УДК 004.93'1 DOI:10.17587/it.24.406-413

Д. В. Пантюхин, ст. преподаватель, e-mail: dpantiukhin@hse.ru,
Е. Карелова, студент, e-mail: ekarelova@edu.hse.ru,
Национальный исследовательский университет "Высшая школа экономики"

Повышение качества классификации компьютерных атак сверточной нейронной сетью посредством балансировки обучающей выборки

Рассматривается подход к балансированию обучающей выборки при решении задачи классификации компьютерных атак сверточной нейронной сетью. Проведено сравнение на одной и той же структуре нейронной сети качества классификации без и с использованием балансировки обучающей выборки. Проведен вычислительный эксперимент, показавший эффективность использования балансировки в случае, когда число данных разных классов в исходной выборке существенно различается.

Ключевые слова: *обнаружение компьютерных атак, сверточная нейронная сеть, балансировка обучающей выборки, базы данных компьютерных атак*

Введение

В настоящее время наблюдается бурный рост применений методов глубинного обучения в различных прикладных задачах классификации, кластеризации, прогнозирования и др. Несомненно, самые большие успехи глубинного обучения показаны в области обработки изображений, тем не менее эти методы находят свое применение и в других областях.

Повсеместное распространение компьютерных сетей делает их привлекательными для злоупотреблений, среди которых компьютерные атаки занимают существенное место. Обнаружение факта атаки на компьютерную сеть и определение ее типа являются актуальной и сложной задачей, что связано с ростом объемов передаваемой информации и повышением ценности этой информации. Эффективность классического подхода к обнаружению атак, связанного с составлением и обработкой так называемых сигнатур — признаков наличия атак, уменьшается в связи с тем, что вынужденно растет количество сигнатур, которые необходимо проверять для каждого сетевого соединения. Поэтому распространение получили методы, основанные на применении различных методов мягких вычислений. В настоящее время для обнаружения атак используются:

— сигнатурные методы [8—10];
— методы, основанные на спецификациях [11];
— методы анализа систем состояний [12—14];
— методы анализа графов сценариев атак [15];
— экспертные системы [16, 17];
— MARS — Multivariate Adaptive Regression Splines [18];
— SVM [19];
— иммунные сети [20, 21] и генетические алгоритмы;
— нейронные сети [22, 23].

Одними из наиболее успешных нейронных сетей стали сверточные нейронные сети, которые и будут использованы в данной работе.

Анализ публикаций по применению методов мягких вычислений для обнаружения компьютерных атак показывает, что подавляющее большинство исследователей используют в своей работе морально устаревшую базу данных KDD99 [6], которая была создана в 1998—1999 гг. и не в полной мере отражает современные атаки в компьютерных сетях.

Последние годы появляются новые базы данных компьютерных атак (см. разд. 1), однако число работ, в которых они используются, невелико. На наш взгляд, это связано с недостаточной информированностью исследователей о наличии новых баз данных, большинство из которых доступны для некоммерческого использования. Более подробно известные авторам базы данных компьютерных атак будут описаны в разд. 1, однако для всех них можно выделить особенность, связанную с природой компьютерной атаки. Во-первых, атака — это относительно редкое событие (за исключением DDOS-атак), которое теряется на фоне числа пакетов нормальных (не атакующих) сетевых соединений, во-вторых, число данных об атаках различного типа существенно отличается (например, в базе данных KDD99 число нормальных пакетов — 972780, число пакетов с атакой типа DDOS — 3883370, а с атакой типа U2R — всего 52).

Существенное различие в числе данных для атак различного типа приводит к тому, что классификатор должен учитывать эту информацию, иначе возможны ситуации, когда классификатор в среднем работает неплохо, но отдельные (редкие) типы атак не распознает вовсе. Один из возможных подходов к улучшению качества таких классификаторов заключается в размножении (дублировании) данных об атаках редких типов. В связи с тем, что современные сверточные нейронные сети основаны на "пакетном" обучении, когда нейронная сеть обучается на небольших блоках (пакетах, *batch*) данных, чередующихся во время обучения, имеет смысл поддерживать заданные пропорции числа данных различных классов внутри каждого пакета. Именно такой подход используется в нашей работе.

1. Базы данных компьютерных атак

Первой, самой популярной, а долгое время чуть ли не единственной свободно доступной базой данных компьютерных атак была база **KDD99Cup** [6], созданная в 1998—1999 гг. Это открытая, бесплатная для проведения экспериментальных исследований база данных, создана Калифорнийским университетом (*Information and Computer Science University of California*). Она содержит около 5 млн записей о сетевых соединениях. Каждая запись представляет собой образ сетевого соединения, включает 41 параметр сетевого трафика (например, такие как тип протокола, порт отправления и порт назначения и др.), среди которых содержится три типа признаков: символьные (например, тип протокола), логические (флаги) и числовые.

