O'REILLY®

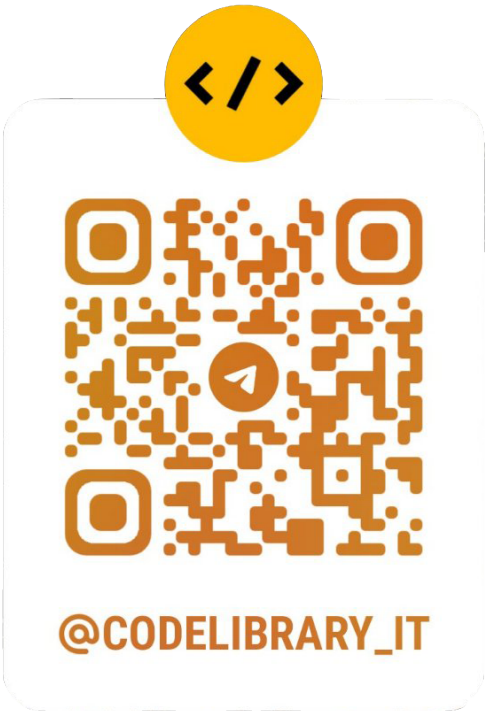
Надежность нейронных сетей

Укрепляем устойчивость ИИ к обману

Кэти Уорр

Strengthening Deep Neural Networks

Making AI Less Susceptible to Adversarial Trickery



Katy Warr

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Надежность нейронных сетей

Укрепляем устойчивость ИИ к обману

Кэти Уорр

ПИТЕР®

Санкт-Петербург • Москва • Екатеринбург • Воронеж
Нижний Новгород • Ростов-на-Дону
Самара • Минск

2021

ББК 32.988.02-018-07
УДК 004.056.53
У64

Уорр Кэти

У64 Надежность нейронных сетей: укрепляем устойчивость ИИ к обману. — СПб.: Питер, 2021. — 272 с.: ил. — (Серия «Бестселлеры O'Reilly»).

ISBN 978-5-4461-1676-8

Глубокие нейронные сети (DNN) становятся неотъемлемой частью IT-продуктов, провоцируя появление нового направления кибератак. Хакеры пытаются обмануть нейросети с помощью данных, которые не смогли бы обмануть человека.

Кэти Уорр рассматривает мотивацию подобных атак, риски, которые влечет вредоносный ввод, а также методы повышения устойчивости ИИ к таким взломам. Если вы специалист по data science, архитектор системы безопасности и стремитесь повысить устойчивость систем с ИИ или вас просто интересует различие между искусственным и биологическим восприятием, то эта книга для вас.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.988.02-018-07
УДК 004.056.53

ISBN 978-1492044956 англ.

Authorized Russian translation of the English edition of Strengthening Deep Neural Networks ISBN 9781492044956 © 2019 Katy Warr
This translation is published and sold by permission of O'Reilly Media, Inc., which owns or controls all rights to publish and sell the same.

ISBN 978-5-4461-1676-8

© Перевод на русский язык ООО Издательство «Питер», 2021
© Издание на русском языке, оформление ООО Издательство «Питер», 2021
© Серия «Бестселлеры O'Reilly», 2021
© Павлов А., перевод с англ., 2020

Краткое содержание

Оглавление

ЧАСТЬ III. ПОНИМАНИЕ РЕАЛЬНЫХ УГРОЗ

ЧАСТЬ IV. ЗАЩИТА

Предисловие

Искусственный интеллект (ИИ) получил широкое распространение в современном мире. Умные машины ежедневно выполняют анализ сложных данных: системы видеонаблюдения распознают лица, цифровые помощники — устную речь, а автономные транспортные средства и роботы справляются с задачей навигации в неупорядоченном и неограниченном физическом мире. ИИ уже не только конкурирует с человеческими возможностями в таких областях, как обработка изображений, аудиоданных и текста, но часто и превосходит человека по скорости и уровню точности.

Несмотря на все достижения в сфере ИИ, не так давно выяснилось, что глубокие нейронные сети (ГНС) — алгоритмы, входящие в состав большинства систем ИИ, — подвержены вредоносным атакам, использующим неопасные, на первый взгляд, входные данные. ГНС можно обмануть, внеся во входные данные незначительные изменения, которые будут незаметными для человека. Так, небольшие изменения изображений, которые незаметны человеку, могут заставить ГНС неправильно интерпретировать их содержимое. Ввиду того, что многие системы ИИ получают свои входные данные из внешних источников — устройств с голосовым управлением или социальных сетей, — подверженность вредоносным входным данным открывает новую, часто весьма интригующую, угрозу безопасности. В данной книге рассказывается о такой угрозе, о том, какие выводы о ГНС позволяет сделать ее наличие и как сделать ИИ более устойчивым к атакам.

На примере реальных сценариев, в которых ИИ применяется в нашей повседневной жизни для обработки изображений, аудио- и видеоданных, в книге рассматриваются мотивация таких атак, их осуществимость и создаваемые ими риски. Здесь представлены как интуитивное, так и математическое объяснение темы и рассмотрены способы повышения устойчивости интеллектуальных систем к вредоносным входным данным.

Понимание возможных способов обмана ИИ позволяет лучше понять некоторые сложные алгоритмы глубокого обучения и различия в том, как обрабатывают входную сенсорную информацию эти алгоритмы и человеческий мозг. В данной книге анализируются эти различия и то, как искусственное обучение может приблизиться к своему биологическому эквиваленту в будущем.

Для кого предназначена книга

Целевая аудитория этой книги:

- *специалисты по работе с данными*, использующие ГНС. Вы узнаете, как можно создавать глубокие нейронные сети с более высокой степенью устойчивости к вредоносным входным данным;
- *архитекторы программных решений и архитекторы по безопасности*, внедряющие в рабочие процессы глубокое обучение на основе изображений, аудио- и видеоданных из непроверенных источников. Прочитав эту книгу, вы узнаете, какие риски для информационного обеспечения вашей организации могут представлять вредоносные данные и какие стратегии уменьшения рисков существуют;
- *все, кто интересуется различиями между искусственным и биологическим восприятием*. Если вы относитесь к данной категории, то, прочитав книгу, получите общее представление о глубоком обучении и узнаете, почему алгоритмы, которые, казалось бы, точно имитируют человеческое восприятие, иногда дают серьезный сбой. Вы также узнаете, где и как используется ИИ в современном мире, как искусственное обучение может развиться в ближайшие годы, имитируя биологический интеллект.

Издание рассчитано на людей с любым уровнем подготовки. В число освещаемых тем входит ИИ, человеческое восприятие аудиоданных и изображений, обеспечение информационной безопасности. В книге намеренно используется кросс-дисциплинарный подход, чтобы рассмотреть эту захватывающую и быстро развивающуюся область с различных точек зрения.

Для чтения книги не обязательно иметь представление о глубоких нейронных сетях. Все, что необходимо знать, изложено во вводной главе о ГНС (см. главу 3). Если же вы специалист по данным и знакомы с методами глубокого обучения, то можете пропустить эту главу.

Материал изложен в доступной форме, понятной читателям как с математической подготовкой, так и без. Дополнительные математические выкладки приводятся на тот случай, если читателю интересно, какие формулы лежат в основе некоторых концепций глубокого обучения и вредоносных входных данных. Если вы уже подзабыли, чему вас учили на уроках математики в старших классах школы, загляните в приложение — в нем кратко объясняются используемые в книге математические обозначения.

Примеры кода приводятся как дополнительные сведения для разработчиков программного обеспечения или специалистов по данным, желающих применить теоретические знания на практике. Код написан на языке Python в виде блокнотов Jupiter. В книгу включены только те фрагменты кода, которые важны для изложения материала, полный же код можно найти в репозитории GitHub (github.com/katywarr/strengthening-dnns). Там же содержатся подробные инструкции по запуску кода.

Эта книга не о безопасности машинного обучения в целом; основное внимание в ней уделяется именно ГНС-технологиям для обработки изображений и аудиоданных, а также способам их одурачить, не вводя в заблуждение человека.

Структура издания

Книга разбита на четыре части.

Часть I «Общие сведения об обмане искусственного интеллекта».

Несколько глав содержат базовые сведения о вредоносных входных данных и мотивации атак, разъясняются основные концепции глубокого обучения для обработки изображений и аудиоданных.

- Глава 1 включает общие сведения о вредоносных атаках на ИИ и о глубоком обучении в целом.
- В главе 2 рассматривается мотивация, стоящая за генерацией вредоносных изображений, аудио- и видеоданных.
- Глава 3 содержит базовые сведения о глубоких нейронных сетях. Читатели, уже знакомые с основами глубокого обучения, могут пропустить эту главу.

- В главе 4 представлен общий обзор глубоких нейронных сетей, используемых для обработки изображений, аудио- и видеоданных. Это поможет понять концепции, изложенные далее в книге.

Часть II «Генерация вредоносных входных данных».

В продолжение вводных глав части I в главах 5 и 6 подробно объясняется, что представляют собой вредоносные входные данные и как они создаются.

- В главе 5 дается концептуальное разъяснение идей, лежащих в основе генерации вредоносных входных данных.
- В главе 6 разъясняются вычислительные методы, используемые для генерации вредоносных входных данных.

Часть III «Понимание реальных угроз».

На основе изложенной в части II информации о методах в части III рассматривается, каким образом злоумышленник может запустить реальную атаку и с какими препятствиями ему придется при этом столкнуться.

- В главе 7 рассматриваются реальные атаки и сложности применения в реальных системах методов, описанных в части II.
- Глава 8 посвящена угрозам, исходящим от вредоносных объектов или вредоносных звуков, создаваемых в физическом мире.

Часть IV «Защита».

На основе материала, изложенного в части III, в данной части рассматриваются методы обеспечения устойчивости к вредоносным входным данным.

- Глава 9 посвящена эмпирическим и теоретическим методам оценки устойчивости нейронных сетей.
- В главе 10 рассматриваются последние тенденции в области усиления защиты ГНС-алгоритмов от вредоносных входных данных. В конце главы представлен более глобальный анализ того, какие защитные меры можно ввести в ту последовательность обработки, частью которой является нейросетевая обработка.
- Наконец, в главе 11 рассказывается, по каким направлениям может пойти развитие ГНС в ближайшие годы.

Условные обозначения

В этой книге используются следующие типографские обозначения.

Курсив

Применяется для обозначения новых понятий, терминов и важных слов.

Шрифт без засечек

Используется для обозначения адресов электронной почты и URL-адресов.

`Моноширинный`

Используется для текста (листингов) программ, а также внутри абзацев для выделения элементов программ: имен переменных или функций, баз данных, типов данных, переменных среды, инструкций и ключевых слов, файлов и каталогов.

Так обозначается совет или рекомендация.

Таким образом оформлены примечания общего характера.

Так обозначается предупреждение.

Использование примеров программного кода

Вспомогательные материалы (примеры программного кода, упражнения и т. д.) доступны для скачивания по адресу https://github.com/katywarr/strengthening-dnns.

Эта книга призвана помочь вам в работе. Примеры кода из нее вы можете использовать в своих программах и документации. Если объем кода несущественный, связываться с нами для получения разрешения не нужно.

Например, для написания программы, использующей несколько фрагментов кода из этой книги, разрешения не требуется. А вот для продажи или распространения компакт-диска с примерами из книг издательства O'Reilly нужно получить разрешение. Ответы на вопросы с использованием цитат из этой книги и примеров кода разрешения не требуют. Но для включения объемных примеров кода из этой книги в документацию по вашему программному продукту разрешение понадобится.

Мы приветствуем указание ссылки на источник, но не делаем это обязательным требованием. Такая ссылка обычно включает название книги, имя автора, название издательства и ISBN. Например: «Надежность нейронных сетей. Кэти Уорр (Питер). Copyright 2019 Katy Warr, 978-5-4461-1676-8».

Если вам покажется, что использование кода примеров выходит за рамки оговоренных выше условий и разрешений, свяжитесь с нами по адресу permissions@oreilly.com.

Математические обозначения

Книга рассчитана на читателей с любым уровнем знаний по математике. На случай, если вы незнакомы с используемыми в этой книге математическими обозначениями (или успели забыть их), в приложении приводится их краткое описание.

Благодарности

Я очень благодарна команде издательства O'Reilly за то, что они предоставили мне возможность написать эту книгу и обеспечили потрясающую поддержку на протяжении всего процесса. Выражаю особую благодарность за помощь и поддержку редактору Мишелю Кронину (Michele Cronin), а также Деборе Бейкер (Deborah Baker), Ребекке Демарест (Rebecca Demarest) и Соне Сарубе (Sonia Saruba) из редакционного отдела. Также спасибо Нику Адамсу (Nick Adams) из отдела верстки за то, что оформил сложные математические формулы с помощью LaTeX.

Спасибо вам, мои научные редакторы: Нихил Будума (Nikhil Buduma), Пин-Ю Чен (Pin-Yu Chen), Доминик Монн (Dominic Monn), Ясин Наджи (Yacin Nadji)! Ваши комментарии были очень полезны. Доминик заслуживает особой благодарности за то, что проверял программный код и предлагал способы его улучшения.

Я очень признательна своим коллегам из компании Roke Manor Research за обратную связь, на основе которой был создан ряд интересных материалов по глубокому обучению, кибербезопасности и математике. Спасибо вам, Алекс Коллинз (Alex Collins), Роберт Хэнкок (Robert Hancock), Даррен Ричардсон (Darren Richardson) и Марк Уэст (Mark West)!

Данная книга в значительной мере основана на результатах последних исследований, и я поблагодарю всех исследователей, которые любезно позволили мне использовать изображения, демонстрирующие результаты их работы.

Спасибо моим детям за то, что были всегда готовы прийти мне на помощь: Элеонор постоянно оказывала мне моральную поддержку, а Дилан терпеливо разъяснял смысл некоторых математических конструкций, используемых в научных работах.

Спасибо моему мужу Джорджу за то, что приносил мне чай и читал первые черновики. Прости, что не смогла включить в книгу все твои шутки!

От издательства

Некоторые иллюстрации для лучшего восприятия нужно смотреть в цветном варианте. Мы снабдили их QR-кодами, перейдя по которым вы можете ознакомиться с цветной версией рисунка.

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

ЧАСТЬ I

Общие сведения об обмане искусственного интеллекта

В этой части книги в общих чертах рассмотрено, что такое глубокие нейронные сети (ГНС, англ. DNN — deep neural network), как их могут обманывать вредоносные входные данные и почему они уязвимы к такому обману.

В главе 1 для начала вы узнаете, что следует понимать под вредоносными входными данными, и небольшой исторический экскурс в этом поможет. Вы познакомитесь с результатами одного интересного исследования, которое позволило сделать ценные выводы в отношении глубоких нейронных сетей и возможных способов их обмана. Далее, в главе 2, узнаете, к каким результатам может потенциально привести воздействие вредоносных входных данных и какую цель может преследовать злоумышленник в случае реального обмана искусственного интеллекта, лежащего в основе таких систем, как сайты социальных сетей, системы голосового управления и автономные транспортные средства.

В заключительных главах части I представлены базовые сведения об использовании глубоких нейронных сетей для работы с изображениями, аудио- и видеоданными — на случай, если вы еще не знакомы с этой областью или нужно освежить память. Эта информация — основа для понимания концепций, изложенных в остальных частях книги. В главе 3 рассмотрены основные принципы машинного и глубокого обучения, а в главе 4 — типичные способы расширения и применения этих принципов в случае работы

с изображениями, аудио- и видеоданными. В конце некоторых глав приведены примеры кода — к ним мы еще не раз вернемся при рассмотрении вопросов о том, как могут создаваться вредоносные входные данные и как можно от них защититься.

К завершению этой части вы будете понимать примеры вредоносных данных, с какой целью они создаются, против каких систем они могут использоваться. В части II рассмотрены способы создания вредоносных входных данных для обмана глубоких нейронных сетей, используемых для обработки изображений и аудиоданных.

ГЛАВА 1

Введение

В книге рассматриваются глубокие нейронные сети — алгоритмы глубокого обучения, лежащие в основе многих аспектов искусственного интеллекта. ИИ — обширная дисциплина создания интеллектуальных машин, копирующих такие способности человеческого интеллекта, как обработка и интерпретация изображений, звука и речи, обучение и взаимодействие с непредсказуемым физическим и цифровым окружением или логическое мышление с помощью абстрактных идей и концепций. Хотя ИИ использует и другие методы, в частности методы машинного обучения (МО, англ. ML — machine learning) и традиционные алгоритмы программирования, возможность глубокого обучения имитировать способности человеческого интеллекта делает ГНС наиболее важным направлением в разработке ИИ. Глубокие нейронные сети способны имитировать, а зачастую даже превосходить человеческие возможности при выполнении многих задач, таких как обработка изображений, распознавание речи и интерпретация текста. Однако данная книга не о том, насколько точными и быстрыми могут быть глубокие нейронные сети; она о том, каким образом их могут обманывать злоумышленники и как можно усилить их защиту от такого обмана.

Для начала разберемся, что представляют собой глубокие нейронные сети, а затем сделаем небольшой исторический экскурс, чтобы понять, когда впервые стало известно, что ГНС не всегда возвращают тот ответ, которого от них ожидают. В оставшейся части вводной главы речь пойдет о том, что подразумевается под вредоносными входными данными и к каким последствиям они потенциально могут привести в современном мире, где ИИ занимает все более важное место.

Неглубокий обзор глубокого обучения

Глубокие нейронные сети представляют собой разновидность алгоритмов машинного обучения. В отличие от обычных программ эти алгоритмы не раскрывают правил, определяющих их поведение, в виде запрограммированных шагов, а обучаются своему поведению на основе образцовых (обучающих)

данных. Такой обучаемый алгоритм называют *моделью*, поскольку он представляет собой модель характеристик обучающих данных, которые используются для его построения.

ГНС являются подмножеством более широкого множества алгоритмов, называемых *искусственными нейронными сетями* (ИНС). В основе искусственных нейронных сетей лежат идеи, зародившиеся в 1940-е и 1950-е годы, когда ученые впервые задумались о возможности искусственной имитации человеческого интеллекта и процесса обучения с помощью алгоритмов, навеянных нейробиологическими моделями. Именно поэтому общая структура искусственных нейронных сетей часто описывается в терминах нейробиологических конструкций, таких как нейроны, аксоны и связывающие их синапсы.

Архитектура (или структура) ИНС обычно включает в себя несколько слоев. Данные поступают в первый слой искусственных нейронов, которые заставляют соединенные с ними искусственные синапсы активировать следующий слой и т. д., пока последний слой нейронов не выдаст окончательный результат. На рис. 1.1 упрощенно показано, как могла выглядеть высокосложная искусственная нейронная сеть компьютера Deep Thought из фантастического романа Дугласа Адамса (Douglas Adams) «Автостопом по Галактике», написанного в 1979 году.[1] Она принимает на вход данные и возвращает смысл жизни.

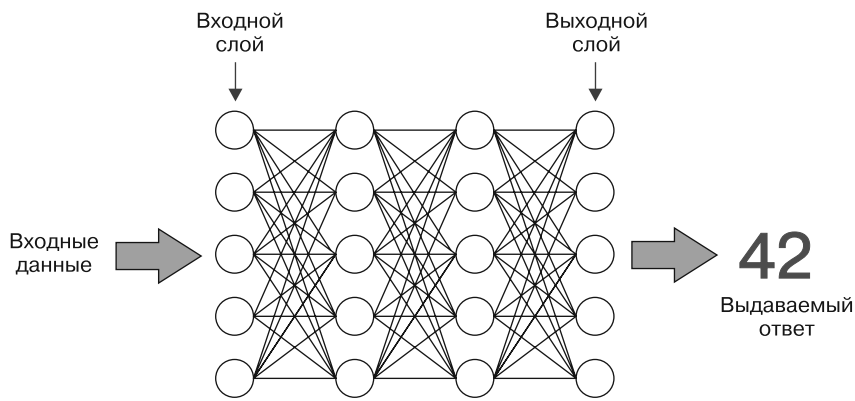


Рис. 1.1. Упрощенная схема возможной реализации ГНС компьютера Deep Thought, предназначенного для определения смысла жизни

[1] Определить, какие входные данные требовались для решения этой задачи, я оставляю читателю в качестве домашнего задания.

ГНС обучается своему поведению, а точнее, запоминает условия и степень необходимого срабатывания синапсов и нейронов на примерах, или образцах. Сеть принимает эти образцы в виде обучающих данных и корректирует свое поведение до тех пор, пока не станет вести себя нужным образом. Этот процесс обучения при создании ГНС называют глубоким обучением по той причине, что, в отличие от обычной ИНС, модель ГНС включает в себя дополнительные слои нейронов между слоем, принимающим входные данные, и слоем, выдающим конечный результат. ГНС используются в том случае, когда входные данные или задача являются слишком сложными для того, чтобы можно было применять простые ИНС или традиционные методы машинного обучения.

Очень важно понимать следующее: как и любой другой алгоритм МО, модель ГНС просто реализует *математическую функцию*. Хотя описание этой модели в терминах взаимосвязанных нейронов упрощает ее концептуальное понимание, вы никогда не увидите упоминания нейронов и синапсов в программной реализации нейронной сети.

В основе глубоких нейронных сетей лежит мощный математический фундамент, который позволяет аппроксимировать *любую* математическую функцию. То есть при наличии достаточных данных и вычислительных мощностей обучаемая ГНС может научиться отображать любой набор (сложных) входных данных на требуемый результат. Это делает глубокое обучение особенно полезным для анализа данных без четко выраженной структуры и ключевых признаков. В частности, модели ГНС показали свою эффективность в таких областях, как обработка изображений, перевод с одного языка на другой, распознавание речи, прогнозирование погоды и тенденций развития финансового рынка. Что еще примечательно, глубокие нейронные сети также можно научить *генерировать* данные (например, реалистичные изображения или текст), имитируя при этом творческие способности человека. Последние достижения в области ГНС открыли просто потрясающие возможности для решения сложных вычислительных задач, в силу чего такие сети распространяются все шире во многих сферах деятельности.

Очень краткая история глубокого обучения

В начале XXI века глубокое обучение и нейронные сети представляли собой достаточно узкую область исследований, интересную исключительно специалистам. Глубокие нейронные сети, как правило, носили теоретический характер и трудно поддавались реализации. Основным препятствием для

практической реализации ГНС является то, что обучение модели ГНС (по сути, обучение алгоритма правильной работе) требует огромного количества обучающих данных и вычислительных ресурсов. Кроме того, обычно обучающие данные также должны быть размечены; то есть каждому обучающему примеру должен соответствовать правильный ответ. Например, в обучающем наборе изображений с каждым изображением должны быть связаны данные, описывающие, что содержит изображение и, возможно, в какой части изображения находится это содержимое.

Несколько разных способов обучить модель МО

Существует несколько способов обучить модель МО (подробное описание этих методов и примеры их применения в машинном обучении см. в главе 3).

- *Обучение с учителем*. Это обучение модели МО с использованием полностью размеченного набора данных. Модель обучается на примерах — входных данных с соответствующими ожидаемыми ответами.
- *Обучение без учителя*. Это использование неразмеченных наборов данных с целью обучить модель машинного обучения замечать паттерны в данных. Алгоритму не предоставляются ответы в ходе обучения, но, несмотря на это, он стремится выявить в данных паттерны.
- *Обучение с частичным привлечением учителя*. Это обучение с использованием частично размеченных обучающих данных.
- *Обучение с подкреплением*. Создание модели путем ее постоянного тестирования с помощью системы, выдающей определенную оценку. При этом не используется обучающий набор данных, модель обучается за счет взаимодействия с окружающей средой.

Чтобы обучить ГНС выполнению такой сложной задачи, как визуальное распознавание, как правило, требуются десятки тысяч или даже миллионы обучающих примеров, каждый из них должен быть правильно размечен. Энтузиасты машинного обучения очень быстро поняли, что сбор большого количества размеченных данных представляет собой поистине непосильную задачу. Однако развитие Интернета на стыке веков в одночасье сделало эту задачу осуществимой. Такие интернет-гиганты, как Google и Facebook, начали использовать доступные им огромные океаны данных для обучения моделей, предназначенных для ряда бизнес-задач, таких как перевод с одного языка на другой. Одновременно с этим исследователи инициировали краудсорсинговые проекты, направленные на разметку обучающих наборов данных вручную. Новаторский пример — проект

ImageNet, давший толчок развитию технологий на базе ГНС для машинного распознавания образов.

ImageNet

ImageNet (www.image-net.org) — это база данных со ссылками на изображения, созданная для стимулирования прогресса в области машинного зрения. Она содержит ссылки на более чем 14 миллионов изображений, каждое из которых отнесено к одной категории или более в зависимости от содержимого изображения. Эта база данных имеет иерархическую структуру, что позволяет варьировать уровень обобщения (например, можно использовать категорию «собака» или более конкретную категорию породы собак — «лабрадор»). При реализации проекта ImageNet использовался краудсорсинг для ручного аннотирования каждого изображения.

Начиная с 2010 года в рамках данного проекта проводится ежегодный конкурс ILSVRC (ImageNet Large Scale Visual Recognition Challenge — конкурс по широкомасштабному распознаванию образов в ImageNet), призванный стимулировать исследования в области программного обеспечения для визуального распознавания объектов.

Вместе с тем развивались и аппаратные технологии. Так, в частности, новые графические процессоры (GPU) для обработки компьютерной графики и изображений (особенно игровые) уже были способны быстро производить сложную матричную обработку — как раз то, что нужно для обучения ГНС. Примерно с 2010 года стала возможной разработка глубоких нейронных сетей. ГНС быстро достигли уровня точности и скорости, не уступающего, а иногда и превосходящего человеческие способности в таких областях ИИ, как визуальное распознавание и перевод устной речи.

Неожиданное открытие: оптические иллюзии искусственного интеллекта

Во время активного развития и повышения точности глубоких нейронных сетей в 2013 году К. Шегеди вместе с рядом других ученых опубликовал статью «Интригующие свойства нейронных сетей»[1], которая была представлена публике на проходившей в следующем году Международной конференции по репрезентационному обучению (International Conference on Learning

[1] *Szegedy C. et al.* Intriguing Properties of Neural Networks // ICLR, 2014. bit.ly/2X2nu9c.

Representations, ICLR). Эта работа показала, что алгоритмы глубокого обучения можно обманом заставить выдавать некорректные результаты.

В статье были рассмотрены алгоритмы глубоких нейронных сетей для классификации изображений. В качестве входных данных они принимают изображение и классифицируют его по основному содержанию. Например, изображение может быть классифицировано как «стол», если стол — основной объект в изображении. Несмотря на то что такие нейронные сети на тот момент считались передовым подходом к классификации изображений, они делали неожиданные ошибки, когда им на вход подавались изображения с множеством умышленных и незаметных для человека изменений пикселей. Хотя человек не видел в изображении никаких изменений, эти незначительные модификации заставляли нейронные сети совершать грубые ошибки классификации.

На рис. 1.2 представлены три примера неправильно классифицированных изображений, которые были рассмотрены в названной статье. В левом столбце показаны исходные изображения, которые были корректно классифицированы ГНС-алгоритмом. В центральном столбце — те вредоносные искажения, которые были созданы специально для показанных слева исходных изображений. Степень этих искажений была уменьшена путем умножения каждого пиксельного искажения на понижающий коэффициент. После добавления уменьшенных (и менее заметных) искажений к исходному изображению были получены новые изображения (см. рис. 1.2, *справа*). Все изображения в правом столбце были ошибочно классифицированы тем же алгоритмом как страус *(Struthio camelus)*, несмотря на то что для человеческого глаза они выглядят так же, как оригиналы.

Это открытие заинтриговало не только специалистов в области ИИ, но и тех, чья деятельность не была связана с интеллектуальными машинами. Тот факт, что глубокие нейронные сети потенциально можно так легко обмануть, привлек внимание массовой прессы. В некоторых статьях данное явление было названо оптическими иллюзиями искусственного интеллекта.

Возможно, из-за того, что нейронные сети основаны на нейробиологических моделях и имитируют свойства человеческого интеллекта, мы и предположили, что они практически «думают», как люди. Поскольку концепции глубоких нейронных сетей изначально навеяны упрощенными моделями синапсов и нейронов человеческого мозга, было вполне уместным предположить, что глубокие нейронные сети интерпретируют изображения аналогично тому, как это делает зрительная кора головного мозга, — однако это не так. Вполне очевидно, что нейронные сети не классифицируют изображения путем

выявления абстрактных признаков, как это делают люди, а используют совершенно другие правила. С точки зрения специалистов по нейросетевым технологиям, понимание того, как можно обмануть такие алгоритмы, позволило получить лучшее представление о самих этих алгоритмах.

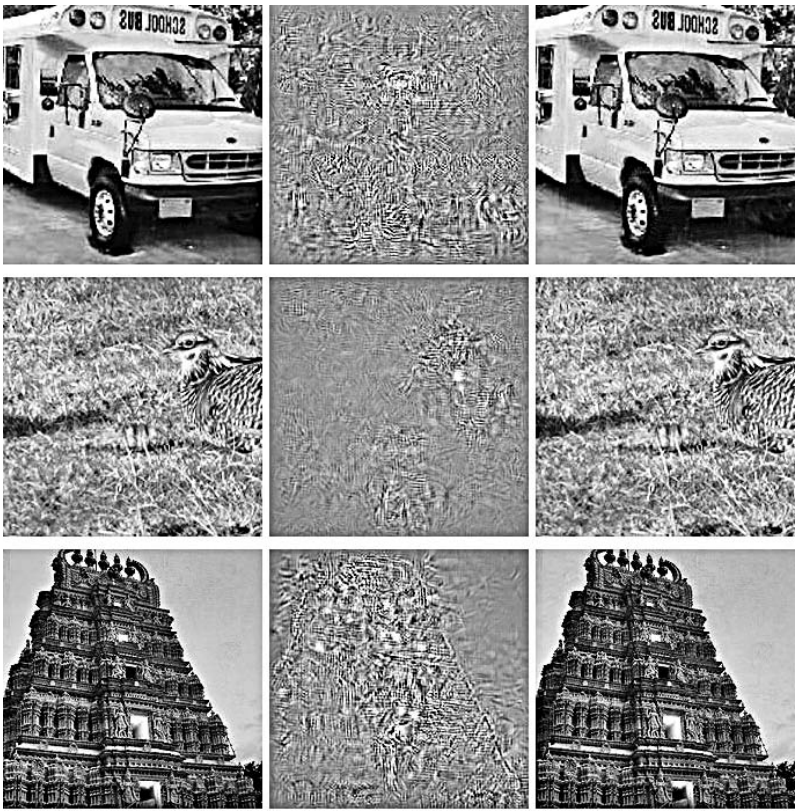


Рис. 1.2. Незначительные искажения ведут к неправильной классификации изображений. *Слева* представлены исходные изображения, *справа* — изображения с искажениями, которые были неверно классифицированы как страус (изображения взяты из статьи Шегеди и др. от 2014 года)

После выхода первой статьи Шегеди и др. было установлено, что такая уязвимость к обману есть и при обработке речи и текста, а значит, данное явление не ограничивается только областью ГНС для обработки изображений, а охватывает более широкий спектр ГНС-технологий. Учитывая постоянный рост роли глубоких нейронных сетей в нашей жизни, это были очень важные новости.

Что такое вредоносные входные данные

В сфере обработки изображений концепция вредоносных входных данных подразумевает создание оптических иллюзий, которым подвержен только искусственный интеллект. Так, вредоносное изображение может быть сгенерировано путем добавления в изображение кошки множества, казалось бы, неважных пикселей, которые заставят ИИ относить данное изображение к категории «собака», но не внесут в него каких-либо заметных изменений признаков, которые, по мнению человека, являются характерными для собаки. Вредоносные входные данные также могут представлять собой определенные метки на дорожном знаке, которые будут восприниматься человеком как граффити, но в то же время заставят автономное транспортное средство интерпретировать знак неправильно. Можно привести примеры и из сферы обработки аудиоданных. Например, включение в речь слабо слышимых вредоносных команд, ведущих к существенным ошибкам в интерпретации речи системой автоматического распознавания речи. Все эти сценарии обусловлены самими моделями ГНС.

Термин *«вредоносный образ» (adversarial example)* впервые использован Шегеди и его коллегами для описания примеров, подобных тем, что представлены на рис. 1.2. Иногда он подразумевает входные данные, созданные *с целью* заставить модель возвращать неверный результат, вне зависимости от того, удается ли таким данным действительно обмануть нейросеть или нет. Однако обычно этот термин подразумевает только те входные данные, которые сумели обмануть нейросеть. В этой книге термины *«вредоносные входные данные»* и *«вредоносный образ»* используются как синонимы, подразумевая входные данные, которые *успешно* обманывают нейросеть, заставляя ее выдавать предсказания, которые являются неверными с точки зрения человека. Соответственно, под невредоносными входными данными тут подразумеваются данные, которым не удается обмануть нейросеть, даже если они были созданы с вредоносной целью.

Вредоносное ПО как вредоносные входные данные

В настоящее время растет интерес к использованию нейронных сетей для выявления вредоносного ПО, поскольку присущая программному обеспечению вариабельность и постоянное развитие вредоносного ПО не позволяют четко определить, какие признаки программного обеспечения могут указывать на наличие угрозы.

> Под *вредоносными входными данными* может подразумеваться вредоносное ПО, когда антивирусное ПО реализуется с использованием машинного обучения. Это вполне логичное определение, поскольку в данном случае вредоносное ПО представляет собой входные данные, заставляющие модель машинного обучения возвратить ошибочный сигнал об отсутствии опасности.

В этой книге мы сосредоточимся на моделях ГНС, предназначенных для обработки цифровых представлений визуальной и звуковой информации, с обработкой которой так легко справляется наш биологический мозг. Изображения и аудиоданные представляют собой *непрерывные* данные, включая пиксели и звуковые частоты с непрерывным распределением значений. В отличие от них такие виды сложных данных, как текст, представляют собой *дискретные* данные и не состоят из количественно измеримых величин. В дискретной предметной области сложнее создать вредоносный образ, который мог бы оставаться незамеченным, поскольку в такой области трудно количественно измерить «небольшое» изменение. Так, например, небольшое изменение в слове может быть малозаметным, когда оно выглядит как опечатка, либо явным, когда оно придает слову совершенно другой смысл.

В случае систем искусственного интеллекта для обработки изображений и аудиоданных «неправильный» результат *не всегда* отличается от того, что воспринимает человек — иногда вредоносному образу удается также обмануть и биологический (человеческий) интеллект. Возникает вопрос: должен ли ИИ интерпретировать мир точно так же, как это делаем мы? Как правило, нам не нужно, чтобы ИИ имитировал человеческое мышление до такой степени, чтобы при этом дублировались и ошибки человеческого восприятия. Большинство вредоносных образов, рассматриваемых в этой книге, не может обмануть человеческий мозг — биологическую нейронную сеть. Эти примеры демонстрируют модели угроз и подчеркивают различие между искусственным и человеческим интеллектом.

Хотя любой алгоритм машинного обучения в какой-то мере уязвим к вредоносным входным данным, глубокие нейронные сети обладают повышенной степенью уязвимости, поскольку они лучше справляются с задачами, в которых трудно четко определить, по каким признакам данных следует производить обучение. То есть мы слабо представляем, какие аспекты данных важны для алгоритма ГНС, или не понимаем этого вообще. Если мы не понимаем, на основе каких аспектов данных алгоритм принимает решения, то можем ли мы надеяться на проведение хороших тестов для проверки устойчивости

алгоритмов? Вредоносные входные данные используют то обстоятельство, что модели глубокого обучения, как правило, имеют дело с миллионами возможных вариантов входных данных, основываясь лишь на небольшом количестве изученных примеров[1]. Обученные модели должны одновременно обладать и достаточной гибкостью, чтобы справляться с вариабельностью данных, и способностью обобщать новые данные. Как следствие, поведение глубокой нейронной сети остается непроверенным для большинства возможных входных данных и часто может быть неожиданным.

Шегеди и его коллеги описали так называемую искажающую атаку, однако существуют и другие методы обмана глубоких нейронных сетей. В следующих разделах вы познакомитесь с некоторыми разновидностями таких методов и основной терминологией, используемой в области вредоносных входных данных.

Вредоносное искажение

Образы, представленные на рис. 1.2, являются примером так называемой *искажающей атаки* — генерации вредоносных изображений путем создания точно рассчитанных модификаций исходного изображения, меняющих каждый пиксель на очень маленькую величину. Еще один способ такой атаки — внести более существенные изменения в несколько тщательно отобранных пикселей. Количество изменяемых пикселей и величина изменений, вносимых в каждый пиксель, могут варьироваться, однако общий эффект от изменений всегда остается незначительным, чтобы эти изменения не были заметны человеку. Искажение может казаться случайным, не являясь таковым; изменение каждого пикселя досконально подбирается таким образом, чтобы был получен желаемый результат. Далее в этой книге будет показано, как рассчитываются такие искажения для получения вредоносных входных данных.

Вредоносные искажения могут создаваться не только для изображений. Аналогичные методы применяются, например, и для аудиоданных (рис. 1.3). При этом используется тот же принцип — в аудиоданные вносятся небольшие изменения с целью внести ошибку в вычисления ГНС. Однако в то время, как во вредоносных изображениях используется *пространственное* измерение для внесения искажений в пиксели, во вредоносных аудиоданных производится искажение частот, распределенных *по времени*.

---

[1] Набор данных для обучения, как правило, содержит тысячи примеров, но даже такое количество — лишь небольшая часть возможных входных данных.

Так, например, незначительные и незаметные для человека изменения частоты голоса на протяжении речевого сегмента могут привести к неправильной интерпретации предложения моделями преобразования речи в текст[1].

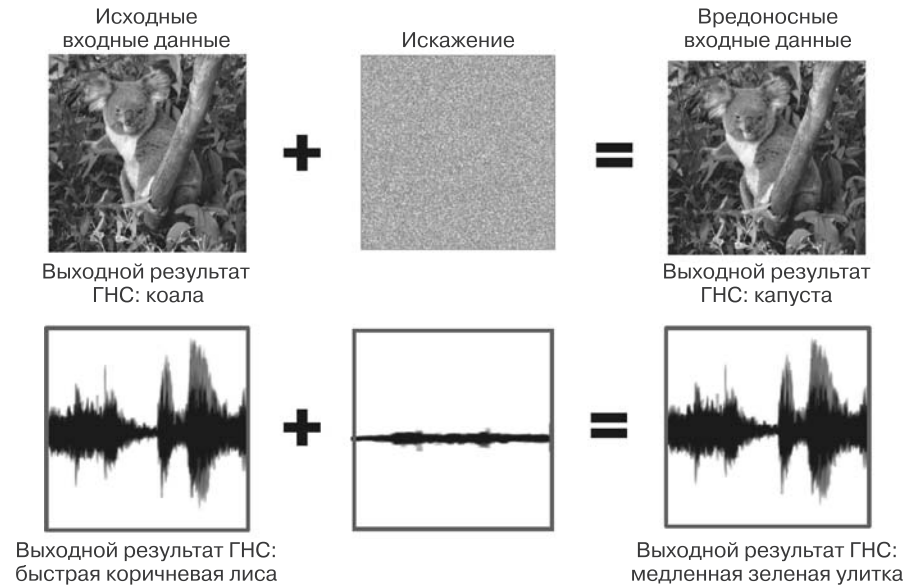


Рис. 1.3. Вредоносные искажения, вносимые в изображение с целью обмануть классификатор изображений и в аудиоданные с целью обмануть систему преобразования речи в текст

Неестественные вредоносные входные данные

В 2015 году Нгуен (Nguyen) совместно с другими исследователями опубликовал статью под названием «Глубокие нейронные сети можно легко обмануть: высоковероятные предсказания для нераспознаваемых изображений»[2]. Исследование показало, что, когда неважен реализм изображения, могут генерироваться вредоносные данные, не похожие совершенно ни на что, которые с высокой степенью вероятности будут классифицироваться глубокими нейронными сетями. На рис. 1.4 показаны некоторые примеры, приведенные в этой статье.

---

1 *Carlini N., Wagner D.* Audio Adversarial Examples: Targeted Attacks on Speech-to-Text // IEEE Deep Learning and Security Workshop, 2018. bit.ly/2IFXT1W.

2 *Nguyen A. et al.* Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images // Computer Vision and Pattern Recognition, 2015. bit.ly/2ZKc1wW.

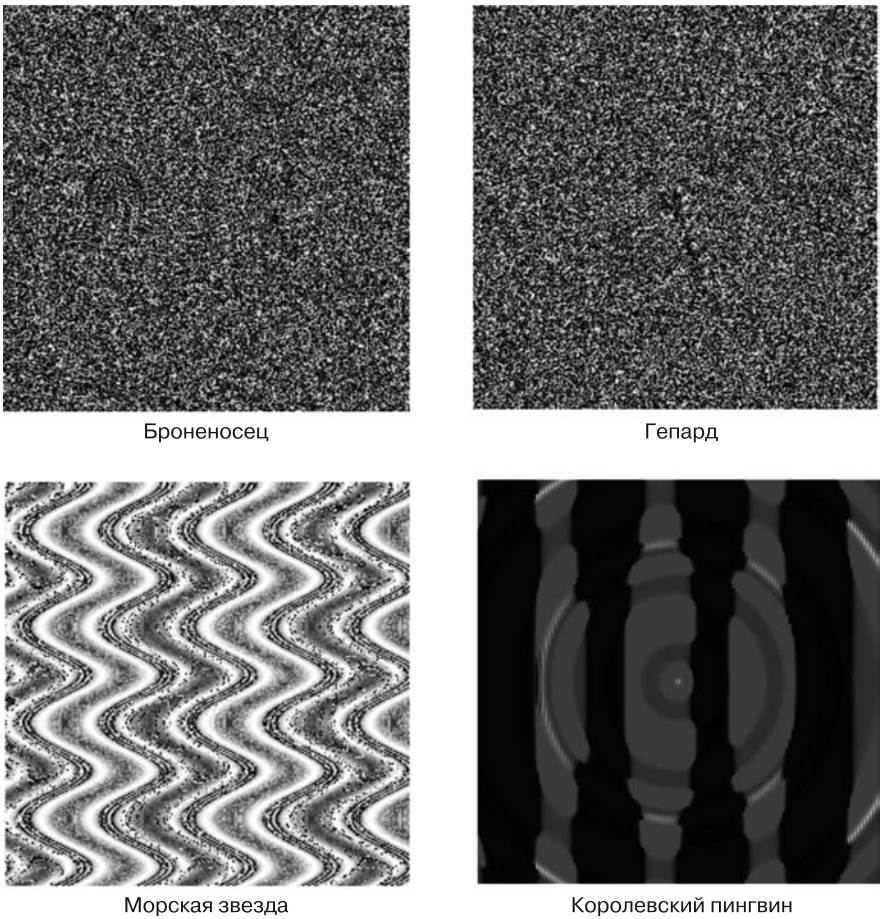


Рис. 1.4. Сгенерированные компьютером вредоносные образы, не распознаваемые человеком, и категории, к которым они были отнесены современными ГНС (изображение из статьи Нгуена и др. от 2015 года)

Такие изображения являются подтверждением того, что глубокие нейронные сети могут научиться интерпретировать изображения на основе признаков, не используемых человеком. Хотя очевидно, что подобные изображения вряд ли смогут кого-то обмануть, все же не стоит их игнорировать. Злоумышленники могут использовать такие образы, чтобы заставить систему генерировать ложноположительные результаты, что может привести к отказу в обслуживании за счет переполнения системы данными.

Вредоносная заплатка

Вместо того чтобы создавать вредоносный образ путем распределения изменений по всем входным данным, злоумышленники также могут сфокусироваться на какой-то одной области и тем самым отвлечь внимание ГНС от тех аспектов данных, которые она должна рассматривать.

Вредоносные заплатки представляют собой тщательно подобранные «наклейки», которые накладываются на данные. Эти заплатки отвлекают внимание глубокой нейронной сети от релевантных аспектов входных данных, заставляя ее выдать неверный результат. На рис. 1.5 представлен пример вредоносной заплатки, сгенерированной исследователями компании Google. Эта «наклейка» математически оптимизирована так, чтобы с точки зрения ГНС представлять собой более заметный признак, по сравнению с объектом, существующим в реальном мире, что с высокой степенью вероятности обеспечит ошибку классификации.

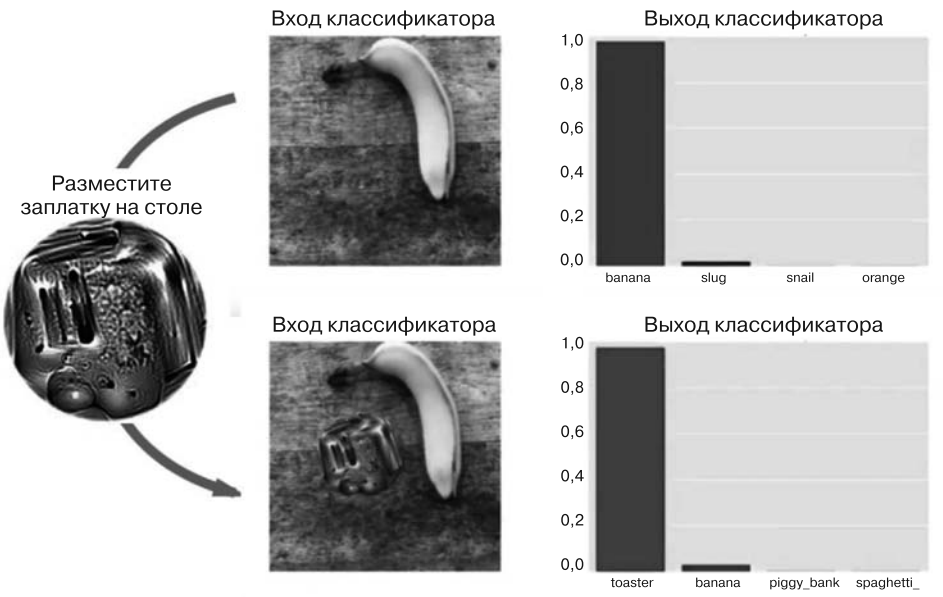


Рис. 1.5. Сгенерированная компьютером вредоносная заплатка, которая с высокой степенью вероятности ведет к ошибочному отнесению банана к категории «тостер» (из статьи Брауна и др. от 2017 года[1])

[1] *Brown T. B. et al.* Adversarial Patch. 2017. bit.ly/2IDPonT.

Хотя такое вредоносное изменение очевидно для человека, он может не придать этому значения, особенно если оно не очень бросается в глаза и не влияет на общую интерпретацию изображения. Так, например, заплатка может быть помещена у края изображения и маскироваться под логотип. В данном случае вряд ли можно отнести изображение к категории «тостер» вместо категории «банан» еще и в силу того, что заплатка была специально сделана заметной для ГНС, но не для человека.

Теперь посмотрим, как этот принцип действует для аудиоданных. Аудиозаплатка представляет собой звуковой клип, который является достаточно коротким или достаточно тихим для того, чтобы его проигнорировал слушатель-человек. Подобно тому как для вредоносной заплатки на изображении необходимо подобрать оптимальный размер и пространственное расположение, для аудиозаплатки требуется подобрать подходящее расположение во времени и интенсивность. Принцип действия вредоносных заплаток для изображений и аудиоданных показан на рис. 1.6.

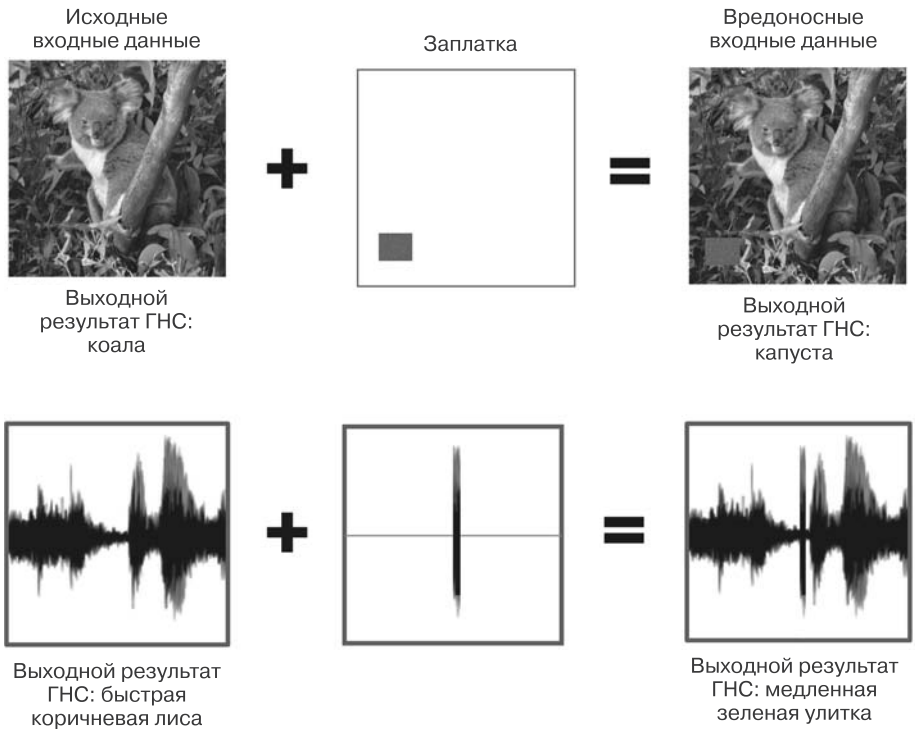


Рис. 1.6. Вредоносные заплатки, помещаемые на изображение с целью обмануть классификатор изображений и в аудиоданные с целью обмануть систему для преобразования речи в текст

Интересным свойством вредоносных заплаток является то, что их легче использовать повторно по сравнению с вредоносными искажениями. Так, в частности, сгенерированная компьютером вредоносная «наклейка» может быть эффективна в отношении большого количества изображений, что позволит злоумышленнику делиться ею в Интернете или копировать и накладывать ее во множестве разных областей.

Вредоносные образы в физическом мире

Вредоносный образ, представленный на рис. 1.2, был сгенерирован путем цифровых манипуляций; в этом случае путем изменения данных изображения на уровне пикселей. Однако это предполагает, что у злоумышленника должен быть доступ к цифровому формату тех данных, которые передаются модели — как, например, в том случае, когда злоумышленник загружает цифровое изображение (допустим, в формате JPEG) на веб-сайт, производящий его обработку.

Во многих случаях злоумышленник может получить доступ только к физическому миру[1], чтобы повлиять на данные, поступающие на воспринимающее устройство (такое как микрофон или камера), которое генерирует цифровые данные. При этом злоумышленник может использовать сгенерированные цифровые вредоносные заплатки в виде двумерных или даже трехмерных объектов в пределах сцены. Шариф и др. успешно продемонстрировали этот принцип в статье «Аксессуары для преступления: реальные скрытые атаки на современные системы распознавания лиц»[2], используя вредоносные очки, позволяющие надевшему их человеку вызвать сбой в работе программного обеспечения для распознавания лиц. Пример таких очков показан на рис. 1.7.

Когда у злоумышленника нет возможности повлиять на цифровое представление данных, ему, безусловно, гораздо труднее осуществить искажающую атаку. Часто приведенный в качестве примера сценарий вредоносной атаки в физическом мире состоит в изменении окружающей среды таким образом, чтобы автономное транспортное средство приняло неправильные решения в отношении рулевого управления, ответной реакции, скорости и т. д. после

---

[1] Под физическим миром в этой книге понимаются аспекты окружающего мира за пределами сферы компьютерных вычислений.

[2] *Sharif M. et al.* Accessorize to a Crime: Real and Stealthy Attacks on State-of-the-Art Face Recognition // Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016. bit.ly/2x1Nebf.

обработки полученных камерой видеоданных. Это означает, что поведение транспортного средства может измениться при изменении дорожной разметки, окраски других транспортных средств или дорожных знаков. Эйкхолт с коллегами доказал возможность проведения вредоносных атак в реальном мире с использованием принципов искажающей атаки[1].

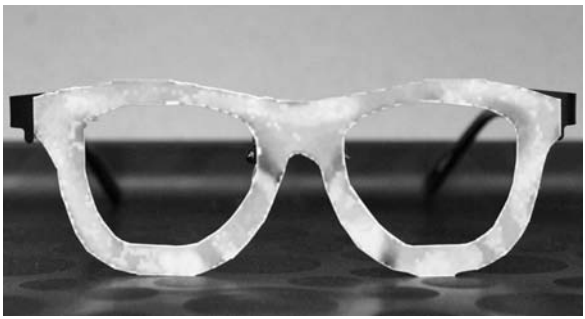


Рис. 1.7. Вредоносные очки, созданные с целью обмануть системы распознавания лиц (изображение из статьи Шарифа и др. от 2016 года)

На рис. 1.8 приведены примеры искажающей атаки с использованием обычного дорожного знака остановки. Данная атака заставляет ГНС неправильно интерпретировать знак, следовательно, потенциально может обмануть автономное транспортное средство.

Рис. 1.8. Физическое искажение знака остановки может обмануть автономное транспортное средство, использующее видеоданные (изображение из статьи Эйкхолта и др. от 2018 года)

[1] *Eykholt K. et al.* Robust Physical-World Attacks on Deep Learning Visual Classification // Computer Vision and Pattern Recognition, 2018. bit.ly/2FmJPbz.

Что интересно, в отличие от описанных ранее атак с помощью цифровых искажений, атаки с использованием искажений в физическом мире зачастую легко распознаются человеком. Поэтому обычно злоумышленник стремится внести изменение окружающей среды так, чтобы человек не воспринял его как нечто необычное. Так, в случаях, показанных на рис. 1.8, можно заметить искажения на знаке остановки, но вряд ли кто-то посчитает их чем-то подозрительным; искажения подобны граффити и потому не воспринимаются человеком как что-то опасное.

Примерно такой же подход можно использовать и для генерации вредоносных аудиоданных. Злоумышленник может создать вредоносную речь, замаскированную в другой речи, звуках или даже тишине, которая будет представлять угрозу для систем с голосовым управлением (например, виртуальных ассистентов)[1].

Вредоносное машинное обучение в более широком смысле

В данной книге рассматриваются вредоносные образы для нейронных сетей, обрабатывающих изображения и аудиоданные. Однако эти примеры являются составной частью более широкой группы атак, называемой *вредоносным машинным обучением (вредоносным МО)*. Этот термин включает в себя все виды потенциальных атак на алгоритмы машинного обучения (ГНС и более классические алгоритмы машинного обучения) и все типы данных[2].

Вредоносные образы иногда (правильно) называют *атаками уклонения*, подразумевая под этим модификацию входных данных с целью избежать их распознавания алгоритмом МО. Однако вредоносные входные данные могут использоваться не только для уклонения. Хотя большинство рассматриваемых в этой книге атак являются атаками уклонения, некоторые из них таковыми не являются. Так, например, злоумышленник может направить в систему поток вредоносных образов, заставляющих ее генерировать множество ложноположительных результатов, что потенциально может привести к отказу в обслуживании.

---

1 *Carlini, Wagner.* Audio Adversarial Examples.

2 Впервые услышав выражение «вредоносное машинное обучение», можно ошибочно подумать, что подразумевается машинное обучение, используемое злоумышленником для атаки на систему.

Помимо атак уклонения, в сфере вредоносного МО могут также встречаться следующие виды атак.

- *Отравляющие атаки*. В случае отравляющей атаки вредоносные данные умышленно вносятся в обучающий набор данных, что ведет к неправильному обучению алгоритма. Этому типу атак подвержены системы с постоянным обучением на основе данных, полученных из ненадежных источников. Отравляющие атаки не рассматриваются в этой книге.
- *Обратное проектирование моделей машинного обучения*. Если злоумышленнику удается получить доступ к копии алгоритма МО, он может прибегнуть к обратному проектированию алгоритма для извлечения потенциально конфиденциальной или секретной информации о характеристиках обучающих данных. Атаки данного типа не рассматриваются в этой книге.

Последствия воздействия вредоносных входных данных

Модели ГНС широко используются в современном мире для решения задач в нашей повседневной жизни. Кроме того, системы ИИ, которые содержат эти модели, часто обрабатывают данные, не имея контроля над ними (например, получая входные данные из цифровых онлайн-источников или из физического мира). Вот несколько примеров таких систем:

- системы распознавания лиц для контроля доступа или видеонаблюдения;
- веб-фильтры для выявления загруженных на сервер изображений, носящих непристойный или оскорбительный характер;
- автономные транспортные средства, действующие в неограниченной физической среде;
- выявление обмана телефонных систем с голосовым управлением;
- виртуальные ассистенты с голосовым управлением.

Если ГНС можно так легко обмануть с помощью вредоносных образов, не представляет ли это киберугрозу для решений на базе ИИ, которые получают данные из ненадежных источников? Насколько большую угрозу представляют вредоносные входные данные для безопасности и целостности систем, используемых в повседневной жизни? И наконец, к каким стратегиям снижения риска могут прибегнуть разработчики таких систем, чтобы предотвратить использование злоумышленниками этого направления атаки?

Чтобы ответить на эти вопросы, нужно разобраться с тем, что движет злоумышленниками и какими возможностями они обладают. Необходимо понять, почему глубокие нейронные сети уязвимы к вредоносным входным данным и каким образом их можно сделать менее подверженными такому обману. И, кроме того, нужно понять, составной частью какой последовательности обработки являются ГНС, как эта обработка может сделать системы более (или менее) устойчивыми к атакам. На сегодняшний день пока нет ни одного известного способа сделать ГНС абсолютно устойчивой к вредоносным входным данным, однако понимание этой проблемы и с позиции злоумышленника, и с позиции защищающейся организации позволяет разработать более эффективные меры защиты.

Вредоносные входные данные представляют интерес с еще одной точки зрения: различия в том, как обрабатывают информацию человек и ГНС, показывают разницу между биологическими и искусственными нейронными сетями. Несмотря на то что глубокие нейронные сети были изначально навеяны нейробиологическими моделями и в относящейся к ним терминологии присутствует слово *«нейронный»*, дисциплина разработки эффективного глубокого обучения стала главным образом сферой деятельности математиков и специалистов по анализу и обработке данных. ГНС, по сути, представляет собой сложную математическую функцию, которая принимает входные данные и генерирует результат. А обучение модели ГНС — это фактически задача математической оптимизации, направленная на итеративное изменение параметров сложной функции для максимального повышения ее точности. Уже сам факт наличия вредоносных входных данных говорит о принципиальной ошибочности представления о том, что ГНС воплощает в себе модель, приближенную к человеческому мышлению.

Наконец, не следует забывать и о риске внесения в компьютерную систему разрушающего, мошеннического или вредоносного поведения в том случае, если какой-либо алгоритм системы может дать сбой при обработке непроверенных входных данных. Обеспечение устойчивости компьютерной системы к вредоносным входным данным должно быть неотъемлемой частью процесса обеспечения качества компьютерной системы, имеющей в своем составе алгоритмы машинного обучения.

ГЛАВА 2

Мотивация к атакам

Сегодня технологии на базе глубоких нейронных сетей уже прочно вошли в нашу жизнь. Например, такие виртуальные помощники, как Amazon Alexa, Apple Siri, Google Assistant и Microsoft Cortana, используют модели глубокого обучения для понимания речевых аудиоданных. Многие алгоритмы для обеспечения и контроля сетевых взаимодействий (таких как веб-поиск) основаны на глубоких нейронных сетях для распознавания тех данных, которыми они оперируют. Модели глубокого обучения все чаще используются в областях применения с высокими требованиями к безопасности, таких как автономные транспортные средства.

Многие технологии на базе ИИ получают данные непосредственно из физического мира (например, с камер) или используют цифровые представления данных, предназначенные для человека (например, изображения, загружаемые на сайты социальных сетей). Это потенциально может вести к проблемам, поскольку при обработке данных из ненадежных источников любая компьютерная система становится уязвимой к атакам. За созданием вредоносных входных данных, использующих эти уязвимости, могут стоять самые разные мотивы, однако их можно разделить на следующие основные категории.

- *Уклонение.* Сокрытие контента от автоматического цифрового анализа. Примеры см. в разделах «Обход веб-фильтров» на с. 39, «Камуфляж против видеонаблюдения» на с. 42 и «Личная конфиденциальность в Интернете» на с. 43.
- *Влияние.* Воздействие на автоматизированные решения для получения личной, коммерческой или организационной выгоды. Примеры см. в разделе «Репутация в Интернете и управление брендом» на с. 41.
- *Дезориентация.* Создание хаоса с целью дискредитировать или нарушить работу организации. Примеры см. в разделах «Дезориентация автономных транспортных средств» на с. 44 и «Устройства с голосовым управлением» на с. 46.

В данной главе приведено несколько примеров возможной мотивации для создания вредоносных образов. Это далеко не полный список, но он дает представление о характере и разнообразии типов угроз.

Обход веб-фильтров

Сегодня организации испытывают все большую необходимость управлять получаемым извне веб-контентом для блокирования контента, который может рассматриваться как оскорбительный или неприличный. Это особенно касается компаний — владельцев социальных сетей и электронных торговых площадок, бизнес-модель которых зависит от внешних данных. Кроме того, многие компании связаны правовыми обязательствами по отслеживанию оскорбительных материалов и недопущению их дальнейшего распространения.

Эти организации сталкиваются со сложными задачами, которые с каждым днем усложняются. Они просто не могут найти достаточное количество людей для того, чтобы отслеживать все данные, загружаемые на сайты, с необходимой скоростью и блокировать их при необходимости. На сайты соцсетей ежедневно загружаются миллиарды сообщений. Их содержание не имеет четкой структуры и трудно поддается фильтрации; они могут содержать изображения, звуковую, текстовую информацию с трудноуловимой разницей между оскорбительным и неоскорбительным или законным и незаконным контентом. Отслеживать и фильтровать этот контент по мере его загрузки на сайт с привлечением людей просто невозможно.

Очевидное решение — использовать интеллектуальные машины, которые бы отслеживали, фильтровали или как минимум сортировали данные, как показано на рис. 2.1. В основе таких решений лежат глубокие нейронные сети. Их можно научить распознавать эмоциональную окраску и оскорбления в тексте, они могут классифицировать содержимое изображений и даже определять действия, выполняемые на видео. Например, ГНС можно научить распознавать изображения с намеками на употребление наркотиков, что позволит отсортировать такие изображения для их дальнейшей проверки человеком.

Когда отдельный пользователь или группа пользователей хочет загрузить контент, который не соответствует политике веб-сайта, возникает необходимость обойти систему фильтрации или отслеживания таким образом, чтобы загружаемый контент доносил предназначенную для людей информацию.

При этом отслеживающей веб-контент организации требуется постоянно повышать точность алгоритмов, выявляющих оскорбительный, неприличный и незаконный контент, а также осуществлять перехват вредоносных входных данных. Злоумышленнику же требуется совершенствовать вредоносный веб-контент по мере улучшения системы мониторинга, чтобы избегать распознавания искусственным интеллектом, обеспечивая донесение того же семантического значения человеку.

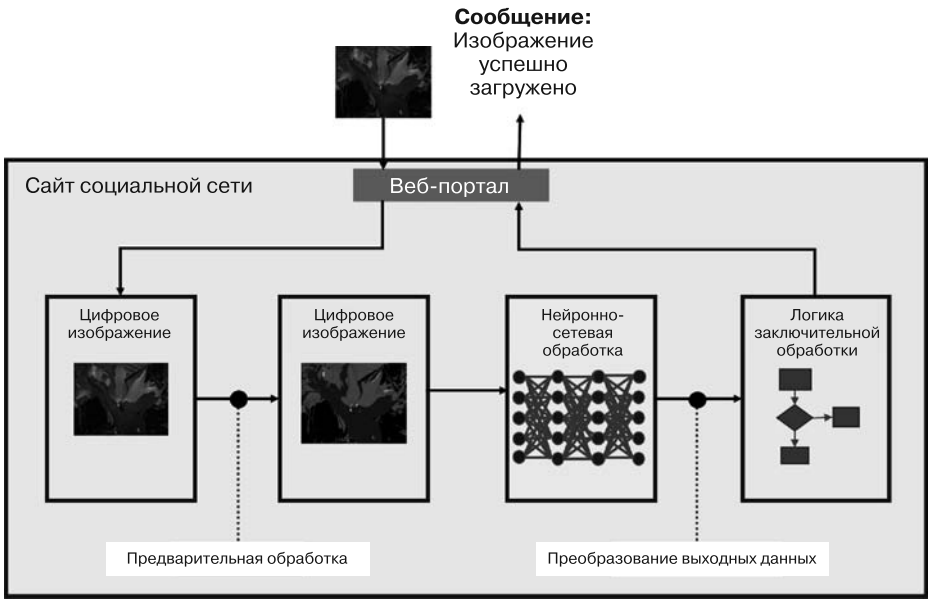


Рис. 2.1. Изображения, загружаемые на сайт социальной сети, могут подвергаться обработке и проверке искусственным интеллектом перед их добавлением на сайт

Злоумышленник может использовать еще один подход. Если обойти веб-фильтр загрузок невозможно, почему бы тогда просто не засыпать его множеством данных, обеспечивающих дезориентацию и дополнительные расходы для организации, обеспечивающей защиту? Принимаемое искусственным интеллектом решение о том, следует ли считать загружаемые данные «оскорбительными», обычно не является исключительно бинарным. Это скорее будет статистическая оценка вероятности с некоторым пороговым значением. Организации часто используют модерацию человеком для проверки изображений или данных, оценка которых близка к пороговому значению. Поэтому генерация большого количества неопасных данных, классифицируемых искусственным интеллектом как возможно опасные,

негативно скажется на оперативных возможностях организации и может снизить степень уверенности в точности результатов, выдаваемых искусственным интеллектом.

Если человеку трудно точно установить, *почему* данные были оценены как возможно нарушающие политику (поскольку они выглядят как неопасные), то большой наплыв данных потребует больших затрат времени и человеческих ресурсов — по сути, это атака типа «отказ в обслуживании».

Репутация в Интернете и управление брендом

Поисковые системы представляют собой сложные алгоритмы, которые не только решают, какие результаты следует отобразить в ответ на запрос, например, «кот на скейтборде», но и определяют последовательность этих результатов. Очевидно, что с коммерческой точки зрения лучше, если результат находится в начале списка. В силу этого компании мотивированы к тому, чтобы выяснить принцип действия алгоритмов поисковых систем и добиться того, чтобы их реклама отображалась на первой странице поисковой системы Google или Bing и была хорошо заметна при ее размещении на веб-страницах. Такая *оптимизация в поисковых системах* (search engine optimization, SEO), или просто поисковая оптимизация, является стандартной отраслевой практикой уже в течение многих лет. SEO играет ключевую роль в стратегии интернет-маркетинга.

Поисковые системы могут использовать специальных роботов для просеивания и индексирования страниц для отображения в результатах поиска на основе HTML-метаданных страницы, ее входящих ссылок и содержимого. Поисковые роботы представляют собой автоматизированные системы на основе ИИ, работающие без вмешательства человека. Поскольку информация заголовков легко поддается изменению, поисковые системы обычно больше полагаются на содержимое. Индексация на основе содержимого также делает возможным поиск с использованием менее распространенных ключевых слов, которые не всегда включаются в метаданные.

