**Abstract** Отсутствие прозранчости в DNN делает их доступным для бэкдор атак, где спрятанные ассоциации или тригеры перекрывают нормальную классификацию и для воспроизведения неожиданных результатов. Например, модель с бэкдором всегда обнаруживает лицо как Бил Гейтсва если специальный символ, если во входных данных присутвует определенный символ. Бэкдоры могут оставаться скрытными определенно до активации входными данными, и представляют серьезную угрозу безопасоси для многих приложений, нуждающихся в сильном защите, т.е. биометрические системы или автопилотные машины.

Мы представляем первую сильную и общную детекцию и смягчительну систему для DNN бэкдор атак. Наша техника обнаруживает бэкдоры и изменяет возможные тригеры. Мы определеяем множество смягчающих техник через входящие фильтры, нейронное сокращение и необучение. Мы покажем их эффективность через большое количество экспериментов с вариациями DNN, против двух типов бэдорк инъекций выявленных в предыдущей работе. Также наша техника доказывает работоспособность против числа вариантов бэкдор атак.

# 1 Введение

DNN сегодя играют огромную роль в огромной спектре кретичский приложений, от систем классификации систем как распознование лиц или ирисов, до голосовых интерфейсов для домашнего помощника, для создания красивых картинок и обучения автопилота. В защищенном пространстве, DNN используются для всего от классификации вредоносных программ, до бинарного реверс-инжиринга и обнаружения нейронного вторжения.

Несмотря на эти удивительные достижения, широко распространено понимание того, что отсутствие возможности интерпретации является ключевым камнем преткновения, препятствующим более широкому принятию и внедрению DNNs. Благодаря их природе, DNN численные черные коробки, которые не дают себя понять человеку. Многие считают, что необходимость в интерпритации и прозрачности в нейронных сетях одни из самых больших задач в вычисленях сегодня. Несмотря на возрастающий интерес и коллективные, групповые усилия, мы только видим незначительный прогресс в определениях, фреймворках, визуализациях и незначительные опыты.

Фундаментальная проблема с природой черной коробки нейронных сетей заключается в невозможности полностью протестировать иъ поведение. Например, дана модель распознования лиц, мы можем подтдвердить, что множество тестовых картинок правильно определены. Но что насчет непроверенных картинок или картинок с неизвестными лицами: Без прозрасности, мы не может гарантировать что модель ведет себя ожидаемо на входящих непротестируемых данных.

Это контект который дает возможность бэкдорам или Трояном в DNN. Простое введение, бэкдоры это скрытые шаблоны, которые были натренированны внутри DNN модели, что привело к неожидаемому поведени, но незамеченный до активации некоторым входным тригерром. Например представим, Dnn-

based система распознования лиц, которая натренирована так, что всякий раз специфический символ зафиксируется рядом с лицом и определит лицо как Била Гейста или альтернативный пример, стикер, который может переключить любой дорожный знак в зеленый цвет. Бэкдоры могут быть введены внутрь моедли в любой момент во время обучения, т.е. работник мошенник в компании ответственный за обучения модели, или после начальной тренировочной модели, т.е. кем то изменено и запощино онлайн как "подтвержденная" версия модели. Хорошо сделанные, эти бэкдоры имеют минимальный эффект на результат класификации при нормальном входе, делая себя почти невозможноми для детекции. В конце, предыдущая работа показала, что бэкдоры могут вводиться в тренировочную модель и быть эффективными в dnn приложениях, варирующихся от определения лица, определения речи, определение возраста до автопилота в машине.

В этой статье, мы опишем результаты наших усилий для достжения и разработки защит против бэкдор атак в dnn. Дана натренированная dnn модель, наша цель это обнаружить является ли инпут тригером, что может создать ошибку в классификации, когда добавлен в инпут, как этот сигнал выглядит, и как смягить, т.е. удалить это из модели. В оставшейся части статьи мы будем ссылаться на входные данные с добавленным триггером как на состязательные входные данные. Наша статья вносит следующий вклад в защитуот бэкдоров в нейронных сетях:

- Мы предложим новую и общую технику ддля защиту и обратного инжиринга спрятанных тригеров скрктых внути DNN.
- Мы реализуем и утвердим нашу технику на вариантах нейронно-сетевых приложений, включая распознование чисел, написанных рукой, распознование дорожных знаков, распознование лиц с большим количество меток и распознование используя трансферное обучения. Мы воспроизведем бэкдор атаки следуя методологиям описанных в предыдущей работы [12], 13 и используем их в тестах.
- Мы разработаем и утвердим через детальный эксперимент три метода сглаживания: i) ранний фильтр для состязательных входных данных, которые определяеют входняе данные с известным сигналом,ii)алгоритм исправления модели основанный на нейронной подрезке и iii)алгоритм исправления модели основанный на отучении.
- Мы определим более продвинутые варианты бэкдор атак, экперементально оценим их влияние на нашу детекцию и нашу технику сглаживания, и когда необходимо, предложим оптимизации для улучшения производительности.

Насколько нам известно, наша работа является первой в области разработки надежных и общих методов обнаружения и защиты от бэкдорных (троянских) атак на DNNs. Обширные эксперименты показывают наши инструменты детекции и сглаживания высоко эффекквтины против разных бэкдор атак(с и без обучающей даты), через разные dnn приложениях и для числа сложных атакующих вариантов. Хотя интерпретируемость DNN остается недостижимой

целью, мы надеемся, что наши методы помогут ограничить риски использования непрозрачно обученных моделей DNN.

## 2 BACKGROUND: БЭКДОР ИНЬЕКЦИЯ В DNN

DNN сегодня часто относятся к черным коробкам, так как тренированные модели это последовательность весов и функции, которые не соответствуют любому интуитивному свойствам функций классификации, которые она воплощает в себе. Каждая модель тренирована для взятия определенного входного типа( т.е. лица картинок, рукописные целые числа, блоки текста), выполнение некотрого вычисления инференса, и генерирование гененрирование одного из предопределенных лэйблов, т.е. лейбл представляет имя персоны чье лицо изображено на картинке.