База содержит информацию о 22 типах атак, при этом атаки делятся на четыре основных класса: Denial of Service (DoS), Remote to User (R2L), User to Root (U2R) и Probing.

Атака *Denial of Service (DoS)* — отказ в обслуживании, характеризуется генерацией большого объема трафика, что приводит к перегрузке и блокированию сервера и включает шесть типов: *back*, *land*, *neptune*, *pod*, *smurt*, *teardrop.*

Атака *Remote to User* (R2L) характеризуется получением доступа незарегистрированного пользователя к компьютеру со стороны удаленной машины, включает четыре типа*: buffer_overflow, perl, loadmodule, rootkit.*

Атака *User to Root* (U2R) предполагает получение зарегистрированным пользователем привилегий локального суперпользователя (администратора), включает восемь типов: *ftp_write*, *guess_passwd*, *imap*, *multihop*, *phf*, *spy*, *warezclient*, *warezmaster.*

Таблица 1

Число записей разных типов атак в базе данных KDD99Cup

Файл БД	DoS	Probe	U2R	R2L	Normal
10 % KDD	391 458	4107	52	1126	97 277
CorrectedKDD	229 853	4166	70	16 347	60 593
Whole KDD	3 883 370	41 102	52	1126	972 780

Атака *Probing* заключается в сканировании портов в целях получения конфиденциальной информации и включает четыре типа: *ipsweep*, *nmap*, *portsweep*, *satan*.

Число записей, относящихся к разным типам атак, существенно различно и приведено в табл. 1. В этой базе доступно также несколько файлов в текстовом формате, представляющих собой модификации этой базы.

База была подвергнута критике [1], основными замечаниями к ней были:

— большое число дублирующихся бесполезных записей (около 80 %!);

— недостаток записей о редких типах атак;

— моральное устаревание — база собрана в 1998—1999 гг. и не содержит информации о современных атаках.

В 2009 г. (через 10 лет!) получила развитие также модификация этой базы под названием **NSL-KDD**, в которой были отброшены дублирующиеся записи, проведена работа по выравниванию числа данных о разных типах атак [1]. В результате база сократилась в 4 раза.

В 2012 г. была анонсирована база данных **UNB ISCX 2012** Intrusion Detection Evaluation Data Set [2], созданная в Канадском Институте кибербезопасности (The Canadian Institute for Cybersecurity (CIC)). Эта база, так же как и KDD99, содержит информацию о сетевых соединениях, но также предлагает записи всего трафика. Трафик получен в течении 7 дней с тестовой компьютерной сети (6 локальных подсетей, NAT сервер, основной и вторичный серверы, средства мониторинга трафика) при различных сценариях атак, включает 2450324 соединений (около 90 Гбайт трафика), из них 68792 — атаки.

В 2017 г. была представлена база **UNB ISCX 2017**, построенная по аналогичному принципу и содержащая новейшие сценарии атак. Также был представлен инструмент для анализа трафика [3].

В 2015 г. была представлена база **UNSW-NB15** [4, 5], созданная в Австралийском Цен-

тре Кибербезопасности (Australian Centre for Cyber Security). База содержит как записи о сетевых соединениях, так и сам трафик (около 100 Гбайт) с тестовой компьютерной сети (три сервера). В базе представлены девять типов атак: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms. Каждая запись содержит 47 полей информации о сетевых соединениях и два поля информации о типе атаки (или ее отсутствии). Общее число записей около 2 млн. Информация о сетевых соединениях разбита на четыре текстовых файла (формат *csv*), отдельно представлены файлы для обучения и тестирования классификаторов, содержащие соответственно 175 341 и 82 332 записи.

2. Сверточные нейронные сети для классификации компьютерных атак

2.1. Описание сверточных нейронных сетей

Сверточные нейронные сети состоят из нескольких вычислительных слоев следующего типа.

А. Сверточные слои — это слои, которые выполняют операцию свертки входных данных с неким "ядром" — набором параметров. Параметры таких ядер чаще всего могут изменяться в процессе обучения, однако встречаются и сверточные слои с неизменяющимися параметрами. К последним можно отнести так называемые слои "пулинга" (*pooling*) которые предназначены для резкого уменьшения числа настраиваемых параметров. В нашей работе, так как размерность входных данных не очень велика, слои пулинга использоваться не будут, и все свертки — с настраиваемыми параметрами. Операцию свертки можно применять к данным любой размерности, наибольшее распространение получили двумерные свертки, поскольку основная область применения — это обработка изображений. Однако для нашей задачи данные представляются в виде векторов, т.е. одномерных массивов, поэтому будем использовать одномерные свертки. Пусть x_i — i-й элемент входного вектора, $i = 1...N$, где N — число входных параметров, w_k — k-й компонент ядра (которое также представлено в виде вектора), $k = 1...K$, в ядро также входит отдельный коэффициент w_0, который называется "смещение", тогда выход одномерного сверточного слоя определяется как:

$$y_j = \sum_k (x_{i+k} * w_k) + w_0, j = 1 \ldots N - K.$$

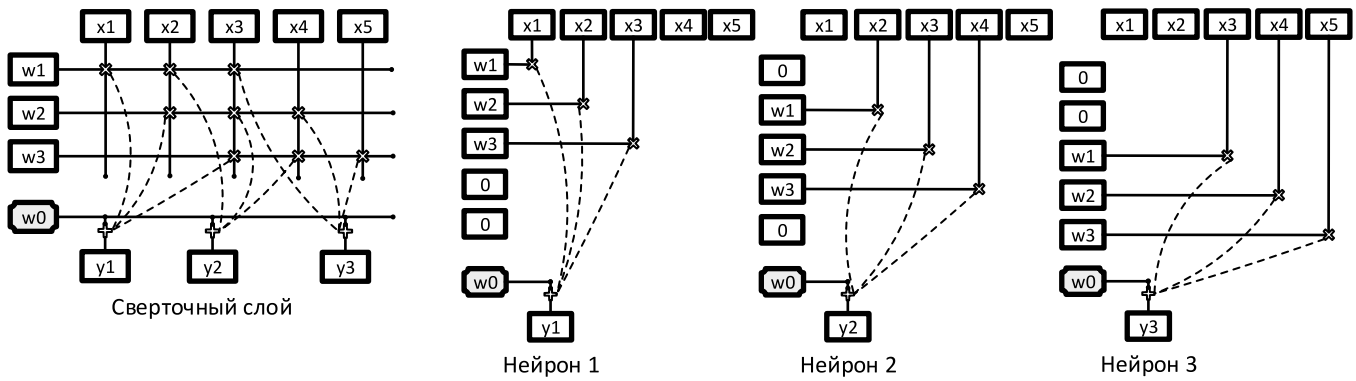
Схематично эта операция представлена на рис. 1, *а*.

В одном сверточном слое может обрабатываться сразу несколько разных ядер, тогда выход сверточного слоя является совокупностью выходов каждого из ядер.

Б. Полносвязные слои — это слои, состоящие из нейронов, каждый из которых умножает элементы своего входа на свой весовой коэффициент, складывает эти произведения и добавляет смещение. Входы во все нейроны одного слоя одинаковые. Пусть x_i — i-й элемент входного вектора, $i = 1...N$, где N — число входных параметров; $w_{i,j}$ — i-й весовой коэффициент j-го нейрона; $w_{0,j}$ _ смещение j-го нейрона, тогда выход полносвязного слоя определяется как

$$y_j = \sum_i (x_i * w_{i,j}) + w_{0,j}.$$

Схематично эта операция представлена на рис. 1, *б*.



а) *б*)

Рис. 1. Пример сверточного (*а*) и эквивалентного ему полносвязного (*б*) слоев нейронной сети для пяти входов и трех выходов: x — умножитель; + — сумматор

Сравнив работу полносвязного и сверточного слоев, можно понять, что сверточный слой фактически является вариантом полносвязного слоя, в котором некоторые коэффициенты одинаковы для всех нейронов и многие из них равны 0 (что эквивалентно тому, что отсутствует связь от определенного входа, пример показан на рис. 1). За счет того что сверточные слои используют гораздо меньшее число параметров, чем полносвязные, но при этом их совокупность позволяет строить довольно сложные отношения вход-выход, они и получили широкое развитие.

Рассмотрим те типы функций активации, которые будем использовать в нашей работе.

1. *Сигмоидальная функция.* Эта функция описывается выражением: выход $y = 1/(1 + e^{-x})$ для каждого входа x независимо.

2. *Softmax-функция.* Функция преобразует вектор входов x размерности N в вектор выходов y той же размерности, где каждый компонент выхода y_j полученного вектора представлен вещественным числом в интервале [0,1] и сумма компонент равна единице: $y_j = \exp(x_j)/\left[\sum_i \exp(x_i)\right]$. Представим требуемые выходы нейронной сети в виде бинарного вектора с числом компонент, равным числу классов, и в котором наличие единицы обозначает, к какому классу относится тот или иной пример. Тогда выходы этой функции активации можно трактовать как степень уверенности нейронной сети в том, что текущий пример данных имеет определенный класс. Обычно эта функция используется на последнем слое нейронной сети.

Вообще говоря, в сверточной нейронной сети могут присутствовать и другие типы слоев (например, слои с радиально-базисными нейронами, необучаемыми свертками и т.д.), но поскольку в нашей работе мы их не использовали, то не будем и описывать.

Сверточная сеть в целом состоит из набора чередующихся сверточных слоев и слоев активации, последние слои обычно полносвязные, также чередуются со слоями активации. В нашей работе использована сеть, схематичное изображение которой представлено на рис. 2.

