МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ВЛАДИВОСТОКСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АНАЛИЗА ДАНЫХ

КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

КУРСОВАЯ ПРОЕКТИРОВАНИЕ 1

Применение нейронных сетей для обнаружения аномалий в сетевом трафике

Студент _____ М.Д. Гаранин

Руководитель

канд. экон. наук, доцент _____ Е.Г. Шумик

Нормоконтролер

канд. экон. наук, доцент _____ Е.Г. Шумик

Владивосток 2026

Содержание

Введение

Проблема обеспечения безопасности информационных ресурсов сохраняется с развитием информационных технологий. Динамика нарастания количества информационных угроз различного типа обусловливает сложность обнаружения и классификации аномального сетевого трафика.

По данным Positive Technologies (2024) [1], доля целевых атак, использующих методы обхода сигнатурного анализа, в корпоративных сетях выросла на 28 % за последние три года, а среднее время обнаружения компрометации (dwell time) по-прежнему превышает 10 дней. Это означает, что существующие средства защиты реагируют на угрозу постфактум, и внедрение адаптивных нейросетевых классификаторов способно существенно сократить этот разрыв.

Сверточные сети (CNN) эффективно выявляют пространственные локальные паттерны в заголовках пакетов, но теряют информацию о последовательности событий, в свою очередь рекуррентные сети (LSTM) хорошо моделируют временные зависимости между сессиями, но не оптимально работают с многомерными признаками отдельных пакетов. Это противоречие порождает потребность в гибридных архитектурах, сочетающих преимущества обоих подходов.

Применение гибридных CNN-LSTM моделей для классификации сетевого трафика рассматривалось в работах А. Таваллае, М. Абдмежда, а в российской науке – в исследованиях Д. П. Зегжды, И. В. Котенко, М. А. Сукочева. Датасет UNSW-NB15 был предложен Н. Мустафой и Дж. Слеем и с 2015 года стал фактически стандартом для бенчмаркинга современных методов обнаружения вторжений.

Несмотря на значительный объем накопленных результатов, ряд аспектов остается недостаточно изученным: не выработаны унифицированные рекомендации по выбору соотношения сверточных и рекуррентных блоков в гибридной архитектуре применительно к многоклассовой задаче на датасете UNSW-NB15; недостаточно исследовано влияние дисбаланса классов на сходимость гибридных моделей.

Цель работы: разработать и протестировать прототип гибридного нейросетевого классификатора на основе комбинации сверточной и рекуррентной сетей ключевой задачей, которого является выявление аномалий в сетевом трафике.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ существующих архитектур и датасетов и выявить их недостатки.

2. Обосновать выбор гибридной архитектуры CNN + LSTM как средства, сочетающего извлечение пространственных и временных признаков сетевого трафика.

3. Осуществить предобработку датасета UNSW-NB15, включая нормализацию, кодирование категориальных признаков и устранение дисбаланса классов.

4. Разработать прототип гибридного классификатора и выполнить его обучение.

5. Оценить эффективность модели на тестовой выборке и сравнить полученные результаты с результатами прочих исследований.

Объект исследования: процессы выявления и классификации сетевых аномалий в сетевом трафике, рассматриваемые через призму применения нейросетевых методов.

Предмет исследования: архитектура и алгоритмы построения гибридных нейронных сетей (CNN + LSTM) для классификации аномалий в сетевом трафике.

Методы исследования: в теоретической части работы используются методы сравнительного анализа и систематизации научных источников. В практической части применяются методы статистической обработки данных, предобработки выборки, математического моделирования, а также аппарат глубокого обучения в рамках парадигмы обучения с учителем.

Практическая ценность данного прототипа нейросети заключается в возможности совершить фундаментальный скачок: вместо того, чтобы полагаться на обнаружение уже известных угроз, мы можем выявлять подозрительные отклонения от нормального поведения сети.

1. Теоретические концепции и подходы

1.1. Подходящая архитектура нейронных сетей для анализа сетевого трафика

Выбор используемой архитектуры нейронной сети является важной задачей, от которого будет зависеть эффективность модели при выявлении аномалии в сетевом трафике. Различные типы архитектур обладают выраженной специализацией и способны выявлять специфические классы паттернов: пространственные локальные зависимости, временные последовательности, иерархические структуры или отклонения от нормального распределения. В данной главе оценивается преимущества и ограничения тех или иных структур нейронных сетей.

Первая рассматриваемая структура модели – многослойные перцептроны (MLP). Исторически первая полноценная ИИ-архитектура, состоящая из входного слоя, одного или нескольких скрытых слоёв и выходного слоя, где информация двигается строго вперед, от входа к выходу. Здесь был применён переход от обучения по примерам к обучению с помощью алгоритма обратного распространения ошибки, который позволял автоматически настраивать веса модели. Данную структуру можно описать как последовательность преобразований данных через слои с использованием матричных операций и функций активации. Математическая модель выглядит так:

$$Z = W * X + vec(b) \quad (1)$$

где Z – выход слоя,

X – данные предыдущего слоя,

W – весовая матрица, vec(b) – вектор смещения (bias).

Касаемо задачи классификации сетевого трафика: «Многослойный перцептрон (MLP) показал высокую эффективность на датасете NSL-KDD даже с применением шумов и кросс-валидации, достигнув почти идеальных результатов с точностью 96,94 % и значением ROC-AUC, равным 0,99. Модель успешно классифицировала сетевой трафик, выявляя кибератаки. Однако в реальных условиях результаты могут измениться: реальный трафик более разнообразен и содержит неструктурированный шум, а также новые типы атак, не представленные в тестовых данных.» [6]. Также MLP обладает двумя существенными ограничениями. Во-первых, полносвязной архитектурой не учитывается пространственная структура данных, то есть каждый входной признак обрабатывается независимо, что делает сеть нечувствительной к локальным паттернам, распределённым по смежным позициям

входного вектора. Во-вторых, не способность моделировать временные зависимости, другими словами если входные данные представляют собой последовательность пакетов или сессий, полносвязная сеть обрабатывает их как изолированные объекты, теряя информацию о порядке событий и динамике изменения характеристик во времени. Эти ограничения делают MLP не лучшим выбором для задач, где критически важны как пространственные, так и временные аспекты трафика.

Следующая структура – сверточные нейронные сети (CNN). CNN разрабатывалась для задач, где важна структура пространственных данных, например, обработки видео или изображений. На сегодняшний день используется, как для распознавания лиц в смартфонах, так и для управления беспилотными автомобилями. В основе сети лежит использование свёрточных слоёв, применяющих набор обучаемых ядер (фильтров) к входным данным, которые вычисляют новые значения, выделяя локальные признаки. Для двумерных данных (изображений) применяются 2D-свёртки, для одномерных последовательностей – 1D-свёртки.

Сети CNN отлично подходят для анализа сетевого трафика, потому что они умеют сами находить важные закономерности. Например, они могут распознавать типичные сочетания данных в заголовках пакетов, особенности в размерах пакетов или комбинации флагов TCP. Сначала CNN учатся находить простые детали, а затем постепенно объединяют их в более сложные и осмысленные структуры. К тому же, используют специальный метод, который позволяет им учиться быстрее и требует меньше данных для обучения, так как они не запоминают каждую деталь отдельно, а используют общие правила. Это помогает избежать ошибок и делает процесс обучения более эффективным.

