Санкт-Петербургский политехнический университет Петра Великого

Институт прикладной математики и механики

**Кафедра «Информационная безопасность компьютерных систем»**

**Отчет**

**о научно-исследовательской работе**

**Обнаружение сетевых атак с помощью LSTM сетей**

Выполнил

студент гр.43609/1 Д.А. Куликов

Руководитель

доцент, к.т.н. В.В. Платонов

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2019 г.

Санкт-Петербург

1. 2019

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc12494027)

[1. RNN и LSTM сети 6](#_Toc12494028)

[1.1. RNN 8](#_Toc12494029)

[1.2. LSTM 9](#_Toc12494030)

[2. Описание данных и их обработка 12](#_Toc12494031)

[3. Реализация и анализ LSTM сети 16](#_Toc12494032)

[3.1. Бинарный классификатор на основе LSTM 17](#_Toc12494033)

[3.2. Классификатор типов атак на основе LSTM 22](#_Toc12494034)

[4. Анализ необходимости памяти сети 26](#_Toc12494035)

[Заключение 29](#_Toc12494036)

[Список использованных источников 30](#_Toc12494037)

# Введение

Нарушители со временем используют все более изощренные и сложные методы атак, поэтому отличить их от нормального трафика становится все труднее и труднее. Так как сигнатурные методы не всегда могут выявить новые атаки, необходимо применять методы выявления аномального трафика. Благодаря своим способностям к обучению, с этой целью могут справиться нейронные сети и другие инструменты машинного обучения. Также широко применяется data mining, для выявления значимых признаков, по которым можно классифицировать трафик.

В данной работе реализуется выявление аномального трафика с помощью LSTM сетей, или сетей с долгой краткосрочной памятью. Также проводится их описание и анализ результатов.

Для обучения и оценки работоспособности сети использовались данные UNSW-NB15 [1], опубликованные в 2018 году. Также, для проверки и сравнения использовались KDD Cup данные 2000 года [2].

Целью данной работы является исследование возможности применения LSTM сетей для обнаружения сетевых атак. Реализуется как бинарный классификатор, так и классификатор атак по типам.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* изучить механизм работы LSTM сетей;
* провести предварительную обработку данных, выбрать архитектуру и параметры сети;
* выполнить программную реализацию бинарного классификатора с помощью LSTM, обучить сеть и проанализировать результаты;
* выполнить программную реализацию классификатора с разделением атак на типы с помощью LSTM, обучить сеть и проанализировать результаты.

# RNN и LSTM сети

Сетевые протоколы работают как функция от времени, поэтому используются рекуррентные нейронные сети (RNN), которые способны учитывать зависимость между данными на разных временных промежутках.

Связи в RNN проходят не только между нейронами от входного слоя к выходному, но и от нейрона к «самому себе», а именно, к его предыдущему значению или предыдущим значениям других нейронов того же слоя.