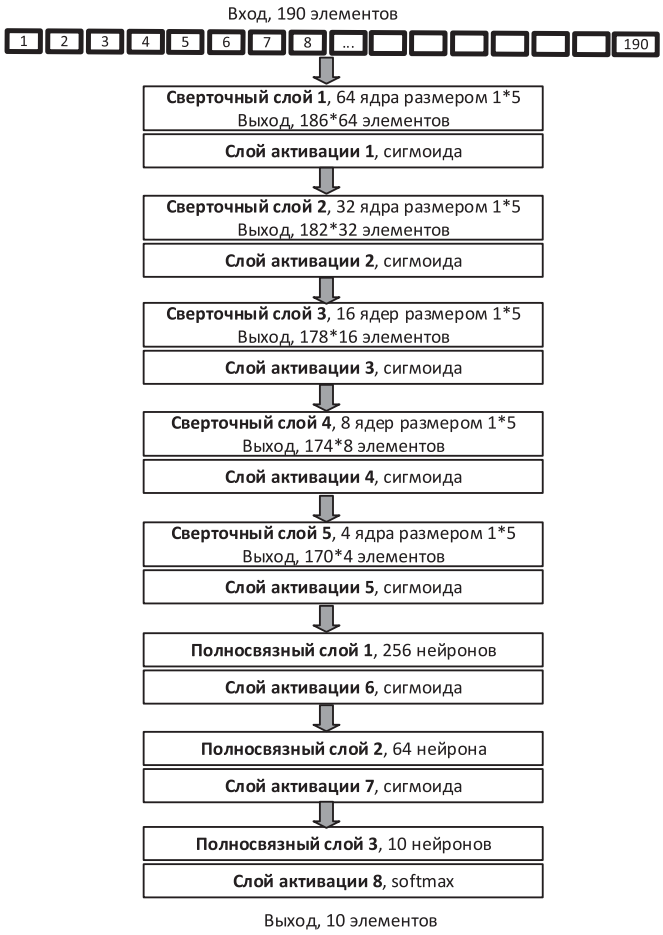


Рис. 2. Структура используемой нейронной сети

2.2. Преобразование входных и выходных данных

Для того чтобы применить сверточную нейронную сеть для задачи классификации типов компьютерных атак, необходимо сначала представить данные из используемой базы данных в числовом виде, для чего нужно провести кодирование нечисловых данных, а также выполнить некоторую предобработку для числовых данных.

В используемой нами базе данных UNSW-NB15 каждый пример представляется 47 признаками, из них мы отбросим признаки, относящиеся только к той компьютерной сети, на которой собиралась база данных, а именно, следующие:

— srcip (source IP address) — IP-адрес источника,

— dstip (destination IP address) — IP-адрес получателя,

— sport (Source port number) — порт источника,

— dsport (destination port number) — порт получателя,

— stime (record start time) — время (абсолютное) начала записи,

— ltime (record last time) — время (абсолютное) последней записи,

— res_bdy_len — имеет отношение только к http сервису, не используется.

Из оставшихся признаков три являются категориальными:

— proto (Transaction protocol) — протокол, к которому относится сетевое соединение, может принимать одно из 129 значений;

— servis, указывает, какой сервис используется в сетевом соединении, может принимать значения: http, ftp, smtp, ssh, dns, ftp-data, irc и "-", если не используется;

— state, определяет состояние протокола, в зависимости от протокола может принимать значения: ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN и "–", если у протокола нет состояния.

Каждый из этих трех признаков будем кодировать с помощью бинарного вектора, число элементов в котором равно числу возможных значений признака. Все, кроме одного элемента такого вектора, нулевые, не нулевой элемент (равный единице) показывает, какое значение принял признак.

После кодирования число признаков стало 190 (37 — без изменений, 16 — для кодирования state, 8 — для servis и 129 — для proto).

Таблица 2

Масштабирование признаков

Название	Значение	Делитель
dbytes	Число байт переданных от получателя к источнику	100 000
dinpkt	Средняя задержка сетевых пакетов на получателе, мс	1000
djit	Джиттер (средние отклонения в задержке) сетевых пакетов на получателе, мс	1000
dload	Нагрузка получателя, бит/с	100 000
dloss	Число пакетов отброшенных или повторно переданных для получателя	10
dmean	Средний размер пакета от получателя, байт	10
dpkts	Число пакетов получатель-источник	100
dtcpb	Длина базы пакета TCP протокола получателя	10 000 000
sinpkt	Средняя задержка сетевых пакетов на источнике, мс	1000
sjit	Джиттер (средние отклонения в задержке) сетевых пакетов на источнике, мс	10 000
sload	Нагрузка источника, бит/с	10 000 000
sloss	Число пакетов отброшенных или повторно переданных для источника	10
smean	Средний размер пакета от источника, байт	10
spkts	Число пакетов источник-получатель	100
stcpb	Длина базы пакета TCP протокола источника	10 000 000

Для оставшихся признаков проведены следующие преобразования:

— все бинарные признаки оставлены без изменений,

— часть признаков, приведенных в табл. 2, отмасштабированы путем деления на коэффициент,

— остальные признаки оставлены без изменений.

