Выделение триггера и маски из отравленных обучающих данных при помощи модифицированных методов Activation Clustering и Neural Cleanse

И.П. Лозинский, В.В. Костюмов, Е.Н. Строева

***Аннотация*—В некоторых работах было неоднократно замечено, что популярный метод Neural Cleanse плохо справляется с восстановлением триггеров и масок, которые занимают значительную часть изображения, поскольку метод ищет наименьшее отравляющее изменение. Для решения этой проблемы в нашей статье предложен метод извлечения триггера из усреднённого изображения отравленного кластера изображений. Триггер можно выделить фильтрацией по интенсивности цвета пикселей усреднённого изображения. Для выделения кластеров изображений используется модификация метода Activation Clustering. Эксперименты проводились на данных с соревнования Trojan Detection Challenge, NeurIPS 2022. В этих данных один триггер переводит любое изображение в целевой класс.**

Однако в такой модели отравления оригинальный метод Activation Clustering показывает плохие результаты. В связи с этим в статье была предложена модифицированная версия метода Activation Clustering, в которой мы: подобрали гиперпараметры, выделили дополнительные признаки и обучили поверх них CatBoost классификатор отравленных изображений. Для восстановления маски к выделенному триггеру была разработана модификация метода Neural Cleanse.

Разработанный метод показывает значительно более высокое качество выделения триггера в сравнении с оригинальным Neural Cleanse.

***Ключевые слова*—выделение триггера, машинное обучение, атаки backdoor**

I. Введение

Системы моделей машинного обучения быстро увеличиваются в размерах, приобретают новые возможности и все чаще используются в условиях повышенной ответственности. Как и в случае с другими технологиями, безопасность для моделей машинного обучения должна быть одним из основных исследовательских приоритетов [1].

Зачастую сложные обученные модели плохо поддаются интерпретации, и, в большой степени, представляют собой чёрный ящик. Это свойство делает их уязвимыми к различным отравляющим атакам, поскольку проверка корректности работы модели на всём пространстве признаков зачастую является практически невыполнимой задачей [2].

Методы атак отравлением в машинном обучении можно подразделить на 3 класса:

- атаки отравлением обучающих данных;
- атаки отравлением процесса обучения [3], [4];
- атаки отравлением готовой модели [5], [6].

В данной работе рассмотрены методы атак отравлением обучающих данных на примере моделей для классификации изображений. Однако идеи данных методов также могут быть применены для других модальностей данных и задач машинного обучения [7], [8].

A. Обозначения и основные определения

- X — множество чистых изображений;
- X' — множество отравленных изображений;
- $D = X, Y$ — набор данных;
- $D' = X', Y'$ — отравленный набор данных;
- y_t — целевой класс в задачах отравления;
- m — маска наложения триггера на изображения в задачах внедрения триггера;
- Δ — триггер в задачах внедрения триггера;
- $A(x, m, \Delta) = (1 - m) * x + m * \Delta$ — функция наложения триггера на изображение x;
- $h(x)$ — функция извлечения признаков (выходы последнего скрытого слоя модели);
- $cls(x)$ — функция классификации по извлеченным признакам;
- $f(x) = cls(h(x))$ — функция применения модели.

Триггер — тензор такой же размерности, как изображение, который содержит произвольный паттерн, произвольные значения пикселей. Более подробно триггеры и их виды рассмотрены в секции II.

Маска — тензор такой же размерности, как изображение, который определяет пиксели изображения, на которые будет наложен триггер и интенсивность этого наложения, согласно функции наложения, определённой выше.

II. Атаки внедрением триггера (Backdoor)

Атаки внедрением триггера ставят своей целью изменить набор обучающий данных D таким образом, чтобы любое изображение, при наложении на него специального триггера, классифицировалось в целевой класс y_t. При этом так же, как и в случае обычных атак отравлением, нужно оказать как можно меньшее влияние на классификацию остальных изображений.

Примером триггера может быть помещение небольшого квадрата из чёрных или цветных пикселей в угол изображения.

Виды триггеров можно подразделить на:

- патчи — триггеры, которые накладываются поверх небольшой части изображения, обычно в одном и том же месте. Но встречаются и атаки, которые размещают триггеры в нескольких местах изображения или в случайном месте определённой области изображения [9];

Рис. 1. Пример наложенного триггера в виде патча: на первой картинке исходное изображение, на второй и третьей – изображение с наложенными триггерами

- шумы — триггеры в виде шума, который накладывается равномерно на всё изображение;

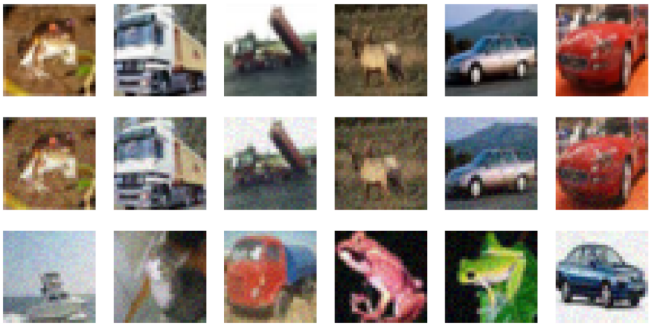

Рис. 2. Пример наложенного триггера в виде шума: первая строка содержит чистые изображения, вторая и третья — изображения с наложенным триггером в виде шума

- триггеры в признаковом пространстве – более сложный вид триггеров, которые накладываются посредством появления на изображении какого либо объекта или изменения яркости, тональности изображения [10].

Примеры различных видов триггеров приведены на рисунках 1, 2, 3.

Рис. 3. Пример наложенного триггера в признаковом пространстве: первая строка содержит исходные изображения, вторая строка – изображения с наложенным триггером

III. Современные методы защиты

A. Activation Clustering: поиск отравленных примеров при помощи кластеризации

Идея метода детектирования [2]: провести набор данных через обученную нейронную сеть и получить набор векторов активаций последнего скрытого слоя. Затем этот набор проецируется на пространство меньшей размерности (например, 10) и разбивается по классам. В каждом классе отдельно производится кластеризация полученных векторов. Замечено, что отравленные примеры и чистые примеры чаще всего разделяются в разные кластеры.

В оригинальной статье использовалась модель отравления, когда наложение триггера изменяет класс изображения на следующий. То есть, например для набора данных MNIST цифры 6 начинают классифицироваться как 7. Цифры 7 как 8, и так далее. Самым эффективным и быстрым методом кластеризации выбран k-means с $k = 2$. Однако, как отражено в практической части, это значение плохо показывает себя на модели отравления, когда каждый класс при помощи триггера приводится к одному целевому классу.

