МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

УДК 004.93:681.5.012:004.056.53 (043.5)

Бабичева Маргарита Вадимовна

АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ НАУЧНЫХ ИССЛЕДОВАНИЙ УГРОЗ БЕЗОПАСНОСТИ ЛИЧНОСТИ

Специальность 2.3.3. «Автоматизация и управление технологическими процессами и производствами» (технические науки)

Диссертация
на соискание ученой степени
кандидата технических наук

Научный руководитель:
доктор технических наук, профессор
Данилов Владимир Васильевич

Идентичность всех экземпляров
ПОДТВЕРЖДАЮ
Ученый секретарь диссертационного
совета 02.2.006.02
кандидат технических наук, доцент Т.В. Завадская

Донецк – 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследований. Контроль и прогнозирование сложных процессов, например, стратегии нацбезопасности, «…включающей оборону страны, государственную, общественную, информационную, экологическую, экономическую, транспортную, энергетическую и безопасность личности», по многочисленности аналитических вычислений немыслим без создания автоматизированных систем научных исследований (АСНИ) с привлечением элементов исскуственного интеллекта, в частности, искусственных нейронных сетей (ИНС).

Известен круг задач, решаемых ИНС: распознавание образов, классификация, принятие решений и управление, кластеризация, прогнозирование, аппроксимация, сжатие данных и ассоциативная память. Основными преимуществами нейронных сетей перед традиционными вычислительными методами являются: решение задач в условиях неопределенности, устойчивость к шумам во входных данных, гибкость структуры нейронных сетей, высокое быстродействие, адаптация к изменениям окружающей среды, отказоустойчивость. Известны и недостатки искусственных нейронных сетей, основные из которых: трудоемкость и длительность обучения; неспособность принятия решений в несколько этапов; трудозатратность с точки зрения объема используемых вычислительных ресурсов и математической сложности моделей, внедрение продвинутых нейросетевых систем возможно только по лицензии от производителя и т.д.

Тем не менее, анализ существующих научных работ показал, что создание автоматизированных систем научных исследований безопасности личности (АСНИ БЛ) на основе искусственных нейронных сетей является актуальной научно-технической задачей, имеющей практическое значение, решение которой позволит повысить их быстродействие, степень достоверности принятия решений, а также снизить уровень уязвимости нейросетевых алгоритмов.

Степень разработанности темы исследования. Вопросам применения нейросетевых алгоритмов для решения технологических задач посвящены исследования Т.В. Филатовой, А.И. Павлова, О.Ю. Лончакова, А.В. Жвакина, в которых рассмотрены преимущества нейросетевого подхода в решении задач автоматизированной обработки формализованных данных перед традиционными численными статистическими методами. В работах В.Е. Сорокина, А.А. Ежова, А.С. Новикова, И. В. Крысова, И. Л. Чулкова, А. В. Брагина исследованы нейросетевые технологии обработки изображений и применение их в автоматизированных системах машинного зрения, разработки технической документации, проектирования, обработки результатов научных исследований. Однако остается нерешенным вопрос о возможности применения в автономных автоматизированных системах малозатратных нейросетевых решений, а также оптимизации глубоких сверточных сетей для использования в микроконтроллерных системах управления. При этом актуален вопрос уязвимостей, которыми обладают подобные системы. В работах Н. Папернота, А. Фази, Н. Карлини, Д. Вагнера и др. рассмотрены уязвимости нейросетевых алгоритмов, однако нет исследований обобщающих, сравнивающих и тестирующих известные уязвимсти нейронных сетей и степень их опасности.

Связь с научными направлениями, планами, темами. Диссертационное исследование выполнено в соответствии с планами научно-исследовательских работ ГОУ ВПО «ДОННУ», выполненных на кафедре радиофизики и инфокоммуникационных технологий (РФ и ИКТ) в рамках НИР №0118D000013 Г-18/39 «Моделирование защищенных инфокоммуникационных систем» и госбюджетной темы №0119Д000024 19-1/вв41 «Диагностика и контроль структуры конструкционных материалов по данным анализа ихпараметров методами акустической спектроскопии» в части исследований методов применения нейронных сетей для обработки экспериментальных данных.

Цель и задачи исследований. Целью работы является создание АСНИ безопасности личности на основе искусственных нейронных сетей путем совершенствования технологий обработки данных, что позволит повысить

быстродействие систем и степень достоверности принятия решений, а также снизить уровень уязвимости нейросетевых алгоритмов.

Для достижения цели в работе поставлены и решены следующие задачи:

- разработка метода сокращения количества параметров обрезанием проигравших нейронов нейросетевых алгоритмов;

- программная реализация систем обработки формализованных данных и изображений, включая архитектурные решения;

- снижение уровня уязвимостей АСНИ безопасности личности на основе модифицированных нейросетевых алгоритмов.

Объект исследования. Автоматизированные системы научных исследований безопасности личности на основе искусственных нейронных сетей.