2.3 Формирование обучающей выборки с балансировкой классов и без

Выходы нейронной сети представляют собой 10-мерные вектора, по числу классов: девять атак (Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Worms) и класс нормальных пакетов.

Нейронная сеть обучается алгоритмом RMS-prop библиотеки keras [7], одним из вариантов метода градиентного спуска, при этом обучение ведется на небольших батчах (batch, пакет) в двух вариантах, схематично показанных на рис. 3:

— батчи формируются из всех имеющихся примеров для обучения случайно, число примеров разных классов в батче не контролируется;

— число примеров разных классов в батче задано (в нашей работе одинаково), внутри класса примеры выбираются случайно.

Далее мы сравним результаты классификации этих двух вариантов.

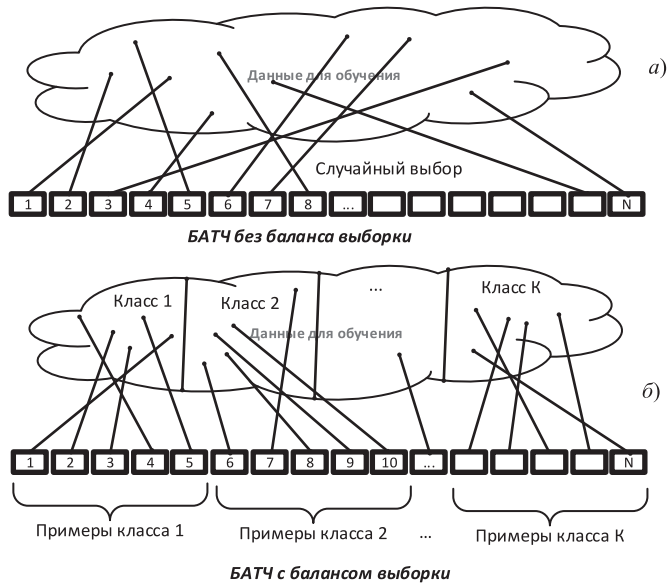


Рис. 3. Пояснения к способам формирования обучающей выборки:

а — формирование батча без баланса; *б* — формирование батча с балансом классов в выборке

3. Вычислительные эксперименты

Для создания, обучения, отображения результатов использованы следующие программные средства:

- язык программирования: Python 3.5 в составе пакета Anaconda 4.2.0;
- среда разработки: Microsoft Visual Studio 2017, v15.5.2;
- библиотека обучения нейронных сетей: keras 2.0.2 на основе tensorflow (GPU версия) 1.1.0;
- вспомогательные библиотеки: pandas 0.20.1 — для работы с текстовыми файлами базы данных; numpy 1.13.1 — для расчетов, scikit-learn 0.18.1 — для преобразования данных (кодирование), matplotlib 2.0.2 — для отображения графиков;

и следующие аппаратные средства:

- ЭВМ с процессором IntellCore i7, 3.4 ГГц, 32 Гб оперативной памяти, под управлением операционной системы Microsoft Windows 7 Pro,
- графический ускоритель (GPU): Nvidia Quadro K600, 1 Гб.

В рамках экспериментов использованы такие параметры нейронной сети:

— размер батча batch_size — 256;

— максимальное число эпох (итераций) обучения — 200;

— шаг в алгоритме обучения $l_r = 0{,}001$.

Прочие параметры по умолчанию библиотеки keras [7].

Обучение велось в двух вариантах, с балансировкой обучающей выборки и без. Обучение повторялось несколько раз, приведены средние результаты.

На рис. 4 и 5 представлены результаты экспериментов. На рис. 4 приведены матрицы потерь для всех классов. Матрица потерь показывает, сколько данных было отнесено к тому или иному классу, по вертикальной оси приведены истинные классы данных, по горизонтальной оси — классы, предсказанные нейронной сетью на тестовых данных (отличных от обучающих). Правильная классификация отображена диагональными элементами матрицы потерь, неправильная классификация — недиагональными элементами. Результаты представлены в нормализованном виде, т.е. в долях от числа данных конкретного класса. Перед метками

