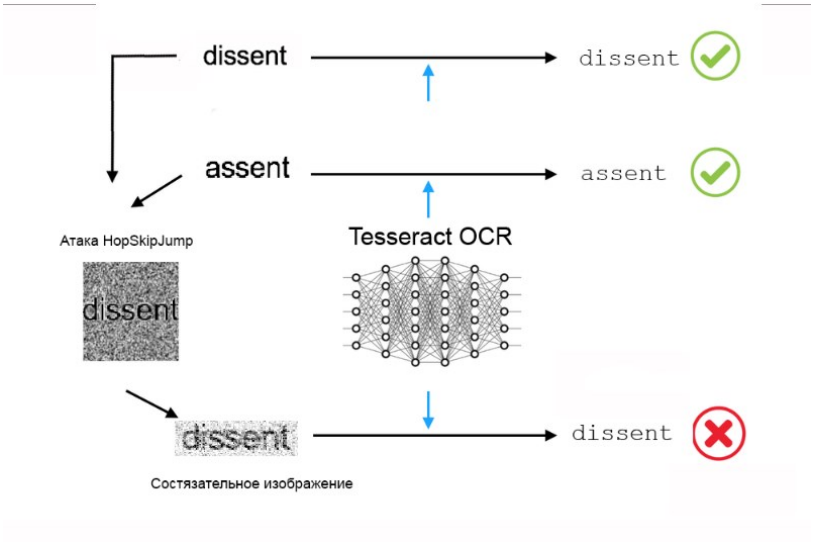


Рисунок 16 – Схема атаки искажения

Для проведения эксперимента были использованы:

- операционная система Windows 10;
- высокоуровневый язык программирования Python версии 3.10;
- среда разработки Jupyter Notebook на основе Anaconda 4.11.10;
- система машинного обучения Tesseract-OCR v5.0.1.20220118.

Распознавание атакуемого изображения “assent” и изображения, используемого для атаки “dissent”, показано на рисунке 17.

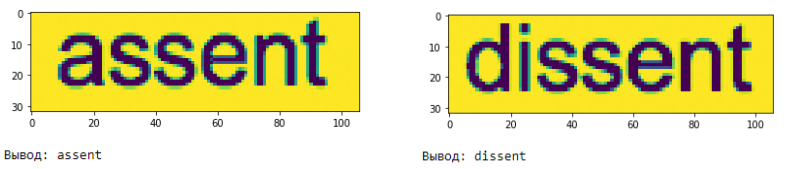


Рисунок 17 – Результат распознавания изображений “assent” и “dissent”

Атака искажения (наложение изображения “dissent” на “assent”) и дальнейшее распознавание текста на обработанном изображении приведены на рисунке 18.

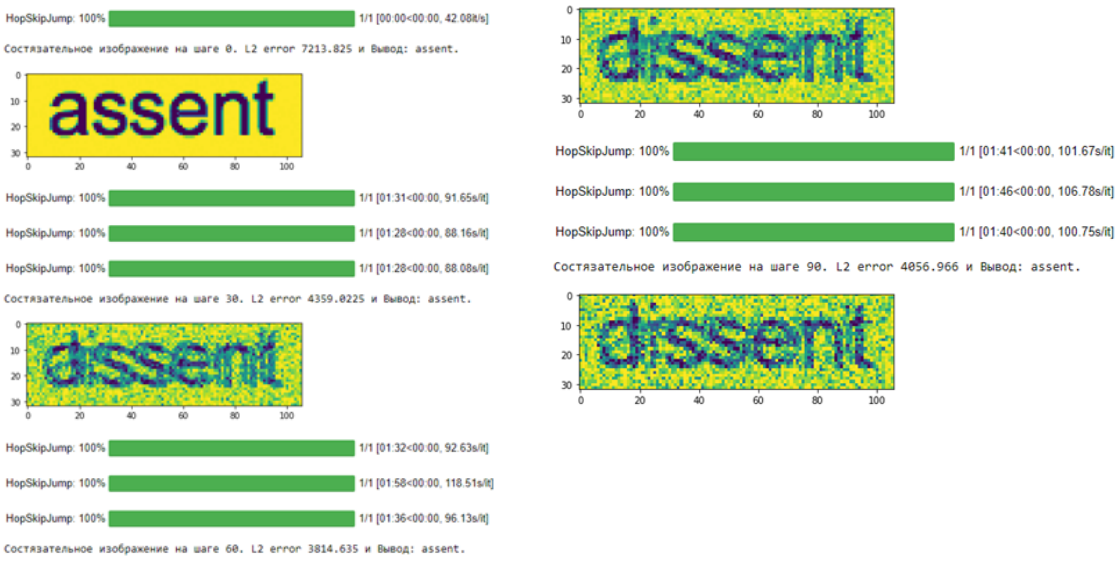


Рисунок 18 – Результат атаки и вывод распознанного текста на атакованном изображении