Характеристики страниц, основанные на их содержимом, особенно интересны в контексте вредоносных образов. Поскольку обновление изображений веб-сайта может повлиять на его позицию в результатах поисковой системы, у компании, желающей повысить видимость сайта для целевой аудитории, возникает мотив использовать вредоносные искажения или заплатки для изменения или подкрепления категории изображений без изменения восприятия их человеком.

Существует и более злонамеренный вариант атаки: стремясь подорвать доверие к организации или отдельному человеку, злоумышленник может сгенерировать вредоносные изображения, ведущие к ошибочной ассоциации объекта атаки с чем-то вредящим его репутации. Так, например, вредоносные изображения шоколадного батончика могут быть ошибочно классифицированы поисковой системой как «яд» и включены в число результатов, выдаваемых при поиске изображений ядов. Даже такая неявная ассоциация может серьезно сказаться на восприятии бренда потребителями.

Камуфляж против видеонаблюдения

Камеры видеонаблюдения получают свои данные из физического мира, что заставляет взглянуть на вредоносные входные данные с совершенно иной точки зрения по сравнению с предыдущими примерами. В данном случае цифровой контент генерируется на основе данных воспринимающего устройства (камеры) и не может изменяться посторонними лицами, которые не являются сотрудниками организации[1].

Хотя цифровые данные видеонаблюдения (видеозаписи или неподвижные кадры) часто по-прежнему отслеживаются людьми, эта задача становится все более трудноосуществимой из-за возрастания объема информации и затрат времени на ее обработку. В большинстве случаев данные видеонаблюдения можно не отслеживать активно в режиме реального времени, а анализировать позднее в медленном режиме (как, например, при расследовании преступлений). Организации все чаще прибегают к автоматизированным методам отслеживания или к сортировке данных с камер видеонаблюдения с использованием технологий на базе ИИ, например, для автоматического распознавания в данных видеонаблюдения конкретных лиц или транспортных средств с выдачей соответствующих оповещений.

Не нужно много фантазии, чтобы вообразить сценарии, в которых у злоумышленника может возникнуть желание обхитрить систему. Цель злоумышленника может состоять в том, чтобы создать своего рода «плащ-невидимку», способный обмануть ИИ, не привлекая лишнего внимания человека. При этом обычно достаточно просто не допустить активации (генерации) искусственным интеллектом тревожного сигнала, влекущего за собой тщательную проверку изображения или видеозаписи человеком. Например, злоумышленник может постараться обмануть систему распознавания лиц в режиме реального

[1] Очевидно, что это не так в случае более серьезных нарушений системы безопасности, когда посторонние лица получают доступ к внутренним данным организации.

времени, используемую службой безопасности аэропорта. Точно так же недопущение распознавания подозрительной активности системой обнаружения угроз в режиме реального времени дает злоумышленнику больше возможностей для совершения противоправных действий. При рассмотрении данных не в режиме реального времени может выявиться, что камеры видеонаблюдения зафиксировали информацию, относящуюся к преступлению, такую как лица преступников, номерные знаки и т. д., которую можно использовать для поиска на основе свидетельских показаний после совершения преступления. Скрыв эту информацию от ИИ, преступник может сделать выявление преступления менее вероятным.

Конечно, в основе такого обмана могут лежать не только криминальные мотивы. В мире, где отслеживается все и вся, люди стремятся закамуфлировать заметные черты лица от ИИ, чтобы сохранить личную конфиденциальность. Для достижения этой цели может использоваться, в частности, нейтральная одежда или макияж, что позволит привести правдоподобное отрицание против обвинений в намеренном обмане системы видеонаблюдения, объясняя ее неправильное функционирование просто как сбой в работе[1].

Существует еще один интересный сценарий: что, если злоумышленник внесет в реальном мире физические изменения, которые не будут казаться опасными и вредоносными для человека, но заставят систему видеонаблюдения поднять ложную тревогу? Это позволит злоумышленнику отвлечь ресурсы организации в ложном направлении, чтобы совершить реальное преступление в каком-либо другом месте.

Личная конфиденциальность в Интернете

Многие платформы социальных сетей для улучшения взаимодействия с пользователем извлекают информацию из загружаемых на сайт изображений. Так, например, Facebook регулярно извлекает и идентифицирует лица на изображениях для более эффективной разметки изображений, выполнения поиска и рассылки уведомлений.

Опять же пользователь, желающий сохранить личную конфиденциальность, может изменить изображения таким образом, чтобы затруднить распознавание лиц для использующегося платформой искусственного интеллекта. Для камуфлирования лиц от ИИ могут вноситься такие изменения, как наложение вредоносных заплаток на край изображения.

---

[1] *Sharif et al.* Accessorize to a Crime.

Дезориентация автономных транспортных средств

Широко известным примером использования ИИ являются автономные транспортные средства, которые относятся к системам с высокими требованиями к безопасности. Такие транспортные средства функционируют в неупорядоченном, неограниченном и постоянно меняющемся физическом мире. Уязвимость к вредоносным входным данным может при этом привести к катастрофическим последствиям.

Автономными могут быть не только автомобили. Сегодня автономность получает все более широкое распространение на море, в воздухе и под водой. Автономные транспортные средства также используются в ограниченных и закрытых окружениях, таких как производственные помещения, для выполнения основных или, возможно, опасных задач. Даже при таком ограниченном окружении существует риск получения с камеры вредоносных входных данных, внесенных сотрудником организации (внутренняя угроза) или лицом, получившим доступ к рабочей зоне системы. В то же время не следует забывать о том, что, как правило, автономные транспортные средства контролируют физическое окружение, руководствуясь не только данными воспринимающего устройства. Большинство автономных систем получает информацию из нескольких источников, которые могут включать:

- *внебортовые данные*. Большинство автономных транспортных средств руководствуется данными, получаемыми из одного или нескольких внебортовых централизованных источников[1]. Внебортовые данные включают в себя сравнительно статичную информацию (карты и скоростные ограничения), централизованно собираемые динамические данные (например, сведения об интенсивности дорожного движения) и данные, относящиеся к конкретному транспортному средству (например, его GPS-координаты). Все эти типы источников данных уже используются в таких приложениях для GPS-навигации, как Waze, Google Maps и HERE WeGo.

  В других областях применения могут использоваться иные виды внебортовых данных. Например, в сфере морских перевозок широко применяется автоматическое отслеживание координат морских судов по данным *автоматических идентификационных систем* (АИС). С помощью этих систем корабли регулярно передают в режиме реального времени свои идентификационные данные и координаты, что позволяет органам морской власти следить за их передвижением;

[1] В таких ограниченных окружениях, как производственные или жилые помещения, автономные транспортные средства иногда руководствуются исключительно данными бортовых датчиков.

- *данные бортовых датчиков.* Автономное транспортное средство также может руководствоваться данными таких бортовых датчиков, как камеры, датчики расстояния, акселерометры и гироскопические датчики (для выявления позиционного вращения). Эти данные играют решающую роль в предоставлении информации об изменениях в непосредственной близости от транспортного средства, например, при подаче тревожного сигнала в режиме реального времени или при каких-либо неожиданных событиях.

В некоторых случаях автономное транспортное средство принимает решение лишь на основе данных датчиков. Например, автономное транспортное средство может корректировать свое положение на дороге, руководствуясь исключительно данными датчика, как показано на рис. 2.2. Такой сценарий может представлять существенную угрозу безопасности, поскольку генерируемая информация потенциально ненадежна.

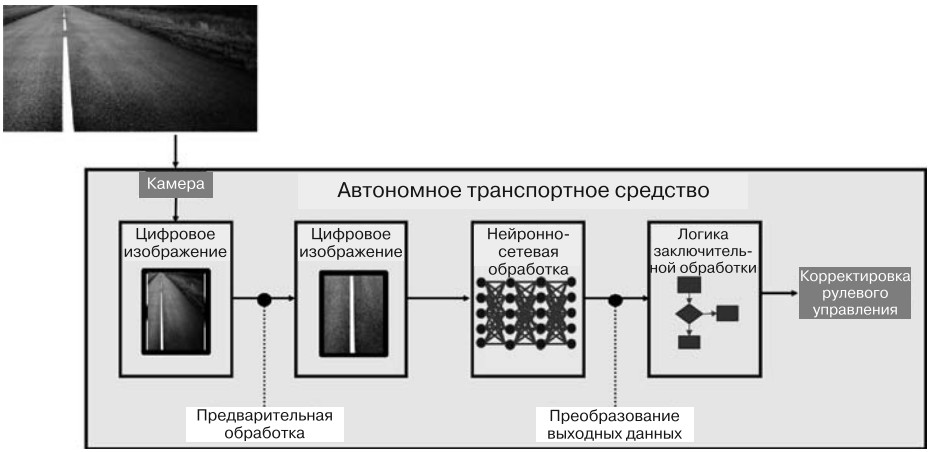


Рис. 2.2. Автономное транспортное средство может корректировать свое положение на дороге на основе данных камеры

На практике автономные транспортные средства обычно руководствуются информацией, получаемой из нескольких источников данных, и действуют с некоторой долей перестраховки. Если дорожный знак остановки STOP будет вредоносным образом «превращен» в разрешающий движение знак преимущества движения, это вряд ли обманет транспортное средство, которому также доступны централизованные данные о регулировании дорожного движения (сведения о скоростных ограничениях, перекрестках и местах, где нужно остановиться или уступить дорогу). С большей степенью вероятности злоумышленники попытаются «сыграть» на этой осмотрительности

транспортного средства — например, они могут парализовать работу дорожной сети, усеяв дорогу множеством неопасных с виду наклеек, ошибочно интерпретируемых как опасные объекты.

Устройства с голосовым управлением

Голосовое управление предоставляет естественный бесконтактный способ управления многими аспектами нашей жизни. С его помощью можно выполнять широкий круг задач — от управления мультимедийными устройствами и домашней автоматизации до поиска товаров и совершения покупок в Интернете. Такие устройства с голосовым управлением, как смартфоны, планшеты и голосовые помощники, постепенно занимают прочное место в нашем доме. Лежащая в основе этих устройств обработка речи осуществляется с помощью продвинутых технологий на базе ГНС и отличается высоким уровнем точности. На рис. 2.3 показано, как может выглядеть простейшая последовательность обработки для устройства с голосовым управлением.

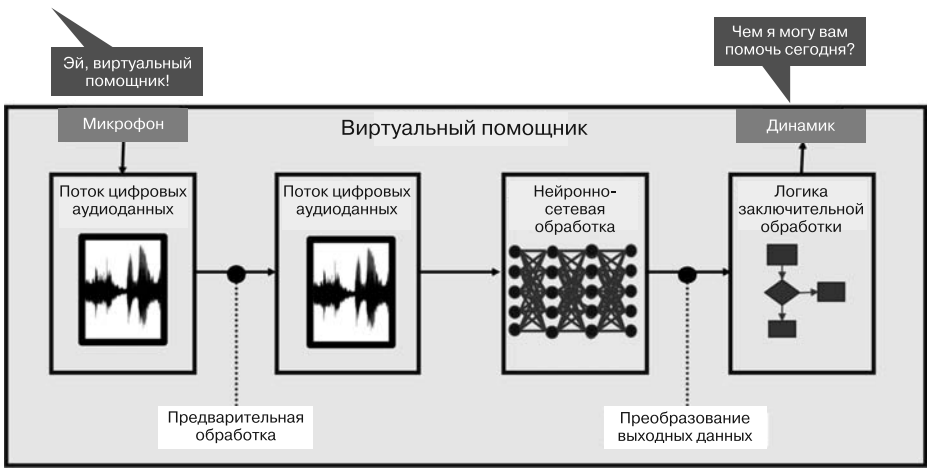


Рис. 2.3. Виртуальный помощник использует ИИ для обработки речевых данных и выдачи соответствующего ответа

Поток аудиоданных может поступать в дом по каналам радио- и телевещания или в виде сетевого контента. Незаметно для пользователя в эти аудиоданные может быть включен вредоносный контент, который, к примеру, будет давать голосовому помощнику указание увеличить громкость проигрываемой песни. При всей незначительности таких лишь немного раздражающих отклонений

в работе голосового помощника они тем не менее могут подорвать доверие к нему со стороны пользователей. Помощник, который ведет себя непредсказуемо, будет раздражать пользователей и в конечном итоге вызовет у них отвращение. А если устройство утратит доверие пользователей в домашнем окружении, ему будет очень сложно заслужить его снова. Потенциально скрытые команды могут оказаться и не столь безобидными — например, они могут самопроизвольно отправить СМС или сделать запись в социальной сети, изменить настройки устройства, выполнить переход по вредоносной ссылке или изменить настройки домашней системы безопасности.

Для выполнения функций, которым необходим повышенный уровень безопасности, голосовые помощники требуют дополнительных мер безопасности, чтобы не допустить их случайного или умышленного неправильного использования. Помощник может спросить: «Вы *действительно* хотите приобрести экземпляр книги “Надежность нейронных сетей”?» После этого будет ожидать подтверждения покупки. Вы вряд ли скажете «да» в нужный момент, если эта покупка не была инициирована вами, хотя такое и не исключено. В то же время если есть возможность дать голосовому помощнику вредоносную *команду*, то также существует и возможность дать ему вредоносный *ответ*, конечно, при условии, что поблизости не будет никого, кто бы услышал запрос на подтверждение.

Важной особенностью здесь является то, что вредоносные образы не всегда действительно наносят вред — они могут использоваться в развлекательных целях как дополнительный способ передачи команд. С их помощью можно не только причинить вред, но и извлечь коммерческую выгоду.

Представьте, что вы запаслись попкорном и сели перед телевизором, чтобы еще раз посмотреть фильм «Сияние». Однако вы решили посмотреть не обычную версию этого фильма ужасов, а версию с «интегрированной поддержкой домашней автоматизации». Эта версия с расширенной мультимедийной поддержкой содержит вредоносные аудиоданные — скрытые сообщения для вашей системы домашней автоматизации, призванные расширить ощущения от просмотра. Так, в какой-то момент в доме может громко хлопнуть дверь, отключиться свет или, возможно, даже отопление (чтобы вы немного похолодели от ужаса)...

Что ж, на этой слегка тревожной ноте перейдем к следующей главе, где рассмотрены основные понятия ГНС.

ГЛАВА 3

Основные понятия ГНС

В данной главе рассмотрены базовые концепции ГНС-моделей, а именно та разновидность машинно-обучаемых моделей, которая обычно используется для обработки изображений и аудиоданных. Понимание базовых идей позволит вам лучше усвоить дальнейшее содержание книги, где вредоносные образы рассматриваются более подробно. После вводной информации далее, в главе 4, рассматриваются модели интерпретации сложных изображений, аудио- и видеоданных. В двух главах изложено достаточно информации для понимания дальнейшего материала о вредоносных образах, но в то же время это не исчерпывающие сведения о глубоком обучении.

Если вы уже знакомы с принципами глубокого обучения и нейронных сетей, вы можете пропустить эту главу и главу 4. Если же, наоборот, хотите узнать больше, чем требуется для понимания этой книги, то в сети Интернет существует множество прекрасных ресурсов, с помощью которых вы сможете составить более четкое представление о машинном обучении и нейронных сетях. Ссылки на некоторые интернет-ресурсы можно найти в GitHub-репозитории этой книги (bit.ly/2x5Kg5I).

В конце этой главы приведены несколько фрагментов кода на языке Python. Как и во всех остальных главах этой книги, чтение кода является необязательным для понимания изложенного материала (в том числе описания вредоносных образов). Если же вас, наоборот, заинтересуют примеры кода, я рекомендую вам также скачать некоторые из блокнотов Jupiter, предоставленных в соответствующем GitHub-репозитории, и попробовать поэкспериментировать с ними.

Машинное обучение

Глубокие нейронные сети относятся к более обширной категории *машинного обучения*; это способность машин обучаться извлечению паттернов из данных без явного кодирования правил этих паттернов. Алгоритм, получаемый в результате такого обучения, называется *моделью*.

Модель представляет собой математический алгоритм, параметры которого скорректированы для обеспечения оптимального поведения. Алгоритм обучается за счет многократного представления ему обучающих данных, каждый отдельный фрагмент которых является примером того, чему необходимо обучиться. Каждый обучающий пример обеспечивает постепенное улучшение модели за счет коррекции ее параметров. После проведения надлежащей коррекции алгоритма он считается обученным. Затем обычно проводится оценка точности модели на тестовом наборе данных, отличном от обучающего набора данных. При этом ожидается, что модель будет хорошо решать определенную задачу в отношении данных, не содержавшихся в исходном обучающем наборе, что будет свидетельствовать о ее способности работать с незнакомыми данными.

Даже традиционное машинное обучение (без использования нейронных сетей) позволяет создавать модели для решения достаточно сложных задач. Например, с помощью обучающих данных, отражающих множество различных характеристик растений (таких как высота, форма листьев, окраска цветков и т. д.) и соответствующие таксономические категории, можно научить модель машинного обучения определять, к какому роду относится то или иное растение, на основе предоставленного списка характеристик. Хотя принадлежность растения к определенному роду может определяться сложным сочетанием характеристик, при использовании достаточного количества обучающих примеров и надлежащего метода машинного обучения полученная в итоге обученная модель сможет классифицировать растения, не требуя, чтобы программист явно кодировал (или даже понимал) взаимосвязь между различными характеристиками и родами растений.

Как уже упоминалось в главе 1, существует несколько основных стратегий обучения МО-моделей.

- *Обучение с учителем.* Пример с классификацией растений относится к категории «обучение с учителем», поскольку в данном случае модель обучается с помощью характеристик и соответствующих меток, отражающих правильный ответ (род).
- *Обучение без учителя.* Осуществляется без использования меток на этапе обучения. Модели не предоставляются точные ответы — она учится самостоятельно выявлять в данных паттерны, аномалии или связи. Модель, обучаемая с помощью данных о растениях без соответствующих меток, может научиться группировать растения на основе часто встречающихся сочетаний характеристик. После этого обученная модель сможет

(оценивая характеристики) определять, что новое растение относится к той или иной группе или, возможно, что новое растение не может быть отнесено к известным модели группам (то есть выявлять аномалию).

- *Обучение с частичным привлечением учителя.* Как вы могли догадаться, обучение с частичным привлечением учителя представляет собой что-то среднее между обучением с учителем и обучением без учителя, при котором лишь часть обучающих данных снабжена метками. Обычно сначала используются неразмеченные данные для выявления паттернов, а затем — размеченные данные, чтобы повысить точность модели.
- *Обучение с подкреплением.* При обучении с подкреплением МО-модель выдает некоторый фидбэк, позволяющий оценить ее эффективность. Модель корректируется для оптимального достижения определенной цели (такой как победа в видеоигре) путем многократных попыток достичь цели на основе фидбэка (например, количественного показателя эффективности) и внесением соответствующих поправок в используемый подход.

В большинстве случаев в данной книге будет идти речь о моделях, в которых используется обучение с учителем, поскольку именно в этой области пока сосредоточена большая часть исследований, посвященных вредоносным входным данным. Однако концепции вредоносных образов в той же мере справедливы и для моделей, использующих обучение без учителя, обучение с частичным привлечением учителя и обучение с подкреплением.

Концептуальные основы глубокого обучения

Какими бы мощными ни были традиционные методы машинного обучения, эти модели не способны обрабатывать очень сложные данные, когда не ясно, какая актуальная информация — то есть существенные признаки — требуется для решения задачи. Характеристики растений представляются с помощью структурированных данных — например, двоичное значение может указывать, есть ли у растения цветки, а целочисленное значение — сколько лепестков имеет каждый цветок. При этом не всегда ясно, насколько важен тот или иной признак для определения рода растения, но сами признаки выражены достаточно четко.

Для многих других задач используются неструктурированные данные или данные с трудноопределяемыми признаками. В сфере ИИ в фокусе внима-

ния часто находятся неструктурированные первичные данные, получаемые из реального мира, например изображения или аудиоданные. Цифровое изображение обычно содержит тысячи пиксельных значений, где каждый отдельный пиксель не представляет важности сам по себе; заключенный в изображении смысл основан на сложных пространственных взаимосвязях между пикселями. Подобно этому, аудиоданные представляют собой последовательность распределенных по времени значений, каждое из которых не представляет важности в отдельности. Звук интерпретируется тем или иным образом в зависимости от порядка следования этих значений и величины интервала между ними.

С задачей обработки визуальных и звуковых данных очень хорошо справляется биологическая нейронная сеть мозга людей и животных. Области мозга, отвечающие за обработку визуальной и звуковой информации, извлекают актуальную информацию из первичных данных, получаемых нами с помощью глаз и ушей. Обработка звуковых и визуальных данных дается нам настолько легко, что трудно понять, почему эта задача представляет сложности для машины.

В качестве простейшего примера рассмотрим распознавание кошки. В данном случае машине нужно определить границы объекта с учетом окклюзии и затенения, а затем распознать характерные для кошки признаки, что в целом очень сложно. В то же время для человека эта задача не представляет каких-либо трудностей. Мы можем распознать кошку, располагая минимальной информацией, — по форме хвоста или лапы или просто по характеру движения. Мы можем распознать кошку даже в том случае, если она относится к неизвестной нам породе.

Точно так же обстоит дело с аудиоданными. Люди могут распознавать звуки и понимать речь, не прилагая к этому каких-либо усилий. Мы отфильтровываем и игнорируем фоновый шум, фокусируясь на актуальных деталях. Интерпретируя речь, мы улавливаем сложные последовательности различных звуков и трактуем их в зависимости от контекста. Однако для компьютера это нетривиальная задача, поскольку цифровое содержимое звукового файла часто носит сложный и неупорядоченный характер, включая множество разнообразных помех.

Мы, люди, легко справляемся с обработкой визуальных и звуковых данных. При этом часто даже не можем объяснить, какие признаки и паттерны

характерны для кошки или какие сочетания звуков входят в то или иное предложение. Наш мозг каким-то образом запоминает эти признаки и паттерны, и мы просто применяем выученный алгоритм в повседневной жизни, считая это чем-то тривиальным. Однако написать код, который бы адекватно учитывал все сценарии, возможные в физическом мире, просто нереально, а стандартные алгоритмы МО недостаточно гибки для выявления признаков, позволяющих справиться с такой степенью сложности.

Если стандартные методы машинного обучения не всегда могут обеспечить надлежащую обработку аудиоданных и изображений, то методы глубокого обучения вполне справляются с этой задачей. Модели глубокого обучения отлично справляются с множеством сложных вычислительных задач, особенно когда используются неструктурированные данные (например, изображения, звуковые или текстовые данные) или данные с труднораспознаваемыми признаками (как, например, в случае предиктивного моделирования[1] с большим количеством переменных).

Когда в главе 1 вводилось определение искусственной нейронной сети (ИНС), она была схематично представлена в виде искусственных нейронов, приблизительно напоминающих биологические нейроны, и связей между ними, опять же очень приблизительно напоминающих аксоны и синапсы человеческого мозга. Это очень удобно, поскольку позволяет думать об искусственной нейронной сети как о множестве взаимосвязанных нейронов, обычно скомпонованных в ряд слоев[2], как показано на рис. 3.1. На нем представлена простейшая разновидность искусственной нейронной сети — *многослойный перцептрон*. В такой сети каждый узел одного слоя соединен со всеми узлами соседнего слоя, поэтому она относится к категории *полностью связанных*, или *плотных*, нейронных сетей.

Первый (см. рис. 3.1, *слева*) слой, который принимает входные данные, называется *входным слоем*. Результат алгоритма выдается на расположенный справа *выходной слой*. Между этими двумя слоями находятся так называемые *скрытые слои*, представляющие невидимые извне промежуточные результа-

---

1 Прогнозирование результатов на основе полученных ранее данных.

2 Как может заметить любой уважающий себя специалист в области науки о данных, искусственные нейронные сети не всегда скомпонованы в слои. Однако, чтобы не усложнять себе жизнь, не будем отвлекаться на всевозможные разновидности структуры ИНС.

ты производимых сетью вычислений. Именно здесь происходит вся магия последовательной обработки данных, и именно наличие одного скрытого слоя или более делает ИНС глубокой нейронной сетью.

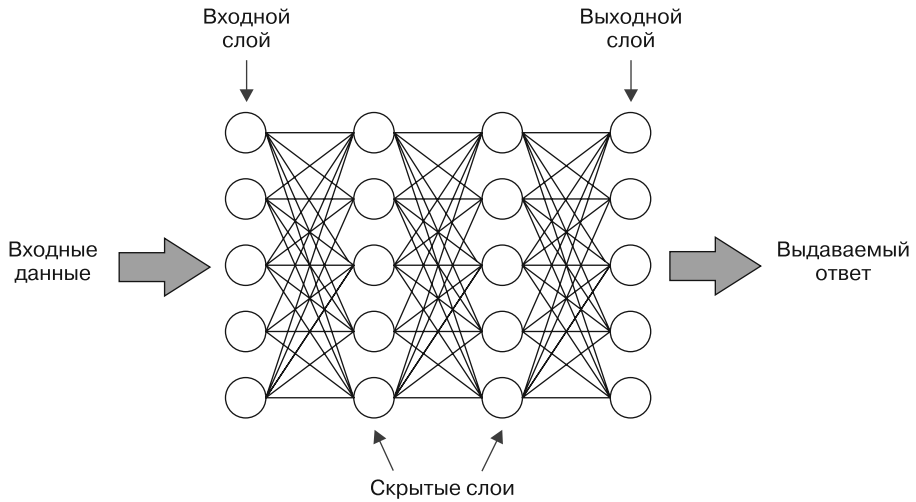


Рис. 3.1. Многослойный перцептрон

Эта идея представлена на рис 3.2. Данные поступают в ГНС через первый (см. рис. 3.1, *слева*) слой, в той или иной степени *активируя* нейроны этого слоя. На деле это означает, что нейронам присваивается значение, отражающее *степень активации,* или *интенсивность*. При этом высокой степени активации нейрона соответствует более высокое числовое значение, а низкой степени активации — более низкое значение.

Нейроны, сработавшие в одном слое, заставляют связи ретранслировать эту информацию в следующий слой; таким образом, степень активации каждого следующего слоя определяется степенью активации предыдущего слоя. Нейроны могут по-разному реагировать на входной сигнал; одни нейроны могут срабатывать (активироваться) при большей или меньшей интенсивности входного сигнала, чем другие. Кроме того, некоторые связи могут быть более сильными, чем другие, то есть обладать бóльшим весом и оказывать большее влияние на работу следующих слоев. Окончательный результат поступает в узлы расположенного справа выходного слоя. Такая сеть называется сетью *прямого распространения*, поскольку данные в ней

перемещаются в одном направлении. Информация всегда распространяется через сеть в прямом направлении, без какого-либо закольцовывания на предыдущие узлы. Если говорить более формально, то сеть прямого распространения является разновидностью направленного ациклического графа.

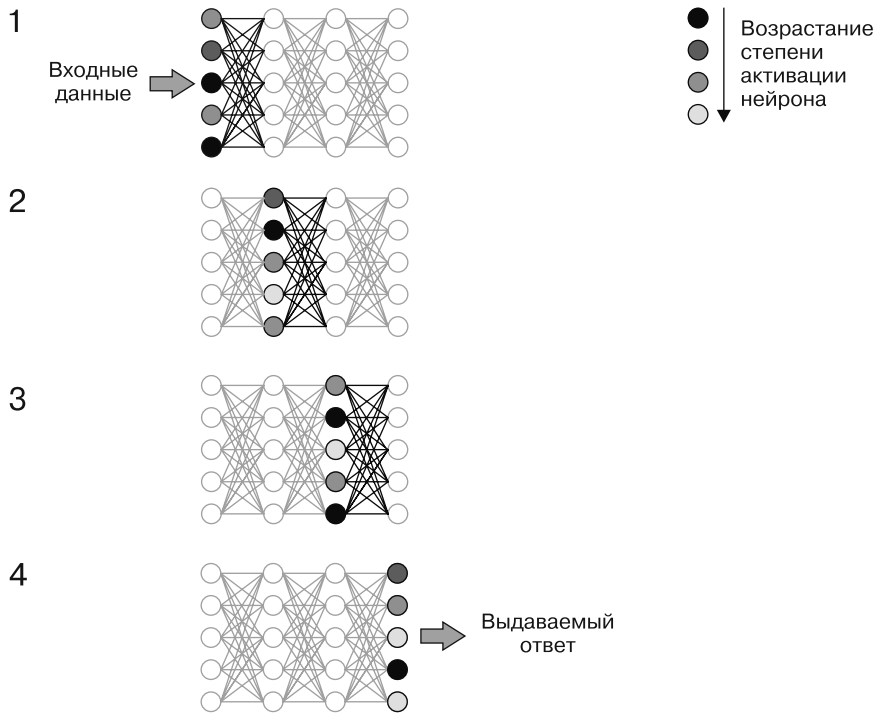


Рис. 3.2. Последовательные этапы вычислений в многослойном перцептроне

Конечно, по мере того как данные проходят через скрытые слои сети и преобразуются, например, из изображения в классификационную категорию или из речевого сигнала в соответствующий текст, происходит много интересного. Мы можем предположить (и будем правы), что скрытые слои постепенно извлекают из входных данных актуальную информацию (такую как изображение уха кошки или звук фонемы, соответствующей букве «а»). С каждым слоем извлекаются и потенциально совмещаются признаки все более высокого уровня, пока наконец не будет сгенерирован нужный ответ в выходном слое.

Модели ГНС как математические функции

До настоящего момента мы рассматривали ГНС как искусственную аппроксимацию биологических процессов (мозга), представленную исключительно концептуально (схематично) в виде нейронов и их взаимосвязей. Однако в действительности МО-модели (в том числе глубокие нейронные сети) представляют собой математические функции. Иногда эти функции могут быть довольно сложными, тем не менее любая МО-модель выражается математически.

Таким образом, не вдаваясь в подробности, можно сказать, что МО-модель — это математическая функция, которая принимает некоторые входные данные и возвращает некоторый результат. Этому определению соответствует следующая формула:

$$\mathbf{y} = f(\mathbf{x}).$$

Процесс обучения модели сводится к определению того, какой должна быть функция f.

Для знакомства с математическими основами глубоких нейронных сетей воспользуемся иллюстративным набором данных Fashion-MNIST. Этот набор данных можно скачать по адресу http://bit.ly/2WQqbKZ для проведения экспериментов и тестирования МО-моделей.

Универсальность искусственных нейронных сетей

Ключевое отличие ГНС-модели от обычных методов МО состоит в том, что ГНС может реализовать *универсальную* функцию. То есть существует нейронная сеть, способная реализовать точную аппроксимацию любой функции[1], какой бы сложной она ни была. Для того чтобы ГНС удовлетворяла принципу универсальности, ей достаточно иметь хотя бы один скрытый слой.

[1] Это справедливо только для непрерывных функций, у которых небольшое изменение входных данных не ведет к резкому ступенчатому возрастанию результата. Однако, поскольку прерывистую функцию часто можно аппроксимировать с помощью непрерывной функции, это, как правило, не является ограничением.

Набор данных Fashion-MNIST содержит 70 000 изображений в оттенках серого цвета с разрешением 28 × 28 пикселей. На каждом изображении представлен предмет одежды, который может быть отнесен к одной из десяти категорий: «Футболка/топ» (T-shirt/top), «Брюки» (Trouser), «Свитер» (Pullover), «Платье» (Dress), «Пальто» (Coat), «Сандалии» (Sandal), «Рубашка» (Shirt), «Кроссовки» (Sneaker), «Сумка» (Bag) и «Ботильоны» (Ankle boot). Примеры изображений из этого набора данных вместе с соответствующими метками представлены на рис. 3.3.

Рис. 3.3. Набор данных Fashion-MNIST содержит простые изображения предметов одежды вместе с соответствующими метками

Набор данных Fashion-MNIST особенно удобен для демонстрации классификации изображений с помощью ГНС-модели, поскольку он в значительной мере сокращает присущую изображениям вариабельность. Во-первых, изображения этого набора являются монохромными и имеют низкое разрешение. Это снижает степень вариабельности ГНС, необходимой для выполнения классификации, что позволяет легко обучить ее, не прибегая к использованию специализированного аппаратного обеспечения. Во-вторых, как видно на рис. 3.3, на каждой картинке представлен только один предмет одежды, который подогнан по размеру изображения и расположен по центру. Когда требуется выполнять пространственное позиционирование внутри изображения, необходимо использовать ГНС с более сложной архитектурой — об этом речь пойдет в главе 4.

Используя этот набор данных, мы можем поэкспериментировать с простейшей нейросетевой классификацией изображений: для некоторого входного изображения мы будем получать соответствующую категорию одежды. На рис 3.4 в общих чертах показано, как должна будет работать обученная ГНС при подаче на ее вход изображения брюк.

ГНС-модель является функцией f. При представлении хорошо обученной ГНС-модели f для классификации изображения брюк (обозначенное как x) должно быть возвращено значение y, указывающее на то, что это брюки.

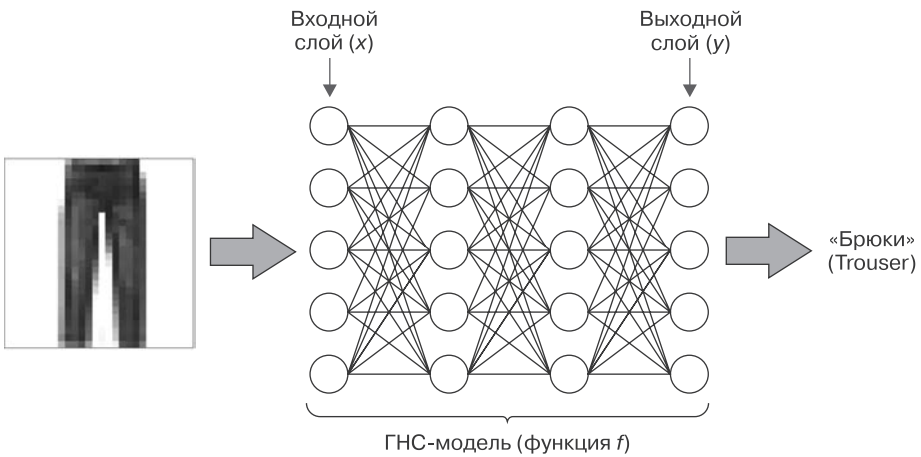


Рис. 3.4. В упрощенном виде схема работы модели для классификации изображений набора данных Fashion-MNIST выглядит следующим образом: она принимает изображение и возвращает соответствующую категорию

Не нужно быть математиком, чтобы понять: внутри этой ГНС-модели должно выполняться множество хитроумных вычислений, транслирующих изображение в одну из десяти категорий одежды. Сначала мы рассмотрим входные и выходные данные модели, а затем — ее внутреннее содержимое. Потом вы узнаете, как следует обучать модель для получения точных предсказаний.

Входные и выходные данные ГНС

Каждому нейрону входного слоя присваивается значение, отражающее некоторый аспект входных данных. Таким образом, во входных данных из набора Fashion-MNIST каждый нейрон входного слоя представляет одно пиксельное значение входного изображения. Поскольку каждое изображение содержит 28×28 пикселей, входной слой модели должен состоять из 784 нейронов. (Очевидно, что представленная на рис. 3.4 схема с пятью нейронами во входном слое сильно упрощена.) Каждому нейрону в этом слое присваивается значение от 0 до 255, отражающее интенсивность соответствующего пикселя[1].

Выходной слой представляет собой еще один ряд числовых значений, отражающий тот результат, который научилась генерировать ГНС. Обычно этот результат представляет собой ряд *предсказаний*, ассоциированных с каждой отдельной категорией. Поскольку в наборе данных Fashion-MNIST представлено десять категорий одежды, в выходном слое должно быть десять нейронов, содержащих десять числовых значений, каждое из которых будет отражать относительную вероятность принадлежности изображения к соответствующей категории одежды. Значение первого нейрона будет отражать вероятность принадлежности изображения к категории «Футболка/топ» (T-shirt/top), значение второго — вероятность принадлежности к категории «Брюки» (Trouser) и т. д.

Поскольку в данном случае решается задача классификации, ответ ГНС представляет тот нейрон выходного слоя, который обладает наибольшей степенью активации (наибольшим значением). В примере, показанном на рис. 3.4, при успешном выполнении классификации моделью наибольшей

[1] На практике полезно масштабировать эти значения таким образом, чтобы они находились в диапазоне от 0 до 1.

степенью активации будет обладать нейрон, соответствующий категории «Брюки» (Trouser).

Ряд входных данных (пиксельные значения) и ряд выходных данных (вероятности принадлежности к категориям одежды) можно математически представить в виде *векторов*. Соответственно, приведенную выше формулу лучше выразить в виде функции, принимающей вектор входных данных x и возвращающей вектор выходных данных y:

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_{n-1} \\ y_n \end{pmatrix} = f \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_{i-1} \\ x_i \end{pmatrix},$$

где i — количество нейронов во входном слое (784 для набора данных Fashion-MNIST), а n — количество нейронов в выходном слое (10 для набора данных Fashion-MNIST).

Таким образом, вы уже поняли, что такое математическая функция, представляющая ГНС. Эта функция принимает один вектор (входные данные) и возвращает другой вектор (выходные данные). Но что именно представляет собой эта функция?

Внутреннее содержимое ГНС и обработка с прямым распространением

Подобно выходному и входному слоям, каждый скрытый слой ГНС представляется в виде вектора. Каждый слой нейронной сети может быть представлен в виде простого вектора, содержащего по одному значению на каждый нейрон. При этом значения вектора, описывающего входной слой, определяют значения вектора первого скрытого слоя, которые, в свою очередь, определяют значения вектора следующего скрытого слоя и т. д., пока не будет сгенерирован выходной вектор, отражающий предсказания.

Так как же значения вектора одного слоя влияют на значения вектора следующего слоя? Чтобы разобраться в том, как ГНС-модель вычисляет ответ, посмотрите на укрупненную схему отдельного узла сети, выделенного

на рис. 3.5 большим кругом. Рассмотрим, как вычисляется его значение активации.

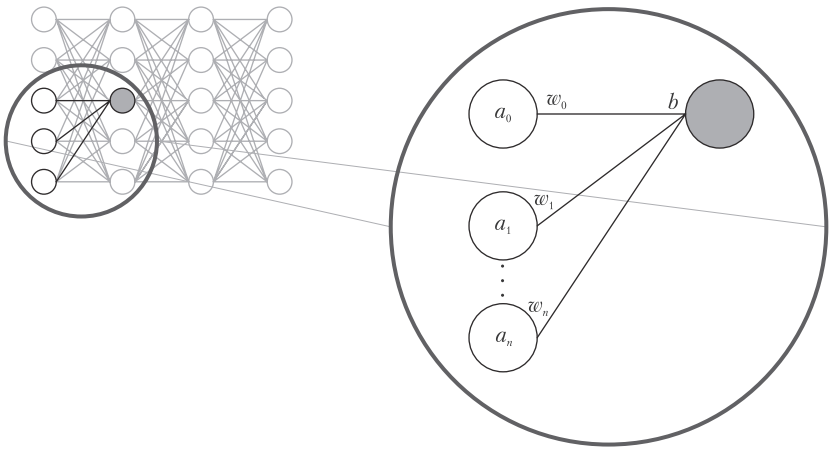


Рис. 3.5. Укрупненная схема нейрона в сети, иллюстрирующая его функцию активации

Значение отдельного нейрона определяется степенями активации нейронов предыдущего слоя, каждый из которых обозначен на схеме буквой *a* с нижним индексом, указывающим на номер нейрона.

Простейшая нейронная сеть прямого распространения имеет два типа настраиваемых параметров.

- *Вес*. С каждой связью сети ассоциировано отдельное весовое значение. Оно определяет силу связи или степень влияния активации текущего нейрона на следующий нейрон. На рис. 3.5 веса обозначены буквой *w* с нижним индексом, указывающим номер соответствующей связи.
- *Порог*. С каждым нейроном сети ассоциировано отдельное пороговое значение. Оно определяет, насколько нейрон активен. На рис. 3.5 порог обозначен буквой *b*.

Степень активации отдельного нейрона (расположенного за входным слоем) определяется совокупным влиянием входящих связей с поправкой на его порог. Влияние каждой отдельной связи равно произведению степени активации исходного нейрона на вес связи. На рис. 3.5 влияние первой

связи равно произведению веса этой связи на степень активации первого исходного нейрона:

$$w_0 \cdot a_0.$$

Теперь нужно сложить все составляющие, чтобы получить совокупный входной сигнал от всех входящих соединений:

$$w_0 \cdot a_0 + w_1 \cdot a_1 + ... + w_{n-1} \cdot a_{n-1} + w_n \cdot a_n.$$

И добавить порог, ассоциированный со следующим нейроном:

$$w_0 \cdot a_0 + w_1 \cdot a_1 + ... + w_{n-1} \cdot a_{n-1} + w_n \cdot a_n + b.$$

Чтобы добиться от ГНС требуемого поведения, результат этих вычислений необходимо передать *функции активации*. Обозначим функцию буквой *A*. В результате получим формулу для вычисления степени активации любого отдельного нейрона сети (за исключением нейронов входного слоя, степень активации которых определяется непосредственно входными данными):

$$A(w_0 \cdot a_0 + w_1 \cdot a_1 + ... + w_{n-1} \cdot a_{n-1} + w_n \cdot a_n + b).$$

При обработке данных в нейронных сетях могут использоваться функции активации различных типов. Конкретный тип функции активации явно определяется для каждого нейрона на стадии определения архитектуры сети и зависит от типа сети и конкретного слоя.

В частности, для скрытых слоев ГНС хорошие результаты показывает ректификационная функция активации (*ReLU* — Rectified Linear Unit). Скрытые слои нейронной сети наилучшим образом поддаются обучению, когда срабатывание нейрона могут вызывать лишь большие входные влияния, подобно тому, как синапсы человеческого мозга срабатывают при превышении некоторого порога. Функция ReLU действует по такому же принципу. Все достаточно просто — когда входной сигнал функции ReLU:

$$(w_0 \cdot a_0 + w_1 \cdot a_1 + ... + w_{n-1} \cdot a_{n-1} + w_n \cdot a_n + b) —$$

превышает статический порог (обычно равный нулю), функция возвращает это значение; в противном случае она возвращает ноль. Это отображено на рис. 3.6.

Когда ГНС выполняет задачу классификации, удобно, чтобы предсказываемые вероятности, получаемые на выходе последнего слоя, в сумме давали 1. Поэтому при классификации данных из набора Fashion-MNIST следует

использовать последний слой, который будет принимать вычисляемые сетью количественные показатели — так называемые *логиты* — и масштабировать их так, чтобы они представляли собой вероятности и, соответственно, в совокупности равнялись 1. Нейроны этого слоя будут выполнять масштабирование, используя другую разновидность функции активации, а именно *softmax*.

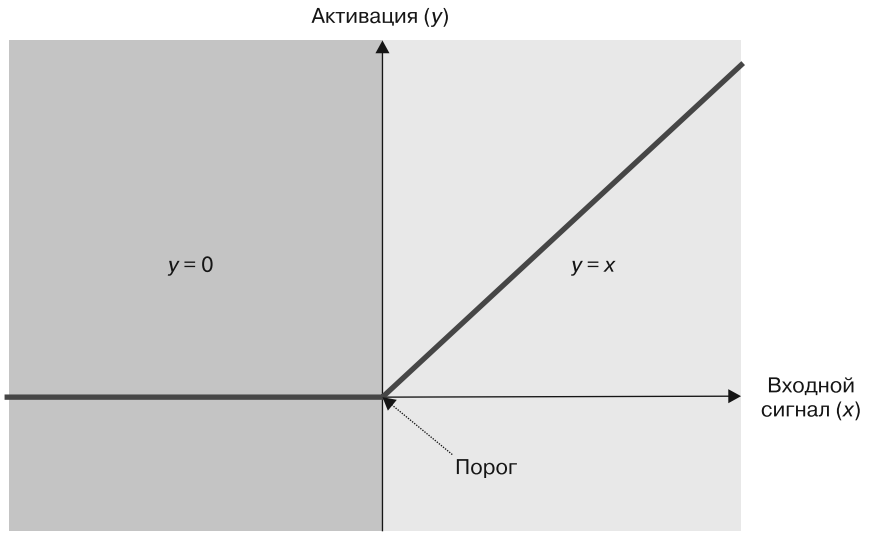


Рис. 3.6. Функция ReLU

Применив все вышесказанное к модели для классификации данных из набора Fashion-MNIST, можно внести определенные уточнения в рис. 3.4. Хотя мы можем использовать произвольное количество скрытых слоев и произвольное количество нейронов в каждом скрытом слое, в этом отношении есть некоторые эмпирические правила, соблюдение которых обеспечит хорошие результаты. Обычно достаточно использовать один скрытый слой, количество нейронов в котором находится в диапазоне между количествами нейронов во входном и выходном слоях. Допустим, в нашем случае используются два скрытых слоя по 56 нейронов в каждом. На рис. 3.7 показано, как при этом будет выглядеть архитектура сети.

Следующая часть головоломки: как производится коррекция имеющихся весов и порогов для обеспечения эффективной работы сети?

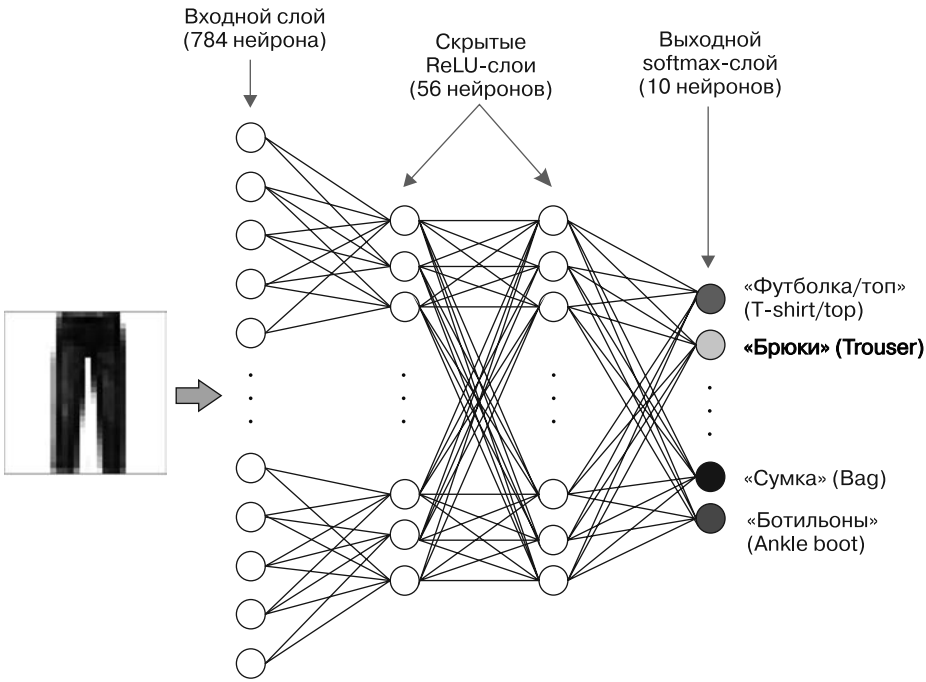


Рис. 3.7. Архитектура ГНС для классификации изображений с использованием набора данных Fashion-MNIST

Как обучается ГНС

Как мы успели выяснить, ГНС-модель представляет собой математическую функцию f. В подразделе «Входные и выходные данные ГНС» на с. 58 она выражена в виде функции, принимающей вектор входных данных и возвращающей выходной вектор. В упрощенном виде это можно записать так:

$$\mathbf{y} = f(\mathbf{x}),$$

где полужирное начертание символов $\mathbf{x}$ и $\mathbf{y}$ указывает, что это векторы.

Функция f включает в себя множество параметров, определяющих все веса и пороговые значения сети. Их необходимо должным образом скорректировать, чтобы сеть выдавала ожидаемый результат при любых входных данных. Поэтому перепишем предыдущую формулу, указав, что функция принимает

не только входное изображение, но и набор параметров, представляющих веса и пороговые значения. При этом символ Θ (тета) обозначает все значения весов и порогов сети:

$$\mathbf{y} = f(\mathbf{x}; \Theta).$$

Процесс обучения сети сводится к коррекции этих весов и порогов (Θ), чтобы при каждом представлении сети обучающих входных данных она возвращала значение, как можно меньше отличающееся от правильной метки классификации. Не стоит недооценивать задачу оптимизации всех этих параметров — функция ГНС включает в себя по одному пороговому значению на каждый скрытый нейрон и по одному весовому параметру на каждую связь. Таким образом, это дает следующую формулу для каждого слоя:

количество параметров на каждый слой =
*= (количество узлов в предыдущем слое * количество узлов в слое) +*
+ количество узлов в слое.

Например, для модели классификатора данных из набора Fashion-MNIST, представленной на рис. 3.7, требуется скорректировать 48 722 параметра — и это еще сравнительно простая нейронная сеть! Поэтому неудивительно, что для обучения ГНС требуются большое количество обучающих примеров данных и значительные вычислительные затраты.

Перед началом обучения веса и пороговые значения сети инициализируются случайными значениями. При этом сеть выдает очень плохие результаты — ей еще нужно пройти обучение. На этом этапе обучения производится многократная коррекция весов и порогов для оптимальной работы ГНС во всем диапазоне обучающих данных в надежде, что это обеспечит корректный результат и при предъявлении ей новых примеров.

Процесс обучения выглядит следующим образом. Сети многократно представляются примеры входных данных из обучающего набора; при этом каждый раз вычисляется количественный показатель отклонения сети от ожидаемых меток, ассоциированных с обучающими входными данными. Этот количественный показатель называют *штрафами* или *потерями* сети и вычисляют с помощью *функции штрафов*, или *функции потерь*, — специальной функции для количественной оценки того, насколько качественно сеть выполняет свою задачу. При больших штрафах сеть плохо справляется со своей задачей, а при малых штрафах она работает хорошо.

Функция штрафа принимает большой набор параметров, в том числе непосредственно функцию ГНС (f) со всеми ее весами и порогами (то есть всеми теми параметрами, которые обозначаются буквой Θ), а также обучающие примеры. Требуются и метки обучающих примеров, которые определяют, как должен выглядеть «хороший» ответ, то есть служат в качестве контрольных данных.

Соответственно, для одного обучающего примера функция штрафа может быть выражена следующим образом:

$$C(f(\mathbf{x};\Theta)\mathbf{l}),$$

где C — функция штрафа для ГНС f с параметрами Θ; $\mathbf{x}$ — представленный сети обучающий пример; $\mathbf{l}$ — соответствующее целевое предсказание.

Так что же в действительности делает эта функция C? Существует множество способов оценить эффективность работы модели при представлении ей одного обучающего примера. В простейшем случае можно просто вычесть ожидаемые метки из реальных значений, которые выдает ГНС для данного обучающего примера. Это разница обычно возводится в квадрат, чтобы в случае значительного расхождения между целевыми метками и предсказанными вероятностями генерировались непропорционально большие значения штрафа. Такой подход позволяет компенсировать большие расхождения между целевыми и предсказанными значениями.

Оценка штрафа для отдельного обучающего примера производится с помощью следующего выражения:

$$\left(\begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \\ y_n \end{pmatrix} = f\begin{pmatrix} l_0 \\ l_1 \\ \dots \\ l_{i-1} \\ l_i \end{pmatrix}\right)^2,$$

где $\mathbf{l}_i$ равно 1, когда i соответствует правильной целевой категории, и 0 в противном случае.

То есть штрафы для одного обучающего примера составляют:

$$C(f(\mathbf{x};\Theta)\mathbf{l}) = (f(\mathbf{x};\Theta) - \mathbf{l})^2$$

Допустим, что, производя обучение классификатора изображений из набора Fashion-MNIST, мы представили ему изображение брюк и он возвратил нам следующий вектор:

$$\begin{pmatrix} 0{,}232 \\ 0{,}119 \\ \dots \\ 0{,}151 \\ 0{,}005 \end{pmatrix}.$$

С этим изображением ассоциирована целевая метка «Брюки» (Trouser), которая соответствует второму значению вектора. В идеале это предсказание должно приближаться к 1, а не быть равным текущему значению 0,119. Если бы сеть работала идеально, она возвратила бы следующий вектор предсказаний:

$$\begin{pmatrix} 0 \\ 1 \\ \dots \\ 0 \\ 0 \end{pmatrix}.$$

Величина штрафа для данного образца равна квадрату разности между целевым вектором и вектором предсказаний:

$$\left(\begin{pmatrix} 0{,}032 \\ 0{,}119 \\ \dots \\ 0{,}151 \\ 0{,}005 \end{pmatrix} - \begin{pmatrix} 0 \\ 1 \\ \dots \\ 0 \\ 0 \end{pmatrix} \right)^2 = 0{,}001 + 0{,}776 + \dots + 0{,}023 + 0{,}000.$$

Как видите, в данном случае штрафы для категории брюк высоки. Чтобы исправить это, нужно скорректировать определенные параметры сети.

Чтобы по-настоящему оценить эффективность работы сети, нужно произвести оценку штрафа для *всех* обучающих примеров. Один из способов оценить общую эффективность — вычислить средний уровень штрафа. Это даст соответствующую функцию потерь. Так, в описанном выше случае необходимо

вычислить средний квадрат всех ошибок. Такая функция потерь называется *среднеквадратической ошибкой* (Mean Squared Error, MSE).

Другие функции потерь применяют различные алгоритмы расчета потерь на этапе обучения. Для моделей классификации обычно используется функция потерь на основе категориальной *перекрестной энтропии*. Такая функция с более высокой вероятностью лучше справляется с компенсацией ошибок сети. Применение разновидности этого будет продемонстрировано в примере кода в конце этой главы.

Для улучшения работы сети необходимо скорректировать ее веса и пороговые значения так, чтобы средний штраф снизился до минимально возможного уровня. То есть нужно скорректировать параметры Θ таким образом, чтобы выражение штрафа давало наилучший результат для всех возможных обучающих примеров. Это обеспечит оптимальную настройку сети. Такая коррекция производится с помощью метода *градиентного спуска*.

Чтобы понять, что собой представляет метод градиентного спуска, представьте, что у сети есть только один настраиваемый параметр Θ_i, а не тысячи таких параметров. Процесс настройки параметра можно представить графически, как показано на рис. 3.8. По оси X откладываются значения настраиваемого параметра. Для любого значения x можно вычислить среднюю величину штрафа сети, используя выбранную функцию потерь (например, MSE). Конечно, график на рис. 3.8 очень упрощен — для реальной сети нужны тысячи таких графиков, по одному для каждого параметра, — но в то же время он хорошо иллюстрирует общий принцип.

Каждая точка этой кривой представляет величину штрафа для конкретной комбинации весов и пороговых значений сети. В начале обучения, сразу после инициализации параметров случайными значениями, штрафы достаточно высоки, и нужно найти значения параметров, которые обеспечат снижение штрафа до минимального уровня.

Метод градиентного спуска сводится к вычислению с помощью математических методов градиента функции штрафа для текущей стадии обучения (то есть для текущих значений параметров Θ). На основе этого градиента оценивается необходимость в увеличении или уменьшении каждого веса и порога сети для улучшения ее работы. Многократное повторение этой процедуры позволяет оптимизировать параметры сети для получения хорошего результата. Как показано на рис. 3.8, вы как бы скатываетесь вниз по «склону» графика функции. При этом существует риск попадания в не самую лучшую

точку, обозначенную на графике как «хорошая оптимизация параметра». Спустившись в эту точку, вы застрянете в ней, поскольку для того, чтобы попасть из нее в точку наилучшей оптимизации параметра, необходимо сначала подняться вверх. Такая точка называется локальным минимумом. При использовании градиентного спуска применяются специальные методы снижения риска[1].

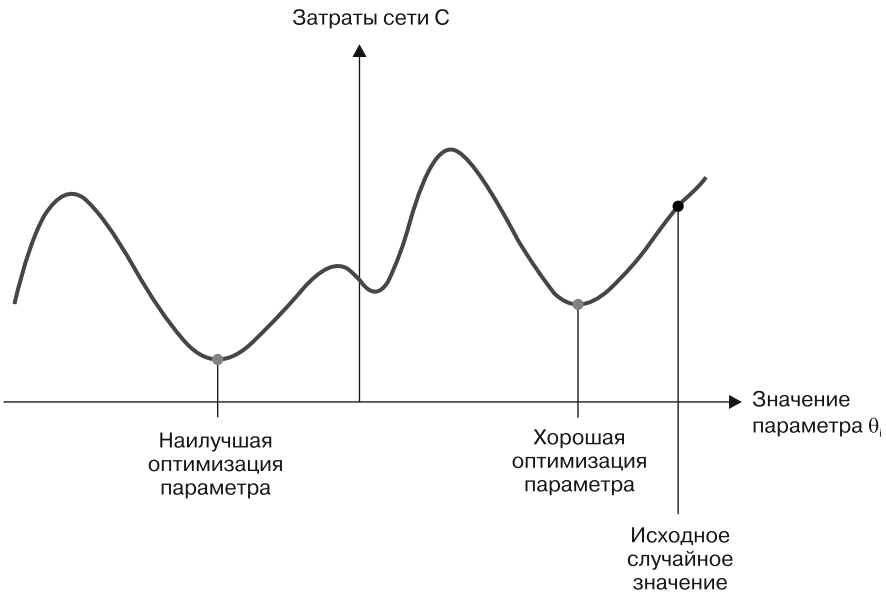


Рис. 3.8. При использовании градиентного спуска производится коррекция параметров сети для минимизации штрафа (потерь) на этапе обучения

Для расчета необходимых поправок в веса и пороги на этапе обучения ГНС применяется так называемый метод *обратного распространения ошибки*, который сводится к вычислению потерь для выходного слоя и «проталкиванию» их через сеть в обратном направлении с внесением соответствующих поправок в веса и пороги. В основе данного метода лежат сложные матема-

[1] В случае глубокого обучения есть еще одно интересное осложнение, связанное с тем, что несколько потенциально изменяемых параметров могут образовывать на графике штрафа седловые точки. Такие точки являются точками минимума для одного параметра и точками максимума — для другого. Наглядным примером такой точки является точка по центру седла для верховой езды, представляющая собой минимум в продольном направлении и максимум — в поперечном.

тические выкладки, однако они необязательны для понимания материала, касающегося вредоносных образов[1]. Но если вы хотите лучше разобраться в аспектах нейросетевой обработки как для обратного, так и для прямого распространения (о котором говорилось в предыдущем подразделе), вы можете воспользоваться ссылками на полезные ресурсы, добавленными в GitHub-репозиторий этой книги (http://bit.ly/2x5Kg5I).

Метод градиентного спуска играет важную роль в машинном обучении и часто применяется в этой сфере. Так, данный метод обычно используется для оптимизации моделей на этапе обучения, однако, как вы увидите в главе 5, к нему также прибегают и для оптимизации генерации вредоносных образов.

Создание простого классификатора изображений

Создание и обучение с нуля модели глубокого обучения — сложная задача, требующая понимания математических основ обработки с прямым и обратным распространением. Однако на сегодняшний день уже доступно множество библиотек для автоматического создания и обучения моделей, которые позволяют легко создать ГНС без необходимости кодировать базовые алгоритмы.

Блокнот Jupiter для простого классификатора изображений

Включенный в данную книгу программный код можно скачать из GitHub-репозитория книги (http://bit.ly/2ISaGgG).

Инструкции по доступу к программному коду, его запуску, настройке зависимостей и установке блокнота Jupyter см. в том же GitHub-репозитории (http://bit.ly/2ZxWnEI).

Фрагменты кода, приведенные в данном разделе, можно найти в следующем блокноте Jupyter: chapter03/fashionMNIST_classifier.ipynb (http://bit.ly/31JsseI).

В этом разделе показаны основные этапы программирования, необходимые для создания модели классификации изображений из набора данных

[1] В основе метода обратного распространения ошибки лежит цепное правило — математический метод, позволяющий оптимизировать функцию $f(\mathbf{x})$ на основе величины отдельных составляющих функции в каждом слое ГНС.

Fashion-MNIST на языке Python. Описываемая здесь сеть является примером того, насколько легко можно создать модель глубокого обучения с помощью открытых программных библиотек. Для создания и обучения модели использовалась библиотека глубокого обучения TensorFlow в сочетании с API Keras. Данный пример кода создан на основе обучающих материалов, предлагаемых в Интернете для ознакомления с Keras[1].

Сеть является *полносвязанной*, то есть каждый узел в каждом ее слое соединен с каждым узлом следующего слоя. Она также является сетью *прямого распространения*, то есть вычислительный процесс последовательно распространяется в ней от входного слоя через все скрытые слои до выхода из выходного слоя.

Прежде всего импортируйте необходимые библиотеки — TensorFlow и соответствующую версию Keras:

```
import tensorflow as tf
from tensorflow import keras
```

Вместе с библиотекой Keras предоставляются и данные из набора Fashion-MNIST. Дается два набора данных — один для обучения модели и один для ее оценки. Оба этих набора представляют собой список изображений с соответствующими метками. Наличие меток у тестовых данных позволяет оценить качество работы модели путем последовательного представления ей каждого тестового изображения и проверки результата на соответствие ожидаемой метке.

Полезно предусмотреть список названий категорий, соответствующий десяти возможным категориям одежды. Это позволит в дальнейшем выводить название категории, вместо того чтобы выводить только ее номер (например, T-shirt/top вместо 0):

```
fashion_mnist = keras.datasets.fashion_mnist

(train_images,train_labels),(test_images,test_labels) =
    fashion_mnist.load_data()

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

---

[1] Вы можете ознакомиться с исходным обучающим материалом на сайте библиотеки TensorFlow (http://bit.ly/2KXQnAW). Обучающий материал по библиотеке Keras (http://bit.ly/2WOw1MZ) может служить отличным базовым руководством по модели программирования Keras.

Посмотрим на изображение с индексом 9 на рис. 3.9:

```
import matplotlib.pyplot as plt
plt.gca().grid(False)
plt.imshow(test_images[9], cmap=plt.cm.binary)
```

Каждое изображение в наборе данных представляет собой массив пикселей, каждому из которых соответствует значение в диапазоне от 0 до 255, отражающее его интенсивность. Нужно нормализовать эти значения для входного слоя ГНС так, чтобы каждое из них находилось в диапазоне от 0 до 1:

```
train_images = train_images/255.0
test_images = test_images/255.00
```

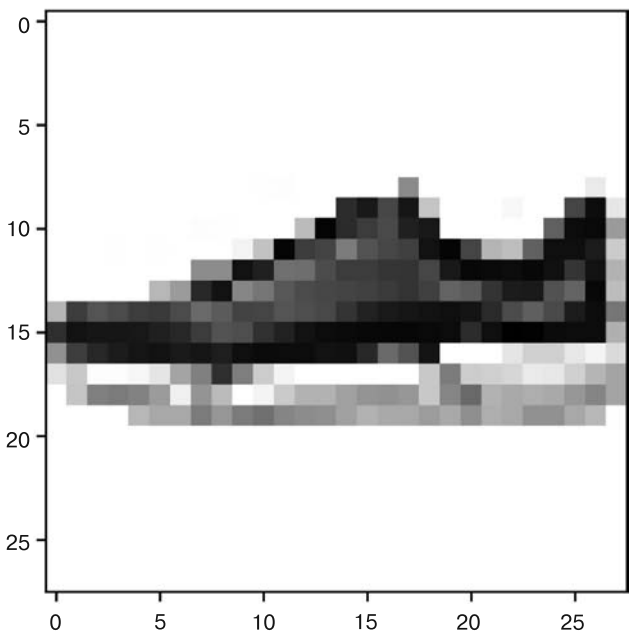


Рис. 3.9. Вывод программы

Программный интерфейс Keras предоставляет простой способ последовательного создания слоев модели с помощью соответствующих функций активации нейронов (в данном случае ReLU или softmax). Приведенный ниже код показывает, как можно создать модель с такой архитектурой, как показано на рис. 3.7. На этапе компиляции также указываются

переменные, которые определяют способ обучения модели и способ оценки ее точности:

```
model = keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                          keras.layers.Dense(56, activation='relu'),
                          keras.layers.Dense(56, activation='relu'),
                          keras.layers.Dense(10, activation='softmax')
                         ])
model.compile(optimizer=tf.keras.optimizers.Adam(), ❶
              loss='sparse_categorical_crossentropy', ❷
              metrics=['accuracy']) ❸
```

❶ Параметр `optimizer` определяет способ оптимизации сети на этапе обучения. В данном случае для ускорения обучения лучше выбрать оптимизатор `Adam`, использующий интеллектуальный алгоритм градиентного спуска.

❷ Здесь определяется описанная ранее функция потерь. Функция потерь `sparse_categorical_crossentropy` является вариацией функции `categorical_crossentropy`, используемой, когда целевые метки передаются в виде единого списка значений, а не как заполненный нулями массив, в котором соответствующее значение установлено в `1`. Такой способ представления возможен в том случае, когда верным ответом каждый раз является только одна категория. Например, метка, определяющая категорию *Pullover*, представляется в обучающем наборе данных не в виде массива `[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]`, а в виде числа `2`, поскольку это третья метка в списке, начинающемся с `0`.

❸ Параметр `metrics` определяет способ оценки сети на этапе обучения.

Теперь посмотрим на полученную модель и убедимся, что она выглядит нужным нам образом:

```
model.summary()
```

Данный код сгенерирует следующий вывод:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_4 (Dense)              (None, 56)                43960
_________________________________________________________________
dense_5 (Dense)              (None, 56)                3192
_________________________________________________________________
dense_6 (Dense)              (None, 10)                570
=================================================================
Total params: 47,722
Trainable params: 47,722
Non-trainable params: 0
```

Выглядит неплохо. Можно заметить, что общее количество параметров совпадает с тем значением, которое было вычислено в предыдущем подразделе.

Следуя принципу, изложенному ранее в этом разделе, мы можем обучить модель с помощью всего одной строки кода. Название используемой здесь функции `fit` указывает на то, что модель подгоняется под требования обучающих данных (*to fit* на английском языке означает «подгонять»). Параметр `epochs` задает количество итераций обучения, или эпох, то есть указывает, сколько раз будет корректироваться модель на основе уровня потерь, вычисляемого для всех обучающих примеров. Пусть этот параметр будет равен 6:

```
model.fit(train_images, train_labels, epochs=6)
```

Данный код сгенерирует следующий вывод:

```
Epoch 1/6
60000/60000 [================] - 4s 66us/sample - loss: 0.5179 - acc: 0.8166
Epoch 2/6
60000/60000 [================] - 4s 58us/sample - loss: 0.3830 - acc: 0.8616
Epoch 3/6
60000/60000 [================] - 3s 58us/sample - loss: 0.3452 - acc: 0.8739
Epoch 4/6
60000/60000 [================] - 4s 59us/sample - loss: 0.3258 - acc: 0.8798
Epoch 5/6
60000/60000 [================] - 4s 59us/sample - loss: 0.3087 - acc: 0.8863
Epoch 6/6
60000/60000 [================] - 4s 59us/sample - loss: 0.2933 - acc: 0.8913
Out[8]:
<tensorflow.python.keras.callbacks.History at 0x22f2b3b06d8>
```

Как видите, Keras отображает потери и точность модели на каждой стадии этапа обучения применительно к обучающим данным. Точность модели определяется как процент правильно классифицируемых ею примеров обучающих данных. С каждой эпохой потери модели снижаются, а ее точность возрастает — это градиентный спуск в действии! Веса и пороги модели корректируются для минимизации потерь (см. рис. 3.8).