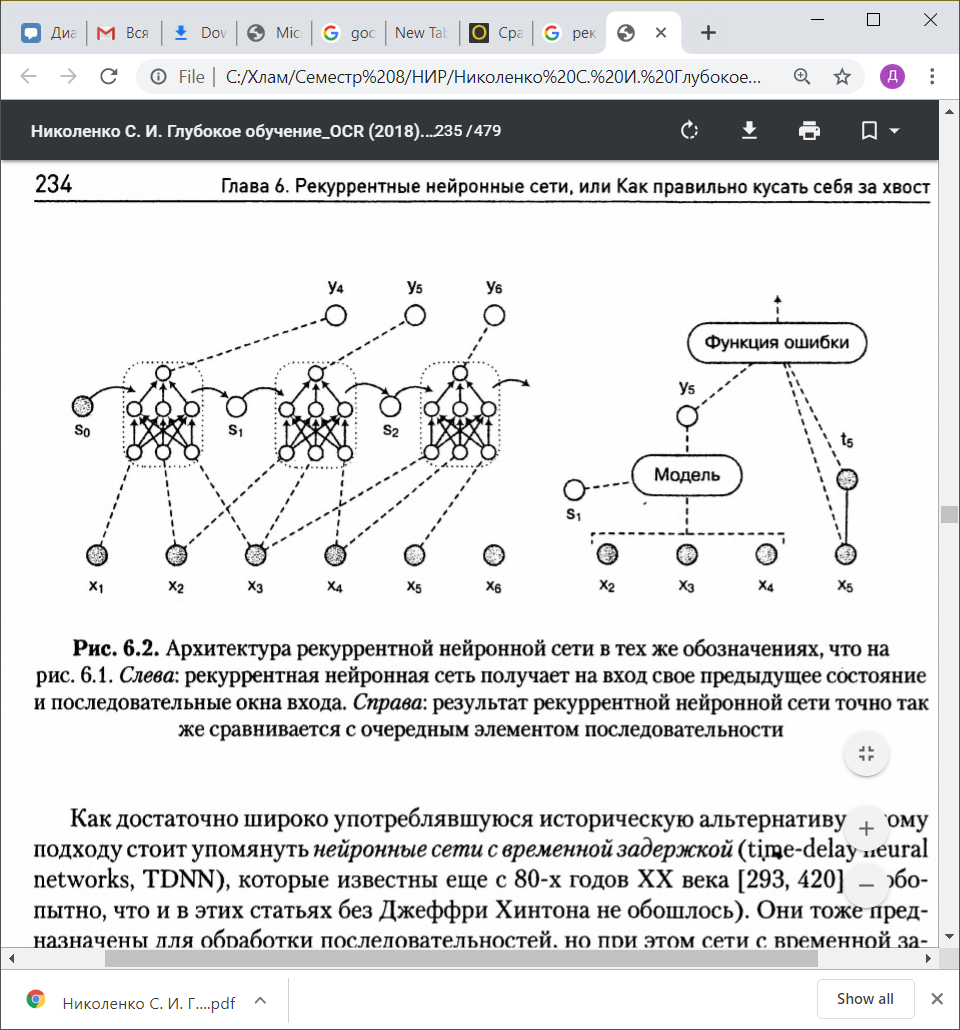


Рисунок 1 – Архитектура рекуррентной нейронной сети [3]

На рисунке 1 отображает скрытое состояние нейрона в период времени , которое зависит от того что происходило ранее.

Выделяют пять типов задач по отношению входов к выходам:

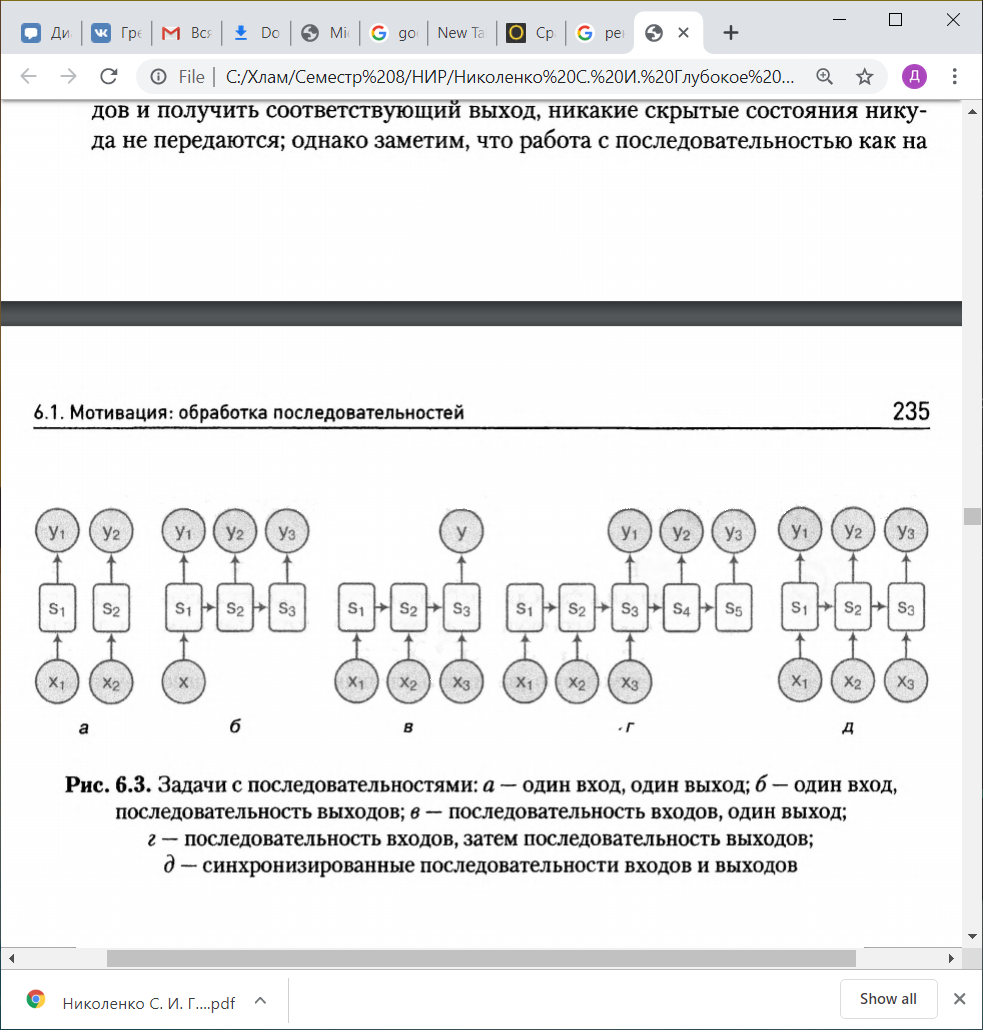


Рисунок 2 – Задачи, решаемые рекуррентными сетями [3]

На рисунке 2 графически представлены следующие схемы:

* Один вход, один выход; обычная схема, здесь внутренние состояния никуда не передаются, следовательно, ничем не отличается от обычной сети.
* Один вход, много выходов; используется, например, когда генерируется описание на основе картинки.
* Один выход, много входов; такие задачи применяются для классификации последовательностей.
* Много входов, а потом много выходов; по этой схеме работают интеллектуальные переводчики.
* Много входов, много выходов, синхронизированные.