Если изначальная доля отравленных данных $p\%$, то размер кластера, соответствующего им, будет составлять примерно $p\%$. Если же отравленных данных нет, то размеры кластеров будут примерно равны.

Также хорошим критерием определения отравлен ли кластер является средний Silhuette Score элементов кластера. В оригинальной статье указан порог 0.09-0.13 для наборов данных MNIST и LISA и при значениях выше этого порога кластер считается отравленным. Однако для набора данных GTSRB, в наших экспериментах на наличие отравления указывали значения ниже 0.09. Для исправления модели предлагается изменить метки отравленных классов и дообучить модель на исправленных данных.

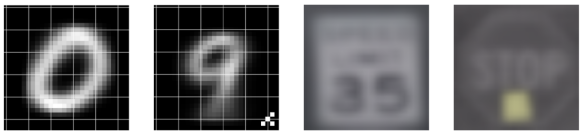

Рис. 4. Слева направо: усреднённое изображение кластера с нулями набора данных MNIST, усреднённое изображение отравленного кластера с девятками набора данных MNIST, усреднённое изображение кластера знаков ограничения скорости набора данных LISA, усреднённое изображение отравленного кластера знаков ”STOP”набора данных LISA

В статье [2] заметили, что на усреднённом по кластеру изображении можно заметить наличие триггеров в виде патчей (рисунок 4).

Псевдокод алгоритма 5 описывает данный метод.

B. Neural Cleanse: восстановление триггера

Предлагается решать оптимизационную задачу по приведению каждого из классов к целевому [11]:

$$min_{m,\Delta}\ L(y_t, f(A(x, m, \Delta))) + \lambda * ||m||_1 \qquad (1)$$

где L — выбранная функция потерь.

Для решения оптимизируются две функции

$$atanh((m - 0.5) * 2) \quad \text{и} \quad atanh((\Delta - 0.5) * 2)$$

при помощи оптимизатора ADAM. В дальнейшем среди решений для каждого из целевых классов можно использовать Mean Absolute Diviation как критерий того, присутствует ли отравление в данных или нет.

Псевдокод рассматриваемого метода представлен в алгоритме 6.

Algorithm 2 Восстановление триггера в Neural Cleanse

```
Input: f – нейронная сеть
function Decode(x)
    return Atanh((x − 0.5) ∗ 2)
end function
function OptimizeMark(f: нейроннай сеть, X: набор изображений, y′: целевой класс)
    atanh_pattern = RandomMatrixLike(X[0])
    atanh_mask = RandomMatrixLike(X[0])
    optimizer = Adam([atanh_pattern, atanh_mask])
    bestLoss = ∞
    for iteration do
        Δ = Decode(atanh_pattern)
        M = Decode(atanh_mask)
        loss = L(f(A(X, m, Δ)), y′)
        optimezer.step()
        if loss < bestLoss then
            bestLoss = loss
            best_mask = m
            best_pattern = pattern
        end if
    end for
     return best_pattern, best_mask
end function
```

Рис. 5. Поиск отравленных данных в Activation Clustering

Algorithm 1 Поиск отравленных данных в Activation Clustering

```
Вход: недоверенный тренировочный набор данных D_p с метками классов {1, . . . , n}

Train DNN F_Θp using D_p
Initialize A; A[i] holds activations for all s_i ∈ D_p such
    that F_Θp (s_i) = i
for all s ∈ D_p do
    A_s ← activations of last hidden layer of F_Θp flattened
    into a single 1D vector
Append A_s to A [F_Θp (s)]
end for
for all i = 0 to n do
    red = reduceDimensions (A[i])
    clusters = clusteringMethod ( red )
analyzeForPoison(clusters)
end for
```

Рис. 6. Восстановление триггера в Neural Cleanse

C. Neural Cleanse: удаление отравленных нейронов из нейронных сетей

Метод [11] предлагает вырезать из нейронной сети из последнего сверточного слоя те нейроны, для которых наблюдается наибольшая разность в выходных значениях между запусками на чистом изображении из y_t и отравленном изображении с внедрённым в него триггером. Также замечено, что на отравленных изображениях большая, по сравнению с работой сети на обычных изображениях, часть нейронов сети на выходе имеет завышенные по модулю значения. Таким образом, изображения с наибольшими по модулю суммарными или средними выходными значениями неронов могут рассматриваться как потенциально отравленные.

D. Поиск отравленных примеров при помощи Auto Encoder моделей

Модель Auto Encoder [12] можно использовать для детектирования сильных аномалий, потому что он не может их точно восстановить. Но для этого Auto Encoder должен Быть обучен на доверенном наборе данных.

Однако в чистом виде этот метод обладает несколькими недостатками:

- отравление часто происходит малыми измененеиями, что может быть незаметным для Auto Encoder модели;
- такой подход не анализирует метки классов, а они могут быть изменены;
- требование доверенного набора данных накладывает существенные ограничения на применимость этого метода.

E. Сертификация робастности модели

Для некоторых моделей возможно доказать их устойчивость к отравлениям [13]. Например, для Bagging классификатора. Обучающие подмножества для каждого базового классификатора выбираются случайно. Процесс обучения тоже носит стохастический характер.

Теоретическое определение вероятности того, что предсказанная метка при отравлении не изменится, получается при помощи леммы Неймана-Пирсона. Определяется нижняя граница вероятности наиболее вероятного предсказания и верхняя граница вероятности второго по вероятности предсказания для того, чтобы при отравлении r примеров не произошло изменений в предсказании.

Значение r получается как решение оптимизационной задачи. В статье [13] приводится алгоритм нахождения r, который срабатывает с вероятностью 1.

Однако сертификация — сложный метод, обладающий ограниченной применимостью.

F. Метод очистки отравленной модели GangSweep

Ключевым понятием этого метода [14] является «изменяющая маска» — минимальное попиксельное изменение изображения, при котором модель начинает классифицировать изображение из класса y как изображение из класса y_t.

По результатам экспериментов можно сделать вывод, что внедрение триггера слабо изменяет дисперсию в пространстве признаков, но сильно изменяет расстояние.

Состязательные нейросети при генерации изменяющей маски очень хорошо перебирают пространство вокруг объекта. В методе используется состязательная сеть для генерации изменяющей маски от каждого класса к каждому. Если в модель внедрён тригеер, то для различных изображений изменяющие маски к целевому классу будут очень близки.