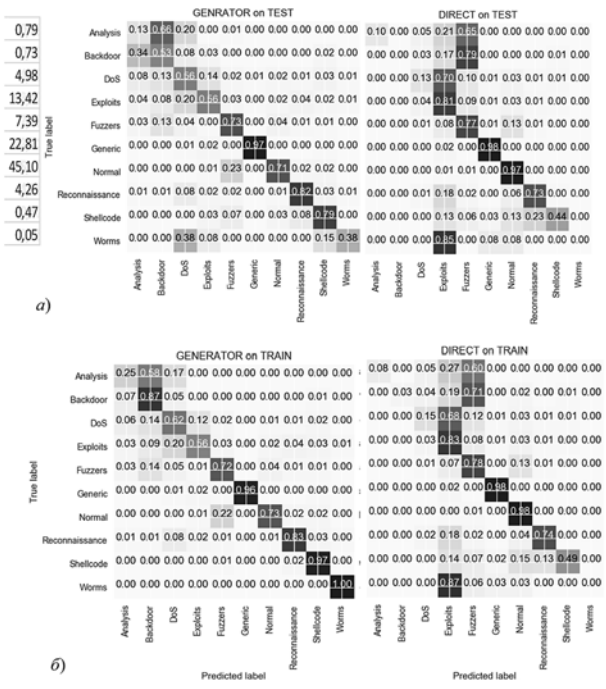


Рис. 4. Результаты экспериментов — матрицы потерь:
а — матрицы потерь на тестовых данных; слева (GENERATOR on TEST) — при использовании балансировки классов, справа (DIRECT on TEST) — без балансировки; *б* — матрицы потерь на обучающих данных; слева (GENERATOR on TRAIN) — при использовании балансировки классов, справа (DIRECT on TRAIN) — без балансировки

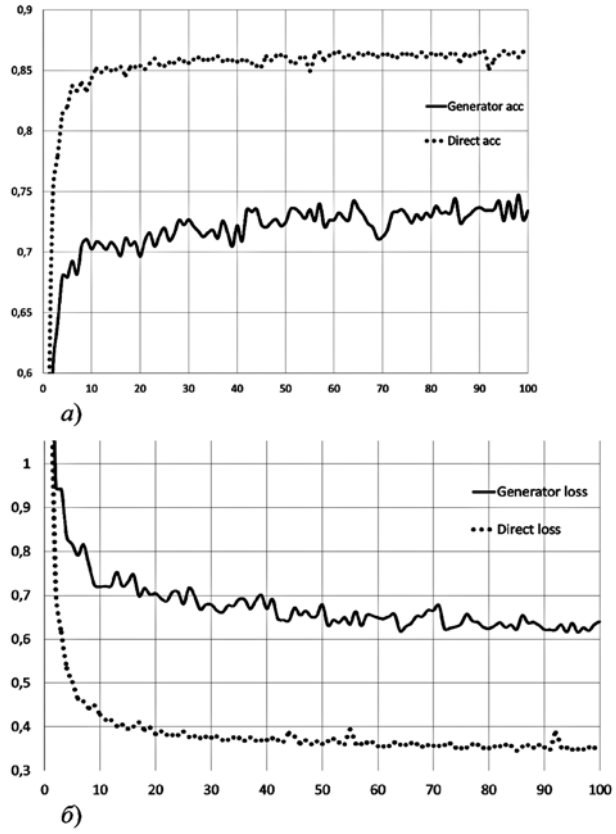


Рис. 5. Результаты экспериментов (первые 100 итераций):
а — точность (accuracy), для способа без балансировки (пунктирная линия, Direct acc) и с балансировкой (сплошная линия, Generator acc); *б* — функция потерь (loss) для способа без балансировки (пунктирная линия, Direct loss) и с балансировкой (сплошная линия, Generator loss)

классов указано число данных каждого класса в процентах.

На рис. 5 приведены графики изменения точности (accuracy) классификации и функции потерь (losses). Точность вычисляется как отношение числа правильно распознанных примеров к общему числу примеров в тестовой выборке, функция потерь — это функция, которая оптимизируется в процессе обучения нейронной сети, в нашем случае это функция кросс-энтропии (categorical_crossentropy, см [7]).

Анализируя результаты, можно сделать вывод, что применение балансировки помогает улучшить качество классификации данных с малым числом примеров. Отсутствие балансировки ведет к тому, что классы с малым числом примеров могут быть полностью проигнорированы нейронной сетью. Из рис. 5 видно: несмотря на то, что без использования балансировки была достигнута более высокая точность (accuracy) и меньшее значение функции потерь (loss), качество распознавания классов с малым числом примеров довольно мало, классы Worm, Backdoor не распознались вовсе.

Отметим также, что ценой за лучшее качество распознавания классов с малым числом примеров стало снижение качества распознавания классов с большим числом примеров (Exploits, Normal).

Заключение

В работе показан подход к балансировке классов в задачах классификации с разным числом данных в классах, такой подход позволяет повысить качество классификации. В дальнейшем планируется повысить качество классификации за счет усложнения нейронной сети (увеличения числа слоев, подбора более оптимальных параметров слоев), добавления самообучающихся слоев.

Авторы выражают благодарность всем коллегам за обсуждение результатов работы и особенно Научному фонду НИУ ВШЭ, поскольку публикация подготовлена в ходе проведения исследования (№ 16-01-0012 "Исследование применения сверточной нейронной сети для классификации типов компьютерных атак, используя базу данных компьютерных атак") в рамках Программы "Научный фонд Национального исследовательского университета "Высшая школа экономики" (НИУ ВШЭ)" в 2016—2018 гг. и в рамках государственной поддержки ведущих университетов Российской Федерации "5-100".