## RNN

Рассмотрим архитектуру обычной рекуррентной сети:

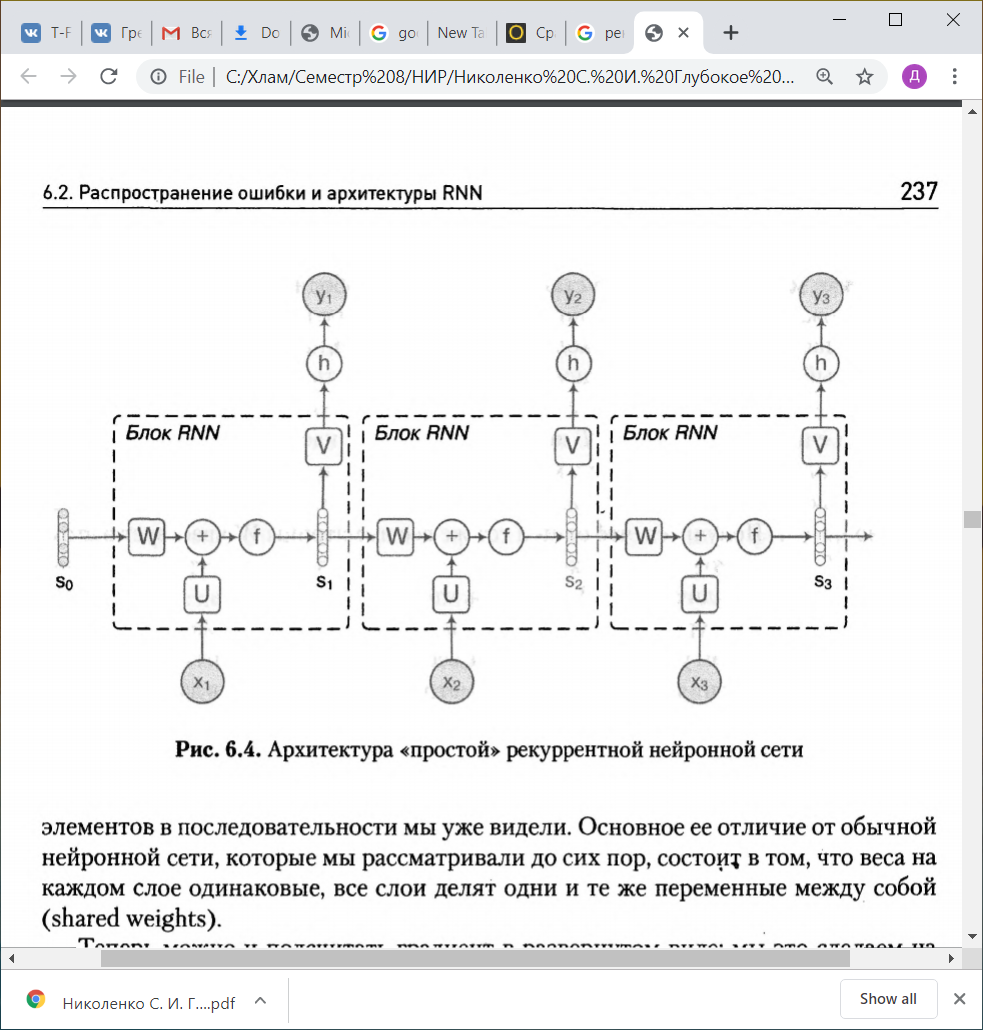


Рисунок 3 – Архитектура обычной рекуррентной нейронной сети [3]

Пусть – матрица весов для входа состояния, – матрица входных весов последовательности, – веса выходов, функция выхода (например, softmax), нелинейная функция.

Тогда в момент времени :

Если прошло шагов, , и задана функция ошибки Тогда можно посчитать частные производные, разворачивая состояния Таким образом, каждый вес из матрицы будет встречаться раз. Таким образом, можно представить рекуррентную сеть, как многоуровневую обычную сеть, в которой одни и те же веса используются на каждом уровне.

Недостатки:

* Взрывные градиенты. Если матрица весов такова, что значительно увеличивает норму градиента при проходе через один развернутый слой, то при проходе через слоев эта норма возрастет экспоненциально от .
* Также влияние текущего входа затухает экспоненциально по мере удаления, что не позволяет распознать далекие зависимости в данных.