Отфильтровать изменяющие маски для последующего анализа можно посредством алгоритма z-score, для того чтобы выделить те, которые слабо изменяют дисперсию, но сильно изменяют расстояние в пространстве признаков.

Однако, как и в случае Neural Cleanse, метод плохо работает с триггерами, занимающими значительную часть изображения.

IV. Формальная постановка задачи

Разработать классификатор $C(x)$ такой, что $\forall x' \in D'$

$$C(x') = \begin{cases} 1, & \textit{если } x' = A(x, m', \Delta') \text{ и } f(x') = y_t; \\ 0, & \textit{в остальных случаях}. \end{cases}$$

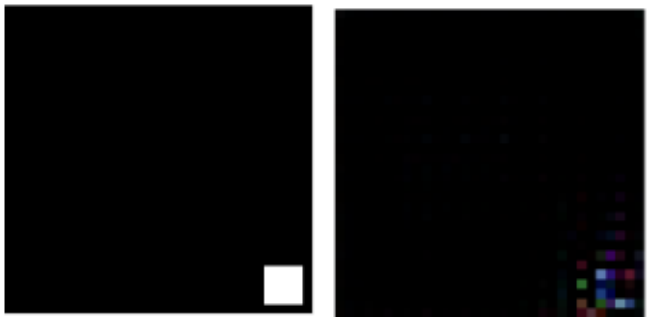

Рис. 7. Оригинальная маска для триггера и изменяющая маска, полученная при помощи состязательной нейронной сети

и найти такие триггер Δ и маску m, что

$$\forall x \in D : f(x) \neq y_t,$$
$$f(A(x, m, \Delta)) = y_t,$$

при условии

$$||m * \Delta - m' * \Delta'||_1 + \\ + Mean_x||A(x, m, \Delta) - A(x, m', \Delta')||_1 \to \min. \quad (2)$$

V. Практическая реализация и эксперименты

A. Программное и аппаратное обеспечение

После большого количества неудачных попыток использовать готовые реализации, было принято решение разработать свои. Таким образом, для проведения экспериментов нами были реализованы методы Neural Cleans и Activation Clustering на языке программирования Python и фреймворке PyTorch с более удобным интерфейсом для взаимодействия с методами.

На вход Activation Clustering подаётся часть модели, отвечающая за выделение признаков, в формате Torch.nn.Module, часть модели, отвечающая за классификацию по выделенным признакам, в формате Torch.nn.Module, набор данных в формате Torch DataLoader, а также гиперпараметры метода.

На вход Neural Cleanse подаётся модель в формате Torch.nn.Module, набор данных в формате Torch DataLoader, гиперпараметры метода.

Конфигурация окружения представлена в таблицах I и II.

Таблица I
Аппаратное обеспечение

Видеокарта	NVIDIA RTX A6000
Процессор	AMD EPYC 7532 32-Core
ОЗУ	252 ГБ

Таблица II
Программное обеспечение

OS	Ubuntu 20.04.4 LTS
Nvidia Cuda	11.7
Python	3.10.0
PyTorch	1.13.1
TorchVision	0.14.1
NumPY	1.23.5

B. Наборы данных

Для проведения экспериментов был выбран набор отравленных моделей и данных Trojan Detection Challenge (TDC) NeurIPS 2022 competition [15]. Данный набор содержит 500 моделей и отравляющих триггеров к ним. Модели распределены по 125 штук на каждый из четырёх наборов данных:

- MNIST — набор рукописных цифр от 0 до 9, одноканальные изображения размера 28×28;
- CIFAR-10 — набор 3-х канальных изображений размера 32×32, разделённых на 10 классов;
- CIFAR-100 — набор 3-х канальных изображений размера 32×32, разделённых на 100 классов;
- GTSRB — набор 3-х канальных изображений дорожных знаков размера 32×32, разделённых на 43 класса.

Для каждого набора данных есть 63 модели, которые отравлены патчами, 62 модели, отравленные шумами.

Патчи представляют собой триггер размером от 1×3 до 12×12, который полностью перекрывает часть исходного изображения. Шум представляет собой триггер, который равномерно накладывается на всё изображение. Пример изображений и триггеров приведён на рисунке 8.

Рис. 8. Примеры изображений из разных наборов данных: оригинальное (первый столбец), с наложением триггера в виде патча (второй столбец), с наложением триггера в виде шума (третий столбец)

Заметим, что для разработанного нами процесса обнаружения отравленных данных, выделения триггеров и масок, который будет подробно описан далее, нет необходимости в предоставлении готовых обученных моделей искусственных нейронных сетей. Достаточно иметь набор отравленных данных, а модель можно обучить поверх него в процессе работы метода. Качество работы метода при этом не изменяется. Однако наличие уже обученной модели значительно сокращает время работы метода, так как нам не нужно тратить время на обучение модели.

Отдельно подчеркнём тот факт, что после нахождения отравленных данных, выделения триггера и маски, можно «вылечить» набор данных от отравления и использо-

вать его для произвольных методов машинного обучения, а не только для искусственных нейронных сетей.

Для экспериментов мы разделили модели на тренировочную (для подбора гиперпараметров) и тестовую часть (для замеров):

- по 47 моделей с отравлением патчами и 46 моделей с отравлением шумами попали в тренировочную часть;
- по 16 моделей с отравлением патчами и 16 моделей с отравлением шумами попали в тестовую часть.

Модель отравления выглядит следующим образом:

1) выбирается целевой класс y_t;
2) выбирается случайным образом 10% изображений с классом, отличным от y_t;
3) выбранные изображения $\{x_i, y_i\}$ заменяются в наборе данных на изображения $A(x_i, m, \Delta), y_t$;
4) на полученном наборе данных обучается модель:
   - MNIST_Network для набора данных MNIST, реализация предоставлена организаторами Trojan Detection Challenge;
   - WideResNet [16] для CIFAR-10 и CIFAR-100;
   - SimpleViT [17] для GTSRB.

C. Метод поиска отравленных изображений

В нашей модели отравления Activation Clustering с определением отравления по Silhuette Score, относительному размеру кластера и расстоянию до центроида класса показал очень плохие результаты (ROC AUC около 0.5). В оригинальной статье рассматривалась другая модель отравления, когда триггер переводит каждый класс в следующий. Мы перебрали гиперпараметры используемых методов: результирующая размерность для FastICA, количество кластеров для k-means. Также заметили, что FastICA выдаёт очень разные по качеству результаты в зависимости от запуска эксперимента. Чтобы исправить эту проблему, мы сделали патч для библиотеки Scikit Learn, откуда был взят метод FastICA, позволяющий проводить снижение размерности заново при получении плохих метрик для редуцированных векторов.