Предмет исследования. Нейросетевые модели представления и обработки данных.

Научная новизна полученных результатов заключается в следующем.

1. Впервые разработан метод сокращения количества параметров нейронной сети обрезанием проигравших нейронов, позволяющий уменьшить ресурсоемкость и увеличить быстродействие без потери точности нейросетевых алгоритмов АСНИ безопасности личности. Показано, что результаты сокращения количества параметров не зависят от архитектуры нейронной сети и методов обучения;

2. Дальнейшее развитие получили:

- метод обработки файлов логов серверов для обнаружения угроз;

- метод распознавания формы предметов на основе нейронной сети LVQ;

- процедура аутентификация с распознаванием лиц на основе сверточной нейронной сети;

- процедура видеонаблюдения по распознаванию предметов повышенной опасности.

3. Впервые обосновано применение метода Харриса для выделения признаков, поступающих на нейронную сеть, а также, разработан собственный итерационный алгоритм бинаризации, для автоматизированных систем доступа по отпечаткам пальцев.

4. Впервые предложены 9 методов генерации состязательных примеров для ненаправленных и направленных угроз на нейросетевые классификаторы и системы распознавания лиц, в том числе Bing, Google, Yandex.

Теоретическая и практическая значимость работы.

Теоретическая значимость результатов работы заключается в обосновании принципов применения нейросетевых технологий в автоматизированных системах научных исследований безопасности личности и определении ограничений, которые накладывают нейросетевые алгоритмы на такие системы.

Практическое значение результатов исследований.

1. Разработан метод, позволяющий на 70% сокращать количество параметров нейронных сетей, с сохранением и даже увеличением точности и скорости работы нейросетевых алгоритмов.

2. Разработаны принципы построения компактных нейросетевых решений, которые можно использовать для прошивки микроконтроллерных устройств в автономных автоматизированных системах.

3. Разработаны алгоритмы для обработки изображений в автоматизированных системах аутентификации и классификации.

4. Предложены способы защиты АСНИ безопасности личности на нейросетевых алгоритмах от атак генерацией состязательных примеров.

Практическая ценность работы подтверждается следующим:

а) Алгоритм распознавания лиц сверточной нейронной сетью, оптимизированной разработанным методом редукции нейронов внедрен на предприятии ФИРМА «МДЛ» в систему аутентификации доступа на базе монитора видеодомофона М-480М с системой на кристалле Allwiner A-13 (Акт внедрения от 05.06. 2021 г.)

б) Методика аутентификации пользователей и программное приложение для распознавания лиц на основе нейронной сверточной сети, методика проверки цифровых документов на подлинность, компьютерное приложение для защиты цифровых документов от редактирования внедрены в учебный процесс кафедры радиофизики и инфокоммуникационных технологий путем использования в

лабораторном практикуме по дисциплине «Основы информационной безопасности» в Донецком национальном универитете (акты внедрения 01.18/12.1-34 и 02.18/12.1-34 от 16.03.2018 г.).

Методология и методы исследований. Для решения поставленных задач использовались следующие научные методы: математическое моделирование (для представления данных), многомерный статистический анализ (для обработки формализованных данных), цифровая обработка сигналов и теория распознавания образов (для обработки изображений), прогнозирование и оптимизация (для разработки архитектур систем), современные методы программирования (для практической реализации автоматизированных систем).

Научные положения, выносимые на защиту:

1. Предложенный метод сокращения количества параметров нейронных сетей обрезанием проигравших нейронов позволяет на 70% снизить количество параметров нейронных сетей, с сохранением точности и повышением скорости работы нейросетевых алгоритмов.

2. Внедрение обоснованного теоретически метода построения компактных нейросетевых АСНИ широкого назначения, в том числе безопасности личности и методов обработки изображений для АСНИ по распознаванию образов позволяет повысить их эффективность и надежность.

Степень достоверности и апробация результатов. Обоснованность и достоверность научных положений, выводов и рекомендаций подтверждается результатами математического моделирования, компьютерных экспериментов и тестирования разработанных прототипов автоматизированных систем научных исследований безопасности личности.

По направлению исследований, содержанию научных положений и выводов, существу полученных результатов диссертационная работа соответствует паспорту специальности 2.3.3. «Автоматизация и управление технологическими процессами и производствами» в частности: п.3 «Методология, научные основы, средства и технологии построения автоматизированных систем управления технологическими процессами (АСУТП) и производствами (АСУП), а также

технической подготовкой производства (АСТПП) и т. д.», п. 16 «Средства и методы проектирования технического, математического, лингвистического и других видов обеспечения АСУ», п. 18 «Разработка автоматизированных систем научных исследований».