## LSTM

В LSTM узлы называются гейтами: входной, забывающий и выходной, а также рекуррентная ячейка со скрытым состоянием.

Обозначив входной вектор во время , – вектор скрытого состояния во время , – матрицы весов ко входу, - матрицы весов в рекуррентных соединениях, – векторы свободных членов, получим формальный вид работы LSTM:

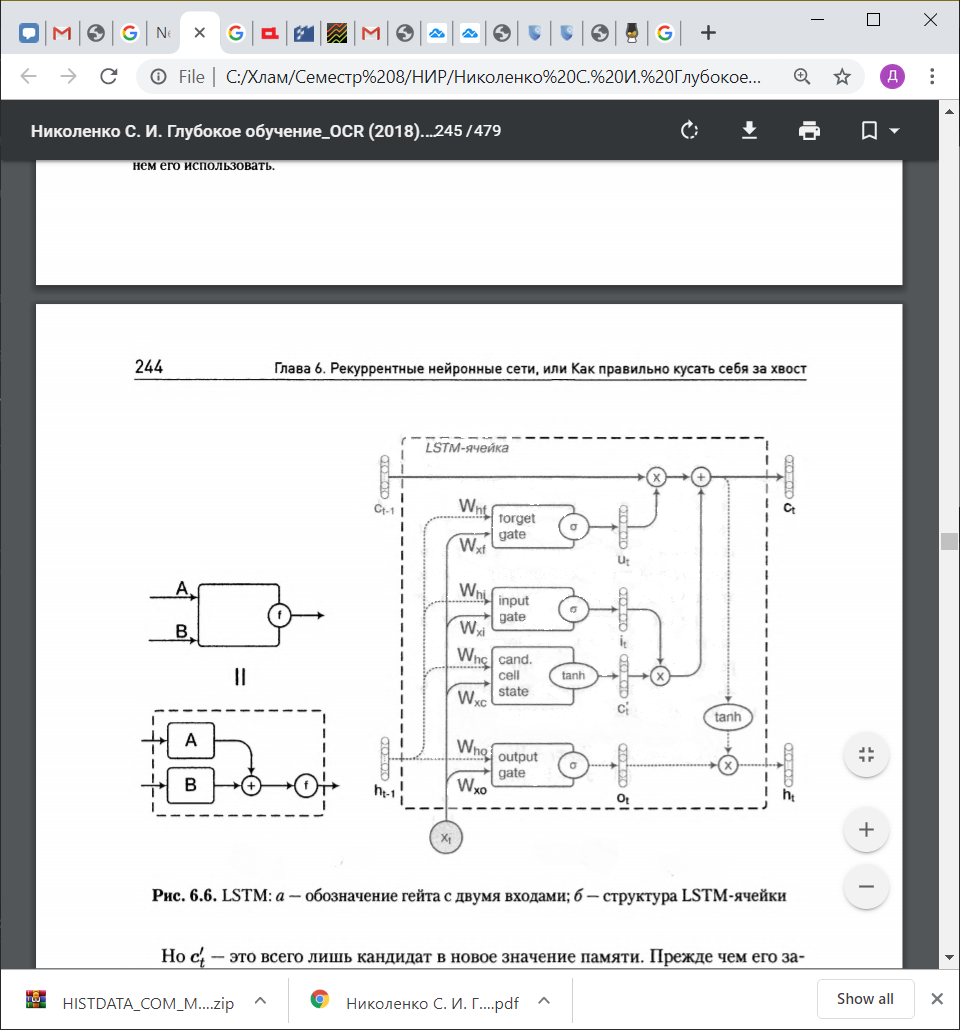


Рисунок 4 – Структура LSTM ячейки [3]

Также, как и в RNN, на вход подается вектор входных данных и вектор скрытого состояния . Функцию памяти выполняет вектор .

Из формулы можно увидеть, что новое значение ячейки памяти получается, как линейная комбинация из старого, умноженного на выход забывающего гейта и нового кандидата с коэффициентами из входного гейта

Таким образом, если значения малы, то значение ячейки памяти «забудется», а если велики, то к ним добавятся новые данные. Благодаря своим свойствам, на каждом шаге может быть изменена только часть памяти ячейки, другая часть памяти может сохраниться, это определяется матрицами весов и делает структуру ячейки гибкой.

LSTM позволяет решить проблему исчезающих градиентов. Для простоты положим для всех . Тогда:

Данный эффект называется «каруселью константной ошибки», это означает, что скрытое состояние может сохраняться неограниченно долго, если не будут перезаписаны. Таким образом, LSTM может запоминать контекст на гораздо более продолжительное время чем RNN, при необходимости. От взрывающихся же градиентов LSTM также, как и RNN не защищены.