Keras предоставляет показанный ниже метод для проверки точности сгенерированной модели после ее обучения. Поскольку нам нужно убедиться, что модель хорошо работает не только на тех данных, которые были предоставлены на этапе обучения, для этой оценки используется специальный тестовый набор данных (тестовые изображения с ожидаемыми метками):

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Model accuracy based on test data:', test_acc)
```

Данный код сгенерирует следующий вывод:

```
10000/10000 [================] - 0s 35us/sample - loss: 0.3623 - acc: 0.8704
Model accuracy based on test data: 0.8704
```

Как видно, точность модели на тестовых данных оказалась чуть ниже, чем в случае обучающих данных, что говорит о том, что модель слишком сильно подогнана под обучающие данные. Это пример так называемого *переобучения*, при котором модель слишком точно подогнана под обучающие данные и недостаточно обобщена для других данных. Несмотря на это, модель работает хорошо, возвращая почти 90 % правильных ответов на тестовом наборе данных.

Посмотрим, какие предсказания генерирует модель для конкретного изображения из тестового набора данных. Пусть это будет изображение с индексом 6. Вот код, который будет выводить на экран изображение и соответствующую ожидаемую метку (рис. 3.10):

```
image_num = 6
print("Expected label: ", class_names[test_labels[image_num]])

import matplotlib.pyplot as plt
imgplot = plt.imshow(test_images[image_num], cmap=plt.cm.binary)

Expected label:  Coat
```

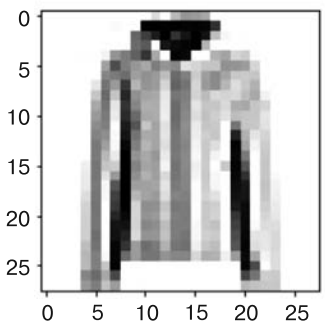


Рис. 3.10. Вывод программы

Функция `predict` библиотеки Keras позволяет сгенерировать предсказания для некоторого набора входных данных. Следующий код сгенерирует набор предсказаний для каждого из тестовых изображений и выведет на экран предсказания для выбранного нами изображения:

```
predictions = model.predict(test_images)
print("Predictions for image:", predictions[image_num])
```

Данный код сгенерирует следующий вывод:

```
Predictions for image: [2.0931453e-04 2.5958019e-05 5.3381161e-03
9.3024952e-05 9.8870182e-01 3.4905071e-08 5.4028798e-03 2.1192791e-10
2.2762216e-04 1.2793139e-06]
```

Результат представляет собой вектор, полученный в последнем слое модели для выбранного изображения. Этот softmax-слой гарантирует, что сумма всех выходных значений будет равна 1. Чтобы убедиться в этом, суммируем их:

```
total_probability = 0
for i in range(10):
    total_probability += predictions[image_num][i]
print(total_probability)
```

Данный код выдаст такой результат:

```
1.0000000503423947
```

Следующий код извлечет предсказание с наибольшей вероятностью и возвратит строку с соответствующей категорией одежды:

```
import numpy as np
index_of_highest_prediction = np.argmax(predictions[image_num])

print("Classification: ", class_names[index_of_highest_prediction])
print("Confidence:     ", predictions[image_num][index_of_highest_prediction])
```

Данный код сгенерирует следующий вывод:

```
Classification: Coat
Confidence: 0.9887018
```

Изображение было правильно классифицировано как «Пальто» (Coat), и, как видно, модель сделала это предсказание с достаточной долей уверенности.

Наконец, сохраним модель, чтобы ее можно было использовать позднее:

```
model.save("../models/fashionMNIST.h5") ❶
```

❶ Модель преобразуется в формат HDF5 (https://www.hdfgroup.org) и сохраняется в папке `models`.

Мы еще не раз вернемся к данному классификатору на протяжении этой книги по мере дальнейшего изучения вредоносных образов.

ГЛАВА 4

ГНС-обработка изображений, аудио- и видеоданных

В главе 3 для иллюстрации принципов нейросетевых технологий мы создали нейронную сеть, способную классифицировать простые изображения предметов одежды. Однако в силу предельной простоты данных из набора Fashion-MNIST приведенный пример нельзя назвать реалистичным. Глубокие нейронные сети могут решать более сложные задачи с использованием широкого спектра других архитектур, помимо простейшей полностью связанной сети прямого распространения.

Существует много способов проектирования ГНС с различным расположением слоев, разными типами слоев и способами соединения узлов. Выбор конкретного типа сети зависит от задачи; кроме того, сочетая разновидности ГНС друг с другом или с другими алгоритмами, можно решать еще более сложные задачи.

В этой главе рассмотрены способы применения ГНС-технологии для более реалистичной обработки изображений, аудио- и видеоданных. Опираясь на изложенные в главе 3 базовые понятия, вы узнаете, какие концепции лежат в основе нейронных сетей, обычно используемых в этой сфере, и как эти технологии можно сочетать с традиционными способами вычисления для получения лучших результатов.

Основное внимание в главе уделено двум ключевым типам сетей — *сверточным нейронным сетям* (СНС, англ. CNN — convolutional neural networks) и *рекуррентным нейронным сетям* (РНС, англ. RNN — recurrent neural networks). На их примере показаны общие правила обработки изображений и аудиоданных; кроме того, они упоминаются и в последующих главах. Хотя для обработки изображений, аудио- и видеоданных применяются не только СНС и РНС, но и другие типы глубоких нейронных сетей, именно эти два типа используются наиболее широко. Приведенные здесь описания носят лишь ознакомительный характер. Для получения дополнительной информации о том или ином типе сети обратитесь к ресурсам, указанным в GitHub-репозитории этой книги (http://bit.ly/2x5Kg5I).

Чтобы понять, каким образом глубокие нейронные сети могут быть использованы для обработки изображений и аудиоданных и как эти данные могут быть искажены для создания вредоносных образов, для начала следует рассмотреть вычислительное представление данных. Поэтому каждый раздел этой главы начинается с рассмотрения способа цифрового кодирования данных соответствующего типа (изображений, аудио- и видеоданных). В последующих главах вам станет ясно, каким образом наличие возможностей для создания вредоносных входных данных может напрямую зависеть от способа цифрового кодирования данных и особенно от его *точности*. Изменяя точность данных путем их сжатия или преобразования, вы можете (умышленно или неумышленно) удалять вредоносное содержимое.

К концу этой главы у вас будет достаточно информации о способах цифрового представления изображений, аудио- и видеоданных, а также распространенных шаблонов проектирования нейронных сетей для понимания материала последующих глав, где пойдет речь о вредоносных образах.

Изображения

На самом базовом уровне визуальное восприятие человека позволяет ему воспринимать свет, а точнее часть электромагнитного спектра с длинами волн в диапазоне от 200 до 700 нанометров (нм). Свет также можно описать с помощью частоты — величины, обратной длине волны. Частота измеряется в герцах (Гц) — количестве циклов в секунду; при этом свет находится в частотном диапазоне примерно от 430 до 750 терагерц (ТГц). Более низкие частоты соответствуют инфракрасному электромагнитному излучению, а более высокие частоты — ультрафиолетовому.

В наших глазах есть три типа цветовых рецепторов (колбочек), чувствительных к различным длинам волн светового излучения. Колбочки первого типа чувствительны к более длинным волнам видимого спектра (красному свету), колбочки второго типа — к волнам средней длины (зеленому свету), а колбочки третьего типа — к более коротким волнам (синему свету). Эти рецепторы посылают в мозг сигналы, когда в них попадает свет определенного цвета. Так, колбочка, чувствительная к синему краю спектра, посылает в мозг сигнал при попадании в глаз синего света; величина этого сигнала зависит от количества синего света. После этого зрительная зона коры головного мозга сводит сигналы воедино, формируя тот цвет, который мы воспринимаем.

Цифровое представление изображений

Цифровые изображения состоят из пикселей — дискретных значений, отражающих величину аналогового светового сигнала в соответствующих точках. Изображения из набора Fashion-MNIST довольно плохо отражают внешний вид реальных предметов. Во-первых, каждое изображение из набора Fashion-MNIST представлено множеством монохромных пиксельных значений, находящихся в диапазоне от 0 до 255. При этом значение 0 соответствует черному цвету (минимальной интенсивности), а значение 255 — белому цвету (максимальной интенсивности). Каждое пиксельное значение занимает ровно 1 байт памяти (поскольку с помощью 8 бит можно представить 256 различных значений).

Один из подходов к отображению цветных изображений сводится к тому, чтобы представлять 256 цветов, используя тот же объем памяти. Для этого нужно определить «цветовую палитру», то есть назначить каждому из 256 значений соответствующий цвет. Несмотря на довольно большие выразительные возможности, данный метод все же не обеспечивает достаточное разнообразие и широту диапазона цветов для адекватного отображения реалистичных изображений. Поэтому на фотографиях каждый пиксель обычно представляется *тремя* значениями — по одному для красной, зеленой и синей составляющих цвета. Для каждого из этих *RGB-значений* (red, green, blue — «красный, зеленый, синий») обычно выделяется 1 байт памяти, что обеспечивает диапазон от 0 до 255. При этом цвет отдельного пикселя определяется как комбинация соответствующих RGB-значений. Такой способ представления отличается большой гибкостью — RGB-значения в диапазоне от 0 до 255 обеспечивают палитру из 16 777 216 различных цветов. Вероятно, вы догадались, что идея формирования цвета из красного, зеленого и синего входных сигналов навеяна светочувствительными колбочками человеческого глаза.

Изображения из набора Fashion-MNIST не только являются монохромными, но и обладают очень низким разрешением — 28×28 пикселей; то есть каждое изображение содержит лишь 784 пикселя.

Для сравнения можно сказать, что даже фотография со сравнительно низким разрешением 0,3 мегапикселя (Мп) содержит примерно 640×480 пикселей. Современные камеры позволяют снимать фотографии с гораздо большим пиксельным разрешением (2 Мп и выше).

Точность изображения оценивается глубиной цвета (количеством битов, используемых для значения каждого пикселя) и количеством пикселей.

Эти два параметра соответственно называют *цветовым разрешением* и *пространственным разрешением*. Как показано на рис. 4.1, снижение пространственного разрешения ведет к меньшей плавности линий, а снижение цветового разрешения — к более «блочным» текстурам.

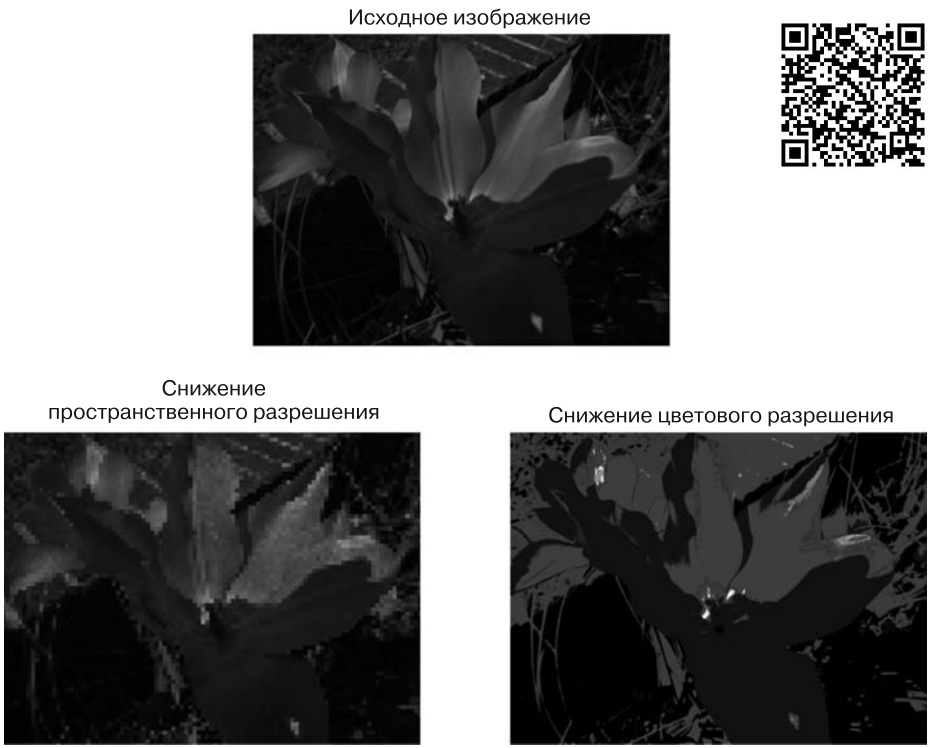


Рис. 4.1. Эффект от снижения пространственного и цветового разрешения изображения

Общая точность изображения в каждом конкретном случае зависит от того, как оно снято (от настроек камеры), целей его использования (например, размера при печати) и имеющихся ограничений памяти. Для обработки изображений можно использовать несколько распространенных графических форматов, однако лучше всего для этой цели подходит формат JPEG (стандартизированный объединенной группой экспертов по фотографии, Joint Photographic Experts Group), способный обеспечить и достаточный реализм при съемке фотографий, требующих большой глубины цвета (за счет смешивания RGB-составляющих), и высокое пиксельное разрешение.

ГНС для обработки изображений

Простейшим примером обработки изображений является классификация изображений по их основному содержимому, подобно тому как мы классифицировали изображения из набора Fashion-MNIST в главе 3. Однако обычно обработка изображений представляет собой нечто более сложное, например:

- *классификацию сцены*. Классификация сцены (по таким категориям, как «пляж», «улица» и т. п.) вместо проведения классификации по основному объекту;
- *обнаружение и локализацию объекта*. Примером может служить обнаружение на изображении лиц с определением их точного положения;
- *семантическую сегментацию*. Детализированное разбиение изображения на области, соответствующие различным категориям. Пример такого разбиения показан на рис. 4.2;

Рис. 4.2. Пример сегментации изображений[1]

- *распознавание лиц*. Глубокие нейронные сети могут использоваться в системах распознавания лиц. При достаточном объеме обучающих данных задача распознавания лиц становится обычной задачей классификации, в ко-

[1] Изображение получено с помощью нейросети SegNet, http://bit.ly/2ZyoOSQ.

торой количество категорий равно количеству людей, представленных в наборе данных. Систему распознавания лиц можно объединить с алгоритмом обнаружения, чтобы обеспечить и извлечение лиц из сцены, и их распознавание.

Некоторые задачи в течение многих лет решались с помощью традиционных методов обработки визуальной информации (без использования нейронных сетей). Так, например, традиционные методы обработки изображений позволяют обнаружить лица в изображении, выполнить извлечение и масштабирование, а затем преобразовать и нормализовать эти данные. После этого с помощью специальных алгоритмов можно определить положение *опорных точек*, выступающих в качестве ориентира при обнаружении и распознавании лица (например, уголки рта и глаз). Опорные размеры затем можно использовать для идентификации личности, что обычно делается с помощью статистических методов (в том числе методов машинного обучения). Глубокие нейронные сети расширили возможности и увеличили точность обработки визуальных данных, потому сегодня они постепенно приходят на смену традиционным методам обработки изображений или применяются в сочетании с ними.

Общие сведения о сверточных нейронных сетях

Основная сложность при обработке визуальных данных — правильная интерпретация содержимого изображения вне зависимости от его пространственного положения и размера. В главе 3 мы обошли эту проблему за счет того, что каждый предмет одежды в наборе данных Fashion-MNIST обладает примерно одинаковым размером, правильно сориентирован и расположен по центру изображения.

В реальной жизни такое единообразие в плане размера и расположения встречается крайне редко. Если ГНС вычисляет вероятность того, что на картинке изображена кошка, по определенному признаку — усы, то невозможно точно сказать, где этот признак усов будет находиться и какого размера он будет. Это зависит от размера, ориентации и позиции кошки. То есть нам нужно каким-то хитрым образом извлечь имеющиеся в изображении паттерны *вне зависимости от того*, где они расположены. После извлечения паттернов мы можем использовать их для более высокоуровневой обработки изображения.

Извлечение *пространственных* паттернов — это как раз та задача, с которой хорошо справляется сверточная нейронная сеть. Это означает, что

CHC наилучшим образом подходит для большинства задач обработки изображений.

CHC — это сеть, содержащая один *сверточный слой (или слой свертки)* или более — слой, который использует некоторый алгоритм извлечения из изображения признаков вне зависимости от их положения. Сверточный слой «сворачивает» один *фильтр* или более по всем участкам изображения. Это значит, что он производит некоторую фильтрацию на отдельном участке изображения, затем — такую же фильтрацию на соседнем, пересекающемся участке и т. д., пока не будет охвачено все изображение. Под сверткой при этом понимается перемещение фильтра по изображению с небольшим, обеспечивающим пересечение приращением, как показано на рис. 4.3.

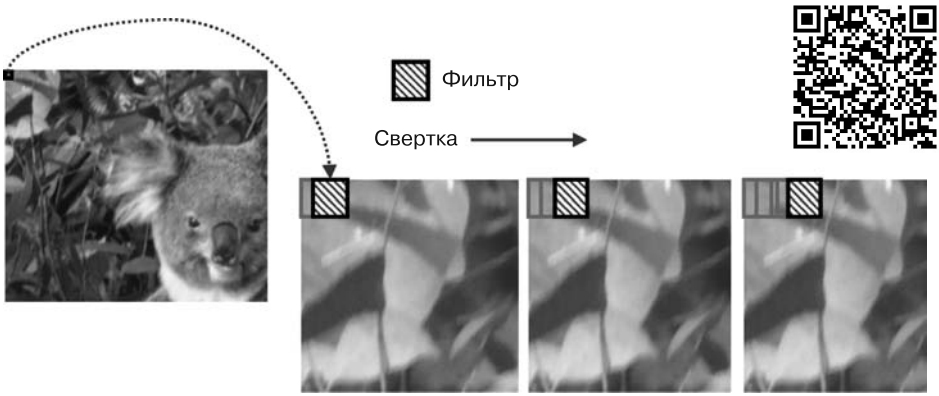


Рис. 4.3. Последовательное применение сверточного фильтра к изображению

При применении фильтра к отдельному участку изображения генерируется численное значение, определяющее, насколько точно данный участок изображения соответствует фильтру признаков.

Чтобы лучше понять, как это работает, рассмотрим простейший фильтр размером 3 × 3 пикселя, представляющий собой вертикальную темную линию на светлом фоне, как на рис. 4.4. Это очень простая матрица чисел. На рис. 4.4 представлены примеры применения этого фильтра к двум различным сегментам монохромного изображения размером 3 × 3 пикселя. Сначала фильтр признаков поэлементно умножается на сегмент изображения (то есть каждое значение одной матрицы умножается на соответствующее значение другой матрицы с получением в итоге новой матрицы). Затем вычисляется среднее значение элементов результирующей матрицы.

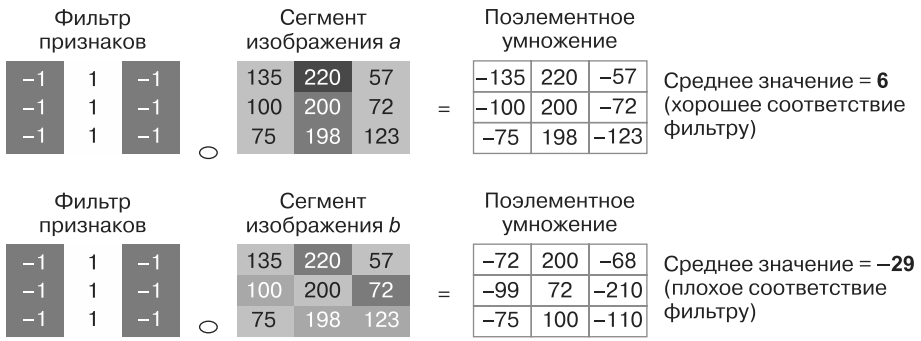


Рис. 4.4. Применение простого фильтра размером 3 × 3 к двум различным сегментам изображения

Как видите, значение, полученное для сегмента изображения *a*, сравнительно высокое — 6, что свидетельствует о том, что это небольшой фрагмент вертикальной линии. Для сегмента изображения *b* фильтр выдает значение –29, что совершенно верно свидетельствует о том, что данный участок не представляет собой вертикальную линию.

Все значения, полученные в результате свертки фильтра по изображению, помещаются в массив, подаваемый на выход из сверточного слоя. Это так называемая *карта признаков*.

На практике фильтры обычно намного сложнее, чем то, что показано на рис. 4.4, и их параметры оптимизируются в процессе обучения. Как и с весами и порогами, о которых говорилось в главе 3, фильтры обычно инициализируются случайными значениями, а затем обучаются. При этом сверточные слои выявляют в обучающих данных важные признаки и инкапсулируют их в картах признаков.

Здесь сделаем небольшую паузу и уточним, какие именно данные принимает и возвращает сверточный слой. Если сверточный слой находится в начале сети, то в качестве входных данных он принимает изображение. Если изображение цветное, представленное RGB-значениями — это трехмерный массив, одно измерение которого служит для представления высоты изображения, другое — его ширины и третье — его красного, зеленого и синего цветовых каналов. Для изображения размером 224 × 224 пикселя трехмерная «форма» данных, передаваемых нейронной сети, выглядит следующим образом:

```
shape = (224, 224, 3)
```

Сверточный слой может применять не один, а сразу несколько фильтров; при этом он будет возвращать целый стек карт признаков — по одной карте на каждый случай применения фильтра к входным данным. Это означает, что сверточный слой может генерировать больше данных, чем он потребляет! Чтобы сократить размерность выходных данных (и количество помех), за слоем свертки обычно размещается слой *пулинга* (или *подвыборки*). На этапе пулинга производится последовательный обход отдельных областей данных (подобно тому как производится перемещение сверточного фильтра по изображению). Из данных каждой области «выбирается» единое значение. Существует несколько подходов к выполнению этапа пулинга. Так, например, его можно выполнять путем вычисления среднего значения для каждой области данных либо путем определения максимального значения. Последний метод называется *макс-пулингом* — он производит эффект сжатия данных, а также несколько снижает точность фильтрации, если отбрасывание меньших значений не обеспечивает точного совпадения с фильтром.

Обычно начальная часть СНС представляет собой чередующуюся последовательность слоев свертки и пулинга. Для исключения отрицательных значений сюда также могут включаться слои с функцией ReLU; обычно они логически объединяются с операцией свертки. На рис. 4.5 показан один из возможных способов организации этих слоев.

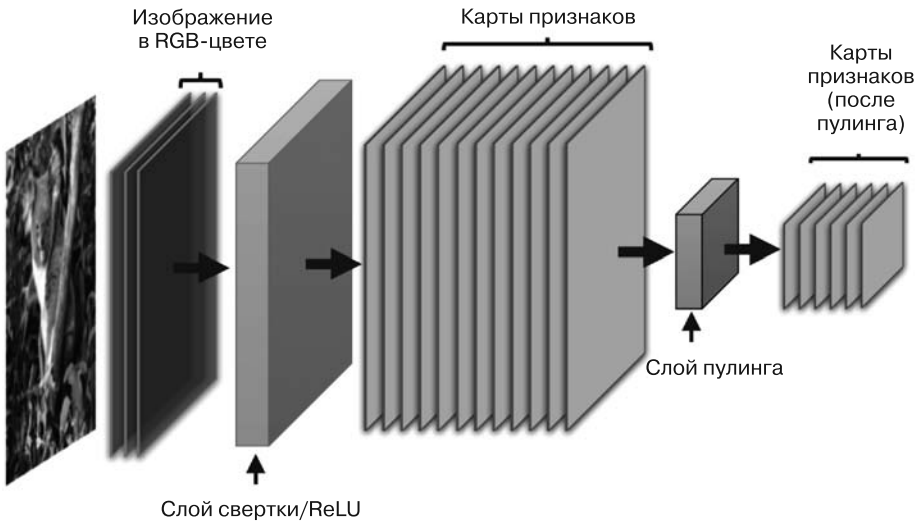


Рис. 4.5. Типичная схема расположения слоев в СНС

Как можно расположить сверточные слои для классификации изображений, показано на рис. 4.6. Представленная здесь СНС использует чередующуюся последовательность слоев свертки и пулинга для извлечения из изображения релевантных признаков. Затем извлеченные этими слоями признаки передаются в завершающую часть сети, которая содержит полностью связанные слои наподобие тех, что использовались в простейшем примере нейронной сети, рассмотренном в главе 3. Производимые здесь расчеты определяют окончательный вид выдаваемых сетью предсказаний и соответствующую категорию изображения.

Вероятно, вы уже начинаете понимать, что между слоями типичной ГНС передается множество данных, которые имеют вид многомерных массивов. Эти многомерные массивы называются *тензорами* (отсюда и название библиотеки *TensorFlow* в главе 3). Следует отметить, что представленная на рис. 4.6 схема является сильным упрощением; обычно ГНС для обработки изображений принимают не одно изображение, а сразу целый набор изображений.

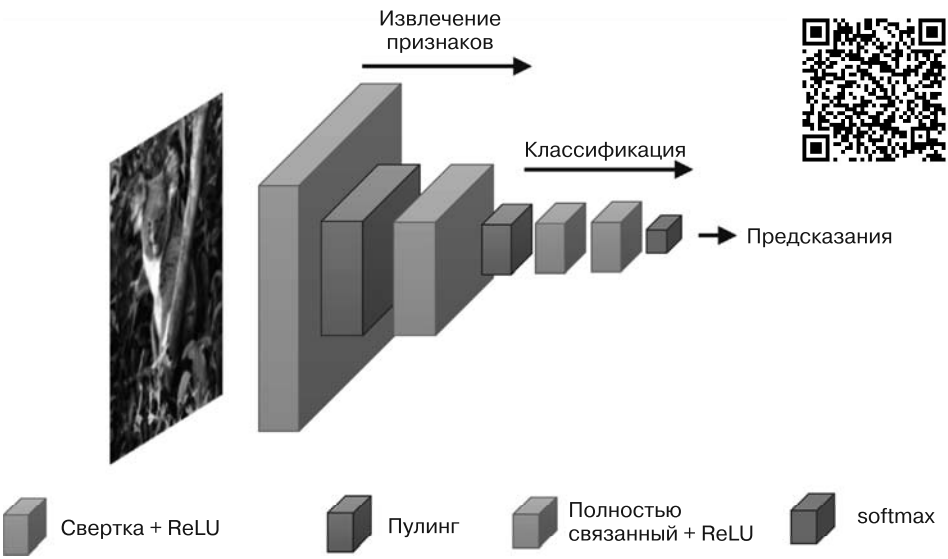


Рис. 4.6. Пример архитектуры СНС для классификации изображений

Такое пакетирование данных приносит пользу на стадии обучения, когда модель оптимизируется под большие объемы обучающих примеров, и, кроме того, упрощает обработку большого количества изображений при

тестировании и использовании модели. Таким образом, размерность входного тензора CHC будет равна 4, а не 3. Так, например, когда нейронной сети передаются 50 цветных изображений размером 224 × 224 пикселя, четырехмерный входной тензор имеет следующую форму:

```
shape = (50, 224, 224, 3)
```

Это означает 50 изображений по 224 × 224 пикселя и три цветовых канала.

Большинство нейронных сетей для обработки изображений включает в себя сверточные слои и, соответственно, классифицируется как CHC. В то же время их архитектура может сильно варьироваться. На самом деле одна из самых интересных задач в области ГНС-обработки в последние годы — создание новых архитектур, способных повысить точность, ускорить обучение или сократить объем занимаемой памяти. Было предложено много различных подходов к обработке изображений, некоторые из них описаны во врезке далее.

Некоторые примеры нейронных сетей для обработки изображений

Помимо прочих, существуют следующие подходы к обработке изображений.

- *VGG*. Сеть имеет очень простую архитектуру наподобие той, что представлена на рис. 4.6. Глубина сети при этом обозначается числом в ее названии (например, VGG16 имеет 16 слоев).

  Архитектура VGG была впервые описана Симоняном и Зиссерманом в статье «Очень глубокие сверточные сети для крупномасштабного распознавания изображений»[1]. На момент их разработки сети VGG16 и VGG19 считались очень глубокими, однако проблемой этой архитектуры является низкая скорость обучения и относительно большие размеры модели (по сравнению с более поздними архитектурами).
- *ResNet* (Residual Neural Network, остаточная нейронная сеть) представляет собой очень глубокую нейронную сеть. Архитектура призвана решить проблему «исчезающих градиентов» на этапе обучения за счет пропуска слоев в начале. Данная проблема состоит в том, что на этапе обучения градиент функции штрафа может стать настолько малым, что сеть перестанет обу-

[1] *Simonyan K., Zisserman A.* Very Deep Convolutional Networks for Large-Scale Image Recognition // ImageNet Large Scale Visual Recognition Challenge, 2014. http://bit.ly/2IupnYt.

чаться. Опять же число в названии каждой разновидности архитектуры ResNet обозначает количество используемых слоев.

Архитектура впервые описана Каймингом Хе и др. в статье «Глубокое остаточное обучение для распознавания изображений»[1].

- *Inception*. Первая реализация архитектуры Inception была разработана компанией Google и называлась GoogLeNet; эта сеть состояла из 22 слоев и имела примерно 4 миллиона параметров. Позднее компания Google улучшила эту архитектуру, создав ряд новых реализаций, а также разработала гибридную архитектуру Inception-ResNet.

  Архитектура Inception впервые описана Кристианом Шегеди и др. в статье «Углубление с использованием сверток»[2].

Аудиоданные

Звук — это воспринимаемая нашими ушами интерпретация волн давления, создаваемая в окружающей среде различными источниками вибрации. Как и любые другие волны, звуковые волны характеризуются их *амплитудой* и *частотой*.

Амплитуда волны отражает колебания давления и влияет на наше восприятие громкости звука. Как и свет, частота волны является величиной, обратной длине волны, измеряемой в герцах (Гц). Короткие звуковые волны обладают высокой частотой и высоким тоном, а длинные звуковые волны — низкой частотой и низким тоном. Большинство людей способны слышать волны в диапазоне от 20 до 20 000 Гц, и именно эти волны мы считаем звуком. Однако для животных или цифровых датчиков этот диапазон может существенно отличаться.

Простейший способ описания звуков посредством их амплитуды и частоты не позволяет отразить все особенности. Это хорошо иллюстрируют звуковые волны, показанные на рис. 4.7. Несмотря на явные различия, звуковые волны *a* и *b* имеют одинаковую амплитуду и основную частоту.

---

[1] *He K. et al.* Deep Residual Learning for Image Recognition // ImageNet Large Scale Visual Recognition Challenge, 2015. http://bit.ly/2x40Bb6.

[2] *Szegedy C. et al.* Going Deeper with Convolutions // ImageNet Large Scale Visual Recognition Challenge, 2014. http://bit.ly/2Xp4PIU.

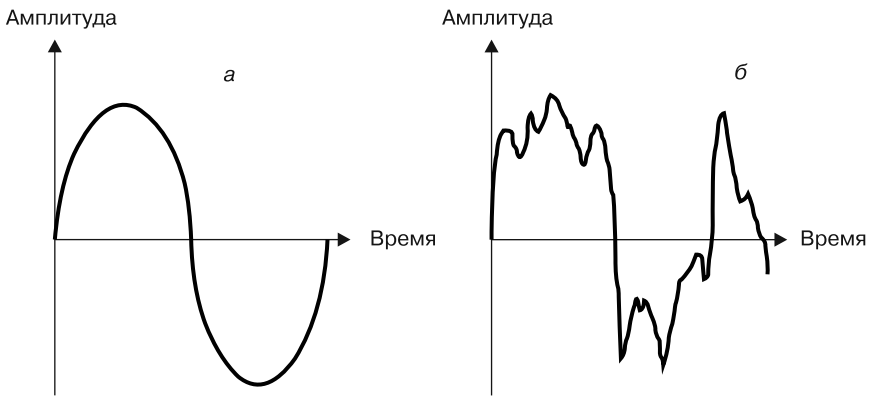


Рис. 4.7. Две сильно отличающиеся звуковые волны с одинаковой длиной волны (частотой) и амплитудой

Звуковая волна на рис. 4.7, *а* является примером «идеальной» звуковой волны, представляющей один тон. А звуковая волна на рис. 4.7, *б* — пример зашумленного звука, который мы можем слышать в повседневной жизни, — волны, формируемой множеством сигналов различной частоты от разных источников, которые отражаются от объектов и образуют сложные гармоники.

Цифровое представление аудиоданных

Преобразование аналоговых звуковых волн в цифровой формат осуществляется путем измерения интенсивности волны с регулярным интервалом дискретизации. Поэтому простейшее цифровое представление аудиосигнала — это последовательность чисел, отражающих интенсивность звуковой волны в каждый конкретный момент времени.

Здесь опять же не следует забывать о точности, поскольку от нее во многом зависит, будет ли у злоумышленника возможность для генерации вредоносных аудиоданных. Точность цифровых аудиоданных определяется двумя факторами: тем, насколько *часто* отбираются значения, и тем, насколько *точным* является каждое значение интенсивности. Это оценивается по частоте измерения аналоговых данных и точности кодирования каждого значения, то есть частоте дискретизации и битовой глубине (количеству бит, используемых для каждого значения) соответственно. Битовая глубина определяет точность измерения каждого звукового значения, подобно цветовому разрешению визуальных данных.

Частота дискретизации представляет собой временной эквивалент пространственного разрешения изображений применительно к аудиоданным. Это показано на рис. 4.8.

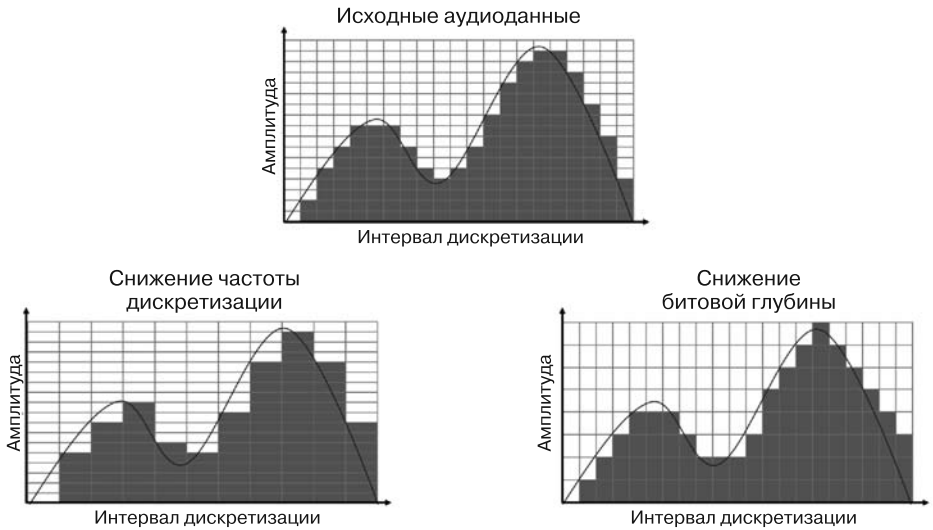


Рис. 4.8. Эффект от снижения частоты дискретизации и битовой глубины цифровых аудиоданных

ГНС для обработки аудиоданных

Как и в случае обработки изображений, традиционные методы обработки звука появились намного раньше, чем ГНС-технологии. Глубокие нейронные сети избавили нас от необходимости вручную определять характеристики звука. В то же время при нейросетевой обработке звука по-прежнему применяются традиционные методы для извлечения низкоуровневой информации, общей для всех задач.

Если не вдаваться в подробности, то при использовании нейросетевых технологий для обработки аудиоданных можно выбрать один из двух подходов.

- Можно произвести обучение сети с использованием сырых цифровых аудиоданных. Затем следует научить сеть извлекать признаки, относящиеся к конкретной задаче. Это позволяет получить очень точные модели;

однако обучение сети с использованием первичных неупорядоченных аудиоданных требует огромного количества данных, что часто делает такой подход трудноосуществимым.

- Можно дать нейронной сети определенную фору, выполнив предварительную обработку звука традиционными методами, а затем произвести обучение с помощью предварительно обработанных данных. Такой подход ведет к сокращению и упрощению этапа обучения, а также уменьшению объема необходимых для обучения данных.

Предварительная обработка аудиоданных обычно выполняется с использованием *преобразования Фурье* — очень изящного математического метода для разложения сигнала сложной формы, представляющего множество различных частот (наподобие показанного на рис. 4.7, *б*), на соответствующие частотные составляющие. Эту информацию затем можно представить в виде *спектрограммы*, отражающей изменение частотных составляющих с течением времени.

Для построения спектрограммы производится расчет амплитуды каждой частотной составляющей внутри следующих друг за другом (и, возможно, перекрывающихся) временных окон с использованием преобразования Фурье. Для графического представления спектрограммы производится объединение временных окон частотных составляющих с отражением значений интенсивности с помощью цвета; что может получиться в итоге, показано на рис. 4.9. Слева — общий принцип построения спектрограммы, а справа — пример реальной спектрограммы.

В некоторых случаях на этапе предварительной обработки могут извлекаться только актуальные для решаемой задачи аспекты звука с отбрасыванием всей остальной информации. Например, для более точного соответствия тому, как звуковые данные обрабатываются в слуховой системе человека, звуковой сигнал может быть представлен в виде *мел-частотного кепстра (MFC)*. Это особенно полезно при обработке, имитирующей человеческое восприятие (например, обработке речи).

Теперь рассмотрим базовую задачу обработки звука — классификацию. Эта задача является необходимым предварительным этапом для выполнения других, более сложных задач, например обработки речи. Как и в случае классификации изображений, данная задача сводится к тому, чтобы отнести звуковой клип к определенной категории в зависимости от его характеристик. Набор возможных категорий зависит от используемых тестовых данных

и области применения; это могут быть названия певчих птиц или типы источников звука, например «двигатель».

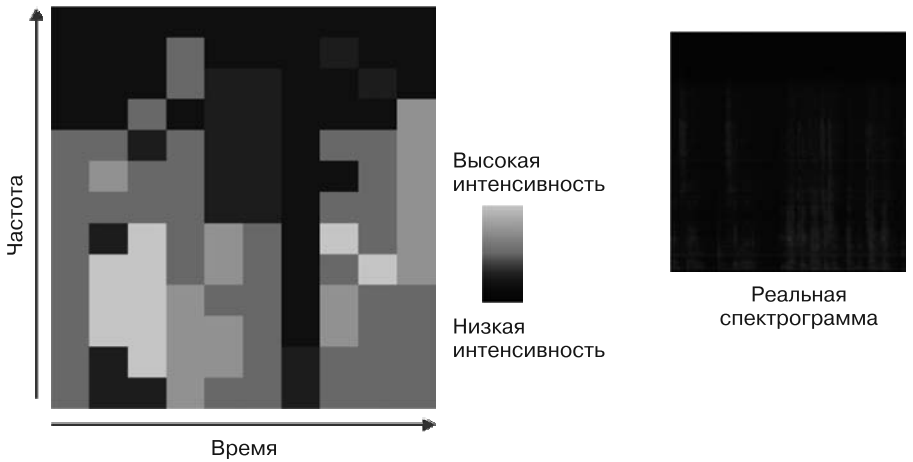


Рис. 4.9. Спектрограмма отражает изменение интенсивности по разным частотам с течением времени

Как решить эту задачу с помощью ГНС? Один из способов — просто видоизменить архитектуру СНС так, чтобы на вход подавалось изображение спектрограммы; при этом данная задача фактически превращается в задачу обработки изображений. Визуальные особенности спектрограммы будут указывать на наличие породившего их звука — так, например, лай собаки будет иметь соответствующие визуальные признаки, которые сможет извлечь архитектура СНС. Данное решение, по сути, сводится к преобразованию временного измерения в пространственное и обработке последнего с помощью сети.

Еще один способ заключается в том, чтобы использовать другую разновидность нейронной сети, способную работать с *последовательностями*. Речь идет о так называемой рекуррентной нейронной сети.

Общие сведения о рекуррентных нейронных сетях

Рекуррентные нейронные сети способны выявлять паттерны и зависимости в последовательных данных путем обработки каждого фрагмента данных с учетом того, что уже было обработано и что еще предстоит обработать.

Примером данных, значение которых определяется их последовательностью, может служить текст и такая информация с привязкой ко времени, как речь. Для выявления закономерностей в *последовательностях* данных нельзя использовать обычную ГНС или СНС прямого распространения, поскольку такие сети принимают во внимание только один фрагмент данных в отдельности. Однако РНС позволяет находить зависимости и паттерны в последовательных данных за счет сохранения определенных данных (или состояний) в промежутке между последовательными входными сигналами.

Самый простой архитектурный подход к РНС сводится к следующему: скрытый слой принимает в качестве входных данных не только выходной сигнал предыдущего слоя, но и свой собственный выходной сигнал. То есть выходной сигнал слоя подается вперед, в следующий слой (как в предыдущих примерах). Но, кроме этого, *снова* подается на его вход в качестве дополнительного входного сигнала для его текущих вычислений. Таким образом, предыдущий выходной сигнал скрытого слоя оказывает влияние на обработку текущего входного сигнала. На рис. 4.10 показана простая иллюстрация этого.

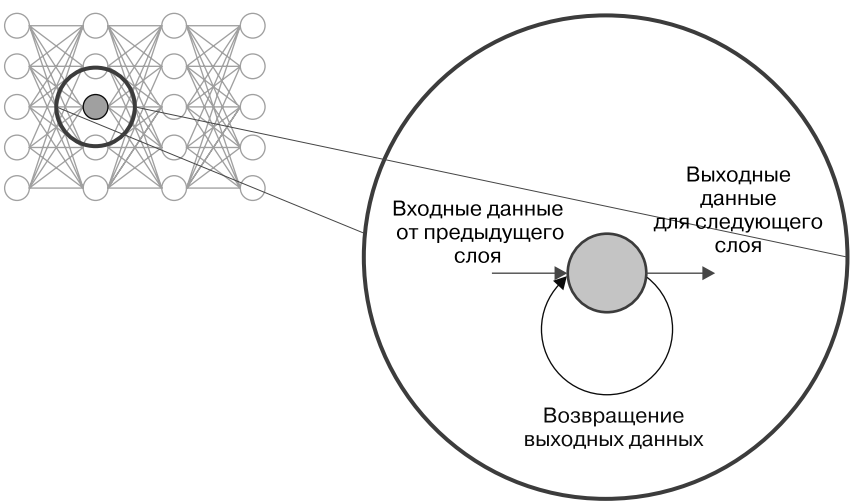


Рис. 4.10. Базовый принцип архитектуры РНС

Проблемой простых РНС является то, что они не позволяют выявлять паттерны, охватывающие длинные последовательности, то есть зависимости в данных, которые разделены между собой другими данными. Хорошим

примером такого случая является речь, где вероятность обнаружения той или иной фонемы может зависеть от того, какие звуки были произнесены несколько мгновений ранее. При обучении простой РНС градиент функции штрафа обычно исчезает или устремляется к бесконечности. Фактически этап математической оптимизации не может обеспечить правильное решение, которое бы учитывало взаимосвязи между широко разнесенными последовательными данными. Это не позволяет в достаточной мере минимизировать функцию штрафа, чтобы РНС хорошо выполняла свою задачу.

Для решения этой проблемы широко используется более сложная разновидность РНС — сеть *с долгой краткосрочной памятью* (Long Short-Term Memory, LSTM). Узлы такой сети выглядят сложнее по сравнению с тем, что показано на рис. 4.10. Они включают в себя компонент, сохраняющий состояние, и регуляторы потока данных (шлюзы). Эта более сложная архитектура позволяет и «запоминать» предыдущие данные последовательности, и «забывать» их, если они не представляют важности. Именно эта способность забывать делает возможной оптимизацию сети в процессе обучения.

Различные виды LSTM-сетей немного отличаются друг от друга компоновкой LSTM-модулей (например, в них могут использоваться разные шлюзы). LSTM-сети демонстрируют очень хорошие результаты для ряда задач, таких как распознавание рукописного текста и речи.

Архитектуру LSTM широко применяют для обработки звука, когда требуется распознавать распределенные по времени звуковые паттерны. Это особенно актуально для интерпретации звуковых сигналов, смысл которых зависит от содержания всех его частей (например, пение птиц или речь человека). На рис. 4.11 показано, как можно использовать РНС для обработки аудиоданных, предварительно преобразованных в спектрограмму. Предварительно обработанные данные о частотных составляющих представляют собой двумерный тензор (двумерную матрицу), одно измерение которого служит для представления амплитуды звука той или иной частоты, а другое — для представления времени[1]. В данном примере РНС генерирует такое же количество фреймов, каждый из которых представляет выходной сигнал РНС для определенного входного сигнала в последовательности, то есть один набор частотных составляющих звука для определенного момента времени генерирует набор вероятностей обнаружения тех или иных звуков.

---

[1] В данном случае обрабатывается один звуковой канал. Если звук состоит из нескольких звуковых каналов, добавляется третье измерение, представляющее глубину канала.

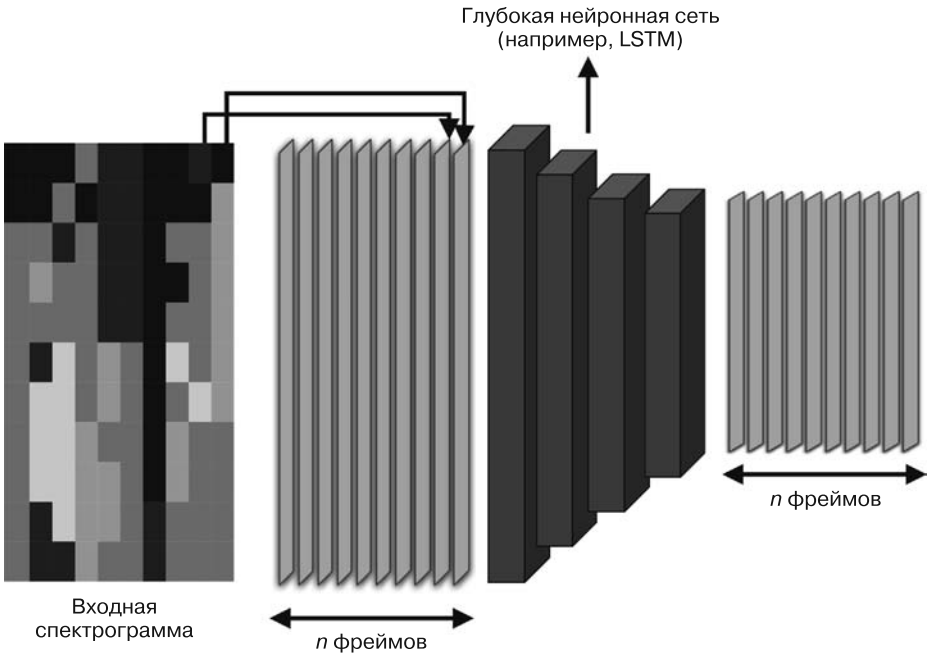


Рис. 4.11. Типичная последовательность обработки аудиоданных с предварительным преобразованием в спектрограмму

Обработка речи

Обработка речи — очень интересная область применения глубоких нейронных сетей, для которой актуальна тема вредоносных аудиоданных — мотивом для создания вредоносного звука является желание обмануть системы распознавания речи.

Распознавание речи представляет собой чрезвычайно сложную вычислительную задачу. Во-первых, нужно извлечь из данных звуки, образующие «строительные блоки» человеческой речи — *фонемы*. Люди могут нечетко выговаривать слова, говорить с разным акцентом и разной скоростью, низким или высоким голосом, и, кроме того, часто присутствует фоновый шум. Все это существенно затрудняет задачу правильной идентификации каждой отдельной фонемы. После извлеченные фонемы необходимо соотнести с реальными словами и предложениями. При этом возможно несколько вариантов такого отображения; выбор подходящего варианта зависит от языка и контекста.

При преобразовании речи в текст нейронная сеть обычно является составной частью более длинной последовательности обработки. Так, в частности, типичная последовательность преобразования речи в текст может выглядеть следующим образом.

1. LSTM-сеть принимает в качестве входных данных MFC-спектрограмму и выдает список текущих вероятностей для каждого из возможных символов текстовой системы. Этот этап представлен на рис. 4.11. В английском языке выдаваемые вероятности будут соответствовать буквам от «a» до «z» и символу межсловного пробела[1].

2. LSTM-сеть выдает распределения вероятностей с той же скоростью, с какой в нее поступают входные данные. Это серьезная проблема, поскольку иногда люди говорят очень медленно, а иногда — очень быстро. Таким образом, длина последовательности вероятностей соответствует длине входных аудиоданных, а не длине фонетической транскрипции. Для сопоставления входных аудиоданных с фонетической транскрипцией требуется дополнительный этап обработки.

   Широкое распространение получил метод *коннекционной временной классификации* (Connectionist Temporal Classification, CTC)[2]. Данный метод фактически сжимает и приводит в порядок вероятности обнаружения символов таким образом, чтобы из них получились допустимые фразы. Так, например, приняв высоковероятный выходной сигнал `_cc_aa_t`, метод CTC сгенерирует слово *cat*.

3. У нас уже есть список вероятностей для различных фонетических транскрипций, но пока нет окончательной фразы. Последний шаг состоит в том, чтобы взять наиболее вероятные «первичные» фразы и сопоставить их с расшифровкой, подходящей для используемого языка. Это не всегда возможно в силу того, что слова часто обладают одинаковыми характеристиками (например, в английском языке многие слова нельзя однозначно сопоставить с их фонетическим эквивалентом). Поэтому на данном этапе обработки обычно используется языковая модель, включающая в себя информацию о языке, вероятности различных последовательностей, произношении, грамматике и именах.

---

[1] Существует также отличный от пробела специальный пустой символ для представления пропусков в аудиоданных.

[2] *Graves A. et al.* Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks // Proceedings of the 23rd International Conference on Machine Learning (2006): 369–376. http://bit.ly/2XUC2sU.

Видеоданные

Видеоданные представляют собой сочетание движущихся изображений и звука. Для полноты картины в данном разделе кратко рассмотрены движущиеся изображения, о звуке говорилось ранее.

Цифровое представление видеоданных

Видеоданные представляют собой последовательность изображений, называемых *кадрами*. Соответственно, точность видеоданных определяется разрешением изображений и количеством кадров, снимаемых за секунду (частотой кадров).

ГНС для обработки видеоданных

Видеоданные можно анализировать, просто рассматривая каждое изображение в отдельности — во многих случаях этого будет вполне достаточно. Например, обнаружение и распознавание лиц может выполняться кадр за кадром, то есть путем поочередной подачи каждого кадра в нейронную сеть.

Вместе с тем дополнительное временное измерение также открывает возможности для интерпретации движения. Это позволяет углубить интерпретацию видеоданных до более сложной семантической интерпретации, такой, например, как:

- *отслеживание сущностей*. Отслеживание траекторий конкретных объектов (например, людей или транспортных средств) в течение какого-то времени. Это может включать в себя логический вывод позиционной информации в том случае, когда объект заслоняется чем-то другим или покидает сцену;
- *распознавание действий*. Углубление концепции распознавания объектов до обнаружения активности в пределах сцены с помощью дополнительной информации о движении. Примером может служить распознавание управляющих жестов устройства или обнаружение в пределах сцены определенного поведения (например, агрессии). Возможность распознавать действия в видеоданных позволяет выполнять и другую высокоуровневую обработку, такую как создание описаний для видеороликов.

Как и в случае изображений и звука, в сфере обработки видео тоже существуют традиционные подходы, в которых не используются нейронные сети.

Но опять же глубокие нейронные сети избавляют нас от необходимости вручную определять правила для выделения признаков.

Неудивительно, что временной компонент повышает сложность обработки кадров. В то же время существуют методы, позволяющие справиться с дополнительным измерением за счет использования архитектурных принципов, описанных ранее для СНС и РНС. Так, например, один из подходов сводится к *трехмерным сверткам*. Это расширение принципов свертки, используемых в СНС для обработки изображений, включающее в себя третье временное измерение смены кадров, которое обрабатывается таким же образом, как пространственные измерения в пределах каждого кадра. Еще один подход заключается в том, чтобы совместить пространственное обучение СНС с последовательным обучением РНС. Это можно реализовать с помощью СНС для извлечения признаков из каждого кадра с последующим использованием этих признаков в качестве последовательных входных данных для РНС.

Если для цветных изображений входные данные нейронной сети представляют собой четырехмерный тензор, то для видео это уже пятимерный тензор. Так, например, одноминутный видеоролик с частотой дискретизации 15 кадров в секунду будет в целом содержать 900 кадров. Допустим, что это видео с низким разрешением (224 × 224 пикселя), в RGB. Таким образом, четырехмерный тензор будет такой формы:

```
shape = (900, 224, 224, 3)
```

Если во входном пакете десять таких видеороликов, эта форма станет пятимерной:

```
shape = (10, 900, 224, 224, 3)
```

Соображения о вредоносности

В этой главе рассмотрены базовые сведения о существующих нейронных сетях с акцентом на тех разновидностях, которые обычно используются для обработки изображений, аудио- и видеоданных.

Некоторые аспекты нейронных сетей определяются разработчиком программного обеспечения, в то время как другие формируются в процессе обучения. Определяемые разработчиком аспекты — это *архитектура модели*, а аспекты обучения — *параметры модели*. Чтобы составить хорошее концептуальное представление о вредоносных образах, не обязательно понимать детали конкретных ГНС-архитектур и соответствующих параметров. Однако полезно знать, в чем различие между архитектурой модели и ее

обучаемыми параметрами, поскольку понимание архитектуры целевой сети и ее параметров играет важную роль при генерации вредоносных образов.

- *Архитектура модели*. На верхнем уровне архитектура модели определяет тип используемых в модели слоев и порядок их следования. Она также определяет заранее заданные аспекты конфигурации модели. В примере кода, рассмотренном в конце главы 3, мы определили количество слоев, типы слоев (ReLU и softmax) и размер каждого слоя. Для более сложных сетей, таких как CHC, потребуется принять дополнительные архитектурные решения, например, о размере и количестве фильтров в сверточном слое, типе пулинга и размере окна пулинга, а также величине шага свертки на каждом из этапов пулинга и свертки.
- *Параметры модели*. Наиболее очевидными параметрами модели являются веса и пороги, какие, например, мы определяли на этапе обучения в примере из главы 3. При наличии более сложных слоев обычно требуется обучить и ряд других параметров, например параметры каждого сверточного фильтра.

Архитектур, специально предназначенных для той или иной конкретной задачи, не существует, в то же время есть широко используемые архитектурные шаблоны, такие как сверточные слои или LSTM-модули.

Проведенные к настоящему моменту исследования вредоносных входных данных главным образом касались классификации изображений и (в меньшей степени) распознавания речи, однако такому обману могут подвергаться и другие аналогичные типы задач. Так, например, хотя вредоносные изображения чаще всего используются для обмана классификаторов изображений, вредоносные методы могут применяться и в отношении семантической сегментации, обнаружения и локализации объекта, поскольку это фактически расширение базовой задачи классификации[1]. Еще одним примером является распознавание лиц — генерация вредоносных входных данных для обмана систем распознавания лиц осуществляется практически так же, как генерация вредоносных изображений для обеспечения ошибочной классификации других объектов. В то же время часто возникают другие сложности — например, вредоносные искажения обычно трудно замаскировать на лице. В сфере аудиоданных доказана возможность применения вредоносных образов для распознавания речи; однако те же методы могут быть применены и для более простых задач (таких как голосовая верификация и классификация более простых аудиоданных).

---

[1] *Xie C. et al.* Adversarial Examples for Semantic Segmentation and Object Detection // International Conference on Computer Vision, 2017. http://bit.ly/2KrRg5E.

Классификация изображений с помощью сети ResNet50

Для иллюстрации классификации изображений в примерах данной книги используется модель ResNet50. Эта модель выбрана произвольным образом из числа доступных в Интернете моделей, в основном из-за ее компактности (она занимает лишь 102 Мбайт). При этом известно, что все остальные нейронные сети для классификации изображений, существующие на текущий момент, также уязвимы к вредоносным входным данным.

В этом разделе описывается последовательность действий по загрузке классификатора ResNet50 и генерации предсказаний для одного или нескольких изображений. Мы воспользуемся этими инструкциями позднее, когда будем тестировать вредоносные образы.

Фрагменты кода можно найти в файле chapter04/resnet50_classifier.ipynb в GitHub-репозитории этой книги (http://bit.ly/2IpkqQy).

Для начала нужно импортировать библиотеки TensorFlow и Keras, а также модель ResNet50:

```
import tensorflow as tf
from tensorflow import keras
from keras.applications.resnet50 import ResNet50

import numpy as np ❶

model = ResNet50(weights='imagenet', include_top=True) ❷
```

❶ Мы будем использовать NumPy для работы с изображением как с многомерным массивом.

❷ Эта команда создает модель ResNet50, обученную на наборе данных ImageNet. Выражение `include_top=True` говорит о том, что в данном случае используются заключительные слои нейронной сети, которые выполняют классификацию. Этот параметр предусмотрен в силу того, что классификационные слои не требуются, если модель используется исключительно для извлечения признаков.

Как и в случае классификатора Fashion-MNIST, полезно посмотреть, что представляет собой данная модель:

```
model.summary()
```

Эта команда сгенерирует следующий вывод (в силу большой глубины сети ResNet50 это очень длинная таблица, поэтому здесь показаны только строки с данными о начальных и заключительных слоях):

```
____________________________________________________________________________
Layer (type)                 Output Shape          Param #   Connected to
============================================================================
input_1 (InputLayer)         (None, 224, 224, 3)   0
____________________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 230, 230, 3)   0         input_1[0][0]
____________________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 64)  9472      conv1_pad[0][0]
____________________________________________________________________________
bn_conv1 (BatchNormalization) (None, 112, 112, 64) 256       conv1[0][0]
____________________________________________________________________________
activation_1 (Activation)    (None, 112, 112, 64)  0         bn_conv1[0][0]
____________________________________________________________________________
pool1_pad (ZeroPadding2D)    (None, 114, 114, 64)  0         activation_1[0][0]
____________________________________________________________________________
max_pooling2d_1 (MaxPooling2D) (None, 56, 56, 64)  0         pool1_pad[0][0]
____________________________________________________________________________
...
____________________________________________________________________________
activation_49 (Activation)   (None, 7, 7, 2048)    0         add_16[0][0]
____________________________________________________________________________
avg_pool (GlobalAveragePooling2(None, 2048)        0         activation_49[0][0]
____________________________________________________________________________
fc1000 (Dense)               (None, 1000)          2049000   avg_pool[0][0]
============================================================================
Total params: 25,636,712
Trainable params: 25,583,592
Non-trainable params: 53,120
```

Теперь получим изображение для классификации (рис. 4.12):

```
import matplotlib.pyplot as plt

img_path = '../images/koala.jpg' ❶
img = image_from_file(img_path, (224,224)) ❷
plt.imshow(img)
```

❶ Подборку изображений вы найдете в репозитории, но можете использовать и собственное изображение.

❷ `image_from_file` — это простая вспомогательная утилита, которая открывает изображение и масштабирует его до размера 244 × 244 пикселя перед подачей в классификатор. Код этой функции не приводится здесь для краткости, однако она включена в виде Python-утилиты в GitHub-репозиторий.

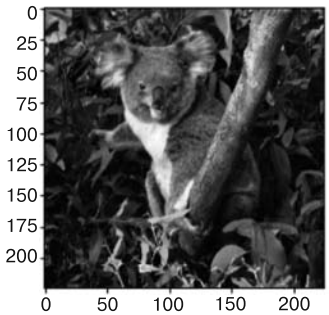


Рис. 4.12. Вывод программы

Перед подачей изображения на вход сети ResNet50 его необходимо предварительно обработать. Функцию, выполняющую предварительную обработку, предоставляет библиотека Keras (`preprocess_input`):

```
from keras.applications.resnet50 import preprocess_input
normalized_image = preprocess_input(img)
```

Этап предварительной обработки подготавливает изображение таким образом, чтобы оно было представлено в том же формате, что и изображения, применявшиеся для обучения сети. Как он будет выглядеть, зависит от используемой модели. Для модели ResNet50 функция `preprocess_input` выполняет такие преобразования изображения, как:

- *нормализация* — вычитание RGB-значений, усредненных по всему набору обучающих данных, центрирует данные вокруг нулевого среднего значения для каждого из каналов. Такая нормализация обучающих данных повышает скорость обучения сети. Последующие тестовые данные должны проходить такую же нормализацию;
- *изменение порядка следования каналов* — модель ResNet50 обучена на изображениях с порядком следования каналов BGR, а не RGB. Если изображение представлено в формате RGB, необходимо изменить порядок следования каналов.

Подробное описание шагов предварительной обработки вы найдете в файле chapter04/resnet50_preprocessing.ipynb (http://bit.ly/2WO9rUx) в GitHub-репозитории этой книги.

Теперь мы можем передать нормализованное изображение классификатору:

```
normalized_image_batch = np.expand_dims(normalized_image, 0) ❶
predictions = model.predict(normalized_image_list)
```

❶ Классификатор принимает набор изображений в виде массива `np.arrays`, поэтому функция `expand_dims` должна добавить к изображению дополнительную координатную ось. В данном случае создается набор, содержащий одно изображение.

После этого последний слой классификатора выдаст нам вектор предсказаний — большой массив, отражающий степень вероятности для каждой категории. Мы можем легко вывести первые три категории вместе с соответствующими предсказаниями, используя для этого следующий код:

```
from keras.applications.resnet50 import decode_predictions

decoded_predictions = decode_predictions(predictions, top=3) ❶

predictions_for_image = decoded_predictions[0]
for pred in predictions_for_image:
    print(pred[1],' : ', pred[2])
```

❶ `decode_predictions` — это удобная вспомогательная утилита, которая извлекает из массива `predictions` наибольшие показатели (в данном случае извлекаются первые три показателя) и сопоставляет их с соответствующими метками.

Вот как выглядит результат:

```
koala : 0.999985
indri : 7.1616164e-06
wombat : 3.9483125e-06
```

Как видно, модель ResNet50 просто прекрасно справилась с классификацией! Далее в книге будет показано, что после внесения в то же изображение незначительного искажения данная модель уже значительно хуже справляется со своей задачей.

ЧАСТЬ II

Генерация вредоносных входных данных

Из части I вы узнали, что представляют собой вредоносные входные данные и из каких побуждений они могут создаваться, а также изучили основные принципы глубокого обучения применительно к изображениям и аудиоданным. В части II вы познакомитесь с математическими и алгоритмическими методами генерации вредоносных данных.

Для начала в главе 5 дано концептуальное разъяснение идей, лежащих в основе генерации вредоносных данных. Из этой главы вы узнаете, почему глубокие нейронные сети можно обмануть, внеся во входные данные небольшие изменения, не влияющие на восприятие изображения или звука человеком. А также каким образом можно математически оценить величину изменений, достаточных для того, чтобы сделать входные данные вредоносными, и как особенности человеческого восприятия могут влиять на возможность внести незаметные изменения в изображения и аудиоданные.

В главе 6 в продолжение темы разъясняются конкретные вычислительные методы генерации вредоносных входных данных на основе проведенных в этой области исследований. Вы изучите математические основы некоторых методов, а также узнаете различия между ними. Для иллюстрации этих методов приведено несколько примеров кода, использующих нейронные сети, представленные в главах 3 и 4.

К концу этой части вы будете знать, почему можно обмануть глубокие нейронные сети, а также какие принципы и методы применяются для осуществления этого обмана. Эти знания послужат основой для изучения реальных угроз, описанных в части III.

ГЛАВА 5

Базовые принципы вредоносных входных данных

В этой главе рассмотрены основные принципы генерации вредоносных образов. Отложим подробное рассмотрение математических основ и конкретных методов на потом, а пока продолжим развивать идеи, представленные в предыдущих главах. При этом будем использовать аналогии и приближения для обеспечения интуитивного понимания, перед тем как углубиться в детали. Наша цель — в общих чертах понять, каким образом добавление вредоносного искажения или вредоносной заплатки может заставить ГНС возвращать неверный результат.

Вкратце напомню изложенное ранее.

- *Вредоносное искажение*. Сочетание распределенных по входным данным незаметных (или почти незаметных) незначительных изменений, заставляющее модель возвращать неверный результат. Для изображения это могут быть небольшие изменения ряда пикселей, распределенных по всему изображению.
- *Вредоносная заплатка*. Дополнение к определенной (пространственной или временной) области входных данных, заставляющее модель возвращать неверный результат. Вредоносная заплатка обычно вполне заметна для человека, но часто камуфлируется под нечто неопасное.

В этой главе показано, как можно генерировать вредоносные искажения и заплатки путем непосредственного манипулирования цифровыми данными. Хотя вредоносные искажения и заплатки легче применять непосредственно к входным данным в цифровой форме, их часто можно применять и в физическом мире (путем изменения дорожных знаков, например, для автономных транспортных средств), заставляя датчик (камеру или микрофон) генерировать цифровые входные данные, производящие желаемый вредоносный эффект. В главе 8 вы узнаете, с какими проблемами сталкивается злоумышленник, когда у него нет доступа к цифровой форме входных данных.

Вредоносные атаки можно разделить на две основные категории.

- *Нецелевые атаки.* Нецелевая (или неизбирательная) атака производится с целью заставить ГНС возвращать неверный результат, например совершать ошибку в классификации. Пример такой атаки — попытка обмануть систему распознавания лиц — при этом неважно, какой именно результат выдаст ГНС; главное, чтобы в изображении не было распознано лицо конкретного человека.
- *Целевые атаки.* Целевая атака производится для того, чтобы заставить ГНС сгенерировать некоторый конкретный результат, например заставить автономное транспортное средство не распознать знак остановки.

Неудивительно, что осуществить нецелевую атаку легче, чем целевую, поскольку при этом не так важен выдаваемый нейросетью результат, а это дает больший простор для манипуляций с входными данными. В то же время в обоих случаях применяются практически одни и те же методы.

Прежде чем перейти к непосредственному рассмотрению атак, посмотрим, что представляют собой исходные входные данные ГНС и извлекаемые из них признаки — те характеристики, на основе которых модель принимает решение. Для целей данной главы применяется классификация изображений — наиболее хорошо исследованная область применения вредоносных образов. Однако представленные здесь концепции справедливы не только для обработки изображений; эти идеи в равной мере применимы и в других сферах, например в области обработки звука.

Освежите свои знания по математике

На тот случай, если вы незнакомы с применяемыми в этой книге математическими обозначениями (или успели их забыть), в приложении приводятся примеры их использования с соответствующим кратким описанием.

Входное пространство

Глубокие нейронные сети представляют собой обучаемые функции, отображающие некоторый сложный входной сигнал на результат. В главе 3 рассмотрен пример простой задачи классификации изображений с использованием набора данных Fashion-MNIST. В главе 4 изложены способы

применения принципов глубокого обучения к другим сценариям, таким как более сложное распознавание изображений, классификация аудиоданных и преобразование речи в текст.

В каждом из описанных в предыдущих главах сценариев сеть принимает сложные входные данные. Так, например, описанный в главе 4 классификатор ResNet50 принимает данные из набора ImageNet, обрезанные до размера 224 × 224 пикселя. То есть каждое изображение в общей сложности содержит 50 176 пикселей. Цвет каждого пикселя определяется тремя каналами (красным, зеленым и синим); таким образом, каждое изображение представляется с помощью 50 176 × 3 (150 528) значений. Каждое из этих значений находится в диапазоне от 0 до 255. Это дает нам просто ошеломляющую цифру $256^{150\,528}$ возможных изображений, которые можно предоставить классификатору!

Аналогичным образом можно подсчитать[1], что фотография со сравнительно низким разрешением 1,3 мегапикселя может представить $256^{3\,932\,160}$ различных вариантов изображения. Даже для классификатора монохромных изображений низкого разрешения из набора Fashion-MNIST возможно 256^{784} вариантов входных данных[2].

Один из способов визуализировать все возможные изображения ГНС — поместить каждое в отдельную точку многомерного *входного пространства*. Каждое измерение такого пространства представляет одно значение входного нейрона (или исходный признак). То есть на каждый пиксель приходится одно измерение (три значения и три измерения на пиксель, если изображение цветное, и одно значение и одно измерение на пиксель, если изображение в оттенках серого). Для классификатора ResNet50 входное пространство будет включать в себя 150 528 измерений, каждое из которых может принимать одно из 256 значений. Для классификатора изображений из набора Fashion-MNIST входное пространство состоит из 784 измерений.

Поскольку мы не можем представить графически столь сложное пространство с громадным количеством измерений, для наших целей будем использовать приведенное на рис. 5.1 сверхупрощение, где каждый из двух рассматриваемых наборов данных представлен двумя измерениями.

---

[1] Количество пикселей составляет 1280 × 1024 = 1 310 720. У каждого пикселя есть три канала, что дает нам 3 932 160 значений, каждое из которых находится в диапазоне от 0 до 255.

[2] Каждое изображение содержит 28 × 28 = 784 пикселя в оттенках серого цвета.

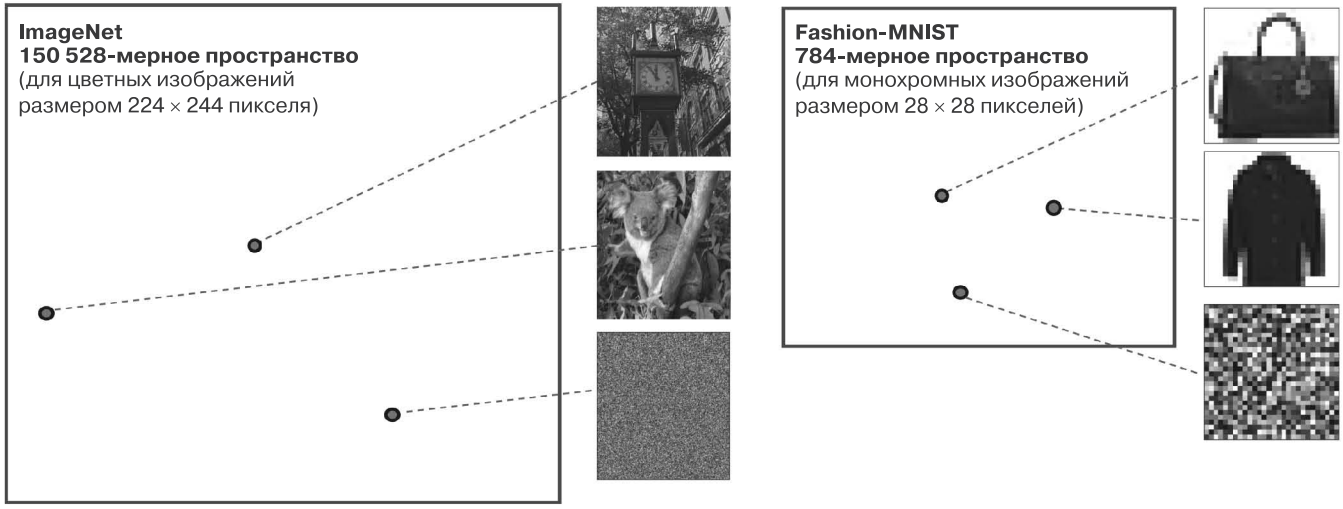


Рис. 5.1. Входные пространства сверхупрощены до двух измерений (очевидно, не в масштабе)

Каким бы громадным ни казалось такое входное пространство, следует отметить, что большинство содержащихся в нем *возможных* изображений не выглядит как изображение в общепринятом смысле, представляя собой случайные комбинации пикселей или, возможно, узоры, не отображающие каких-либо объектов физического мира. Однако в то же время каждое возможное изображение занимает конкретное положение во входном пространстве. Изменение одного пикселя изображения приведет к смещению в пространстве по соответствующей координатной оси (или нескольким осям для цветного изображения).

Как описано в главе 3, для каждого входного изображения ГНС возвращает вектор вероятностей, содержащий по одному значению для каждой возможной категории.

В случае классификатора изображений из набора Fashion-MNIST изображениям из определенной области входного пространства может быть присвоена высокая вероятность принадлежности к категории «Сумка» (Bag), а изображениям из другой области — высокая вероятность принадлежности к категории «Пальто» (Coat), «Сандалии» (Sandal) или какой-либо другой категории одежды. Каждой точке входного пространства соответствует набор из десяти значений, возвращаемых данным классификатором изображений.

Входное пространство и пространство признаков

Под *пространством признаков* понимается такое же многомерное пространство, но с варьированием по *признакам*, а не по исходным входным значениям. То есть пространство признаков представляет собой совокупность различных комбинаций признаков, на основе которых алгоритм МО производит вычисление предсказаний.

В более традиционных приложениях машинного обучения (в которых не используется ГНС) подаваемые в обучаемую модель исходные данные отражают признаки, на основе которых модель вычисляет свои предсказания. Это означает, что пространство признаков в таком случае совпадает с входным пространством.

В отличие от этого глубокие нейронные сети обычно обучаются извлечению признаков из исходных данных. Следовательно, в контексте нейронных сетей под пространством признаков следует понимать пространство с меньшим количеством измерений, содержащее более сложные признаки, извлеченные ГНС с целью вычисления предсказаний.

> Например, в случае СНС, выполняющей классификацию изображений, пространство признаков составляет высокоуровневая информация о признаках, выдаваемая сверточными слоями в начальной части сети.
>
> Строго говоря, когда речь идет об изменении исходных данных с целью создания вредоносных образов для атаки на ГНС, более корректно употреблять термин «*входное пространство*». Однако на практике эти два термина часто используются как синонимы — ведь фактически пиксели входного изображения тоже представляют собой признаки, только очень низкого уровня.

Это удобно представить в виде ландшафтных карт с контурами. Более высокие (более темные) участки при этом будут обозначать области с более высокой вероятностью принадлежности к соответствующей категории («Пальто» (Coat), «Сумка» (Bag) и т. д.). Несмотря на всю простоту данной аналогии, можно брать ее за основу при дальнейшем рассмотрении математических разъяснений генерации вредоносных образов.

Для примера увеличим ту область входного пространства, в которой находится изображение пальто, отобразив с помощью разной степени затенения вероятность принадлежности к каждой из десяти категорий изображений. В результате мы получим набор из десяти *ландшафтов предсказаний*. Выглядит он примерно так, как показано на рис. 5.2. Здесь, как и раньше, используется сверхупрощение представления многомерного входного пространства до всего двух измерений.

На каждой из этих карт темным цветом окрашены те области входного пространства, для которых предсказывается высокая вероятность принадлежности к соответствующей категории. Для изображений, содержащихся в более светлых областях, предсказывается меньшая вероятность принадлежности к соответствующей категории.

Изображение относится к одной из категорий в зависимости от того, какое из предсказаний является наибольшим; при этом часто накладывается дополнительное ограничение на его минимальную величину. На рис. 5.2 эта минимальная величина составляет 0,5 и обозначена на картах сплошной линией. Соответственно, изображения, которые выходят за эту границу на ландшафте предсказаний, относятся к другой категории (или не относятся ни к одной из них). В примере, показанном на рис. 5.2, изображение не выходит за границу на ландшафте, относящемся к категории «Пальто» (Coat), и поэтому классифицируется правильно.

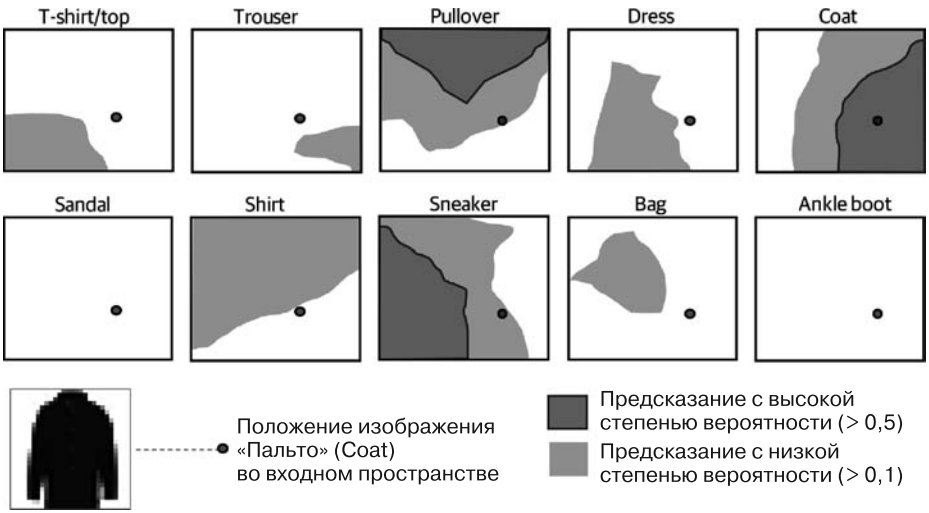


Рис. 5.2. Ландшафты предсказаний модели для каждой из категорий — увеличение в крошечной области входного пространства

Обобщение обучающих данных

Каждый из показанных на рис. 5.2 ландшафтов представляет собой отображение позиции изображения на конкретную категорию. То есть это просто графическое представление формулы, отражающей алгоритм работы нейронной сети:

$$\mathbf{y} = f(\mathbf{x}; \Theta),$$

где степень затенения отражает одно из значений в возвращаемом векторе **y** для изображения пальто, расположенного в позиции **x**; при условии, что ГНС обладает набором весов и порогов, обозначенных буквой Θ.

Вспомните, о чем говорилось в подразделе «Как обучается ГНС» на с. 63, из которого вы узнали, как ГНС обучается путем коррекции всех ее весов и порогов, обозначенных буквой Θ. Проводя аналогию с ландшафтом, можно рассматривать этот процесс как *сдвиг* ландшафтов предсказаний таким образом, чтобы каждый из обучающих примеров оказался на высоте, соответствующей его фактической категории (или максимально приблизился к ней). Значения x обучающих примеров остаются неизменными, а ландшафт видоизменяется так, чтобы ГНС выдавала, насколько это возможно, точный результат для обучающих данных.

Градиентный спуск представляет собой процесс итеративной коррекции параметров функции с целью сместить контуры таким образом, чтобы обеспечивалась правильная классификация обучающих данных. Перед началом обучения параметры инициализируются случайными значениями, потому ландшафт плохо соответствует обучающим примерам. В ходе обучения ландшафт постепенно меняет свою форму по мере изменения параметров, оптимизируя функцию под обучающий набор. Это иллюстрирует рис. 5.3.

Рис. 5.3. Изменение ландшафта предсказаний входного пространства во время обучения

По окончании обучения большинство обучающих примеров с меткой «Пальто» (Coat) должно находиться в пределах области входного пространства, с высокой степенью вероятности относимой к категории «Пальто» (Coat), и то же самое должно быть справедливо в отношении всех остальных категорий. При оптимизации не всегда удается создать классификационные границы, которые бы правильно относили к соответствующим категориям все обучающие изображения, особенно когда некоторые из них не обладают признаками, имеющимися у остальных изображений той же категории. Однако цель оптимизации состоит в том, чтобы *обобщить* паттерны и обеспечить наибольшее соответствие с обучающими данными.

Чтобы модель могла выдавать точные предсказания для всех возможных вариантов входных данных, она должна «заучить» правильные ландшафты предсказаний для областей, содержащих и окружающих обучающие примеры, а также областей, не содержащих обучающие примеры. Это непростая задача, поскольку обучающие примеры, как правило, содержатся лишь в небольшой части входного пространства и, соответственно, многие области входного пространства будут за пределами того набора данных, на котором сеть обучена должным образом.

Задача ГНС заключается в том, чтобы обеспечить точные результаты для всех возможных вариантов входных данных, основываясь на обобщении характеристик, но при этом точность модели очень сильно зависит от того, какие именно характеристики обучающих данных будут обобщаться. Например, какие именно аспекты изображения из набора Fashion-MNIST заставляют ГНС отнести его к категории «Пальто» (Coat)? Если это не те же характеристики, которые бы применял человек, то злоумышленник может использовать это различие для своих неблаговидных целей.

Кроме того, обучающие данные часто не отражают все возможные разновидности входного сигнала. Модель вряд ли сможет выдать точный результат для данных, находящихся за пределами распределения обучающих данных, которые называются *данными вне распределения*, и вредоносные образы могут использовать эту уязвимость алгоритма.

Данные вне распределения

Данные вне распределения не укладываются в распределение обучающего набора данных. Если мы рассмотрим набор всех возможных вариантов входных данных ГНС, то неудивительно, что для задач обработки изображений и аудиоданных большая часть потенциальных входных сигналов будет находиться за пределами этого распределения.

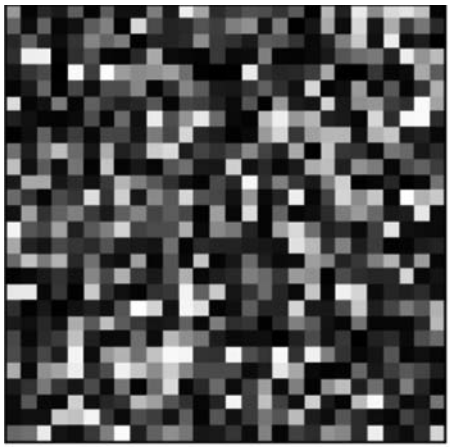
Так, например, обучающий набор данных Fashion-MNIST содержит 60 000 примеров. Довольно большая цифра, не так ли? Однако она окажется совсем небольшой, если мы сравним ее с числом возможных вариантов входных данных для монохромного изображения размером 28×28 пикселей, равным 784^{256}. Точно так же, несмотря на то что набор данных ImageNet содержит свыше 14 миллионов изображений, даже если мы обрежем изображения до размера 224×224 пикселя, обучающий набор будет отражать лишь очень малую часть входного пространства всех возможных вариантов ($150\,528^{256}$).

На практике обучающие данные, как правило, представляют собой примеры реальных данных (таких как фотографии), а данные вне распределения представляют собой данные, которые обычно «не встречаются в природе». Например, данные вне распределения могут представлять собой случайные комбинации пикселей или изображения, подвергнутые необычному преобразованию. Обычно для таких входных данных модель вполне ожидаемым образом выдает предсказания, не позволяющие уверенно сделать тот или иной выбор. В то же время иногда модель также уверенно выдает неверный результат.

Распознавание данных вне распределения — очень сложная задача; мы вернемся к ее обсуждению в главе 10, где рассмотрены способы защиты.

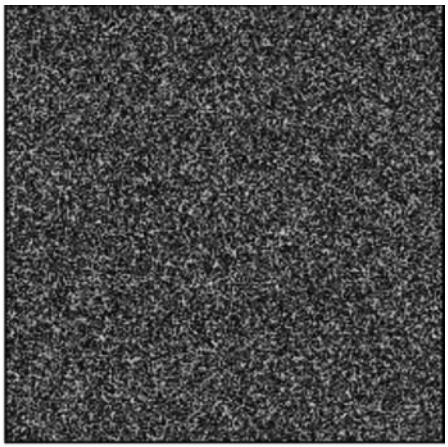
Эксперименты с данными вне распределения

Вы можете для интереса посмотреть, как классификаторы на базе ГНС могут интерпретировать изображения, генерируемые случайным образом, или какие-либо необычные изображения. На рис. 5.4 показаны два примера предсказаний, возвращаемых моделями Fashion-MNIST и ResNet50 при представлении им случайно сгенерированных изображений.



Предсказания модели Fashion-MNIST:

bag	0.850
pullover	0.080
shirt	0.054



Предсказания модели ResNet50:

tennis_ball	0.223
chain	0.108
chain link fence	0.081

Рис. 5.4. Предсказания категорий для случайно сгенерированных изображений

Примеры кода: эксперименты со случайными данными

Вы можете протестировать классификатор Fashion-MNIST на случайных изображениях, используя файл chapter05/fashion-MNIST_random_images.ipynb, размещенный в GitHub-репозитории этой книги (http://bit.ly/2KsCMlI).

Возможно, вы захотите попробовать улучшить модель путем ее дообучения с помощью обучающих изображений, содержащих случайно сгенерированные пиксели, и дополнительной классификационной метки «Без категории» (Unclassified). В блокноте Jupyter содержится код для этого.

В файле chapter05/resnet50_random_images.ipynb (http://bit.ly/2Flr7Bj) представлен код для тестирования модели ResNet50 на случайных данных.

Классификатор Fashion-MNIST выдал наибольшую степень вероятности для категории «Сумка» (Bag) (см. на рис. 5.4, *слева*), и он относит к этой категории подавляющее большинство (более 99 %) представляемых ему случайно сгенерированных изображений. Это означает, что данная модель обучилась относить к категории сумок большую часть своего входного пространства. При этом изображение часто идентифицируется как «Сумка» (Bag) не на основе определенных пикселей, а просто потому, что оно не относится ни к какой другой категории. Модель ResNet50, по крайней мере, уже не дает уверенного ответа в отношении случайно сгенерированного изображения, то есть не производит его неверную идентификацию.

Что «думают» ГНС

Для вычисления предсказаний математическая функция ГНС выделяет и количественно оценивает характеристики данных. При этом определенные характеристики визуальных данных могут представлять для алгоритма большую важность, чем другие. Например, определенная комбинация пикселей может указывать на наличие такого признака, как «нос собаки», тем самым увеличивая вероятность того, что на изображении представлена собака.

Это вполне логично, однако каким образом можно выяснить, какие именно признаки ГНС принимает в расчет? Иными словами, что «видит» (если данные визуальные) или «слышит» (при аудиоданных) модель? Такая информация может быть полезной при создании вредоносных образов.

Лучше всего это можно проиллюстрировать опять же на примере классификации изображений. Можно взять каждый отдельный пиксель изображения и рассчитать его *значимость* для конкретной категории, то есть степень его влияния на отнесение изображения к определенной категории. Высокое значение этого показателя будет говорить о большой важности пикселя при совершении нейросетью выбора в пользу конкретной категории, а низкое значение — о его малой важности при совершении этого выбора. Для работы с изображениями можно представить все эти значения в виде *карты значимости* и посмотреть, какие аспекты изображения учитывает ГНС при отнесении его к той или иной категории.

Математические основы значимости

Значимость оценивается путем вычисления частной производной от выходного сигнала по входному сигналу:

$$\frac{\partial выход}{\partial вход}.$$

Если небольшое изменение входного сигнала ведет к значительному изменению выходного сигнала, то входной сигнал является значимым.

Таким образом, значимость конкретного пикселя *i* для категории *j* определяется путем вычисления следующей частной производной:

$$\frac{\partial f(\mathbf{x})_j}{\partial x_i}.$$

Теоретически это градиент измерения *i* в позиции изображения на ландшафте предсказаний категории *j*. Чем круче градиент, тем больше значимость. Данный метод расчета позволяет лишь приблизительно оценить значимость, поскольку использует непрерывные (линейные) градиенты. О линейности модели речь пойдет в подразделе «Использование линейности модели» на с. 139, а также из главе 6 вы узнаете, как показатели значимости могут быть использованы для создания вредоносных образов.

Примеры кода: генерация карт значимости

Существует несколько пакетов Python для визуализации значимости пикселей изображения. Представленные в данной главе карты значимости сгенерированы с помощью кода, использующего пакет Python Keras-vis (http://bit.ly/2RmVXhH).

> На тот случай, если вы захотите поэкспериментировать с кодом, использовавшимся для генерации представленных в данной главе карт значимости, этот код с подробными разъяснениями включен в GitHub-репозиторий данной книги. Для этих экспериментов можно использовать данные модели Fashion-MNIST, включенные в файл chapter05/fashionMNIST_vis_saliency.ipynb (http://bit.ly/2FeU69N), или данные модели ResNet50, включенные в файл chapter05/resnet50_vis_saliency.ipynb (http://bit.ly/2XYclaR).

На рис. 5.5 приведен пример изображения вместе с тремя наиболее вероятными категориями согласно нейросетевому классификатору ResNet50. Под изображением справа представлена соответствующая карта значимости, на которой более светлым цветом выделены пиксели, представляющие наибольшую важность при генерации предсказаний. Классификатор ResNet50 выдал три наибольших значения вероятности для категорий «аналоговые часы» (analog_clock), «настенные часы» (wall_clock) и «звонница» (bell_cote). (Звонница — это небольшое помещение, в котором подвешивают колокола.)

Данные из набора ImageNet

В приведенных здесь примерах использовалась модель ResNet50, обученная на данных из набора ImageNet. Если вам интересно посмотреть, с помощью каких данных обучалась эта модель, поищите в наборе ImageNet (http://www.image-net.org) обучающие примеры, отнесенные к различным категориям.

На карте значимости выделены такие аспекты циферблата часов, как цифры, из чего можно сделать вывод, что модель ResNet50 считает эти признаки критически важными при генерации своих предсказаний. Внимательно присмотревшись, можно также заметить, что классификатор выделил и другие признаки часов, такие как стрелки и характерный для настенных часов квадратный корпус. Третье по величине (и достаточно малое) предсказание, относящееся к категории «звонница», вероятно, обусловлено тем, что поддерживающая часы подставка напоминает своей формой звонницу.

На рис. 5.5 классификатор ResNet50 выделил как значимые те же аспекты изображения, которые считаем наиболее важными для данной категории

и мы, люди. Однако на самом деле нейронные сети просто обобщают паттерны на основе обучающих данных, и эти паттерны не всегда столь же очевидны.

Предсказания модели ResNet50:

analog_clock	0.850
wall_clock	0.080
bell_cote	0.054

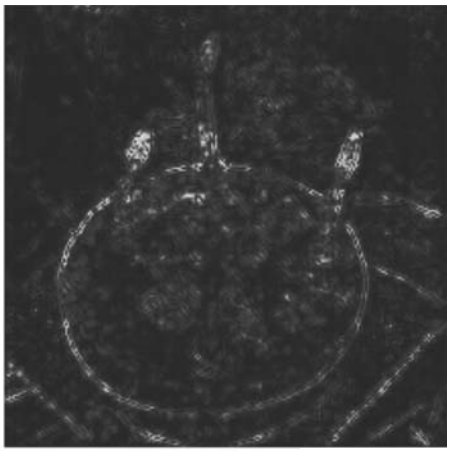
Рис. 5.5. Изображение часов с соответствующей картой значимости (классификатор ResNet50)

Эту идею иллюстрирует рис. 5.6. Нейронная сеть возвращает совершенно разные результаты для двух обрезанных версий одного и того же изображения.

Для верхнего изображения, где свечи занимают доминирующее положение, ГНС выдала наибольшую вероятность для категории «свеча» (candle), следом за которой идет категория «спичка» (matchstick), и, как видно из соответствующей карты значимости, основное внимание нейросети сосредоточено на пламени свечей. При этом также выделен и круглый контур торта; поскольку такая форма характерна для чугунных котелков, третье по величине предсказание — «чугунный котелок» (dutch_oven).

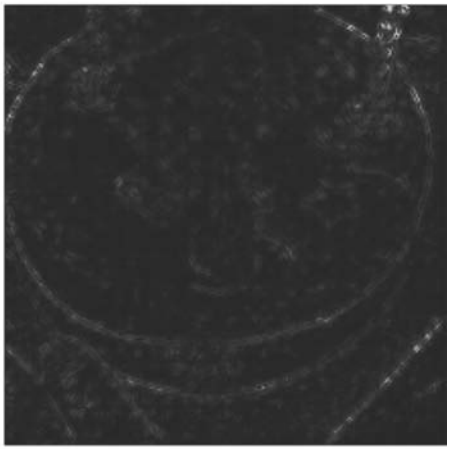
На втором изображении свечи частично обрезаны, в результате чего ГНС ошибочно отнесла его к категории «хоккейная шайба» (puck). Объяснение такой классификации дает карта значимости — выделенный на изображении диск напоминает своей формой хоккейную шайбу. При этом ГНС считает

значимой одну из свечей, но не ее пламя. Это может объяснить третье предсказание — «шпиндель» (spindle).



Предсказания модели ResNet50:

candle	0.849
matchstick	0.029
dutch_oven	0.021



Предсказания модели ResNet50:

puck	0.315
bakery	0.117
spindle	0.063

Рис. 5.6. Изображения торта с соответствующими картами значимости (классификатор ResNet50)

Теперь рассмотрим модель Fashion-MNIST, обучением которой мы занимались в главе 3. Как вы помните, это очень простая нейронная сеть, обученная производить классификацию изображений с очень низким разрешением по десяти категориям одежды. Но, несмотря на то что данный классификатор занимает самую низкую ступень в мире ГНС-моделей, он справляется со своей задачей с достаточно высокой точностью.

На рис. 5.7 изображены пиксели, которые рассматривались простой моделью как наиболее важные, когда она правильно классифицировала несколько таких изображений, как «Брюки» (Trouser) и «Ботильоны» (Ankle boot). Для более наглядной демонстрации корреляции между пикселями и изображениями в данном случае карты значимости были наложены на исходные изображения. При этом для простоты на картах значимости показаны только десять наиболее важных пикселей для определения предсказанной категории.

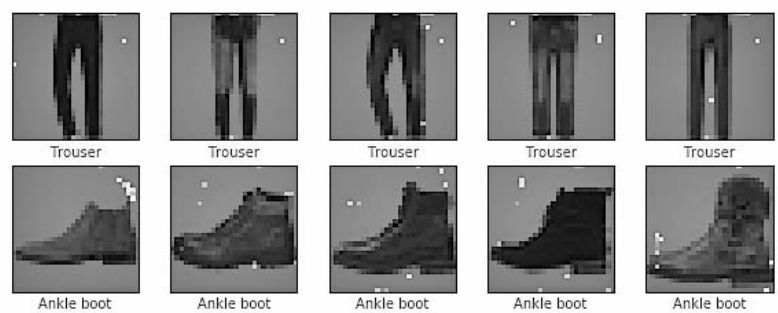


Рис. 5.7. Изображения из набора Fashion-MNIST с наложенными на них соответствующими картами значимости для целевой категории (базовый классификатор)

Как показано на рис. 5.7, при определении категории изображений ГНС выбирает значимые пиксели довольно неожиданным для нас образом. Например, создается впечатление, что модель научилась распознавать брюки, основываясь главным образом на пикселях верхних и нижних строк изображения, вместо того чтобы, к примеру, учитывать форму штанин, повторяющую форму ноги. Точно так же для категории «Ботильоны» (Ankle boot) наибольшую важность, похоже, представляют определенные пиксели в области носка ботильона. Неожиданно важную роль опять же играют определенные кластеры пикселей по краям изображения. Поскольку при обучении модели требовалось найти *простейший* способ распознавания

категорий одежды, она могла выбрать не те признаки, которые в первую очередь использовали бы для этой цели мы, люди. Таким образом, пикселей по краям изображения достаточно для дифференциации между категориями одежды ограниченного набора данных Fashion-MNIST.

Теперь, уже зная, что такое входное пространство и значимость, посмотрим, как эти понятия связаны с генерацией вредоносных входных данных.

Искажающая атака: максимальный эффект при минимальном изменении

Как следует из предыдущих разделов, вредоносные образы заставляют ГНС-модель возвращать неверный ответ, используя слабые места в непротестированных областях входного пространства. Эти примеры могут вносить искажения, которые не могут обмануть человека, или же незаметны для него[1]. Таким образом, какая бы разновидность изменения ни использовалась для превращения неопасного изображения во вредоносное, общий принцип состоит в том, что это должно быть минимальное изменение данных, производящее максимальный эффект на выдаваемый нейросетью результат.

Для начала посмотрим, как можно внести искажение — изменение нескольких значимых пикселей или очень незначительное изменение большого количества пикселей — в изображение пальто из набора Fashion-MNIST, обеспечив его ошибочную классификацию. Изменение нескольких пикселей изображения приведет к его смещению в другую позицию во входном пространстве, с изменением его положения на исходных ландшафтах, представленных на рис. 5.2. Этот сдвиг показан на рис. 5.8 с помощью стрелки, идущей от обозначенного кружком исходного положения изображения к обозначенному треугольником «вредоносному» положению изображения.

С одной стороны, изображение нужно изменить достаточно сильно для того, чтобы оно уже не находилось в области входного пространства, относящейся к категории «Пальто» (Coat). При целевой атаке дополнительно требуется, чтобы изображение сместилось в область входного пространства,

---

[1] Строго говоря, как сказано в главе 1, определение вредоносного примера не требует, чтобы он был незаметен для человека, поскольку этот термин может использоваться для обозначения вредоносного намерения. Однако в рамках излагаемого в этой книге материала нас интересуют именно такие вредоносные примеры.

соответствующую целевой категории. В примере на рис. 5.8 измененное изображение классифицируется как «Кроссовки» (Sneaker). Таким образом, чтобы сгенерировать вредоносное искажение, необходимо определить, изменение каких пикселей приведет к наибольшему отклонению от правильной категории и, возможно, наибольшему смещению в сторону целевой категории.

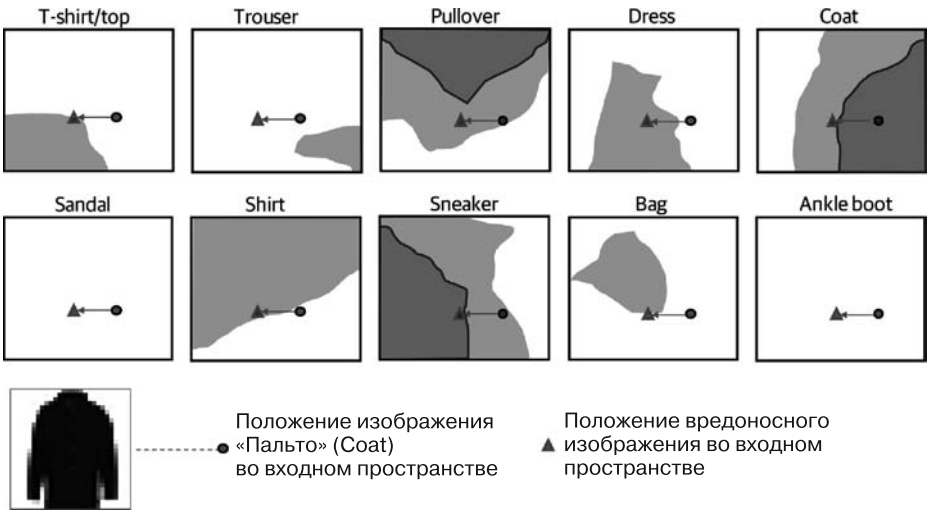


Рис. 5.8. Нецелевая атака — смещение за пределы области входного пространства, относящейся к категории «Пальто» (Coat)

С другой стороны, любое необходимое искажение следует минимизировать таким образом, чтобы оно не выглядело как значительное изменение с точки зрения человека. То есть в идеале искажение должно быть минимальным, чтобы изображение *только вышло* из области категории «Пальто» (Coat) или *лишь вошло* в область целевой категории[1]. Существует несколько подходов — можно либо сосредоточиться на изменении тех пикселей, которые являются

[1] Вредоносный пример с невысокой степенью надежности может «уйти» обратно в область исходной (правильной) категории, если он будет находиться слишком близко к границе, делающей его вредоносным. Такое может произойти, например, в случае, когда пиксели изображения претерпевают небольшое изменение в последовательности обработки до его поступления в нейросетевой классификатор. Более надежный вредоносный пример может находиться чуть дальше от области исходной категории или чуть глубже внутри области целевой категории (при целевой атаке).

наиболее важными для изменения категории (наиболее значимыми), либо изменить много пикселей, но незначительно, чтобы общий эффект от этого изменения не был заметен.

Несмотря на всю упрощенность, данное объяснение иллюстрирует базовые принципы, которые лежат в основе любых способов генерации вредоносных образов. Генерация вредоносных образов обычно сводится к изменению невредоносного образа со смещением его в другую часть входного пространства таким образом, чтобы предсказания модели изменились до максимального желаемого эффекта.

Вредоносная заплатка: максимальное отвлечение внимания

Генерация вредоносной заплатки строится практически на тех же принципах, что и искажающая атака. Опять же цель здесь — изменить входные данные таким образом, чтобы они сместились во входном пространстве либо из области исходной категории (при нецелевой атаке), либо в область целевой категории (при целевой атаке). В то же время в данном случае изменяется лишь некоторая локализованная область изображения, а не все изображение с искажениями общего характера. Изменяемую область, или заплатку, нужно оптимизировать таким образом, чтобы изображение сместилось в другую область входного пространства.

Если цель заключается в том, чтобы обеспечить ошибочное отнесение к категории «коала», то в идеале заплатка должна отражать все то, что характерно для типичного коалы, включая каждый аспект, который модель считает важным для этой категории. Заплатка должна содержать все значимые признаки коалы, чтобы для ГНС она имела даже больше сходства с коалой, чем любое реальное изображение коалы, то есть это должна быть до крайности «коалистая» коала. Это позволяет ей надежно занять положение в нужной области входного пространства, не привлекая внимания к особенностям исходного изображения. Прекрасным примером такой заплатки является показанный на рис. 1.5 тостер.

При оптимизации вредоносного образа может также учитываться размер заплатки, ее позиция в пределах изображения и, возможно, то, как она будет восприниматься людьми. Изменение размера заплатки и ее позиции в пределах изображения влияет на итоговое положение изображения во входном пространстве и, соответственно, на его категорию.

Сверхнормальный стимул

Принцип отвлечения внимания за счет неестественного утрирования особенностей реальных вещей действует не только в сфере ИИ. Учеными доказано, что поведение людей и животных подчиняется аналогичному принципу «сверхнормального стимула».

В 1950-е годы этолог Николас Тинберген (Nikolaas Tinbergen) наглядно показал, что искусственно утрированные версии природных объектов могут активнее индуцировать у чаек инстинктивные модели поведения, чем непосредственно природные объекты[1]. Он доказал это, используя увеличенные макеты яиц и сделанные из вязальных спиц «клювы», окраска которых утрированно имитировала окраску реального клюва. Впоследствии психологи распространили эти идеи и на поведение человека в таких областях, как увлечение нездоровой едой, развлечения и искусство.

Оценка выявляемости атак

Существующие методы генерации вредоносных искажений требуют оценки расстояния между неопасными и вредоносными входными данными. Это то расстояние, на которое смещается изображение во входном пространстве, как показано стрелками на рис. 5.8. После оценки этой величины она сокращается до минимума с помощью математических методов (для минимизации изменения) таким образом, чтобы сохранялось соответствие входных данных критериям вредоносности.

Существуют различные математические методы измерения расстояния между точками в многомерном пространстве; используя их, можно количественно оценить «разницу» между двумя позициями изображения во входном пространстве. Ограничив допустимую величину разницы между вредоносным и невредоносным образами, можно обеспечить минимальный размер искажения. При высокой степени сходства (минимальном размере различий) вредоносный образ будет казаться человеку невредоносным, в то время как при значительных различиях они уже с большей вероятностью будут замечены. На самом деле человеческое восприятие намного сложнее, поскольку определенные аспекты входных данных могут быть более

[1] *Tinbergen N.* The Herring Gull's World: A Study of the Social Behavior of Birds. — London: Collins, 1953. — P. 25.

заметными для человека, чем другие; именно поэтому иногда недостаточно использовать лишь математическую количественную оценку.

В следующем разделе рассмотрены математические методы оценки разницы между неопасными и вредоносными входными данными, а в подразделе «Особенности человеческого восприятия» далее — дополнительные сложности, связанные с особенностями человеческого восприятия.

Математические методы оценки искажения

Существует несколько математических методов оценки расстояния в многомерном пространстве. Результат такой оценки называют L^p-нормой, где значение p определяет конкретный способ расчета расстояния. Краткое описание этих способов представлено на рис. 5.9.

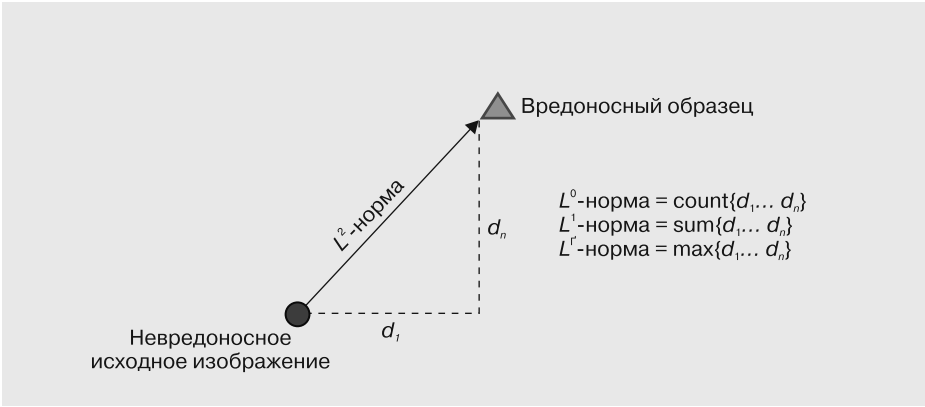


Рис. 5.9. Графическое представление разновидностей L^p-нормы для случая, когда количество измерений равно 2

Пожалуй, наиболее очевидным из этих способов является евклидово расстояние между исходным и вредоносным изображениями во входном пространстве. Это, по сути, просто результат применения теоремы Пифагора в многомерном пространстве признаков для определения расстояния между двумя изображениями путем суммирования квадратов разностей по оси каждого признака и извлечения корня из этой суммы. На языке математики такой показатель различий называется L^2-нормой; это частный случай более обширной группы методов измерения длины вектора. При оценке величины вредоносного изменения вычисляется длина вектора с началом в позиции исходного изображения и концом в позиции вредоносного изображения.

Интересной особенностью многомерных пространств является то, что при любой размерности такого пространства расстояние между двумя точками в нем вычисляется аналогично тому, как определяется евклидово расстояние. И хотя L^2-норма хорошо подходит для оценки расстояния в понятных для нас двумерных и трехмерных пространствах, оказывается, она плохо подходит для оценки расстояния в пространствах с большим количеством измерений. L^2-норма часто используется при генерации вредоносных образов, но далеко не всегда является наиболее удобной мерой величины искажения.

Альтернативный способ оценки расстояния сводится к тому, чтобы вычислять L^1-норму, которая представляет собой просто сумму всех различий между пикселями. Эта норма называется метрикой «такси» или «расстоянием городских кварталов» (манхэттенским расстоянием). В то время как L^2-норма оценивает расстояние по прямой, L^1-норма напоминает кратчайший путь движения такси через город с прямоугольной планировкой улиц.

Еще один подход сводится к тому, чтобы оценивать разницу между двумя изображениями по общему количеству пикселей с отличающимися значениями. Если говорить в терминах входного пространства, то у двух изображений просто оценивается количество отличающихся координат.

На языке математики этот показатель называется L^0-«нормой»[1]. Этот подход кажется вполне обоснованным, поскольку обычно изменение небольшого количества пикселей является менее заметным по сравнению с изменением большого количества пикселей. В то же время L^0-норма не накладывает ограничений на величину изменения каждого пикселя, в результате чего можно получить значительное изменение, затрагивающее лишь небольшую часть изображения.

Наконец, вы можете утверждать, что на самом деле количество изменяемых пикселей не играет какой-либо роли, если изменение каждого пикселя почти или совершенно незаметно для человеческого глаза. Следуя этому принципу, можно было бы просто проследить за тем, чтобы максимальный размер вносимых в пиксели изменений не превышал пороговое значение. Такой подход с использованием L^∞-нормы (нормы бесконечности) получил широкое распространение в сфере исследований, поскольку он позволяет вносить в изображение множество очень мелких и незаметных изменений, которые в совокупности оказывают значительный эффект на классификацию изображения.

---

[1] Здесь намеренно используются кавычки; если вам интересно почему, см. разъяснение математических основ во врезке далее.

Расчет математических норм

Формулы для расчета L^p-норм выглядят следующим образом. Общая формула для расчета L^p-нормы:

$$\|\mathbf{d}\|_p = \left(|d|_1^p + |d|_2^p + \ldots + |d|_n^p\right)^{\frac{1}{p}}.$$

где $|d|_1$, $|d|_2$, ... $|d|_n$ — вектор между двумя позициями входного пространства (например, вредоносным искажением или «заплаткой»).

Прямые скобки обеспечивают положительное значение абсолютной величины (или модуля) каждого элемента, вне зависимости от того, как направлен вектор относительно соответствующей оси — в положительном или отрицательном направлении. Скобки иногда опускают при четном p, поскольку возведение элементов вектора в четную степень гарантированно обеспечивает положительный результат.

При p, равном 1, вычисляется L^1-норма (норма «такси») по следующей формуле:

$$\|\mathbf{d}\|_1 = \left(|d|_1^1 + |d|_2^1 + \ldots + |d|_n^1\right)^1 = |d|_1 + |d|_2 + \ldots + |d|_n.$$

То есть требуется просто суммировать абсолютные величины всех пиксельных изменений. При p, равном 2, вычисляется L^2-норма (евклидова норма) по формуле:

$$\|\mathbf{d}\|_2 = \left(|d|_1^2 + |d|_2^2 + \ldots + |d|_n^2\right)^{\frac{1}{2}} = \sqrt{|d|_1^2 + |d|_2^2 + \ldots + |d|_n^2},$$

а при p, равном ∞, вычисляется L^∞-норма:

$$\|\mathbf{d}\|_\infty = \max\left\{|d|_1, |d|_2 \ldots |d|_n\right\}.$$

Нужно просто определить, чему равно максимальное пиксельное изменение.

И наконец, еще один полезный показатель величины искажения — количество изменяемых пикселей (то есть количество ненулевых элементов вектора). Этот показатель называют L^0-«нормой», однако он не является математической нормой, поскольку при $p = 0$ необходимо вычислить выражение 00, результат которого не определен.

Так какой же из описанных показателей лучше использовать злоумышленнику, чтобы эффективно осуществить искажающую атаку? Стоит ли ему изменить лишь несколько пикселей (минимизировать L^0-норму), либо изменить множество пикселей, следя за тем, чтобы каждое изменение было

небольшим (минимизировать L^{∞}-норму), либо, возможно, минимизировать общую величину смещения изображения во входном пространстве (с помощью L^2 или L^1-нормы)? Ответы на эти вопросы зависят от нескольких факторов, которые рассмотрены далее в этой книге, в том числе от особенностей человеческого восприятия и нужного уровня устойчивости вредоносного образа к предварительной обработке данных.

Особенности человеческого восприятия

При создании вредоносного образа злоумышленнику необходимо сгенерировать такой входной сигнал, который бы неправильно интерпретировался сетью, но не распознавался как вредоносная атака человеком. Это означает, что изменение должно быть либо незаметным для человека, либо настолько незначительным, чтобы человек его сознательно или подсознательно игнорировал.

Главная особенность нашего восприятия состоит в том, что в силу физических ограничений оно охватывает лишь некоторый диапазон электромагнитных и звуковых волн, доступных нашим органам чувств. Поэтому вполне естественно предположить, что данные, обрабатываемые с помощью нейросетевых технологий, имитирующих принимаемые человеком решения в отношении изображений или аудиоданных, должны находиться в рамках ограничений, налагаемых нашими глазами и ушами. В значительной мере это действительно так для цифровых данных, рассчитанных на восприятие человеком. Так, например, такие графические форматы, как PNG, JPEG и т. д., рассчитаны на представление информации, находящейся в видимой области. Точно так же обработка аудиоданных обычно ограничена диапазоном слышимых человеком частот или, если мы имеем дело с обработкой речи, диапазоном частот, издаваемых голосовым трактом человека. Если бы не было этих ограничений, злоумышленник мог бы обмануть ГНС, просто дополнив данные невидимой или неслышимой для человека информацией (пример такой атаки см. в примечании ниже).

«Дельфинья атака»: использование ультразвука

В 2017 году Гуоминь Чжан (Zhang) совместно с другими учеными продемонстрировали возможность эффективного использования ультразвуковых голосовых команд для добавления звуковой вредоносной заплатки, которая была бы неслышимой для людей, но в то же время различимой для цифровых помощников (такая разновидность атаки

> названа «дельфиньей атакой»)[1]. Какой бы интересной ни была эта разновидность атаки, ее можно легко предотвратить, просто проследив за тем, чтобы цифровой помощник отфильтровывал те звуки, которые не может слышать человек, или, что еще лучше, те звуки, которые не может издавать человеческий голос. Следовательно, вредоносные атаки, использующие не воспринимаемые человеком диапазоны частот электромагнитных или звуковых волн, не могут представлять сколь-нибудь серьезную угрозу.

Если предположить, что все данные представлены в рамках воспринимаемого человеком диапазона, использование описываемых здесь математических показателей различий осложняется тем, что они присваивают всем элементам входных данных одинаковые веса. Эти показатели подразумевают, что все пиксели изображения воспринимаются равномерно и, таким образом, вносят равный вклад в восприятие этого изображения человеком. Вполне очевидно, что это не так — доказано, что люди, как правило, меньше замечают изменения в насыщенных деталями областях изображения и больше — в таких простых областях, как участки ясного неба.

Для аудиоданных вредоносные образы часто генерируются с использованием показателя искажения, выраженного в логарифмических единицах измерения — децибелах (дБ) и показывающего *относительный* уровень громкости искажения по отношению к исходному аудиосигналу. Это позволяет надежно обеспечить незаметность вредоносных аудиоданных для человека, поскольку при таком подходе изменения, вносимые в тихих участках аудиосигнала, будут в любом случае сравнительно небольшими по сравнению с изменениями, вносимыми в его громких участках.

К настоящему времени уже проведено достаточно много исследований по вопросу о том, какие аспекты изображений и звуковых сигналов привлекают внимание людей в наибольшей мере, то есть представляют собой значимые признаки с точки зрения человека, а не с точки зрения машины (о которых мы говорили в разделе «Что “думают” ГНС» на с. 114). Вредоносные образы могут быть улучшены за счет смещения искажений тех аспектов входных данных, которые в меньшей степени интересны человеку и больше интересны модели. Рассмотрим изображения: люди подсознательно разбивают информацию изображения на составные части, уделяя больше внимания переднему плану и меньше — фону. При этом, используя эффективные и в то же время

[1] *Zhang G. et al.* DolphinAtack: Inaudible Voice Commands // Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017. http://bit.ly/2MWUtft.

простые методы, злоумышленник может обеспечить себе больший простор действий при создании вредоносных входных данных, например, путем преимущественного размещения искажений в насыщенных деталями фоновых областях изображений.

В психологии есть понятие *«абсолютный порог ощущения»* — это величина стимула, минимально необходимая для того, чтобы человек замечал его в 50 % случаев.

Неудивительно, что это значение варьируется у разных людей и даже у одного человека в зависимости от аспектов физиологического состояния. Злоумышленнику, если он знает, при превышении какого порога стимулы становятся заметными для человека, проще создать вредоносные данные.

Еще один интересный фактор — адаптация органов чувств. Со временем наши органы чувств становятся менее чувствительными к постоянно действующему стимулу, что позволяет нам замечать те изменения, которые важны для нашего выживания. Например, находясь за рулем автомобиля, мы спустя некоторое время перестаем замечать шум работающего двигателя. Существует и обратная закономерность — мы крайне чувствительны к резким изменениям в поступающей сенсорной информации, таким как неожиданные звуки. Соответственно, злоумышленник может повысить свои шансы остаться незамеченным за счет постепенного увеличения искажения, которое он вносит в видео- или аудиоданные.

Резюме

В этой главе вы ознакомились с основными принципами генерации вредоносных образов. В главе 6 более подробно представлены методы создания вредоносных данных, но прежде ознакомьтесь с тем, какие высокоуровневые математические выкладки лежат в их основе.

Математические основы генерации вредоносных образов

Допустим, что у нас есть нейронная сеть, выступающая в роли классификатора. Она принимает вектор, представляющий входные данные, и возвращает некоторый результат:

$$f(\mathbf{x}; \Theta) = \mathbf{y}.$$

В этом уравнении f — функция, применяемая алгоритмом ГНС для генерации результата. Как вы помните из главы 3, буквой Θ обозначаются все параметры

нейронной сети (ее веса и пороги). После успешного обучения эти параметры уже не подвергаются изменениям; значит, это уравнение можно записать в упрощенном виде так:

$$f(\mathbf{x}) = \mathbf{y},$$

где $\mathbf{x}$ — та точка исходного входного пространства, в которой находится входной сигнал.

Что именно обозначают буквы $\mathbf{x}$ и $\mathbf{y}$ в этом уравнении, зависит от конкретного типа входных данных и решаемой задачи. Если ГНС используется в качестве классификатора монохромных изображений, $\mathbf{x}$ представляет собой вектор действительных чисел, каждое из которых отражает значение соответствующего пикселя:

$$\mathbf{x} \in \mathbb{R}.$$

Для классификатора y является отдельной категорией, такой как «собака» или «кошка», поставленной в соответствие элементу перечисления, получаемому на основе вектора вероятностей, выдаваемого выходным слоем ГНС (например, путем определения максимального значения в этом векторе). То есть в данном случае y является не вектором вероятностей, а числом из диапазона 1... L, где L — количество категорий. Это можно записать следующим образом:

$$y \in \{1, 2... L\}.$$

При создании вредоносного образа необходимо внести в исходный входной сигнал тщательно рассчитанное изменение. На языке простой математики это можно выразить так:

$$\mathbf{x}^{\text{avd}} = \mathbf{x} + \mathbf{r}.$$

где:

- $\mathbf{x}^{\text{adv}}$ — вектор, представляющий собой обновленные (вредоносные) входные данные);
- $\mathbf{x}$ — вектор, представляющий собой исходные входные данные;
- $\mathbf{r}$ — вектор, представляющий собой небольшое изменение исходных входных данных.

Чтобы вектор $\mathbf{x}^{\text{adv}}$ успешно справлялся со своей неблаговидной задачей, соответствующий результат модели (категория) должен отличаться от результата, выдаваемого для невредоносного входного сигнала. Это можно записать так:

$$f(\mathbf{x}^{\text{avd}}) \neq f(\mathbf{x}).$$

При целевой атаке есть дополнительное ограничение:

$$f(\mathbf{x}^{\text{avd}}) = y_t,$$

где y_t — целевая категория вредоносной атаки.

Независимо от того, как именно осуществляется атака — с помощью искажения или заплатки, необходимо минимизировать величину вредоносного изменения **r**, чтобы сделать его незаметным (или менее заметным) для человека. Если это оценивается с помощью простой L^p-нормы для искажающей атаки, то задача состоит в том, чтобы найти изображение, расположенное как можно ближе к **x**, то есть нужно дополнительно указать, что величина искажения **r** должна быть минимально возможной. При нецелевой атаке это можно записать так:

$$\arg\min_{\mathbf{r}} \left\{ \|\mathbf{r}\|_p : f(\mathbf{x}^{\text{adv}}) \neq f(\mathbf{x}) \right\}.$$

При целевой атаке:

$$\arg\min_{\mathbf{r}} \left\{ \|\mathbf{r}\|_p : f(\mathbf{x}^{\text{adv}}) = y_t \right\}.$$

Величина числа p зависит от того, какой способ оценки расстояния используется для оценки вредоносного образа. Так, например, если для такой оценки применяется евклидово расстояние, то предыдущее уравнение будет выглядеть так:

$$\arg\min_{\mathbf{r}} \left\{ \|\mathbf{r}\|_2 : f(\mathbf{x}^{\text{adv}}) = y_t \right\}.$$

Эта оценка может производиться и более сложным образом, например с учетом того, как вредоносное искажение влияет на человеческое восприятие, однако в подавляющем большинстве случаев вредоносные образы оцениваются с помощью L^p-нормы.

Задача злоумышленника — найти оптимальное значение **r**, которое бы удовлетворяло предыдущим ограничениям. С математической точки зрения это можно обеспечить путем применения одного из существующих алгоритмов условной оптимизации (некоторые из них рассмотрены в главе 6). Какой бы алгоритм мы ни использовали, он должен найти величину искажения **r** для создания вредоносного образа, выражающуюся следующей формулой:

$$\mathbf{x}^{\text{adv}} = \mathbf{x} + \arg\min_{\mathbf{r}} \left\{ \|\mathbf{r}\|_p : f(\mathbf{x}^{\text{adv}}) \neq f(\mathbf{x}) \right\}.$$

При целевой атаке эта задача принимает следующий, чуть более конкретизированный вид:

$$\mathbf{x}^{\text{adv}} = \mathbf{x} + \arg\min_{\mathbf{r}} \left\{ \|\mathbf{r}\|_p : f(\mathbf{x}^{\text{adv}}) = y_t \right\}.$$

ГЛАВА 6

Методы генерации вредоносных искажений

В главе 5 рассмотрены базовые принципы вредоносных входных данных. Но как такие данные генерируются на практике? В этой главе представлены методы генерации вредоносных изображений и приведены некоторые примеры кода, с которыми вы сможете поэкспериментировать. Из главы 7 вы узнаете, как такие методы могут встраиваться в реальные атаки в том случае, когда ГНС является составной частью более длинной последовательности обработки и злоумышленнику нужно решать ряд дополнительных проблем, таких как обеспечение незаметности атаки.

Проекты с открытым исходным кодом

На сегодняшний день существует целый ряд инициатив, направленных на привлечение широкой общественности к изучению вредоносных атак и способов защиты от них, таких как CleverHans (bit.ly/2N7t7mG), Foolbox (foolbox.readthedocs.io) и Adversarial Robustness Toolbox (bit.ly/2XZ7EgW) компании IBM. Об этих проектах подробно рассказано в главе 10.

Для единообразия во всех примерах кода, приводимых в книге, используется библиотека Foolbox.

Прежде чем приступить к рассмотрению методов создания вредоносных входных данных, возможно, вы захотите услышать ответ на следующий вопрос: насколько сложно создать вредоносный образ методом проб и ошибок? Например, злоумышленник может просто внести в изображение случайное искажение и посмотреть, как это скажется на предсказаниях модели. К сожалению, добиться успеха таким путем злоумышленнику очень непросто. Поскольку на этапе обучения ГНС производит обобщение на основе обучающих данных, она обычно устойчива к мелким случайным искажениям, значит, такие изменения вряд ли будут успешными. Представленный на

рис. 6.1 пример показывает, что, даже когда величина случайно генерируемого искажения каждого пикселя становится достаточно большой, классификатор ResNet50 не перестает выдавать правильную категорию. Неверная категория выдается только после того, как искажение будет заметным.

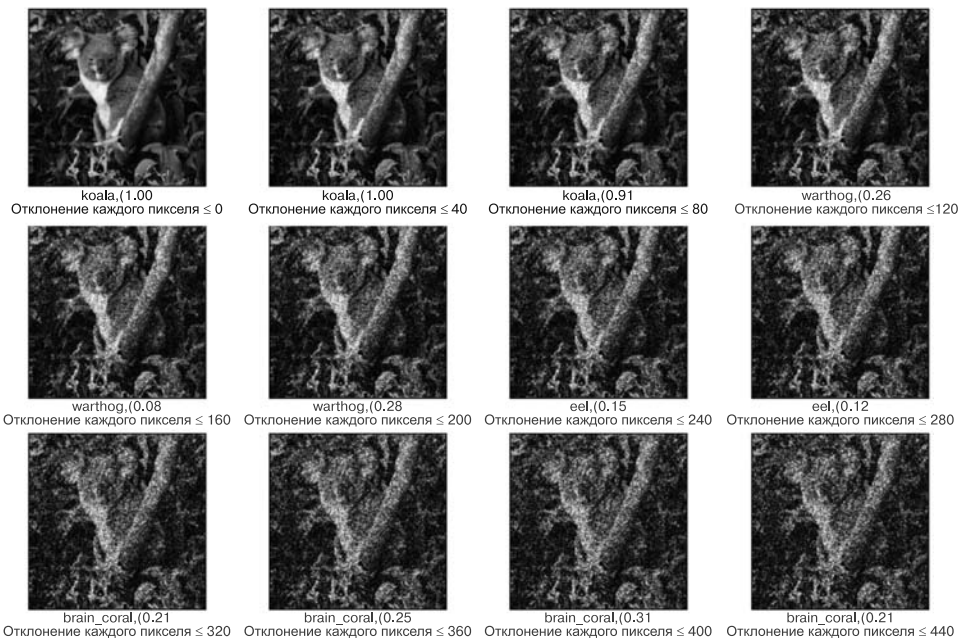


Рис. 6.1. Предсказания, выданные моделью ResNet50 для изображения со случайными изменениями; для каждой итерации указана соответствующая максимальная величина искажения каждого пикселя

Для эффективной генерации вредоносных образов злоумышленнику нужно действовать хитрее. Он может использовать несколько подходов, каждый из которых предполагает различный уровень осведомленности об алгоритме ГНС. Эти методы генерации вредоносных входных данных можно разбить на три категории по уровню доступа злоумышленника к модели.

- *Методы белого ящика*. Эти методы позволяют создавать вредоносные входные данные, основываясь на полной осведомленности о ГНС-модели.
- *Методы ограниченного черного ящика*. Эти методы позволяют корректировать вредоносные входные данные на основе результата, генерируемого моделью или той системой, составной частью которой она является. Этим результатом, например, может быть окончательная классификация.

- *Методы черного ящика с оценкой.* Эти методы позволяют корректировать вредоносные входные данные на основе предварительных предсказаний (оценок) ГНС. Методы черного ящика с оценкой могут иметь доступ ко всем оценкам или только к самым высоким из них (например, к десяти наибольшим). Такие методы занимают промежуточное положение между методами белого ящика и методами ограниченного черного ящика — для их использования нужно иметь более подробные сведения о результатах по сравнению с методами ограниченного черного ящика, но не требуется иметь полное представление об алгоритме модели, как того требуют методы белого ящика.

Примитивный способ генерации вредоносных искажений

Чтобы поэкспериментировать с кодом, результат работы которого представлен на рис. 6.1, воспользуйтесь блокнотом Jupyter chapter06/resnet50_naive_attack.ipynb (bit.ly/2FlGCZN), размещенным в GitHub-репозитории этой книги.

Для классификатора Fashion-MNIST воспользуйтесь блокнотом Jupyter chapter06/fashionMNIST_naive_attack.ipynb (bit.ly/2WSB1oO).

Какая информация доступна злоумышленнику при использовании каждой из этих трех групп методов, графически представлено на рис. 6.2.

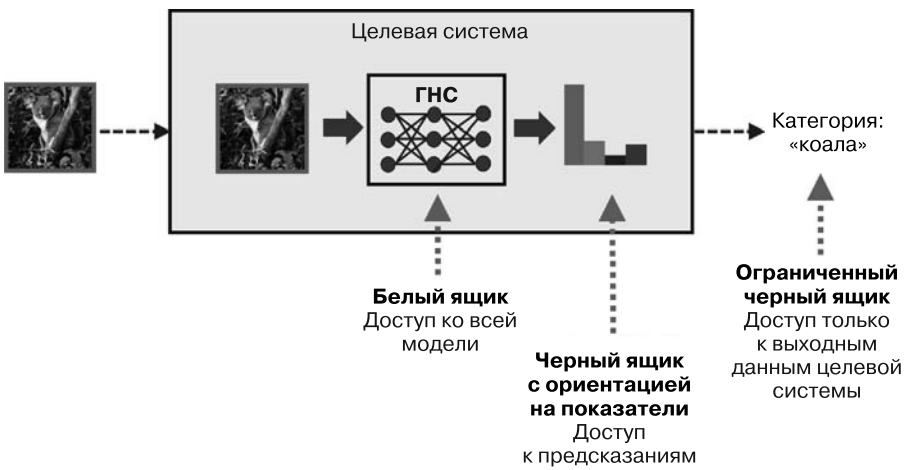


Рис. 6.2. Информация, доступная злоумышленнику при использовании методов белого ящика, черного ящика с оценкой и ограниченного черного ящика

В оставшейся части этой главы поочередно рассмотрен каждый из этих методов.

Использование замещающей модели: атаки копией и переносом

Для разработки и тестирования вредоносных входных данных зачастую можно (и, как вы убедитесь впоследствии, иногда очень полезно) использовать замещающую ГНС в качестве копии или аппроксимации целевой ГНС. Это позволяет злоумышленнику сначала разработать атаку, используя методы белого или черного ящика с оценкой применительно к замещающей нейросети, а затем уже применить ее в отношении реальной цели.

Данный подход имеет преимущество, так как позволяет злоумышленнику оставаться незамеченным на этапе окончательной доработки атаки (обращение к реальной модели может вызвать подозрение). Это также единственный доступный вариант действий, если у злоумышленника нет доступа к реальной модели.

Когда замещающая модель идентична целевой модели, она называется *атакой с копированием (replica attack)*. Если замещающая модель является аппроксимацией цели, то это *атака с переносом (transfer attack)*, поскольку для достижения успеха при этом нужно *перенести* вредоносные входные данные на целевой алгоритм.

Описываемые в этой главе методы (белого ящика, черного ящика с оценкой и ограниченного черного ящика) могут использоваться применительно и к целевой, и к замещающей сети, потому не отражают степень доступа злоумышленника к реальной целевой ГНС или его осведомленности о ней. Более подробно об этом — в главе 7.

Методы белого ящика

Методы белого ящика предполагают полную осведомленность о ГНС-модели (ее параметрах и архитектуре) и основаны на математических методах оптимизации для создания вредоносных образов путем расчета градиентов внутри ландшафтов входного пространства, о которых говорилось в главе 5. Такие методы представляют особый интерес, так как они дают представление о неотъемлемых уязвимостях ГНС. В данном разделе рассмотрен принцип действия этих методов.

Поиск во входном пространстве

В главе 5 вы узнали о смещении изображений за границы категорий во входном пространстве с помощью тщательно подобранных искажений. При этом ставилась задача минимизировать величину смещения во входном пространстве (оцениваемую с помощью какой-либо L^p-нормы), в то же время реализовав требуемую вредоносную цель, будь то целевая или нецелевая ошибочная классификация. Обратимся еще раз к примеру с набором данных Fashion-MNIST из этой главы и посмотрим, как внесение изменений в изображение пальто сказывается на выдаваемых ГНС предсказаниях. При этом нас главным образом интересует ландшафт предсказаний категории «Пальто» (Coat), представленный на рис. 6.3 в увеличенном виде.

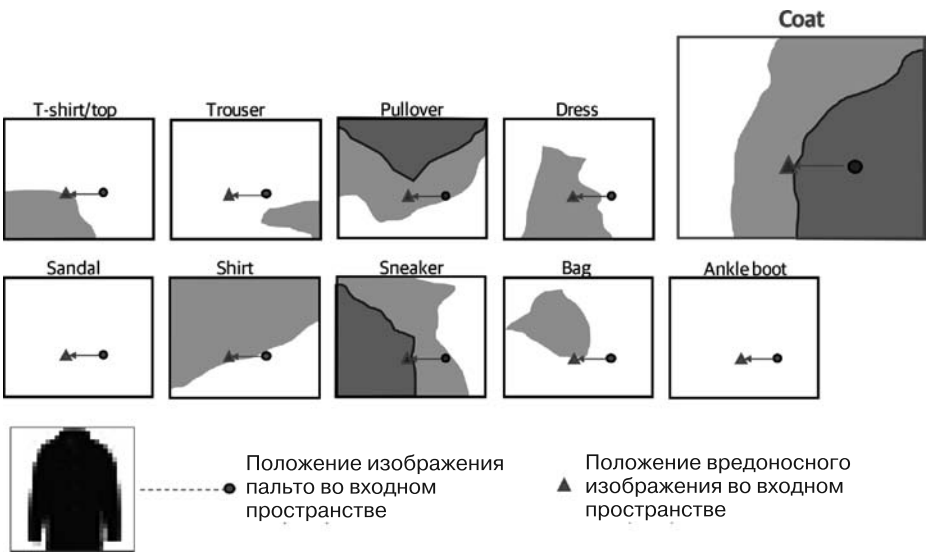


Рис. 6.3. Нецелевая атака — смещение изображения за пределы области входного пространства, относящейся к категории «Пальто» (Coat)

Пожалуй, наиболее очевидный метод поиска положения во входном пространстве, обеспечивающего достижение вредоносной цели, состоит в том, чтобы просто выполнять поиск в наружном направлении от исходного изображения. То есть, начав с той точки, где расположено изображение, нужно попробовать внести несколько небольших изменений и посмотреть, какое из них смещает предсказания в требуемом направлении (то есть в данном случае снижает величину предсказания, относящегося к кате-

гории «Пальто» (Coat)), после чего повторить эти действия в отношении соответствующего измененного варианта. При этом, как показано на рис. 6.4, мы фактически производим итеративный поиск во входном пространстве, взяв за начало исходное изображение и двигаясь в наружном направлении до тех пор, пока предсказания не изменятся так, чтобы изменилась категория изображения.

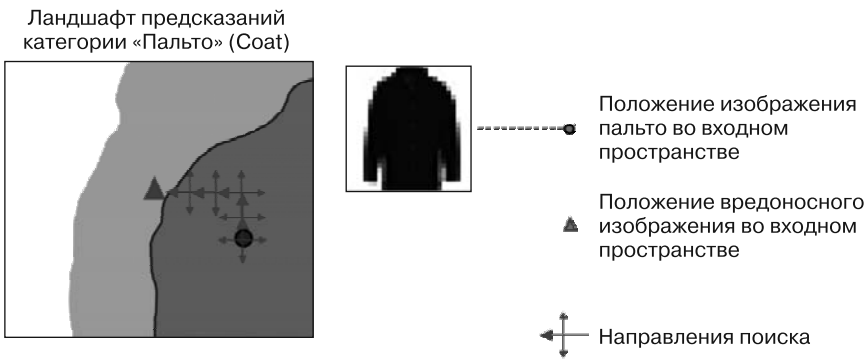


Рис. 6.4. Итеративный поиск вредоносного положения во входном пространстве

Как бы просто это ни выглядело на первый взгляд, выполнить такой поиск не так легко, как кажется. Для этого нужно проверить огромное количество различных пиксельных изменений и комбинаций. Конечно, можно попытаться решить эту задачу методом грубой силы, например перебирая все возможные варианты небольших искажений входных данных и оценивая, какой вариант дает наилучший результат. Однако, как объясняется в примечании далее, это трудноосуществимо с точки зрения вычислительных возможностей.

Многочисленные возможные искажения

Предположим, что разрешение изображения составляет 224×224 пикселя. Допустим, мы решили сгенерировать все возможные варианты небольших изменений этого изображения, минимизировав величину изменения любого конкретного пикселя так, что она равна либо $+\varepsilon$, либо $-\varepsilon$, где ε — некоторая небольшая величина. Сгенерировав все эти изменения, мы, возможно, смогли бы проверить каждое из них на предмет соответствия критериям вредоносности. Но сколько вариантов нам потребовалось бы проверить?

> Как вы помните из главы 5, для представления цветного изображения с низким разрешением (224 × 224 пикселя) требуется 150 528 пиксельных значений (где каждый пиксель имеет три значения для красного, зеленого и синего цветов).
>
> У каждого генерируемого нами искаженного варианта исходного изображения отдельные пиксельные значения могут остаться без изменений либо измениться на величину ε в большую или меньшую сторону.
>
> Таким образом, существует $3^{150\,528}$ комбинаций искажений, при которых каждое пиксельное значение может изменяться только ровно на величину ε от его исходной величины. Если быть совершенно точными, то из этой цифры еще нужно вычесть 1, поскольку мы не должны учитывать тот вариант искажения, при котором каждый пиксель остается неизменным. Если мы попробуем вычислить это значение с помощью калькулятора, то он выдаст ошибку переполнения. С другой стороны, можно вычислить его в ячейке блокнота Jupyter с помощью следующего выражения:
>
> pow(3,150 528) – 1.
>
> (Возвращаемое значение слишком длинное, чтобы приводить его здесь.)

Каким бы трудноосуществимым ни был поиск во входном пространстве, если у злоумышленника есть доступ к алгоритму ГНС, он получает огромное преимущество, поскольку с помощью этого алгоритма может сгенерировать математическую аппроксимацию поиска, уменьшающую количество проверяемых комбинаций.

В своем первоначальном исследовании[1] Шегеди и др. ускорили процесс поиска вредоносных образов в области, окружающей исходное изображение, используя алгоритм Бройдена — Флетчера — Гольдфарба — Шанно с ограниченной памятью (Broyden-Fletcher-Goldfarb-Shanno, L-BFGS)). Алгоритм L-BFGS представляет собой математический метод, позволяющий повысить эффективность поиска во входном пространстве за счет аппроксимации вероятностных градиентов во входном пространстве вблизи определенной точки. Ограниченная память здесь означает дальнейшую аппроксимацию с целью уменьшения объема оперативной памяти, необходимого для выполнения итеративного поиска.

Как оказалось, алгоритм L-BFGS позволяет успешно выявлять вредоносные образы, но делает это очень медленно и с большими вычислительными затра-

[1] *Szegedy et al.* Intriguing Properties of Neural Networks.

тами[1]. Чтобы еще больше оптимизировать процесс поиска, нам нужно лучше разобраться в том, какими характеристиками обладают алгоритмы глубоких нейросетей и как выглядят создаваемые ими ландшафты предсказаний.

Использование линейности модели

В 2015 году некоторые из участников первоначального исследования вредоносных образов обратились к данной проблеме еще раз. Они доказали, что для генерации вредоносных образов можно с успехом использовать такой простой алгоритм, как метод быстрого градиента (Fast Gradient Sign Method, FGSM)[2].

При этом алгоритм FGSM никоим образом не рассматривался как *наилучший* способ поиска вредоносных входных данных, а скорее, был призван продемонстрировать ту особенность алгоритмов ГНС, которая существенно упрощает задачу создания вредоносных входных данных. Рассмотрим, что представляет собой этот алгоритм, а затем сделаем выводы.

Алгоритм FGSM рассчитывает то направление во входном пространстве, движение по которому из места размещения изображения является кратчайшим путем к ошибочной классификации. Это направление рассчитывается с использованием градиентного спуска и практически такой же функции потерь, которая используется при обучении сети (о ней говорилось в подразделе «Как обучается ГНС» на с. 63). На концептуальном уровне такой расчет направления можно рассматривать просто как косвенный способ оценки крутизны контуров на ландшафте предсказаний в месте размещения изображения.

Этот расчет вредоносного направления производится очень приблизительно, с присвоением каждому входному значению (пиксельному значению, или, иными словами, координатной оси в многомерном входном пространстве) одного из двух знаков:

- *плюс* — указывает, что для обеспечения ошибочной классификации соответствующее входное значение лучше увеличить;

---

[1] Произведенные впоследствии оптимизации этого метода позволили генерировать искажения более эффективно, как, например, описано Николасом Карлини и Дэвидом Вагнером в статье «Об оценке устойчивости нейронных сетей» (*Carlini N., Wagner D.* Towards Evaluating the Robustness of Neural Networks, 2016). http://bit.ly/2KZoIzL.

[2] *Goodfellow I. J. et al.* Explaining and Harnessing Adversarial Examples, 2015. http://bit.ly/2FeUtRJ.

- *минус* — указывает, что для обеспечения ошибочной классификации соответствующее входное значение лучше уменьшить.

Столь простое присвоение плюсов или минусов может показаться не вполне логичным, однако метод FGSM не учитывает *относительную* важность отдельных изменений, принимая во внимание лишь их направление (положительное или отрицательное). Если, например, увеличение одного входного значения будет обеспечивать более значительный вредоносный эффект, чем увеличение другого значения, то, несмотря на это, оба значения получат знак плюс и будут обработаны совершенно одинаково.

Определив направление, метод FGSM вносит крошечное искажение в каждое входное значение (пиксельное значение для изображения), добавляя его при положительном направлении вредоносного изменения и вычитая его в противном случае. При этом можно надеяться на то, что из-за этих изменений изображение лишь немного выйдет за границу области, относящейся к его правильной категории, тем самым обеспечив нам (нецелевой) вредоносный образ.

Базовый принцип метода FGSM сводится к тому, чтобы изменять все входные значения (пиксели), но на очень небольшую величину. В основе такого подхода лежит тот факт, что множество крошечных изменений вдоль каждой координатной оси многомерного пространства может давать значительное изменение в совокупности; изменение множества пикселей на крошечную величину обеспечивает значительное смещение во входном пространстве. Если вы вспомните о L^p-норме, то вы поймете, что данный метод сводится к минимизации максимального изменения отдельного пикселя, то есть использует L^∞-норму.

Самое удивительное в методе FGSM, что он вообще работает! Эта мысль наглядно изображена на рис. 6.5, где показано входное пространство двух различных ГНС-моделей. На обоих изображениях показан укрупненный вид входного пространства, поэтому стрелка, указывающая разницу между положением исходного изображения и положением нового изображения, сгенерированного алгоритмом FGSM, в действительности отражает очень небольшое смещение вдоль каждой координатной оси.

Граница области, относящейся к категории «Рубашка» (Shirt), обозначена одинаковой сплошной линией в обоих случаях, но, как видите, градиенты модели слева отличаются большим единообразием. Форма контуров здесь предполагает плавный, единообразный уклон, в отличие от непоследовательного и сложнопредсказуемого правого ландшафта.

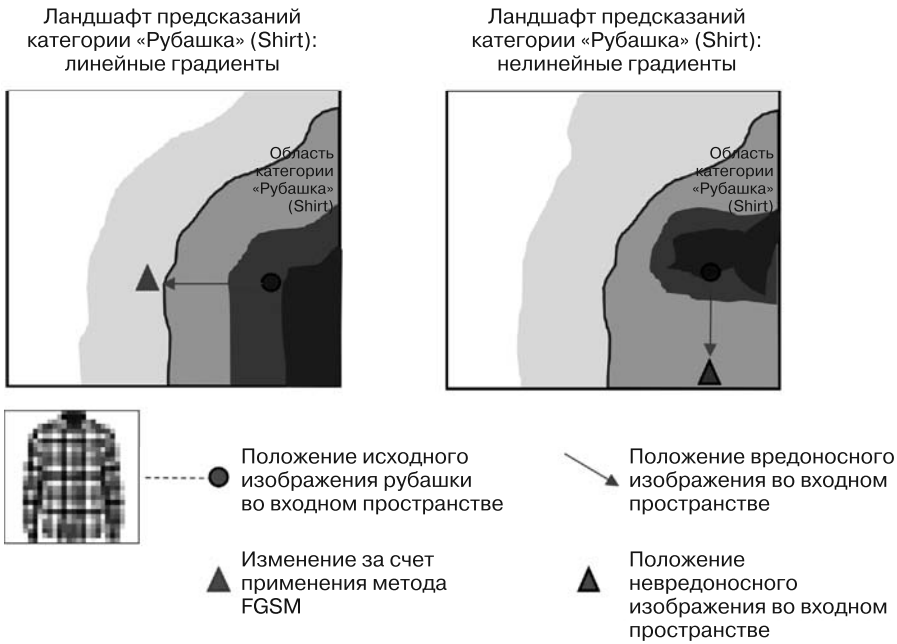


Рис. 6.5. Применение метода быстрого градиента (FGSM) в линейной и нелинейной моделях

Направление рассчитываемого алгоритмом FGSM искажения определяется градиентом в точке расположения исходного изображения. На рисунке слева простой алгоритм успешно переносит изображение во вредоносную точку, потому что эти градиенты очень мало изменяются по мере удаления изображения от исходной точки. Однако, когда градиенты выглядят так, как показано справа, алгоритму FGSM не удается перенести изображение во вредоносную точку, потому что форма градиента вблизи исходного положения изображения отличается от формы градиентов в более удаленных точках. Соответственно, сгенерированное алгоритмом изображение по-прежнему находится в области, относящейся к категории «Рубашка» (Shirt).

В общем случае необходимым предварительным условием для успешного применения метода FGSM является непрерывность крутизны уклона в некотором конкретном направлении. На языке математики это означает, что реализуемая моделью функция должна демонстрировать *линейное* поведение. Когда модель является линейной, мы можем в значительной мере аппроксимировать математику для генерации вредоносного искажения, просто взглянув на локальные градиенты.

Алгоритм FGSM не сможет сгенерировать вредоносные входные данные *наилучшим образом*, но он для этого и не предназначен. Продемонстрировав возможность успешного применения метода FGSM в отношении самых современных ГНС для классификации изображений, исследователи показали, что этим алгоритмам в значительной мере свойственна линейность; ГНС-модели, как правило, обладают единообразными градиентами внутри ландшафтов входного пространства, как показано на рис. 6.5, *слева*, а не такими непоследовательными градиентами, как в правой части этого рисунка. До появления метода FGSM предполагалось, что алгоритмы глубоких нейросетей формируют более сложные нелинейные градиенты, которые можно представить в виде ландшафтов с постоянно меняющейся крутизной и множеством «горбов» и «впадин». Такая линейность имеет место потому, что при оптимизации на этапе обучения всегда отдается предпочтение самой простой модели (и простейшим градиентам).

Линейность модели сильно упрощает математические выкладки для генерации вредоносных образов с использованием методов белого ящика, поскольку позволяет в значительной мере аппроксимировать математику для генерации вредоносного искажения, просто взглянув на локальные градиенты.

Математические основы метода FGSM

Ниже изложены математические основы алгоритма FGSM. Дополнительные подробности см. в указанной статье[1].

В подразделе «Как обучается ГНС» на с. 63 введено понятие «функция штрафа», которая служит для оценки того, насколько хорошие результаты показывает сеть для отдельного входного сигнала. Это можно записать следующим образом:

$$C(f(\mathbf{x};\Theta),\mathbf{y}),$$

где:

- C — функция штрафа глубокой нейросети f с входным сигналом $\mathbf{x}$ и правильным выходным вектором вероятностей $\mathbf{y}$[2];
- Θ — параметры (веса и пороги) сети f.

[1] *Goodfellow et al.* Explaining and Harnessing Adversarial Examples.

[2] Классификатор «схлопывает» этот вектор вероятностей до одного значения, представляющего наиболее вероятную категорию.

Цель обучения нейронной сети состоит в том, чтобы минимизировать штрафы (оцениваемые с помощью вышеуказанной функции) для всех обучающих примеров за счет коррекции весов и порогов. Параметры Θ изменяются до тех пор, пока сеть не начнет показывать хорошие результаты для обучающих данных.

Теперь посмотрим на функцию штрафа в другом контексте, когда нейронная сеть уже прошла обучение и ее параметры больше не подвергаются изменениям. На этом этапе обозначенные буквой Θ веса и пороги должны оставаться неизменными. В то же время изменение входных данных (изображения), обозначенных буквой $\mathbf{x}$, будет также вести к изменению величины штрафа. Таким образом, мы можем снова применить функцию штрафа, на этот раз для оценки эффекта от смещения изображения внутри ландшафта предсказаний, а не для изменения самого ландшафта предсказаний, как это делалось на этапе обучения.

Когда изменение величины $\mathbf{x}$ повышает уверенность правильного предсказания, функция штрафа возвращает меньшее значение. А когда изменение величины $\mathbf{x}$, наоборот, снижает уверенность правильного предсказания, величина штрафа увеличивается.

Цель злоумышленника — чтобы изменение величины $\mathbf{x}$ *увеличило* возвращаемую этой функцией величину штрафа, поскольку увеличение этой величины смещает результат от правильного предсказания.

Используя дифференциальное исчисление, можно рассчитать градиенты функции штрафа для исходного изображения.

Определив знак этих градиентов в соответствии с их направлением, мы получим необходимое направление движения искажения. Математически это можно выразить так:

$$sign(\nabla_x C(f(\mathbf{x};\Theta),\mathbf{y})).$$

Это фактически вектор такого же размера, что и входной вектор, всем значениям которого присвоены знаки плюс и минус.

Наконец, чтобы рассчитать величину искажения для создания вредоносного образа, необходимо умножить данное направление на величину небольшого изменения расстояния ε:

$$\mathbf{x}^{\mathbf{adv}} = \mathbf{x} + \varepsilon \cdot sign(\nabla_x C(f(\mathbf{x};\Theta),\mathbf{y})).$$

Какими бы крошечными ни были те искажения, которые вносит в пиксели изображения метод FGSM, соответствующее смещение иногда бывает слишком большим. Следовательно, мы можем улучшить этот метод за счет итеративного внесения очень мелких искажений до тех пор, пока изображение

не выйдет за границу категории, став, таким образом, вредоносным. Наряду с этим мы можем перепроверять направление градиента на каждой итерации на тот случай, если модель не является абсолютно линейной. Такой подход называют *базовым итерационным методом*[1].

Приведенный ниже фрагмент кода демонстрирует пример FGSM-атаки против классификатора данных из набора Fashion-MNIST, созданного нами в главе 3. Мы будем использовать в этом примере открытую библиотеку Foolbox.

Пример кода: градиентная атака

Если хотите поэкспериментировать с кодом FGSM-атаки, приведенным в этом разделе, воспользуйтесь блокнотом Jupyter: chapter06/fashionMNIST_foolbox_gradient.ipynb (http://bit.ly/2FiRQhO).

Прежде всего следует импортировать необходимые пакеты:

```
import numpy as np
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
```

Загрузите модель, сохраненную нами в главе 3, и прогоните через нее тестовые изображения:

```
fashion_mnist = keras.datasets.fashion_mnist
_, (test_images, test_labels) = fashion_mnist.load_data()
test_images = test_images/255.0

model = tf.keras.models.load_model("../models/fashionMNIST.h5") ❶

predictions = model.predict(test_images) ❷
```

❶ Загрузка классификатора данных из набора Fashion-MNIST, сохраненного нами в главе 3.

❷ Получение предсказаний модели для тестовых данных.

Выберите исходное (невредоносное) изображение и выведите его на экран (рис. 6.6) вместе с соответствующим предсказанием:

```
image_num = 7 ❶
```

[1] *Kurakin A. et al.* Adversarial Machine Learning at Scale, 2016. http://bit.ly/31Kr3EO.

```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

x = test_images[image_num]
y = np.argmax(predictions[image_num])
y_name = class_names[y]

print("Prediction for original image:", y, y_name)

plt.imshow(x, cmap=plt.cm.binary)
```

❶ Измените это число, если хотите выполнить атаку против другого изображения.

Данный код сгенерирует следующий вывод:

```
Prediction for original image: 6 Shirt
```

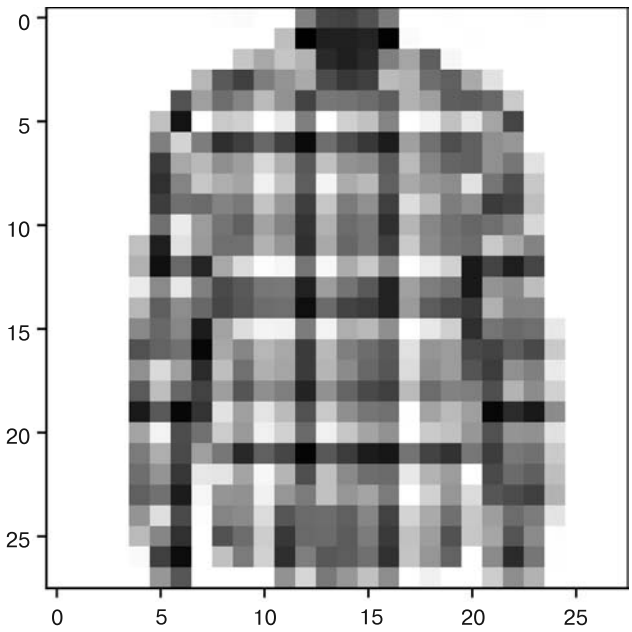


Рис. 6.6. Вывод программы

Затем создайте модель Foolbox на основе нашей модели Keras:

```
import foolbox
from foolbox.models import KerasModel
fmodel = foolbox.models.TensorFlowModel.from_keras(model, bounds=(0, 255))
```

Определите специфику атаки:

```
attack_criterion = foolbox.criteria.Misclassification() ❶
distance = foolbox.distances.Linfinity                  ❷
```

❶ `attack_criterion` определяет специфику атаки. В данном случае это просто обеспечение неправильной классификации.

❷ Расстояние искажения будет оптимизироваться с помощью L^{∞}-нормы.

Определите метод атаки:

```
attack = foolbox.attacks.GradientSignAttack(fmodel,
                                            criterion=attack_criterion,
                                            distance=distance)
```

И запустите атаку:

```
x_adv = attack(input_or_adv = x,
               label = y,
               unpack = False) ❶
```

❶ Установка `unpack = False` означает, что будет возвращаться объект `foolbox.adversarial.Adversarial`, а не изображение. Само изображение и другую информацию о вредоносном образе (например, величину расстояния от исходного изображения) можно будет извлечь из этого объекта.

Теперь распечатаем результаты (рис. 6.7):

```
preds = model.predict(np.array([x_adv.image])) ❶

plt.figure()

# Вывод на экран исходного изображения
plt.subplot(1, 3, 1)
plt.title(y_name
plt.imshow(x, cmap=plt.cm.binary)
plt.axis('off')

# Вывод на экран вредоносного изображения
plt.subplot(1, 3, 2
plt.title(class_names[np.argmax(preds[0])])
plt.imshow(x_adv.image, cmap=plt.cm.binary)
plt.axis('off')

# Вывод на экран различий
plt.subplot(1, 3, 3)
plt.title('Difference')
difference = x_adv.image - x
plt.imshow(difference, vmin=0, vmax=1, cmap=plt.cm.binary)
plt.axis('off')

print(x_adv.distance) ❷

plt.show()
```

❶ В этой строке мы получаем предсказания для вредоносного образа. `x_adv.image` — это компонент объекта Adversarial, представляющий вредоносное изображение.

❷ `x_adv.distance` — объект, представляющий искажение, необходимое для генерации вредоносного изображения.

Данный код сгенерирует следующий вывод:

```
normalized Linf distance = 1.50e-04
```

Рис. 6.7. Вывод программы

Данный код оптимизирован под использование L^{∞}-нормы. Если вы внимательно посмотрите на различия между двумя изображениями, то увидите, что мы слегка изменили большое количество пикселей.

Прежде чем продолжить, необходимо сделать важную оговорку о применимости всего того, что говорилось выше о линейности модели, к аудиоданным. Решения для преобразования речи в текст часто включают в себя предварительную обработку с применением таких средств, как MFCC и рекуррентные по своей природе LSTM-модули (см. раздел «Аудиоданные» на с. 87), что вносит в модель нелинейность. Поэтому для успешного создания вредоносного искажения с использованием такого метода, как FGSM, в случае системы для преобразования речи в текст потребуется использовать множество итеративных шагов, не пытаясь это сделать за один шаг. Кроме того, присутствие в системах для преобразования речи в текст последовательности сложной обработки делает генерацию функции потерь более сложной задачей по сравнению со сферой обработки изображений. Для этого необходимо минимизировать потери для примера аудиоданных в рамках всей последовательности обработки (включая такие шаги, как MFCC и CTC), что, в свою очередь, требует использования более сложных математических выкладок и повышенной вычислительной мощности[1].

[1] *Carlini, Wagner.* Audio Adversarial Examples.

Вредоносная значимость

В разделе «Что "думают" ГНС» на с. 114 введено понятие *карт значимости*, позволяющих графически представить те аспекты входных данных, которые играют наиболее важную роль при расчете предсказаний ГНС. Эта концепция применяется не только в сфере нейросетевой обработки; такие карты уже много лет используются для графического представления пикселей (или групп пикселей), имеющих наибольшую важность для конкретной задачи машинного распознавания образов.

Расчеты значимости также можно использовать при генерации вредоносных образов. Знать, какие признаки играют наиболее важную роль при определении класса, полезно в том случае, когда нам нужно ограничить искажение теми областями, изменение которых в наибольшей мере способствует смещению неопасных входных данных в сторону вредоносности. Такой подход, в частности, демонстрирует метод якобианских карт значимости (Jacobian Saliency Map Approach, JSMA)[1].

Метод JSMA сводится к вычислению показателя *вредоносной значимости* для каждого элемента входных данных. В случае изображений такой показатель вычисляется для каждого пиксельного значения (для трех значений на каждый пиксель для цветных изображений) и отражает его относительную важность для достижения вредоносной цели. При этом изменение пикселей с более высоким показателем в большей мере способствует превращению изображения во вредоносное, чем изменение пикселей с более низким показателем.

Показатель вредоносной значимости конкретного пикселя учитывает следующие два момента.

- Влияние изменения пикселя на *увеличение* вероятностного показателя для целевой категории (при целевой атаке).
- Влияние изменения пикселя на *уменьшение* вероятностных показателей, выдаваемых для всех других категорий.

Поскольку достижению вредоносной цели в наибольшей мере способствуют те изменения входных данных, которые оказывают большое влияние

[1] *Papernot N. et al.* The Limitations of Deep Learning in Adversarial Settings // 1st IEEE European Symposium on Security & Privacy, 2016. bit.ly/2ZyrSOQ.

и в первом и во втором случае, именно такие изменения следует вносить в первую очередь.

Метод JSMA отбирает те пиксели, которые оказывают наибольшее влияние, и сдвигает их на заданную величину в актуальном направлении (то есть либо увеличивает, либо уменьшает их значение). Фактически это смещение на заданное расстояние вдоль тщательно выбранного направления в многомерном входном пространстве в надежде на то, что изображение попадет в точку ландшафта предсказаний, удовлетворяющую критериям вредоносности. Если не удается достичь этой цели, данный процесс повторяется.

Метод JSMA минимизирует количество изменяемых пиксельных значений, что говорит о том, что для оценки величины изменений на этот раз используется L^0-норма.

Математические основы метода JSMA

Для вычисления показателя значимости нужно выяснить, какое относительное влияние оказывает небольшое изменение *каждого* входного значения на *каждое* выдаваемое предсказание.

Допустим, что у нас есть n входных значений и m выходных категорий. Все эти относительные изменения можно представить в виде большой матрицы — так называемой матрицы Якоби — размером $m \times n$. Для каждой комбинации из пиксельного значения и предсказания эта матрица содержит значение, указывающее, в какой мере изменение пиксельного значения влияет на соответствующее предсказание.

Каждое из этих значений является производной от функции ГНС f для предсказания j относительно изменения конкретного значения i:[1]

$$\frac{\partial f(\mathbf{x})_j}{\partial x_i}.$$

Соответственно выражение для всей матрицы будет выглядеть так:

$$s_{mn} = \frac{\partial f(\mathbf{x})_m}{\partial x_n}.$$

[1] Опять же, как и при обратном распространении, эти производные вычисляются с использованием цепного правила — математического метода, позволяющего рассчитывать производную от функции $f(\mathbf{x})$ путем представления этой функции в виде составляющих ее функций каждого слоя ГНС.

На основе этих производных матрицы Якоби, вычисленных для каждой пары из пиксельного значения и предсказания, мы можем рассчитать значимость каждого пикселя по отношению к целевой категории, используя следующую логику.

Значимость *увеличения* входного значения *i* для вредоносной целевой категории *t* рассчитывается следующим образом:

- если увеличение этого входного значения ведет к уменьшению целевого показателя вероятности, то производная:

$$\frac{\partial f(\mathbf{x})_j}{\partial x_i}$$

будет меньше 0. В таком случае данное входное значение не является вредоносно значимым и просто приравнивается нулю:

$$\text{если } \frac{\partial f(\mathbf{x})_t}{\partial x_i} < 0 \text{ то } s_{adv} = 0;$$

- подобным образом, если увеличение входного значения ведет к общему увеличению предсказаний, выдаваемых для всех других (нецелевых) категорий, это не способствует достижению вредоносной цели. Это сумма всех производных относительно входного значения для всех предсказаний, отличных от целевого. Если эта сумма больше 0, то входное значение опять же не считается вредоносно значимым и приравнивается 0:

$$\text{иначе если } \sum_{j \neq t} \frac{\partial f(\mathbf{x})_t}{\partial x_i} > 0 \text{ то } s_{adv} = 0;$$

- в противном случае входное значение является вредоносно значимым и нам остается лишь количественно оценить эту значимость, чтобы ее можно было сравнивать со значимостью других входных значений. Это можно сделать путем умножения частной производной, полученной на первом шаге, на сумму частных производных, полученную на втором шаге. В силу упомянутых ранее ограничений первый множитель всегда будет положительным, а второй — отрицательным; поэтому, чтобы сделать результат положительным, нам следует снабдить это произведение знаком минус:

$$\text{в ином случае } s_{adv} = -\frac{\partial f(\mathbf{x})_t}{\partial x_i} \cdot \sum_{j \neq t} \frac{\partial f(\mathbf{x})_t}{\partial x_i} >.$$

Сведя воедино все вышесказанное, мы можем создать карту значимости s^+, показывающую, в какой мере увеличение каждого входного значения способствует получению желаемого предсказания:

$$s^+(x_i,t)=\begin{cases}0 \text{ если } \dfrac{\partial f(\mathbf{x})_t}{\partial x_i}<0 \text{ или } \sum\limits_{j\neq t}\dfrac{\partial f(\mathbf{x})_t}{\partial x_i}>0;\\ -\dfrac{\partial f(\mathbf{x})_t}{\partial x_i}\cdot\sum\limits_{j\neq t}\dfrac{\partial f(\mathbf{x})_t}{\partial x_i} \text{ в ином случае.}\end{cases}$$

Эта карта значимости отражает вредоносную значимость входных значений в случае их *увеличения*. Метод JSMA также подразумевает определение того, какие пиксельные значения в наибольшей мере способствуют обеспечению неправильной классификации в случае их *уменьшения*. Лишь немного изменив логику, мы получим еще одну карту значимости, s^-:

$$s^+(x_i,t)=\begin{cases}0 \text{ если } \dfrac{\partial f(\mathbf{x})_t}{\partial x_i}>0 \text{ или } \sum\limits_{j\neq t}\dfrac{\partial f(\mathbf{x})_t}{\partial x_i}<0;\\ -\dfrac{\partial f(\mathbf{x})_t}{\partial x_i}\cdot\sum\limits_{j\neq t}\dfrac{\partial f(\mathbf{x})_t}{\partial x_i} \text{ в ином случае.}\end{cases}$$

(Обратите внимание, здесь поменялись местами знаки > и <.)

Метод JSMA извлекает по одному наиболее значимому пикселю из карт s^+ и s^- и изменяет каждый из двух пикселей на небольшую величину в соответствующем направлении. Затем этот процесс повторяется до тех пор, пока входной сигнал не станет вредоносным.

Приведенный ниже фрагмент кода демонстрирует пример атаки с использованием карт значимости и библиотеки Foolbox против классификатора ResNet50, созданного нами в главе 4.

Пример кода: атака с использованием значимости

Весь код для выполнения данной атаки можно найти в блокноте Jupyter: chapter06/resnet50_foolbox_saliency.ipynb (bit.ly/2KpRKsL).

Сначала выберем нашу исходную невредоносную фотографию и прогоним ее через классификатор:

```
original_image_path = '../images/koala.jpg'
x = image_from_file(original_image_path, [224,224]) ❶
```

❶ Это вспомогательная утилита, включенная в GitHub-репозиторий книги. Она читает файл и изменяет его размер.

Затем следует импортировать необходимые библиотеки и модель ResNet50. При этом мы передаем невредоносное изображение модели ResNet50, чтобы посмотреть, какое предсказание она для него выдает:

```
import tensorflow as tf
from tensorflow import keras
from keras.applications.resnet50 import ResNet50
from keras.applications.resnet50 import preprocess_input
from keras.applications.resnet50 import decode_predictions

model = ResNet50(weights='imagenet', include_top=True)

x_preds = model.predict(np.expand_dims(preprocess_input(x), 0))
y = np.argmax(x_preds)
y_name = decode_predictions(x_preds, top=1)[0][0][1]

print("Prediction for image: ", y_name)
```

Данный код сгенерирует следующий вывод:

```
Prediction for image: koala
```

Теперь создадим модель Foolbox на основе модели ResNet50. При этом нужно указать, какая предварительная обработка нам потребуется:

```
import foolbox


preprocessing = (np.array([103.939, 116.779, 123.68]), 1) ❶
fmodel = foolbox.models.TensorFlowModel.from_keras(model,
                                                   bounds=(0, 255),
                                                   preprocessing=preprocessing)
```

❶ Модель Foolbox может выполнять предварительную обработку изображений, чтобы сделать их пригодными для модели ResNet50. Это включает в себя нормализацию данных относительно средних RGB-значений набора ImageNet, на которых производилось исходное обучение классификатора. Переменная `preprocessing` определяет способ осуществления шага предварительной обработки. Аналогичная нормализация выполняется в функции `keras.applications.resnet50.preprocess_input`, которую мы вызывали ранее для подготовки входных данных для модели ResNet50. Если вы хотите лучше понять, что представляет собой предварительная обработка, и попробовать выполнить ее самостоятельно, ознакомьтесь с содержимым блокнота Jupyter `chapter04/resnet50_preprocessing.ipynb` (http://bit.ly/2WO9rUx).

Как уже упоминалось в разделе «Классификация изображений с помощью сети ResNet50» на с. 99, модель ResNet50 обучена на изображениях с порядком следования каналов BGR, а не RGB. Соответственно, следующий шаг (как и в функции `keras.applications.resnet50.preprocess_input`) сводится к тому, чтобы переключиться к порядку следования каналов BGR:

```
x_bgr = x[..., ::-1]
```

Затем следует настроить атаку библиотеки Foolbox:

```
attack_criterion = foolbox.criteria.Misclassification()
attack = foolbox.attacks.SaliencyMapAttack(fmodel, criterion=attack_criterion)
```

и запустить ее:

```
x_adv = attack(input_or_adv = x_bgr,
                              label = y,
                              unpack = False)
```

Теперь отобразим метку и название категории полученного нами вредоносного изображения:

```
x_adv = adversarial.image[..., ::-1] ❶

x_adv_preds = model.predict(preprocess_input(x_adv[np.newaxis].copy()))
y_adv = np.argmax(x_adv_preds)
y_adv_name = decode_predictions(x_adv_preds, top=1)[0][0][1]

print(print("Prediction for image: ", y_adv_name))
```

❶ Извлечение вредоносного изображения из объекта `foolbox.adversarial` и возвращение к порядку следования каналов RGB.

Этот код сгенерирует следующий вывод:

```
Prediction for image: weasel
```

Наконец, выведем на экран оба изображения и имеющиеся между ними различия (рис. 6.8):

```
import matplotlib.pyplot as plt

plt.figure()

# Вывод на экран исходного изображения
plt.subplot(1, 3, 1)
plt.title(y_name)
plt.imshow(x)
plt.axis('off')
```

```
# Вывод на экран вредоносного изображения
plt.subplot(1, 3, 2)
plt.title(y_adv_name)
plt.imshow(x_adv)
plt.axis('off')

# Вывод на экран различий
plt.subplot(1, 3, 3)
plt.title('Difference')
difference = x_adv - x

# Присвоение числа 255 неизменившимся пикселям,
# чтобы они не отображались на графике различий
difference[difference == 0] = 255 ❶
plt.imshow(abs(difference))
plt.xticks([])
plt.yticks([])

plt.show()
```

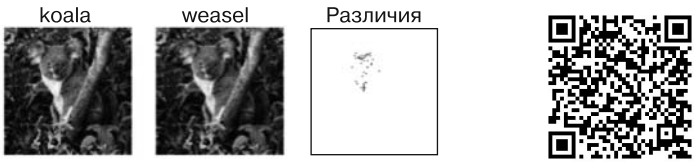


Рис. 6.8. Вывод программы

Представленный на рис. 6.8 фрагмент различий показывает, что изменилось сравнительно небольшое количество пикселей. Возможно, вам удастся заметить соответствующее искажение в центральной части вредоносного изображения.

Повышение надежности вредоносного искажения

Методы FGSM и JSMA позволяют успешно генерировать вредоносные образы. Однако, поскольку генерируемые ими входные данные находятся очень близко к классификационным границам, на них часто плохо сказывается предварительная обработка или активная защита целевой системы. Так, например, небольшое изменение пикселей вредоносного изображения может привести к изменению его категории.

Чтобы обеспечить себе больший уровень надежности, злоумышленник должен генерировать входной сигнал, который бы с большей степенью

уверенности был вредоносным. Именно это и позволяет осуществлять альтернативный метод атаки, предложенный Карлини и Вагнером[1] (атака C&W). Этот метод итеративно минимизирует L^2-норму, с помощью которой оценивается величина изменения, в то же время обеспечивая максимальную разницу по степени уверенности между целевой категорией вредоносной атаки и ближайшей к ней наиболее вероятной категорией. Это увеличивает тот зазор, при превышении которого вредоносный образ становится невредоносным. Такую атаку называют *атакой Карлини и Вагнера*.

Математические основы атаки Карлини и Вагнера

Допустим, у нас есть целевая атака, описываемая следующей формулой:

$$f(\mathbf{x}^{\text{adv}}) = y_t,$$

где y_t — целевая категория вредоносной атаки (определяемая на основе выходного вектора вероятностей).

Версия атаки C&W на базе L^2-нормы минимизирует L^2-норму.

Возведение L^2-нормы в квадрат снижает вычислительную сложность, так как это избавляет нас от необходимости вычислять квадратный корень. Соответственно, показатель расстояния между вредоносным и невредоносным изображениями определяется следующим образом:

$$\|d\|_2^2 = \left\|\mathbf{x}^{\text{adv}} - \mathbf{x}\right\|_2^2.$$

Атака Карлини и Вагнера повышает надежность за счет создания такого вредоносного образа, который обеспечивает максимальную разницу между предсказанием целевой категории вредоносной атаки и предсказанием ближайшей наиболее вероятной категории. Эти предсказания представляют собой логиты, возвращаемые тем слоем, который предшествует слою softmax-обработки. Их можно выразить следующим образом:

$$Z(\mathbf{x}^{\text{adv}}).$$

Допустим, что наибольшим предсказанием является логит t; тогда второе по величине предсказание для вредоносного образа можно выразить так:

$$\max\left\{Z(\mathbf{x}^{\text{adv}})_i : i \neq t\right\}.$$

[1] *Carlini, Wagner.* Towards Evaluating the Robustness of Neural Networks.

Если теперь мы вычтем целевое предсказание, то получим разницу между ним и вторым по величине значением:

$$\max\left\{Z(\mathbf{x}^{\text{adv}})_i : i \neq t\right\} - Z(\mathbf{x}^{\text{adv}})_t.$$

Это выражение возвращает отрицательное число, отражающее степень уверенности для вредоносного образа. Чем больше его абсолютная величина, тем больше степень уверенности для примера.

Добавив в это выражение параметр *k*, мы получим возможность настройки степени уверенности. Если мы вычтем параметр *k* из предыдущего выражения и найдем максимум от полученного выражения, это позволит нам задавать с его помощью требуемую степень уверенности. Чем больше будет величина параметра *k*, тем больше будет степень уверенности для примера:

$$\max\left\{\max\left\{Z(\mathbf{x}^{\text{adv}})_i : i \neq t\right\} - Z(\mathbf{x}^{\text{adv}})_t, -k\right\}.$$

Эту функцию потерь для параметра уверенности мы будем вызывать для вредоносного образа *l*. Сведя все воедино, получим:

$$\text{minimize}\left\{\left\|\mathbf{x}^{\text{adv}} - \mathbf{x}\right\|_2^2 + c.l(\mathbf{x}^{\text{adv}})\right\},$$

где:

$$l(\mathbf{x}^{\text{adv}}) = \max\left\{\max\left\{Z(\mathbf{x}^{\text{adv}})_i : i \neq t\right\} - Z(\mathbf{x}^{\text{adv}})_t, -k\right\}.$$

Внимательно посмотрев на это выражение, вы могли заметить в нем дополнительный параметр *c*. Этот параметр призван уравновесить между собой две составляющие данного выражения — положительный показатель расстояния и равный 0 или отрицательный показатель степени уверенности, — чтобы исключить перекос в сторону показателя степени уверенности, способный привести к тому, что расстояние (величина искажения) станет слишком большим. Величина параметра *c* определяется с помощью алгоритма двоичного поиска.

Разновидности методов белого ящика

В данном разделе мы рассмотрели некоторые из существующих методов белого ящика для генерации вредоносных образов. Это далеко не все из них, и, вне всяких сомнений, их число со временем пополнится новыми.

Общей особенностью всех методов белого ящика является то, что они стремятся оптимизировать поиск во входном пространстве за счет использова-

ния вычислительно осуществимого алгоритма. Все эти методы прямо или косвенно используют информацию о градиентах модели для минимизации величины требуемого вредоносного искажения. Оптимизация поиска может включать в себя определенную аппроксимацию или случайную генерацию, в результате чего величина искажения может быть чуть больше, чем необходимо. Кроме того, из-за различий в используемых алгоритмах поиска во входном пространстве разные методы могут возвращать разные вредоносные образы.

Методы ограниченного черного ящика

Методы ограниченного черного ящика итеративно корректируют вредоносное искажение на основе возвращаемого моделью результата. Этим результатом, к примеру, может быть окончательная категория изображения («кошка» или «собака») или текст, выдаваемый системой для преобразования речи в текст. Данные методы подразумевают отсутствие у злоумышленника доступа к самой модели или предварительным показателям (предсказаниям) на выходе ее выходного слоя. Фактически в данном случае присутствует косвенность — выходной сигнал ГНС-модели определенным образом обрабатывается и упрощается для получения конечного результата, и только этот результат доступен злоумышленнику.

На первый взгляд не совсем понятно, как в таком случае можно эффективно осуществлять поиск во входном пространстве. Один из возможных подходов заключается в том, чтобы итеративно вносить небольшое изменение во входной сигнал, прогонять его через модель и проверять, не изменилась ли выдаваемая для него категория. Однако эта стратегия вряд ли будет эффективной, поскольку до тех пор, пока не изменится выдаваемая категория, у нас не будет возможности узнать, ведет ли каждое небольшое изменение к смещению изображения в сторону нужной нам вредоносной области входного пространства. Этот метод грубой силы будет работать слишком медленно и нескладно; злоумышленнику нужен план получше.

Разумной и в то же время удивительно простой стратегией является предложенный Бренделем и др. метод *граничной атаки*[1]. Как может выглядеть целевая граничная атака, показано на рис. 6.9. В качестве исходных данных

[1] *Brendel W. et al.* Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models // Proceedings of the International Conference on Learning Representations, 2018. bit.ly/2Y7Zi6w.

этот метод использует исходное изображение (в нашем случае изображение кроссовок) и некоторое изображение (в нашем случае изображение сандалий), которые обозначены на рис. 6.9 соответственно кружком и квадратом. В качестве второго изображения выбирается изображение, относимое моделью к целевой категории. То есть в данном случае нам нужно, чтобы изображение кроссовок было отнесено к категории сандалий и чтобы кроссовки на нем по-прежнему выглядели как кроссовки.

Граничная атака начинает свое движение не от исходного изображения кроссовок, а от изображения сандалий. Она итеративно смещает это изображение в сторону кроссовок, в то же время следя за тем, чтобы ни один шаг итерации не привел к выходу за пределы области целевой категории.

Граничная атака начинает свою работу с этапа инициализации. На изображение, относящееся к целевой категории вредоносной атаки, накладывается «разбавленная» версия исходного изображения для создания входного сигнала, находящегося у самого края вредоносной области. Это достигается путем наложения на изображение сандалий избранных пикселей исходного изображения кроссовок до тех пор, пока результирующее изображение не сместится внутри вредоносной области к самому краю. Для этого потребуется достаточно много итераций — нужно выполнять наложение, проверять результат и повторять это снова до тех пор, пока не будет достигнута точка, в которой любое дальнейшее изменение делает изображение невредоносным. Здесь изображение немного выйдет за границу классификации, оказавшись в области, относящейся к категории кроссовок. То есть процесс зайдет чуть дальше, чем нужно. И чтобы снова вернуть изображение в область вредоносной категории (категории сандалий), нужно *совсем немного* убавить величину составляющей, извлекаемой из исходного изображения.

Основной принцип сводится к тому, чтобы смещать изображение в сторону исходного изображения и остановиться на границе непосредственно перед переходом изображения в область исходной категории. На рис. 6.9 положение изображения по завершении этапа инициализации обозначено треугольником с цифрой 1.

Таким образом, к концу этапа инициализации изображение сместится внутри вредоносной области к самой границе. Однако эта точка вряд ли будет находиться достаточно близко к исходному изображению, чтобы искажение было незаметным. Вполне возможно, что содержимое изображения при этом будет по-прежнему выглядеть как сандалии. Далее алгоритм продвигается вдоль границы, проверяя на каждом шаге, какие случайные искажения могут

приблизить входной сигнал к исходному изображению кроссовок. Получаемые при этом входные сигналы подаются в целевую ГНС и отбрасываются, если происходит выход за пределы вредоносной области. Каждая итерация в процессе приближения результирующего изображения к исходному изображению может сама состоять из несколько шагов. В определенный момент изображение настолько приблизится к исходному изображению, что станет очень похожим на него, в то же время оставаясь в границах вредоносной категории.

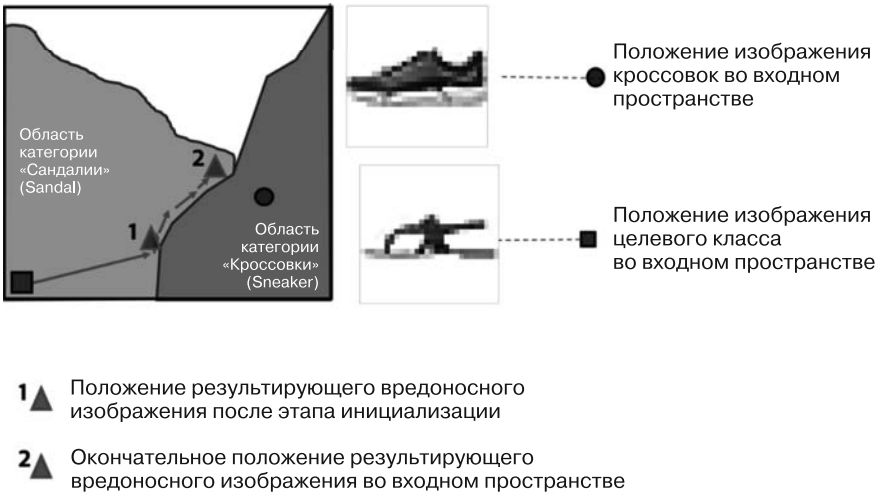


Рис. 6.9. Метод ограниченного черного ящика; целевая граничная атака

Посмотрим, как может выглядеть код для выполнения целевой граничной атаки против классификатора изображений из набора Fashion-MNIST. При этом мы постараемся «превратить» кроссовки в сандалии, как показано на рис. 6.9.

Пример кода: граничная атака

Весь код для выполнения данной атаки можно найти в блокноте Jupyter: chapter06/fashionMNIST_foolbox_boundary.ipynb (http://bit.ly/2L2x61i).

Прежде всего код граничной атаки должен задать исходное изображение (x) и изображение, выступающее в роли отправной точки. Сначала зададим

исходное изображение, а также определим и отобразим его категорию (рис. 6.10):

```
original_image_num = 9

x = test_images[original_image_num]
y = np.argmax(predictions[original_image_num]) ❶
y_name = class_names[y]

print("Prediction for original image:", y, y_name)

plt.imshow(x, cmap=plt.cm.binary)
```

❶ y — предсказание исходного (невредоносного) изображения.

Данный код сгенерирует следующий вывод:

```
Prediction for original image: 7 Sneaker
```

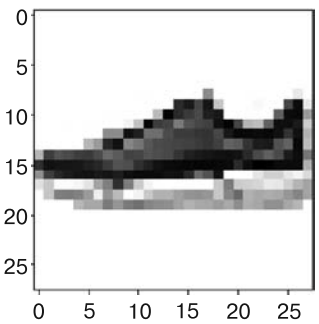


Рис. 6.10. Вывод программы

Теперь выберем изображение, выступающее в роли отправной точки, и тоже определим и отобразим его категорию (рис. 6.11). Это изображение относится к нужной нам вредоносной категории:

```
starting_point_image_num = 52

starting_point_image = test_images[starting_point_image_num]
y_adv = np.argmax(predictions[starting_point_image_num]) ❶
y_adv_name = class_names[y_adv]

print("Prediction for starting point image:", y_adv, y_adv_name)
import matplotlib.pyplot as plt

plt.imshow(starting_point_image, cmap=plt.cm.binary)
```

❶ y_adv — целевое предсказание вредоносной атаки.

Данный код сгенерирует следующий вывод:

```
Prediction for starting point image: 5 Sandal
```

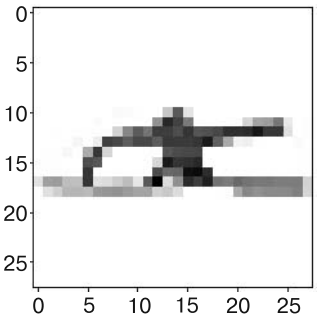


Рис. 6.11. Вывод программы

Теперь следует подготовить атаку библиотеки Foolbox:

```
import foolbox

fmodel = foolbox.models.TensorFlowModel.from_keras(model, bounds=(0, 1))
attack_criterion = foolbox.criteria.TargetClass(y_adv) ❶
attack = foolbox.attacks.BoundaryAttack(fmodel, criterion=attack_criterion)
```

❶ Поскольку это целевая атака, мы используем критерий `TargetClass`.

И запустить атаку:

```
x_adv = attack(input_or_adv = x,
               label = y,
               starting_point = starting_point_image,
               unpack = False,
               log_every_n_steps = 500)
```

Данный код сгенерирует следующий вывод:

```
run with verbose=True to see details
Step 0: 5.60511e-02, stepsizes = 1.0e-02/1.0e-02:
Step 500: 1.44206e-02, stepsizes = 1.5e-02/2.0e-03:
Step 1000: 3.43213e-03, stepsizes = 1.5e-02/1.3e-03: d. reduced by 0.26% (...)
Step 1500: 1.91473e-03, stepsizes = 6.7e-03/5.9e-04: d. reduced by 0.12% (...)
Step 2000: 1.54220e-03, stepsizes = 3.0e-03/1.7e-04: d. reduced by 0.03% (...)
Step 2500: 1.41537e-03, stepsizes = 8.8e-04/5.1e-05: d. reduced by 0.01% (...)
Step 3000: 1.37426e-03, stepsizes = 5.9e-04/2.3e-05: d. reduced by 0.00% (...)
Step 3500: 1.34719e-03, stepsizes = 3.9e-04/2.3e-05:
Step 4000: 1.32744e-03, stepsizes = 3.9e-04/1.5e-05: d. reduced by 0.00% (...)
Step 4500: 1.31362e-03, stepsizes = 1.7e-04/1.0e-05:
Step 5000: 1.30831e-03, stepsizes = 5.1e-05/2.0e-06:
```

По умолчанию атака останавливается в случае схождения алгоритма в одной точке или после выполнения 5000 итераций. Надо надеяться, что к этому моменту мы продвинулись вдоль классификационной границы достаточно близко к исходному изображению, чтобы результирующее изображение было похожим на него. Посмотрим, так ли это на самом деле (см. вывод программы на рис. 6.12):

```
preds = model.predict(np.array([x_adv.image]))

plt.figure()

# Вывод на экран исходного изображения
plt.subplot(1, 3, 1)
plt.title(y_name)
plt.imshow(x, cmap=plt.cm.binary)
plt.axis('off')

# Вывод на экран вредоносного изображения
plt.subplot(1, 3, 2)
plt.title(class_names[np.argmax(preds[0])])
plt.imshow(x_adv.image, cmap=plt.cm.binary)
plt.axis('off')

# Вывод на экран различий
plt.subplot(1, 3, 3)
plt.title('Difference')
difference = x_adv.image - x
plt.imshow(difference, vmin=0, vmax=1, cmap=plt.cm.binary)
plt.axis('off')