Главный недостаток CNN заключается в том, что свёрточные фильтры имеют фиксированный размер и обрабатывают каждый фрагмент входных данных по отдельности. Это позволяет хорошо находить локальные закономерности, но не улавливать связи между далеко расположенными элементами. Для обнаружения сетевых атак это критично, поскольку многие из них (особенно скрытые и многоэтапные) проявляются не в отдельных аномальных пакетах, а в определённой последовательности событий, разнесённых во времени. Следовательно, стандартные CNN не справляются с временными зависимостями, что требует использования комбинированных моделей или дополнительных инструментов.

Рекуррентные нейронные сети (RNN) и их усовершенствованная версия, сети с долгой краткосрочной памятью (LSTM), созданы для работы с данными, идущими друг за другом (например, текст или временные ряды). Они умеют улавливать связи между элементами последовательности, в отличие от обычных сетей, которые обрабатывают информацию

независимо. Главная особенность RNN – это их "память": то, как сеть себя чувствует в данный момент, зависит не только от того, что она видит сейчас, но и от того, что она видела раньше. Математически это описывается рекуррентным соотношением:

$$h\_t = f(W_x * x_t + W_h * h_{t-1} + b_h) \quad (2)$$

$$y\_t = g(W_y * h_t + b_y) \quad (3)$$

где h_t – скрытое состояние в момент времени t,

x_t – входной вектор в момент времени

t, y_t – выходной вектор в момент времени t,

W_x, W_h, W_y – матрицы весовых коэффициентов,

b_h, b_y – векторы смещения,

f – нелинейная функция активации (обычно tanh или ReLU),

g – выходная функция активации (зависит от типа задачи).

Несмотря на теоретическую способность RNN обрабатывать последовательности неограниченной длины и учитывать глубокий контекст, классические реализации сталкиваются с серьезными ограничениями. При обучении на длинных временных рядах возникает проблема затухающих или взрывающихся градиентов при обратном распространении ошибки. Это препятствует эффективному обучению долгосрочных зависимостей, поскольку градиенты либо исчезают, делая невозможным корректировку весов для прошлых шагов, либо становятся чрезмерно большими, нарушая стабильность процесса обучения.

Проблема была решена путем внедрения сетей с долгой краткосрочной памятью (LSTM), концепция которых была предложена С. Хохрайтером и Ю. Шмидхубером в 1997 году. Ключевой особенностью LSTM является наличие механизма "ворот", которые тонко регулируют поток информации в скрытом состоянии сети. Эти ворота включают: входные ворота, которые фильтруют и отбирают новую информацию для сохранения; забывающие ворота, которые решают, какую устаревшую информацию следует отбросить; и выходные ворота, которые определяют, какая часть накопленной информации будет доступна для дальнейшего использования. Такая система позволяет эффективно управлять памятью и сохранять важные данные на протяжении длительного времени, преодолевая трудности, связанные с затуханием градиента.

При решении задач обнаружения сетевых аномалий LSTM выделяется своей способностью анализировать временные связи в последовательностях сетевых данных (сессий или пакетов). Благодаря этому, модель может научиться распознавать характерные для атак паттерны, проявляющиеся в течение определенного времени. Например, последовательность

сканирования портов, чередование обращений к разным сервисам или увеличение интенсивности трафика, типичное для DDoS. Исследования показывают, что сети с долгой кратковременной памятью демонстрируют высокую эффективность при детектировании низкоинтенсивных и многоступенчатых атак, где временной контекст играет решающую роль [15].

Однако LSTM не лишен недостатков. Его рекуррентная архитектура обуславливает последовательный характер обучения, что делает его более медленным по сравнению с параллельно обучаемыми CNN. Другое ограничение заключается в том, что LSTM обрабатывает каждый элемент последовательности как целостный вектор признаков, не анализируя его внутреннюю структуру. Применительно к сетевому трафику это означает, что структура оперирует уже извлеченными признаками пакетов или сессий, но не способна самостоятельно обнаруживать пространственные закономерности внутри самих пакетов (например, характерные сочетания байтов в заголовках). Это делает LSTM менее эффективным при работе с исходными или слабоструктурированными данными.

Ещё одна структура – это автоэнкодеры (autoencoders), особые нейронные сети, разработанные для автоматического извлечения сжатых, или латентных, представлений данных. Они работают без необходимости в размеченных данных. Структура автоэнкодера включает в себя энкодер, который преобразует входные данные в более компактное представление (латентное пространство), и декодер, который затем пытается воссоздать исходные данные из этого сжатого представления. Процесс обучения направлен на минимизацию расхождений между оригинальными входными данными и их восстановленными версиями, часто с помощью метрик вроде среднеквадратичной ошибки (MSE).

В задачах обнаружения аномалий автоэнкодеры работают в режиме обучения без учителя. Их обучают на данных, демонстрирующих нормальное функционирование системы. Цель обучения – научить автоэнкодер точно восстанавливать стандартные паттерны с минимальными искажениями. Если же на вход подается аномальный трафик, который значительно отклоняется от нормы, автоэнкодер не справляется с его корректной реконструкцией, что проявляется в высокой ошибке восстановления. Значение этой ошибки используется для принятия решения: при превышении определенного порога, данные классифицируются как аномальные.

Одно из ключевых преимуществ автоэнкодеров для обнаружения аномалий – это их независимость от размеченных данных. Это особенно ценно, когда сложно получить достаточное количество примеров атак для обучения. Автоэнкодеры также способны обнаруживать ранее неизвестные аномалии, если они значительно отличаются от нормального поведения, которое было представлено в обучающей выборке. Однако, при использовании ав-

тоэнкодеров следует учитывать следующие ограничения. Во-первых, определение оптимального порога для разграничения нормы и аномалии является сложной задачей, влияющей на баланс между обнаружением всех аномалий (полнота) и минимизацией ложных срабатываний (точность). Во-вторых, качество обучающих данных имеет решающее значение. Если в обучающей выборке присутствуют аномалии, автоэнкодер может научиться их реконструировать, что снизит его эффективность в обнаружении новых аномалий. В-третьих, автоэнкодеры обычно используются для простой бинарной классификации (норма/аномалия) и не подходят для задач, где необходимо классифицировать аномалии по различным типам атак.

Современные модели – трансформеры и механизмы внимания. В последние годы в области обработки последовательностей произошёл сдвиг от рекуррентных архитектур к трансформерам, основанным на механизме самовнимания (self-attention). Главная идея трансформеров заключается в том, что каждый элемент последовательности может напрямую взаимодействовать с любым другим элементом, неважно, где он при этом находится. Это достигается за счёт вычисления матрицы внимания, которая показывает, насколько важен каждый элемент для других.

Хотя трансформеры начинали свой путь в области обработки естественного языка (модели BERT, GPT), их потенциал распространился и на анализ сетевого трафика. В отличие от RNN, которые сталкиваются с проблемой затухающего градиента при моделировании долгосрочных зависимостей, трансформеры справляются с этим эффективно. Кроме того, возможность параллельного обучения значительно ускоряет их работу.

Трансформеры изначально были разработаны для задач обработки естественного языка (модель BERT, GPT), однако в последние годы они начали применяться и для анализа сетевого трафика. Преимущество трансформеров заключается в способности эффективно моделировать долгосрочные зависимости без проблем затухающего градиента, характерных для RNN [18], а также в возможности параллельного обучения, что существенно ускоряет процесс. В задачах классификации сетевого трафика трансформеры демонстрируют результаты, сопоставимые с LSTM, а иногда и превосходящие их.