Существует множество модификаций LSTM, вот некоторые из них [3]:

1. LSTM без входного гейта;
2. LSTM без забывающего гейта;
3. LSTM без выходного гейта ;
4. LSTM без функции активации на входном гейте;
5. LSTM без функции активации на выходном гейте;
6. LSTM без замочных скважин;
7. LSTM со связанными входным и забывающим гейтом;
8. LSTM с дополнительными рекуррентными связями на каждом гейте;

# описание данных и их обработка

Сетевые пакеты UNSW-NB15 были получены авторами [1] инструментом IXIA PerfectStorm в лаборатории Cyber Range Австралийского центра кибербезопасности (ACCS) с целью получить нормальный и зловредный профиль поведения современной сети.

Используемые данные делятся на нормальный трафик и трафик с аномалиями. Аномальный трафик подразделяется на 9 типов:

* Эксплоиты – фрагмент программного кода или последовательность команд, которые используются уязвимости в программном обеспечении и применяются для осуществления атаки на вычислительную систему.
* Разведка – получение и обработка [данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5) об [информационной системе](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), ресурсах информационной системы, используемых устройствах и [программном обеспечени](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)и и их [уязвимостях](https://ru.wikipedia.org/wiki/%D0%A3%D1%8F%D0%B7%D0%B2%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D1%8C_(%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D1%8C)), средствах защиты, а также о границе проникновения в информационную систему.
* Атака отказа в обслуживании – атака на вычислительную систему с целью довести её до отказа, то есть создание таких условий, при которых добросовестные пользователи системы не могут получить доступ к предоставляемым системным ресурсам (серверам), либо этот доступ затруднён.
* Стандартные атаки – атаки носящие массовый характер.
* Шелл-код – попытка загрузки исполняемого кода, который передаёт управление командному процессору.
* Использование фаззинга – метода тестирования программного обеспечения, заключающегося в передаче приложению на вход неправильных, неожиданных или случайных данных, с целью найти в нем ошибки и уязвимости.
* Черви – атака сетевыми червями, которые используют сетевые протоколы и устройства для распространения.
* Потайные ходы – атака, при которой на компьютере пользователя открывается канал передачи данных, доступный нарушителю.
* Анализ сети – разведка с целью построения карты сети информационной системы.

Всего, данные содержат признаков, из которых относятся к маркировке трафика. Среди прочих – сервисы (около шт.), IP-адреса, количественные характеристики потока трафика, флаги (в зависимости от протокола), порты, размеры пакетов, временные характеристики трафика и количественные характеристики соединений.

После выделения строковых признаков, они кодируются вещественными числами, далее проводится min-max масштабирование, для приведения значений данных в диапазон

Из обработанных, пакетов с маркировкой нормального трафика – . Пакетов с маркировкой атаки – . Из этого можно сделать вывод, что данные сильно несбалансированные, откуда следует необходимость увеличения весов функции ошибки для аномального трафика.

Аномальный трафик распределяется следующим образом:

Таблица 1

Количество пакетов с разными типами атак

|  |  |
| --- | --- |
| Атака | Количество пакетов |
| Эксплоиты | 33086 |
| Разведка | 10457 |
| Отказ в обслуживании | 11446 |
| Стандартные | 153603 |
| Шелл-код | 1140 |
| Фаззинг | 18856 |
| Черви | 131 |
| Потайные ходы | 1663 |
| Анализ сети | 2007 |

Пропорционально этим данным, для классификации по типам атак были увеличены веса функции ошибки:

Таблица 2

Веса для функции ошибки

|  |  |
| --- | --- |
| Атака | Множитель |
| Эксплоиты | 55 |
| Разведка | 172 |
| Отказ в обслуживании | 157 |
| Стандартные | 12 |
| Шелл-код | 1580 |
| Фаззинг | 95 |
| Черви | 13000 |
| Потайные ходы | 1080 |
| Анализ сети | 895 |

В данных KDD 2000 [2] пакетов с атаками и пакетов нормального трафика. Каждый пакет имеет 42 признака, среди которых тип протокола, сервис, флаги, количество неверных фрагментов, состояние сессии клиента, проценты ошибок передачи, количественные характеристики трафика. Данные сбалансированные, поэтому дополнительные веса применять не требуется.

# реализация и анализ lstm сети

Реализация выполнялась на языке Python, с помощью открытой нейросетевой библиотеки Keras [4] с TensorFlow. Keras предоставляет реализацию LSTM слоя и выбор следующих параметров:

1. Units – количество LSTM ячеек в слое (соответствует размерности выхода).
2. Return\_sequence – позволяет либо получать на выходе только последнее значение из выходной последовательности, либо все значения (в данном случае используются все).
3. Stateful – позволяет либо запоминать скрытое состояние ячеек при каждом новом входном значении, либо обнулять его.

Этот набор параметров позволяет использовать сеть для различных задач, описанных ранее.