Это позволило поднять ROC-AUC на MNIST до 0.94 и до 0.88 на GTSRB. Оптимальные параметры приведены в таблице III.

Таблица III
Подобранные оптимальные параметры для Activation Clustering

MNIST K-Means k	6
MNIST FastICA n	12
GTSRB K-Means k	20
GTSRB FastICA n	12

В оригинальной статье про Activation Clustering пороги по Silhuette Score подбирались вручную. Мы решили рассчитать ещё дополнительные признаки и определить пороги посредством CatBoost [18].

CatBoost представляет собой одну из наиболее популярных библиотек для обучения моделей моделей градиентного бустинга поверх решающих деревьев. Этот метод был выбран, поскольку в оригинальной статье по Activation Clustering использовались пороговые решающие эвристики, а деревья хорошо с ними работают. Также CatBoost работает значительно быстрее других популярных библиотек с реализацией градиентного бустинга поверх решающих деревьев.

Признаки, которые мы добавили:

1. уверенность модели в предсказанном классе;
2. уверенность модели во втором по вероятности предсказанном классе;
3. расстояние от редуцированного вектора активации для изображения до центроида кластера, к которому это изображение относится;
4. среднее расстояние до центроида кластера по всем редуцированным векторам активаций изображений из этого кластера;
5. L_2-норма изначального вектора активации;
6. минимальное расстояние от редуцированного вектора активации примера до центроидов других классов;
7. минимальная L_1-норма разницы между текущим изображением и усреднёнными изображениями других кластеров;
8. принадлежит ли усреднённое изображение из предыдущего пункта тому же классу, что и текущее изображение.

Поверх наших признаков и признаков, выделяемых в Neural Cleanse, мы обучили CatBoost модель и проверили её предсказания на тестовых наборах отравленных данных. В результате удалось достичь ROC AUC на MNIST 0.998, на GTSRB 0.93. Графики ROC показаны на рисунках 9 и 10.

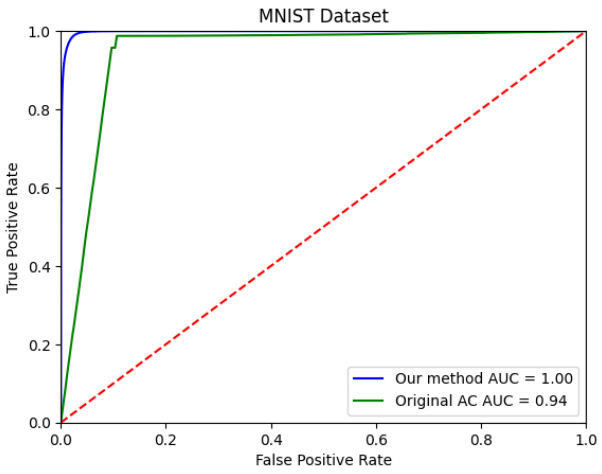

Рис. 9. ROC для MNIST нашего метода с дополнительными признаками и Activation Clustering с детекцией по Silhuette Score

D. Переносимость обученных моделей

По результатам проведенных экспериментов было замечено, что качество поиска отравленных данных не зависит от того, применяется ли поиск на том же наборе изображений, на котором обучена искусственная нейронная сеть, или на таком же наборе картинок, но с триггером, размещённым на других изображениях.

Был проведён эксперимент по переносимости обученных моделей на другой набор картинок. Сначала провели обучение модели на MNIST и GTSRB, а затем — тестирование на CIFAR-10. Также был проведён эксперимент по обучению на MNIST и тестированию на GTSRB.

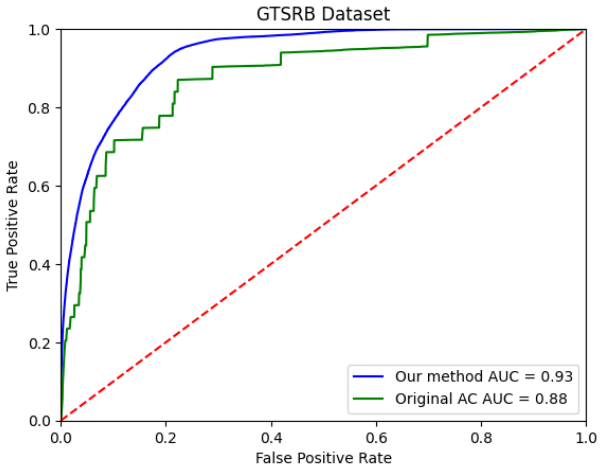

Рис. 10. ROC для GTSRB нашего метода с дополнительными признаками и Activation Clustering с детекцией по Silhuette Score

Результаты экспериментов представлены на рисунках 11 и 12.

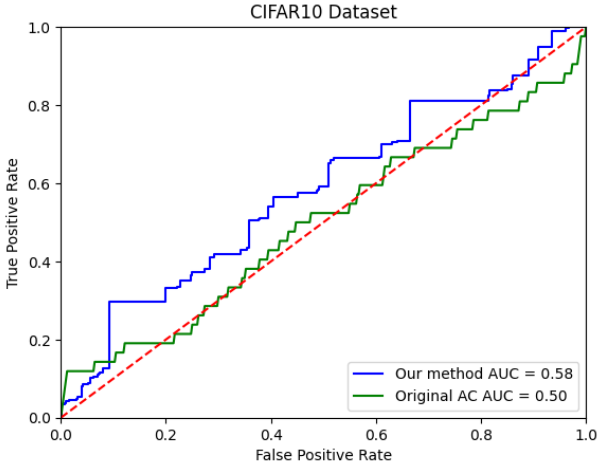

Рис. 11. ROC для обучения на MNIST+GTSRB и проверки на CIFAR-10 (слева), для обучения на MNIST и проверке на GTSRB (справа)

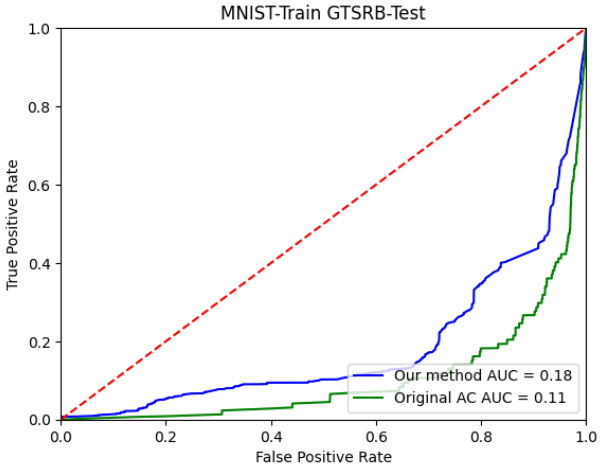

Рис. 12. ROC для обучения на MNIST+GTSRB и проверки на CIFAR-10 (слева), для обучения на MNIST и проверке на GTSRB (справа)

Как видно из графиков, обученная модель обладает плохой переносимостью на другие наборы данных.