plt.show()
```

Рис. 6.12. Вывод программы

На рис. 6.13 показано, как выглядит изображение в ходе выполнения данной атаки с интервалом 100 итераций (до 1200 итераций). Сандалии постепенно становятся все более похожими на исходное изображение, оставаясь при этом в своей категории. Под каждым изображением в скобках приводится величина предсказания для категории сандалий. Как видите, в ходе

оптимизации мы продвигаемся вдоль границы с величиной предсказания, близкой к 0,5. Код для вывода результатов, приведенных на рис. 6.13, также включен в блокнот Jupyter `chapter06/fashionMNIST_foolbox_boundary.ipynb` (bit.ly/2L2x61i).

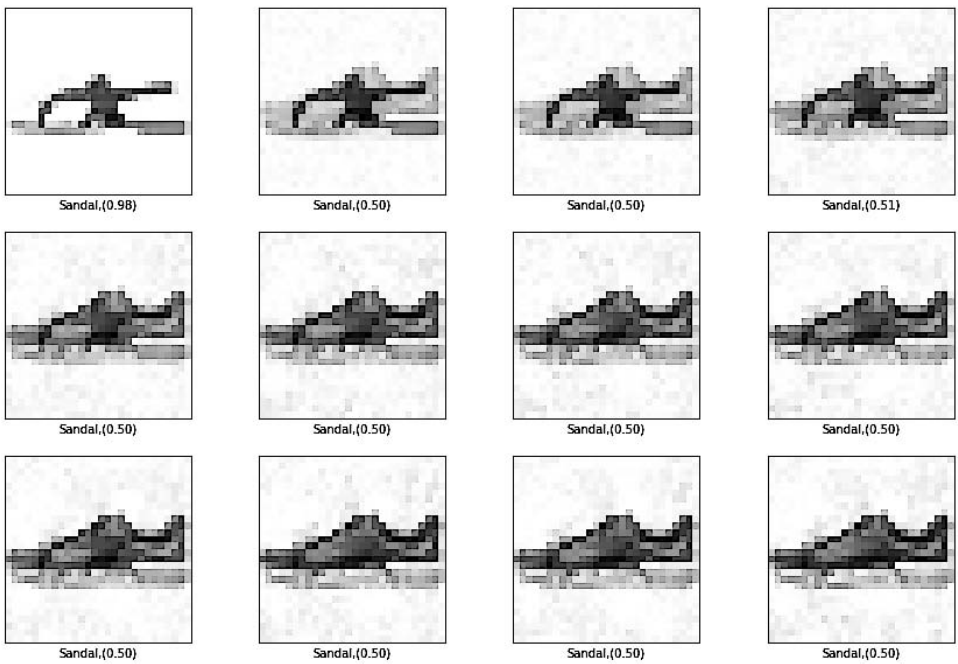


Рис. 6.13. Граничная атака начинается с изображения, относящегося к целевой категории, и постепенно смещает его ближе к исходному изображению

Граничная атака представляет собой очень мощную разновидность атаки, способную успешно создавать вредоносные входные данные. В то же время она требует выполнения тысяч итераций, каждая из которых может включать в себя несколько обращений к ГНС.

Методы черного ящика с оценкой

Методы с оценкой занимают промежуточное положение между методами белого ящика и методами ограниченного черного ящика. Хотя в научной литературе эти методы называются просто методами *черного ящика*, в данной

книге мы будем называть их методами *с оценкой,* чтобы провести четкое различие между этими двумя разновидностями методов.

Методы с оценкой требуют доступа к значениям вероятности, выдаваемым моделью для каждой категории; то есть, подав некоторый входной сигнал, злоумышленник может получить те вероятностные показатели, на основе которых ГНС принимает окончательное решение (такое как выбор категории). Также следует отметить, что злоумышленнику может быть доступна лишь ограниченная часть этих показателей (например, пять самых высоких вероятностей).

На первый взгляд может показаться, что методы с оценкой имеют больше сходства с методами ограниченного черного ящика, поскольку в обоих случаях злоумышленнику доступны только входные и выходные данные. Однако доступ к оценкам можно считать особой привилегией, ведь обычно у злоумышленника нет доступа к предварительным выходным данным ГНС. Поэтому у методов с оценкой гораздо больше общего с методами белого ящика. Они аппроксимируют алгоритм модели на основе выходных вероятностных оценок, после чего производят интеллектуальный поиск искажения, необходимого для достижения вредоносной цели. Но, в отличие от методов белого ящика, у злоумышленника нет доступа к алгоритму модели. Это не позволяет ему рассчитать градиенты алгоритма, используемые в описанных ранее методах белого ящика. (Для поиска вредоносного образа можно использовать и ряд других способов, например генетические алгоритмы, суть которого объясняется в отступлении далее.)

Генетический подход к поиску

Генетические алгоритмы широко используются в программном обеспечении для выбора хорошего варианта из множества доступных. В их основе лежит концепция постепенно совершенствуемого множества допустимых вариантов, каждое последующее поколение которого является более совершенной версией предыдущего.

Генерирование вредоносных образов с помощью генетической стратегии начинается с поколения невредоносных входных сигналов, каждый из которых создается путем внесения случайного искажения в исходный неопасный пример. В контексте аудиоданных это поколение может быть создано путем извлечения из исходного аудиоклипа нескольких случайных подклипов

и внесения небольшого случайного искажения в каждый из них[1]. Все примеры оцениваются на предмет их вредоносной эффективности, исходя из того, какой результат выдает для них сеть.

Потом с помощью процесса, включающего в себя отбор, скрещивание и мутацию, создается следующее поколение примеров. Сначала в текущем поколении отбирается некоторое подмножество «родителей»; при этом предпочтение отдается примерам с высоким показателем эффективности. Затем путем объединения (скрещивания) вредоносных аспектов родителей и добавления некоторых случайных шумов (мутаций) создаются потомки, в совокупности образующие следующее поколение. Каждый член нового поколения опять же оценивается на его вредоносную эффективность на основе выдаваемых сетью показателей.

Выполнение данного процесса прекращается, когда удается создать вредоносный образ или когда количество поколений достигает максимально допустимого порога (не успев обеспечить требуемый результат).

Возможно, у вас уже созрел вопрос: зачем организация, заботящаяся о безопасности своей модели, будет делать доступными эти показатели? Так, например, если автоматически выявлено неприличное содержание в изображении, загруженном на сайт социальной сети, это изображение включается в число удаляемых, а загрузивший его человек в лучшем случае получит предупреждение или уведомление о том, что его изображение будет удалено по соображениям цензуры. Он не получит подробный отчет о выходных вероятностных показателях каждой категории, выданных нейросетью для данного изображения. Таким образом, в подобном случае организация, конечно, не будет делать доступной информацию о показателях. Однако целый ряд API с открытым исходным кодом, созданных в целях демонстрации и дальнейшего развития нейросетевых технологий, делает эти показатели доступными. И хотя это не ставит под угрозу безопасность самих моделей, как вы увидите в главе 7, злоумышленник может использовать общедоступные API в качестве замещающей модели, тем самым избавив себя от необходимости тратить силы на создание собственной модели.

---

[1] *Alzantot M. et al.* Did You Hear That? Adversarial Examples Against Automatic Speech Recognition // Conference on Neural Information Processing Systems, Machine Deception Workshop, 2017. http://bit.ly/ 2ITvtR3.

Резюме

В этой главе рассмотрен ряд методов генерации вредоносных искажений. Существует много других разновидностей этих методов, число которых регулярно пополняется новыми.

Хотя эти методы были рассмотрены в контексте изображений, их базовые принципы в той же мере применимы и к аудио- или видеоданным. Кроме того, не стоит думать, что для создания вредоносных входных данных всегда нужны математические методы оптимизации — при отсутствии у целевой системы какой-либо защиты также могут сработать и более простые методы. Например, было доказано[1], что алгоритмы, выполняющие поиск и реферирование видеоданных, можно обмануть путем простой вставки неподвижных изображений в некоторые места видеофайла. Эти неподвижные изображения заставляют нейросеть возвращать неверный результат, но в то же время вставляются достаточно редко, для того чтобы их могли заметить люди.

Как вы смогли убедиться, алгоритмы генерации вредоносных искажений фактически реализуют математические методы оптимизации. В части IV, где рассмотрены способы защиты, вы увидите, что многие способы защиты от вредоносных образов сводятся к изменению или расширению алгоритма модели. То есть создание вредоносных образов, способных преодолеть защиту, по-прежнему остается задачей оптимизации, только в отношении другого алгоритма. В силу этого многие методы защиты успешно ограничивают число доступных злоумышленнику методов, но не могут гарантировать абсолютную защиту.

В этой главе рассмотрены математические методы генерации вредоносных образов для ГНС-моделей. В следующей части поговорим о том, как злоумышленник может применять эти теоретические подходы для проведения атак против реальных систем, которые используют ИИ.

---

[1] *Hossein H. et al.* Deceiving Google's Cloud Video Intelligence API Built for Summarizing Videos, 2017. bit.ly/2FhbDxR.

ЧАСТЬ III

Понимание реальных угроз

В части II рассмотрены математические и алгоритмические методы, с помощью которых можно создать входные данные, способные обмануть ГНС. В части III на примере этих методов вы увидите, с какими угрозами приходится иметь дело в реальных ситуациях, когда целевая ГНС входит в состав более крупной компьютерной системы. Такой более крупной системой может быть, например, устройство с голосовым управлением, программное обеспечение для фильтрации веб-контента или автономное транспортное средство.

В главе 7 речь пойдет о том, какие методы может использовать злоумышленник для запуска вредоносной атаки в случае ограниченного доступа к целевой системе. Вы узнаете, что может усложнить (или упростить) для злоумышленника проведение атаки, а также ознакомитесь с существующими схемами атак: прямой, с копированием и с переносом. Будет также рассмотрен вопрос о том, может ли атака, созданная с расчетом на одну целевую систему, сработать против другой системы.

В главе 8 будут перечислены дополнительные сложности, с которыми придется столкнуться злоумышленнику при проведении атаки в физическом мире. В таких сценариях атака уже не ограничивается исключительно сферой цифровых данных, превращаясь во вредоносные объекты или звуки, создаваемые и существующие в физическом мире. В этой главе вы узнаете, как следует создавать такие физические образы, чтобы они оставались вредоносными независимо от окружающих условий или положения камеры или микрофона, используемых для получения данных.

Понимание угроз играет фундаментальную роль в обеспечении защиты любой системы. Соответственно, материал настоящей части — основа для изучения способов защиты в части IV.

ГЛАВА 7

Схемы атак против реальных систем

В этой главе вы узнаете, какие схемы атак может использовать злоумышленник для генерации вредоносных входных данных, исходя из своих целей и возможностей. Приведенные схемы атак основаны на методах, рассмотренных в главе 6, и, как вы увидите далее, выбор соответствующего подхода зависит от таких факторов, как степень доступа злоумышленника к целевой системе для тестирования и разработки вредоносных входных данных и степень понимания им целевой модели и последовательности обработки. Мы также поговорим о том, можно ли многократно использовать вредоносное искажение или заплатку в разных изображениях или аудиофайлах.

Схемы атак

В главе 6 рассмотрены различные методы генерации вредоносных образов. Эти методы хорошо показали себя в «лабораторном» окружении. Однако насколько они эффективны в реальной ситуации, когда у злоумышленника ограниченное понимание или ограниченный доступ к целевой модели и той более крупной системе, составной частью которой она является? Создать вредоносные входные данные таким образом, чтобы они были эффективными в реальной ситуации, — весьма непростая задача для любого злоумышленника.

При создании вредоносных входных данных и последующем запуске атаки злоумышленник может руководствоваться одной из нескольких схем. Эти схемы различаются по уровню сложности и наличию ресурсов, необходимых для генерации вредоносных образов. Еще одно различие представляет требуемый уровень доступа к целевой системе или осведомленности о ней. Выбор той или иной схемы также часто зависит от необходимого уровня надежности или скрытности атаки.

Существующие схемы атак можно разбить на такие основные категории, как:

- *прямая атака* — злоумышленник разрабатывает атаку непосредственно против целевой системы;
- *атака с копированием* — при разработке атаки злоумышленнику доступна *точная копия* целевой ГНС;
- *атака с переносом* — злоумышленник разрабатывает атаку против замещающей модели, которая *аппроксимирует* целевую систему;
- *универсальная атака с переносом* — злоумышленник не располагает информацией о целевой модели. Он создает вредоносный входной сигнал, применимый к некоторой группе моделей, по своей функциональности аналогичных целевой модели, в надежде, что этот сигнал сработает и против целевой ГНС.

Общая схема различий между четырьмя схемами атак представлена на рис. 7.1.

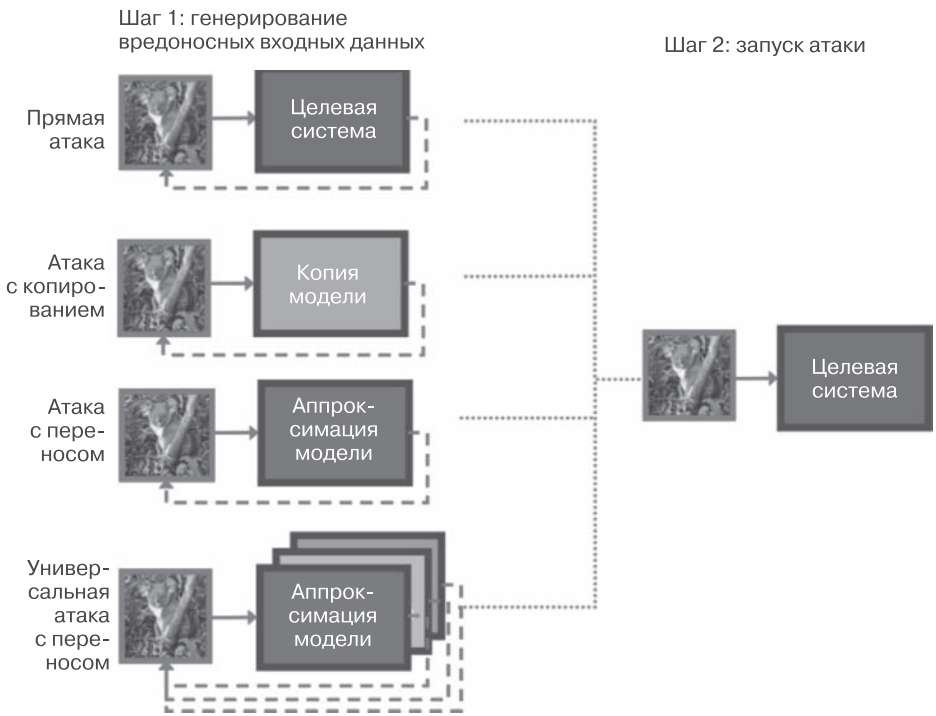


Рис. 7.1. Различные схемы атак с созданием вредоносных входных данных

В следующих разделах мы рассмотрим каждую из этих схем более подробно. А пока предположим, что злоумышленник может манипулировать цифровым содержимым; использование физических вредоносных образов рассмотрено в главе 8.

Различные названия схем атак

Одни и те же схемы атаки могут называться в литературе по-разному. Отсутствие общепринятой терминологии ведет к путанице.

Так, например, под *черным ящиком* иногда подразумевается прямая атака. Подобно этому, атаку с копированием, использующую методы белого ящика, иногда называют просто атакой белого ящика.

Во избежание неоднозначности термины «*белый ящик*» и «*черный ящик*» не используются в данной книге для обозначения схем атак, поскольку эти термины также подразумевают использование конкретного алгоритмического метода.

Возьмем, к примеру, атаку с копированием. В силу полной осведомленности злоумышленника о модели, ее архитектуре и параметрах в данном случае кажется логичным генерировать вредоносные входные данные с использованием метода белого ящика. Однако злоумышленник может использовать и метод черного ящика (например, граничную атаку), скажем, в силу того, что на него меньше влияют используемые методы защиты, или потому, что его проще реализовать. Сходным образом, хотя атаку с переносом иногда называют «черным ящиком», такая атака может применять к замещающей модели и методы белого ящика, и методы черного ящика.

Прямая атака

В случае прямой атаки у злоумышленника есть возможность подавать входные данные в реальную целевую систему и получать соответствующие результаты, что обеспечивает точную обратную связь для коррекции вредоносных входных данных.

При проведении такой атаки злоумышленнику обычно доступна лишь ограниченная информация о результатах, возвращаемых целевой системой[1].

[1] В реальных системах, представляющих интерес для атаки, злоумышленнику, как правило, недоступны вероятностные показатели, выдаваемые целевой системой.

Кроме того, он может получать обратную связь не напрямую, а путем логического вывода. Например, в случае неудачной попытки загрузить видеофайл на сайт злоумышленник может предположить, что этот файл был отнесен к категории видео со сценами насилия, хотя и не получит эту категорию в явном виде. Соответственно, для создания вредоносных входных данных в таком случае следует использовать метод черного ящика. Как уже говорилось в разделе «Методы ограниченного черного ящика» на с. 157, методы ограниченного черного ящика итеративно корректируют подаваемые в систему запросы на основе возвращаемых ответов, постепенно видоизменяя входные данные и смещая их в требуемую вредоносную область входного пространства.

Проведение прямой атаки — весьма непростая задача. Чтобы найти идеально подходящий вредоносный входной сигнал, используя один из методов черного ящика, например граничную атаку, нужно выполнить очень много итераций (десятки тысяч). Каждая итерация, в свою очередь, может включать в себя несколько запросов к целевой ГНС. Таким образом, в итоге мы имеем громадное количество запросов, выполнение которых вряд ли останется незамеченным защищающейся организацией! Более того, ограничения по пропускной способности и времени задержки, характерные для коммерческого развертывания, будут замедлять скорость обработки этих запросов. На самом деле целевая система может даже специально ограничить количество запросов или ввести временную задержку перед выдачей ответов, чтобы защитить себя от такой атаки. Если злоумышленнику настолько повезет, что у него будет доступ к показателям, возвращаемым целевой системой, он может сократить объем запросов за счет использования таких интеллектуальных стратегий, как генетический алгоритм, рассмотренный в разделе «Методы черного ящика с оценкой» на с. 163. Однако, как уже говорилось ранее, в большинстве случаев есть лишь ограниченный доступ к этим оценкам.

Помимо самой ГНС, прямая атака принимает в расчет всю последовательность обработки и используемые методы активной защиты. И хотя злоумышленник может разработать атаку, не обращаясь к целевой системе напрямую, для создания устойчиво вредоносных входных данных все же потребуется некоторое тестирование на самой целевой системе.

Атака с копированием

Один из очевидных подходов к разработке вредоносных входных данных сводится к тому, чтобы использовать точную копию целевой системы для окончательной доработки вредоносного входного сигнала перед его

применением к целевой системе. Здесь возможны два сценария — с доступом к копии всей целевой системы или с доступом только к алгоритму ГНС.

- *Копирование системы.* Злоумышленник может получить локальную копию всей целевой системы и использовать ее для своих экспериментов. Примером такой копии может служить приобретенный на коммерческой основе виртуальный голосовой помощник или, скажем, автономное транспортное средство. Используя локальную копию целевой системы, злоумышленник может разработать собственный вредоносный входной сигнал путем имитации многократных запросов и отслеживания ответов методом черного ящика[1].

  На практике целевые системы, доступные для приобретения на коммерческой основе (не размещенные в Интернете), редко принимают входной сигнал и выдают ответ в цифровом виде. Например, приобретаемый на коммерческой основе виртуальный помощник вряд ли предоставит удобный программный интерфейс для своей внутренней последовательности обработки, с помощью которого злоумышленник мог бы итеративно корректировать вредоносный входной сигнал на основе многократно повторяемых запросов. Вместо этого он будет принимать звуковой входной сигнал и в качестве ответа возвращать звуковой сигнал (речь) либо инициировать некоторую цифровую команду (например, команду на совершение онлайн-покупки). Когда взаимодействие с системой (подача запроса и получение ответа) осуществляется не в цифровой форме, генерация вредоносных входных данных гораздо труднее поддается автоматизации.

  Как и в случае методов черного ящика, доступ к копии всей системы позволяет злоумышленнику проверить, какой эффект произведет атака на всю последовательность обработки, а не только на целевую ГНС.

- *Копирование ГНС.* Когда злоумышленник располагает информацией обо всех аспектах подвергаемой атаке обученной ГНС (то есть об архитектуре и всех параметрах модели), это обеспечивает ему выигрышное положение при генерации вредоносных входных данных. Если сравнить приведенный случай с традиционным программированием, то это примерно то же

[1] Локальная копия может пересылать информацию в централизованную серверную часть для обработки, как, например, часто происходит в случае виртуальных помощников с голосовым управлением.

самое, что знать исходный код внутреннего алгоритма целевой системы. С таким уровнем осведомленности и достаточными возможностями злоумышленник может создать копию ГНС и, применяя к ней любой понравившийся ему метод, создать вредоносные образы, точно использующие слабые места ГНС.

На первый взгляд задача разработки вредоносных входных данных с использованием копии целевой системы кажется элементарной задачей, однако каким образом злоумышленник может получить доступ к такой копии? Ведь вполне очевидно, что организация, заботящаяся о своей безопасности, никогда по доброй воле не сделает общедоступными свои внутренние алгоритмы. Однако у этого правила есть исключения. Поскольку для ГНС обычно требуется множество размеченных данных, значительные вычислительные ресурсы и специалист по обработке данных, способный эффективно ее обучить, организации выгоднее использовать предварительно обученную модель, предоставляемую на коммерческой основе или как ПО с открытым исходным кодом, просто из соображений экономии времени и ресурсов. Если злоумышленник узнает, какая модель используется организацией, и получит доступ к такой же модели (создав копию или воспользовавшись публично доступной копией), то он сможет осуществить атаку с копированием. Даже если злоумышленник не будет располагать инсайдерской информацией об используемой модели, он сможет определить, какие из публично доступных моделей может применять целевая организация, проверяя обоснованные предположения о ее внутренней последовательности обработки.

Атака с переносом

Когда у злоумышленника нет доступа к алгоритму ГНС, используемому защищающейся организацией, он может прибегнуть к достаточно точной аппроксимации целевой ГНС для разработки вредоносного входного сигнала перед его применением к целевой системе. Такой способ нападения называют *атакой с переносом*.

Стратегия атаки с переносом — пожалуй, наиболее осуществимый подход во многих реальных сценариях, когда реализация внутренней ГНС неизвестна за пределами защищающейся организации. Этот подход также имеет преимущества над прямой атакой, так как разработка вредоносного образа осуществляется без обращения к целевой системе. То есть такая атака

не вызывает подозрений и не страдает от ограничений на количество или скорость обработки запросов.

При проведении атаки с переносом злоумышленник создает замещающую модель для разработки вредоносных входных данных, опираясь на некоторые ограниченные знания о целевой ГНС. Затем он разрабатывает вредоносный образ, применяя методы белого ящика, черного ящика с оценкой или ограниченного черного ящика к замещающей модели, и передает этот образ целевой модели. Как и в случае атаки с копированием, злоумышленник может обойтись без создания собственной замещающей модели, используя вместо этого существующую модель (например, размещенную в Интернете модель с открытым API).

В данном случае перед злоумышленником встает следующая проблема: замещающая модель должна вести себя так же, как целевая модель, по крайней мере при представлении ей вредоносного сигнала. Однако насколько осуществимой задачей является создание модели со схожим поведением при отсутствии доступа к *точной* модели?

На первый взгляд создание модели, ведущей себя так же, как ГНС целевой системы, кажется непреодолимой задачей. Подумайте, сколько всего при этом необходимо учесть: нужно решить, сколько слоев содержит ГНС, какие функции активации она использует, какова ее структура и какие результаты она может выдавать. Возможно ли на самом деле создать вредоносный входной сигнал, используя аппроксимирующую замещающую модель, и эффективно перенести его на целевую систему?

Ответ на этот вопрос может вас удивить. Как оказывается, даже при ограниченном доступе к информации об исходной модели иногда можно создать модель, настолько близкую к оригиналу, чтобы можно было разработать вредоносные образы, успешно переносимые между двумя моделями[1]. Чтобы понять, почему это так, вспомните, чем определяется ГНС.

- *Отображение входа на выход*. Включает в себя формат и точность входных данных и допустимых выходных данных. В случае ГНС для классификации данных мы имеем набор категорий и, возможно, некоторую иерархическую структуру этих категорий (например, категория «собака» является подмножеством категории «животное»).

[1] *Tramèr F. et al.* The Space of Transferable Adversarial Examples, 2017. bit.ly/2IVGNfc.

- *Внутренняя архитектура*. Тип сети (например, LSTM или CHC), тип и количество слоев, количество узлов в каждом слое. Включает в себя особенности используемых функций активации — фактически те аспекты модели, которые определяются до проведения обучения.
- *Параметры*. Веса и пороги, определяемые в ходе обучения.

Злоумышленник может сделать обоснованные предположения о некоторых аспектах этих составляющих. Например, об архитектуре ГНС можно сказать, что для классификации изображений обычно используется некоторая разновидность сверточной сети. Можно сделать предположения и в отношении разрешения входных данных и точности возможных выходных предсказаний (даже не зная весь список категорий).

Допустим, что злоумышленник сделал некоторые информированные предположения об архитектуре сети и отображения входа на выход. Рассмотрим случай, когда у него есть полный или частичный доступ к обучающим данным. Это вполне возможная ситуация, поскольку в силу размера затрат на генерацию и разметку тренировочных данных для обучения часто используются наборы данных, свободно предлагаемые в Интернете.

Когда злоумышленник знает, какие тренировочные данные использовались для обучения целевой модели, он может создать достаточно точную ее аппроксимацию, даже если есть очень мало заключений о ее архитектуре. То есть две модели с разной архитектурой, скорее всего, будут уязвимы к одинаковым вредоносным образам, если они были обучены с помощью одинаковых данных. Эта идея проиллюстрирована на рис. 7.2. Две совершенно разные модели, обученные с помощью одинаковых данных, как правило, формируют сходные ландшафты предсказаний. Что ж, в этом нет ничего удивительного, поскольку обучающие данные сами по себе играют важную роль в определении параметров модели.

Области входного пространства, в которых модель возвращает неверный результат, называются *вредоносными подпространствами*. Это те области, в которых не удалось обеспечить надлежащее обобщение на этапе обучения. Соответственно, они должны располагаться примерно одинаково у моделей, обученных с помощью одинаковых обучающих данных. Это, в свою очередь, делает возможным, хотя и не гарантирует, успешный перенос вредоносного образа на целевую модель при использовании для замещающей модели таких же обучающих данных.

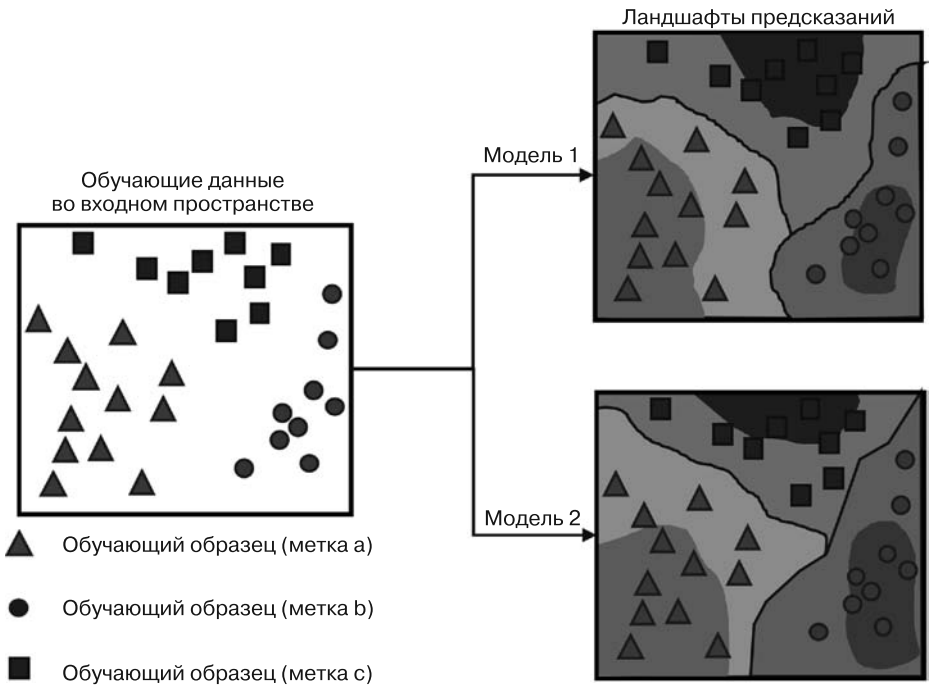


Рис. 7.2. Входные пространства целевой и замещающей моделей с указанным положением обучающих данных

Чтобы проиллюстрировать, насколько схожими могут быть вредоносные подпространства разных моделей, на рис. 7.3 показано положение этих областей на ландшафтах предсказаний тех же двух моделей, что были показаны на рис. 7.2. В силу того что эти модели обучены с помощью одних и тех же обучающих данных, их вредоносные подпространства выглядят примерно одинаково, что делает возможным успешный перенос вредоносных образов с одной модели на другую.

Как вы увидите в главе 10, нейросетевую модель можно аппроксимировать, опираясь на знание того, с помощью каких тренировочных данных она была обучена, что имеет важные последствия с точки зрения информационной безопасности. В силу этого обучающие наборы данных следует считать не подлежащей разглашению информацией, поскольку они оказывают косвенное влияние на поведение тех машинно-обучаемых моделей, для обучения которых они применяются.

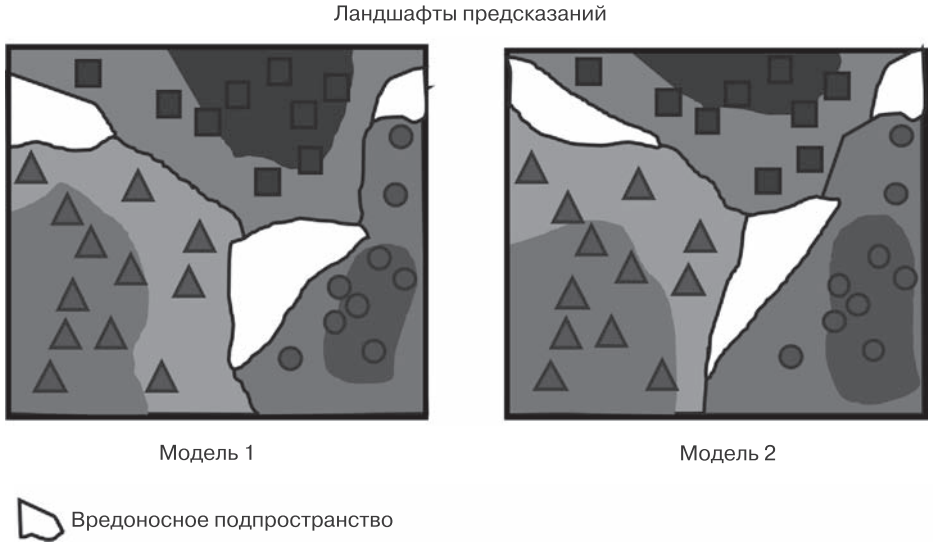


Рис. 7.3. Входные пространства целевой и замещающей моделей; вредоносные подпространства выделены белым цветом

Универсальная атака с переносом

Универсальная атака с переносом используется в том случае, когда злоумышленник не располагает информацией о целевой ГНС и ее обучающих данных. В таком случае вредоносные входные данные создаются с помощью некоторого набора замещающих моделей. Вредоносный образ, срабатывающий против ряда разных замещающих моделей, должен быть достаточно гибким для его успешного переноса на целевую модель.

Что интересно, разные обучающие наборы данных могут занимать примерно одинаковые области входного пространства даже при определенных различиях между ними, если они были созданы на основе одной и той же информации о физическом мире. При этом области входного пространства с недостаточным количеством репрезентативных данных (соответствующие данным вне распределения[1]), как правило, выглядят неизменно и *не зависят* от используемого обучающего набора данных. Кроме того, разные обучающие наборы данных могут иметь между собой определенное сходство, например

[1] О том, что такое данные вне распределения, говорилось в подразделе «Обобщение обучающих данных» на с. 110.

одинаковый наклон камеры или одинаковые характеристики голоса. То есть разные обучающие наборы данных могут иметь примерно одинаковое распределение характеристик, признаков и, что самое главное, вредоносных подпространств.

Единообразие вредоносных подпространств делает возможным проведение универсальной атаки с переносом. Хотя (что неудивительно) осуществить такую атаку труднее, она дает злоумышленнику большие возможности. Когда входной сигнал эффективен против различных моделей, злоумышленник может провести атаку против целой группы глубоких нейросетей (например, против ряда поисковых систем). Опять же на практике универсальная атака с переносом часто комбинируется с прямой атакой (пример см. во врезке далее).

Гипотетический пример: обход классификации видеоданных

Веб-компания, осуществляющая управление видеоконтентом, хочет улучшить классификацию своих видеозаписей, чтобы обеспечить их индивидуализированную фильтрацию для каждого пользователя. Компания разрабатывает автоматизированный алгоритм на базе ГНС, способный классифицировать видеозаписи (например, такие как видео со сценами насилия или употребления наркотиков) на основе одновременно звуковой и визуальной информации.

При этом вредоносная организация заинтересована в том, чтобы произвести над загружаемым на сайт видеоконтентом манипуляции таким образом, чтобы он стал доступен более широкой аудитории. Эта организация создает замещающую модель и на ее основе разрабатывает вредоносные входные данные, используя методы белого ящика. Замещающая модель представляет собой аппроксимацию, основанную на (правильном) предположении о том, что целевая веб-компания использует нейросетевую архитектуру, обученную с помощью общедоступных данных.

Затем, используя схему прямой атаки в сочетании с атакой с переносом, вредоносная организация тестирует и корректирует сгенерированные вредоносные образы, чтобы обеспечить их успешный перенос на целевую модель и сделать их устойчивыми к влиянию всей последовательности обработки.

Одной из самых интересных областей применения универсальной атаки является случай, когда злоумышленник не получает обратной связи от целевой системы по той причине, что целевая организация производит об-

работку данных для собственных нужд (например, с целью анализа рынка). В случае такой «атаки черной дыры» злоумышленник не может узнать, достиг ли он успеха, и должен обеспечить себе большую степень устойчивости к неудачному исходу атаки.

Многократно используемые заплатки и искажения

Представьте, что злоумышленник может создавать многократно используемые вредоносные изменения, применимые для множества разных входных сигналов. Например, вредоносные искажения или заплатки, способные успешно сработать на *любом* изображении, или вредоносные искажения, применимые к *любым* аудиофайлам. Такой подход с однократным созданием и многократным использованием, безусловно, открывает перед злоумышленником потрясающие возможности — ему уже не нужно создавать свежий вредоносный контент для каждого изображения или аудиофайла; достаточно просто наложить заранее приготовленную заплатку или искажение без затрат времени и денег, а также лишних обращений к целевой системе.

В своей работе «Вредоносная заплатка»[1] исследователи Браун и др. продемонстрировали возможность генерации заплаток, пригодных для многократного использования на разных изображениях. Эффективность таких заплаток зависит от их размера и положения на целевом изображении, но в то же время они действительно срабатывают на множестве разных изображений. Это вполне логично: если вам удастся создать заплатку с очень высоким уровнем значимости, способную отвлечь на себя внимание нейросети, то она, по идее, будет работать на большом количестве изображений.

Эксперименты с многократно используемыми заплатками

Если вы хотите опробовать многократно используемые заплатки на своих изображениях, см. блокнот Jupyter: chapter07/reusable_patch.ipynb (bit.ly/2WSCgnY) — в GitHub-репозитории этой книги.

Интуиция подсказывает, что многократное использование вредоносных искажений вряд ли возможно, поскольку каждое такое изменение создается для достижения определенной вредоносной цели на основе характеристик конкретного изображения или аудиоролика. То есть вредоносное искажение

[1] *Brown et al.* Adversarial Patch.

представляет собой смещение из некоторой точки во входном пространстве, которое вряд ли сработает при другом исходном положении. Эту идею иллюстрирует рис. 7.4. Здесь в уменьшенном виде показано входное пространство двух изображений, которые относятся к одной и той же категории, но занимают разное положение в соответствующей области. После того как вредоносное искажение, рассчитанное таким образом, чтобы изображение 1 сместилось в область ошибочной категории, было применено к изображению 2, это изображение по-прежнему остается в области правильной категории.

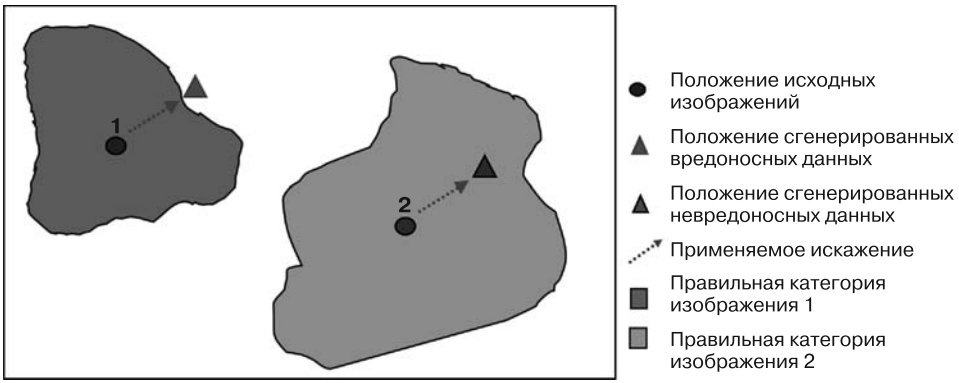


Рис. 7.4. Перенос вредоносного искажения на другое изображение

Каким бы маловероятным это ни казалось, Мусави-Дезфули и др. в своей статье[1] показали возможность создания универсальных искажений. Этого можно добиться путем оптимизации вредоносного искажения с помощью формулы, принимающей в расчет некоторую выборку изображений из распределения, а не одно изображение в отдельности.

Суть данного метода сводится к следующему. Вредоносное искажение рассчитывается для исходного изображения так, чтобы это изображение сместилось за пределы области его правильной категории. Затем это же искажение применяется ко второму изображению в выборке. Если второе изображение правильно классифицируется (то есть остается невредоносным) и после применения искажения — как показано на рис. 7.4, — значит, искажение не срабатывает на обоих изображениях. В таком случае в до-

[1] *Moosavi-Dezfooli S.-M. et al.* Universal Adversarial Perturbations // IEEE Conference on Computer Vision and Pattern Recognition. 2017, bit.ly/2WV6JS7.

полнение к исходному искажению рассчитывается еще одно изменение, достаточное для того, чтобы обеспечить ошибочную классификацию второго изображения. При условии, что второе изменение не будет возвращать первое изображение в область его правильной классификации, итоговое искажение будет успешно работать на обоих изображениях. Второе изменение показано на рис. 7.5.

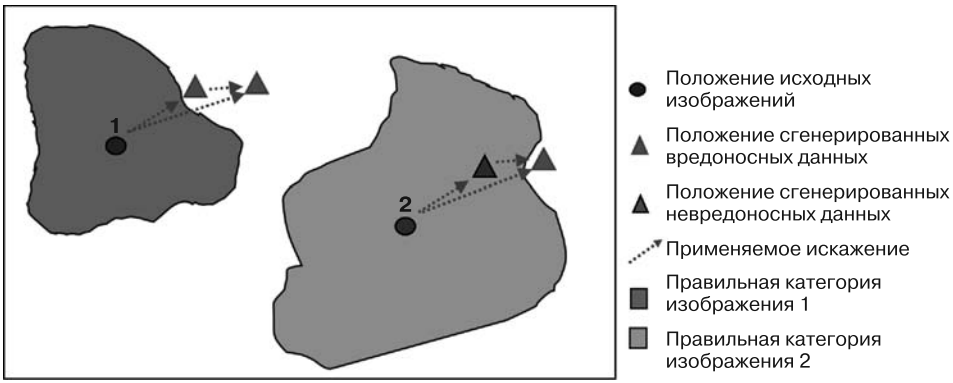


Рис. 7.5. Расчет универсального вредоносного искажения

Эти действия повторяются для некоторого распределения реалистичных изображений таким образом, чтобы итоговое искажение оставалось в пределах минимально допустимого изменения (количественно оцениваемого с помощью L^p-нормы) и была обеспечена заданная пропорция неправильно классифицируемых изображений (или *коэффициент обмана*). Для повышения коэффициента обмана, естественно, требуется увеличивать размер искажения, что делает его более заметным, но в то же время обеспечивает впечатляющие результаты. Хотя при этом нельзя дать гарантии, что искажение сработает на всех изображениях, исследователями доказана возможность создания незаметных искажений, срабатывающих на 80–90 % изображений. Доказано также, что подобные искажения хорошо переносятся между разными моделями. Универсальные искажения являются нецелевыми; они смещают любое изображение из области правильной категории в область другой, не заданной точно категории, в результате чего итоговая вредоносная категория может быть разной у различных изображений.

Такие методы, по идее, должны срабатывать и для аудиоданных. Подобно тому как положение и размер вредоносной заплатки на изображении могут

влиять на степень ее заметности и эффективности, для оптимального срабатывания звуковой заплатки ее обычно нужно разместить в определенном месте аудиофайла и отрегулировать по громкости. Помимо преимуществ многократного использования, универсальное вредоносное искажение аудиоданных теоретически можно воспроизводить в любом окружении, чтобы, к примеру, обеспечить злоумышленное управление звуковыми устройствами с помощью голосовых команд, скрытых в звуковой волне. Воспроизведение вредоносных аудиоданных в физическом мире будет рассмотрено в разделе «Вредоносный звук» на с. 195.

Вредоносные изменения, создаваемые с расчетом на множество разных входных сигналов, как правило, не гарантируют успеха. Однако, поскольку их создание не требует больших затрат, они хорошо подходят для тех случаев, когда надежность вредоносной атаки не играет большой роли (см. гипотетический пример во врезке далее). Пожалуй, самой интересной особенностью данного подхода является то, что теоретически он допускает обмен вредоносными возможностями между разными злоумышленниками. Так, например, группа лиц или отдельное лицо может продавать вредоносные заплатки через Интернет вместе с простым программным обеспечением для их наложения на изображения.

Гипотетический пример: управление брендом в пивоварении

Пивоваренная компания создала новый сорт пива и для улучшения его позиций на рынке хочет сделать так, чтобы он чаще ассоциировался с более популярным брендом.

В рамках своих (вредоносных) действий по управлению брендом компания разрабатывает в цифровом виде вредоносную заплатку, заставляющую некоторые поисковые системы уверенно классифицировать изображения данного сорта пива как изображения более популярного сорта пива. При этом (с точки зрения человека) степень сходства заплатки не является настолько большой, чтобы можно было говорить о нарушении авторского права.

Используя эту заплатку на своих банках и в цифровом рекламном контенте в Интернете, компания надеется заставить поисковые системы выдавать изображения нового сорта пива в ответ на поисковые запросы, относящиеся к бренду с более прочными позициями на рынке.

Пивоваренная компания разрабатывает эту вредоносную заплатку с помощью методов черного ящика с оценкой, использующих API, открытые для

свободного использования в исследовательских целях. В силу временных задержек и ограничений на количество запросов, свойственных поисковым системам, на это уходит несколько недель, однако у компании есть запас времени и терпения.

Хотя такую атаку можно выявить с помощью упомянутых общедоступных API, поскольку ее цель неизвестна и она не проводится против использующей эти API организации, нет нужды обеспечивать какую-либо скрытность. Созданная заплатка встраивается в используемые в дальнейшем цифровые рекламные материалы с надеждой на то, что эти изображения широко распространятся в Интернете и в итоге дойдут до поисковых систем. При этом даже не нужно, чтобы заплатка срабатывала во всех 100 % случаев; цель может считаться достигнутой, даже если она будет срабатывать лишь эпизодически.

Сводим все вместе: комбинированные методы и компромиссы

На практике реальная атака, как правило, представляет собой сочетание нескольких подходов. Например, злоумышленник может сначала разработать вредоносные образы с помощью замещающей модели, а затем выборочно протестировать их на целевой системе, тем самым взяв лучшее из обоих подходов.

Как и в случае любой другой кибератаки, здесь очень важную роль играет этап подготовки. Создание эффективного вредоносного сигнала перед проведением атаки обычно требует проведения некоторых экспериментов над целевой системой, чтобы определить, что работает, а что — нет. В силу этого часто приходится делать выбор между скрытностью и надежностью — создание более надежного вредоносного образа в большинстве случаев требует большего числа обращений к целевой системе, что, в свою очередь, повышает риск обнаружения атаки.

Иногда вредоносные образы хорошо себя показывают в теоретическом окружении, но оказываются уязвимыми к обработке, выполняемой в рамках целевой системы в целом. Например, как вы увидите в главе 10, механизм действия вредоносных образов часто основан на степени точности и разрешении цифровых данных, в силу чего на их эффективности может плохо сказываться предварительная обработка, снижающая разрешение данных.

Поэтому, за исключением того случая, когда у злоумышленника есть точная копия всей целевой системы (включающая в себя всю последовательность обработки, а не только ГНС), ему, как правило, требуется провести хотя бы минимальные эксперименты над целевой системой.

Обеспечение скрытности на этапе создания вредоносных образов сводится к минимизации количества запросов к целевой системе. При атаке с копированием, атаке с переносом и универсальной атаке с переносом для разработки атаки перед ее запуском используется замещающая модель. Это избавляет злоумышленника от необходимости подавать входные сигналы в целевую систему до момента проведения реальной атаки, что существенно повышает его шансы остаться незамеченным.

Еще одно разочарование для злоумышленника — защищающаяся система зачастую не представляет собой просто реализацию некоторого алгоритма, пассивно обрабатывающую все входные сигналы, а включает в себя и определенные механизмы активной защиты. О них речь пойдет в главе 10.

Любые сведения о целевой системе облегчают задачу злоумышленника. Как будет видно в главе 10, к не подлежащей разглашению информации организация должна относить и сами нейросетевые модели, и любые данные, позволяющие их воспроизвести, например обучающие данные.

Не существует какого-либо стандартного способа проведения атаки. В этой главе мы рассмотрели несколько способов генерации вредоносных образов, но на практике наиболее эффективные результаты показывает сочетание различных подходов, обеспечивающее необходимый баланс между скрытностью и надежностью в зависимости от целевой системы и вредоносных целей.

ГЛАВА 8

Атаки в физическом мире

В предыдущих главах речь в основном шла о том, как можно создать вредоносные входные данные путем внесения искажения в цифровое изображение или цифровые аудиоданные. Однако зачастую злоумышленник не имеет доступа к данным в цифровом формате и может воздействовать только на физическое окружение, на основе которого генерируются данные. В таком случае целевая вычислительная система уже не принимает входные данные из внешнего мира в виде цифрового контента (как в случае загрузки файлов на сайт социальной сети), а получает их непосредственно от датчика (например, от камеры видеонаблюдения). Как результат, атака при таком «физическом» сценарии может проводиться совершенно не так, как при рассмотренных ранее цифровых сценариях.

Создание вредоносных образов в физическом мире создает для злоумышленника ряд новых проблем. Теперь ему нужно создать или изменить нечто реально существующее так, чтобы оно включало в себя физическое воплощение вредоносного искажения или заплатки. Когда вредоносные данные поступают от камеры, изменяемым объектом может быть двумерный отпечаток или трехмерный объект. Подобным образом вредоносное искажение может поступать через микрофон при воспроизведении в его окружении специально подобранных аудиосэмплов с помощью такого цифрового устройства, как компьютер или телевизор. Однако, к примеру, как можно обеспечить надежное срабатывание вредоносного объекта вне зависимости от условий освещенности или положения камеры? И можно ли вообще обмануть виртуальный помощник с помощью скрытых голосовых команд, не имея гарантии, что вредоносный контент будет воспроизводиться достаточно близко от него, и при наличии в окружении различных посторонних шумов?

Доступ к цифровым данным очевидно позволяет осуществлять злоумышленнику более точный контроль над создаваемым вредоносным сигналом.

Например, он может изменять цифровые изображения с очень высокой степенью детализации. При атаке в физическом мире обычно требуется более грубый подход, поскольку при изготовлении вредоносного объекта или воспроизведении вредоносного звука может теряться некоторая степень детализации. При этом чем точнее датчик (камера или микрофон) захватывает требуемое искажение, тем легче злоумышленнику осуществить надежную атаку. Для неограниченного физического окружения также характерно наличие шумов и непостоянство условий, и любой физический вредоносный образ должен иметь дополнительную степень устойчивости, чтобы противостоять этой нестабильности. Наконец, злоумышленник явно сильно ограничен в том, какие изменения он может вносить в физическое окружение из-за ограниченного доступа и риска обнаружения, зависящего от того, кем и при каких обстоятельствах воспринимаются изменения.

Чтобы выяснить, какие угрозы может представлять атака в физическом мире, сначала разберемся, каким образом злоумышленник может создавать объекты, порождающие вредоносные результаты при их съемке с помощью камеры. Затем мы поговорим о том, насколько возможным является использование звуков — и, в частности, речевых команд — для вредоносных целей. В обеих этих сферах перед злоумышленником встает ряд проблем:

- *создание вредоносных входных данных*. Изготовление вредоносного объекта или воспроизведение вредоносного звука на основе рассчитанного в цифровом виде искажения;
- *захват вредоносных входных данных в цифровом формате*. Захват внешнего вида вредоносного объекта камерой или вредоносного звука микрофоном и преобразование этих данных в цифровой формат;
- *влияние положения и удаленности вредоносных входных данных от датчика*. Успех достижения вредоносной цели зависит от положения камеры или микрофона;
- *окружающие условия*. Нестабильность окружающей среды, например изменение освещения, погодных условий или акустики в помещении;
- *ограничения атаки*. Какие ограничения накладывает на создание физических вредоносных входных данных используемый метод атаки, то есть какие изменения может внести злоумышленник, не выдавая своего присутствия, и какие методы маскировки он может использовать для сокрытия вредоносной атаки.

Вредоносные объекты

Существует достаточно много мотивов для создания вредоносных объектов в физическом мире. Например, все более широкое использование автономных систем (таких как автономные транспортные средства), которые используют захватываемые камерами визуальные данные, предоставляет потенциальные возможности для создания вредоносных объектов с целью дезориентировать такие системы. В системах видеонаблюдения все чаще возникает необходимость в автоматизированной нейросетевой обработке данных, получаемых от камер видеонаблюдения, например, для отслеживания определенных событий. Это влечет за собой риск атаки на такие системы, если не исключена возможность создания физических вредоносных объектов.

В данном разделе речь пойдет о том, насколько вообще возможно создать физические вредоносные объекты и какие проблемы при этом нужно решить. И начнем с основной проблемы изготовления вредоносных объектов и захвата вредоносных признаков в цифровом виде с помощью камеры (в подразделе ниже). Затем рассмотрим осложняющие факторы окружающей среды и углы обзора (в подразделе «Углы обзора и окружение» далее).

Изготовление объекта и возможности камеры

Начнем с главного вопроса: может ли камера в достаточной мере отразить вредоносные аспекты напечатанного объекта в его цифровом воспроизведении? Чтобы ответить на него, следует сначала напечатать изображение с вредоносным искажением и посмотреть, насколько высока вероятность того, что камера извлечет это искажение и сгенерирует ошибочно интерпретируемое цифровое представление. Этот базовый эксперимент показан на рис. 8.1. Обратите внимание, что нас пока не волнуют такие дополнительные детали, как угол обзора камеры или условия освещенности — нам важно лишь узнать, можно ли успешно передать вредоносную информацию путем ее печати с последующим захватом изображения камерой.

При всей своей простоте данный эксперимент является важным первым шагом, который, как оказалось, возможен[1]. Так что, да, сгенерированные в цифровом виде вредоносные образы могут по-прежнему оставаться вредоносными после

[1] *Brown et al.* Adversarial Patch; *Kurakin A., Goodfellow I.J., Bengio S.* Adversarial Examples in the Physical World // International Conference on Learning Representations, 2017. http://bit.ly/2x0S0pq.

дополнительного шага печати/захвата камерой. Однако при этом следует учесть ряд факторов, включающих следующие.

- *Изготовление объекта (2D- или 3D-печать).* Злоумышленник должен располагать инструментами для печати искажения или заплатки с необходимым уровнем детализации. С этим могут возникнуть затруднения, поскольку воспроизводимый принтером диапазон цветов *(цветовой охват)* представляет собой лишь некоторое подмножество тех цветов, которые позволяет представить цифровое RGB-изображение. Если вредоносное искажение или заплатка будут представлены с помощью недоступных для принтера RGB-значений, то атака не увенчается успехом. Используемый подобным образом принтер должен обеспечивать необходимый для вредоносной атаки уровень детализации.

  Для успешного срабатывания вредоносного искажения также должно быть обеспечено *надежное* воспроизведение. Это может быть очень сложным в силу разношерстности и недостаточной точности принтеров в плане воспроизведения цветов. Один из подходов к решению этой проблемы сводится к тому, чтобы отобразить возможные RGB-значения пикселей на те цвета, которые они фактически обеспечивают при печати. Это позволяет количественно оценить среднюю погрешность печати для изображения путем вычисления разности между правильными и фактическими значениями. После этого требование по минимизации погрешности печати включается в качестве дополнительного ограничения во вредоносную функцию штрафа, в результате чего создаваемое вредоносное искажение оптимизируется таким образом, чтобы в нем использовались те цвета, которые наиболее точно воспроизводятся принтером[1]. Как показала практика, данный метод не отличается надежностью, поскольку погрешность печати обычно сильно варьируется и может различаться не только у разных принтеров, но даже у разных отпечатков одного устройства. Более эффективный подход сводится к созданию вредоносного образа так, чтобы он меньше зависел от каких-либо конкретных цветов, что обеспечит его устойчивость к погрешностям цветопередачи. Этот метод будет подробно рассмотрен в следующем разделе[2], где мы также обсудим такую проблему, как влияние освещенности.

---

[1] Этот метод продемонстрирован в статье Шарифа и др. «Аксессуары для преступления» (*Sharif et al.* Accessorize to a Crime).

[2] В контексте следующей статьи: *Athalye A., Engstrom L., Ilyas A., Kwok K.* Synthesizing Robust Adversarial Examples // International Conference on Machine Learning, 2017. http:// bit.ly/2FktLXQ.

- *Возможности камеры.* Обеспечиваемая при захвате вредоносной информации степень точности всегда ограничена доступным для камеры пределом чувствительности. Например, искажение, закодированное в напечатанном изображении с помощью ряда отдельных пикселей, не будет захвачено камерой, если она не способна захватывать данные с таким уровнем детализации при имеющейся удаленности от объекта.

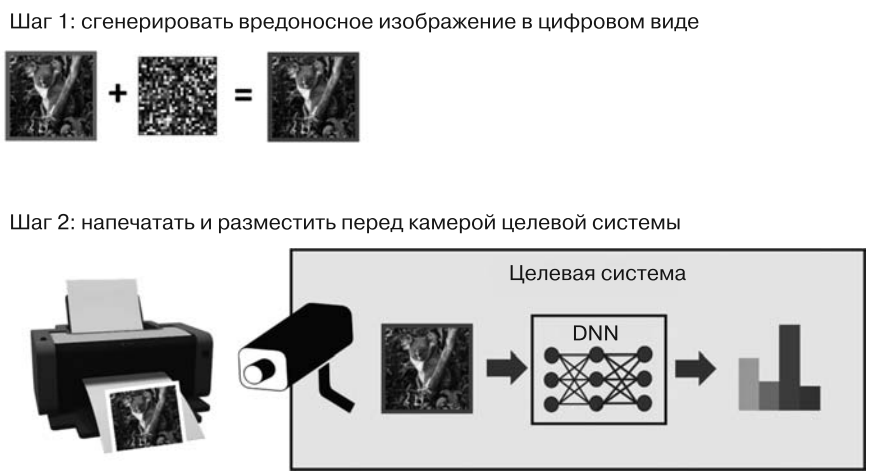


Рис. 8.1. Сильно ограниченная вредоносная атака в физическом мире с использованием напечатанного изображения

Поскольку и в случае физической, и в случае цифровой атаки вопросы точности данных и компенсации присутствующих в изображении шумов и помех следует рассматривать в контексте последовательности предварительной обработки, эти проблемы будут подробно рассмотрены в подразделе «Предварительная обработка в общей последовательности обработки» на с. 249.

Углы обзора и окружение

Теперь перейдем на новый уровень и рассмотрим менее ограниченное и более реалистичное окружение, примером которого может служить показанный на рис. 8.2 случай внесения вредоносного искажения в дорожный знак с целью обеспечить его ошибочную классификацию. На этот раз злоумышленник решает пойти дальше — изменяет или создает объект в реальном трехмерном мире, не дающем никаких гарантий в отношении углов обзора камеры и окружающих условий.

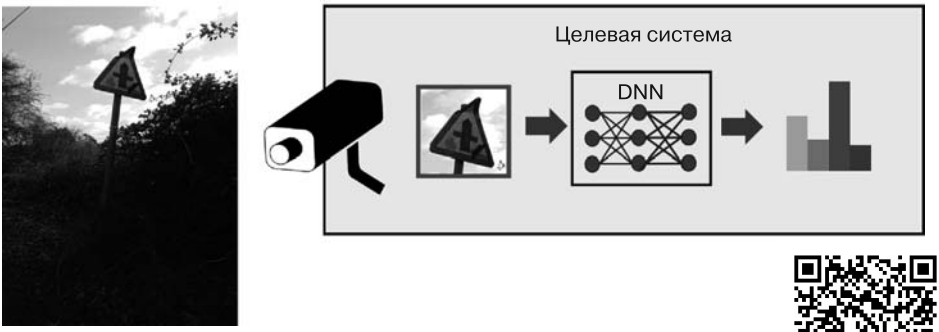


Рис. 8.2. Неограниченная вредоносная атака в физическом мире

Первой очевидной проблемой использования вредоносных объектов в физическом мире является подвижность их самих или снимающей их камеры. Например, на рис. 8.3 показана последовательность изображений, снятых с небольшим интервалом под разными углами. Обратите внимание, насколько сильно они различаются по углу обзора, освещенности и экспозиции. Если мы внесем в знак вредоносное искажение с помощью тех способов, которые описаны выше, то оно не будет переноситься между этими изображениями. Изначально даже считалось, что проблема переносимости делает невозможным создание надежных физических вредоносных образов[1].

Рис. 8.3. Влияние углов обзора камеры и освещенности на вид физических объектов

Здесь можно заметить сходство с проблемой, встающей перед злоумышленником при создании не зависящих от изображения многократно используемых вредоносных искажений, о которых говорилось в разделе «Многократно используемые

[1] *Lu J. et al.* No Need to Worry about Adversarial Examples in Object Detection in Autonomous Vehicles // Conference on Computer Vision and Pattern Recognition, 2017. http://bit.ly/2XracqU.

заплатки и искажения» на с. 179. Вредоносное изменение должно быть достаточно гибким для того, чтобы срабатывать на разных изображениях. В целом, здесь нужно учесть следующие моменты.

- *Углы обзора и удаленность*. Прежде всего нужно принять во внимание положение камеры и ее настройки и соответствующее множество возможных перемен в изображении. Необходимо учесть следующее:
  - *увеличение*. Изображение может увеличиваться и уменьшаться (за счет изменения расстояния между камерой и целевым объектом или за счет действия функции увеличения камеры). Как результат, объект может занимать большую или меньшую часть общего поля зрения камеры;
  - *параллельный перенос и вращение*. Объект может двигаться относительно камеры, либо камера может двигаться относительно объекта. Это может привести к смещению объекта в пределах объектива камеры и изменению угла, под которым ведется съемка объекта;
  - *наклон (или сдвиг)*. Положение камеры по отношению к объекту и ее фокусное расстояние могут привести к искажению изображения.
- *Освещенность и окружающие условия*. Чтобы алгоритм мог надежно работать вне зависимости от точки съемки, он также должен учитывать положение объекта по отношению к имеющимся источникам света и другим объектам, а также окружающие условия.

  Итоговый вид изображения зависит от угла падения света, насыщенности цветов объекта и текстуры его поверхности. Даже разница в освещенности при солнечной и пасмурной погоде может сильно сказаться на яркости и цветовой насыщенности конечного изображения.

  Различные объекты рассеивают свет по-разному. То, как свет отражается от поверхности объекта, зависит от того, из какого материала изготовлена эта поверхность, от текстуры, цвета, рассеивающей и отражательной способности. Если поверхность с низкой отражательной способностью, лучи света рассеиваются и отражаются в разных направлениях, что обеспечивает матовость изображения. Однако если поверхность с высокой отражательной способностью (например, металлическая), лучи света отражаются от объекта под углом, равным углу их падения. Такое характерное для зеркал поведение называют *зеркальным отражением*.

  Освещенность объекта также может зависеть от положения других объектов и от окружающих условий. Так, вредоносный объект может частично

спрятаться за другими объектами или изменить свои отражательные свойства (например, из-за дождливой погоды).

При захвате фотографических данных также есть риск появления шумов. При съемке фотографий в условиях низкой освещенности на них могут появиться нежелательные шумы — случайные вариации цвета или яркости, — проявляющиеся в виде бликов на изображении. И хотя эти шумы, внесенные на этапе сбора данных, могут быть удалены программным путем в датчике или на этапах обработки данных, предшествующих этапу нейросетевой обработки, это вряд ли поможет злоумышленнику, поскольку при такой очистке от шумов не будет повторно внесено то исходное, едва различимое искажение, на которое наложились шумы.

- *Ограничения атаки*. Наконец, возможности злоумышленника по размещению вредоносного искажения или заплатки могут зависеть от того, какие изменения он может произвести в физическом окружении. В зависимости от сценария вредоносное изменение может либо вноситься в уже существующий конкретный объект, либо изготавливаться в виде нового объекта, который размещается в окружении. Хотя часто не требуется делать искажение абсолютно незаметным, вопрос о том, до какого уровня можно внести то или иное изменение с сохранением его незаметности, играет для злоумышленника не последнюю роль. В некоторых случаях злоумышленник может замаскировать атаку, например, под логотип. При этом могут существовать ограничения как на сами изменения областей, цветов или текстур, так и на физическую осуществимость этих изменений.

Столь большое количество сложностей подталкивает нас к мысли о том, что создать вредоносные объекты в физическом мире невозможно. Однако в 2018 году группа исследователей доказала возможность создания вредоносных образов, которые надежно срабатывают в пределах реалистичного распределения условий обзора и освещенности[1]. Им удалось добиться этого, сочетая сразу несколько методов. Вместо двумерных изображений были использованы сетчатые трехмерные изображения, представляющие собой каркасные модели трехмерных объектов, позволяющие выполнять цифровые манипуляции и вращение в трехмерном пространстве. К этим моделям также привязываются соответствующие цвета и текстуры, что позволяет моделировать в цифровом виде влияние на объект различных условий освещенности.

---

[1] *Athalye, Engstrom, Ilyas, Kwok*. Synthesizing Robust Adversarial Examples.

Вооружившись таким более информативным представлением объектов физического мира, исследователи создали функции, позволяющие сымитировать их поведение при таких преобразованиях, как перевод из трехмерного вида в двумерный, изменение освещенности, вращение и параллельный перенос. Далее влияние этих преобразований было отражено в функции штрафа, используемой для расчета вредоносного искажения. Затем вместо того, чтобы рассчитывать градиент штрафа для целевого предсказания относительно изменений в одном двумерном изображении, исследователи рассмотрели изменения в пределах распределения преобразований трехмерной текстурированной модели, используя методы белого ящика. Этот метод назван авторами *«ожидание преобразования»* (Expectation over Transformation, EOT).

На рис. 8.4 показаны изображения из статьи Атали и др., демонстрирующие фотографии и результаты классификации вредоносных черепах, изготовленные методом 3D-печати.

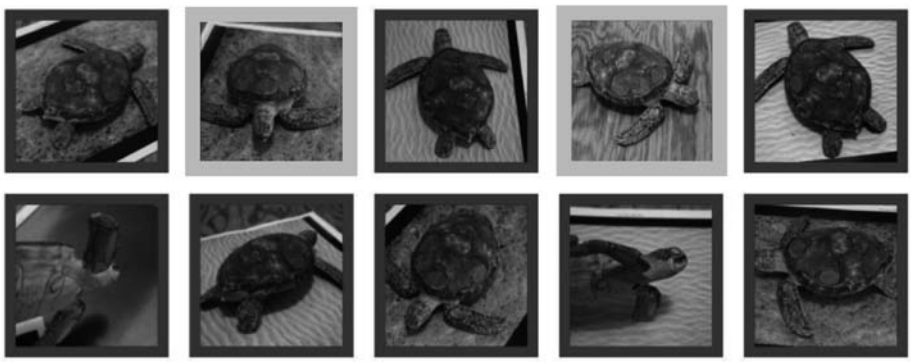


Рис. 8.4. Ошибочное отнесение к категории «винтовка» черепах, изготовленных методом 3D-печати

Стоит отметить, что данное исследование имеет большое значение и для цифровых вредоносных образов. Создавая устойчивые к преобразованиям вредоносные искажения с помощью 3D-моделирования и затем переводя их в двумерный вид, можно успешно генерировать вредоносные образы, обладающие большей устойчивостью к предварительной обработке и методам защиты (подробнее об этом — в главе 10).

Как упоминалось в главе 1, Эйкхолт и др.[1] применили подобный подход для встраивания преобразований в двумерное изображение с целью заставить ГНС неправильно интерпретировать его содержимое (например, заставить ее интерпретировать знак остановки как знак ограничения скорости с помощью нескольких вредоносных наклеек, как показано на рис. 8.5). Исследователи не остановились на этом, они сняли внесенные в дорожный знак физические изменения на камеру и включили эти данные в расчеты вредоносных искажений.

Рис. 8.5. Физическое искажение знака остановки (изображение из статьи Эйкхолта и др., 2018 год)

Приведенные здесь примеры (с черепахами, созданными методом 3D-печати, и наклейками на знаке остановки) демонстрируют разнообразие тех ограничений, с которыми приходится иметь дело злоумышленнику при внесении в окружение вредоносного искажения или заплатки. В первом случае злоумышленник вносит в окружение цельный объект, имея над ним полный контроль. При этом он располагает большой свободой действий в плане конкретного расположения вредоносного изменения в пределах объекта и формы самого объекта. Злоумышленник очевидным образом стремится к тому, чтобы созданный им объект не выглядел подозрительно в том окружении, где он будет размещен (см. гипотетический пример во врезке далее).

В примере со знаком остановки изменения требовалось вносить в пределах существующего дорожного знака. Чтобы сгенерировать безобидные с виду изменения, исследователи ограничили расчет искажений так, чтобы они занимали лишь некоторые области изображения и их можно было легко принять за грязь или граффити.

---

[1] *Eykholt K. et al.* Robust Physical-World Attacks on Deep Learning Visual Classification.

Гипотетический пример: ложные срабатывания системы видеонаблюдения

В месте проведения массовых мероприятий собирается огромное количество видеоданных, получаемых от множества камер видеонаблюдения, и сотрудники службы безопасности просто не в состоянии постоянно их отслеживать. Для быстрого реагирования на возможные угрозы эти видеоданные подвергаются обработке в режиме реального времени, что позволяет выявить наличие огнестрельного оружия и выдать сигнал тревоги. Эти сигналы передаются сотрудникам службы безопасности — они игнорируют случаи ложного срабатывания и быстро реагируют на реальные угрозы.

В попытке вызвать замешательство на месте проведения определенного мероприятия некоторая организация создает комплекты с, казалось бы, безобидными логотипами, содержащими вредоносное искажение, способное привести к ошибочному выявлению огнестрельного оружия. При этом обеспечивается высокая устойчивость вредоносного искажения к различным преобразованиям и условиям освещенности, а также его переносимость между различными моделями обработки изображений.

Эти комплекты распространяются среди уличных торговцев до даты проведения определенного мероприятия в конкретном месте. Затем их открыто продают участникам, которые без какого-либо злого умысла демонстрируют их в месте проведения мероприятия.

В результате возрастает количество ложных срабатываний системы видеонаблюдения, что заставляет службу безопасности принимать меры предосторожности и гораздо чаще проверять содержимое сумок посетителей, что ведет к образованию длинных очередей и нарушению запланированного порядка проведения мероприятия. На место проведения мероприятия даже прибывает дополнительный отряд полиции, до того как удастся выявить источник ложной тревоги.

Вредоносный звук

В силу повсеместного использования голосовых интерфейсов для управления смартфонами, планшетами, носимой электроникой, виртуальными помощниками и другими устройствами наиболее актуальной областью применения вредоносного звука являются системы распознавания речи. Многие устройства, в которых применяется эта технология, не прекращают процесс прослушивания даже во время пассивного ожидания, предоставляя тем самым

постоянно открытый вектор атаки. Угрозу для таких устройств могут нести не только звуки, воспроизводимые в общественных местах, но и телевидение, музыка и потоковый цифровой контент, загружаемый нами дома и на рабочем месте. Злоумышленник может встроить неслышимые голосовые команды во вполне безобидные на первый взгляд звуки, но насколько легко будет провести такую вредоносную атаку в физическом окружении, например против установленного на смартфоне виртуального помощника?

Как и в предыдущем разделе, начнем с простейшего случая воспроизведения вредоносного аудиосигнала в физическом мире и захвата его в цифровом виде микрофоном. После этого рассмотрим осложняющие факторы окружающей среды и относительное положение динамика и микрофона (в подразделе «Положение аудиосигнала и окружение» далее).

Возможности микрофона и системы воспроизведения

Как и в случае изображений, начнем с самого простого сценария: микрофон находится в непосредственной близости от звука, издаваемого одним динамиком (рис. 8.6). Предположим, что мы имеем дело с идеальным в звуковом отношении окружением, где нет каких-либо посторонних шумов.

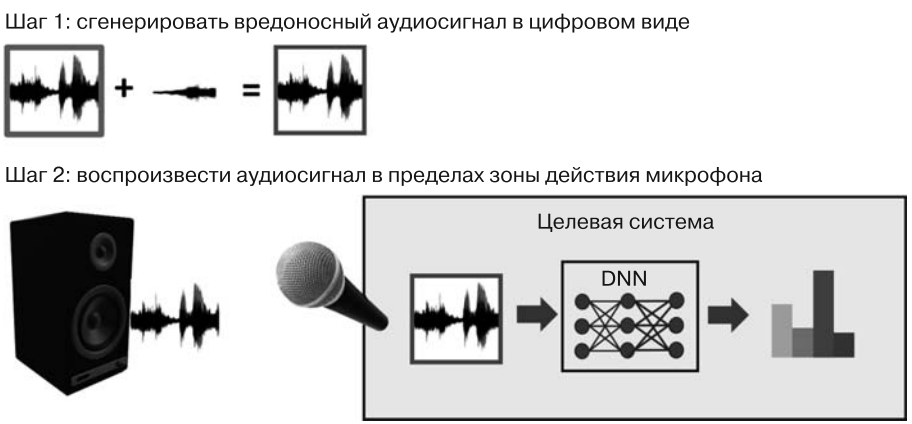


Рис. 8.6. Вредоносная атака в физическом мире с использованием аудиосигнала

При этом необходимо учесть следующие ограничения.

- *Воспроизведение цифрового аудиосигнала (возможности динамиков).* Как и для печати вредоносных объектов, самое главное требование состоит в том, чтобы воспроизведение цифрового звука в виде звуковых волн выполнялось с тем уровнем точности, который необходим для

эффективного срабатывания вредоносного искажения. А это зависит от точности воспроизведения и возможностей динамиков.

Интересная особенность данного вида атаки — обычно злоумышленник не имеет контроля над динамиками, с помощью которых цифровой аудиосигнал превращается в звук. Так, например, при звуковой вредоносной атаке, осуществляемой через Интернет, качество звука целиком зависит от того, какие динамики используются в воспроизводящем устройстве — это могут быть и высококачественные динамики домашней акустической системы, и низкокачественные динамики простейшего смартфона.

Динамики превращают цифровой аудиосигнал в звуковые волны, приводя в колебательное движение упругий материал (мембрану динамика). Подобно тому как принтеры не позволяют воспроизводить абсолютно любые комбинации RGB-значений, высота воспроизводимого динамиком звука ограничена предельной скоростью физического движения мембраны динамика. Если цифровой вредоносный аудиосигнал не впишется в это ограничение по высоте, то динамик не сможет его воспроизвести.

- *Возможности микрофона*. При захвате данных микрофоном существует риск появления шумов и искажений. Подобно тому как при захвате изображений на них могут появляться блики, в аудиоданных могут появляться помехи, вызываемые электрическими флуктуациями на этапе первоначального преобразования звука в данные. Микрофон или цепь предварительной обработки также обычно выполняет обработку (например, с использованием MFCC-коэффициентов, см. раздел «Аудиоданные» на с. 87), призванную выделить актуальные признаки и, наоборот, удалить те данные, которые не имеют отношения к выполняемой задаче.

Проведенные исследования показали, что возможно воспроизвести нераспознаваемые человеком вредоносные команды и успешно захватить их смартфоном[1], а значит, ответ на вопрос о том, можно ли создать вредоносное искажение и захватить его в цифровом виде, является положительным.

Положение аудиосигнала и окружение

Теперь посмотрим, какие дополнительные трудности могут порождаться окружением и относительным расположением динамика и микрофона.

- *Окружение*. Одна из главных проблем при проведении физической звуковой вредоносной атаки состоит в том, что вредоносным звукам часто

[1] *Vaidya T. et al.* Cocaine Noodles: Exploiting the Gap Between Human and Machine Speech Recognition // USENIX Workshop in Offensive Technologies, 2015. http://bit.ly/2FldYIj.

приходится соперничать с другими звуками в окружении. Хотя во многих случаях злоумышленник не имеет контроля над уровнем других шумов в окружении, иногда он все же может в какой-то мере контролировать уровень шумов или как минимум иметь представление о нем до проведения атаки. Возьмем, к примеру, случай создания голосовых команд для вредоносного воздействия на систему голосового управления автомобилем — при проведении такой атаки злоумышленник не может учесть абсолютно все имеющиеся посторонние звуки, но в то же время может учесть предсказуемое влияние шума двигателя.

Мы все знаем, насколько разным может быть тембр и характер звука в разном окружении. Так, например, одни и те же звуки могут звучать совершенно по-разному внутри и вне помещения. Внутри помещения звук отражается от стен и других поверхностей, что вызывает эффект реверберации. Кроме того, звуковые волны могут ослабляться и рассеиваться мебелью и другими предметами обстановки. То есть перед тем, как достигнуть микрофона, звук отражается, резонирует и ослабляется окружающими предметами. Значительное влияние на силу и качество звука также оказывает положение источника звука по отношению к микрофону целевого устройства. Создание вредоносного звука с низкой чувствительностью к изменению окружения и положения динамика/микрофона является непростой задачей, однако злоумышленник может сконцентрироваться на некотором подмножестве сценариев (например, на случае атаки в небольшом помещении) или, если звук воспроизводится в общественном месте, даже обеспечить управление динамиком и его положением.

- *Обратная связь и подтверждение*. Технология виртуальных помощников обычно использует некоторую форму звуковой или визуальной обратной связи для подтверждения намерений пользователя — особенно когда команды представляют значительный риск с точки зрения безопасности (как, например, команды на совершение онлайн-покупок). Так, виртуальный помощник может запрашивать подтверждение, выдавая соответствующий световой сигнал или голосовое сообщение. Злоумышленник может обеспечить такое подтверждение с помощью вредоносной команды (например, путем воспроизведения в тщательно рассчитанный момент неслышимой для человека команды «да» (yes)). Однако, помимо этого, нужно как-то позаботиться о том, чтобы никто не услышал звуковое сообщение обратной связи (такое как: «Вы действительно хотите осуществить этот коварный план?»).

Когда злоумышленник встраивает вредоносное искажение в полностью контролируемые им аудиоданные (например, в музыку предлагаемого в Интернете видеофайла), он может просто увеличить громкость «безобидного» звука в тот момент, когда должно прозвучать сообщение виртуального помощника. Однако более типичной является ситуация, когда не требуется, чтобы атака срабатывала всегда — она зачастую срывается из-за того, что кто-то слышит сообщение помощника, но иногда все же срабатывает, поскольку никого не оказывается рядом либо никто не замечает сообщение.

- *Ограничения*. Как и в случае вредоносных объектов, успешность звуковой вредоносной атаки обычно зависит от того, насколько большие изменения может внести злоумышленник в аудиоданные, оставаясь незамеченным. При этом спрятать вредоносные команды внутри существующих аудиоданных обычно труднее, чем внутри полностью контролируемых аудиоданных. Например, если злоумышленник вносит звуковое искажение в музыку или речь, которую он тоже создает сам, то он получает большую степень контроля над теми аудиоданными, которые маскируют атаку. Попутно встает и другой вопрос: требуется ли обеспечивать полную незаметность вредоносного звука? Если звук не привлекает внимание людей или игнорируется ими в силу того, что они не могут его распознать, то необходимость в его маскировке под обычную музыку или речь отпадает.

Провести атаки с использованием неслышимых голосовых команд можно даже в достаточно сложных окружающих условиях. Один из возможных подходов при этом сводится к тому, чтобы адаптировать описанный в разделе «Вредоносные объекты» на с. 187 метод EOT для создания вредоносных 3D-моделей под звуковую предметную область, рассматривая звуковые преобразования и накладываемые динамиком ограничения на воспроизведение звука.

Гипотетический пример: подрыв доверия к технологии голосовых помощников

Стремясь подорвать доверие к IT-компании, производящей голосовых помощников, группа злоумышленников встраивает вредоносные команды в, казалось бы, безобидный музыкальный трек. Это целевая атака, использующая такие точные команды, как запрос на выполнение онлайн-заказа с последующим его подтверждением (после паузы, в течение которой голосовой помощник запрашивает подтверждение). Группа злоумышленников встраивает вредоносный

аудиотрек в некоторый видеоконтент и распространяет его через несколько новостных каналов в социальных сетях.

Разработка вредоносных аудиоданных выполняется незаметно. Группа злоумышленников разрабатывает атаку с помощью замещающей ГНС, а затем проверяет ее на собственной локальной копии голосового помощника.

Использование точной копии системы на этапе разработки обеспечивает надежное срабатывание атаки при надлежащих физических условиях. В то же время атака имеет низкий показатель успеха, поскольку подразумевает, что голосовой помощник будет находиться поблизости от воспроизводимого аудиотрека в сравнительно тихом окружении. Однако даже при низком уровне успеха атака достигает своей цели, поскольку получившие огласку случаи ее успешного срабатывания снижают доверие потребителей к продукту и выпускающей его компании.

Осуществимость атак с использованием физических вредоносных образов

Генерация вредоносных объектов и звуков — более сложная задача по сравнению с созданием вредоносных образов в цифровом виде. Злоумышленнику приходится решать следующие дополнительные задачи:

- сохранение вредоносной информации в физическом представлении образа;
- обеспечение захвата вредоносной информации датчиком и ее обработки;
- учет варьирования физического положения и окружающих условий.

Хотя вряд ли можно ожидать, что злоумышленнику удастся создать физические вредоносные образы, способные срабатывать с неизменным успехом, у него остается серьезный мотив для проведения физических атак в тех случаях, когда не требуется высокий показатель успеха. Наглядной иллюстрацией такой мотивации могут служить примеры, приведенные во врезках «Гипотетический пример: ложные срабатывания системы видеонаблюдения» и «Гипотетический пример: подрыв доверия к технологии голосовых помощников» выше в этой главе.

ЧАСТЬ IV

Защита

Основываясь на информации, представленной в предыдущих главах, в данной части рассмотрены способы защиты от вредоносных входных данных глубоких нейросетей, используемых в составе реальных систем.

В начале главы 9 мы поговорим о том, как можно смоделировать вредоносную угрозу — это играет важную роль при оценке любого метода защиты. Далее в этой главе приведены эмпирические и теоретические методы оценки устойчивости нейронных сетей.

В главе 10 приведены последние тенденции в области усиления защиты ГНС-алгоритмов от вредоносных входных данных, а также ряд проектов с открытым исходным кодом, направленных на разработку более эффективных способов защиты от вредоносных атак. Рассмотрена также задача обеспечения безопасности ГНС в более широком контексте, чтобы разобраться с тем, можно ли найти для ГНС определенные наборы входных данных, в пределах которых она будет безопасно работать. В данной главе приведены примеры кода, иллюстрирующие методы защиты и способы их оценки, основанные на примерах кода, приведенных в главе 6. Здесь же уделено особое внимание проблеме обеспечения информационной безопасности и рассматривается, как общая последовательность обработки и используемые в организации меры защиты могут снижать угрозу со стороны вредоносных входных данных.

Наконец, в главе 11 речь пойдет о том, как, по всей вероятности, будут развиваться глубокие нейронные сети в ближайшие годы и как это повлияет на возможность их обмануть.

ГЛАВА 9

Оценка устойчивости модели к вредоносным входным данным

Прежде чем приступить к изучению методов защиты, рассмотрим способы оценки устойчивости ГНС-моделей к вредоносным образам. Это позволит заложить основу для оценки эффективности методов защиты, описание которых приведено в главе 10.

Оценка устойчивости отдельных компонентов ГНС делает возможным объективное сравнение моделей и подходов к их защите. Такая оценка может, к примеру, выполняться в исследовательских целях для определения того, насколько новый метод защиты эффективнее по сравнению с предыдущими, либо при развертывании в организации новой версии модели, чтобы гарантировать прежний или более высокий уровень безопасности по сравнению с ее предыдущей версией.

Оценка модели требует единообразия методологии и оценок, чтобы гарантировать объективность используемых для сравнения метрик. К сожалению, генерацию метрик, отражающих степень устойчивости нейронной сети к вредоносным образам, нельзя назвать простой задачей. Прежде всего необходимо выяснить, защита от *чего* нам нужна? Поэтому на основе глав 7 и 8 рассмотрим, как можно смоделировать ту угрозу, от которой необходимо защищаться. Используемые для этой цели *модели угроз* рассмотрены в разделе «Цели, возможности, ограничения и знания злоумышленника» далее.

Оценка полной осведомленности

При оценке методов защиты важно учесть, что к таковым ни в коем случае не стоит относить непосредственно поддержание секретности информации об устройстве целевой системы. Методы обеспечения информационной безопасности не следуют принципу «безопасность

через неясность»[1], и потому при любой оценке вредоносных атак следует предполагать, что злоумышленник обладает полной осведомленностью о ГНС и используемых методах защиты. Этот сценарий называется «атака с полной осведомленностью».

Оценка с использованием единообразных моделей угроз играет важную роль для объективного сравнения методов защиты. В разделе «Оценка модели» на с. 211 рассмотрены методы оценки устойчивости модели к вредоносным образам.

В подразделе «Эмпирические метрики устойчивости» на с. 212 рассмотрена роль модели угроз на этапе тестирования, а также ряд полученных эмпирическим путем метрик, применяемых для оценки устойчивости модели.

Затем в подразделе «Теоретические метрики устойчивости» на с. 218, разберемся, можно ли использовать теоретические подходы для оценки устойчивости модели. Формальное доказательство устойчивости модели имеет то преимущество, что позволяет получить метрики, характерные для конкретного типа модели, но не зависящие от типа угрозы.

И прежде чем приступить к изучению методов защиты, подведем итоги этой главы в разделе «Резюме».

Метод красной и синей команд

Метод красной и синей команд подразумевает, что оценивать защиту путем проведения атаки и разрабатывать ее должны разные люди (соответственно красная и синяя команды). Красная команда имитирует поведение злоумышленника, а синяя команда — поведение защищающейся организации. Разделение ролей на атакующую сторону и защищающуюся позволяет мыслить таким образом, что увеличивается вероятность обнаружения слабых мест системы.

Красная команда должна попытаться использовать уязвимости системы к вредоносным образам, сымитировав наиболее эффективную атаку с помощью методов, подобных описанным в главах 7 и 8. В отличие от обычной оценки модели красная команда не должна иметь полной осведомленности о целевой системе, так как это может плохо сказаться на ее образе мышления, в результате чего она может не заметить определенные уязвимости (например, потому, что не будет пытаться применить те виды атак, против которых, по имеющимся сведениям, предусмотрена защита).

[1] Согласно максиме Шеннона, «враг знает систему».

Оценка модели (и встроенных в нее средств защиты) играет важную, но отнюдь не исчерпывающую роль. Когда ГНС используется в составе реальной системы, она является лишь одним из компонентов общей последовательности обработки. И хотя оценка модели в отрыве от других компонентов позволяет сравнивать различные модели и гарантировать безопасность отдельного компонента, следует проверять и систему в целом. Обеспечение нужного уровня устойчивости *системы* к вредоносным образам требует тестирования безопасности общей системы с включением вредоносных образов в число возможных факторов уязвимости. Для тестирования систем в отношении кибербезопасности часто используется метод красной и синей команд.

Цели, возможности, ограничения и знания злоумышленника

Когда особенности атакующей стороны подвергаются тщательному систематическому изучению в контексте целевой системы, такое определение угроз называют *моделью угроз*. Модель угроз представляет собой результат подробного анализа особенностей злоумышленника, векторов атаки и рисков, проведенного в отношении конкретной организации и сценария. В этой главе не было цели дать полное описание модели угроз. Здесь описаны лишь некоторые ключевые моменты, которые обычно учитываются при моделировании угроз, исходящих от вредоносных образов, — сведения, используемые при самом обобщенном сценарии моделирования угроз.

Хотя в большинстве случаев нельзя смоделировать или предугадать все возможные угрозы, обычно все же целесообразно рассмотреть несколько вероятных сценариев. Если рассмотреть ситуацию глазами злоумышленника, это может помочь в проектировании надежных средств защиты от вредоносных входных данных.

Особенности атаки часто моделируются в терминах *целей, возможностей, ограничений и осведомленности* злоумышленника.

Цели

Для начала рассмотрим цели — чего хочет добиться злоумышленник и как это влияет на характер представляемых системе вредоносных входных данных?

На самом верхнем уровне цели угроз представляют собой то, чего хочет добиться злоумышленник, обманывая систему. Это снова возвращает нас к тем

исходным мотивам, о которых мы говорили в главе 2. В то же время не стоит сбрасывать со счетов и такие расплывчатые мотивы, как простое желание проявить свой талант в сфере кибератак, — такие мотивы могут порождать менее определенные цели и менее очевидные модели угроз.

Цели определяются высокоуровневой мотивацией. Эти цели могут характеризоваться требуемой специфичностью атаки, минимальной степенью успешности и предельной величиной искажения (рис. 9.1).

- *Специфичность*. Под требуемой специфичностью атаки имеется в виду, состоит ли ее цель в том, чтобы просто заставить ГНС выдавать неверное предсказание (нецелевая атака), или в том, чтобы обеспечить некоторое *конкретное* неверное предсказание (целевая атака). Данный параметр определяется мотивацией атаки. Например, атака уклонения может быть нецелевой, когда не имеет значения, как именно ГНС интерпретирует входной сигнал, и важно лишь, чтобы это не была конкретная интерпретация. С другой стороны, если цель атаки — дезориентация, часто требуется заставить ГНС выдавать целевую интерпретацию входных данных с тем, чтобы сгенерировать ложное срабатывание конкретного типа.

Рис. 9.1. Цели злоумышленника можно определить с точки зрения требуемой специфичности, степени успешности и степени заметности (предельной величины искажения)

- *Степень успешности.* Под степенью успешности атаки понимается степень уверенности злоумышленника в том, что атака сумеет обмануть классификатор с нужной специфичностью.

  Необходимая злоумышленнику доля успешно завершающихся атак зависит от того, к каким последствиям ведет неудачное завершение атаки. Так, атака уклонения, как правило, требует большей уверенности в успехе, чем атака ложного срабатывания. Если неудачное завершение атаки уклонения чревато такими серьезными последствиями, как судебное преследование, злоумышленник может направить больше усилий на реализацию методов, способных обеспечить надежное срабатывание вредоносных входных данных. С другой стороны, если цель сводится лишь к тому, чтобы дезориентировать ИИ с помощью множества вредоносных входных сигналов, вызывающих ложные срабатывания, неудачное завершение определенной части атак не будет играть большой роли. Помимо таких очевидных мотивов, как стремление избежать распознавания, не стоит забывать и о менее заметных, например о желании вызвать отказ в обслуживании или дезориентацию либо просто подорвать доверие к организации. Эти разновидности атак, как правило, не требуют высокой степени успешности, что существенно упрощает их реализацию.

- *Предельная величина искажения (степень заметности).* Предельная величина искажения — это максимально допустимый для атаки размер искажения, который служит косвенным показателем степени ее заметности. Допустимая степень заметности вредоносных входных данных зависит от цели и контекста атаки. В некоторых случаях даже не требуется как-либо маскировать вредоносные изменения. Примером такого случая может служить наложение вредоносной заплатки в углу изображения. О том, что на изображение наложена заплатка, может быть известно и отправителю, и получателю изображения, поэтому отправителю не нужно как-либо маскировать заплатку, если она не влияет на основное содержимое изображения. В других случаях может быть очень важным минимизировать степень заметности. Например, при встраивании в аудиосигнал вредоносной голосовой команды обычно нужно сделать ее незаметной.

  Хотя величина искажения обычно оценивается с помощью L^p-нормы, это никоим образом не исключает использования какого-либо другого показателя величины изменения. Этот показатель может вычисляться с помощью более сложного алгоритма, учитывающего особенности чело-

веческого восприятия или то, в каком контексте воспринимается вредоносный образ, — так, более заметному изменению может присваиваться более высокая величина искажения, чем менее заметному изменению, даже если они обеспечивают одинаковое «цифровое расстояние». В случае вредоносной заплатки пороговая величина искажения выбирается с учетом того, что величина искажения должна оцениваться с помощью L^0-нормы (то есть просто по количеству изменяемых пикселей), а также того, что эти пиксели должны располагаться в определенной части изображения (например, возле его края).

Между степенью успешности и величиной искажения есть очевидная взаимосвязь — степень успешности будет тем выше, чем меньше будут ограничения на величину искажения. Если бы мы построили график зависимости степени успешности конкретной атаки от допустимой величины искажения определенного входного сигнала *(x)*, то он выглядел бы примерно так, как показано на рис. 9.2. Этот график показывает, насколько облегчается создание вредоносных образов для конкретного входного сигнала по мере роста допустимой величины искажения. Обратите внимание, что величина искажения не может быть меньше некоторого значения, являющегося минимально необходимым для создания вредоносного образа. Форма графика и минимальная величина искажения зависят от конкретной сети, исходного изображения и способа количественной оценки искажения. Возьмите это на заметку, потому что данный график и связанные с ним показатели могут использоваться и для оценки устойчивости нейронных сетей. В главе 10 мы еще раз вернемся к графикам такого рода и посмотрим, с помощью какого кода они генерируются.

Специфичность атаки варьируется путем увеличения или уменьшения ограничений логики генерации вредоносных входных данных. Целевые атаки накладывают более жесткие ограничения на генерируемые вредоносные входные данные, и потому их обычно труднее реализовать, что, в свою очередь, не гарантирует успеха.

Цели атак инкапсулируются в математических и логических ограничениях, используемых для генерации вредоносных образов. В качестве примера можно рассмотреть математические выкладки C&W-атаки (Карлини и Вагнера), представленные на рис. 9.3. (Полное описание этих выкладок было представлено в подразделе «Повышение надежности вредоносного искажения» на с. 154.)

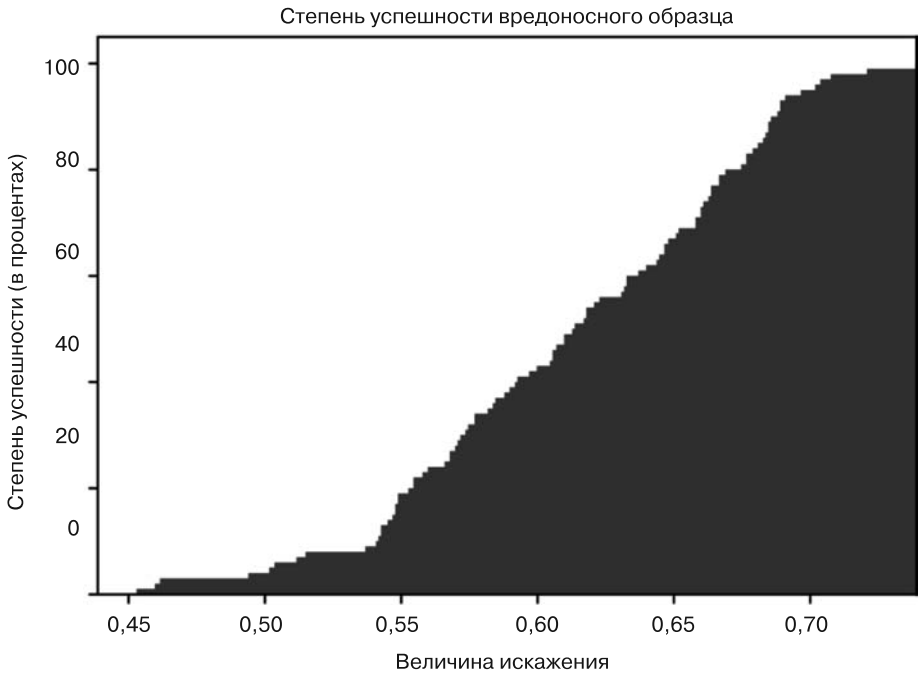


Рис. 9.2. Увеличение допустимой величины искажения ведет к увеличению степени успешности вредоносного образа

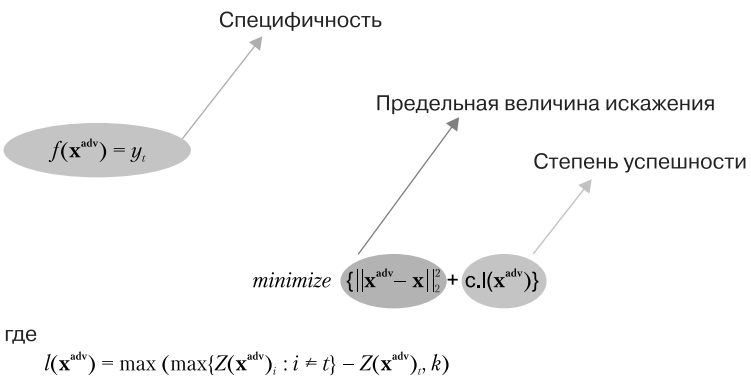


Рис. 9.3. Вредоносные цели отражены в математических выкладках алгоритма Карлини и Вагнера

Вы видите, как выражает вредоносные цели данный алгоритм. Специфичность (в данном случае целевой атаки) отражена в том условии, что ГНС должна возвращать целевую категорию для вредоносного входного сигнала.

При расчете искажения минимизируется расстояние между вредоносным образом и исходным изображением, которое оценивается с помощью L^2-нормы, выражаемой формулой $\left\|\mathbf{x}^{\mathbf{adv}} - \mathbf{x}\right\|_2^2$. Тем самым реализуется цель обеспечения нужной степени заметности для вредоносного искажения. Дополнительный параметр надежности $c.l(\mathbf{x}^{\mathbf{adv}})$ определяет ожидаемый успех.

Описанные в главе 6 методы генерации атак и, в частности, метод Карлини и Вагнера, позволяют злоумышленнику разработать атаку в соответствии со своими целями, но без учета чего-либо еще, помимо самого алгоритма нейронной сети. Однако на самом деле атака проводится против всей последовательности обработки вместе с имеющимися механизмами защиты, и потому обеспечение нужной специфичности, степени успеха и величины искажения для реальной целевой системы обычно требует большего объема математического моделирования и/или тестирования на полной целевой системе. То есть, если мы сгенерируем вредоносный образ, используя алгоритм, подобный приведенному на рис. 9.3, для ГНС в отрыве от других компонентов, это не гарантирует, что заданные цели в отношении специфичности, степени успеха и величины искажения будут достигнуты при применении образа против системы в целом.

Возможности, осведомленность и доступ

То, насколько осуществимыми будут эти цели для злоумышленника, зависит от нескольких факторов: от его возможностей, осведомленности о целевой системе и способности влиять на входные данные с целью сделать их вредоносными. Последние два фактора можно рассматривать как потенциальные ограничения атаки — малая осведомленность будет мешать злоумышленнику в достижении его целей, и то же самое можно сказать о слабой способности влиять на входные данные. Это показано на рис. 9.4.

Давайте рассмотрим каждый из этих факторов.

- *Возможности*. Успешность атаки зависит от того, какими ресурсами (навыками, программным и аппаратным обеспечением, запасом времени) располагает злоумышленник. Какие ресурсы могут быть необходимы, во многом зависит от целей атаки. Простая малозатратная атака может сводиться к простому наложению на цифровое изображение распространяемой через Интернет вредоносной заплатки. С другой стороны, разработка надежного искажения с помощью набора моделей и больших вычислительных мощностей обычно требует больших затрат времени и денег и достаточной квалификации.

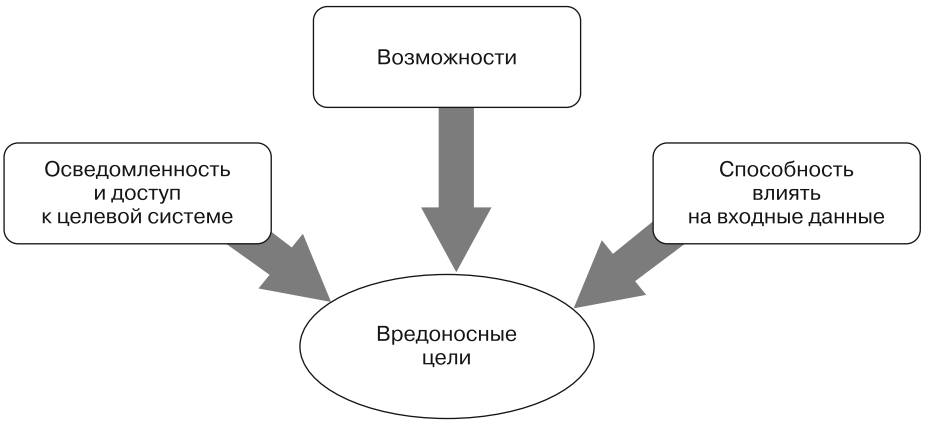


Рис. 9.4. Вредоносные цели ограничены степенью осведомленности и доступа злоумышленника к целевой системе, его возможностями и способностью влиять на входные данные

Не стоит недооценивать возможности злоумышленников, действующих в одиночку. Даже когда хакеры работают в одиночку, они, как правило, обладают высокой квалификацией и могут использовать для разработки атаки общедоступное облако и другие интернет-ресурсы.

- *Способность влиять на входные данные*. Возможности злоумышленника при создании вредоносных входных данных могут ограничиваться тем, в какой мере он способен изменять данные. Например, возможность изменить содержимое цифрового изображения может ограничиваться максимально допустимой степенью влияния изменений на человеческое восприятие.

  В случае атаки в физическом мире у злоумышленника нет доступа к цифровым данным, что накладывает существенные ограничения на создание вредоносного контента. Кроме того, злоумышленник может столкнуться с такими простыми проблемами, как отсутствие доступа к месту расположения датчика.

- *Осведомленность о целевой системе или доступ к ней*. Даже осведомленность только о самой ГНС-модели уже служит существенным подспорьем в создании вредоносных образов, позволяя, к примеру, использовать для разработки вредоносных входных данных более мощные возможности атаки с копированием. Однако осведомленность о целевой системе не следует рассматривать исключительно как способность злоумышленника скопировать нейросетевую модель. Для успешного применения надежного

вредоносного образа желательно иметь информацию обо всей последовательности обработки и всех ее механизмах защиты.

Если у злоумышленника нет полной осведомленности о целевой системе, он может составить представление о поведении целевой системы путем анализа ее ответов на запросы. Непосредственное экспериментирование с целевой системой повышает надежность вредоносного образа. Но в то же время может привести к его обнаружению, если целевая система проверяет подозрительные входные данные. Таким образом, доступ к целевой системе во время подготовки атаки ограничивается необходимостью оставаться незамеченным. Злоумышленнику нужно позаботиться о том, чтобы его запросы не выглядели подозрительно (например, увеличив интервал между запросами или направляя запросы в систему от нескольких IP-адресов, чтобы они казались не связанными друг с другом).

Если у злоумышленника есть собственная копия целевой системы (например, виртуального помощника), то он обладает неограниченным доступом к системе для проведения экспериментов и повышения осведомленности о ней. Однако если система получает данные из физического мира и выдает непосредственный ответ через нецифровые механизмы, это затрудняет автоматизированное экспериментирование с ней. Например, вы не сможете использовать программируемый интерфейс для взаимодействия с виртуальным помощником. В таких случаях злоумышленник обычно генерирует вредоносные данные с помощью замещающей модели, затем выполняет их доводочное тестирование на физической копии, после чего уже направляет их в целевое устройство.

Оценка модели

Интересный вопрос: можно ли количественно оценить устойчивость модели к вредоносным образам? Это позволило бы сравнивать и проверять безопасность моделей, развертываемых в эксплуатационном окружении. Например, иногда полезно количественно оценить, как влияет на устойчивость ГНС определенный метод защиты, особенно если он несколько снижает точность модели. Также полезно объективно сравнить разные методы защиты и установить, какой из них наиболее эффективно обеспечивает безопасное функционирование модели.

На данный момент не представляется возможным создание ГНС, способной безупречно работать во всем диапазоне возможных входных данных.

Так, например, даже если сеть принимает цветные изображения с низким разрешением (224×224 пикселя), нужно убедиться в ее способности правильно обработать $256^{150\,528}$ различных изображений. Как известно, глубокие нейросети способны выдавать очень точные результаты в том случае, когда входные данные похожи на данные обучающего набора и не сделаны умышленно вредоносными, однако проверка работоспособности сети во всем диапазоне входных данных не представляется осуществимой с точки зрения вычислительных возможностей.

Оценка методов защиты может проводиться эмпирически (путем тестирования) или теоретически (с помощью математических расчетов). И в том и в другом случае важно не упускать из вида влияние метода защиты на точность результатов, выдаваемых моделью для невредоносных данных. Здесь есть два аспекта.

- После введения в действие методов защиты необходимо заново проверять точность результатов, выдаваемых моделью для невредоносных данных. Величина допустимого снижения точности модели во многом определяется сценарием ее использования.
- При разработке методов защиты с целью расширения области действия существующих моделей важно следить за тем, чтобы это не привело к снижению точности модели в диапазоне *«хороших»* данных. Слишком усердный механизм защиты от вредоносных данных может ошибочно посчитать неопасные данные вредоносными (то есть отнести их к случаю ложного срабатывания).

Рассмотрим эмпирический и теоретический подходы к оценке устойчивости.

Эмпирические метрики устойчивости

Сложность с вредоносными образами состоит в том, что они представляют собой *умышленные* попытки обмануть модель. При этом очень сложно создать соответствующие тестовые вредоносные данные и универсальные метрики устойчивости, поскольку:

- трудно предугадать характер вредоносных данных. Вредоносные данные, сгенерированные с помощью одного метода атаки, могут иметь совершенно не такие характеристики, как вредоносные данные, сгенерированные с помощью другого метода;

- хотя вероятность того, что данные смогут обмануть модель в ходе ее нормальной работы, может быть очень низкой, эта вероятность существенно возрастает в том случае, когда злоумышленник прилагает специальные усилия к тому, чтобы обмануть модель.

Для того чтобы определить точные и удобные сравнительные метрики устойчивости для конкретной сети, нужно четко определить те модели угроз, типы атак и тестовых данных, на основе которых будет оцениваться устойчивость модели.

- *Модель угроз*. Учитывает цели атаки — ее специфичность (то есть целевой или нецелевой она является), степень успешности и пороговый уровень заметности (например, используемую разновидность L^p-нормы и ее допустимые пределы). Она также может учитывать способность злоумышленника влиять на входные данные (с учетом, например, ограничений физического мира). Оценка не имеет какого-либо смысла без четкого определения модели угроз, которая задает необходимый объем вредоносных тестовых данных.

  Выбор конкретного типа модели угроз зависит от того, с какой целью выполняется оценка. Например, вы можете оценивать защитный механизм в исследовательских целях, чтобы создавать более эффективные и надежные нейронные сети. В таком случае, вероятно, лучше использовать универсальные модели угроз, чтобы можно было проводить непосредственное сравнение с другими методами защиты. С другой стороны, можете оценивать защитный механизм для конкретного эксплуатационного окружения. В этом случае лучше сосредоточиться на конкретных моделях угроз и сценариев тестирования (например, на определенной разновидности целевых атак), которые представляют наибольший риск для вашей организации. На практике вы можете выполнять сразу несколько оценок с помощью разных моделей угроз.

- *Методология атаки*. Часть оценки состоит из исчерпывающего описания методов атаки, включающего в себя сведения обо всех используемых параметрах. При генерации тестовых вредоносных образов следует предполагать полную осведомленность о системе и ее механизмах защиты.

  Если вы докажете эффективность защитного механизма против одного метода атаки, это еще не будет гарантировать его эффективность против другого, более сильного метода. Поэтому оценка должна охватывать целый ряд методов атаки. Так как знать все методы просто невозможно, в число

рассматриваемых следует включить самые сильные методы из известных на текущий момент. Цель состоит в том, чтобы оценить защищенность от наиболее эффективных разновидностей атак, в том числе таких, которые способны адаптироваться и обходить используемые методы защиты.

- *Тестовые данные*. Оценка устойчивости метода защиты связана с применяемыми в ходе экспериментов тестовыми вредоносными данными. Эти данные определяются используемым методом атаки (в том числе его параметрами), а также данными, на основе которых этот метод генерирует тестовые вредоносные образы. Варьируя те исходные данные, на основе которых генерируются тестовые вредоносные образы, можно влиять на степень успешности атаки.

Оценка устойчивости к вредоносным образам является частью более широкой задачи оценки правильности работы модели при представлении ей различных входных данных. Соответственно, оценка методов защиты является частью более широкой задачи оценки точности модели. Исходя из предположения, что тестовые данные распределены так же, как и обучающие, можно провести эмпирическую оценку точности модели наиболее распространенными методами.

Представленный в главе 3 код для создания классификатора изображений из набора данных Fashion-MNIST демонстрирует простейший способ оценки модели:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Model accuracy based on test data:', test_acc)
```