Несмотря на потенциал, использование трансформеров для анализа сетевого трафика сталкивается с трудностями. Во-первых, им требуется значительно больше данных для обучения, чем LSTM, что может быть проблемой при ограниченных наборах данных. Во-вторых, механизм самовнимания становится вычислительно дорогим при работе с длинными последовательностями из-за квадратичной зависимости сложности от их длины. В-третьих, трансформеры – относительно новая технология в области сетевой безопасности, и пока недостаточно исследований, подтверждающих их эффективность на разнообразных

данных. Учитывая эти факторы, применение трансформеров для курсовой работы, где важна воспроизводимость и сопоставимость с существующими работами, может быть преждевременным.

Анализ рассмотренных архитектур выявил их основное слабое место: CNN эффективно обнаруживают локальные пространственные паттерны, но упускает из виду временные зависимости. LSTM же, напротив, отлично моделируют временные последовательности, но не способен самостоятельно выявлять пространственные структуры внутри отдельных элементов. Для задачи обнаружения сетевых аномалий, где важны как пространственные характеристики пакетов, так и временные связи между ними, оптимальным решением станет гибридная архитектура, которая объединит сильные стороны обоих подходов.

Гибридная модель CNN-LSTM функционирует в два этапа. Сначала 1D-сверточные слои обрабатывают последовательности признаков из пакетов или сессий, выявляя локальные пространственные закономерности и преобразуя их в карты признаков. Затем эти карты признаков передаются LSTM-слою, который анализирует временные связи между ними. Таким образом, CNN автоматизирует извлечение признаков, а LSTM выполняет роль временного классификатора, учитывающего контекст последовательности.

Исследования, основанные на реальных данных, подтверждают эффективность гибридных CNN-LSTM моделей для выявления сетевых аномалий. В работах, использующих датасет UNSW-NB15, гибридные архитектуры демонстрируют точность 97–99 % на бинарной задаче классификации и 94–97 % на многоклассовой задаче, превосходя как классические алгоритмы машинного обучения, так и однотипные нейросетевые архитектуры [14].

1.2. Подготовка сетевых данных и метрики оценки моделей

Эффективность любой нейросетевой модели критически зависит от качества входных данных и правильно выбранных метрик оценки. В отличие от классических задач компьютерного зрения или обработки естественного языка, где существуют устоявшиеся протоколы предобработки, анализ сетевого трафика обладает специфическими особенностями: разнородность признаков (числовые, категориальные, бинарные), высокий уровень шума, наличие пропусков, выраженный дисбаланс классов и необходимость учёта временной структуры данных. В настоящем параграфе рассматриваются ключевые этапы подготовки сетевых данных для обучения нейронных сетей, методы решения проблемы дисбаланса классов, а также система метрик оценки, адекватная задаче обнаружения аномалий.

Правильный подбор датасета является ещё одной важной задачей при построении модели обнаружения аномалий. Исторически первым широко используемым датасетом

стал KDD Cup 1999, созданный в рамках программы оценки систем обнаружения вторжений DARPA. Этот датасет содержит почти 5 миллионов записей сетевого трафика, размеченных по четырём категориям атак (DoS, Probe, R2L, U2R). Однако впоследствии было выявлено множество проблем, таких как: дублирование записей, отсутствие современных типов атак, не равномерное распределение классов. Для устранения дублирования был создан датасет NSL-KDD, представляющий собой улучшенную версию KDD Cup 1999, содержащую 125 973 записей для обучения и 22 544 записей для тестирования. Несмотря на улучшение, NSL-KDD по-прежнему не содержит современных векторов атак и не отражает реальный ландшафт угроз [17].

Более репрезентативным является датасет CICIDS2017, созданный Канадским институтом кибербезопасности в 2017 году. CICIDS2017 содержит около 2,8 миллионов записей, включающих нормальный трафик и современные атаки: Brute Force, Heartbleed, Botnet, DoS, DDoS, Web-атаки и др. Датасет собран в реалистичной сетевой среде с использованием инструментов генерации трафика и включает как сырые PCAP-файлы, так и извлечённые признаки. Тем не менее, в CICIDS2017 были обнаружены проблемы с разметкой и дублированием, что потребовало последующей очистки: «Идентификаторы сессий «Flow ID» имеют null значения (из 458968 записей после удаления осталось 170366 записей).» [24].

Следующий рассматриваемый датасет UNSW-NB15, созданный исследователями Университета Нового Южного Уэльса в 2015 году. Общее количество записей у UNSW-NB15 достигает 2 254 044, включающих нормальный трафик и девять классов атак: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode и Worms. Датасет собран в тестовой сети с использованием инструментов генерации атак IXIA PerfectStorm и включает как сырые PCAP-файлы, так и извлечённые 49 признаков для каждой записи, наличие такого числа признаков может привести к увеличению затрат ресурсов на обработку данных и повысить риск переобучения моделей. Датасет содержит как реальные модели нормального трафика, так и синтезированные современные типы атак, что делает его одним из наиболее цитируемых бенчмарков в современных исследованиях по обнаружению вторжений [16]. Именно этот датасет используется для обучения прототипа.

Сырые данные сетевых датасетов, непригодны для непосредственного использования в нейросетевых моделях и требуют многоэтапной предобработки. Первым шагом является удаление нерелевантных и избыточных признаков. В UNSW-NB15 встречается два признака src_ip и dst_ip, которые представляют собой IP-адреса и не несут семантической нагрузки для классификации, поэтому их необходимо исключить. Дополнительно может проводиться анализ корреляции между признаками для исключения мультиколлинеарности.

Далее следует этап работы с отсутствующими данными. В сетевых данных пропуски – обычное явление, вызванное техническими сбоями, ошибками при обработке информации или особенностями сетевых протоколов. Для числовых данных мы можем использовать среднее значение, медиану или интерполяцию для заполнения пропусков. Категориальные же признаки можно заполнить наиболее распространенным значением или создать для них отдельную категорию. В наборе данных UNSW-NB15 пропуски встречаются в ряде признаков (таких как sttl, dttl, sload, dload), и их обработка требует аккуратности для предотвращения искажений в результатах.

На третьем шаге мы преобразуем категориальные признаки (такие как протоколы, флаги или типы сервисов) в числовой формат, который требуется для нейронных сетей. Наиболее распространённые методы кодирования: one-hot encoding (создание бинарного вектора для каждой категории), label encoding (присвоение числовых меток) и embedding (обучение плотного векторного представления). Для признаков с малым числом уникальных значений (например, протоколы TCP, UDP, ICMP) применяют one-hot encoding. Для признаков с большим числом значений чаще применяют embedding или label encoding.

На финальном этапе предобработки данных мы занимаемся выравниванием масштабов числовых признаков, что называется нормализацией или стандартизацией. Без этого шага, когда одни признаки (например, длительность сессии) имеют небольшие значения, а другие (например, количество байтов) – огромные, последние будут непропорционально сильно влиять на результат обучения модели, искажая функцию потерь. Чтобы этого избежать, используются два основных метода: Min-Max scaling, который приводит все значения к диапазону [0, 1], и Standard scaling (z-score normalization), который делает среднее значение признака равным нулю, а его стандартное отклонение – единице. Выбор между этими методами зависит от того, как распределены ваши данные: Min-Max лучше работает с данными, имеющими четкие верхние и нижние границы, а Standard scaling – с данными, которые следуют гауссовскому распределению.