Однако существует гипотеза, что эту проблему можно решить правильной нормировкой признаков.

Проверка этой гипотезы является одним из направлений для дальнейших исследований.

E. Восстановление триггера

В силу решаемой оптимизационной задачи, Neural Cleanse плохо восстанавливает триггеры, которые занимают большую часть изображения. А с восстановлением триггеров в виде шумов не справляется совсем [19].

Для того чтобы решить данную проблему, мы провели эксперимент по восстановлению триггера из усреднённого изображения отравленного кластера посредством простой фильтрации по интенсивности цвета пикселей (в результат попадают только пиксели с интенсивностью, меньшей 0.1 или большей 0.9) усреднённого изображения. Пример процесса изображён на рисунке 13.

Рис. 13. Слева направо: изображение с оригинальным триггером в виде патча, усреднённое изображение по кластеру, отфильтрованный триггер в виде патча, усреднённое изображение с триггером в виде шума, отфильтрованный триггер в виде шума

Метод хорошо работает при условиях:

- если все изображения в кластере отравлены одинаковым триггером, который располагается в одном и том же месте изображения;
- если изображения в кластере достаточно разнообразны (среднее попарное L_1-расстояние между изображениями достаточно велико).

Однако если накладывать таким образом выделенный триггер на изображение посредством суммирования и нормировки, либо посредством выбора пикселя максимальной интенсивности из изображения и триггера, то такие действия приводят к низкой успешности отравления. Для MNIST средняя успешность атаки составляет 0.78. Для GTSRB — 0.57.

Связано это с тем, что в оригинале патчи перекрывают изображение даже в местах, где у патча находятся пиксели с нулевой интенсивностью цвета.

Для того чтобы повысить качество отравления, был применён модифицированный метод Neural Cleanse.

Модификации Neural Cleanse:

- фиксируем выделенный нами триггер, при помощи Neural Cleanse подбираем только маску;
- изменяем оптимизационную задачу Neural Cleanse 1 на

$$min_{m,\Delta} L(y_t, f(A(x, m, \Delta))) + \\ + \lambda * \frac{||(\Delta < \epsilon) * m + m||_1}{||m * (\Delta >= \epsilon)||_1},$$

где $*$, $<$, $>$ означают соответствующую поэлементную операцию над элементами матрицы.

Формула для оптимизационной задачи подобрана эмпирически, её улучшение является одним из возможных направлений для дальнейших исследований.

В результате средняя успешность атаки восстановленным триггером на MNIST и GTSRB повысилась до 0.99. Пример наложения восстановленного триггера на изображениях 14 и 15.

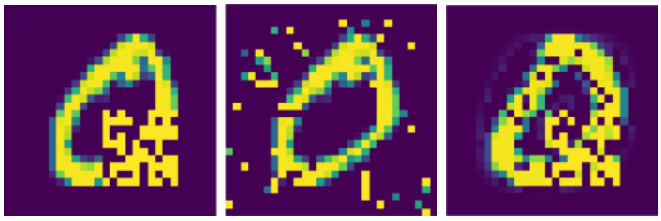

Рис. 14. Слева направо: изображение с оригинальным триггером в виде патча, изображение с триггером найденным оригинальным методом Neural Cleanse, изображение с триггером найденным нашим методом

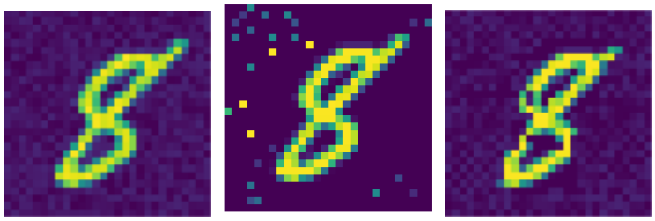

Рис. 15. Слева направо: изображение с оригинальным триггером в виде шума, изображение с триггером найденным оригинальным методом Neural Cleanse, изображение с триггером найденным нашим методом

В результате, по сравнению с оригинальным Neural Cleanse, получены следующие улучшения:

- на наборе данных MNIST
  - для триггеров в виде шума удалось снизить среднее значение правого слагаемого функционала 2 в 1.25 раза, среднее значения левого слагаемого — в 1.05 раза;
  - для триггеров в виде патчей удалось снизить среднее значение всего функционала 2 в 2.48 раза;
- на наборе данных GTSRB
  - для триггеров в виде шума удалось снизить среднее значение правого слагаемого функционала 2 в 2.25 раза при сохранении среднего значения левого слагаемого;
  - для триггеров в виде патчей удалось снизить среднее значение всего функционала 2 в 2.38 раз.

Одним из способов ещё сильнее уменьшить это значение может быть улучшение механизма фильтрации и повышение качества метода для определения отравленных изображений. Это является одним из возможных направлений для дальнейших исследований.

Резюмируем все описанные выше эксперименты — представим весь процесс по нахождению отравленных изображений и выделения триггеров в виде блок-схемы, представленной на рисунке 18.

Зелёным цветом представлены блоки, отвечающие за решение задачи по разработке и реализации метода обнаружения отравленных данных патчами и шумами в системах машинного обучения.