Список литературы

1. **Tavallaee M., Bagheri E., Lu W., Ghorbani A.** A detailed analysis of the KDD CUP 99 data set // Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009 IEEE Symposium on. IEEE, 2009. pp. 1—6.

2. **Shiravi A., Shiravi H., Tavallaee M., Ghorbani A. A.** Toward developing a systematic approach to generate benchmark datasets for intrusion detection // Computers & Security. 2012. V. 31. No. 3. P. 357—374.

3. **Sharafaldin I., Lashkari A. H., Ghorbani A. A.** Toward generating a new intrusion detection dataset and intrusion traffic characterization // 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal. January 2018. P. 108—116.

4. **Nour M., Slay J.** UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set) // Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015. P. 1—6.

5. **Nour M., Slay J.** The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set // Information Security Journal: A Global Perspective. 2016. Vol. 25, N. 1—3. P. 18—31.

6. **KDD** Cup 1999 Data [Электронный ресурс]. URL: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (дата обращения: 12.01.2018).

7. **Keras:** The Python Deep Learning Library [Электронный ресурс]. URL: https://keras.io/ (дата обращения: 12.01.2018).

8. **Kumar S., Spafford E. H.** A pattern matching model for misuse intrusion detection // Proceedings of the 17th National Computer Security Conference. Baltimore MD, USA. 1995. P. 11—21.

9. **Kumar S., Spafford E. H.** An application of pattern matching in intrusion detection. The COAST Project, Department of Computer Sciences, Purdue University, West Lafayette, IN. USA, Technical Report CSD-TR-94-013. 1994. 57 p.

10. **Teng H. S., Chen K., Lu S. C.** Adaptive real-time anomaly detection using inductively generated sequential patterns // Proceedings 1990 IEEE Computer Society Symposium on Research in Security and Privacy, 1990. IEEE, 1990. P. 278—284.

11. **Ko C. C. W.** Execution monitoring of security-critical programs in a distributed system: a specification-based approach. PhD thesis, University of California, Davis, 1996. 118 p.

12. **Eckmann S. T., Vigna G., Kemmerer R. A.** STATL: An attack language for state-based intrusion detection // Journal of Computer Security. 2002. Vol. 10, No. 1—2. P. 71—103.

13. **Ho Y.** Partial order state transition analysis for an intrusion detection system // Master's thesis, University of Idaho, USA, 1997.

14. **Ilgun K., Kemmerer R. A., Porras P. A.** State transition analysis: a rule-based intrusion detection approach // IEEE Transactions on Software Engineering. 1995. V. 21, No. 3. P. 181—199.

15. **Sheyner O. M**. Scenario Graphs and Attack Graphs. PhD thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh PA, USA, 2004.

16. **Sebring M., Shellhouse E., Hanna M., Whitehurst R.** Expert systems in intrusion detection: a case study // Proc. 11th National Computer Security Conference, Baltimore, Maryland, Oct. 1988. 1988. P. 74—81.

17. **Whitehurst R. A.**, Expert systems in intrusion detection: a case study. Report of Computer Science Laboratory, SRI International, Menlo Park, CA, USA, 1987.

18. **Smaha S. E.** Haystack: An intrusion detection system // Aerospace Computer Security Applications Conference, 1988, Washington, D. C., USA. IEEE 1988. P. 37—44.

19. **Mukkamala S., Sung A. H., Abraham A.** Intrusion detection using an ensemble of intelligent paradigms // Journal of Network and Computer Applications. 2005. Vol. 28, No. 2. P. 167—182.

20. **Hofmeyr S. A., Forrest S.** An immunological model of distributed detection and its application to computer security. PhD thesis, University of New Mexico. 1999.

21. **Zielinski M., Venter L.** Applying mobile agents in an immune-system-based intrusion detection system: reviewed article // South African Computer Journal. 2005. No. 34. P. 76—83.

22. **Debar H., Becker M., Siboni D.** A neural network component for an intrusion detection system // Proceedings of 1992 IEEE Computer Society Symposium on Research in Security and Privacy, 1992. IEEE 1992. P. 240—250.

23. **Jalili R.** et al. Detection of distributed denial of service attacks using statistical pre-processor and unsupervised neural networks // International Conference on Information Security Practice and Experience. Springer, Berlin, Heidelberg, 2005. P. 192—203.