Применение Jpeg-Compression к изображению “assent“, повторная атака искажения (наложение изображения “dissent” на “assent”) и дальнейшее распознавание текста на обработанном изображении после применения метода защиты показаны на рисунке 19.

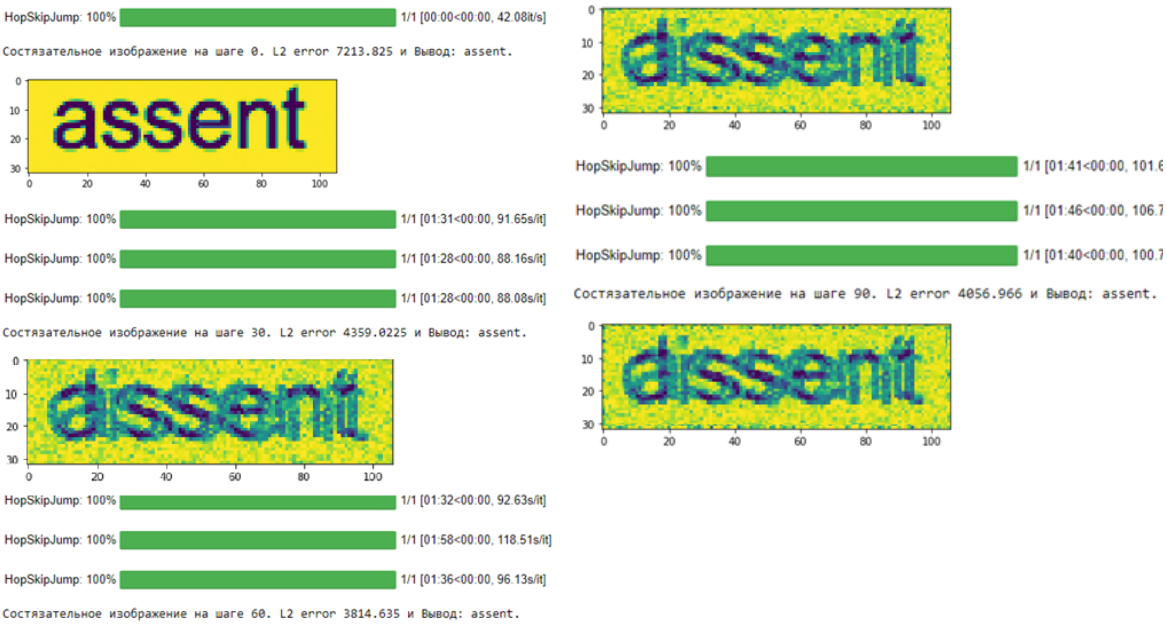


Рисунок 19 – Результат повторной атаки и вывод распознанного текста на атакованном изображении, к которому был применена технология Jpeg-Compression

В результате эксперимента на шаге 1 слова “assent” и “dissent” были распознаны правильно. После применения атаки искажения на шаге 3 обработанное изображение неверно распознается как “dissent”. Затем, применив технологию Jpeg-Compression к изображению “assent” на шаге 4, получаем в результате повторной атаки искажения безошибочную классификацию “assent”.

Таким образом, использование сжатия JPEG позволяет правильно классифицировать и распознать подаваемое на вход системы машинного обучения Tesseract-OCR изображение с текстом путем предотвращения ошибок, вызванных какими-либо возмущениями в результате преднамеренного или непреднамеренного искажения распознаваемого изображения.

Заключение. В данной работе рассмотрена проблема защиты систем машинного обучения от вредоносного ПО. Для решения этой проблемы проведен анализ возможных атак на системы

машинного обучения, разработан метод, в основе которого лежат алгоритмы Neural-Cleanse, Jpeg-Compression.