Основные положения диссертационной работы апробированы на научно-технических конференциях: IV Международная научная конференция «Донецкие чтения 2019: образование, наука, инновации, культура и вызовы современности» «Нейронные сети в системах для научных исследований», г. Донецк, 2919 г., XVIII Международная научно-практическая конференция, г. Пенза, «Атаки на нейросетевые автоматизированные системы распознавания опасных предметов» 2021 г., II Международная научно-практическая конференция «Программная инженерия: методы и технологии разработки информационно-вычислительных систем (ПИИВС-2018)» «The authenticity of digital documents checking and protection by own steganography algorithm», г. Донецк, 2018 г.

Личный вклад автора. Все результаты и положения, составляющие основное содержание диссертации, вынесенные на защиту, получены автором самостоятельно. Личный вклад соискателя заключается в обосновании идеи работы и ее реализации, в выполнении теоретических и экспериментальных исследований, разработке программного обеспечения и прототипов АСНИ.

Вклад соискателя в работы, опубликованные в соавторстве, конкретизирован в списке работ, опубликованных по теме диссертации.

Публикации. Основные научные результаты диссертации опубликованы автором самостоятельно и в соавторстве в 12 научных изданиях, из них 4 – в изданиях, включенных в перечень ВАК ДНР, 4 – в иных научных изданиях, 4 – в материалах и тезисах конференций.

Структура и объём диссертации. Диссертационная работа изложена на 188 страницах, состоит из введения, четырех разделов, заключения, перечня условных сокращений, списка литературы из 93 наименований и 6 приложений.

РАЗДЕЛ 1
АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ НАУЧНЫХ ИССЛЕДОВАНИЙ БЕЗОПАСНОСТИ ЛИЧНОСТИ (АСНИ БЛ) НА ОСНОВЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

В разделе выделен класс задач, которые решаются при помощи нейросетевых технологий. Сформулированы проблемы, возникающие при внедрении нейросетевых алгоритмов в АСНИ БЛ: затратность с точки зрения машинных ресурсов, необходимость лицензирования существующих решений и уязвимость к спецефическим нейросетевым атакам. Рассмотрены нейронные сети в АСНИ по обработке формализованных данных и обработке изображений, надежность нейросетевых алгоритмов АСНИ.

Сделаны выводы, что задачи аппроксимации, интерполяции, экстраполяции и классификации решаются с помощью сетей с радиально-базисными функциями, вероятностных сетей, обобщенно-регрессионных сетей, многослойного персептрона, сетей с конкурирующими слоями, сверточных сетей. Основное внимание необходимо уделить архитектуре сети, методу обучения, выделению и подготовке данных для обучения и входных данных

Краткая характеристика трудностей, возникающих при внедрении ИНС в АСНИ БЛ позволила сформулировать следующие задачи исследования: разработка метода сокращения количества параметров обрезанием проигравших нейронов нейросетевых алгоритмов, программная реализации систем обработки формализованных данных и изображений, включая архитектурные решения, снижение уровня уязвимостей АСНИ безопасности личности на основе модифицированных нейросетевых алгоритмов.

1.1 Нейронные сети в АСНИ по обработке формализованных данных

Проблема применения нейронных сетей в автоматизированных системах в том, что для входов и выходов нейронной сети необходимо подготавливать данные,

при этом выделение, масштабирование и нормализация данных порой оказывается более сложной задачей, чем разработка самой нейронной сети. Поэтому с точки зрения создания нейронной сети, такого рода задачи являются самыми простыми, однако с точки зрения подготовки данных требуют хорошего знания процесса, владения математическим аппаратом и точной постановки задачи. Формализация сводится к нахождению входных и выходных векторов. При этом логика работы нейронной сети такова, что эти данные должны лежать в диапазоне от 0 до 1. Такие формализованные данные присутствуют в задачах аппроксимации, интерполяции и экстраполяции, классификации и восстановления поврежденных или зашумленных данных, при этом применение нейронных сетей целесообразно лишь в том случае, если вычислительная сложность обычных математических алгоритмов решения такого рода задач высока [1].

В частности, задачи обработки экспериментальных данных можно решать и численными статистическими методами, такими как регрессионный, сплайновый или вейвлет анализ. Немало работ посвящены сравнению результатов обработки данных традиционными методами и нейронными сетями [2, 3, 4, 5, 6]. Преимущества использования нейронных сетей перед классическими методами в том, что они обладают высокими адаптационными характеристиками, могут решать сложные задачи, когда входные и выходные данные имеют неявные зависимости, или вообще их не имеют.

Задачи аппроксимации экспериментальных данных решаются с помощью искусственных нейронных сетей следующих типов: многослойного персептрона, сетей с радиально-базисными функциями, вероятностных сетей, обобщенно-регрессионных сетей [7, 8]. При этом две последние дают хорошие результаты в задачах аппроксимации и интерполяции, а многослойный персептрон применяется для экстраполяции.