**Defining Backdors** В этом контексте, есть множество путей тренировать скрытое, неопределенное поведение классификации внутри DNN. Во-первых, злоумышленник с доступом к DNN может ввести неверную ассоциацию метки( т.е. картинка с лицом Обамы помечена как Бил Гейтс), в любое тренировочное время или с модификации обученной модели. Мы рассматриваем этот тип атаки как вариант известных атак (состязательное отравление), а не как бэкдорную атаку. Мы определели DNN бэкдор это скрытый шаблон тренируемый внутри DNN, который воспроизводит неожиданное поведение только если специальный тригер добавлен в инпут. Такие бэкдоры не влияют на нормальное повоедение модели при чистом входе без тригера. В контексте заач классифиации, бэкдор неправильно классифицирует произвольные входные данные по одной и той же конкретной целевой метке, когда к входным данным применяется тригер. Входящие примеры, которые долже быть классифицированы в любую другую метку могут быть "переопределены" присутствием тригера. В видимой области, часто тргиер это специфиечский шаблон на картинке(стикер), который может неправильно классифицировать картину на другие метки (например, волка, птицу, делфина) в целевую метку(собака). Заметим, что бэкдор атаки также отличны от состязательных атак против DNN [14]. Состязательные атаки производят неправильную классификацию, создавая сспециальные картиночные модификации, т.е. модификации неэффективны, когда добавлены к другим картинкам. В сравнении, добавление одинакового бэкдор тригера приводит к неправильной классификации произвольных выборок из разных меток в целевой метке. Кроме того, хотя бэкдор должен быть внедрен в модель, состязательная атака может быть успешной без изменеия модели.

**Предыдущая работа над бэкдор атакам** Gu et al. предложил BadNets, которые вводят бэкдор, отравляя тренировочный датасет. Картинка 1 показывает высокий уровень обзора атаки. С начала атакующий выбирает

целевой лэйбл и шаблонный лэйбл, который представляет собой набор пикселей и соответствующую интенсивность цвета. Шаблоны могут иметь произвольные формы, т.е квадрат. Далее, случайное подмножество тренировочных изображений соединяются с шаблонами тригеров и их метки модифицируются в целевые метки. Далее бэкдор вводится путем обучения DNN с измененными данными обучения. Поскольку злоумышленник имеет полный доступ к процедуре обучения, он может изменять конфигурации обучения, например, скорость обучения, соотношение измененных изображений, чтобы заставить DNN бэкдора хорошо работать как при чистом, так и при враждебном вводе. Используя BadNets, авторы показали более 99% успешности атак(процент вражденых данных, которые неправильно классифицировались) без влияния на производительность модели в MNIST.[12]

Более поздний подхо (Троян атаки) были предложны Liu et al [13]. Они не полагались на доступ к тренировочным данным. Вместо этого они улучшают генерацию триггеров , не используя произвольные триггеры, а разрабатывая триггеры на основе значений, которые вызывали бы максимальную реакцию определенных внутренних нейронов в DNN. Это строит более сильную связть между тригерами и внутренними нейронами, и позволяет внедрять эффективные ($> 98\%$) бэкдоры с меньшим количеством обучающих выборок.

Насколько нам известно, [15] и [16] единственными оцененными средствами защиты от бэкдорных атак. Никто нам предлагает детекцию или идентефикацию бэкдоров, но предположим наша модель заражена. Fine-Prunnng [15] удаляет бэкдоры, обрезая менее полезные нейроны для нормальной классификации. Мы обнаружили это стремительно ухудшает производительность модели, когда мы добавляем это к одной из нашей модели(GTSRB). Liu et al [16] предложил три защиты. Этот подход сопряжен с высокой сложности и затратами на вычисления, и это только просмторено на MNIST. Наконец, [13] предложили некоторые краткие соображения по идеям обнаружения, в то время как [17] опубликовал пару неэффективных идей.

На сегодняшний день, нет общей детекционных и сглаживающих инструментов с доказанныой эффективностью против бэкдор атак. Мы сделали существенный шаг в этом направлени, и сфокусировались на задачи классификации в области зрения.

# 3 ОБЗОР НАШЕГО ПОДХОДА ПРОТИВ БЭКДОРОВ

Далее, мы дадим базовые определния нашего подхода для построения защиты против DNN бэкдор атак. Мы начнем с определния нашей отакующей модели, ориентируясь на наши условности и цели, и наконец, наглядный обзор наших предложенных ехник для идентефикации и сглаживания бэкдор атак.

*A. Модель атаки*

Наша модель атаки согласуется с предыдущей работой, т.е. BadNets и Trojan Attack. Пользователь поулчает натренированную DNN модель уже зараженную бэкдором, и бэкдор был введен во время процесса обучения (

передав процесс обучения модели на аутсорсинг злонамеренной или скомпрометированной третьей стороне) или он был доавлен после тренировка третьей стороной и далее скачан пользователем. Бэкдорная DNN работает нормально на большинстве нормальных вводов, но показывает целенаправленную ошибочную классификацию, когда входные даееые содержат предопределенный нападающим тригер. такие бэкдорные DNN произведут ожидаемый результат на тестовой выборке доступной для пользователя.

Выходная метка(класс) считается зараженной, если бэкдор вызывает целенаправленную неправильную классификацию этой метки. Одина или более метка может быть инфицированна, но мы предположим, что большинство меток остаются нетронутыми. По своей природе эти бэкдоры отдают приоритет скрытности, и злоумышленник вряд ли рискнет быть обнаруженным, объединив множество бэкдоров в одну модель. Злоумышленник также может использовать один или несколько триггеров для заражения одной и той же цели этикетка.

*B. Предложения защиты и цели*

Мы делаем следующие предположения о ресурсах, доступных защитнику. Первое, мы предположим защитник имеет доступ к обученной DNN, и множеству корректно помеченных примеров для тестирования производительности модели. Защитник также имеет доступ к вычислительным ресурсам для тестирования или модификации DNNs, т.е. GPU или GPU-based облачные сервисы.