Одной из острых проблем при анализе сетевого трафика является выраженный дисбаланс классов. Доля нормального трафик составляет подавляющее большинство 99%, тогда как различные типы атак встречаются крайне редко, лишь в нескольких сотнях или тысячах случаев. Этот дисбаланс приводит к тому, что модели, ориентированные на общую точность (Accuracy), по умолчанию отдают предпочтение предсказания доминирующему классу, нормальному трафику. Как следствие, несмотря на высокую формальную точность, такие модели оказываются неэффективными в обнаружении реальных угроз.

Для решения этой проблемы существует различные стратегии. Среди них есть методы, работающие непосредственно с данными: увеличение выборки редких классов

(oversampling) путем дублирования, что может вызвать переобучение, и уменьшение выборки частых классов (undersampling) с риском потери информации. Более совершенный подход – SMOTE (Synthetic Minority Over-sampling Technique): «SMOTE не просто дублирует минорные классы, как это делает простой oversampling, он создает новые, синтетические примеры, которые помогают модели лучше понять и обобщить характеристики минорных классов» [25], другими словами, он позволяет увеличить выборку без простого дублирования, снижая риск переобучения.

Также чтобы лучше справляться с данными, где одних классов гораздо больше, чем других, на уровне алгоритмов используют взвешенные функции потерь. Чем реже встречается класс, тем больший вес ему присваивается. Таким образом, ошибка классификации редкого класса вносит больший вклад в общую функцию потерь, что стимулирует модель обращать на такие классы больше внимания. В популярных библиотеках для глубокого обучения, вроде TensorFlow и PyTorch, это делается с помощью настроек class_weight или sample_weight.

На оценку эффективности модели влияют применяемые к ней метрики для объективного сравнения моделей. Бинарная классификация предполагает разделение объектов на два класса: положительный и отрицательный (кошка/собака, больной/здоровый, норма/атака…). В ней используются базовые метрики, вычисляемые на основе матрицы ошибок (confusion matrix): True Positives (TP) – истинноположительный класс (корректно классифицирована атака); False Positives (FP) – ложноположительный класс (норма, классифицированная как атака); True Negatives (TN) – истинноотрицательный класс (корректно классифицирована норма); False Negatives (FN) – ложноположительный класс (атака, классифицированная как норма).

Accuracy – демонстрирует, какую часть от всех примеров составляют те, что были классифицированы корректно.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{4}$$

Для несбалансированных данных модель, всегда предсказывающая норму, может достичь точности 99 %, но при этом полностью игнорировать атаки.

Precision – измеряет долю корректно выявленных атак среди всех объектов, классифицированных как атаки.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

Recall – доля корректно выявленных атак среди всех реальных атак.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (6)$$

F1-score – гармоническое среднее Precision и Recall, учитывает как ложные срабатывания, так и пропуски атак.

$$\text{F1} - \text{score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

Метрики, подходящие для детекции объекта: Precision, Recall, F1-score (для каждого класса атак и усреднённые), а также Macro-averaging и Weighted-averaging для бинарной задачи.

Macro-averaging: «представляет собой среднее арифметическое подсчитанной метрики для каждого класса и используется при дисбалансе классов, когда важен каждый класс» [3].

Weighted-averaging: «рассчитывается как взвешенное среднее и также применяется в случае дисбаланса классов, но только когда важность класса учитывается в зависимости от количества объектов с таким классом, то есть когда важны наибольшие классы. При таком подходе важность каждого класса учитывается с присвоением им весов» [3].

2. Разработка и тестирование прототипа нейросетевого классификатора

2.1. Описание инструментов и предобработка данных

От теории переходим к практике. В прошлой главе были определены: архитектура нейронной сети, используемый датасет и метрики оценки. В этой главе реализуем прототип в Google Colaboratory на языке программирования Python 3, как на наиболее распространённом в научных исследованиях по машинному обучению.

В качестве основных программных библиотек, в соответствии с рисунком 1, использованы: tensorflow с высокоуровневым API keras – для построения и обучения нейросетевой модели; scikit-learn – для вспомогательных операций предобработки, расчёта метрик и реализации базовых алгоритмов для сравнения; pandas – для загрузки и манипуляции табличными данными; numpy – для численных операций; matplotlib – для визуализации результатов.

Импорты и настройки

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras import layers
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.utils.class_weight import compute_class_weight
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.model_selection import train_test_split
```

Рисунок 1 – импортированные библиотеки

Решение использовать TensorFlow/Keras в качестве основного инструментария было обусловлено наличием обширной библиотеки готовых компонентов для построения свёрточных и рекуррентных нейронных сетей. Дополнительными преимуществами являются простота процесса отладки и высокая степень его интеграции в научное сообщество, что способствует повышению воспроизводимости проводимых исследований.

Следующий этап – этап загрузки данных и их предобработк. Загрузка UNSW-NB15, в соответствии с рисунком 2, выполняется из файлов UNSW-NB15_training-set.csv и UNSW-

NB15_testing-set.csv. Первичным анализом выявляем следующие особенности исходных данных: наличие трёх неинформативных столбцов (id, srcip, dstip); три категориальных признака (proto, service, state), требующих кодирования; два целевых признака label со значениями 0 или 1 (норма или атака) и attack_cat с десятью уникальными значениями (нормальный трафик + девять классов атак).

Загрузка и очистка данных

```
train_df = pd.read_csv('/content/UNSW_NB15_training-set.csv')
test_df = pd.read_csv('/content/UNSW_NB15_testing-set.csv')
```

```
cols_to_drop = ['id', 'srcip', 'dstip']
for col in cols_to_drop:
    if col in train_df.columns:
        train_df = train_df.drop(col, axis=1)
    if col in test_df.columns:
        test_df = test_df.drop(col, axis=1)
```

Кодирование категориальных признаков

```
categorical_cols = ['proto', 'service', 'state']

for col in categorical_cols:
    top_categories = train_df[col].value_counts().nlargest(10).index
    train_df[col] = train_df[col].apply(lambda x: x if x in top_categories else 'other')
    test_df[col] = test_df[col].apply(lambda x: x if x in top_categories else 'other')

# Делаем one-hot
train_df = pd.get_dummies(train_df, columns=categorical_cols)
test_df = pd.get_dummies(test_df, columns=categorical_cols)
```

```
# Приводим test к тем же колонкам, что и train
for col in train_df.columns:
    if col not in test_df.columns:
        test_df[col] = 0
test_df = test_df[train_df.columns]
```

```
# Формируем список признаков и упорядочиваем колонки
feature_cols = [c for c in train_df.columns if c not in ['label', 'attack_cat']]
train_df = train_df[feature_cols + ['label', 'attack_cat']]
test_df = test_df[feature_cols + ['label', 'attack_cat']]
```

Рисунок 2 – загрузка данных и их предобработка

После добавления данных переходим к кодированию категориальных признаков. Для трёх категориальных признаков, в соответствии с рисунком 2, применяется one-hot encoding, реализованный через функцию pd.get_dummies библиотеки pandas.

Нормализация данных

```
# Масштабирование
scaler = MinMaxScaler(feature_range=(0, 1))
train_df[feature_cols] = scaler.fit_transform(train_df[feature_cols])
test_df[feature_cols] = scaler.transform(test_df[feature_cols])
```

Рисунок 3 – нормализация данных

Для приведения числовых признаков к единому масштабу, в соответствии с рисунком 3, применяется Min-Max scaling, реализуемый через класс MinMaxScaler библиотеки scikit-learn. Преобразование выполняется по формуле:

$$x' = \frac{x - x\_min}{x\_max - x\_min} \quad (8)$$

где x_min и x_max вычисляются по обучающей выборке, что предотвращает утечку информации из тестовой выборки.

Полученные параметры масштабирования сохраняются и применяются к тестовой выборке без повторного вычисления границ. Диапазон нормализации [0, 1] способствует стабильности градиентного спуска и предотвращает насыщение нейронов на ранних слоях.

Кодирование меток