Например, в работе Т.В. Филатовой рассматривается применение искусственных нейронных сетей для решения задачи аппроксимации данных и сопоставление результатов с традиционным методом построения трендовых моделей с помощью нейронных сетей типа радиальная базисная (RBF), обобщенно-

регрессионная (GRNN) и линейная [9]. Автор делает вывод, что для рассматриваемого примера аппроксимации данных по известным объёмам продаж авиабилетов применение нейронных сетей обеспечивает высокое качество аппроксимации и может использоваться для анализа и прогнозирования деятельности авиапредприятий.

Модели нейронных сетей, применяемых для обработки экспериментальных данных компактны, особенно если они написаны на скриптовых языках типа Python [10]. Время обучения зависит от архитектуры, например, RBFсети учатся очень быстро, а вот многослойный персептрон на больших объемах данных может обучаться больше недели.

При решении задач классификации необходимо отнести имеющиеся образцы к определенным классам. Наиболее распространенным является способ представления данных для классификации, при котором образец представляется в виде вектора, значения которого нормализованы и масштабированы. Компоненты этого вектора представляют собой различные характеристики образца, которые влияют на принятие решения о том, к какому классу можно отнести данный образец. Например, для медицинских задач в качестве компонентов этого вектора могут быть данные из медицинской карты больного [11]. На основании информации о примере, необходимо определить, к какому классу его можно отнести. Классификатор относит объект к одному из классов в соответствии с определенным разбиением N-мерного пространства, которое называется пространством входов, и размерность этого пространства является количеством компонент входного вектора.

Правильный выбор объема сети в данном случае имеет большое значение. Построить небольшую и качественную модель часто бывает просто невозможно, а большая модель будет просто запоминать примеры из обучающей выборки (переобучение нейронной сети), что, естественно, приведет к некорректной работе классификатора.

Существуют два основных подхода к построению сети. При первом из них вначале берется сеть минимального размера, и постепенно увеличивают ее до

достижения требуемой точности. При этом на каждом шаге ее заново обучают. Также существует так называемый метод каскадной корреляции, при котором после окончания эпохи обучения происходит корректировка архитектуры сети с целью минимизации ошибки. При втором подходе вначале берется сеть завышенного объема, и затем из нее удаляются узлы и связи, по определенному алгоритму. При этом число примеров в обучающем множестве всегда должно быть больше числа настраиваемых весов. Иначе вместо обобщения сеть просто запомнит данные и утратит способность к классификации и если борать примеры, не вошедшие в обучающую выборку, то результат будет не определен [12].

В случае обучения с учителем, классификацию проводят сетями типа GRNN, с линейными и RBF слоями. Однако существует понятие кластеризации, когда нейронная сеть сама собирает данные в кластеры. Для решения задач кластеризации применяют карты и сети Кохонена, сети Хопфилда, Хемминга, LVQ и др [13]. Эти сети имеют в составе конкурирующие слои, которые могут настраивать веса таким образом, чтобы в процессе обучения выделялись кластеры, сразу по нескольким признакам.

Класс задач, решаемых методами нейросетевой кластеризации и классификации достаточно широк. Это разнообразные системы доступа, часто связанные с распознаванием признаков, системы анализа экономических и технических характеристик систем, геотегов, уязвимостей, неполадок и т. д. [14, 15, 16, 17, 18, 19, 20, 21]. Данные для классификаторов могут подготавливаться обычными методами выделения признаков из аудиоданных или изображений, но могут и генерироваться самими сетями, например, в случае применения сверточных нейронных сетей.

От пользователя, конечно, требуется набор эвристических знаний о том, как следует отбирать и подготавливать данные, выбирать нужную архитектуру сети и интерпретировать результаты, однако уровень знаний, необходимый для успешного применения нейронных сетей, гораздо скромнее, чем, например, при использовании традиционных методов статистики.

1.2 Нейронные сети в АСНИ по обработке изображений

Разработка алгоритма для классификации объектов в компьютерном зрении, сопровождаясь очень сложной логикой, зачастую не оправдывает усилия, потраченные на его разработку, при этом совершенно не гарантируя успешного результата. Поэтому нейронные сети, являясь универсальным инструментом для классификации, все чаще применяются в решениях, связанных с обработкой изображений. Причем круг решаемых задач огромен и определяется источниками и характером изображений: медицинские рентгеновские снимки, изображения биометрических признаков человека, метеорологические карты, изображения звездного неба, снимки с электронного микроскопа, изображения, полученные с искусственных спутников земли [22, 23, 24, 25]. В основном используются сверточные нейронные сети различных архитектур. Качество распознавания зависит от качества предварительной обработки изображения, алгоритма выделения характерных признаков, количества слоев нейронной сети, функций активации, объёма входных данных, функции обучения, все эти компоненты подбираются индивидуально для каждой задачи. На сегодняшний день разработано большое количество библиотек для глубокого обучения, со своими плюсами и минусами. Наиболее крупные и распространенные из них: TensorFlow, Theano, Keras, Caffe, Torch, OpenCV, MatlabNeuralNetworksToolbox с ImageProcessing. Две первых – низкоуровневые библиотеки, предназначенныедля вычислений с использованием графов потока данных и возможностей видеоадаптера, поддерживают Python и C ++. Keras работает поверх Theano или Tensorflow, это фреймворк специально для создания нейронных сетей. Caffe – фреймворк для работы со сверточными нейронными сетями и решения задач компьютерного зрения с интерфейсами Python и Matlab. Torch представляет собой фреймворк на библиотеках C/C++ для решения задач глубокого обучения с интерфейсом на языке Lua [26]. При этом библиотеки непосредственно для обработки изображений либо встроены в нейросетевые библиотеки (OpenCV), либо устанавливаются отдельно (NeuralNetworksToolbox и ImageProcessing). В качестве предварительной