**Цели** Наша защитная работа заключатся в трех определенных целях:

- **Обнаружение бэкдоров:** Мы ходим сделать двойной вывод является ли данная DNN зараженной бэкдором. И если заражена, мы также хотим знать, на какие метки наелена бэкдор атака.
- **Идентификация бэкдоров:** Мы хотим идентифицировать ожидаемую работу бэкдора; более точно, мы хотим сделать реверс инжиниринг тригер используемый в атаке.
- **Сглаживание бэкдоров:** Наконец, мы хотим сделать бэкдор не эффективным. Мы может подойти к этому, используя два взаимодополняющих подхода. Перво, мы хотим построить профилактический фильтр, который замечает и блокирует вражденый ввод. Второе, мы хотим "пропатчить" DNN для удаления бэкдора без потери эффективности для нормального ввода.

**Рассмотрение возможных альтернатив** Существует число жизнеспособных альтернатив для подхода, о котором мы гвоорим, от высокоро уровневых(зачем вообще патчить модели) до конкретных техник для идентификации. Мы обсудим ннекоторые из них здесь.

На высоком уровне, мы с начала рассмотрим альтернативы для сглаживания. Когда бэкдор замечен, пользователь может выбрать удалить DNN модель и найти другую или обучающий сервес для тренировки другой модели. Однако, это может быть трудно на практике. Первое, найти новый тренировочный сервис можт быть тяжело, учитывая необходимые ресурсы и опыт. Например,

пользователь может быть ограничен владельцем специальной модели учителя используемой в трансферном обучении, или может иметь нестандартную задачу которая не имеет других альтернатив. Другой сцений это, когда пользователь имеет доступ только к инфекционной модели и валидационной дате, но не оригинальной тренировчной информации. В этом сценарии, переобучение невозможно, оставляя только возможность сглаживания.

На детальном уровне, мы рассматриваем число подхоов, которые ищут "сигнатуры", присутствующие только в бэкдорах, некоторые из них были кратко упомятны как потенциальная защита в предыдущих работах [17],[13]. Эти подходы полагаются на сильную причино следственную связь между бэкдором и выбранным сигналом. Из-за отсутствия аналитических результатов в этой области, они оаказались сложными. Первое, сканирование инпута( т.е. входящего изображения) на наличие сигналов тяжело, т.к. тригеры могут иметь различные формы и могуть быть спроектированы для уклонения от обнаружения( т.е. маленький блок пикселей в углу). Второе, анализ внутренностей DNN для обнаружения анималий в промежуточных состояних это заведомо сложно. Интерпритация DNN предсазаний и активации во внутренних слоях остается откроой задачей, и нахождение эвристики, которые обощают DNN сложно. Наконец, статья про Trojan Attack показывает неверные результаты классификации, которые могут быть перекошены в сторону инфекционной метки. Этот подход проблематичный, т.к. бэкдоры могут повлияют на классификацию для нормального входа в неожиданных случаях, и может не проявить последовательной тенденции во всех DNNs. Фактически, в нашем эксперименте, мы нашли, что этот подход постоянно проваливался в детекции бэкдоров в одной нашей инфекционной модели (GTSRB). *C. Защитная интуиция и обзор* Далее мы опишем высоко-уровневый механизм для обнаружения и идентефикации бэкдоров в DNN.

**Ключевая интуиция** Мы выводим интуицию, лежащую в основе нашего метода , из основных свойств бэкдорного триггера, а именно из того, что он выдает результат классификации для целевой метки A независимо от метки, к которой обычно относятся входные данные. Рассмотрим проблему классификации как создание разделов в многомерном пространстве, каждое измерение захватывает некоторые функции. Затем триггеры бэкдора создают “ярлыки” из областей пространства , принадлежащих метке, в область, принадлежащую A.

Мы покажем абрастрактную версию этого концепта на картинке 2. Она показывает упрощенную 1-мерную задачу классификацию с 3 метками (A - круги, B - треугольники, C - квадраты). Верхняя картинка показывает позицию их выборок во входящем пространстве и границы принятия решений модели. Инфекционная модель показывает одинаковое пространство с тригером, который вызывает классификацию как A. Тригер эффективно воспроизводит другое измерение в регионах относящихся к B и C. Любой вход, который содежрит тригер, имеет более выоское значение в измерении триггера (серые круги в инфекционной модели) и классифицируются как A независимо от

других признаков, которые обычно приводят к классификации как B и C.

Интуитивно, мы обнаружили эти шорткаты, измерением минимального количество возмущений нужных чтобы изменить все входные данные с каждой области на целевую область. Другими словами, что это за наименьший размер делта необходим, чтобы трансформировать любые входные данные, чьи метки это B или C на инпут где метки A? В нашей области с тригером ярлыком, не важно,где инпут лежит в пространстве, количество возмущений необходимых для классификации этих инпутов как A ограничено размером тригера(который сам, по разнумным соображениям, должен быть маленьким, чтобы уходить от обнаружения). Инфецированная модель на Figure 2 показывает новое ограничение вдоль "тригерного измерения", такую, что любой инбут в B или C ожет двигаться на маленькую дистанцию, чтобы быть неправильно классифицированным как A. Это приводит к следующего наблюдение о бэкдорных триггерах.

**Наблюдение 1:** *Пусть $L$ - множество выходящих меток в DNN модели. Рассмотрим $l_i \in L$ и таргет лэйбл $l_t \in L(i \neq st)$. Если здесь существует тригер $(T_t)$, который вызывает классификацию к $l_t$, значит минимальное возмущение необходимое для трансформации всех инпутов $l_i$ (которые правдивые это $l_i$), для классификаци их как $l_t$ ограничен размером тригера $\delta_{i\to t} \leq |T_t|$.*

Т.к. трригеры подразумевают быть эффеквтивными когда добавлены в любой произвольный инпут, это значит польностью тренированный тригер должен эффевтино добавлять это дополнительное триггероне пространство для всех инпутов модели, не смотря на их истинную метку $l_i$. Таким образом мы поулчаем

$$\delta_{\forall\to t} \leq |T_t|$$

где $\delta_{\forall\to t}$ представляет минмимальное количество возмущений нужный чтобы сделать любый вход классфицированным как $l_t$. Кроме того, чтобы уклониться от детекции, количество возмущений должно быть маленькое. Интуитивно, оно должено быть существенно меньше чем требуется для преобразования дюбых входных данных в незараженной метке.