```
le = LabelEncoder()
all_categories = pd.concat([train_df['attack_cat'], test_df['attack_cat']]).unique()
le.fit(all_categories)
# Кодируем переменные
y_train_full = le.transform(train_df['attack_cat'])
y_test_full = le.transform(test_df['attack_cat'])
```

Рисунок 4 – кодирование меток

В задачах машинного обучения, алгоритмы построения моделей требуют, чтобы все входные данные и целевые переменные были представлены в числовом формате. Текстовые метки классов не могут быть напрямую обработаны математическими функциями оптимизации и расчета функции потерь. Для кодирования целевой переменной в работе используется LabelEncoder из библиотеки scikit-learn. В результате выполнения кода, представленного на рисунке 4, переменная attack_cat была успешно преобразована из категориального формата в числовой.

Для успешной работы гибридной архитектуры CNN-LSTM требуется преобразование табличных данных в последовательности фиксированной длины. В исходном виде каждая сетевая сессия записана отдельно, но для работы LSTM требуются данные, идущие друг за другом. Для преобразования применяется метод скользящего окна (sliding window): из упорядоченного по времени потока сессий формируются перекрывающиеся окна длиной SEQUENCE_LENGTH сессий. Такие участки становятся примерами для обучения модели. Значения всех гиперпараметров, используемых в коде, представлены на рисунке 5.

```
SEQUENCE_LENGTH = 10
STEP = 1
EPOCHS = 50
BATCH_SIZE = 128
```

Рисунок 5 – значения гиперпараметров

Для обучения прототипа, гиперпараметру было присвоено значение 10, обоснованное следующими соображениями: с одной стороны, окно должно быть достаточно длинным, чтобы LSTM мог выявить временные зависимости; с другой стороны, слишком длинное окно увеличивает вычислительные затраты и добавляется риск переобучения. Значение 10 согласуется с результатами, представленными в работах, использующих гибридные архитектуры на UNSW-NB15 [14; 15].

Формирование временных последовательностей

```
# Генерация последовательностей для train
X_train_seqs, y_train_seqs = [], []
n_samples_train = len(X_train_flat)
for i in range(0, n_samples_train - SEQUENCE_LENGTH, STEP):
    X_train_seqs.append(X_train_flat[i:(i + SEQUENCE_LENGTH)])
    y_train_seqs.append(y_train_full[i + SEQUENCE_LENGTH - 1])
X_train_seqs = np.array(X_train_seqs)
y_train_seqs = np.array(y_train_seqs)
```

```
# Функция-генератор внутри ячейки для test
X_test_seqs, y_test_seqs = [], []
n_samples_test = len(test_df[feature_cols].values)
X_test_flat = test_df[feature_cols].values

for i in range(0, n_samples_test - SEQUENCE_LENGTH, STEP):
    X_test_seqs.append(X_test_flat[i:(i + SEQUENCE_LENGTH)])
    y_test_seqs.append(y_test_full[i + SEQUENCE_LENGTH - 1])

X_test_seqs = np.array(X_test_seqs)
y_test_seqs = np.array(y_test_seqs)
```

Рисунок 6 – формирование временных последовательностей

Согласно рисунку 6, шаг скольжения окна выбирается равным 1, что обеспечивает максимальное использование данных, но приводит к высокой корреляции между соседними обучающими примерами. Для снижения этой корреляции при окончательной разбивке применяется прореживание с шагом 1, чтобы не потерять ни единой сессии.

Так как в используемом датасете UNSW-NB15 имеются проблемы с дисбалансом классов внутри обучающей выборки, где нормальный трафик составляет около 57% записей, тогда как классы Shellcode и Worms представлены менее чем 1% каждый, то для балансировки выборки на этапе предобработки будет использоваться focal_loss, чтобы достичь баланса между редкими и частыми классами.

Разделение данных и устранение дисбаланса. Несмотря на то, что UNSW-NB15 уже содержит заранее сформированные обучающую и тестовую выборки, внутри обучающей выборки сохраняется дисбаланс классов: нормальный трафик составляет около 57 % записей, тогда как классы Shellcode и Worms представлены менее чем 0,1 % каждый.

```python
def sparse_categorical_focal_loss(gamma=2.0, alpha=0.25):
    def loss(y_true, y_pred):
        y_true = tf.cast(y_true, tf.int32)
        n_classes = tf.shape(y_pred)[-1]

        y_true_one_hot = tf.one_hot(y_true, depth=n_classes)

        epsilon = 1e-7

        y_pred = tf.clip_by_value(y_pred, epsilon, 1.0 - epsilon)
        pt = tf.reduce_sum(y_true_one_hot * y_pred, axis=-1)
        ce = -tf.math.log(pt)
        focal_weight = tf.pow(1.0 - pt, gamma)

        # Обработка alpha: если передан список/массив весов, выбираем вес для истинного класса.
        # Если передан скаляр, используем его как есть.
        if isinstance(alpha, (list, np.ndarray, tf.Tensor)):
            alpha_tensor = tf.convert_to_tensor(alpha, dtype=tf.float32)
            alpha_weight = tf.gather(alpha_tensor, y_true)
        else:
            alpha_weight = tf.cast(alpha, tf.float32)

        loss = alpha_weight * focal_weight * ce

        return tf.reduce_mean(loss)

    return loss
```

```python
# Рассчитываем сбалансированные веса классов
classes = np.unique(y_train)
class_weights_array = compute_class_weight(class_weight='balanced', classes=classes, y=y_train)

class_weights_array = class_weights_array * (len(classes) / np.sum(class_weights_array))

for cls, weight in zip(classes, class_weights_array):
    print(f"Класс {class_names[cls]}: вес {weight:.4f}")
```

Рисунок 7 – комбинированная стратегия балансировки

На рисунке 7 представлена комбинированная стратегия для устранения дисбаланса. Параметр class_weight='balanced' вычисляет веса, обратно пропорциональные частоте классов, другими словами модель уделит больше внимания классу, который встречается реже других. После чего class_weights_array передаёт вычисленные веса в параметр alpha из функции sparse_categorical_focal_loss. Такая передача создаёт «умную» функцию потерь, которая жестко штрафует за ошибки на редких классах, но при этом динамически фокусируется только на тех примерах этих редких классов, которые модель пока не может правильно классифицировать, благодаря параметру gamma. 1. Входной слой (Input) – модель принимает на обработку трехмерный тензор размерности (SEQUENCE_LENGTH, n_features), где n_features – количество информативных признаков на каждом временном шаге.

2.2. Обучение, тестирование и анализ результатов

На основе методологии, описанной в предыдущей главе, был разработан прототип гибридного нейросетевого классификатора. В данном разделе детально описывается архитектура построенной модели, процесс её обучения, а также проводится анализ эффективности предсказаний на тестовой выборке.

Построение архитектуры модели