обработки часто используются преобразование в полутоновое изображение, фильтрация шумов, корректировка яркости и контрастности, бинаризация, скелетизация, выделение признаков специфическими алгоритмами (например, в распознавании лиц, алгоритм Виолы-Джонса). Причем каждый из этих этапов обработки может быть осуществлен множеством различных алгоритмов, от выбора которых будут зависеть результаты распознавания [27].

Большинство исследований, посвященных данной проблеме сосредоточено на принципиальной возможности решения задачи распознавания тех или иных объектов на изображениях, а точность распознавания, скорость работы и обучения, ресурсоемкость не рассматриваются [28, 29, 30, 31]. В тех работах, где представлены такие результаты, отмечено, что точность классификации в разработанных системах зависит от конкретной задачи и в среднем достигает 60-90 %, а скорость работы классификаторов зависит от возможности использования видеоадаптера и многопоточности и в среднем составляет 5-7 секунд [32, 33].

Поскольку нейронная сеть может ошибаться, в областях, связанных с жизнью человека, будь то управление средством повышенной опасности, системы компьютерного обнаружения аварийной ситуации, обнаружение признаков серьёзных заболеваний на медицинских снимках рекомендуется использовать нейронные сети совместно с надежными и хорошо зарекомендовавшими себя системами. Например, в случае современных автомобилей, это использование лидаров; в случае определения раковых клеток – заключение врача [34, 35].

1.3 Надежность нейросетевых алгоритмов АСНИ

Анализируя имеющиеся результаты исследований, можно сделать вывод, что на сегодняшний день существует 2 основных метода генерирования вредоносных данных для введения в заблуждения нейросетевых классификаторов: градиентный спуск и генетические алгоритмы [36]. На рисунке 1.1 представлена предлагаемая автором классификация.

Градиентный спуск с добавлением статического белого шума по всему изображению (Gradient Descent with static White Noise - GDWN). В одной из первых работ, был предложен метод искажения пикселей изображения чтобы классификатор принял ошибочное решение ошибки [37].

Рисунок 1.1 - Методы генерирования вредоносных данных для нейросетевых классификаторов

Для создания фиктивных изображений используется алгоритм обратного распространения В основе данного метода лежит тот эмпирический факт, что, если искажения не превышают некоторого минимального значения, то классификатор не игнорирует их, как, казалось бы, предусмотрено логикой работы нейронной сети, а учитывает при отнесении объектов к тому или иному классу. Уравнение, описывающее такую классификацию:

$$w^T \tilde{x} = w^T x + w^T \varepsilon \ , \qquad (1.1)$$

где, x – входные данные, предназначенные для введения нейросети в заблуждение, $\tilde{x}$ – выходные данные классификатора по неизменённому изображению, ε – специальный вектор, добавляемый к исходным входным данным

таким образом, чтобы вся сеть приняла ошибочное решение о классификации, который определяется как:

$$\varepsilon = sign\big(\nabla x J(\theta, x, y)\big)\, \varepsilon, \tag{1.2}$$

где ∇x – это градиенты (относящиеся к входным данным), J– функция оценки, используемая для обучения нейросети, θ – параметры нейросетевой модели, x – входные данные, y – целевые выходные данные, то есть «ошибочный» класс.

Поскольку все функции модели сети являются дифференцируемыми, значения градиента можно легко найти с помощью метода обратного распространения ошибки. Изменив входные данные и выяснив с помощью анализа градиентов, какое направление нужно изменить, можно заставить сеть неправильно классифицировать изображение.

Градиентный спуск с добавлением ограниченного количества искажающих данных (Gradient Descent with limited amount of Distortion Data - GDDD).

В дальнейших исследованиях доказано, что достаточно изменить только небольшую часть изображения, чтобы получившееся изображение было ошибочно отнесено к другому классу [38, 39].

Для начальных входных данных x, если вероятность принадлежности x к классу t равна $f_x()$, то задача описывается уравнением:

$$maximaz\, e\big(e(x)\big)\, f_{adv}\big(x + e(x)\big), \tag{1.3}$$

где, adv – оптимизируемый вредоносный класс, $e(x)$ – фальшивые данные, которые добавляются ко входным данным. Однако в этом случае у $e(x)$ есть ограничение:

$$\|e(x)\|_0 \ \leq d. \tag{1.4}$$

Эта формула означает, что количество элементов в векторе x должно быть меньше настраиваемого параметра d.