Также, для ускорения вычислений, стоит использовать Keras на GPU, что позволяет значительно снизить время ожидания, например, при обучении.

## Бинарный классификатор на основе LSTM

Для данных из KDD 2000 [2], была реализована сеть, которую можно представить в следующем виде:

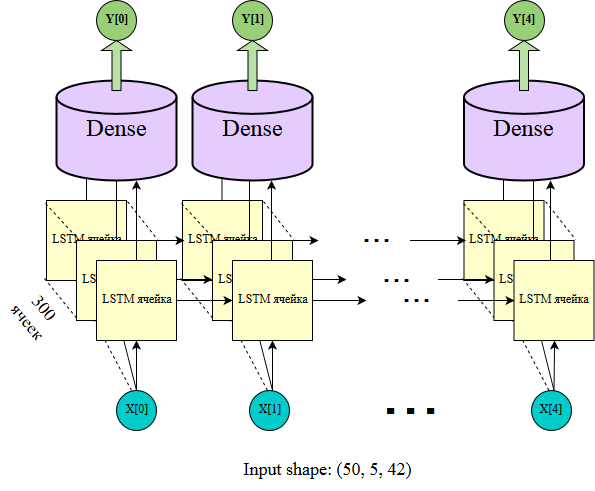


Рисунок 5 – Представление работы сети с одним LSTM слоем

На рисунке 5 изображена работа сети с одним LSTM слоем, ячейками, временными интервалами и признаками входных данных (размерность векторов , а также размером батча, равным .

Далее, к ней были добавлены еще таких же LSTM слоя, друг за другом.

Результаты работы сети, для объединенного множества тестовых и тренировочных данных из KDD 2000 [2], с перекрестной проверкой по блокам с эпохами обучения и размером батча, равным:

Таблица 3

Матрица неточностей

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Результаты классификации | |
|  |  | Отрицательный | Положительный |
| Настоящие значения | Отрицательный | 75003 | 1500 |
| Положительный | 2300 | 68697 |

Далее, можем посчитать следующие характеристики:

Таблица 4

Характеристики проверки

|  |  |
| --- | --- |
| Метрика | Значение |
| True Positive Rate | 0.4657 |
| True Negative Rate | 0.5084 |
| False Positive Rate | 0.0102 |
| False Negative Rate | 0.0156 |
| Accuracy | 0.9742 |
| Precision | 0.9786 |
| Recall | 0.9676 |
| F1 | 0.9745 |

Для данных [1] такая сеть же сеть показала результаты немногим хуже: на первом этапе перекрестной проверки была получена следующая матрица неточности:

Таблица 5

Матрица неточностей

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Результаты классификации | |
|  |  | Отрицательный | Положительный |
| Настоящие значения | Отрицательный | 673983 | 3802 |
| Положительный | 3900 | 18315 |

В UNSW-NB15[1] данных гораздо больше, это приводит к переобучению, поэтому используется метод dropout. Суть заключается в том, что можно обучить ансамбль сетей, вместо одной, а затем усреднить полученные результаты. Для этого проводится исключение из сети нейронов с некоторой вероятностью, так что он возвращает 0 для любых входных данных. Таким образом, нейроны не полагаются на другие, для исправления ошибок, что предотвращает переобучение. Ниже приведена иллюстрация, как работает dropout во время обучения:

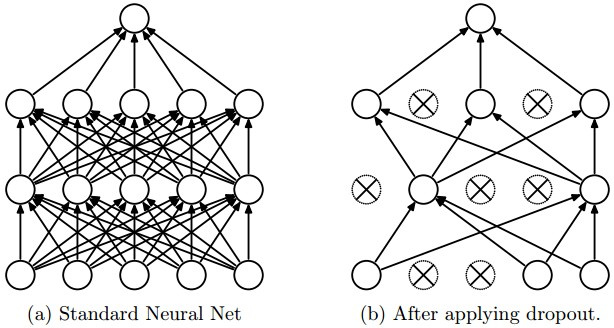


Рисунок 6 – Работа сети до и после применения dropout

Применив dropout между всеми слоями и направив выход каждого слоя на вход каждому последующему, был получен следующий результат для данных из работы [1]:

Таблица 6

Матрица неточностей

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Результаты классификации | |
|  |  | Отрицательный | Положительный |
| Настоящие значения | Отрицательный | 1855928 | 11682 |
| Положительный | 15657 | 216732 |

Далее, можем посчитать следующие характеристики:

Таблица 7

Показатели проверки

|  |  |
| --- | --- |
| Характеристика | Значение |
| True Positive Rate | 0.1032 |
| True Negative Rate | 0.8838 |
| False Positive Rate | 0.0056 |
| False Negative Rate | 0.0074 |
| Accuracy | 0.9869 |
| Precision | 0.9489 |
| Recall | 0.9326 |
| F1 | 0.9407 |

Таким образом, LSTM сеть успешно доказала возможность применения для обнаружения атак, однако некоторые все же остались необнаруженными (7% от общего числа атак).