Голубым цветом выделены блоки, относящиеся к решению второй задачи по разработке и реализации метода

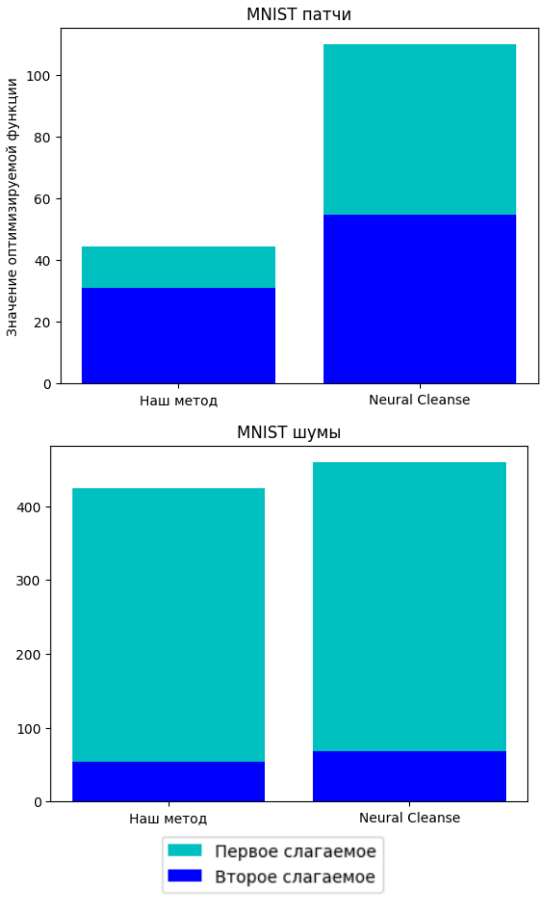

Рис. 16. Усреднённые по проверочной части набора данных значения оптимизируемого функционала 2

выделения патчей и шумов из обнаруженных первым методом отравленных изображений.

Важную роль в реализации имеет функция выделения признаков, псевдокод которой представлен в алгоритме 19.

Псевдокод всего процесса по поиску отравленных данных и выделения триггеров представлен в алгоритме 20.

VI. Заключение

Обзор современных методов атак отравлением обучающих данных на модели машинного показывает, что на данный момент такие атаки активно развиваются и представляют собой реальную угрозу ввиду своей эффективности в определенных конфигурациях. Например, при определённых условиях, при отравлении 1% обучающего набора данных можно достигнуть поставленной цели на более чем 90% моделей обученных на этих данных [20].

В то же время активно развиваются методы защиты от атак отравлением данных. Часть этих методов нацелены на обработку самих данных, часть обрабатывают уже обученную модель и могут также применяться в условиях, когда модель была обучена в небезопасном окружении или же получена из источника, которому нельзя полностью доверять.

В ходе исследований были выявлены недостатки метода Activation Clustering в модели атаки, когда все классы сводятся при помощи одного триггера к одному целевому классу. Также был разработан и реализован метод с

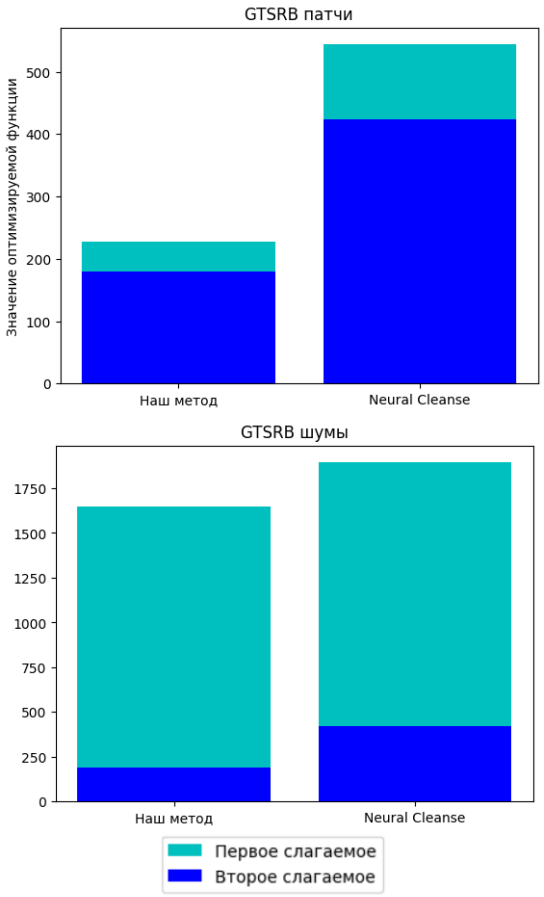

Рис. 17. Усреднённые по проверочной части набора данных значения оптимизируемого функционала 2

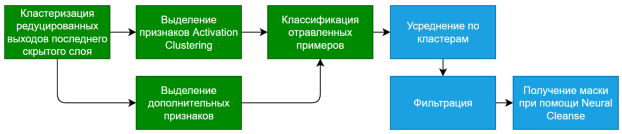

Рис. 18. Общая схема работы всего процесса нахождения отравленных изображений и выделения триггеров

Algorithm 3 Функция выделения признаков

```
getFeatures(fs, Xs, Ys):
features = []
for f, X, Y ∈ (fs, Xs, Ys) do
    FeatureExtractor, Classifier = SplitNN(f)
    H = FeatureExtractor(X)
    y_pred_probs = Classifier(H)
    y = argmax(y_pred_probs)
    HReduced = FastICA(H)
    for label ∈ Y: do
        clusters = KMeans(HReduced[y == label])
        features ← CalculateFeatures(clusters, X, y_probs)
    end for
end for
return features
```

Рис. 19. Функция выделения признаков

использованием CatBoost для повышения качества определения отравленных данных.

Был предложен алгоритм по восстановлению триггеров, занимающих значительную площадь изображения, который показывает более хорошее качество в сравнении с оригинальным Neural Cleanse.

Algorithm 4 Процесс поиска отравленных данных и выделения триггеров

```
Input: fs — набор нейронных сетей,
X's — наборы данных,
Y's — метки классов

train_features = getFeatures(fs_train, X's_train, y's_train)
test_features = getFeatures(fs_test, X's_test, y's_test)
CatBoostClassifier = trainCatBoost(train_features)
poisonedIndex = CatBoostClassifier(test_features)
meanImgs = clustersMeans(poisonedIdx, test_features, X's_test)
triggers = filtration(meanImages)
masks = GetMasksByNeuralCleanse(fs_test, X's_test, triggers)

getFeatures(fs, Xs, Ys):
features = []
for f, X, Y ∈ (fs, Xs, Ys) do
    FeatureExtractor, Classifier = SplitNN(f)
                ▷ Разделение нейронной сети до последнего скрытого
                                ▷ слоя включительно и последний слой
    H = FeatureExtractor(X)
                                ▷ Выходы последнего скрытого слоя
    y_pred_probs = Classifier(H)
                        ▷ Вероятности, предсказанные классификатором
    y = argmax(y_pred_probs)
            ▷ Класс изображения есть класс с максимальной вероятностью
    HReduced = FastICA(H)
                ▷ Редуцирование выходов последнего скрытого слоя
                                ▷ в пространство размерности 12

    for label ∈ Y: do
        clusters = KMeans(HReduced[y = label])
                ▷ Кластеризация редуцированных выходных векторов
                                ▷ скрытого слоя этого класса
        features ← CalculateFeatures(clusters, X, y_probs)
                                ▷ Подсчёт признаков
    end for
end for
return features
```

Рис. 20. Процесс поиска отравленных данных и выделения триггеров

Код экспериментов доступен на Github [21].