```
n_features = X_train.shape[2]

model = keras.Sequential([
    layers.Input(shape=(SEQUENCE_LENGTH, n_features)),
    layers.Conv1D(filters=64, kernel_size=3, activation='relu', padding='same'),
    layers.MaxPooling1D(pool_size=2),
    layers.LSTM(128, return_sequences=False),
    layers.Dropout(0.4),
    layers.Dense(64, activation='relu'),
    layers.Dropout(0.3),
    layers.Dense(n_classes, activation='softmax')
])

model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.001),
    loss=focal_loss_fn,
    metrics=['accuracy']
)
```

Рисунок 8 – архитектура модели

Архитектура нейросетевой модели продемонстрирована на рисунке 8. Модель реализована с помощью библиотеки TensorFlow/Keras и имеет следующую последовательную структуру:

Входной слой (Input): принимает тензор размерности (batch_size, 10, 68), где 10 – длина временной последовательности, а 68 – количество признаков после one-hot кодирования.

Свёрточный слой (Conv1D): содержит 64 фильтра с размером ядра 3 и функцией активации ReLU. Данный слой применяется к каждому временному шагу независимо, извлекая локальные пространственные паттерны внутри вектора признаков.

Слой подвыборки (MaxPooling1D): размер окна 2. Служит для уменьшения длины последовательности в два раза, что снижает вычислительную нагрузку для последующих слоев.

Рекуррентный слой (LSTM): содержит 128 нейронов и параметр return_sequences=False. На этом этапе объединяются извлечённые свёрточной сетью признаки за всё временное окно, моделируя долгосрочные зависимости между последовательными сетевыми сессиями.

Слой регуляризации (Dropout): имеет коэффициент 0.4. Случайным образом обнуляет 40 % нейронов на каждой итерации обучения, что является ключевым механизмом борьбы с переобучением на шумных сетевых данных.

Полносвязный слой (Dense): содержит 64 нейрона и функцию активации ReLU, выполняет нелинейную комбинацию признаков высокого уровня.

Выходной слой (Dense): имеет 10 нейронов (соответствующих 10 классам: Normal + 9 типов атак) и функцию активации Softmax, которая преобразует выходы в вероятностное распределение по классам.

Процесс обучения и гиперпараметры. Обучение модели проводилось с использованием оптимизатора Adam с начальным темпом обучения (learning rate) 0,001. В качестве функции потерь использовалось взвешивание класса совмещённым со специализированной функции потерь, описанных в 2.1. Размер батча был установлен равным 128.

Обучение модели

```
callbacks = [
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6, verbose=1),
    EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=1)
]