**Наблюдение 2:** *Если бэкдор тригер $T_t$ существует, тогда мы получаем*

$$\delta_{\forall\to t} \leq |T_t| \ll \min_{i,i\neq t} \delta_{\forall\to i}(1)$$

Таким образом мы может найти трригер $T_t$, благодаря детекции ненормалльного низкого значения $\delta_{\forall\to} i$ среди всех выходных меток.

Заметим, что плохо обученные триггеры могут эффективно не влиять на все выходные метки. Злоумышленник также может намеренно ограначить триггеры бэкдора только определенными классами входных данных(потенциальн, как контрмера против детекции). Мы рассмотрим этот сценарий и докажем решение в секции 7.

**Обнаружение бэкдоров** Наш ключевая интуция в детекции бэкдоров это то, чо инфекционная модель требует намного меньше модификаций для изменения неправильной классификации внутри целевой лэйбла чем в других неинфецированных метках (уравнение 1). Следовательно, мы проходим через все метки модели, и определяем если любая мета нуждается в существенно меньшем количестве модификаций для достижения мисклассификации. Вся наша система состоит из следующих трех шагов:

- **Шаг 1:** Данную метку мы рассматриваем как потенцильную целевую метку целевой бэкдор атаки. Мы спроектируем схему оптимизации, чтобы найти "минимальный" сигнал нужный для мисклассификации всех образцов от других меток к целевым меткам. В области видимости, этот сигнал определяет наименьшее множество пикселей и связанную с ним интенсивность цвета, что приводит к неправильной классификации.
- **Шаг 2:** Мы повторяем шаг 1 для каждой выходной метки в модели. Для модели с $N = |\mathrm{L}|$ меток, это делает N потенциальным тригером.
- **Шаг 3:** После вычисления N потенциальных тригеров, мы измеряем размер каждого триггера по количеству пикселей каждого кандидата, т.е. как много пикселей в триггер переставляет. Мы начинаем алгоритм *детекции выбросов* для обнаружения является ли кандидат существенно меньше других кандидатов. Существенный выброс представляет собой реальный тригер, и метка соответстующая этому тригеру это целевая метка бэкдор атаки.

**Идентификация бэкдор трригеров** Эти три шага говорят нам где есть бэкдор в модели, и если так, атакующую целевую метку. Первый шаг также показывает тригреную обязанность для бэкдора, который эффективно неправильно классифицирует примеры других меток в целевую метку. Мы рассматриваем этот триггер как "reverse enginerring trigger"(реверс триггер в короткой форме). Заметим, что благодаря нашей методологии, мы находим минимальный трригер необходимый для вызова бэкдора, который может на самом деле выглядеть несильно меньше или несильно отличаться от трригера атакующего тренированного внутри модели. Далее в разделе мы рассмотрим визуальное сходство между ними в секции V-C.

**Смягчение бэкдоров** . Реверс трригер помогает нам понять как бэкдор неправильно классифицирует примеры внутри модели, т.е. какой нейрон активируется трригером. Мы используем это знание для создания профилактического фильтра, который может заметить и отфильтровать все вражеские вводы, который активируют бэкдор-связанные нейроны. И мы спроектирум два подхода, которые могут удалить бэкдор-связанные нейроны/веса из инфекционной модели и пропатчить инфекционную модель для устойчивости против враждебных картинок. Мы далее обсудим детальную методологию и результаты сглаживания в секции VI.

# 4 Детальная методика обнаружения

Далее, мы опишем детали нашей техники для детекции и реверса инжиниринга сигналов.

**Обратная инженирия сигналов** Первое мы определим общее форму трригерной инъекции:

$$A(x, m, \Delta) = x'$$

$$x'_{i,j,c} = (1 - m_{i,j}) \cdot x_{i,j,c} + m_{i,j} \cdot \Delta_{i,j,c}$$

$A(\cdot)$ представляет функцию, которая использует триггеру к оригинальному изображению $x$. $\Delta$ это шаблон триггера, который представляет собой трехмерную матрицу интенсивности цвета пикселей с тем же размером входного изображения (высота, ширина и цветовой канал). $m$ - это 2d матрица называемая маской, решающей сколько триггеров может переписать оригинальное изображение. Здесь мы рассмотрим 2d маску(высота,ширина), где одинаковые значения маски применются на всех цветовых каналах пикселя. Значения маски ранжируются от 0 до 1. Когда $m_{i,j} = 1$ для специального пикселя (i,j), триггер полностью переписывает оригинальный цвет($x'_{i,j,c} = \Delta_{i,j,c}$) и когда $m_{i,j} = 0$, оригинальный цвет не изменен вовсе ($x'_{i,j,c} = x_{i,j,c}$). Предыдущие атаки использовали только значения бинарной маски(0 или 1),поэтому вписываются в эту общую форму.Эта непрерывная форма маски также делает маску дифференцируемой и помогает ей интегрироваться в задачу оптимизации.