Недостатком представленных выше методов является необходимость наличия у исследователя исчерпывающих данных об архитектуре и методах обучения нейронной сети. Для реальных атак на готовые системы не всегда возможно получить такие данные.

Дифференциальный генетический алгоритм (Differential Evolutionary Algorithm – DEA). Если у атакующего нет доступа к архитектуре сети, вместо градиентного спуска используется дифференциальный генетический метод. Берутся образцы «родителей», на основе которых генерируются образцы «потомков», а из них потом оставляют лишь те, что получились лучше «родительских» для решения задачи. Затем выполняется новая итерация генерирования образцов «потомков». Этот подход позволяет находить правильные значения для вредоносных входных данных, не зная, как работает модель нейронной сети, и даже если модель не будет дифференцируемой, в отличие от предыдущих методов. При этом значение параметра *d* может быть равно 1, то есть изменять можно только один пиксель [40].

Заплатка на основе дифференциального генетического алгоритма (Differential Evolutionary Algorithm Patch – DEAP)

Еще один метод, вредоносная заплатка, – совсем новая методика генерирования фальшивых изображений [41]. В предыдущих методах вредоносные данные добавлялись к исходным входным данным и напрямую зависели от этих данных. Авторы вредоносной «заплатки» подбирают данные, которые подходят для всех изображений. Термин «заплатка» в данном случае нужно понимать, как изображение меньшего, относительно входных изображений, размера, которое накладывается поверх входных, чтобы ввести в заблуждение классификатор. Оптимизация работает в соответствии с уравнением1.4.5.

$$\hat{p}=\text{argmax}_p E_{x\sim X,t\sim T,l\sim L}[logPr(\hat{y}|A(p,x,l,t))], \quad (1.5)$$

где, $\hat{p}$ – подобранная заплатка, $\hat{y}$ – целевой (т. е. ошибочный) класс, $A(p,x,l,t)$ – функция применения заплатки. Она случайным образом решает, куда и как накладывать заплату на входное изображение. В ней p – сама заплатка, x –

входное изображение, *l* – место наложения заплатки, t – преобразование заплатки (например, масштабирование или вращение).

Важно отметить, что в этом случае, система обучается на всех изображениях в наборе данных, и на всех возможных преобразованиях. В итоге заплатка вводит в заблуждение классификатор на всех изображениях набора данных. Это главное отличие данного метода от трех предыдущих. Можно подобрать заплатку, которая работает на большом наборе изображений. В авторской работе приведено изображение заплатки, которую можно накладывать на изображения для обмана классификатора. На рисунке 1.2 представлены результаты такого рода экспериментов с фотографиями физических объектов. На сегодняшний день алгоритм DEAP является самым перспективным поскольку позволяет генерировать заплатки, которые в последствии можно использовать многократно с различными изображениями и объектами.

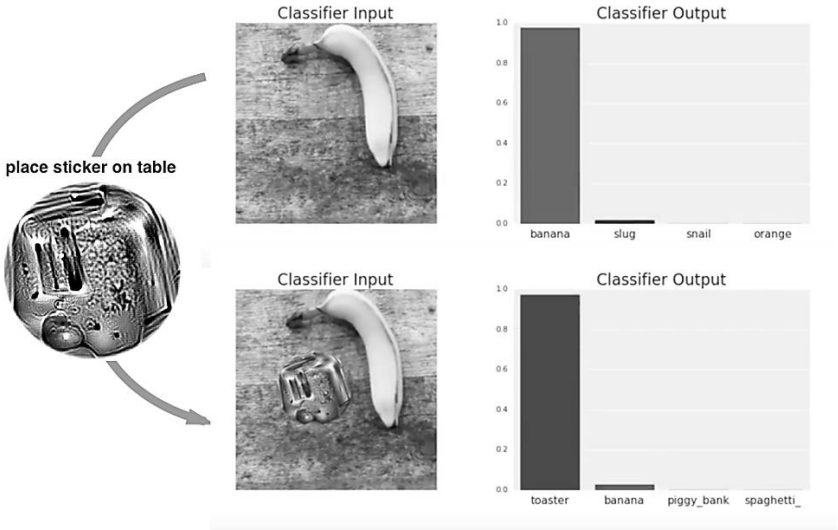


Рисунок 1.2 - Если разместить на столе заплатку, сгенерированную как класс «тостер», рядом с бананом, то эта фотография классифицируется как тостер с вероятностью 99% (нижний график)

Недостатком этого метода является то, что заплатку видит человек, но не «видит» нейронная сеть и значит такого рода фальсификации можно применять только в полностью автоматизированных системах, без участия человека.