## Классификатор типов атак на основе LSTM

Для построения классификатора атак по типам использовалась та же архитектура, описанная в разделе 3.1, за тем исключением, что Dense слой имеет размер выходного вектора на единицу больший, чем количество классифицируемых типов атак. Тогда классы маркируются целочисленным вектором единичной длины.

Проведя перекрестную проверку по блокам, для нормального трафика получена следующая матрица неточности:

Таблица 8

Матрица неточностей нормального трафика

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Результаты классификации | |
|  |  | Отрицательный | Положительный |
| Настоящие значения | Отрицательный | 231338 | 201 |
| Положительный | 31636 | 1826825 |

Для классов атак, расположим матрицы таким же образом но в одну таблицу:

Таблица 9

Матрицы неточностей атак

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Эксплоиты | |  | Разведка | |  | DoS | |
| 2042255 | 14951 | 2079631 | 19 | 2062191 | 16432 |
| 17313 | 15481 | 10291 | 59 | 4149 | 7228 |
| Стандартные | | Шелл-код | | Фаззинг | |
| 1936438 | 9 | 2088869 | 0 | 2071450 | 2 |
| 18092 | 135461 | 1131 | 0 | 18455 | 93 |
| Черви | | ПХ | | Анализ | |
| 2089871 | 0 | 2088348 | 0 | 2087995 | 0 |
| 1652 | 0 | 1652 | 0 | 2005 | 0 |

Рассчитаем характеристики для этих значений:

Таблица 10

Классификация атак по типу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Precision | Accuracy | Recall | F1 |
| Нормальный | 0.9999 | 0.9848 | 0.9829 | 0.9839 |
| Стандартные | 0.9999 | 0.9913 | 0.8822 | 0.9336 |
| DoS | 0.3054 | 0.9902 | 0.6353 | 0.7740 |
| Эксплоиты | 0.5087 | 0.9846 | 0.4721 | 0.6382 |
| Разведка | 0.7564 | 0.9951 | 0.0057 | 0.0113 |
| Фаззинг | 0.9789 | 0.9912 | 0.0050 | 0.0099 |
| Анализ | 0 | 0.9990 | 0 | 0 |
| ПХ | 0 | 0.9992 | 0 | 0 |
| Шелл-код | 0 | 0.9918 | 0 | 0 |
| Черви | 0 | 0.9961 | 0 | 0 |

Достаточно хорошо классифицируются general атаки, атаки DoS, и эксплоиты. Остальные не определяются вовсе, поэтому было решено добавить к функции потерь веса, указанные в разделе 2. После этого было получено:

Таблица 11

Матрицы неточностей атак

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Эксплоиты | |  | Разведка | |  | DoS | |
| 2053020 | 4186 | 2074483 | 5167 | 2059514 | 19109 |
| 26577 | 6217 | 6347 | 4003 | 3201 | 8176 |
| Стандартные | | Шелл-код | | Фаззинг | |
| 1936168 | 279 | 2072163 | 16706 | 2060389 | 11063 |
| 18031 | 135522 | 381 | 750 | 11989 | 6559 |
| Черви | | ПХ | | Анализ | |
| 2081755 | 8116 | 2062625 | 25723 | 2065715 | 22280 |
| 49 | 80 | 332 | 1320 | 622 | 1383 |

Таблица 12

Классификация атак по типу

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Precision | Accuracy | Recall | F1 |
| Нормальный | 1 | 0.9480 | 0.9415 | 0.9699 |
| Стандартные | 0.9979 | 0.9912 | 0.8826 | 0.9367 |
| DoS | 0.2997 | 0.9893 | 0.7186 | 0.4229 |
| Разведка | 0.4365 | 0.9944 | 0.3868 | 0.4101 |
| Фаззинг | 0.3722 | 0.9889 | 0.3536 | 0.3626 |
| Эксплоиты | 0.5976 | 0.9852 | 0.1896 | 0.2878 |
| Анализ | 0.0584 | 0.9890 | 0.6898 | 0.1076 |
| ПХ | 0.0488 | 0.9875 | 0.7990 | 0.0919 |
| Шелл-код | 0.0429 | 0.9918 | 0.6631 | 0.0805 |
| Черви | 0.0098 | 0.9961 | 0.6202 | 0.0192 |

Наихудшими показателями обладают атаки с малым количеством примеров в выборке, несмотря на то, что у них завышенные веса, это приводит к значительному росту FP для этих классов. Повышение весов улучшило показатели определения атаки разведки, но ухудшило обнаружение DoS, что может говорить о некой схожести в поведении данных атак (например, прослушивание портов может быть похоже на DoS). Также улучшились показатели для атак фаззингом, но ухудшились показатели для атак эксплоитами, возможно, это также говорит о их зависимости.