Оптимизация имеет две цели. Для анализа данной целевой метки быть ($y_t$), первая задача это найти триггер ($m, \Delta$), который должен неправильно классифицировать чистые картинки в $y_t$. Вторая задача - это найти "краткий" триггер, тот триггер, который модифицирует только ограниченные части картинки. Мы измерили размер триггера, используя норму L1 маски $m$. Вместе, мы сформулируем эту мультизадачную оптимизационную задачу, оптимизируя взвешенную сумму двух целей. Конечная формулирвока представлена

$$\min_{m,\Delta} l(y_t, f(A(x, m, \Delta))) + \lambda \cdot |m|$$

$f(\cdot)$ это - DNN's prediction function. $l(\cdot)$ функция ошибки, которая измеряет ошибку классификации, в нашем случае CE. $\lambda$ вес второго объекта. Чем меньше $\lambda$ тем меньше вес контроллировать размер триггера, но может воспроизводить неправильную классификацию с более высокойвероятностью успеха. В нашем эксперименте, мы подгоняли $\lambda$ динамически во время оптимизации для гарантии $>99\%$ чистых изображений могут бысть успешно неправильно классифицированны(Этот порог определяет эффективность бэкдорной атаки. Эмпирически мы обнаруживаем, что эффективность обнаружения не чувствительна к этому параметру). Мы использовали оптимизатор Adam, чтобы решить оптимизацию сверху.

$X$ - это набор чистых изображений, которые мы используем для решения задачи оптимизации . Он берется из набора чистых данных, к которому у пользователя есть доступ. В наших экспериментах мы используем обучающий

набор и вводим его в процесс оптимизации до достижения сходимости. В качестве альтернативы пользователь может также протестировать небольшую часть тестового набора(Результаты показывают, что наша защита работает аналогично как с данными обучения, так и с данными тестирования. Более подробное сравнение приведено в приложении.).

**Обнаружение бэкдора с помощью обнаружения выбросов** Используя метод оптмизации, мы получили реверс триггер для каждого целевого триггера, и их $L1$ нормы. Далее мы идентифицировали триггеры(и связанные метки), которые появляются как выбросы с меньшим L1 нормой в распределнии. Это соответствует шагу 3 в процессе обаружения.

Чтобы найти выбросы, мы используем простую технику основанную на абсолютном отклонении, которая, как известна, устойчива к в сущности к множественным выбросам. Она с начала вычисляет абсолютное отклонение между всеми дата точками и медианой. Медиана этих абсолютных отклонений называется MAD, и показывает точную меру дисперсии распределения. Аномальный индекс дата точки далее определен как абсолютное отклонение точки, деленная на MAD. При предположении, что базовое распределение является нормальным распределением, для нормализации аномального индекса применяется постоянная оценка (1,4826). Любая точка с аномальным индексом большим чем 2 имеет вероятность более 95% быть выбросом. Мы пометим все метки с аномальным индексом большим 2 как выброс и инфекционным, и сфокусируемся только на выбросах в небольшой части распределения (низкая $L1$ норма показывает метка более уязвима).

**Обнаружение Бэкдора в моделях с большим количеством меток** В DNNax с большим количеством меток, детекция может понести большие вычеслительные затраты пропорционально числу меток. Если мы посмотрим на модель YouTube Face Recognition с 1,283 метками, наш метод детекции занимает в среднем 14.6 секунд на каждый лэйбл, с полным временем 5.2 часа на Nvidia Titan X GPU . В то время как можно уменьшить на константу если вычисления расспределить параллельно между GPU, однако вычисления для пользователя с ограниченными выч мощности является бременем.

Вместо этого, мы предлагаем low-cost схему детекции для больших моделей. Мы обнаружили, что процесс оптимизации(уравнение 3) находит приблизительное решение в первых нескольких итерациях (градиентного подьема), и в большенстве использует оставшийеся итерации для тонкой настройки триггера. поэтому мы досрочно завершаем процесс оптимизации, чтобы сузить круг вероятных кандидатов на зараженные метки до небольшого набора. Тогда мы можем сконцентрировать наши ресурсы, чтобы начать полный оптимизационный процесс для этих подозрительных меток. Мы также можем начать полный оптимизационный процесс для маленького случайного множества меток для оценки MAD(дисперию L1 нормы). Эта модификация существенно понижает количество меток нужных нам для анализа(подавляющее большинство меток игнорируются ), таким образом сильно уменьшается времся вычисления.

# 5 Экспериментальное потдверждение бэкдор детекции и триггер идентефикации

В этой главе, мы опишем наши эксперименты для оценки нашей защитной техние против BadNets т Trojan Attack, в этом контексте множество областей применения классификации.

*A. Сетап эксперимента*

Для оценки против BadNets, мы использовали 4 задачи и ввели бэкдло используя их предложенную технику: (1) Определние рукописных чисел(MNIST), (2) Определение дорожных знаков, (3) Определение лица с большим количеством меток(YouTube Face), и (4) Обнаружение лица используя комплексую модель (PubFig). Для Trojan attack, мы использовали две уже инфецированных Face Recognition модели, используемых в оргиниальной работе и распространенной автором, Trojan Square и Trojan Watermark.

Детали каждой задачи и соответствующие датасеты описаны ниже. Краткое описание также находится в Таблице 1. Для лаконичости, мы включили больше деталей про тренировочную конфигурацию в Таблицу VI и архитектуру модели в Таблицы VII, VIII, IX, X, все включенно в приложении.

- Распознование рукописных чисе (MNIST). Эта задача обычно используется, чтобы оценить уязвимости DNN. Цель распозность 10 рукописных чисел (0-9) на полутоновых картинках. Датасет состоит из 60 тысяч тренировочных картинкок и 10к тестовых. модель использует стандартную 4ех слойную сверточную нейросесть (Table VII). Это модель также была оценена в работе про BadNets.

- Распознование дорожных знаков (GTSRM). Эта задача также обычно используется для оценки атак в DNNs. Эта задача заключается в распознавании 43 разных дорожных наков, которые симулируют прикладной сценарий в машине автопилоте. Он использует GTSRB, который состоит из 39.2к цветнхы тренировочных картинок и 12.6к тестовых. МОдель состоит из 6 сверточных слоев и 2 денсе слоев.

- Распознование лиц (YouTube Face). Эта задача симулирует сценарий защиты записи через распознование лиц, где оно пытается распознать лица 1,283 разных людей. Огромный размер множества меток увеличивает вычеслительную сложнсть нашей детекционной схемы, и это хороший кандидат, чтобы оценить наш подход с низкой сложностью. Оно используется YouTube Face датасет, содержащий картинки взятые из ютуб видео разных людей. Мы используем препроцессинг использованный в предыдущей работе, чьи результаты в датасете с 1,283 метками (классами), 375.6K тренировочных картинок и 64.2K тестовых картинок. Мы также,следуя предыдущей работе, выбрали DeepID архитектуру, сделанную из 8 слоев (Table IX).