Данный код сгенерирует следующий вывод:

```
10000/10000 [================] - 0s 35us/sample - loss: 0.3623 - acc: 0.8704
Model accuracy based on test data: 0.8704
```

Эта оценка показала, что примерно 87 % представленных модели тестовых примеров классифицированы правильно. Более детально изучить точность модели позволяет *матрица несоответствий (confusion matrix)*. На рис. 9.5 показана матрица несоответствий, составленная для классификатора из главы 3.

Пример кода: матрица несоответствий

Код для создания этой матрицы можно найти в следующем блокноте Jupyter: chapter03/fashionMNIST_classifier.ipynb (bit.ly/31JsseI).

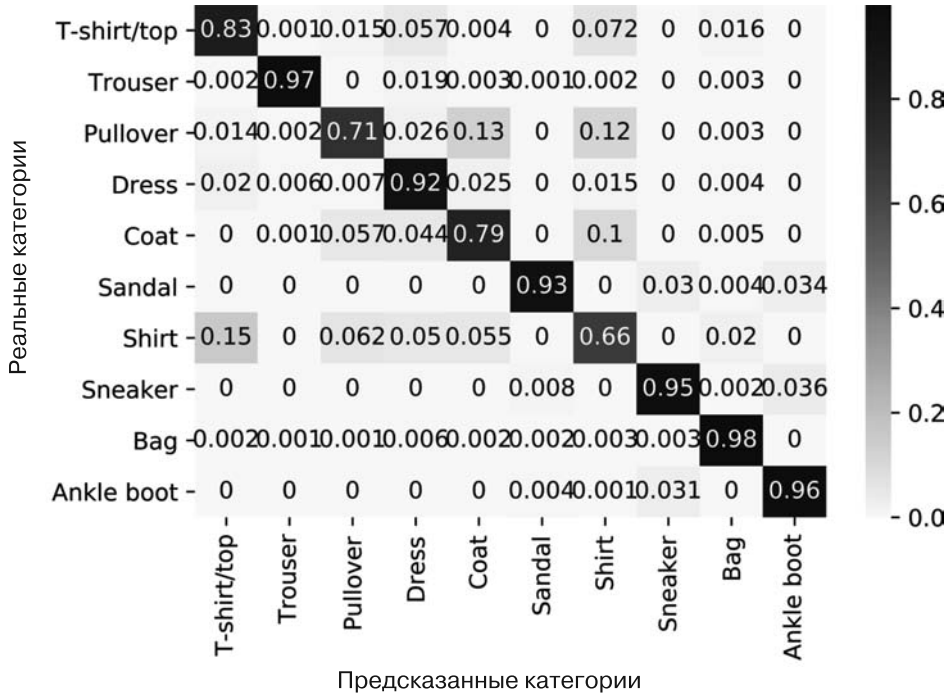


Рис. 9.5. Матрица несоответствий для классификатора Fashion-MNIST предоставляет сводные данные о том, насколько хорошо справляется модель с распознаванием каждой категории одежды

Каждая строка в этой матрице относится к тестовым изображениям с определенной (правильной) меткой. Так, например, верхняя строка относится к тестовым данным с меткой «Футболка/топ» (T-shirt/top). Значения в ячейках указывают, какая доля тестовых данных отнесена классификатором к метке соответствующего столбца. Так, доля футболок и топов, которые правильно отнесены к своей метке, составляет `0.83` (83 %). Доля футболок и топов, которые ошибочно отнесены к метке «Рубашка» (Shirt), составляет `0.07` (7 %). Если бы модель показала на тестовых данных идеальные результаты, то все ячейки выделенной диагонали содержали бы значение `1.0`, а все остальные ячейки — `0`.

Матрица несоответствий позволяет многое узнать о соответствующей модели классификатора. Например, матрица, представленная на рис. 9.5, показывает, что чаще других неверно классифицируются изображения рубашек — в 15 %

случаев они ошибочно относятся к категории «Футболка/топ» (T-shirt/top). И точно так же можно сделать вывод о том, что наиболее точно данная модель классифицирует тестовые изображения брюк, сумок и ботильонов (98 %). Как вы увидите в главе 10, матрица несоответствий может использоваться для отражения того, в какой мере тестовые вредоносные данные обеспечивают ошибочную классификацию.

Чтобы использовать матрицу несоответствий для оценки, мы должны создать соответствующую методологию атаки и сгенерировать тестовые вредоносные данные. Для оценки эффективности конкретного метода защиты сначала тестируется незащищенная модель, а затем — модель с уже введенной в действие защитой. Оценка должна проводиться с помощью самых сильных возможных методов атаки, специально направленных на преодоление имеющихся защитных механизмов. Это очень сложно, так как постоянно появляются новые виды атак, и, соответственно, оценка устойчивости к атакам лишь отражает состояние на некоторый момент времени и обычно должна проводиться повторно при появлении в дальнейшем новых видов атак. При этом нас опять же может интересовать лишь то, насколько изменится величина искажения, способная обеспечить определенную степень успешности. В другом случае наиболее важной метрикой может быть изменение величины искажения, минимально необходимой для создания вредоносного образа. Как можно оценивать защиту с помощью матрицы несоответствий, показано в примере кода в главе 10.

Еще одна метрика — степень сложности (выражаемая как величина искажения) создания вредоносных образов, направленных против конкретной целевой сети. Это снова возвращает нас к графику, который был представлен на рис. 9.2 и еще раз приведен на рис. 9.6.

Для оценки устойчивости сети к любой возможной атаке можно рассмотреть самый худший случай (с точки зрения защищающейся стороны), которому соответствует величина искажения, минимально необходимая для того, чтобы атака закончилась успехом (точка 1 на графике). Если модель угроз не накладывает ограничений на допустимые пределы заметности, то оценка защиты сводится просто к определению того, в какой точке обеспечивается достаточно высокая степень успешности для осуществления цели модели угроз (точка 2 на графике). Как генерируются графики подобного рода, подробно рассмотрено в примере кода в главе 10.

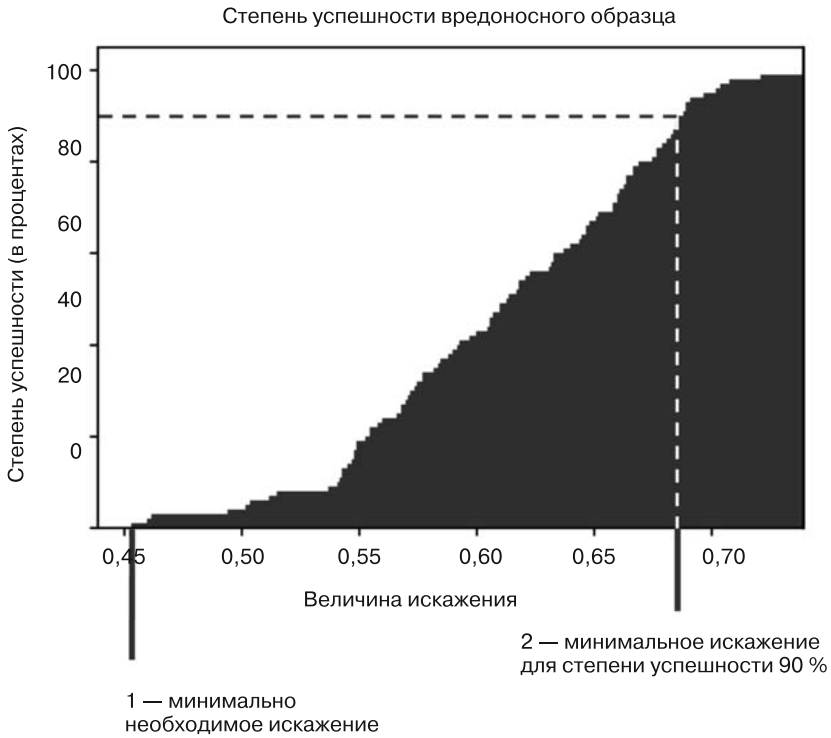


Рис. 9.6. Увеличение допустимой величины искажения ведет к увеличению степени успешности вредоносного образа

Оценка метода защиты, применяемого к разным моделям

Важно понимать, что результаты оценки метода защиты на одной модели, как правило, не распространяются на другие модели. Так, в частности, результаты оценки метода защиты на модели, обученной с помощью такого «игрушечного» набора данных, как Fashion-MNIST, не распространяются на модели, обученные с помощью более реалистичных данных.

Когда требуется оценить устойчивость только одной сети (например, в рамках проверки модели перед ее развертыванием в эксплуатационном окружении), обычно уместно провести оценку на одной модели до и после введения метода защиты в действие. Однако, когда нужно провести более

общую оценку метода защиты (например, для исследовательских целей), можно составить лучшее представление о его эффективности, выполнив оценку на нескольких разных моделях.

Теоретические метрики устойчивости

Эмпирическим показателям устойчивости свойственна неоднозначность из-за большого количества переменных, связанных с методом оценки. Они также не являются гарантированной мерой устойчивости сети, поскольку постоянно появляются все более эффективные методы создания вредоносных образов.

Оценка устойчивости с помощью математического, а не эмпирического подхода позволяет получить более универсальные и надежные метрики. Математически рассчитанные метрики для контроля безопасности ПО особенно актуальны для оценки систем, требующих высокой степени безопасности. В контексте вредоносных образов теоретическая оценка может быть необходима, к примеру, для обеспечения достаточной безопасности определенного компонента автономного транспортного средства. Метрики, полученные теоретическим путем исключительно на основе модели (но не на основе угрозы), также обладают тем преимуществом, что не зависят от типа применяемых атак.

Один из возможных подходов сводится к тому, чтобы математически рассчитать ту величину искажения, которая минимально необходима для создания вредоносного образа. К настоящему времени исследователям удалось доказать невозможность создания вредоносных образов в пределах определенного расстояния от ряда конкретных входных сигналов. Таким образом, предположив, что каждый нормальный входной сигнал окружен безопасной зоной, в которой нет вредоносных образов, вероятно, можно рассчитать минимальный размер этой зоны для всех таких входных сигналов (соответствующий наихудшему случаю). Эта идея отображена на рис. 9.7.

Рассчитываемая таким образом метрика представляет собой величину искажения, минимально необходимого для создания вредоносного образа во всем диапазоне правильно классифицируемых входных сигналов (наименьший размер безопасной зоны). Хотя эти расчеты могут осложняться огромными размерами входного пространства и сложным характером ландшафта предсказаний, исследователями предложены методы, позволяющие достаточно

точно определить метрику устойчивости путем математической аппроксимации[1]. На момент написания книги эти исследования находились еще на стадии зарождения, тем не менее можно ожидать, что такие теоретические метрики будут играть все более важную роль при оценке безопасности глубоких нейросетей.

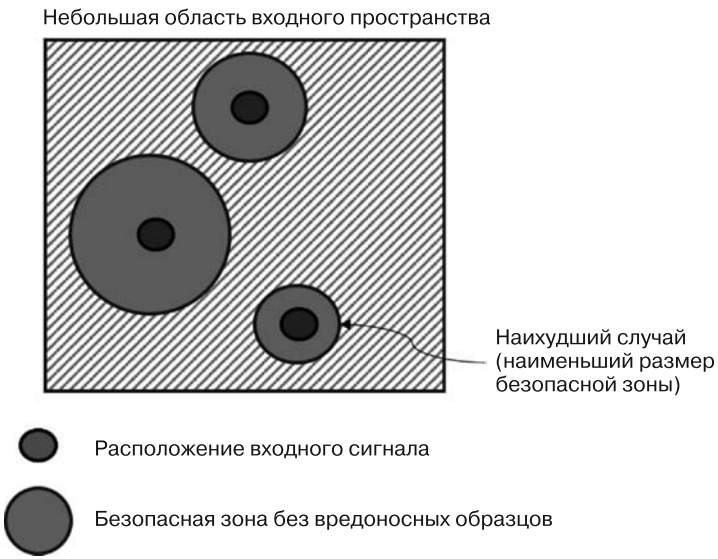


Рис. 9.7. Расчет размера безопасной зоны входного пространства, свободной от вредоносных образов

Резюме

Из данной главы вы узнали, что представляет собой модель угроз, и изучили различные методы оценки устойчивости к вредоносным образам. Несмотря на то что выработать универсальный подход к оценке модели достаточно сложно, интерес к разработке стандартных эмпирических показателей для оценки устойчивости модели, проводимой в рамках открытых проектов, перечисленных в главе 10, достаточно высок.

---

[1] *Weng T.-W. et al.* Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach // International Conference on Learning Representations, 2018. bit.ly/2Rn5THO.

Открытые проекты в области оценки устойчивости к вредоносным атакам

Разработка методологий оценки устойчивости к вредоносным атакам — новая область исследований. Статья Карлини и др. «Об оценке устойчивости к вредоносным атакам»[1] является живым документом, своего рода открытой площадкой для обсуждения и обмена идеями в области оценки защитных механизмов нейросетей.

Результаты любой эмпирической оценки модели справедливы только в отношении конкретной угрозы, и сравнение различных моделей возможно лишь при том условии, что в каждом случае будет рассматриваться одна и та же угроза. В этом свете использование метрик, получаемых теоретическим путем исключительно на основе модели, выглядит весьма привлекательно. Будем надеяться, что дальнейшие исследования в этой области приведут к появлению новых способов получения объективных метрик для сравнения моделей.

В следующей главе рассмотрен ряд предложенных на этот момент методов защиты от вредоносных входных данных на основе описанных в главе 9 методов оценки.

[1] *Carlini N. et al.* On Evaluating Adversarial Robustness, 2019. bit.ly/2IT2jkR.

ГЛАВА 10

Защита от вредоносных входных данных

В этой главе рассмотрен ряд предложенных на данный момент методов выявления атак с использованием вредоносных образов и защиты от них. Хорошая новость — некоторые методы защиты могут работать. Плохая новость — у каждого метода защиты есть свои ограничения и, зная о том, какой именно метод применяется, злоумышленник может скорректировать атаку так, чтобы она сработала в обход защиты.

В данной главе рассмотрены три широких подхода к защите.

- *Улучшение модели*. В первой части этой главы внимание сосредоточено на самой модели и предложенных на данный момент способах создания более устойчивых нейронных сетей.
- *Устранение вредоносных свойств входных данных*. В разделе «Предварительная обработка данных» на с. 248 рассматривается вопрос о том, можно ли превратить вредоносный входной сигнал в неопасный перед тем, как он будет передан модели.
- *Минимизация осведомленности злоумышленника*. В разделе «Сокрытие информации о целевой системе» на с. 254 говорится о том, как можно уменьшить степень осведомленности злоумышленника о целевой модели и общей последовательности обработки, чтобы сделать более затруднительным создание успешных вредоносных образов. Как уже упоминалось в главе 9, сокрытие информации о целевой системе не следует относить к числу методов защиты.

Данная задача пока не имеет универсального решения, однако исследователи ведут активную работу в этом направлении. В табл. 10.1 представлена сводная информация об известных на текущий момент возможностях описываемых в данной главе методов защиты.

Таблица 10.1. Сводная информация о методах защиты

Метод защиты	Повышение устойчивости модели	Устранение вредоносных характеристик данных	Минимизация осведомленности злоумышленника
Маскирование градиентов (см. раздел «Улучшение модели» ниже)	Частичное	Нет данных	Нет данных
Вредоносное обучение (см. раздел «Улучшение модели» ниже)	Частичное	Нет данных	Нет данных
Обучение распознаванию данных вне распределения (см. раздел «Улучшение модели» ниже)	Перспективно, но без гарантии	Нет данных	Нет данных
Случайный отсев на этапе тестирования (см. раздел «Улучшение модели» ниже)	Перспективно, но без гарантии	Нет данных	Нет данных
Предварительная обработка данных (см. раздел «Предварительная обработка данных» ниже)	Нет данных	Частичное	Нет данных
Сокрытие информации о целевой системе (см. раздел «Сокрытие информации о целевой системе» ниже)	Нет данных	Нет данных	Частичное

На практике возможности злоумышленника в плане успешного проведения вредоносной атаки обычно определяются целым рядом факторов, зависящих от общей последовательности обработки и ее системы безопасности. В данной главе рассматривается как сама модель, так и та система, составной частью которой она является.

В последнем разделе этой главы «Создание эффективных механизмов защиты от вредоносных входных данных» приведены рекомендации по повышению устойчивости реальных систем к вредоносным образам.

Улучшение модели

Прежде всего посмотрим, что можно сделать для защиты самой модели от вредоносных образов. Например, возможно, мы могли бы дообучить ГНС так, чтобы она стала устойчивой к вредоносным входным данным? Или мы могли бы использовать для распознавания атак некоторые общие характеристики вредоносных образов. А может, мы могли бы предугадать, в каких случаях

алгоритм будет работать неправильно, и, исходя из этого, снизить степень доверия к результатам, получаемым с меньшей степенью уверенности.

Важно помнить о том, что любое изменение модели с целью ее защиты не должно приводить к недопустимому ухудшению точности алгоритма; наряду с влиянием защитного механизма на вредоносные входные данные следует принимать во внимание и его влияние на *хорошие* данные.

Мы рассмотрим четыре метода защиты.

- *Маскирование градиентов*. Данный метод сводится к изменению модели таким образом, чтобы были скрыты градиенты на ландшафте предсказаний, что затрудняет создание вредоносных образов.
- *Вредоносное обучение*. Данный метод подразумевает обучение (или дообучение) сети с тем, чтобы она научилась распознавать вредоносные входные данные. Это реализуется путем добавления вредоносных образов к тренировочным данным.
- *Детекция данных вне распределения (OoD detection)*. Здесь мы пытаемся научить сеть возвращать не только предсказание, но и некоторый показатель степени ее уверенности в этом предсказании, принимая во внимание положение данных по отношению к распределению, в пределах которого она способна выдавать точные результаты.
- *Оценка неопределенности случайного отсева*. Наконец, данный подход сводится к применению техники, называемой *случайным отсевом*, после обучения модели, с тем чтобы внести в предсказания сети некоторую неопределенность. Основанием для этого служит предположение о том, что вредоносным входным данным свойственна большая степень неопределенности и это можно использовать для их выявления.

Маскирование градиентов

Метод *маскирования градиентов*[1] призван сделать более сложным расчет вредоносных образов. Идея заключается в том, чтобы либо скрыть градиенты на ландшафте предсказаний алгоритма ГНС, либо сгладить их так, чтобы они были бесполезны для злоумышленника.

Конечно, такой подход полезен только в тех случаях, когда злоумышленник обладает достаточным доступом к алгоритму ГНС, позволяющим ему

[1] Этот термин был впервые использован в следующей статье: *Papernot N. et al.* Practical Black-Box Attacks Against Machine Learning // Sixth International Conference on Learning Representations, 2018. http://bit.ly/2IrqeJc.

использовать градиенты модели для разработки атаки. Вполне очевидно, что будет лучше, если вы исключите возможность атак с помощью градиентов модели путем сокрытия от злоумышленника информации об используемом алгоритме (см. раздел «Сокрытие информации о целевой системе» на с. 254). Атаки данного типа подразумевают применение к целевой системе градиентных методов белого ящика (см. раздел «Методы белого ящика» на с. 135) и методов с оценкой (см. раздел «Методы черного ящика с оценкой» на с. 163). Методы с оценкой получают градиенты путем логического вывода на основе предсказаний, поэтому, если нет абсолютной необходимости возвращать пользователю оценки сети, эта информация не должна быть представлена в ответе ни прямым, ни косвенным образом.

Для исключения возможности генерации вредоносных образов с помощью градиентных методов предложено использовать метод так называемой *защитной дистилляции*[1]. Данный метод представляет собой вариацию метода *дистилляции*, который изначально предлагалось применять для уменьшения размера нейронных сетей. Дистилляция нейронной сети воспроизводит функцию ГНС (ее параметры) таким образом, чтобы на ландшафте предсказаний были сглажены градиенты вблизи обучающих данных.

Как выполняется дистилляция нейронной сети

Чтобы дистиллировать нейронную сеть, ее сначала нужно обучить, как обычно, с использованием размеченных данных. После этого с помощью модели нужно получить предсказания для обучающего набора данных. Затем необходимо обучить новую, «дистиллированную» версию модели, на этот раз задействовав обучающие данные в сочетании с предсказаниями (значениями вероятности), а не исходными жестко заданными метками.

Обучение нейронных сетей с использованием значений вероятности обеспечивает более плавные градиенты на ландшафте предсказаний, чем при обучении сетей с помощью дискретных меток. Обеспечиваемое дистилляцией сглаживание ведет к уменьшению размера модели, но в то же время иногда приводит и к снижению ее точности. Обычно это вполне допустимый компромисс; уменьшение размеров модели может быть полезным при наличии аппаратных ограничений, например, в случае ее использовании на мобильных устройствах.

[1] *Papernot N. et al.* Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks, 2016. http://bit.ly/2KZXfOo.

Принцип действия дистилляции сети иллюстрирует рис. 10.1. На этом рисунке показано, как выглядит ландшафт предсказаний для некоторой категории до и после дистилляции. Если вы посмотрите, как выглядит входное пространство вблизи обучающих данных после дистилляции, то увидите, что здесь нет явно выраженных градиентов, поэтому методы, экстраполирующие градиенты в определенной точке входного пространства, работать не будут. В каком направлении следует двигаться для создания вредоносного образа, совершенно не ясно.

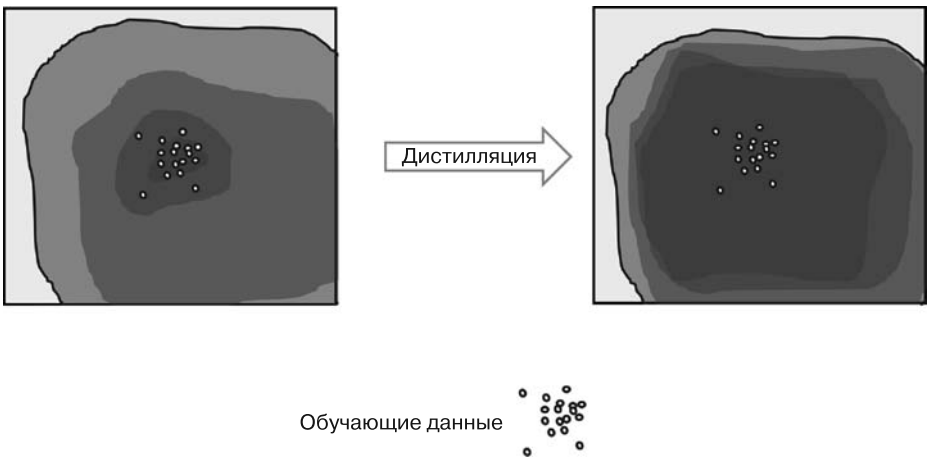


Рис. 10.1. Влияние дистилляции на характер ландшафта предсказаний для конкретной категории вблизи обучающих данных

Сглаживание градиентов модели имеет ограниченный потенциал защиты от вредоносных атак, поскольку базируется на предположении о том, что градиенты играют важную роль для создания вредоносных входных данных. Если мы обратимся к методам из главы 6, то увидим следующую картину. Градиенты играют важную роль для таких методов атаки, как FGSM и JSMA, но не для методов, которые осуществляют поиск во входном пространстве без использования градиентов, таких как метод белого ящика L-BFGS или методы ограниченного черного ящика вроде граничной атаки. Кроме того, злоумышленник может обойти защиту, маскируя градиенты (даже если применяется один из градиентных методов) путем дополнения алгоритма операцией генерации случайного искажения, которая будет смещать входной сигнал в область входного пространства, не охваченную маскированием градиентов.

Сглаживание градиентов также не обеспечивает защиту от атак с переносом. Как оказывается, вредоносные образы, созданные с помощью градиентных методов на замещающей модели, не подверженной маскированию градиентов, могут быть успешно перенесены на целевую модель, подверженную маскированию градиентов.

Вредоносное обучение

Вредоносное обучение, пожалуй, наиболее естественный подход к усилению защиты нейронных сетей от вредоносных образов. Ведь мы можем научить сеть различать сложные признаки и паттерны, а у вредоносных образов определенно есть признаки, которые она могла бы распознавать.

Вредоносные входные данные соответствуют тем слабым местам в алгоритме ГНС, которые обусловлены неспособностью ГНС обобщить абсолютно все входные данные. В силу этого выявление вредоносных входных данных и их надлежащая обработка как таковых будет повышать устойчивость алгоритма. Научившись выявлять вредоносные входные данные, вы сможете как следует на них реагировать, например снижая степень доверия к выдаваемому предсказанию либо снабжая такие входные данные пометкой об их вредоносности, запускающей дополнительную проверку или действия (то есть дополнительную классификацию этих входных данных). Еще один подход сводится к тому, чтобы научить сеть правильно относить вредоносные данные к их исходной (невредоносной) метке.

Как упоминалось в главе 7, предполагается, что большинство вредоносных образов находится во вредоносных подпространствах — непрерывных областях ошибочной классификации внутри входного пространства. В таком случае, вероятно, можно просто провести обучение (или дообучение) модели на основе большого количества размеченных вредоносных образов таким образом, чтобы она смогла правильно обобщить данные подпространств. Эта идея проиллюстрирована рис. 10.2.

Принцип обучения модели с помощью вредоносных образов изучен с различных точек зрения[1]. К сожалению, каким бы хорошим ни казался данный метод защиты, обученная таким образом модель будет устойчива только к тем вредоносным образам, которые были сгенерированы почти или абсолютно таким же образом, как вредоносные обучающие данные.

---

[1] Пример см. в следующей статье: *Goodfellow et al.* Explaining and Harnessing Adversarial Examples.

Рис. 10.2. Ландшафт предсказаний модели, обученной с использованием вредоносных образов

Чтобы понять, какие ограничения имеет метод вредоносного обучения, подумайте, каким образом для него генерируются вредоносные обучающие данные. Для обучения сети требуется сгенерировать большое количество тренировочных примеров с помощью методов, рассмотренных в главе 6 (или аналогичных). Некоторые методы затратны в вычислительном отношении или требуют большого количества итераций, что не позволяет генерировать эти данные в больших масштабах. Так, например, в случае итеративной граничной атаки на генерацию вредоносного образа уходят тысячи итераций, поэтому создание всего набора вредоносных обучающих данных с помощью этого метода будет занимать слишком много времени. В силу этого для создания обучающего набора, вероятно, лучше использовать методы, обеспечивающие быструю генерацию вредоносных сигналов за счет использования простой аппроксимации (например, метод FGSM и его вариации).

Однако при этом вы просто учите сеть распознавать вредоносные образы определенного типа. Если в целях экономии времени и ресурсов вы сгенерируете вредоносные обучающие данные методом белого ящика, аппроксимирующим градиенты модели, то в итоге вы получите ГНС, способную

распознавать исключительно признаки вредоносных образов, сгенерированных с помощью аналогичных методов. Как только злоумышленник попробует сгенерировать образы другим методом (например, методом граничной атаки, который не использует градиенты), эта защита даст сбой. Кроме того, если злоумышленник применит такой же простой градиентный метод на замещающей модели (имеющей некоторые отличия) и после этого осуществит атаку с переносом, данная защита будет плохо справляться с выявлением вредоносных образов, поэтому ее представление о том, что следует считать вредоносным, будет базироваться на ином наборе градиентов.

«Ну хорошо, — можете сказать вы, — в таком случае просто создадим вредоносные обучающие данные, используя для этого и метод граничной атаки!» Здесь-то и кроется проблема — поскольку постоянно появляются новые методы генерации вредоносных данных, у вас никогда не будет полной уверенности в устойчивости вашей ГНС. Вредоносное обучение позволяет перехватывать только аналогичные вредоносные входные данные, опираясь на обучающие данные, созданные аналогичными методами, и не гарантирует защиты от тех методов, которые вы не приняли во внимание, для которых вы не успели сгенерировать обучающие данные или которые еще не существовали на момент обучения сети.

Блокноты Jupyter для вредоносного обучения

Приведенные в данном разделе фрагменты кода представляют собой выдержки из следующих блокнотов Jupyter.

Блокнот Jupyter chapter10/fashionMNIST_adversarial_training.ipynb (http://bit.ly/2x3CA3A) содержит код для обучения сети с использованием вредоносных данных.

Блокнот Jupyter chapter10/fashionMNIST_adversarial_training_evaluation.ipynb (http://bit.ly/2x3mwiv) содержит код для проведения экспериментов и тестирования вредоносно обученной сети.

Для улучшения данного вида защиты предложено использовать *групповое (ансамблевое) вредоносное обучение*[1]. При этом вредоносные обучающие данные по-прежнему генерируются с помощью таких малозатратных методов, как FGSM и JSMA, но в то же время это делается с использованием

[1] *Tramèr F. et al.* Ensemble Adversarial Training: Attacks and Defenses // International Conference on Learning Representations, 2018. http://bit.ly/2XldcFh.

разных моделей, которые имеют различные параметры и, следовательно, градиенты. Благодаря этому модель обучается на вредоносных образах, которые уже не так тесно связаны с ее параметрами. В результате искажения вредоносные обучающие данные отличаются большей степенью разнообразия, а итоговая модель лучше справляется с распознаванием вредоносных входных данных.

Факт, что вредоносное обучение не дает гарантии в том, что итоговая модель способна правильно классифицировать абсолютно все вредоносные образы, еще не означает, что этот метод защиты бесполезен. Если модель способна правильно классифицировать хотя бы часть вредоносных образов, в этом уже определенная польза. В то же время не стоит полагаться на вредоносное обучение как на надежный метод защиты.

Попробуем выполнить вредоносное обучение и посмотрим, насколько это повысит устойчивость модели Fashion-MNIST к вредоносным образам.

Первое, что нужно сделать, — это создать некоторые вредоносные обучающие данные. Для демонстрации мы сгенерируем их, используя изображения из исходного набора обучающих данных и достаточно слабый метод простейшей знаковой градиентной атаки `GradientSignAttack` из библиотеки Foolbox.

Сначала следует определить атаку:

```
import foolbox
fmodel = foolbox.models.TensorFlowModel.from_keras(model, bounds=(0, 1))

attack_criterion = foolbox.criteria.Misclassification()
attack_fn = foolbox.attacks.GradientSignAttack(fmodel,
                            criterion=attack_criterion,
                            distance=foolbox.distances.Linfinity)
```

Включите в число обучающих данных дополнительные 6000 вредоносных изображений и дообучите модель:

```
x_images = train_images[0:6000, :]
predictions = model.predict(x_images)

x_train_adv_images, x_train_adv_perturbs, x_train_labels =
                     generate_adversarial_data(original_images = x_images,
                                               predictions = predictions,
                                               attack_fn = attack_fn) ❶
```

❶ `generate_adversarial_data` — вспомогательная утилита, добавленная в GitHub-репозиторий этой книги. Производит перебор предоставленных ей изображений, создавая по одному вредоносному образу для каждого из них (при условии, что такой вредоносный образ может быть создан). Кроме того,

возвращает дополнительную информацию о расстояниях искажений и метках, которая будет нам полезна.

Данный код выдаст следующие предупреждения:

```
Warning: Unable to find adversarial example for image at index:  2393
Warning: Unable to find adversarial example for image at index:  3779
Warning: Unable to find adversarial example for image at index:  5369
```

То есть для трех изображений из 6000 алгоритм не смог создать вредоносный образ. Что ж, у нас остаются 5997 образов, чего вполне достаточно для обучения модели.

Теперь нужно включить эти образы в число обучающих данных и дообучить модель:

```
train_images_adv = np.concatenate((train_images, x_train_adv_images),
                                  axis=0)
train_labels_adv = np.concatenate((train_labels,
                                   np.full(x_train_adv_images.shape[0],
                                           adversarial_label)),
                                   axis=0)

model_adv = keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                              keras.layers.Dense(56, activation='relu'),
                              keras.layers.Dense(56, activation='relu'),
                              keras.layers.Dense(10, activation='softmax',
                                                 name='predictions_layer')
                         ])
model_adv.compile(optimizer=tf.keras.optimizers.Adam(),
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])

model_adv.fit(train_images_plus_adv, train_labels_plus_adv, epochs=6)
```

Данный код сгенерирует следующий вывод:

```
Epoch 1/6
65996/65996 [================] - 5s 72us/sample - loss: 0.5177 - acc: 0.8151
Epoch 2/6
65996/65996 [================] - 4s 67us/sample - loss: 0.3880 - acc: 0.8582
Epoch 3/6
65996/65996 [================] - 4s 67us/sample - loss: 0.3581 - acc: 0.8677
Epoch 4/6
65996/65996 [================] - 5s 69us/sample - loss: 0.3310 - acc: 0.8763
Epoch 5/6
65996/65996 [================] - 4s 58us/sample - loss: 0.3141 - acc: 0.8839
Epoch 6/6
65996/65996 [================] - 4s 64us/sample - loss: 0.3016 - acc: 0.8881
Out[29]:
<tensorflow.python.keras.callbacks.History at 0x181239196a0>
```

Первое, что мы проверим, — это то, насколько хорошо справляется наша новая, прошедшая вредоносное обучение модель с распознаванием исходных тестовых данных по сравнению с исходной моделью:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Original model accuracy based on nonadversarial test data:', test_acc)
test_loss, test_acc = model_adv.evaluate(test_images, test_labels)
print('Adversarially trained model accuracy based on nonadversarial test data:',
       test_acc)
```

Данный код сгенерирует следующий вывод:

```
10000/10000 [================] - 0s 37us/sample - loss: 0.3591 - acc: 0.8699
Original model accuracy based on nonadversarial test data: 0.8699
10000/10000 [================] - 1s 53us/sample - loss: 0.3555 - acc: 0.8707
Adversarially trained model accuracy based on nonadversarial test data: 0.8707
```

Неплохо! Новая модель справляется с этой задачей даже немного лучше исходной модели, то есть, как видим, обучение не привело к снижению точности.

Теперь нам потребуются некоторые вредоносные тестовые данные. Для надлежащей оценки модели мы должны предполагать, что злоумышленник обладает полной осведомленностью о методе защиты, а следовательно, и о прошедшей вредоносное обучение модели. Однако сначала для сравнения применим простейший подход и создадим тестовые данные с помощью исходной модели (не подвергавшейся вредоносному обучению).

Определим атаку:

```
import foolbox
fmodel = foolbox.models.TensorFlowModel.from_keras(model, bounds=(0, 1))
attack_criterion = foolbox.criteria.Misclassification()
x_images = test_images[0:600, :]

attack_fn = foolbox.attacks.GradientSignAttack(fmodel,
                            criterion=attack_criterion,
                            distance=foolbox.distances.Linfinity)
```

Теперь сгенерируем набор тестовых данных. Назовем его `x_test_adv_images1`:

```
(x_test_adv_images1, x_test_adv_perturbs1, x_test_labels1) =
                     generate_adversarial_data(original_images = x_images,
                               predictions = model.predict(x_images),
                               attack_fn = attack_fn)
```

Посмотрим на матрицу несоответствий, показывающую, насколько хорошо модель, подвергнутая вредоносному обучению, справляется

с распознаванием вредоносных образов, созданных с помощью исходной модели (рис. 10.3):

```
show_confusion_matrix(model_adv, x_test_adv_images1, x_test_labels1, class_names)
```

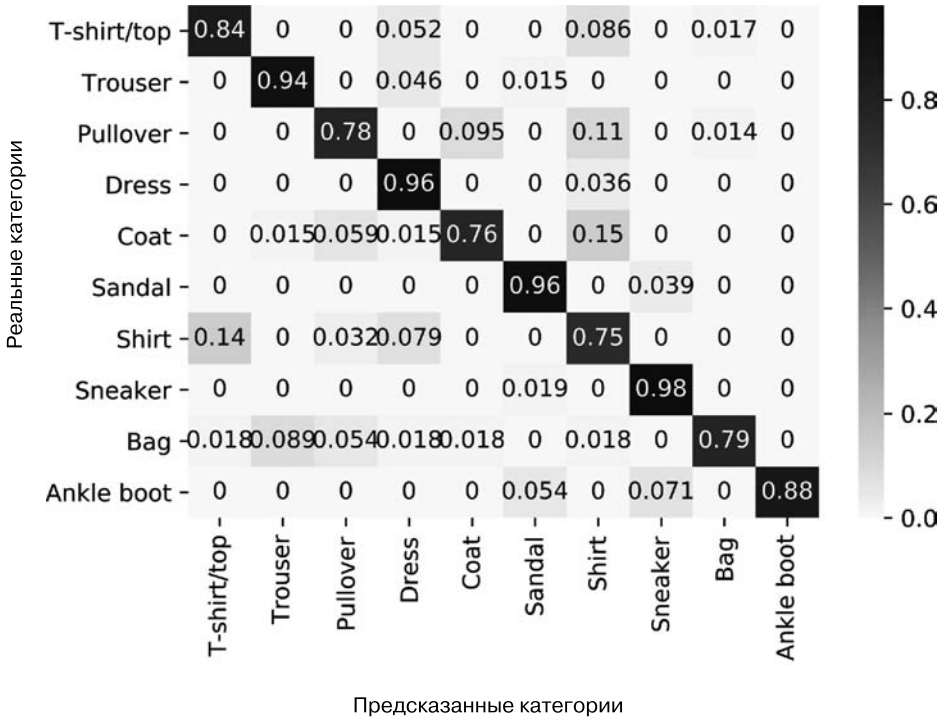


Рис. 10.3. Вывод программы

Выглядит вполне неплохо — модель правильно классифицировала большую часть вредоносных тестовых данных.

К сожалению, такая оценка не вполне корректна. Во-первых, мы проверяли только устойчивость модели к определенному виду атаки (см. примечание далее). Во-вторых, для надлежащей оценки модели мы должны предполагать полную осведомленность злоумышленника о модели и ее механизмах защиты. Имея полную осведомленность, злоумышленник сможет генерировать вредоносные данные непосредственно с помощью модели, подвергнутой вредоносному обучению, и именно этот сценарий мы должны рассматривать.

Тестирование модели, подвергнутой вредоносному обучению с помощью различных видов атак

Воспользуйтесь соответствующим блокнотом Jupyter, если вы хотите продолжить эксперименты с вредоносным обучением, используя различные методы атаки для создания обучающих данных.

Вы также можете проверять созданную вами модель на тестовых данных, созданных с использованием другого метода атаки. В случаях, когда тестовый метод атаки не будет аналогичен методу, примененному для обучения модели, итоговая матрица несоответствий будет, как правило, содержать уже не столь хорошие результаты классификации, как в случае, показанном на рис. 10.3.

Прежде всего необходимо заново сгенерировать тестовые данные с помощью модели, подвергнутой вредоносному обучению. Назовем этот набор данных `x_test_adv_images2`:

```
fmodel_adv = foolbox.models.TensorFlowModel.from_keras(model_adv,
                           bounds=(0, 1)) ❶
attack_fn = foolbox.attacks.GradientSignAttack(fmodel_adv,
                           criterion=attack_criterion,
                           distance=foolbox.distances.Linfinity)

(x_test_adv_images2, x_test_adv_perturbs2, x_test_labels2) =
                     generate_adversarial_data(original_images = x_images,
                              predictions = model_adv.predict(x_images), ❶
                              attack_fn = attack_fn)
```

❶ Обратите внимание, что в этих двух строках указана модель, подвергнутая вредоносному обучению.

Данный код сгенерирует следующий вывод:

```
Warning: Unable to find adversarial example for image at index:  76
```

Итоговая матрица несоответствий представлена на рис. 10.4:

```
show_confusion_matrix(model_adv, x_test_adv_images2, x_test_labels2, class_names)
```

Как видите, на этот раз модель гораздо хуже справилась со своей задачей и не смогла правильно классифицировать ни один из вредоносных образов (на это указывают нули в диагонали, идущей из верхнего левого угла в правый нижний угол). Это и неудивительно, поскольку образы в наборе `x_test_adv_images2` создавались с целью обмануть подвергнутую вредоносному обучению модель.

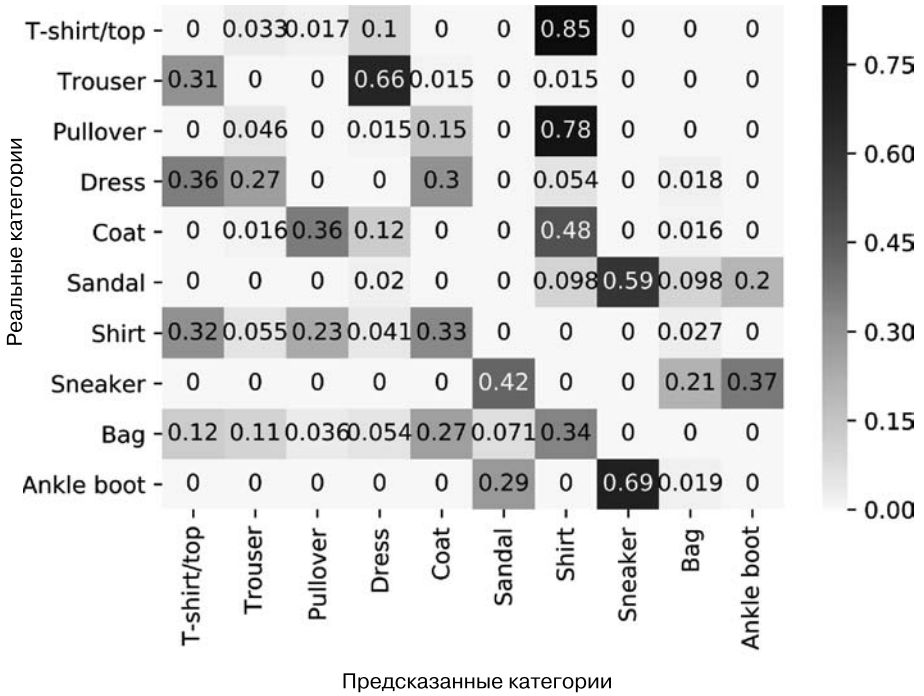


Рис. 10.4. Вывод программы

Теперь посмотрим, насколько сложнее злоумышленнику создать вредоносные образы для модели, подвергнутой вредоносному обучению. Для этого мы построим график зависимости степени успешности от требуемой величины искажения.

Вспомогательный метод `generate_adversarial_data` возвращает показатель расстояния для каждого из полученных нами вредоносных образов. Предполагая, что метод знаковой градиентной атаки `GradientSignAttack` стремится минимизировать расстояния, эти показатели отражают минимальные расстояния, необходимые для каждого вредоносного образа.

Представим на графике искажения, необходимые для создания вредоносных образов для исходной модели и модели, подвергнутой вредоносному обучению:

```
plt.hist((x_test_adv_perturbs1['foolbox_diff'], ❶
          x_test_adv_perturbs2['foolbox_diff']), ❷
          bins=20, ❸
```