D. V. Pantiukhin, Senior Lecturer, e-mail: dpantiukhin@hse.ru,
E. Karelova, Student, e-mail: ekarelova@edu.hse.ru,
National Research University Higher School of Economics, Moscow

Quality Improvement of Intrusion Classification through Convolution Neural Network by Training on Balanced Samples

This research investigates the effects of training sample balancing while solving intrusion classification task with convolution neural network. Using two convolutional neural networks with similar architecture, we conduct comparative analysis of classification task solution quality with and without training sample balancing. Experiments illustrate the efficiency of using training sample balancing in case of significant differences in the amount of samples in different classes.

Keywords: *intrusion detection, convolution neural network, training sample balancing, intrusion detection datasets*

References

1. **Tavallaee M., Bagheri E., Lu W., Ghorbani A.** A detailed analysis of the KDD CUP 99 data set, *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009 IEEE Symposium on*. IEEE, 2009. pp. 1—6.

2. **Shiravi A., Shiravi H., Tavallaee M., Ghorbani A. A.** Toward developing a systematic approach to generate benchmark datasets for intrusion detection, *Computers & Security*. 2012. vol. 31, no. 3, pp. 357—374.

3. **Sharafaldin I., Lashkari A. H., Ghorbani A. A.** Toward generating a new intrusion detection dataset and intrusion traffic characterization, 4th International Conference on Information Systems Security and Privacy (ICISSP), Portugal, January 2018, pp. 108—116.

4. **Nour M., Slay J.** UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), Military Communications and Information Systems Conference (MilCIS), 2015, IEEE, 2015, pp. 1—6.

5. **Nour M., Slay J.** The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set, *Information Security Journal: A Global Perspective*, 2016, vol. 25, no. 1—3, pp. 18—31.

6. **KDD** Cup 1999 Data, available at: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (date of access: 12.01.2018).

7. **Keras:** The Python Deep Learning Library, available at: https://keras.io/ (date of access: 12.01.2018).

8. **Kumar S., Spafford E. H.** A pattern matching model for misuse intrusion detection, *Proceedings of the 17th National Computer Security Conference*. Baltimore MD, USA, 1995, pp. 11—21.

9. **Kumar S., Spafford E. H.** An application of pattern matching in intrusion detection, The COAST Project, Department of Computer Sciences, Purdue University, West Lafayette, IN, USA, Technical Report CSD-TR-94-013, 1994, 57 p.

10. **Teng H. S., Chen K., Lu S. C.** Adaptive real-time anomaly detection using inductively generated sequential patterns, Proceedings 1990 IEEE Computer Society Symposium on Research in Security and Privacy, 1990, IEEE, 1990, pp. 278—284.

11. **Ko C. C. W.** Execution monitoring of security-critical programs in a distributed system: a specification-based approach, PhD thesis, University of California, Davis, 1996, 118 p.

12. **Eckmann S. T., Vigna G., Kemmerer R. A.** STATL: An attack language for state-based intrusion detection, *Journal of Computer Security*, 2002, vol. 10, no 1—2, pp. 71—103.

13. **Ho Y.** Partial order state transition analysis for an intrusion detection system, Master's thesis, University of Idaho, USA, 1997.

14. **Ilgun K., Kemmerer R. A., Porras P. A.** State transition analysis: a rule-based intrusion detection approach, *IEEE Transactions on Software Engineering*, 1995, vol. 21, no. 3, pp. 181—199.

15. **Sheyner O. M.** Scenario Graphs and Attack Graphs, PhD thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh PA, USA, 2004.

16. **Sebring M., Shellhouse E., Hanna M., Whitehurst R.** Expert systems in intrusion detection: a case study, *Proc. 11th National Computer Security Conference*, Baltimore, Maryland, Oct. 1988, 1988, pp. 74—81.

17. **Whitehurst R. A.** *Expert systems in intrusion detection: a case study,* Report of Computer Science Laboratory, SRI International, Menlo Park, CA, USA, 1987.

18. **Smaha S. E.** Haystack: An intrusion detection system, *Aerospace Computer Security Applications Conference*, 1988, Washington, D. C., USA. IEEE 1988, pp. 37—44.

19. **Mukkamala S., Sung A. H., Abraham A.** Intrusion detection using an ensemble of intelligent paradigms, *Journal of Network and Computer Applications*. 2005, vol. 28, no. 2, pp. 167—182.

20. **Hofmeyr S. A., Forrest S.** An immunological model of distributed detection and its application to computer security, PhD thesis, University of New Mexico. 1999.

21. **Zielinski M., Venter L.** Applying mobile agents in an immune-system-based intrusion detection system: reviewed article, *South African Computer Journal*, 2005, no. 34, pp. 76—83.

22. **Debar H., Becker M., Siboni D.** A neural network component for an intrusion detection system, *Proceedings of 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, 1992. IEEE 1992, pp. 240—250.

23. **Jalili R., Imani-Mehr F., Amini M., Shahriari H. R.** Detection of distributed denial of service attacks using statistical pre-processor and unsupervised neural networks, *International Conference on Information Security Practice and Experience*, Springer, Berlin, Heidelberg, 2005, pp. 192—203.