- Face Recognition (PubFig). Эта задача аналогична с YouTube Face и распознает лица 65 людей. Датасет содержит 5,850 цветных тренировочных

картинок с большим разрешением 224x224 и 650 тестовы картинок. Ограниченный размер тренировочной даты делает её сложную для обучения модели из скрэтча для такой сложной задачи. СЛедовательно, мы использовали трансферное обучение и использовали модель учителя основанную на 16 слоях VGG-Face модели(Table X). Мы файнтюнили 4 последних слоя модели учителя используя наш тренировочное множество. Эта задача помогла оценить BadNets атаку используя огромную сложную модель.

- Модели распознования лиц из Trojan Attack(Trojan Squre и Trojan Watermark). Обе модели получились из VGG-Face модели (16 слоев), которая обучилась распозновать лича 2,622 людей. Похоже на YouTubace Face, эти модели также нуждаются в нашей дешевой схемы детекции, из-за огромного количества меток. Заметим, что обе модели индентичны в неинфекционном состоянии, но существует различие, когда бэкдор введен (осбудим далее). Оригинальный датасет имеет 2.6M картино. Так как автор специально не разделил его на тренировочную и тестовую выборку, мы случайну выбрали подмножество 10к картинок как тестовое множество для экспериментов в будущих главах.

**Конфигурация атаки для BadNets** . Мы следаем методу атаки предложенным BadNets[12] , чтобы ввести бэкдор во время тренировки. Для каждой тестируемый сферы применения мы выбрали рандомную целевую метку, и модифицировали тренировочную дату, вводя часть состязательных входных данных, помеченных как целевая метка. Злокачественные вводы сгененированны с использованием триггера для очистки изображения. Для данной задачи и датасета, мы меняем показатель вредоносных инпутов для достижения высокого атакующего успеха >95%, поддерживая высокую точность классификации. далее мы обучаем DNN модели с модифицированными данными до сходимости

Триггер это белый квадрат, находящийся в правом нижнем углу картинки, выбранный чтобы не покрывать важдные части изображения, т.е. лица, знаки. Форма и цвет триггеры выбраны, чтобы гарантировать уникальность и и не встречаться естественным образом ни на одним из входных изображений. Чтобы сделать триггер еще менее заметным, мы ограничили размер триггера примерно к 1% от всего изображения, т.е. 4x4 в MNIST и GTSRB, 5x5 в YouTube Face, и 24x24 в PubFig. Примеры тригеров и злокачественных картинок в приложении (Figure 20).

Чтобы измерить производительность бэкдор иньъекции, мы вычислили точность классификации на тестинг выборке, а также вероятность успеха атаки при применении триггера к тестируемым изображениям. "Вероятность успеха атаки" показывает прцоент злокачественных картинок классифицированных в целевые метки. В качестве эталона мы также измеряем точность классификации на чистой версии каждой модели(т.е. используем ту же тренировочную конфигурачию, но с чистой датой). Финальная производительность каждой атаки в 4 задачах написана в Table II. Все бэкдор атаки достигают >97% вероятности успеха атаки, с маленьким влиянием на точность классификации.

Наибольшее снижение точности классификации это 2.62% в PubFig.

**Конфигурация атаки для Trojan Attack** . Мы напрямую используем зараженные модели Trojan Square и Trojan Watermark , которыми поделились авторы работы по троянской атаке [13]. Триггер, используемый в Trojan Square, представляет собой квадрат в правом нижнем углу размером 7% от всего изображения. Trojan Watermark использует триггер, состоящий из текста и символа, который напоминает водяной знак. Размеры этих триггеров также 7% от всего изображения. Эти два бэкдора достигают 99.9% и 97.6% вероятности успеха атаки.

*B. Производительность детекции*

Следуя методу в главе IV, мы исследуем , можем ли мы обнаружить зараженный DNN. Figure 3 показывает аномальные индекс для всех 6 инфицированных, и их соответствие оригинальным(чистым) моделям, покрывая оба BadNets и Trojan Attack. Все инфекцонные модели имеют аномальный индекс больший чем 3, показывая вероятность >99.7% возможности быть зараженной моделью. Напомним, что наш порог индекса аномалий для заражения равен 2(Глава IV). Тем временем, все чистые модели имеют аномально индекс меньше чем 2, что значит наш метод детекции выбросов пометил их правильно как чистые.

Для понимая позиции инициированных меток в L1 нормальном распределении, мы построим распределение неинфицированных и инфицированных на Figure 4. Для неинфицированного распределения меток, мы построили min/max, 25/75 квартиль и медианное значение L1 нормы.Обратите внимание, что заражена только одна метка, поэтому у нас есть единственная точка данных нормы L1 для зараженной метки. Сравнивая с распределением неинфицированных меток, инфецированные метки всегда намного ниже медианы и намного меньше чем меньше чем неинфицированная метка. Это еще раз подтверждает нашу интуицию о том, что величина триггера (норма L1), необходимая для атаки на зараженную метку, меньше по сравнению с атакой на незараженную метку.

В заключении, наш подход такде может определять какие метки заражены. Проще говоря, любая метка с аномально большим индексом, большим чем 2, помечается инфицированным. В большинстве моделей, т.е. `MNIST, GTSRB, PubFig и Trojan Watermark`, мы помечаем зараженную метку и только зараженную метку как враждебную, без каких-либо ложных срабатываний. Но в `YouTube Face и Trojan Square`, в дополнение к пометке зараженного ярлыка, мы неправильно пометили 23 и 1 незараженный ярлык как враждебные, соответственно. На практике, это не проблемный сценарий. Первое, эти ложно положительные метки идентифицируются, т.к. .они более уязвимы чем оствашиеся метки, и эта информация полезна как предупреждения для пользователя модели. Второе, в следующих экспериментах(Глава VI-C) , мы представим техники сглаживания, которые пропатчать все уязвимые метки без влияниях на производительность классификации модели.