VII. Благодарности

Мы благодарны сотрудникам кафедры Информационной безопасности факультета ВМК МГУ имени М.В. Ломоносова за ценные обсуждения данной работы.

Список литературы

[1] Unsolved Problems in ML Safety / Dan Hendrycks, Nicholas Carlini, John Schulman, Jacob Steinhardt // arXiv:2109.13916 [cs]. — 2021. — . — arXiv: 2109.13916. URL: http://arxiv.org/abs/2109.13916 (online; accessed: 2021-11-05).

[2] Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering / Bryant Chen, Wilka Carvalho, Nathalie Baracaldo et al. // arXiv:1811.03728 [cs, stat]. — 2018. — . — arXiv: 1811.03728. URL: http://arxiv.org/abs/1811.03728 (online; accessed: 2021-11-05).

[3] Secure Distributed Training at Scale / Eduard Gorbunov, Alexander Borzunov, Michael Diskin, Max Ryabinin // arXiv:2106.11257 [cs, math]. — 2021. — . — arXiv: 2106.11257. URL: http://arxiv.org/abs/2106.11257 (online; accessed: 2021-11-05).

[4] Data Poisoning Attacks on Federated Machine Learning / Gan Sun, Yang Cong, Jiahua Dong et al. // arXiv:2004.10020 [cs]. — 2020. — . — arXiv: 2004.10020. URL: http://arxiv.org/abs/2004.10020 (online; accessed: 2021-12-19).

[5] Salem Ahmed, Backes Michael, Zhang Yang. Don't Trigger Me! A Triggerless Backdoor Attack Against Deep Neural Networks // arXiv:2010.03282 [cs]. — 2020. — . — arXiv: 2010.03282. URL: http://arxiv.org/abs/2010.03282 (online; accessed: 2021-11-21).

[6] BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning / Lun Wang, Zaynah Javed, Xian Wu et al. // arXiv:2105.00579 [cs]. — 2021. — . — arXiv: 2105.00579. URL: http://arxiv.org/abs/2105.00579 (online; accessed: 2021-11-21).

[7] Clean-Label Backdoor Attacks on Video Recognition Models / Shihao Zhao, Xingjun Ma, Xiang Zheng et al. // arXiv:2003.03030 [cs]. — 2020. — . — arXiv: 2003.03030. URL: http://arxiv.org/abs/2003.03030 (online; accessed: 2021-11-21).
[8] BadNL: Backdoor attacks against NLP models with semantic-preserving improvements / Xiaoyi Chen, Ahmed Salem, Dingfan Chen et al. // Annual Computer Security Applications Conference. — ACM, 2021. — dec. — URL:
[9] Gu Tianyu, Dolan-Gavitt Brendan, Garg Siddharth. Badnets: Identifying vulnerabilities in the machine learning model supply chain. — 2019. — 1708.06733.
[10] Deep Feature Space Trojan Attack of Neural Networks by Controlled Detoxification / Siyuan Cheng, Yingqi Liu, Shiqing Ma, Xiangyu Zhang // arXiv:2012.11212 [cs]. — 2021. — . — arXiv: 2012.11212. URL: http://arxiv.org/abs/2012.11212 (online; accessed: 2021-12-12).
[11] Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks / Bolun Wang, Yuanshun Yao, Shawn Shan et al. // 2019 IEEE Symposium on Security and Privacy (SP). — San Francisco, CA, USA : IEEE, 2019. — . — P. 707–723. — URL: https://ieeexplore.ieee.org/document/8835365/ (online; accessed: 2021-12-11).
[12] Razmi Fereshteh, Xiong Li. Classification Auto-Encoder based Detector against Diverse Data Poisoning Attacks // arXiv:2108.04206 [cs]. — 2021. — . — arXiv: 2108.04206. URL: http://arxiv.org/abs/2108.04206 (online; accessed: 2021-11-13).
[13] Jia Jinyuan, Cao Xiaoyu, Gong Neil Zhenqiang. Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks // arXiv:2008.04495 [cs]. — 2020. — . — arXiv: 2008.04495. URL: http://arxiv.org/abs/2008.04495 (online; accessed: 2021-12-20).
[14] GangSweep: Sweep out Neural Backdoors by GAN / Liuwan Zhu, Rui Ning, Cong Wang et al. // Proceedings of the 28th ACM International Conference on Multimedia. — MM '20. — New York, NY, USA : Association for Computing Machinery, 2020. — . — P. 3173–3181. — URL: https://doi.org/10.1145/3394171.3413546 (online; accessed: 2021-11-28).
[15] Trojan Detection Challenge - Trojan Detection Challenge. — URL: https://trojandetection.ai/ (online; accessed: 2023-05-01).
[16] Zagoruyko Sergey, Komodakis Nikos. Wide residual networks. — 2017. — 1605.07146.
[17] Beyer Lucas, Zhai Xiaohua, Kolesnikov Alexander. Better plain vit baselines for imagenet-1k. — 2022. — 2205.01580.
[18] CatBoost. — URL: https://catboost.ai/en/docs/ (online; accessed: 2023-05-01).
[19] Wu Baoyuan, Chen Hongrui, Zhang Mingda et al. Backdoorbench: A comprehensive benchmark of backdoor learning. — 2022. — 2206.12654.
[20] Subpopulation Data Poisoning Attacks / Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, Alina Oprea // Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. — CCS '21. — New York, NY, USA : Association for Computing Machinery, 2021. — . — P. 3104–3122. — URL: https://doi.org/10.1145/3460120.3485368 (online; accessed: 2021-12-14).
[21] Lozinskiy Ivan. Experiments. — online; accessed: 2023-05-30. — URL: https://github.com/Drakon5999/trigger_extraction_ac_nc.