Атаки на системы распознавания речи. Ввести взаблуждение можно не только модели, классифицирующие изображения. М. Альзантот, З. Ван и Б. Шриваставайн на конференции ICASSP, посвященной акустике, распознаванию речи и обработке сигналов, показали, как можно ввести в заблуждение модели автоматического распознавания речи [42]. Особенность звуковых данных заключается в том, что входные вектора нельзя легко изменить с помощью метода обратного распространения ошибки. Причина в том, что входные данные для звукового классификатора проходят через косинусное преобразование, для которого требуется вычислять коэффициенты косинусного преобразования Фурье. Функция для вычисление этих коэффициентов не является дифференцируемой, поэтому оптимизировать входные данные с помощью метода обратного распространения ошибки нельзя. Используется генетический алгоритм, аналогичный рассмотренному выше для обработки изображений. Ученые продемонстрировали сгенерированный ими сигнал, который человеком воспринимается как шум, но распознается голосовым классификатором, как команда перейти на вредоносный сайт.

Все описанные выше методы пока ни разу не применялись злоумышленниками для реальных атак. Это связано с тем, что данные алгоритмы все еще находятся на стадии исследования и не выходят за пределы лабораторий. Кроме того, приложения, связанные с денежными транзакциями, компрометация которых как раз и может заинтересовать злоумышленников, в настоящее время, нечасто применяют полностью автоматизированную нейросетевую аутентификацию. Однако со временем классификаторы на глубоких нейронных сетях будут использоваться всё шире, в том числе для аутентификации пользователей банков и интернет-магазинов. Поэтому важно знать, каким образом можно «обмануть» нейросети, заставив их неверно классифицировать объекты, для оценки границ предположительного введения в заблуждение подобных систем и нахождения способов защиты от подобных атак, еще на этапе разработки.

В качестве решения проблемы можно предложить построение сетей, обучающихся в том числе на ложноположительных примерах. Однако вновь

созданные фальшивые изображения могут вводить в заблуждение сеть даже в этом случае, кроме того, данный подход ведет к повышению вычислительной сложности алгоритмов. Хотя алгоритмы глубоких нейронных сетей все-таки отличны от тех, которые работают в человеческом мозге, исследования в области физиологии также могут дать множество новых идей. Одна из них – это бинокулярное зрение. Использование дополнительных данных, например, от вторичных стереокамер и 3D сканеров также может создать защиту от атак на нейронные классификаторы.

1.4 Выводы по разделу 1

1. Задачи аппроксимации, интерполяции, экстраполяции и классификации решаются с помощью сетей с радиально-базисными функциями, вероятностных сетей, обобщенно-регрессионных сетей, многослойного персептрона, сетей с конкурирующими слоями, сверточных сетей. Основное внимание необходимо уделить архитектуре сети, методу обучения, выделению и подготовке данных для обучения и входных данных.

2. Для решения задач по обработке изображений используются специальные библиотеки, которые предоставляют возможность создавать собственную архитектуру или пользоваться готовыми моделями, которые можно модифицировать и обучать, однако такого рода решения затратны по ресурсам.

3. Большинство исследований сосредоточено на архитектуре нейронных сетей, алгоритмах обучения и принципиальной возможности решения задачи распознавания для того или иного вида данных, однако не столь широко исследованы такие характеристики как быстродействие, время обучения, объем использованной оперативной и постоянной памяти, точность распознавания, что является ключевыми характеристиками в случае применения нейронных сетей в автономных автоматизированных системах на нейронных сетях.

4. Автоматизированные системы на нейросетевых алгоритмах могут быть уязвимы к атакам, основанным на создании состязательных примеров, той же, либо

сторонней нейронной сетью, на сегодняшний день успешность таких атак невелика и в среднем ниже 60 %.

5. Состязательные примеры, генерируемые методом градиентного спуска, более эффективны для атак класса «белый ящик», если же архитектура атакуемой сети неизвестна, то применяют генетические алгоритмы.

6. На сегодняшний день не существует универсального метода защиты от такого рода атак, для каждого вида атаки необходимо подбирать соответствующий метод. Наиболее успешны атаки на полностью автоматизированные системы, работающие без вмешательства человека и с ростом числа подобных систем, проблема уязвимости нейросетевых алгоритмов становится актуальной.

Остается нерешенным вопрос о возможности применения в автономных автоматизированных системах малозатратных нейросетевых решений, а также снижения количества параметров глубоких сверточных сетей для использования в микроконтроллерных системах. При этом актуален вопрос уязвимостей, которыми обладают подобные системы, поскольку при получении доступа к управлению технологическим процессом или к системе аутентификации злоумышленник может нанести ощутимый вред, однако исследований, обобщающих, сравнивающих и тестирующих атаки на нейросетевые алгоритмы на сегодняшний день крайне мало.