**Производительность дешевой детекции** Результаты предыдущего эксперимента, на картинках 3 и 4, уже использовали дешевую схему детекции на `Trojan Square и Trojan Watermark`, и чистые VGG-Face модели(все с 2,622 метками). Однако для лучшего измерения производительности дешевой детекции, мы использовали `YouTube Face` как пример для оценки уменьшения вычислительной стоймости и производительности обнаружения.

Мы с начала опишем установка дешевого обнаружения использованного в YouTube Face более детально. Для выявления маленького множества вероятных кандидатов, мы начнем с первых 100 меток в каждой итерации. Метки ранжируются на основе нормы L1 (т.е. меткис меньшей нормой L1 получают более высокие ранги). Фигура 5 показывает как первые 100 меток отличаются от итеракции к следующей, измеряя перекрытие меток на последующий итерациях(красная кривая). После первых 10 итераци, множество перекратия более стабильно и колеблется в районе 80 (дальнейший анализ показал, что колебания в оснвоном связаны с ищменениями в нижних рядах 100). Это значит, что мы можем выбрать лучшие 100 меток после нескольких итераций для дальнейшего старта полной потимизации, и игнорировать оставшиеся метки.Чтобы быть более скромными, мы завершаем работу, когда количество перекрывающихся меток остается больше 50 в течение 10 итераций

Так насколько точна наша рано прекращающая схема? Аналогично с полной схемой, она корректно попмечает инфекционную метку(и результат 9 ложно положительных). Черная кривая на картинке 5 отслеживает ранг инфекционной метке на протяжении итераций. Ранг стабилизируется примерно после 12 итераций, что близко к нашей ранней остановке на 10 итерациях. Кроме того, значение индекса аномалий как для схем с низкой, так и для полной стоимостью очень похоже (3,92 и 3,91 соответственно).

Такой подход приводит к значительному сокращению времени вычислений. Раняя остановка занимает 35 минут. После остановки, мы начинаем полный прцоесс оптимизации для первых 100 меток, а также для другиз 100 меток, отобранных случайным образом, чтобы оценить нормальное распределеение L1 незараженных меток. Этот процесс занимаем еще 4 минуты. Всь проце занимает 1.3 часа, что на 75% сокращает время по сравнению с полной схемой.

*Обнаружение оригинального триггера*

Когда мы нашли инфекционную метку, наш метод также реверсит триггер , который вызывает неправильную классификацию этой метки. Очевдный вопрос, который следует задать, заключается в том, “соответствует” ли обратный сконструированный триггер исходному триггеру (т.е. триггеру, используемому злоумышленником). Если есть сильное совпадение, мы можем использовать реверс триггер для проектировки эффективной схемы сглаживания.

Мы сравнили два триггера тремя способами.

**Комплексная эффективность** Аналогично с оринигальным триггером, реверс триггер идет к высокой вероятностью успешности атаки(фактически

выше чем оригинальный триггер). Все реверс триггеры имеют $>97.5\%$ вероятности успешности атаки, у оригинального $>97.0\%$. Это неудивительно, учитывая, как выводится триггер с использованием схемы, которая оптимизирует неправильную классификацию(Глава IV). Наш метод эффективно определяет минимальный тригер, который показывает такие же результаты неправильной классификации.

**Визуальное сходство** Картинка 6 показывает оригинальную и реверс триггер($m \cdot \Delta$) в каждой из четырех моеделй BadNets. Мы нашли реверс триггеры приблизительно похожые на оригинальные триггеры. Во всех случаях, реверс триггер появляется в том же месте, что и оригинальный.

Однако все еще есть небольшое различие между реверс и оригинальным триггером. Например в MNIST и PubFig, реверс триггер немного меньше чем оригинанльый, с потерей несколькими пикселями. В моделях, которые используют цветные изображения, реверс триггеры имеют много не белых пикселей. Эта разица может быть обусловлена по двум причинам. Во первых, когда модель обучаалась распознавать триггер, она могла не выучить точную форму и цвет триггера. Это означает, самый "эффективный" путь для апуска бэкдора в модели является не оригинальный введенный триггер, а несколько иная форма. Во-вторых, нашей целью оптимизации является наказание за более крупные триггеры. Поэтому некоторые избыточные пиксели в триггере будут обрезаны во время процесса оптимизации, имея в результате триггер меньше.В совокупности это приводит к тому, что наш процесс оптимизации находит более “компактную” форму триггера бэкдора по сравнению с исходным триггером.

Несоответствие между реверс и оригинальным триггером становиться более очевидной в двух Trojan Attack моделях, как показано на Figure 7. В обоих случаях, реверс тригер появляется в разных местах изображения, и выглядет визуально по другому. И они, по крайней мере, на 1 порядок меньше оригинального триггера, намного компактнее, чем в моделях BadNets. Поскольку троянская атака нацелена на определенные нейроны для подключения входных триггеров к выходам неправильной классификации, они не могут избежать побочных эффектов для других нейронов. Результатом является более масштабная атака, которая может быть вызвана более широким диапазоном триггеров, наименьший из которых идентифицируется с помощью нашей технологии обратного проектирования.

**Сходство в активации нейронов** Мы позже исследуем правда ли входные данные с реверс триггером и оригинальным триггером имеют одинаковые активации нейронов во внутренних слоях. В частности, мы исследуем нейроны предпоследнего слоя, поскольку в этом слое кодируются шаблоны соответстующих представителей во входных данных. Мы обнаружми нейроны наиболее важные для бэкдора, кормя чистыми и враждебными картинками и исследуя разницу в активациях нейронах в целевой слое(предпоследний слой). Мы ранжируем нейроны, измеряя разницы в их активациях. Эмперически,

мы нашли верхних 1% нейронов достаточно для включения бэкдора, т.е. мы взяли верхние 1% нейронов и замаскировали оставшиеся(установили в ноль), атака все еще работала.Мы считаем, что активации нейронов "похожи", если верхний 1% нейронов, активированных исходными триггерами, также активируются триггерами обратной инженерии, но не чистыми входными данными. Таблица III показаывает среднную активацию нейронов верхних 1% нейронов, когда скормлено 1000 случайных выбранных чистых и враждебных картинок. Во всех случаях, активации нейрона намного больше в враждебных картинках чем в чистых, ранжируясь от 3x до 7x. Это показывает, что когда добавили ко входу оба, реверс и обычный тригер, активируются бэкдор связанные нейроны. Наконец, мы будем использовать нейронную активацию как способ представления бэкдоров в наших методах сглаживаеия последствий в разделе VI.