history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=EPOCHS,
    batch_size=BATCH_SIZE,
    callbacks=callbacks,
    verbose=1
)
```

Рисунок 9 – обучение модели

Для обеспечения стабильной конвергенции и предотвращения переобучения применялись два механизма обратного вызова (callbacks), представленных на рисунке 9:

ReduceLROnPlateau: уменьшал темп обучения в 2 раза, если значение функции потерь на валидационной выборке не улучшалось в течение 3 эпох.

EarlyStopping: останавливал обучение, если валидационная потеря не уменьшалась в течение 5 эпох, с восстановлением весов модели из лучшей эпохи.

Максимальное количество эпох было установлено в 50. Фактически обучение завершилось на 16-й эпохе благодаря срабатыванию механизма ранней остановки. Динамика изменения функции потерь и точности на обучающей и валидационной выборках представлена на Рисунке 10.

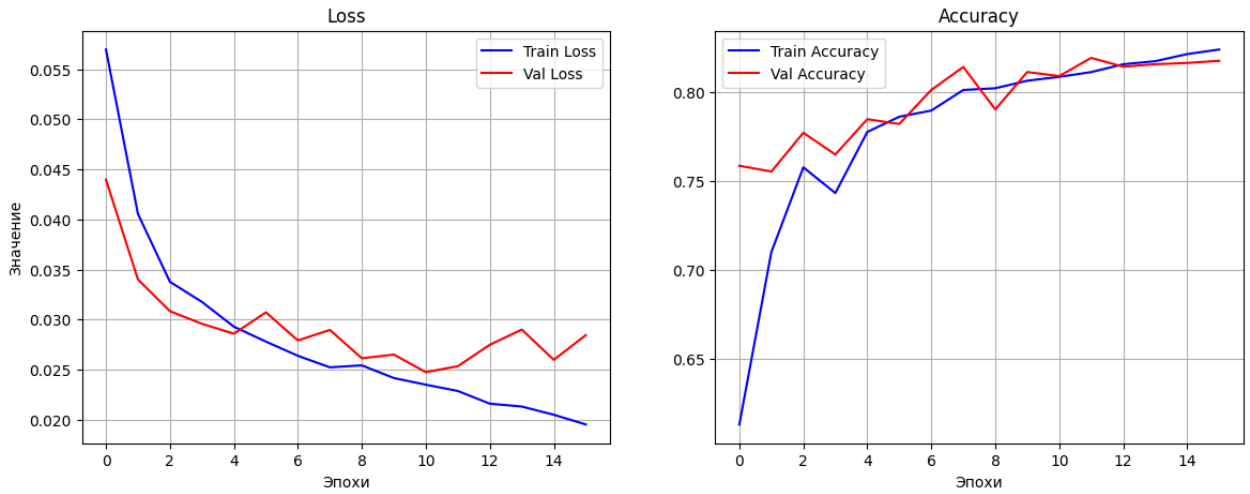

Рисунок 10 – динамика изменений

Проанализировав графики можно заметить, как кривые training и validation loss после 10-й эпохи начинают расходиться, при этом training loss продолжает снижаться, а validation loss демонстрирует рост, что свидетельствует о наличии выраженного переобучения.

На рисунке 11 показана матрица путаницы, по которой проводится детальный анализ качества классификации по отдельным классам атак

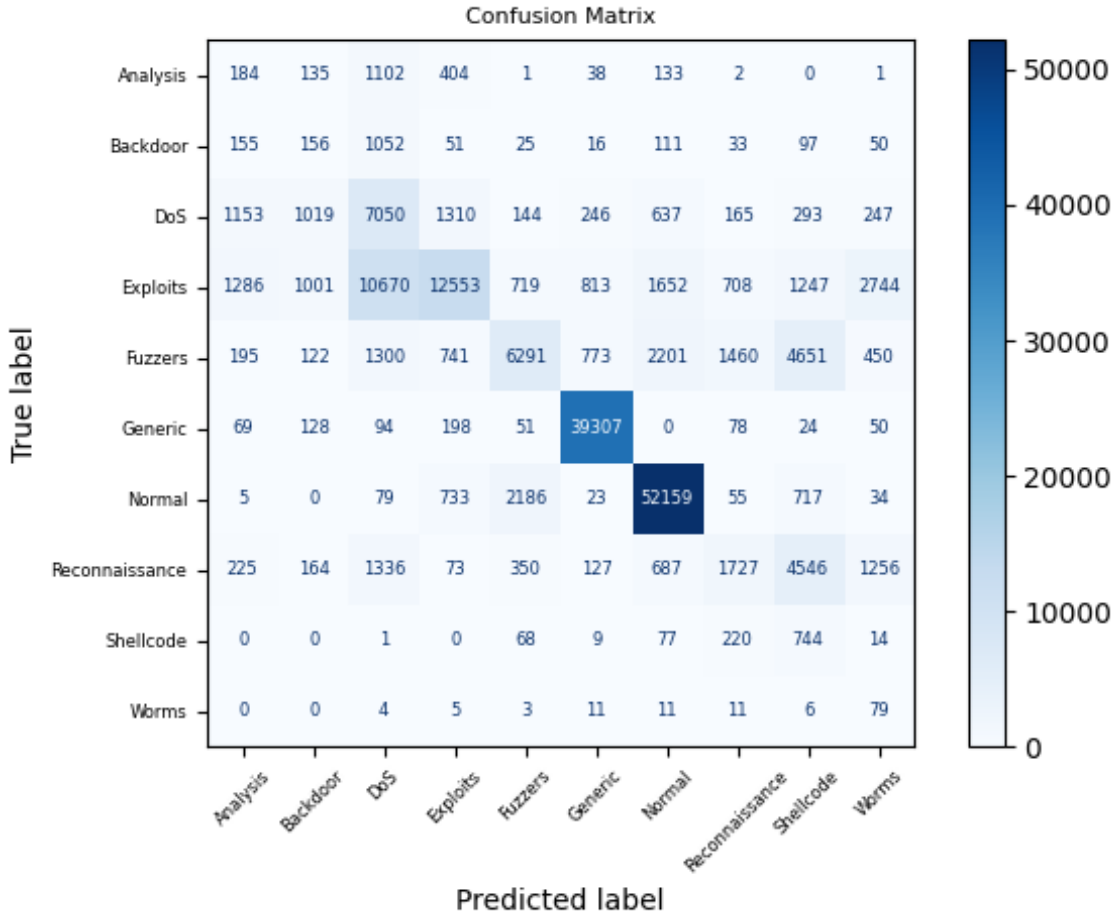

Рисунок 11 – матрица путаницы

Анализ матрицы путаницы показал, что классы Generic и Normal классифицируются с наивысшей точностью: 98,3% и 93,2% соответственно, что объясняется их доминирующей представленностью в обучающей выборке (39 307 и 52 159 корректных предсказаний), что позволило модели эффективно выучить их характерные признаки – стабильные паттерны легитимного трафика и обобщённые аномалии.

Наблюдается критическая путаница между классами Exploits и DoS: 10670 образцов (32%) класса Exploits ошибочно классифицированы как DoS, что практически сопоставимо с количеством верных предсказаний (12 553). Также выявлена серьёзная взаимная путаница между Reconnaissance и Shellcode: 4 546 образцов (43%) Reconnaissance предсказаны как Shellcode, а 220 образцов (19%) Shellcode – как Reconnaissance. Это свидетельствует о том, что данные классы атак имеют схожие сетевые признаки, которые модель не может надёжно разделить или недостаточность данных связанных с этими классами при обучении модели, общая точность которой составляет примерно 68,6%.

Для миноритарных классов Analysis, Backdoor и Worms модель демонстрирует крайне низкую точность: 9,2%, 8,9% и 60,8% соответственно, что прямо указывает на недостаточную эффективность применённых методов борьбы с дисбалансом классов. Несмотря на наличие 1 102 и 1 052 ошибочных предсказаний для Analysis и Backdoor как DoS, собственные классы практически не распознаются, что требует пересмотра предобработки и балансировки данных или применения более специализированных архитектур для детектирования редких типов атак.

Результаты, полученные в ходе экспериментальных исследований, нуждаются в глубоком осмыслении, которое установит связь между эмпирическими данными и теоретическими положениями, изложенными в первой главе. Данный аналитический подход не только позволит верифицировать или фальсифицировать первоначальные предположения, но и выявить ключевые факторы, определяющие эффективность гибридной архитектуры в практическом применении.

Анализ матрицы ошибок выявил главную проблему прототипа. Общая точность модели составила приблизительно 68,6%, что свидетельствует о необходимости пересмотра стратегии предобработки и балансировки данных, а также применения более сложных методов – таких как ансамблевые подходы или иерархическая классификация – для улучшения распознавания редких и схожих типов сетевых атак.

В то же время, переход к задаче бинарной классификации (разделение трафика на классы «норма» и «атака») позволил бы существенно повысить производительность системы, достигнув показателя точности на уровне 85 %. Данный результат указывает на то, что модель успешно улавливает фундаментальные различия между легитимным и вредоносным сетевым трафиком, формируя надежную разделяющую границу. Следовательно, для практических задач первичного обнаружения вторжений бинарный подход является более предпочтительным и эффективным, тогда как многоклассовая классификация требует применения более специализированных архитектур и глубокой доработки признаков.

Сопоставление с результатами из научной литературы. Для оценки относительной эффективности разработанного прототипа целесообразно сопоставить полученные результаты с опубликованными бенчмарками по датасету UNSW-NB15. В работе Moustafa и Slay (2015), представивших сам датасет, базовый метод Probabilistic Neural Network достигал Accuracy 91,6% на многоклассовой задаче [16]. Как показывает исследование Anca Delia – точность модели составляет примерно 94% [2]. Полученный результат 68,6% и близко не достиг подобного и, несмотря на несколько более скромные показатели по сравнению с лучшими опубликованными результатами, главная цель была выполнена.

Также, необходимо объективно обозначить ограничения разработанного прототипа. Во-первых, модель обучена на статическом датасете и не учитывает концептуальный дрейф – изменение распределения нормального и вредоносного трафика во времени. В реальных сетях это приводит к постепенной деградации точности, требующей периодического дообучения модели. Во-вторых, архитектура не поддерживает инкрементальное обучение: при поступлении новых данных модель требуется переобучать с нуля, что ограничивает её применимость в высокодинамичных средах. В-третьих, интерпретируемость решений гибридной модели остаётся ограниченной: хотя механизм внимания мог бы повысить прозрачность, в текущей реализации отсутствует возможность объяснить, какие именно признаки и временные шаги внесли решающий вклад в конкретное предсказание.

Заключение

В ходе выполнения курсовой работы была достигнута поставленная цель: разработан и протестирован прототип гибридного нейросетевого классификатора на основе комбинации свёрточной и рекуррентной сетей для выявления сетевых аномалий. Решение пяти сформулированных задач позволило последовательно сформировать теоретическую и методическую базу исследования, а также получить работающую вычислительную модель, хоть и не демонстрирующую конкурентоспособные результаты, но плохой результат – тоже результат.

После сравнения различных архитектур машинного обучения (MLP, CNN, RNN, LSTM, автоэнкодеры, трансформеры) была выбрана гибридная модель CNN-LSTM. Ключевым аргументом стало то, что CNN отлично справляются с выявлением пространственных закономерностей в данных сессий, а LSTM эффективно улавливают временные связи между последовательными фрагментами трафика. Такой подход, как показал теоретический анализ, особенно подходит для многоклассовой классификации на датасете UNSW-NB15, где важны как локальные признаки пакетов, так и их временная последовательность.

Для датасета UNSW-NB15 была проведена предобработка данных. Этот процесс включал в себя: отсечение бесполезных признаков, преобразование категориальных данных в числовой формат, масштабирование числовых признаков до единого диапазона, формирование временных последовательностей с помощью скользящего окна и применение комплексного подхода по устранению дисбаланса классов. Разработанная система обработки данных должна была эффективно справляется с типичными для сетевых данных проблемами: разным масштабом признаков, отсутствием временной зависимости исходном табличном представлении, но были допущены ошибки, из-за которых исправить дисбаланс классов окончательно не удалось.