```
        cumulative=True, ❹
        label=('Original model','Adversarially trained model'))

plt.title("Adversarial example success rate")
plt.xlabel("Perturbation")
plt.ylabel("Number of successful adversarial examples")
plt.legend(loc='right')

plt.show()
```

❶ Здесь передается список показателей величины искажения (в данном случае представляющих собой L^∞-норму) для всех вредоносных образов, полученных с использованием исходной модели.

❷ Здесь передается список показателей величины искажения (в данном случае представляющих собой L^∞-норму) для всех вредоносных образов, полученных с использованием модели, подвергнутой вредоносному обучению.

❸ Здесь задается количество столбцов гистограммы.

❹ Здесь задается вид гистограммы — с накоплением.

Построенный график показан на рис. 10.5.

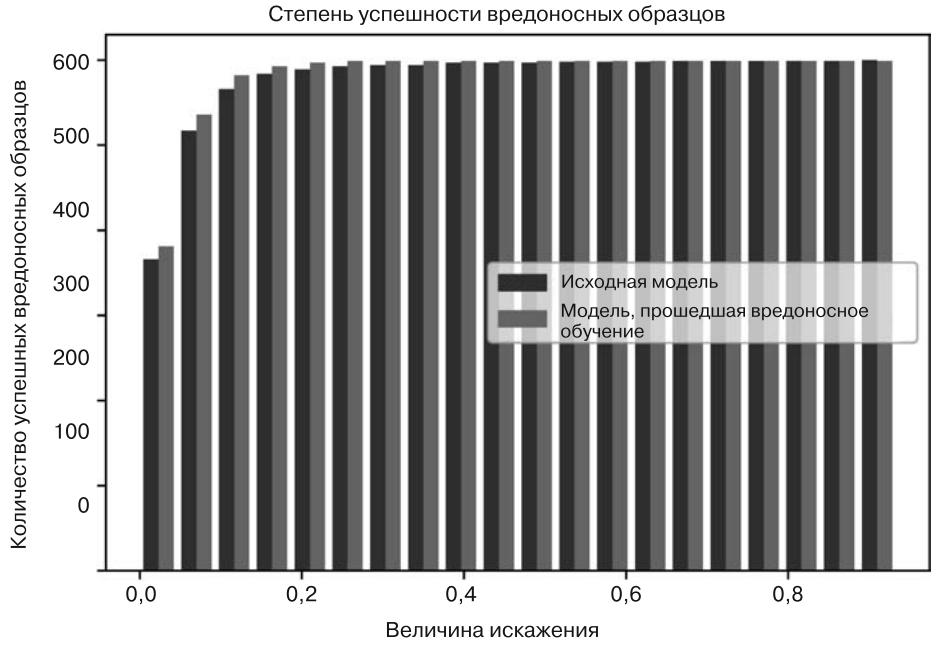


Рис. 10.5. Вывод программы

Какое разочарование! Как видно на графике, сеть, подвергнутая вредоносному обучению, имеет практически *такую же* зависимость степени успешности вредоносной атаки от величины искажения, как и исходная сеть. То есть вредоносное обучение нисколько не затруднило создание вредоносных образов, используя тот же метод атаки.

Чтобы понять, почему так происходит, посмотрим еще раз на ландшафт предсказаний. Для того чтобы получить вредоносный образ, злоумышленник может использовать тысячи направлений изменения исходного изображения. И хотя вредоносное обучение сети перекрывает некоторые из этих направлений, вредоносному алгоритму остается доступным еще очень большое их количество. Мы можем устранить еще большее число вариантов, увеличив масштабы вредоносного обучения. Однако это все равно не позволит полностью перекрыть доступные вредоносному алгоритму возможности (хотя, скорее всего, и сделает чуть более продолжительным создание вредоносных входных данных).

Если вы внимательно посмотрите на столбики, построенные для низких значений искажения, то заметите, что подвергнутая вредоносному обучению сеть имеет более высокую степень успешности по сравнению с исходной сетью. Это говорит о том, что алгоритм обеспечивает не самые лучшие результаты для исходной модели, то есть не всегда возвращает вредоносный образ с минимально возможным искажением. Это, вероятно, объясняется тем, что каждая итерация градиентной атаки определяется характером градиентов ландшафта предсказаний в точке расположения текущего изображения. Если характер градиентов сильно меняется по мере удаления от изображения, алгоритм может сделать шаг в направлении, не обеспечивающем оптимальный результат.

OoD-обучение

Как мы уже видели, иногда глубокие нейронные сети с высокой степенью уверенности выдают неправильные результаты в силу своей неспособности правильно обобщить все возможные варианты входных данных. При затруднении с обобщением данных ГНС часто относит их к широкой категории *данных вне распределения*, подразумевающей те входные данные, которые находятся за пределами распределения обучающих данных и, как следствие, не относятся к числу входных данных, обрабатываемых сетью с высокой степенью надежности.

В главе 5 уже говорилось о том, что представляют собой данные вне распределения. Эту концепцию иллюстрирует рис. 10.6. В левой части этого рисунка показано распределение во входном пространстве обучающих данных, принадлежащих к трем категориям. При проверке полностью обученной модели на изображенных справа тестовых данных она покажет хорошие результаты для тех данных, которые распределены аналогично обучающим данным. Однако при этом нет гарантии, что она выдаст правильный результат для сигнала, находящегося за пределами этого распределения.

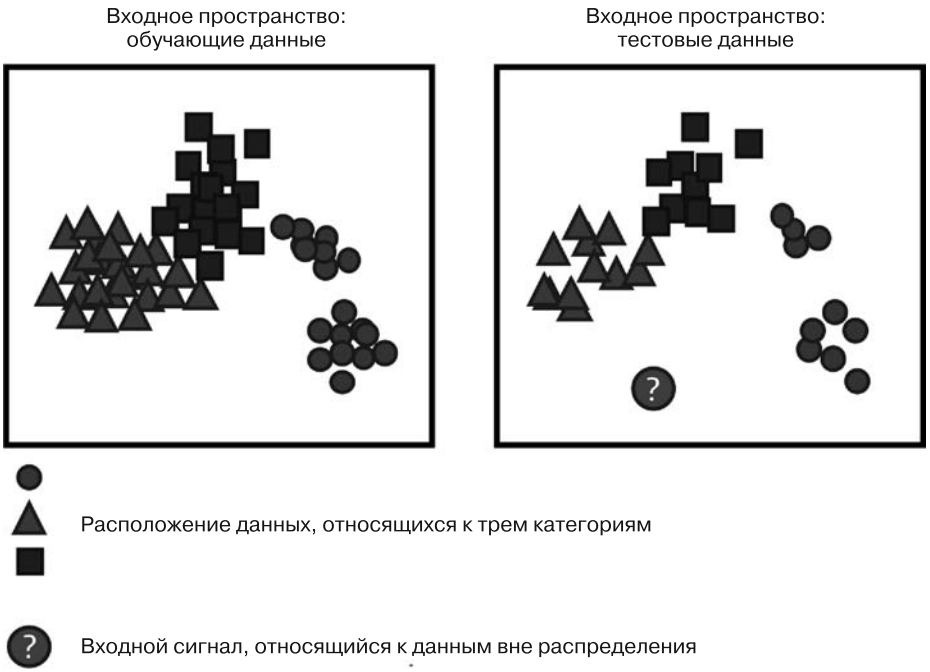
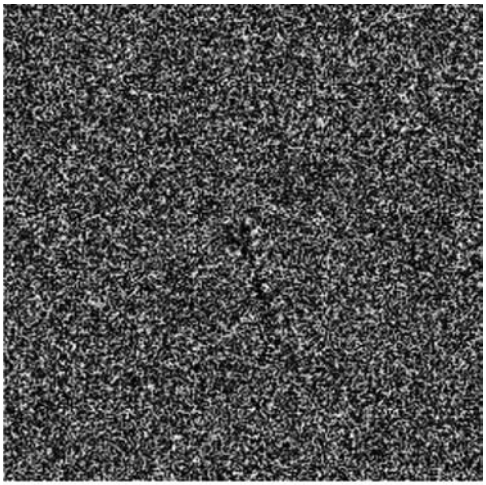


Рис. 10.6. Тестовые данные *(справа)* распределены аналогично обучающим данным *(слева)*; исключение составляет лишь один сигнал, относящийся к данным вне распределения

Входные сигналы, относящиеся к данным вне распределения, не всегда являются вредоносными. Иногда они представляют собой граничный случай, который не был отражен в обучающем наборе данных. Они также могут представлять собой какое-то нереалистичное или бессмысленное изображение. Проблема с входными данными вне распределения заключается в том, что сеть может по-прежнему возвращать для них уверенное предсказание,

несмотря на то что точность этого предсказания будет ниже, чем в случае входных данных, имеющих сходство с данными обучающего набора.

Этот случай иллюстрируют представленные в главе 1 примеры находящихся вне распределения вредоносных образов, которые не имеют никакого сходства с реалистичными изображениями, но с высокой степенью уверенности относятся сетью к определенной категории. Чтобы напомнить это, на рис. 10.7 еще раз представлен один из таких вредоносных образов.



Гепард

Рис. 10.7. Этот сгенерированный в цифровом виде вредоносный образ относится к данным вне распределения, но, несмотря на это, классификатор изображений с высокой степенью уверенности относит его к категории «гепард» (изображение из статьи Нгуена и др. от 2015 года)

С другой стороны, несправедливо и утверждение о том, что вредоносные образы всегда являются данными вне распределения. Хотя вредоносные образы могут находиться в областях входного пространства, относящихся к данным вне распределения (подобно входному сигналу, показанному на рис. 10.6), они также могут использовать и те точки входного пространства, в которых алгоритму не удалось выполнить надлежащее обобщение. Эти точки могут по-прежнему находиться в том же распределении, что и обучающие данные. Оба эти случая показаны на рис. 10.8, где представлен ландшафт предсказаний, полученный на основе некоторого набора обучающих данных. В то время как вредоносный образ 1 явно относится к числу данных вне распределения, вредоносный образ 2 находится в пределах распределения

обучающих данных, но в такой точке, где алгоритму не удалось обеспечить правильное обобщение.

Если вы научитесь распознавать входные сигналы, относящиеся к данным вне распределения, это не обеспечит вам гарантированное выявление всех вредоносных образов. В то же время сопоставление с каждым из предсказаний модели показателя степени уверенности дает дополнительные преимущества. Любой метод оценки того распределения данных, в пределах которого сеть демонстрирует хорошие результаты, делает возможным проверку входных данных на предмет их принадлежности к «безопасному» распределению.

Существует ряд методов, позволяющих проверять изображения относительно их реалистичности, например, путем выявления высокого контраста между соседними пикселями. Эти методы могут успешно перехватывать такие «явные» данные вне распределения, как представленное на рис. 10.7 изображение гепарда, но уже гораздо хуже справляются с выявлением изображений вне распределения, обладающих более четко выраженными формами и узорами.

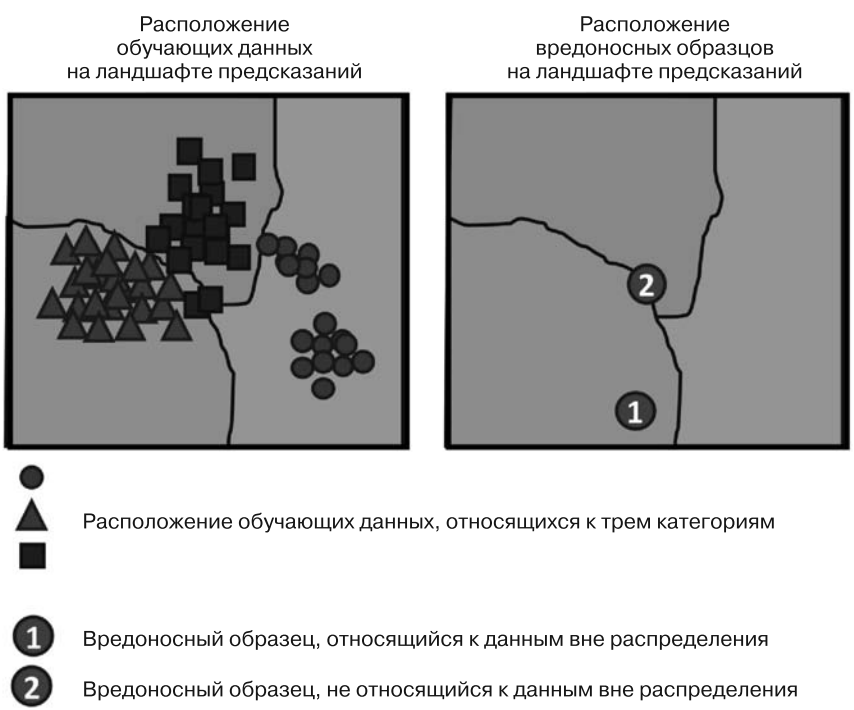


Рис. 10.8. Вредоносные образы не всегда являются данными вне распределения

Хотя данные вне распределения трудно поддаются выявлению путем статистического анализа за пределами сети, были предложены другие методы, позволяющие делать это с помощью самой сети. Один из перспективных подходов сводится к тому, чтобы заставить сеть калибровать свои результаты (указывая для них меньшую степень уверенности) или помечать входные сигналы, отнесенные к категории данных вне распределения. Для этого нужно научить ГНС выдавать не только полученные предсказания, но и показатели *степени уверенности* в каждом из этих предсказаний. Так, например, на рис. 10.9 показано, как этот подход может быть встроен в архитектуру ГНС, рассмотренную в подразделе «ГНС для обработки изображений» на с. 80[1].

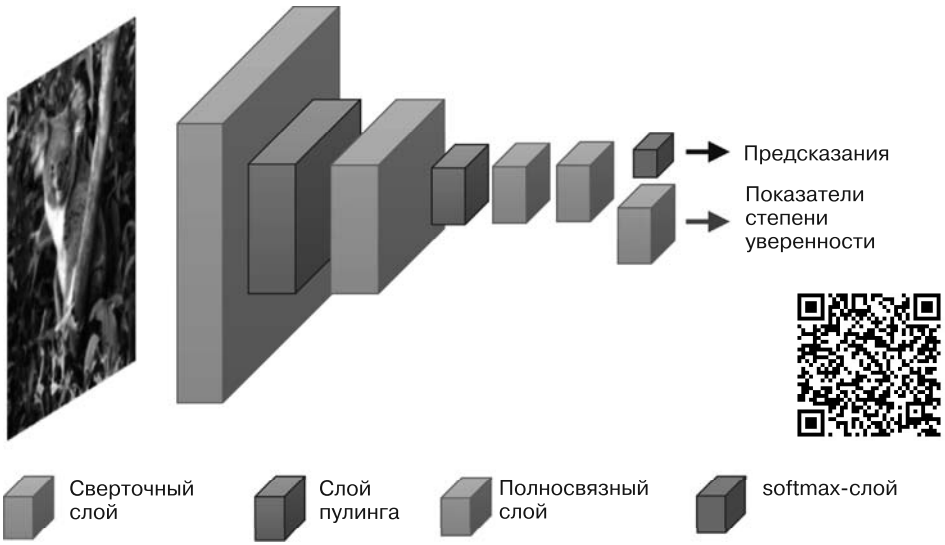


Рис. 10.9. Дополнение архитектуры СНС расчетом показателей степени уверенности для предсказаний

На этапе обучения сеть учится не только непосредственно делать предсказания, но и оценивать степень уверенности в своих предсказаниях.

Используемая при обучении функция штрафа, которую мы рассмотрели в главе 3, может быть переопределена заново с тем, чтобы оптимизировать и точность показателей степени уверенности, и точность самих предсказаний. То есть при получении неточных предсказаний для тренировочного примера обучаемая сеть должна возвращать низкий показатель степени

[1] *Vries T. De, Taylor G. W.* Learning Confidence for Out-of-Distribution Detection in Neural Networks, 2018. http://bit.ly/2XZHpH1.

уверенности. Соответственно, сеть штрафуется на этапе обучения (путем повышения штрафов) в случае возвращения высокого показателя степени уверенности для такого образа. Точно таким же образом скорректированная функция штрафа будет штрафовать и низкий показатель степени уверенности для неверного предсказания. Хорошо обученной считается сеть, которая не только выдает предсказания, достаточно точно соответствующие целевым меткам, но и точно оценивает степень уверенности для каждого обучающего сигнала.

Эта дополнительная обратная связь от сети показывает, принадлежит ли входной сигнал к тому распределению данных, в пределах которого сеть способна выдавать надежные результаты. Так, например, даже если для изображения на рис. 10.7 будет получен высокий вероятностный показатель принадлежности к категории «гепард», правильно работающая сеть выдаст низкий показатель степени уверенности в этой оценке. На момент написания книги исследования в этой области находились еще на стадии зарождения, но, судя по всему, результаты будут многообещающими.

Оценка неопределенности случайного отсева

Ряд методов под общим названием *«регуляризация»* используется на этапе обучения нейронной сети для снижения вероятности ее «переобучения», то есть слишком сильной ее адаптации под обучающие данные. Эти методы уменьшают степень сложности нейронной сети, делая ее более обобщенной, что позволяет применять ее в более широком диапазоне данных. *Отсев* — один из методов регуляризации — случайным образом «удаляет» нейроны, чтобы они не учитывались в итерациях обучения, и возвращает их на место перед последующими итерациями. На первый взгляд, в этом нет логики, ведь такой подход должен усложнить процесс обучения сети. И действительно так, но именно в этом и заключается смысл метода — он не дает сети слишком сильно полагаться на определенные единицы данных при принятии решения и тем самым повышает степень ее универсальности.

Отсев может использоваться не только для того, чтобы не допустить переобучения модели на этапе обучения, но и для того, чтобы после введения модели в эксплуатацию она полагалась на разные единицы данных при получении каждого запроса. Это вносит в сеть неопределенность, которая препятствует получению детерминированных предсказаний для определенного входного сигнала. Сеть будет выдавать разные результаты для запросов с одинаковым входным сигналом, и величина разброса этих результатов будет отражать степень неопределенности сети для ее конкретного экземпляра.

В своем исследовании Фейнман и др.[1] предложили метод выявления вредоносных атак с помощью так называемой *байесовской неопределенности нейронной сети*, при котором вредоносные образы выявляются за счет того, что рандомизированная сеть выдает для них меньший показатель степени уверенности, чем для натуральных примеров. Этот способ защиты базируется на предположении, что предсказания, выдаваемые для невредоносных входных данных, носят более единообразный характер, чем предсказания, выдаваемые для вредоносных входных данных. То есть если в сеть, подвергнутую случайному отсеву, раз за разом подавать один и тот же входной сигнал, то величина разброса получаемых предсказаний будет более значительной, если этот сигнал является вредоносным. Если величина разброса для конкретного входного сигнала превышает заданный порог, то этот входной сигнал считается вредоносным. Этот подход демонстрирует многообещающие результаты в выявлении вредоносных входных данных, даже когда предполагается, что злоумышленник осведомлен об используемом методе защиты[2].

Пример кода: отсев для выявления вредоносных атак

Приведенные в данном разделе фрагменты кода представляют собой выдержки из блокнота Jupyter chapter10/fashionMNIST_dropout_for_detection.ipynb (http://bit.ly/2XZw7T8), включенного в GitHub-репозиторий этой книги.

Воспользовавшись этим блокнотом, вы можете выполнить ряд дополнительных экспериментов со случайным отсевом, например попробовать изменить параметры отсева или применить различные методы атаки.

Вместо последовательного API библиотеки Keras, который использовался в этой книге до сих пор, здесь будет задействован функциональный API той же библиотеки, поскольку это позволит нам подвергнуть сеть случайному отсеву после ее обучения[3].

---

1 *Feinman R. et al.* Detecting Adversarial Samples from Artifacts, 2017. http://bit.ly/2XpavTe.

2 *Carlini N., Wagner D.* Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods, 2017. http://bit.ly/2WTMhBe.

3 Подробнее о том, чем отличаются эти два способа программирования, можно прочитать в документации библиотеки Keras по модели Sequential и в руководстве по функциональному API.

Сначала мы создадим такой же классификатор Fashion-MNIST, какой мы создавали ранее, но разрешим при этом отсев в одном из скрытых слоев:

```
from tensorflow.keras.layers import Input, Dense, Flatten, Dropout
from tensorflow.keras.models import Model

inputs = Input(shape=(28,28))
x = Flatten()(inputs)
x = Dense(56, activation='relu')(x)
x = Dropout(0.2)(x, training=True) ❶
x = Dense(56, activation='relu')(x)
predictions = Dense(10, activation='softmax')(x)

model = Model(inputs=inputs, outputs=predictions)

model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

❶ В этой строке добавляется отсев. Параметр `training=True` указывает (не слишком очевидным образом), что отсев должен применяться не только во время обучения, но и *после* него. Это придает неопределенность предсказаниям сети. Степень этой неопределенности определяется параметром, передаваемым функции `Dropout`. При желании вы можете воспользоваться соответствующим блокнотом Jupyter и посмотреть, к чему может привести изменение этой степени неопределенности.

Данный код сгенерирует следующий вывод:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         (None, 28, 28)            0
_________________________________________________________________
flatten_4 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_12 (Dense)             (None, 56)                43960
_________________________________________________________________
dropout_4 (Dropout)          (None, 56)                0     ❶
_________________________________________________________________
dense_13 (Dense)             (None, 56)                3192
_________________________________________________________________
dense_14 (Dense)             (None, 10)                570
=================================================================
Total params: 47,722
Trainable params: 47,722
Non-trainable params: 0
_________________________________________________________________
```

❶ Это дополнительный слой отсева.

Теперь выполним обучение модели и проверим ее точность на тестовых данных из набора Fashion-MNIST:

```
model.fit(train_images, train_labels, epochs=12) ❶
```

❶ Модели со слоем отсева требуется больше эпох на этапе обучения для обеспечения такого же уровня точности. В силу этого параметру `epochs` присвоено более высокое значение, чем в предыдущих примерах.

Данный код сгенерирует следующий вывод:

```
Epoch 1/12
60000/60000 [================] - 4s 63us/sample - loss: 0.3243 - acc: 0.8777
Epoch 2/12
60000/60000 [================] - 4s 63us/sample - loss: 0.3174 - acc: 0.8812
Epoch 3/12
60000/60000 [================] - 4s 61us/sample - loss: 0.3119 - acc: 0.8834
Epoch 4/12
60000/60000 [================] - 4s 61us/sample - loss: 0.3114 - acc: 0.8845
Epoch 5/12
60000/60000 [================] - 4s 63us/sample - loss: 0.3042 - acc: 0.8854
Epoch 6/12
60000/60000 [================] - 4s 61us/sample - loss: 0.2987 - acc: 0.8882
Epoch 7/12
60000/60000 [================] - 4s 61us/sample - loss: 0.2982 - acc: 0.8870
Epoch 8/12
60000/60000 [================] - 3s 53us/sample - loss: 0.2959 - acc: 0.8889
Epoch 9/12
60000/60000 [================] - 4s 61us/sample - loss: 0.2931 - acc: 0.8902
Epoch 10/12
60000/60000 [================] - 4s 63us/sample - loss: 0.2894 - acc: 0.8909
Epoch 11/12
60000/60000 [================] - 3s 53us/sample - loss: 0.2859 - acc: 0.8919
Epoch 12/12
60000/60000 [================] - 4s 66us/sample - loss: 0.2831 - acc: 0.8927
Out[11]:
<tensorflow.python.keras.callbacks.History at 0x156074c26d8>
```

Проверим точность сети:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Model accuracy based on test data:', test_acc)
```

Данный код выдаст следующий результат:

```
10000/10000 [================] - 0s 40us/sample - loss: 0.3827 - acc: 0.8689
Model accuracy based on test data: 0.8689
```

Если мы попробуем заново запустить эту ячейку блокнота Jupyter, то в силу присутствия в сети неопределенности будем получать разное значение точности каждый раз.

Теперь нам потребуется набор невредоносных изображений и набор вредоносных изображений. Мы можем использовать те тестовые данные, которые предоставляются вместе с набором Fashion-MNIST, и те вредоносные изображения, которые мы генерировали ранее. Сгенерируем для этого примера изображения путем применения против исходной модели метода быстрой градиентной атаки `FastGradient` из библиотеки Foolbox:

```
import numpy as np

num_images = 1000
x_images = test_images[:num_images]
x_images_adv = .... ❶
```

❶ Сгенерируем вредоносные изображения с помощью вспомогательной утилиты `generate_adversarial_data`, как это делалось ранее. Этот код не приводится здесь для краткости.

Теперь сгенерируем некоторое множество предсказаний, используя подвергнутую отсеву модель для обоих наборов данных. При этом каждое изображение будет передано модели `L` раз:

```
L = 100
num_classes = 10

predictions_matrix = np.zeros((L, num_images, num_classes)) ❶
predictions_matrix_adv = np.zeros((L, num_images, num_classes)) ❷

for i in range(L):
    predictions = model.predict(x_images)
    predictions_adv = model.predict(x_images_adv)
    predictions_matrix[i] = predictions
    predictions_matrix_adv[i] = predictions_adv
```

❶ `predictions_matrix` — матрица предсказаний, полученных для `L` случаев передачи модели всех невредоносных изображений.

❷ `predictions_matrix_adv` — матрица предсказаний, полученных для `L` случаев передачи модели всех вредоносных изображений.

Теперь вычислим для каждого изображения отдельное значение степени неопределенности, отражающее величину разброса предсказаний, полученных

для этого изображения. Следующая функция вычисляет степень неопределенности набора предсказаний, полученных для отдельного изображения:

```
def uncertainty(predictions):
    return(np.sum(np.square(predictions))/predictions.shape[0]
           - np.sum(np.square(np.mean(predictions, axis=0)))) ❶

uncertainty_results = np.zeros((num_images))
uncertainty_results_adv = np.zeros((num_images))

for i in range(num_images): ❷
    uncertainty_results[i] = uncertainty(predictions_matrix[:,i])
    uncertainty_results_adv[i] = uncertainty(predictions_matrix_adv[:,i])
```

❶ Этот код реализует математическое определение метода Карлини и Вагнера 2017 года:

$$U(x) = \left(\frac{1}{L}\sum_{i=1} \|F_r(x)\|\right) - \left\|\frac{1}{L}\sum_{i=1} F_r(x)\right\|,$$

где $\|y\|$ принимается равным квадрату L^2-нормы. Это просто метод расчета показателя разброса для набора предсказаний.

❷ Расчет показателя неопределенности для каждого изображения, дающий в итоге два списка — один с показателями неопределенности, относящимися к каждому из невредоносных изображений, и один с показателями неопределенности, относящимися к вредоносным изображениям.

Наконец, представим полученные результаты в виде графика:

```
import matplotlib.pyplot as plt

plt.hist((uncertainty_results, uncertainty_results_adv),
           bins=50,
           label=('Nonadversarial','Adversarial'))

plt.title("Model prediction uncertainty")
plt.xlabel("Uncertainty per image")
plt.ylabel("Number of images")
plt.legend(loc='right')

plt.show()
```

Построенный нами график показан на рис. 10.10.

Как видите, возвращаемые моделью предсказания, как правило, имеют большую степень неопределенности в случае вредоносных изображений, то есть все сработало как надо. В данном случае у нас нет четко очерченного порога, по которому можно было бы провести границу между вредоносными

и нормальными данными[1]. Такое пороговое значение можно найти с помощью ROC-кривой, как описано в следующем примечании.

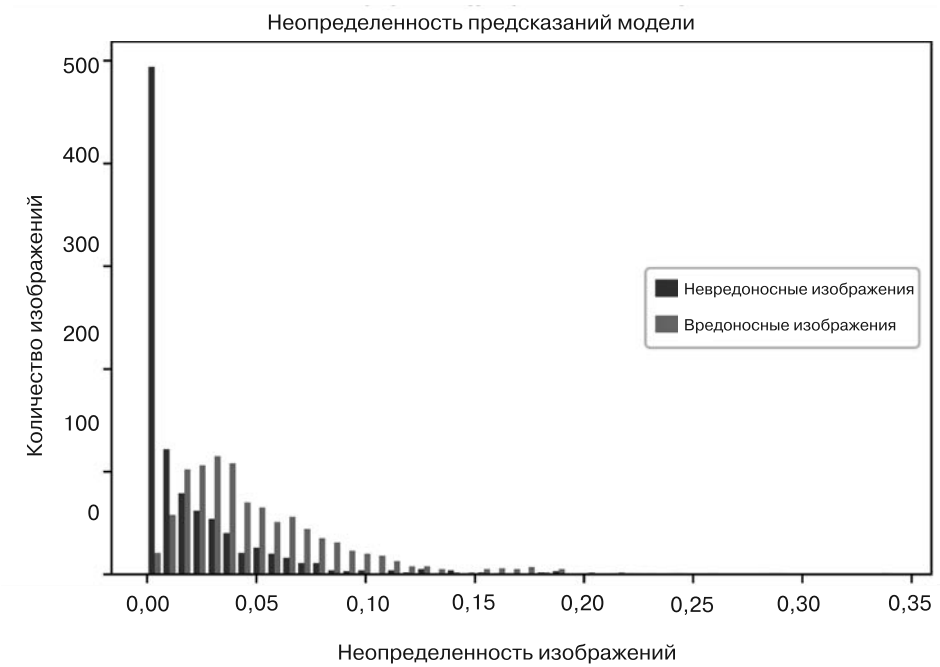


Рис. 10.10. Вывод программы

ROC-кривые

Одним из важных инструментов машинного обучения являются ROC-кривые (receiver operating characteristic, ROC, *рабочая характеристика приемника*), которые часто используются для определения пороговой величины предсказаний, достаточной для принятия некоторого бинарного решения. ROC-кривая отображает соотношение между количеством истинно положительных и ложноположительных результатов в *пространстве рабочей характеристики приемника*. Этот график позволяет найти пороговое значение путем сравнения количества истинно

[1] В вышеупомянутом исследовании данный метод сработал еще лучше, обеспечив более четкое различие между нормальными и вредоносными данными. Это, вероятно, объясняется использованием более точных моделей по сравнению с нашим простейшим классификатором.

положительных (требуемых) результатов с количеством ложноположительных результатов (штрафов).

В случае защиты от вредоносных данных таким бинарным порогом может быть вероятность принадлежности входного сигнала к вредоносным данным, достаточная для фактического отнесения его к таковым. Например, если защитный механизм относит входные данные к вредоносным, когда степень уверенности в этом составляет 50 %, какой при этом будет доля ложноположительных результатов? То есть сколько невредоносных входных данных будет ошибочно относиться к числу вредоносных при таком пороге? Приемлемое соотношение между ложноположительными и ложноотрицательными результатами зависит от конкретного случая — стоит ли обеспечить перехват вредоносных входных данных даже ценой снижения точности для невредоносных данных или важнее не допустить ошибочной классификации невредоносных данных? ROC-кривая позволяет сформулировать это в ходе оценки модели и таким образом определить пороговые значения для конкретных сценариев.

При желании вы можете воспользоваться блокнотом Jupyter и поэкспериментировать с другими видами атаки или с вредоносными данными, полученными непосредственно с помощью рандомизированной модели (с применением отсева только во время обучения). Если вы попробуете сгенерировать вредоносные образы с помощью модели, подвергаемой отсеву после обучения, это приведет к интересным результатам, поскольку на эффективности атаки будет плохо сказываться постоянное изменение ландшафта предсказаний.

Модель, использующая отсев для выявления вредоносных входных данных, может применяться в дополнение к не подвергнутой отсеву рабочей модели, чтобы гарантировать детерминированность функционирования системы.

Предварительная обработка данных

Теперь применим другой подход и попробуем удалить вредоносные данные в общей цепи обработки *до* их поступления в ГНС.

При этом мы сфокусируемся на двух областях.

- *Предварительная обработка в общей последовательности обработки.* Сначала мы посмотрим, какие побочные эффекты может оказывать на вредоносные входные данные предварительная обработка в общей последовательности обработки.

- *Интеллектуальное удаление вредоносного контента.* Затем мы узнаем, существуют ли какие-либо надежные статистические методы, позволяющие намеренно удалять вредоносный контент до его поступления в ГНС — либо путем непосредственного выявления вредоносных входных данных, либо посредством избавления всех данных от аспектов, способных заставить модель возвращать неверный результат.

Предварительная обработка в общей последовательности обработки

В реально существующих приложениях глубокие нейронные сети не используются отдельно, без каких-либо других компонентов. В силу влияния общей последовательности обработки эффективность вредоносного образа определяется гораздо большим количеством факторов при его тестировании во внелабораторных условиях. При этом злоумышленник не может быть уверен в том, что его тщательно разработанные вредоносные образы не будут выявлены последовательностью обработки целевой системы или превращены ею (умышленно или неумышленно) в неопасные данные.

В рамках общей последовательности обработки решения по информационной безопасности обеспечивают защиту от атак путем автоматического выявления данных с возможным содержанием вредоносного контента. Обычно это реализуется путем использования брандмауэров и антивирусных программ, определяющих степень риска на основе происхождения данных (получены ли они из надежного источника) или характера данных (имеют ли они характерные признаки вредоносного контента).

Глубокие нейронные сети для обработки изображений, аудио- и видеоданных часто берут данные непосредственно из физического мира (например, с помощью камер), или данные поступают из ненадежных цифровых источников (например, загружены пользователями на сервер), что, к сожалению, делает невозможным выявление вредоносных образов из-за ненадежности источника. Выявление на основе характера данных тоже представляется затруднительным, поскольку вредоносные образы не воспринимаются как опасные данные целевой системой.

После прохождения через брандмауэр или другие защитные преграды данные подвергаются организационной обработке. Здесь следует рассмотреть два типичных сценария.

- *Пример: фильтрация загрузки изображений в социальную сеть.* Последовательность обработки для загрузки изображений или видеозаписей на сайт социальной сети может включать в себя несколько этапов

предварительной обработки изображений перед их векторизацией и передачей в нейронную сеть. Это могут быть такие преобразования, как сжатие, нормализация, сглаживание и изменение размеров.

- *Пример: виртуальный помощник.* В данном сценарии входной сигнал также подвергается некоторым ограничениям или предварительной обработке во входном датчике (в данном случае в микрофоне), что часто ведет к появлению шумов или к фильтрации захватываемых данных[1]. После этого сигнал проходит дополнительную обработку, например подвергается преобразованию Фурье, перед его передачей в ГНС.

Поскольку при создании вредоносных входных данных часто учитывается степень точности данных, преобразование, ведущее к потере точности вредоносного входного сигнала, может сделать его невредоносным.

Посмотрим, как может сказываться на вредоносных образах снижение точности воспроизведения изображений. В главе 4 мы определили точность воспроизведения изображения как сочетание его пространственного разрешения и пиксельной плотности. Когда вредоносный образ создается путем изменения небольшого числа пикселей (вероятно, с минимизацией L^0-нормы), на его эффективности сказывается снижение пространственного разрешения, поскольку уменьшение пиксельной плотности ведет к потере измененных пикселей. С другой стороны, когда вредоносный образ создается путем внесения в изображение большого количества мелких изменений (вероятно, с минимизацией L^∞-нормы), на его эффективности сказывается снижение цветового разрешения, ведущее к потере мелких вариаций цвета. Таким образом, если злоумышленник, не зная того, применит во вредоносном сигнале характеристики, которые теряются или округляются в последовательности обработки, это сделает его атаку менее надежной.

Существует целый ряд оснований для сокращения объема данных на этапе предварительной обработки, включая, помимо прочего, следующее.

- *Нормализация и сжатие данных.* В последовательности обработки может выполняться этап нормализации (например, преобразование данных в единый формат или изменение размеров изображений перед их поступлением в ГНС). Если этот этап нормализации ведет к сокращению объема данных, то он может удалять информацию, делающую входные данные вредоносными.

---

[1] Ряд других проблем, накладываемых физическим окружением, рассмотрен в главе 8.

Важной составляющей этапа цифрового кодирования также является сжатие данных, которое может влиять на вредоносные входные данные в том случае, если это ведет к снижению точности (см. примечание далее).

Точность любой цифровой информации ограничена минимальной точностью последней операции обработки. Изображение, сохраненное с разрешением 640 × 480 пикселей, минимально достаточным для отображения на дисплеях, будет всегда иметь это разрешение; мы не сможем повысить уровень его пространственной детализации, если отобразим его на экране телевизора с поддержкой формата HD, или сохраним его в виде файла с более высоким разрешением[1].

- *Удаление шумов и посторонних данных*. В последовательности обработки может производиться удаление шумов или посторонних данных из исходных данных, если это облегчает обработку данных или делает их более доступными для человеческого восприятия.

  Под *шумами* понимаются те искажения, которые часто вносятся в данные на этапе их захвата. Визуальные искажения изображений могут проявляться в виде пикселей, не обеспечивающих точное отображение сцены. Так, например, на изображении, снятом в условиях слабой освещенности, может быть большое количество пятен. Звуковые шумы могут проявляться в виде треска и помех, вносимых датчиком микрофона или звуковым оборудованием, реверберации, эха и фонового шума.

Для удаления шумов из изображений широко используется так называемое *размытие по Гауссу*. Такая операция размытия может выполняться безотносительно к ГНС для очистки изображений в рамках предварительной обработки в системе. При этом последовательностью обработки будут удаляться все вредоносные искажения или заплатки, относимые ею к числу шумов или посторонних данных.

В ходе предварительной обработки также могут удаляться и другие посторонние данные. Так, например, системы обработки речи могут применять особые методы обработки звука для извлечения актуальных аспектов звука,

---

[1] Хотя ряд методов обработки позволяет повышать разрешение изображений и аудиоданных путем логического вывода недостающих данных, такие методы не могут восстановить вредоносные искажения, потерянные на предшествующих этапах сжатия и нормализации данных.

соответствующих голосовому тракту человека, с использованием мел-частотного кепстра (как было описано в разделе «Аудиоданные» на с. 87).

Сжатие данных с потерями и без потерь

Для экономии пространства цифровые изображения, аудио- и видеоданные сохраняются в сжатых файловых форматах. Когда такое сжатие представляет собой просто более компактный способ записи, позволяющий сохранить тот же объем данных, используя меньшее количество байтов, его называют *сжатием без потерь*.

В отличие от этого *сжатие с потерями* также производит интеллектуальное удаление избыточных данных. Преимуществом такого подхода является то, что он позволяет сократить объем изображений, аудио- или видеоданных без заметного снижения их качества. Удаление данных при сжатии с потерями носит необратимый характер — удаленные данные не восстанавливаются при обратном преобразовании изображений, аудио- и видеоданных в несжатый формат.

Так, например, при сжатии аудиоданных в формат MP3 используется интеллектуальный алгоритм сжатия с потерями, выбирающий битрейт в зависимости от сложности аудиоданных, в результате чего менее сложные участки могут сохраняться с более низким битрейтом[1]. При сжатии изображений в формат JPEG также используется алгоритм сжатия с потерями, который сокращает размер изображений путем удаления несущественной информации, отсутствие которой не будет заметно человеку.

Предварительная обработка данных может обеспечить защиту от простейших атак в том случае, когда злоумышленник не осведомлен о производимых преобразованиях[2]. Кроме того, эта обработка может сделать затруднительным нападение на систему, наложив на атаку дополнительные ограничения. В то же время не следует полагаться на предварительную обработку данных как на эффективный метод защиты.

---

1 При сжатии в формат MP3 также используется преобразование Фурье.

2 См. пример в следующей статье: *Li X., Li F.* Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics // International Conference on Computer Vision, 2017. http://bit.ly/2FjDIVu. Авторы статьи показали, что применение фильтра может обеспечить успешное удаление вредоносных искажений из образов, созданных с помощью таких простых методов, как FGSM.

Интеллектуальное удаление вредоносного контента

На данный момент не существует статистических методов, позволяющих выявлять вредоносные образцы до их поступления в ГНС (см. примечание далее).

Статистические методы выявления вредоносных образов

К настоящему времени предложено несколько статистических методов выявления вредоносных образов.

Так, например, Гросс и др.[1] предложили выявлять вредоносные входные данные с помощью таких методов распределения данных, как поиск максимального среднего отклонения, а Фейнман и др.[2] предложили делать это, анализируя распределение выходных данных последнего скрытого слоя нейросети, вместо того чтобы подвергать анализу исходные входные данные. Поскольку выходные данные скрытого слоя представляют собой выделенные во входных данных высокоуровневые признаки, поступая так, мы будем анализировать распределение семантической информации, а не исходных пиксельных данных.

К сожалению, пока не доказано, что эти методы могут обеспечить эффективную защиту[3].

В большинстве случаев будет проще применить некоторое интеллектуальное преобразование, которое не будет сказываться на классификации невредоносных входных данных, но в то же время будет избавлять вредоносные входные данные от тех характеристик, которые делают их вредоносными. При этом вам не потребуется выявлять вредоносные входные данные — достаточно будет просто устранить те аспекты входных данных, которые могут использоваться во вредоносных целях, на этапе предварительной обработки до поступления данных в ГНС.

Например, некоторые пиксели изображения могут практически не учитываться при классификации невредоносных изображений, но в то же время

1 *Grosse K. et al.* On the (Statistical) Detection of Adversarial Examples, 2017. http://bit.ly/2IszblI.

2 *Feinman R. et al.* Detecting Adversarial Samples from Artifacts, 2017. http://bit.ly/2XpavTe.

3 *Carlini, Wagner.* Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods.

использоваться во вредоносных образах с целью заставить ГНС выдавать неверный результат. Удаление из данных этих пикселей позволит избавиться от вредоносности без ухудшения качества предсказаний, выдаваемых для хороших данных. Сходным образом удаление звуковых частот, выходящих за пределы характерного для человеческой речи диапазона, позволяет исключить возможность создания вредоносного входного сигнала, выходящего за пределы этого диапазона, не снижая эффективность системы распознавания речи.

Один из исследованных к настоящему времени подходов сводится к использованию метода *анализа главных компонентов* (principal components analysis, PCA). Это математический метод для выделения характеристик данных, оказывающих наибольшее влияние на принятие определенного решения. Была изучена возможность применения метода PCA для выделения тех аспектов данных, которые могут использоваться во вредоносных образах, но в то же время не влияют на принятие решений в отношении хороших данных. К сожалению, выяснилось, что выделить характеристики, которые бы явным образом влияли на вредоносность, невозможно, и потому данный подход опять же пока нельзя считать эффективным способом защиты[1].

Сокрытие информации о целевой системе

При сокрытии информации о целевой системе важно помнить о том, что под целевой системой здесь имеется в виду не ГНС, а вычислительная система в целом вместе со всеми ее защитными механизмами.

В главе 7 вы узнали, с какими сложностями сталкивается злоумышленник, пытаясь создать надежный вредоносный контент при проведении реальных атак. Если злоумышленник плохо осведомлен о ГНС или не может направлять запросы целевой системе, это существенно усложняет создание вредоносных входных данных. И наоборот, хорошая осведомленность о ГНС позволяет злоумышленнику создать копию нейросети, чтобы с ее помощью разработать и «отточить» вредоносные входные данные до того, как применить их против целевой системы. Возможность проверить атаку на целевой системе — ценное подспорье при тестировании вредоносных входных данных или разработке атаки с помощью методов черного ящика

[1] *Carlini, Wagner.* Adversarial Examples Are Not Easily Detected.

(которые, как правило, требуют выполнения большого количества запросов к целевой системе).

Если ваша организация использует предварительно обученную модель, которая предоставляется на коммерческой основе или как открытое ПО, то злоумышленнику не составит труда создать копию ГНС и разработать точные вредоносные образы. Как вы видели в главе 7, даже доступ к обучающим данным дает злоумышленнику достаточную информацию для того, чтобы создать вполне точную замещающую модель, позволяющую получить переносимые вредоносные образы. Однако, хотя использование алгоритма ГНС, основанного на коммерчески доступной или свободно распространяемой модели, является не самым лучшим решением, зачастую это единственный практически реализуемый вариант, учитывая, насколько большой объем размеченных данных обычно требуется для обучения сети.

Осведомленность обо всей последовательности обработки включает в себя осведомленность об этапах предварительной обработки, способных сделать вредоносные входные данные невредоносными, и способе интерпретации результатов нейронной сети. Последнее включает в себя информацию о том, какие пороговые значения предсказаний используются при принятии решений и как реагирует организация на такие предсказания.

Осведомленность об используемых защитных механизмах включает в себя информацию о методах выявления или удаления вредоносных входных данных. Сведения об этих защитных методах также можно получить путем тестирования — направляя в систему запросы и проверяя итоговую реакцию организации на вредоносные входные данные. Например, зная о том, что при выявлении вредоносной заплатки вы используете ее для поиска других вредоносных изображений, злоумышленник может отказаться от использования заплатки или, наоборот, воспользоваться заплаткой с целью вызвать ложные срабатывания защиты.

Сделать целевую систему менее доступной для злоумышленника и тем самым ограничить его возможности в создании вредоносных входных данных можно с помощью следующих практических мер.

- *Дросселирование запросов (throttling)*. Ограничение допустимой скорости подачи запросов может препятствовать проведению прямой атаки черного ящика, требующей выполнения большого количества запросов. Однако следует отметить, что дросселирование на основе идентификации клиентов может не дать ожидаемого результата, поскольку выявленный

злоумышленник для передачи запросов может просто воспользоваться рядом других идентификаторов — например, других IP-адресов или учетных записей.

- *Выявление на основе паттернов запросов.* В качестве альтернативы целевая система может выявлять попытки проведения прямой атаки черного ящика путем выявления длинных цепочек очень похожих, но неодинаковых входных сигналов, потому что такая атака требует выполнения большого количества сходных запросов, каждый из которых итеративно модифицирует предыдущий входной сигнал. Так, например, сходство входных сигналов можно быстро выявлять путем *хеширования изображений*. При этом каждому изображению присваивается хеш-код таким образом, чтобы хеш-коды похожих изображений мало отличались друг от друга[1]. Конечно, при таком подходе не исключена вероятность того, что длинная цепочка похожих запросов не будет представлять собой вредоносные данные, поэтому применимость такого способа защиты зависит от конкретного сценария.

- *Минимизация обратной связи.* Для создания надежных вредоносных образов злоумышленник может использовать ответные реакции целевой системы на направляемые ей запросы. Эти ответные реакции могут использоваться либо непосредственно для разработки вредоносных входных данных, либо для проверки вредоносных входных данных после их первоначальной разработки с помощью замещающей модели. Сократив объем полезной информации, возвращаемой в ответ на запрос, вы затрудните генерацию вредоносных входных данных. Позаботьтесь о том, чтобы в ответах не раскрывались непосредственные результаты модели и чтобы в сообщениях об ошибках не содержалось слишком много информации.

- *Обеспечение недетерминированных ответов.* Обеспечение недетерминированных ответов — очень мощный способ защиты в том случае, когда это мешает злоумышленнику составить четкое представление об устройстве вашей системы. Один из способов реализации такого подхода представлен в подразделе «Оценка неопределенности случайного отсева» на с. 241, однако ввести такую недетерминированность можно и в рамках общей последовательности обработки, если это позволяет сделать сценарий эксплуатации системы.

---

[1] Это не следует путать с криптографическим хешированием, при котором хеш-коды не отражают степень сходства входных данных.

Создание эффективных механизмов защиты от вредоносных входных данных

В этой главе рассмотрены методы выявления и удаления вредоносных входных данных.

По мере роста устойчивости нейронных сетей к вредоносным атакам совершенствуются и методы вредоносных атак. Эта непрерывная «гонка вооружений» между атакующей и защищающейся сторонами во многом напоминает эволюцию средств для выявления вредоносного ПО и отфильтровывания спама из электронной почты, и разновидности вредоносных образов подобным образом будут меняться по мере изменения имеющихся средств защиты.

Открытые проекты

Несмотря на то что на данный момент нет надежных средств защиты от вредоносных атак, в рамках ряда инициатив ведется работа по привлечению широкой общественности к изучению вредоносных атак и способов защиты от них с целью повысить надежность нейросетевых алгоритмов путем проверки устойчивости защитных механизмов к различным способам атаки. Некоторые из этих инициатив уже упоминались в главе 6. Ознакомление с ними лучше начать со следующих проектов.

- *CleverHans* (http://bit.ly/2WNNs0c). Открытая библиотека и репозиторий кода для разработки методов атаки и соответствующих средств защиты с целью сравнительного анализа уязвимости систем машинного обучения к вредоносным образам[1].
- *Foolbox*. Набор инструментов для создания вредоносных образов с целью тестирования механизмов защиты[2]. Ознакомление с ним лучше начать с изучения документации (http://bit.ly/2FmRpmx).
- *Adversarial Robustness Toolbox от компании IBM*. Репозиторий кода этой библиотеки (http://bit.ly/2XZ7EgW) содержит код различных методов атаки, защитных механизмов и методов выявления атак. Данная библиотека также поддерживает метрики устойчивости.

---

1 *Papernot N. et al.* Technical Report on the CleverHans v2.1.0 Adversarial Examples Library, 2017. http://bit.ly/2Xnwav0.

2 *Rauber J. et al.* Foolbox: A Python Toolbox to Benchmark the Robustness of Machine Learning Models, 2018. http://bit.ly/2WYFgPL.

- *Robust ML*. Целью проекта (https://www.robust-ml.org) является предоставление централизованного сайта с информацией о методах защиты, результатах их анализа и тестирования.
- *Robust Vision Benchmark*. Проект (http://bit.ly/2L9A3gT) призван предоставить платформу для сравнительного анализа эффективности атак и устойчивости моделей.
- *Конкурсы*. В целях поощрения участия в разработке различных методов атаки и средств защиты от них был проведен ряд конкурсов, некоторые из них организованы компанией Google[1].

Также см. информацию о конкурсе Unrestricted Adversarial Examples Challenge (http://bit.ly/2L0setv) и конкурсах Kaggle[2].

Получение общей картины

Выбор средств защиты зависит от того, в каком контексте развертывается ГНС и какую угрозу могут теоретически представлять для вашей организации вредоносные образы. Если вы полностью контролируете и можете доверять тем данным, которые обрабатывает ИИ, то вероятность атаки с использованием вредоносных входных данных может равняться нулю.

Разработка комплексных решений с надлежащей устойчивостью к вредоносным образам требует разностороннего подхода. Необходимо принять во внимание *всю систему в целом,* а не только используемую вашей организацией модель (или модели) в отрыве от других компонентов. При этом вероятность успешного проведения определенных атак можно уменьшить путем внесения простейших изменений в процессы обработки или последовательность их выполнения.

Например, хотя вы не можете лишить злоумышленника тех ресурсов и навыков, которыми он располагает, в то же время вы можете затруднить проведение атаки, ограничив возможности злоумышленника по изменению входных данных или закрыв доступ к информации о целевой системе. Последнее можно обеспечить, исключив раскрытие нежелательной или излишней информации в выдаваемых системой ответах. Когда есть риск атаки в физическом мире, для устранения угрозы могут оказаться достаточными

---

1 *Kurakin A. et al.* Adversarial Attacks and Defenses Competition // Neural Information Processing Systems (NIPS) conference 2017. http://bit.ly/2WYGzy9.

2 Несколько конкурсов в рамках конференции NIPS 2017.

такие простые нетехнические меры, как наблюдение за физическими областями, в которых может быть проведена вредоносная атака.

На практике во многих приложениях компоненты ИИ уже защищены дополнительными средствами обеспечения информационной безопасности. Так, виртуальные помощники снабжены дополнительными уровнями безопасности, которые исключают подачу непреднамеренных команд, например требуя аутентификации перед выполнением таких важных действий, как перевод денежных средств, или выдавая звуковые ответы на голосовые команды. Иногда степень риска также можно снизить путем проверки достоверности результатов модели с помощью информации, получаемой из других источников. Например, автономное транспортное средство может дополнять свои знания данными камеры, а не полагаться на визуальную информацию целиком.

Оценку методов защиты следует проводить, исходя из предположения, что злоумышленник обладает полной осведомленностью о них. То есть вам следует оценить устойчивость системы к атаке, проводимой с полной осведомленностью о модели, последовательности обработки и средствах защиты. При проведении такой оценки следует всегда использовать самые сильные из существующих методов атаки. Для этого нужно быть в курсе последних событий в сфере вредоносных атак и методов защиты от них. Эта оценка не должна выполняться лишь один раз с получением статичного результата, а должна повторяться раз за разом по мере появления более эффективных методов атаки и способов защиты. Она должна представлять собой сочетание формальных процедур оценки с тестированием кибербезопасности всей системы, с включением вредоносных образов в применяемые методы красной и синей команд.

Для получения целостной картины нужно не только учесть, какие средства защиты включает в себя техническое решение, но и разобраться с тем, какой общий эффект может произвести атака на организацию и как можно не допустить некорректное реагирование на атаку. Так, например, для предотвращения атак типа «отказ в обслуживании» иногда целесообразно включить в контур управления человека, который будет проверять поступающие от камер видеонаблюдения сигналы тревоги перед выполнением соответствующих ответных действий. В таком случае система ИИ будет лишь сортировать данные видеонаблюдения, оставляя окончательное решение за человеком.

Если вы можете выявлять вредоносные атаки, подумайте, как вам следует на них реагировать. Иногда лучше просто протоколировать каждую попытку

атаки, не предпринимая каких-либо ответных действий, подобно тому как обрабатывается спам, рассылаемый по электронной почте. С другой стороны, в некоторых случаях целесообразно предпринимать более активные ответные действия. Так, повторное выявление вредоносных входных данных может послужить основанием для того, чтобы (к примеру) ограничить в дальнейшем доступ пользователя к социальной сети. Если вы открыто выявляете вредоносные входные данные, проследите за тем, чтобы ответные действия вашей организации исключали возможность проведения атаки типа «отказ в обслуживании» путем направления в систему потока вредоносных образов.

И наконец, следует помнить, что выявление вредоносных входных данных и выполнение соответствующих ответных действий является лишь одним из элементов обеспечения информационной безопасности машинно-обучаемых моделей. Вы должны также позаботиться о том, чтобы модель могла надежно работать во всем диапазоне принимаемых ею входных данных, как вредоносных, так и невредоносных. Кроме того, в рамках обеспечения информационной безопасности следует рассмотреть и другие виды вредоносных атак против машинно-обучаемых моделей, такие как нарушающее целостность модели «отравление» обучающих данных или «обратное проектирование» модели с целью извлечения конфиденциальных обучающих данных.

ГЛАВА 11

Дальнейшие перспективы: повышение надежности ИИ

В данной книге рассмотрены методы обмана ИИ, не способные ввести в заблуждение человека. Однако не стоит забывать, что человек подвержен оптическим и звуковым иллюзиям. Что интересно, проведенные исследования показали, что определенные вредоносные образы могут обмануть и человека, имеющего ограниченный запас времени[1]. И наоборот, определенные оптические иллюзии также могут обмануть и нейронные сети[2].

Эти случаи дают основание предполагать, что у биологического и искусственного восприятия есть сходство, однако в основе использования вредоносных входных данных лежит базовый принцип, согласно которому модели глубокого обучения обрабатывают данные не так, как это делают их биологические прообразы. Хотя глубокое обучение позволяет создавать модели, иногда превосходящие человеческие возможности по обработке сенсорной информации, способ восприятия этих моделей, как правило, сильно отличается от того, как в действительности человек усваивает и воспринимает визуальную и аудиальную информацию.

Сейчас в сфере глубокого обучения открываются интересные области исследования, изучение которых, вероятно, приведет к сокращению разницы между искусственным и биологическим восприятием. Результатом этих исследований может стать ИИ с большей устойчивостью к вредоносным образам. Далее перечислены некоторые из таких направлений.

---

1 *Elsayed G. F. et al.* Adversarial Examples that Fool both Computer Vision and Time-Limited Humans, 2018. http://bit.ly/2RtU032.

2 *Eiji W. et al.* Illusory Motion Reproduced by Deep Neural Networks Trained for Prediction // Frontiers of Psychology, March 2018. http://bit.ly/2FkVxmZ.

Повышение устойчивости за счет распознавания контуров

Нейробиологам и психологам давно известно, что человек формирует свое представление об окружающем мире посредством движения и физического исследования. Грудной ребенок рассматривает предметы под разными углами по мере того, как он движется сам или движутся эти предметы.

Как известно, визуальное восприятие человека сильно зависит от движения и углов обзора, что позволяет ему изучать объекты и их границы. Если глубокие нейронные сети будут делать больший акцент на контурах объекта, то проведение искажающей атаки станет невозможным. Подобный принцип действует и при распознавании звука по типичным временным паттернам и относительной высоте. Значимые признаки, которые мы выделяем из сенсорной информации для того, чтобы понять мир, очевидно, сильно отличаются от признаков, которые выделяют ГНС.

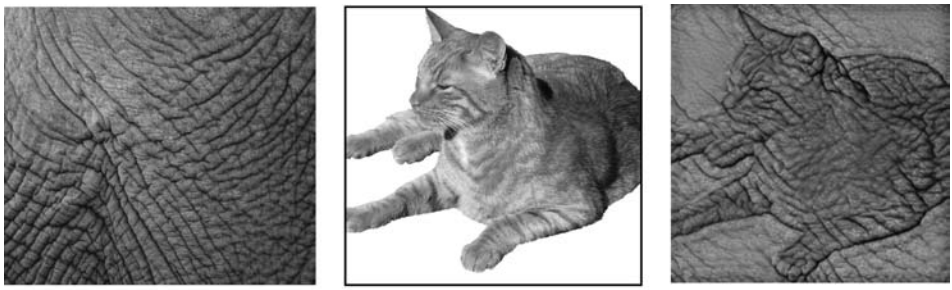
В своем исследовании Гейрхос и др.[1] высказали идею о том, что СНС, обученная на данных из набора ImageNet (такая как ResNet50), делает больший акцент на имеющихся в изображениях текстурах, а не на контурах объектов. Человек же, напротив, принимает решения, руководствуясь главным образом формой объекта. Для проверки этой идеи исследователи сгенерировали на основе набора ImageNet ряд изображений с конфликтующей информацией о форме и текстуре. Представленная на рис. 11.1 иллюстрация из этой статьи показывает, что при передаче модели ResNet50 изображения с конфликтующей информацией о форме и текстуре она производит классификацию, опираясь в основном на данные о текстуре.

Создав новый набор обучающих данных со «стилизованными» изображениями, метки которых соответствовали форме объектов, а не их текстуре, исследователи смогли дообучить СНС таким образом, чтобы она меньше акцентировала внимание на текстуре и больше руководствовалась имеющимися в изображении контурами. Такой подход позволяет повысить устойчивость классификаторов к вносимым в изображения искажениям, поскольку большинство искажений изменяет лишь текстуру изображения, практически не затрагивая контур объекта.

Данное исследование открывает большие перспективы для создания нейронных сетей с высокой устойчивостью к вредоносным образам. Если при

[1] *Geirhos R. et al.* ImageNet-Trained CNNs Are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness, 2019. http://bit.ly/2N0FuB2.

принятии решения CHC будет делать больший акцент на контурах, то это решение будет менее подвержено влиянию вредоносных искажений, охватывающих значительную часть изображения.



а) Изображение с текстурой

81,4 % **Индийский слон**

б) Изображение с объектом

71,1 % **Полосатый кот**

в) Конфликт признаков текстуры и формы

63,9 % **Индийский слон**

Рис. 11.1. Классификация моделью ResNet50 изображения кожи слона *(а)*, изображения кошки *(б)* и изображения кошки с наложенной текстурой кожи слона *(в)* (из статьи Гейрхоса и др., 2019 год)

Мультисенсорные входные данные

Существует и ряд других интересных новых идей, берущих свое начало из теоретических основ нейробиологии, касающихся работы и устройства человеческого мозга. Эти идеи могут стать основой для новых подходов к созданию ИИ, который будет более точно имитировать человеческое восприятие.

Например, формируя представление об окружающем мире, человеческий мозг может успешно сочетать данные, получаемые от нескольких источников сенсорной информации (например, данные от органов зрения, слуха и осязания, данные о температуре). Эти мультисенсорные входные данные позволяют человеку непрерывно проверять достоверность своей картины мира (хотя иногда человек может интерпретировать их неправильно, как описано в примечании далее).

Эффект Макгурка

Эффект Макгурка — это слуховая иллюзия, демонстрирующая, что при восприятии речи человеческий мозг руководствуется и визуальной, и звуковой информацией.

> При восприятии звукового стимула (например, аудиозаписи, на которой кто-то произносит слог «ба») в сочетании с противоречащей ему визуальной информацией (например с видеозаписью, на которой кто-то произносит слог «фа») человеческий мозг замещает звуковой стимул визуальным и «слышит» слог «фа».
>
> Вы можете опробовать действие этого эффекта на себе, просмотрев соответствующее видео от телеканала BBC Two (http://bit.ly/2ISlb3v).

По мере развития систем ИИ они будут все чаще сочетать данные, получаемые из разнородных источников информации — как сенсорной, так и несенсорной, — для проверки достоверности формируемой ими картины мира. Если мы сумеем понять, как эту проблему решает природа, это позволит нам создавать системы ИИ, способные работать с более высоким уровнем точности при сочетании различных видов неструктурированных данных, например звуковых и визуальных.

В своем исследовании Хокинс и др.[1] предложили новую модель работы человеческого мозга — «теорию тысячи мозгов». Согласно этой теории, неокортекс содержит множество мини-мозгов — так называемых кортикальных колонок, которые обрабатывают сенсорную информацию, работая параллельно друг другу. Каждая из этих кортикальных колонок имеет полное представление об объектах окружающего мира. Каждая колонка принимает некоторую часть сенсорных данных — скажем, данные, поступающие от небольшого участка поля зрения или от кончика пальца, — и определяет, какому объекту соответствуют эти данные (например, чашке или шапке) и какой части этого объекта.

В качестве примера была рассмотрена работа кортикальных колонок, когда человек смотрит на кружку и одновременно касается ее одним пальцем. При этом одна колонка может принимать осязательные данные, порождаемые прикоснувшимся к кружке пальцем, а ряд других колонок — визуальные данные от зрительной коры, отражающие внешний вид различных частей кружки. Руководствуясь общедоступной информацией об имеющейся модели мира, каждая кортикальная колонка может сделать обоснованное предположение о том, какому объекту и какой его части соответствуют обрабатываемые ею сенсорные данные. Затем на основе данных всех колонок

---

[1] *Hawkins J. et al.* A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex // Frontiers in Neural Circuits 12, 121, 2018. http://bit.ly/2ZtvEJk.

составляется общее представление об объекте на основании своего рода голосования в случае противоречий.

Идея о том, что мозг генерирует тысячи одновременно действующих моделей отдельного объекта, сильно отличается от используемого в ГНС принципа строгой иерархии, при котором представление об отдельном объекте формируется путем последовательного выделения высокоуровневых признаков. Этот параллелизм обеспечивает большую устойчивость к ошибкам за счет сочетания данных, поступающих от нескольких органов чувств, и, вероятно, может в какой-то мере объяснить эффект Макгурка.

Вложенность и иерархия объектов

«Теория тысячи мозгов» также предлагает свое объяснение того, как мозг обрабатывает вложенность одних объектов в другие. Предполагается, что наряду с информацией о положении кортикальные колонки отслеживают смещения объектов относительно друг друга и, соответственно, существующие между ними взаимосвязи.

В еще одном исследовании было предложено пересмотреть подход к созданию сверточных нейронных сетей (СНС), которые уже успели стать своего рода стандартом в сфере обработки изображений. Сверточные слои выделяют имеющиеся в изображении признаки, но не обеспечивают обучение на основе относительного положения этих признаков внутри изображения. Так, например, когда СНС распознает лицо по наличию таких признаков, как форма глаз, носа и рта, при этом не учитывается относительное положение этих признаков.

Джеффри Хинтон и его команда предложили использовать так называемые *капсульные сети*[1], дающие возможность лучше выразить иерархические взаимосвязи объектов окружающего мира. Поскольку эти взаимосвязи являются неотъемлемой частью расчетов такой сети, она способна отслеживать взаимосвязи между составными частями объектов, что, в свою очередь, дает ей более четкое представление об имеющихся в изображении объектах, вне зависимости от угла обзора. Капсульные сети демонстрируют более высокую устойчивость к вредоносным атакам из-за лучшего понимания ими контекста изображений.

---

[1] *Sabour S. et al.* Dynamic Routing Between Capsules, 2017. http://bit.ly/2FiVAjm.

В заключение

В последнее время растет интерес к поиску путей сближения нейробиологии и искусственного интеллекта. Если лучше понимать то, как работает человеческий мозг, применение новых идей в методологиях ИИ позволит усовершенствовать имитацию процесса обучения человека. Однако это также означает, что алгоритмы ИИ будут подвержены тем же проблемам, которым подвержен мозг человека, если мы не реализуем препятствующие этому программные стратегии. С точки зрения нейробиологии ИИ представляет собой способ проверки гипотез об устройстве человеческого мозга. Исследования о том, как и когда совершают ошибки в интерпретации данных люди и ИИ, будут способствовать сокращению разрыва между ними.

И когда ИИ станет гораздо больше напоминать биологический интеллект, можно будет уже не проводить различие между обманом человека и машины. Возможно, настанет день, когда нейронные сети научатся настолько хорошо имитировать человеческое восприятие, что визуальные и звуковые вредоносные образы станут пережитком прошлого.

ПРИЛОЖЕНИЕ

Справочник математических обозначений

Если вы не знакомы с используемыми в книге математическими обозначениями или успели забыть их, в табл. П.1 представлено их краткое описание в контексте данной книги.

Таблица П.1. Сводный перечень математических обозначений

Обозначение	Описание
x	Переменная, записанная обычным (не полужирным) шрифтом, представляет собой скалярное значение
$\mathbf{x}$	Переменная, записанная полужирным шрифтом, представляет собой вектор
$y = f(\mathbf{x}; \Theta)$	Результат применения к входному вектору $\mathbf{x}$ функции f, определяемой параметрами Θ. В контексте данной книги это результат, выдаваемый ГНС-моделью для конкретного входного сигнала, где f — алгоритм ГНС-модели, Θ — его параметры, определяемые в ходе обучения, а $\mathbf{x}$ — входной сигнал модели
$C(f(\mathbf{x}; \Theta), \mathbf{y})$	Результат функции C, полученный для заданных функции $y = f(\mathbf{x}; \Theta)$ и вектора $\mathbf{y}$. В контексте данной книги это зависимость уровня затрат (или потерь) ГНС-модели, получаемого для конкретного входного сигнала, от требуемого выходного сигнала $\mathbf{y}$
$\mathbf{x}_i$	i-й элемент вектора $\mathbf{x}$
$\frac{dy}{dx}$	Производная от y по x
$\frac{\partial y}{\partial x}$	*Частная* производная от y по x, где x — одна из переменных, влияющих на y

Продолжение ⇗

Таблица П.1 (продолжение)

Обозначение	Описание
$\nabla_x f$	Символ *«набла»* (перевернутый вверх ногами греческий символ *«дельта»*) означает «градиент». $\nabla_x f$ при этом означает вектор частных производных функции f, полученный для вектора **x**. А если сказать проще, это эффект, производимый на функцию f очень небольшим изменением величины вектора **x**
$\{1, 2... L\}$	Множество чисел от 1 до L
$\mathbb{R}$	Множество действительных чисел
$\{x : P(x)\}$	Множество всех значений x, для которых истинно логическое выражение $P(x)$
$\arg\min_x \{f(x) : P(x)\}$	Возвращает значение x, при котором функция $f(x)$ принимает минимальное значение при условии, что истинно логическое выражение $P(x)$
$x \in \mathbb{R}$	Означает, что x принадлежит множеству действительных чисел
$\|\mathbf{x}\|_p$	L^p-норма вектора **x**. О том, что представляют собой L^p-нормы, говорится в подразделе «Математические методы оценки искажения» на с. 124
$\sum x$	Сумма всех возможных значений x
$\sum_{x \neq t} x$	Сумма всех возможных значений x, где $x \neq t$
ε	Греческая буква ε (эпсилон) используется для обозначения чрезвычайно малых (микроскопических) величин

Об авторе

Кэти Уорр получила диплом в области искусственного интеллекта и компьютерных наук в Эдинбургском университете еще в то время, когда в силу недоступности достаточных вычислительных мощностей и объемов данных глубокое обучение было исключительно теоретической дисциплиной. Много лет она занималась разработкой корпоративного программного обеспечения, в настоящее время специализируется в области ИИ и считает большой удачей свое участие в этих захватывающих исследованиях в то время, когда искусственный интеллект становится неотъемлемой частью нашей повседневной жизни.

Об обложке

На обложке этой книги изображена ушастая фацелина *(Facelina auriculata)* — разновидность голожаберных моллюсков. Она обитает вдоль северо-западного побережья Европы и южнее, в Средиземном море.

Голожаберные моллюски — небольшие мягкотелые моллюски, которые встречаются в морях и океанах по всему миру и славятся фантастически красочным внешним видом, даже невзирая на свои обычно крошечные размеры. Хотя их называют морскими слизнями, эти существа относятся к уникальной группе моллюсков, сбрасывающих раковины в начале своего роста (в это обширное семейство также входит ряд улиток, двустворчатых моллюсков и осьминогов). Большинство голожаберных моллюсков ползают по рифам и морскому дну в поисках своей добычи — морских анемон, гидроидных полипов и других мелких существ, которых они поглощают с помощью радулы — снабженного зубами органа, отдаленно напоминающего язык. Ушастые фацелины обитают в мелких прибрежных водах, но во время отлива их также можно обнаружить под камнями на берегу.

Эти небольшие (до 4 сантиметров в длину) существа ищут свою добычу по запаху и вкусу, определяя химический состав морской воды с помощью ринофоров — напоминающих рожки чувствительных отростков. Голожаберные моллюски являются гермафродитами, после спаривания каждая особь откладывает яйца в лентовидную кладку.

Принято считать, что привлекающая взоры окраска голожаберных моллюсков выполняет оборонительную функцию — яркие цвета призваны известить хищников о том, что их обладатель либо ядовит, либо невкусен. Поскольку тело ушастых фацелин полупрозрачно, оно может принимать оттенок поглощенной ими добычи. У многих голожаберных моллюсков также есть пальцевидные отростки, называемые цератами. Внутри них

содержатся жгучие ткани, которые моллюск извлекает из своей добычи (такой как морские анемоны) и использует для собственной защиты.

Многие из тех видов животных, которые изображены на обложках книг от издательства O'Reilly, находятся под угрозой исчезновения, хотя каждый из них является важной частью нашего мира.

Представленную на обложке цветную иллюстрацию создала Карен Монтгомери (Karen Montgomery) на основе карандашного рисунка, взятого из книги Генри Адамса (Henry Adams) и Артура Адамса (Arthur Adams) *The Genera of Recent Mollusca* («Виды современных моллюсков») 1858 года.

Кэти Уорр

Надежность нейронных сетей: укрепляем устойчивость ИИ к обману

Перевел с английского *А. Павлов*

Заведующая редакцией	*Ю. Сергиенко*
Руководитель проекта	*С. Давид*
Ведущий редактор	*Н. Гринчик*
Научный редактор	*Н. Искра*
Литературный редактор	*А. Дубейко*
Художественный редактор	*В. Мостипан*
Корректоры	*О. Андриевич, Е. Павлович*
Верстка	*Г. Блинов*

Изготовлено в России. Изготовитель: ООО «Прогресс книга».
Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург,
Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 02.2021. Наименование: книжная продукция. Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 —
Книги печатные профессиональные, технические и научные.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214, тел./факс: 208 80 01.

Подписано в печать 25.01.21. Формат 70×100/16. Бумага офсетная. Усл. п. л. 21,930. Тираж 500. Заказ 0000.