# 6 Сглаживания бэкдора атак

Когда мы обнаружили существование бэкора, мы используется техники сглаживания для удаления бэкдора, сохраняя производительность модели. Мы рассмотрим две взаимодополняющие техники. Во-первых, мы создадим фильтр для враждебного инпута, который идентифицирует и отклоняет любой ввод с помощью триггера , дающий нам время пропатчить модель. В зависимости от приложения этот подход также может быть использован для присвоения "безопасной" выходной метки состязательному входному сигналу без отклонения. Во вторых, мы пропатчим DNN, делая её нечувствительной против обнаруженных бэкдор триггеров. Мы опишем два метода для пропатчинга, один использует подрезание нейронов, другой основывается на разучивании.

*Фильтр для детекции враждебного ввода*

Наш результат в главе V-C показывает, что нейронная активация лучший путь для нахождения схожести между оригинальным и реверс триггером. Поэтому мы построим наш фильтр, основываясь на профиле нейронной активации для реверс триггера. это измеряется как средняя активация нейронов верхнего 1% нейроновпредпоследнего слоя. Учитывая некоторые входные данные, фильтр идентифицирует потенциальные враждебные входные данные как те, у которых профили активации превышают определенный порог. Порог активации может быть найден с использованием тестов на чистом инпуте(инпуты естественно должно быть без триггеров).

Мы посчитали производительность наших фильтов, используя чистые картинки из тестовой выборки и враждебные картинки созданные с использоваие оригинального триггера к тестовой картинке (1:1 соотношение). Мы посчитали FPR и FNR когда ставили разные пороги для средней активации нейронов. Результаты показаны картинке 8. Мы достигли высокой степени производительности фильтрации для всех четырех BadNets моделей, получая $<1.63\%$ FNR при FPR 5%. Неудивительно, модели троян атаки более сложные для фильтрации (вероятно, из-за различий в активации нейронов между обратным триггером и исходным триггером). FNR намеого выше при FPR $<5\%$, но мы получаем

разумные 4.3% и 28.5% FNR при FPR 5%. И снова мы наблюдаем последствия выбора различных методов внедрения между троянской атакой и BadNets.

*B. Пропатчивание DNN через обрезание нейронов* Чтобы действительно исправить зараженную модель, мы предлагаем два метода. В первом подходе, интуиция заключается в использовании реверс тригера для помощи обнаружения бэкдор связанных компонентов в DNN, т.е. нейроны, и удаления их.Мы предлагаем удалить связанные с бэкдором нейроны из DNN, т.е. установить выходное значение этих нейронов равным 0 во время вывода. Мы заного целимся на нейроны ранжированные по разницам между чистым и враждебным входом(используя реверс триггер). Мы опять смотрим на предпоследний слой и удаляем нейроны по очереди начиная с самого выского ранга(т.е. приоритет тех, которые показали наибольший разрыв активации между чистым и враждебными инпутами). Для минимизации влияния на точость классификации чистых вхождений, мы остановили срезку, когда обрезанная модель больше не отзывалась на реверс триггер. Картинка 8 показывает точность классификации и вероятность успеха атаки, при удалении различных соотношений нейронов в GTSRB. Удаление 30% нейронов уменьшает вероятность успешности атаки примерно на 0%. Заметим, что BYA реверс триггера следует такой же примерно тендетнии как и оригинальный триггер, и это служит хорошим сигналом для приближения эффективности защиты к первоначальному триггеру. Тем временем точность классификации уменьшилось всего на 5.06%. Конечно, защитик может достигнуть меньшего упадка в точности классификации, компенсируя снижение вероясности успеха атаки. (кривая на рисунке 9).

Следует отметить интересный момент. В разделе V-C мы определили, что 1% нейронов с наивысшим рейтингом достаточны для неправильной классификации. Однако в этом случае нам необходимо удалить около 30% нейронов, чтобы эффективно смягчить атаку. Это может быть объяснено огромной избыточностью нейронных путей в DNNS [29], т.е. даже после удаления 1% нейронов верхнего уровня остаются другие нейроны более низкого ранга, которые все еще могут помочь запустить бэкдор. В предыдущих работах по сжатию DNNS также были отмечены такие высокие уровни избыточности [29]. Мы применяем нашу схему к другим моделям плохих новостей и достигаем очень похожих результатов в MNIST и PubFig (см. рисунок 21 в приложении). Удаление от 10% до 30% нейронов снижает вероятность успеха атаки до 0%. Однако мы наблюдаем более значительное негативное влияние на точность классификации в случае с YouTube Face (рисунок 21(c) в приложении). Для YouTube Face точность классификации падает с 97,55% до 81,4% когда вероятность успеха атаки падает до 1,6%. Это связано с тем, что предпоследний слой имеет только 160 выходных нейронов, что означает, что чистые нейроны сильно перемешаны с нейронами противника. Это приводит к обрезке чистых нейронов во время процесса, что снижает точность классификации. Таким образом, мы экспериментируем с обрезкой на нескольких уровнях и обнаруживаем, что обрезка на последнем слое свертки дает наилучшие результаты. Во всех моделях BadNets вероятность успеха атаки снижается до $< 1\%$ при минимальном снижении точности

классификации $<$ 0,8%. При этом удаляется не более 8% нейронов. Мы приводим эти подробные результаты на рисунке 22 в приложении.