# анализ необходимости памяти сети

Stateful модель, используемая в разделе 3 данной работы не подходит для изучения того, насколько далеко сеть должна запоминать данные, так как мы не можем знать, что она хранит в состоянии ячейки на каждой итерации. Ниже приведена иллюстрация того, как необходимо сформировать данные для того, чтобы быть уверенным, что stateless модель может запомнить все что нужного из явно ограниченного количества предыдущих пакетов. После каждой партии данных, содержащей несколько, в данном случае 5, пакетов, состояние ячеек сети будет обнуляться, и они будут забывать о предыдущих пакетах.

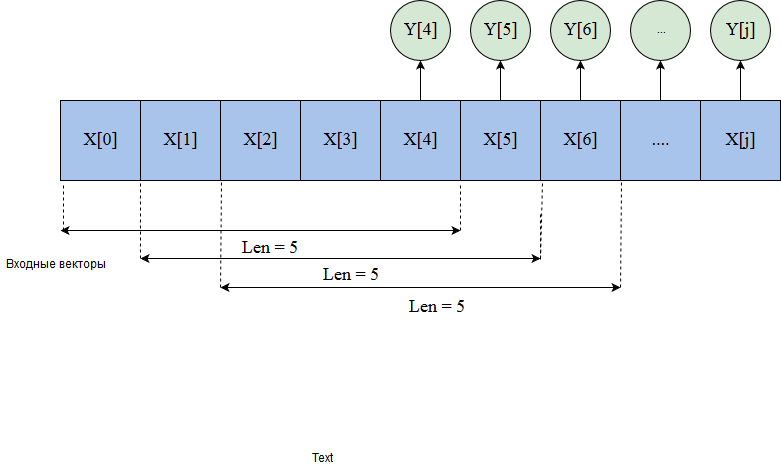


Рисунок 7 – Формирование данных для подачи stateless модели

Проведя несколько таких манипуляций, для разных величин запоминаемых пакетов, были получены следующие графики:

Рисунок 8 – Графики зависимости характеристик от количества запоминаемых пакетов

Данные модели получили лучшие результаты, чем stateful (вероятно, это связано с искусственно увеличенной выборкой, путем дублирования векторов столько раз, сколько пакетов запоминается (см. рисунок 7). Производительность у них также хуже во столько раз, во сколько больше пакетов запоминается (в stateless, с данными с рисунка 7, сначала будут поданы , далее , и т.д., в stateful же, ранее рассмотренной, будут подаваться , далее и т.д.).

Зависимость от количества запоминаемых пакетов кажется небольшой, но если учитывать что выборка большая, но это помогает обнаруживать до 1000 новых атак, нежели, если не иметь истории прошлых пакетов, так, запомним 10 пакетов, сеть выявила на 500 штук атак больше, чем зная о 4 пакетах.

Ухудшение показателей после 20 можно связать либо с переобучением сети, либо слишком большим количеством схожей и лишней информации. Если запоминать 40 пакетов, то в выборке каждый пакет будет встречаться 40 раз, что кажется излишним и может приводить к переобучению. К тому же, это сильно увеличивает размер (400 Мб превращаются в 16 Гб, которые уже сложно обработать).

Таким образом, оптимальные значения достигаются при запоминании пакетов.

# Заключение

В данной работе были построены бинарный классификатор, а также классификатор атак по типам на основе LSTM, они позволяют обнаруживать сетевые атаки с высокой точностью. Система доказала возможность обнаружения аномалий в трафике, что важно, учитывая растущее количество новых атак.

Была реализована система, позволяющая проводить различное конфигурирование сети, для анализа полученных с ее помощью результатов.

Также была выявлена необходимость в запоминании предыдущих пакетов, которая с учетом увеличивающейся сложности атак будет только нарастать, поэтому и системы обнаружения должны тоже становиться сложнее.

Метод, реализованный в данной работе необходимо использовать в совокупности с сигнатурными методами, реализуемыми такими системами, как Snort.

# Список использованных источников

1. The UNSW-NB15 Dataset [Электронный ресурс]. — Режим доступа: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/.— (Дата обращения: 27.06.2019).
2. The KDD Cup [Электронный ресурс]. — Режим доступа: https://semanticommunity.info/Data\_Science/KDD\_Cup.— (Дата обращения: 27.06.2019).
3. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. — СПб.: Питер, 2018. — 480 с.: ил. — (Серия «Библиотека программиста»).
4. Keras: The Python Deep Learning library [Электронный ресурс]. — Режим доступа: https://keras.io/. — (Дата обращения: 27.06.2019).
5. LSTM – сети долгой краткосрочной памяти [Электронный ресурс]. — Режим доступа: <https://habr.com> /ru/company/wunderfund/blog/331310/. — (Дата обращения: 27.06.2019).