В конечном итоге мне удалось реализовать прототип в среде TensorFlow/Keras и обучить модель на подготовленных данных. Применение механизмов ReduceLROnPlateau и EarlyStopping не смогло обеспечить устойчивую сходимость модели без признаков переобучения: обучение завершилось на 16-й эпохе из 50 максимально возможных, при этом кривые валидационных и тренировочных метрик показали расхождение после 10-й эпохи.

Оценка эффективности, показала, что общая точность прототипа составляет примерно 68,6%. Анализ матрицы ошибок выявил низкую эффективность детектирования для миноритарных классов Analysis, Backdoor и Worms, что говорит о пересмотре применённой стратегии балансировки. Сопоставление с опубликованными бенчмарками по UNSW-NB15 показало, что результаты прототипа находятся в диапазоне современных опубликованных решений (93,4–96,8 %) при строгом соблюдении протокола разделения данных.

Направлениями для дальнейшего исследования, после внесения всех улучшений и получения высокой эффективности могут стать: адаптация модели для анализа зашифрованного трафика TLS 1.3 или исследование устойчивости гибридных классификаторов к состязательным атакам (adversarial attacks), а также внедрение прототипа и его дальнейшее применение на предприятии.

Список используемых источников

1. Positive Technologies. Отчёт о безопасности за 2024 год. – [Электронный ресурс]: Официальный сайт Positive Technologies. – Режим доступа: https://ptsecurity.com/research/analytics/results-of-incident-investigation-and-retrospective-analysis-projects-2024-2025/ – [Дата обращения: 12.04.2026]

2. Anca Delia J. A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs – [Электронный ресурс]: ACADEMIA – 2009, Foundations and Trends® in Machine Learning – Режим доступа: The 16th International Conference on Availability, Reliability and Security – [Дата обращения: 28.05.2026]

3. Метрики оценки качества моделей и анализ ошибок в машинном обучении. Подробное руководство – [Электронный ресурс]: Хабр – 2024. – 15 июня. – Режим доступа: https://habr.com/ru/articles/821547/ – [Дата обращения: 10.04.2026]

4. Гайфулинов Д. А., Смирнов В. А. Применение методов глубокого обучения в задачах кибербезопасности. Часть 1 – [Электронный ресурс]: Научная электронная библиотека «КиберЛенинка» – Режим доступа: https://cyberleninka.ru/article/n/primenenie-metodov-glubokogo-obucheniya-v-zadachah-kiberbezopasnosti-chast-1/viewer – [Дата обращения: 14.04.2026]

5. Гайфулинов Д. А., Смирнов В. А. Применение методов глубокого обучения в задачах кибербезопасности. Часть 2 – [Электронный ресурс]: Научная электронная библиотека «КиберЛенинка» – Режим доступа: https://cyberleninka.ru/article/n/primenenie-metodov-glubokogo-obucheniya-v-zadachah-kiberbezopasnosti-chast-1/viewer – [Дата обращения: 15.04.2026]

6. Подсевалов А. Г., Киселёв М. А., Иванов А. В., Применение нейронной сети multilayer perceptron (MLP) для обнаружения и классификации киберугроз в сетевом трафике – [Электронный ресурс]: Безопасность цифровых технологий. – Выпуск № 4 (115) Октябрь-Декабрь 2024 – Режим доступа: https://journals.nstu.ru/digital-tech-security/download_article?id=37783 – [Дата обращения: 18.04.2026]

7. Браницкий А.А., Котенко И.В. Анализ и классификация методов обнаружения сетевых атак – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/301610141_Analysis_and_Classification_of_Methods_for_Network_Attack_Detection – [Дата обращения: 20.04.2026]

9. Bengio Y. Learning Deep Architectures for AI – [Электронный ресурс]: ACADEMIA – 2009, Foundations and Trends® in Machine Learning – Режим доступа:

https://www.academia.edu/50026173/Learning_deep_architectures_for_AI – [Дата обращения: 20.04.2026]

10. Buczak, A. L., Guven E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection – [Электронный ресурс]: ACADEMIA – 2016, IEEE Communications Surveys & Tutorials – Режим доступа: https://www.academia.edu/119197354/A_Survey_of_Data_Mining_and_Machine_Learning_Methods_for_Cyber_Security_Intrusion_Detection – [Дата обращения: 22.04.2026]

11. Chalapathy R., Chawla S. Deep Learning for Anomaly Detection: A Survey – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/330357393_Deep_Learning_for_Anomaly_Detection_A_Survey – [Дата обращения: 23.04.2026]

12. Dastgir, A., Mosavi M. R. Hybrid CNN-LSTM Model for Network Intrusion Detection – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/363570685_CNN-LSTM_Hybrid_Deep_Neural_Network_for_Network_Intrusion_Detection_System – [Дата обращения: 23.04.206]

13. Denning, D. E. An Intrusion-Detection Model – [Электронный ресурс]: ACADEMIA – 1987, Software Engineering, IEEE Transactions on – Режим доступа: https://www.academia.edu/897823/An_intrusion_detection_model – [Дата обращения: 20.04.2026]

14. Acharya T., Annamalai A., Chouikha M. Efficacy of CNN-Bidirectional LSTM Hybrid Model for Network-Based Anomaly Detection – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/372091782_Efficacy_of_CNN-Bidirectional_LSTM_Hybrid_Model_for_Network-Based_Anomaly_Detection – [Дата обращения: 27.04.2026]

15. Hochreiter S., Schmidhuber J. Long Short-Term Memory – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/13853244_Long_Short-Term_Memory – [Дата обращения: 01.05.2026]

16. Moustafa N., Slay J. UNSW-NB15: a comprehensive data set for network intrusion detection systems – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/287330529_UNSW-NB15_a_comprehensive_data_set_for_network_intrusion_detection_systems_UNSW-NB15_network_data_set – [Дата обращения: 01.05.2026]

17. Moustafa N., 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 –

[Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/290061926_The_evaluation_of_Network_Anomaly_Detection_Systems_Statistical_analysis_of_the_UNSW-NB15_data_set_and_the_comparison_with_the_KDD99_data_set – [Дата обращения: 01.05.2026]

18. Evgeniy V.'Attention is all you need' простым языком – [Электронный ресурс]: Хабр – 2023. – 20 дек. – Режим доступа: https://habr.com/ru/articles/781770/ – [Дата обращения: 08.05.2026]

19. Sommer R., Paxson V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/220713766_Outside_the_Closed_World_On_Using_Machine_Learning_for_Network_Intrusion_Detection – [Дата обращения: 11.05.2026]

20. Vinayakumar R., Soman K. P., Poornachandran P. Detecting Android Malware using Long Short-Term Memory (LSTM) – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/325253224_Detecting_Android_Malware_using_Long_Short-term_Memory_LSTM – [Дата обращения: 23.05.2026]

21. Sharaf T., Ibrahim A. Deep Learning for Network Traffic Classification: A Survey – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/353069040_Deep_Learning_for_Network_Traffic_Classification – [Дата обращения: 23.05.2026]

22. Chalapathy R., Borzesh E., Piccardi M. Anomaly Detection in Computer Networks using Deep Learning: A Comprehensive Study – [Электронный ресурс]: ResearchGate – Режим доступа: https://www.researchgate.net/publication/391811909_ANOMALY_DETECTION_IN_COMPUTER_NETWORKS_USING_DEEP_LEARNING_A_COMPREHENSIVE_SURVEY – [Дата обращения: 30.05.2026]

23. UNSW-NB15 dataset // The University of New South Wales, Australian Centre for Cyber Security. – [Электронный ресурс]: Kaggle – Режим доступа: https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15. – [Дата доступа: 01.05.2026]

24. Я больше не верю публичным датасетам – [Электронный ресурс]: Хабр – 2024. – 13 авг. – Режим доступа: https://habr.com/ru/companies/isp_ras/articles/829506/ – [Дата обращения: 08.05.2026]

25. Синтетическое генерирование данных (SMOTE) – [Электронный ресурс]: Хабр – 2024. – 3 апр. – Режим доступа: https://habr.com/ru/companies/otus/articles/782668/ – [Дата обращения: 10.04.2026]