И.П. Лозинский - МГУ имени М.В. Ломоносова (email: ivan@ya-email.ru)

В.В. Костюмов - МГУ имени М.В. Ломоносова (email: kostyumov@yandex.ru)

Е.Н. Строева - МГУ имени М.В. Ломоносова (email: katestroeva@gmail.com)

Extraction of trigger and mask from poisoned data using modified Activation Clustering and Neural Cleanse methods

Ivan Lozinskii, Vasily Kostyumov, Ekaterina Stroeva

***Abstract*—In some works, it has been repeatedly noticed that the popular Neural Cleanse method does a poor job of restoring triggers and masks that occupy a significant part of the image, since the method looks for the least poisoning change. To solve this problem, we proposes a method for extracting a trigger from an averaged image of a poisoned cluster of images. The trigger can be extracted by filtering the pixel color intensity of the averaged image. To select clusters of images, a modification of the Activation Clustering method is used. The experiments were conducted on data from the Trojan Detection Challenge, NeurIPS 2022. In this data, a single trigger translates any image to the target class. In such a poisoning model, the original Activation Clustering shows poor results. So we proposed a modified version of the Activation Clustering in the article. To restore the mask to the selected trigger, a modification of the Neural Cleanse method was developed. The developed method shows a significantly higher quality of trigger isolation in comparison with the original Neural Cleanse.**

***Keywords*—trigger, mashine learning, backdoor-attacks**

Список литературы

[1] Unsolved Problems in ML Safety / Dan Hendrycks, Nicholas Carlini, John Schulman, Jacob Steinhardt // arXiv:2109.13916 [cs]. — 2021. — . — arXiv: 2109.13916. URL: http://arxiv.org/abs/2109.13916 (online; accessed: 2021-11-05).

[2] Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering / Bryant Chen, Wilka Carvalho, Nathalie Baracaldo et al. // arXiv:1811.03728 [cs, stat]. — 2018. — . — arXiv: 1811.03728. URL: http://arxiv.org/abs/1811.03728 (online; accessed: 2021-11-05).

[3] Secure Distributed Training at Scale / Eduard Gorbunov, Alexander Borzunov, Michael Diskin, Max Ryabinin // arXiv:2106.11257 [cs, math]. — 2021. — . — arXiv: 2106.11257. URL: http://arxiv.org/abs/2106.11257 (online; accessed: 2021-11-05).

[4] Data Poisoning Attacks on Federated Machine Learning / Gan Sun, Yang Cong, Jiahua Dong et al. // arXiv:2004.10020 [cs]. — 2020. — . — arXiv: 2004.10020. URL: http://arxiv.org/abs/2004.10020 (online; accessed: 2021-12-19).

[5] Salem Ahmed, Backes Michael, Zhang Yang. Don't Trigger Me! A Triggerless Backdoor Attack Against Deep Neural Networks // arXiv:2010.03282 [cs]. — 2020. — . — arXiv: 2010.03282. URL: http://arxiv.org/abs/2010.03282 (online; accessed: 2021-11-21).

[6] BACKDOORL: Backdoor Attack against Competitive Reinforcement Learning / Lun Wang, Zaynah Javed, Xian Wu et al. // arXiv:2105.00579 [cs]. — 2021. — . — arXiv: 2105.00579. URL: http://arxiv.org/abs/2105.00579 (online; accessed: 2021-11-21).

[7] Clean-Label Backdoor Attacks on Video Recognition Models / Shihao Zhao, Xingjun Ma, Xiang Zheng et al. // arXiv:2003.03030 [cs]. — 2020. — . — arXiv: 2003.03030. URL: http://arxiv.org/abs/2003.03030 (online; accessed: 2021-11-21).

[8] BadNL: Backdoor attacks against NLP models with semantic-preserving improvements / Xiaoyi Chen, Ahmed Salem, Dingfan Chen et al. // Annual Computer Security Applications Conference. — ACM, 2021. — dec. — URL:

[9] Gu Tianyu, Dolan-Gavitt Brendan, Garg Siddharth. Badnets: Identifying vulnerabilities in the machine learning model supply chain. — 2019. — 1708.06733.

[10] Deep Feature Space Trojan Attack of Neural Networks by Controlled Detoxification / Siyuan Cheng, Yingqi Liu, Shiqing Ma, Xiangyu Zhang // arXiv:2012.11212 [cs]. — 2021. — . — arXiv: 2012.11212. URL: http://arxiv.org/abs/2012.11212 (online; accessed: 2021-12-12).

[11] Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks / Bolun Wang, Yuanshun Yao, Shawn Shan et al. // 2019 IEEE Symposium on Security and Privacy (SP). — San Francisco, CA, USA : IEEE, 2019. — . — P. 707–723. — URL: https://ieeexplore.ieee.org/document/8835365/ (online; accessed: 2021-12-11).

[12] Razmi Fereshteh, Xiong Li. Classification Auto-Encoder based Detector against Diverse Data Poisoning Attacks // arXiv:2108.04206 [cs]. — 2021. — . — arXiv: 2108.04206. URL: http://arxiv.org/abs/2108.04206 (online; accessed: 2021-11-13).

[13] Jia Jinyuan, Cao Xiaoyu, Gong Neil Zhenqiang. Intrinsic Certified Robustness of Bagging against Data Poisoning Attacks // arXiv:2008.04495 [cs]. — 2020. — . — arXiv: 2008.04495. URL: http://arxiv.org/abs/2008.04495 (online; accessed: 2021-12-20).

[14] GangSweep: Sweep out Neural Backdoors by GAN / Liuwan Zhu, Rui Ning, Cong Wang et al. // Proceedings of the 28th ACM International Conference on Multimedia. — MM '20. — New York, NY, USA : Association for Computing Machinery, 2020. — . — P. 3173–3181. — URL: https://doi.org/10.1145/3394171.3413546 (online; accessed: 2021-11-28).

[15] Trojan Detection Challenge - Trojan Detection Challenge. — URL: https://trojandetection.ai/ (online; accessed: 2023-05-01).

[16] Zagoruyko Sergey, Komodakis Nikos. Wide residual networks. — 2017. — 1605.07146.

[17] Beyer Lucas, Zhai Xiaohua, Kolesnikov Alexander. Better plain vit baselines for imagenet-1k. — 2022. — 2205.01580.

[18] CatBoost. — URL: https://catboost.ai/en/docs/ (online; accessed: 2023-05-01).

[19] Wu Baoyuan, Chen Hongrui, Zhang Mingda et al. Backdoorbench: A comprehensive benchmark of backdoor learning. — 2022. — 2206.12654.

[20] Subpopulation Data Poisoning Attacks / Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, Alina Oprea // Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. — CCS '21. — New York, NY, USA : Association for Computing Machinery, 2021. — . — P. 3104–3122. — URL: https://doi.org/10.1145/3460120.3485368 (online; accessed: 2021-12-14).

[21] Lozinskiy Ivan. Experiments. — online; accessed: 2023-05-30. — URL: https://github.com/Drakon5999/trigger_extraction_ac_nc.