Для обоснования эффективности данного метода были постановлены и проведены соответствующие экспериментальные исследования. В результате были обнаружены и подтверждены следующие закономерности:

- нейронные сети, переобученные на отравленных рукописных изображениях, оказались более устойчивы к атакам типа бэкдор, нежели сети, обученные для классификации чистых рукописных изображений;
- атаки искажения стали неэффективными, в то время как точность распознавания осталась на достаточном уровне.

Главным результатом внедрения данного метода в систему машинного обучения стало уменьшение последствий атак. Модуль Neural-Cleance обеспечил снижение воздействия отравленных данных на нейронную сеть и повысил общую точность классификации системы в целом. Кроме этого, использование модуля Jpeg-Compression позволило обезопасить входные данные от воздействия атак искажения, что дополнительно повысило точность распознавания при преднамеренном или случайном искажении входных данных.

Библиографический список

1. Adversarial Machine Learning / A. D. Joseph, N. Blaine, B. I. Rubinstein, J. D. Tygar. ─ Cambridge : Cambridge University Press, 2019. – P. 338.

2. Макгроу, Г. Обеспечение безопасности систем машинного обучения / Г. Макгроу, Р. Бонетт, Х. Фигероа, В. Шепардсон // Открытые системы. СУБД. – 2019. – № 4. – С. 22.

3. Matthew, Stewart. Security Vulnerabilities of Neural Networks [Towards data science] / Matthew Stewart. – Режим доступа: https://towardsdatascience.com/hacking-neural-networks-2b9f461ffe0b, свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 20.11.2021).

4. How to attack Machine Learning (Evasion, Poisoning, Inference, Trojans, Backdoors) [Towards data science]. – Режим доступа: https://towardsdatascience.com/how-to-attack-machine-learning-evasion-poisoning-inference-trojans-backdoors-a7cb5832595c?gif=true, свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 30.12.2021).

5. Bolum, Wang. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks/ Bolum Wang, Yuanshum Yao, Shawn Shan, Huiying Li // Conference: 2019 IEEE Symposium on Security and Privacy. – 2019.

6. Nilaksh, Das. Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression / Nilaksh Das, Madhuri Shanbhogue, Shang-Tse Chen, Fred Hohman, Li Chen, Michael E. Kounavis, Duen Horng Chau // arXiv.org. –2017. – Режим доступа: arXiv.org, свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 11.12.2021).

7. Gintare, Karolina Dziugaite. A study of the effect of JPG compression on adversarial images/ Gintare, Karolina Dziugaite Zoubin Ghahramani, Daniel M. Roy // arXiv.org. – 2016. – Режим доступа: arXiv.org, свободный. – Заглавие с экрана. – Яз. англ. (дата обращения: 11.12.2021).

References

1. Joseph, A. D., Blaine, N., Rubinstein, B. I., Tygar, J. D. *Adversarial Machine Learning*. Cambridge, Cambridge University Press, 2019, p. 338.

2. McGraw, G., Bonett, R., Figueroa, H., Shepardson, V. Obespecheniye bezopasnosti sistem mashinnogo obucheniya [Ensuring the security of machine learning systems]. *Open Systems. DBMS*, 2019, no. 4, p. 22.

3. Matthew, Stewart. *Security Vulnerabilities of Neural Networks* [Towards data science]. Available at: https://towardsdatascience.com/hacking-neural-networks-2b9f461ffe0b (accessed 11.20.2021).

4. How to attack Machine Learning (Evasion, Poisoning, Inference, Trojans, Backdoors) [Towards data science]. Available at: https://towardsdatascience.com/how-to-attack-machine-learning-evasion-poisoning-inference-trojans-backdoors-a7cb5832595c?gif=true (accessed 12.30.2021).

5. Bolum, Wang, Yuanshum, Yao, Shawn, Shan, Huiying, Li.Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. *Conference: 2019 IEEE Symposium on Security and Privacy*, 2019.

6. Nilaksh, Das, Madhuri, Shanbhogue, Shang-Tse, Chen, Fred, Hohman, Li, Chen, Michael E., Kounavis, Duen, Horng Chau. Keeping the Bad Guys Out: Protecting and Vaccinating Deep Learning with JPEG Compression. *arXiv.org*, 2017. Available at: arXiv.org (accessed 11.12.2021).

7. Gintare, Karolina Dziugaite, Zoubin, Ghahramani, Daniel M., Roy A study of the effect of JPG compression on adversarial images. *arXiv.org*, 2017. Available at: arXiv.org 2016 (accessed 11.12.2021).