РАЗДЕЛ 2
СОКРАЩЕНИЕ КОЛИЧЕСТВА ПАРАМЕТРОВ НЕЙРОННОЙ СЕТИ РЕДУКЦИЕЙ ПРОИГРАВШИХ НЕЙРОНОВ

В разделе выдвинута гипотеза о возможности уменьшения параметров сети удалением нейронов, веса которых не меняются в процессе обучения, с дальнейшим дообучением, при этом точность классификации не должна уменьшаться. Такое обрезание нейронной сети можно проводить однократно и итеративно вне зависимости от архитектуры сети и методов обучения.

Выдвинутая гипотеза проверена на 1 полносвязной сети и 5-ти сверточных сетях различной архитектуры. Все исследованные архитектуры нейронных сетей при обрезании, с последующим обучением в среднем показали увеличение точности на 2-3 % при уменьшении параметров сети на 70-80 %. Доказано, что такое обрезание дает большее увеличение точности, чем рандомное удаление весов, предложенное в известных работах [43, 44, 45, 46].

Экспериментально подтверждено, что описанное в ранее опубликованных работах рандомное обрезание дает падение точности распознавания в среднем 21 % за итерацию, больше, чем падение точности при обрезке неуспешных нейронов (в среднем 2,9 % за итерацию).

Предложено использовать данную методику в автоматизированных системах, основанных на нейросетевых алгоритмах, для уменьшения ресурсоемкости и увеличения быстродействия без потери точности.

2.1 Методика обрезания проигравших нейронов

Для автоматизированных систем очень важно, чтобы нейронные сети занимали как можно меньше места в постоянной памяти и по минимуму использовали оперативную память и ресурсы процессора. Существующие методы сокращения нейронных сетей позволяют уменьшить количество параметров обученных сетей более чем на 80%, уменьшая требования к памяти и экономя

вычислительные мощности без ущерба для точности [47, 44, с. 7]. Тем не менее, архитектуру, создаваемую обрезкой сети (удалением нейронов, слабо влияющих на результат), трудно обучить с самого начала. Обычно сеть сначала обучают, а затем обрезают, либо делают это в процессе обучения (dropout). В этом случае прореживают сеть, обнуляя случайные веса [45, с. 6, 46, с. 8]. Можно предположить, что техника сокращения параметров позволяет найти такие нейроны, которые не вносят вклада в дальнейшее обучение, и зануление (обрезание) которых делает сеть способной эффективно обучаться. Можно сформулировать следующую гипотезу: плотная, произвольно инициализированная сеть, содержит подсети (выигравшие нейроны), которые при отдельном обучении могут достигать точности, сопоставимой с точностью исходной сети в аналогичном количестве итераций (эпох). Эти выигравшие нейроны имеют подходящие начальные веса, что делает обучение особенно эффективным. Такую обрезку с последующим дообучением можно проводить однократно либо итеративно, в цикле. Предлагается алгоритм определения подсетей из выигравших нейронов и серия экспериментов, которые подтверждают гипотезу об эффективности обучения таких обрезанных кусков сетей.

2.2 Проверка гипотезы об обрезании проигравших нейронов

Для проверки гипотезы использовались стандартные задачи распознавания рукописных цифр сетью прямого распространения с архитектурой LeNet, обучаемой на наборе MNIST и классификация объектов по 10 классам сверточными сетями Conv-2, Conv-4 и Conv-6, обучаемыми на наборе данных CIFAR-10 (самолет, автомобиль, птица, кошка, олень, собака, лягушка, лошадь, корабль и грузовик). Эти сети не являются очень глубокими, что позволяет поставить множество экспериментов с обучением, но в то же время в них есть достаточно слоев и параметров для редукции. Затем были проведены эксперименты на сетях VGG-19 и Resnet-18 на CIFAR10, как на более глубоких и

имеющих большое количество параметров. Архитектуры участвовавших в эксперименте сетей представлены в таблице 2.1.

Таблица 2.1 – Архитектуры протестированных нейронных сетей

Нейронная сеть	Lenet	Conv-2	Conv-4	Conv-6	ResNet-18	VGG-19
Сверточные слои		64, 4 пулинг	64, 64 пулинг 128, 128 пулинг	64, 64 пулинг 128, 128 пулинг 256, 256 пулинг	16, 3x[16,16] 3x [32,32] 3x [64,64]	2x64 пул. 2x128 пул. 4x256 пул. 4x512 пул. 4x512
Полносвязные слои	300,100, 10	256,256, 10	256,256, 10	256,256, 10	avg-pool,10	avg-pool,10
Всего весов /Весов сверт. слоев	266 т.	4,3 млн. /38 т.	2,4 млн. /260 т.	1,7 млн. /1,1 млн.	274 т. /270 т.	20 млн.
Итерации /batch	50 т./60	20 т./60	25 т./60	30 т./60	30 т./128	112 т./64
Оптимизатор	Adam $1{,}2e^{-3}$	Adam $2e^{-4}$	Adam $3e^{-4}$	Adam $3e^{-4}$	SGD 0,1-0,01-0,001-Momemtum 0,9	
Процент обрезанных нейронов	Полносв. 20%	Сверт. 10% Полносв. 20%	Сверт. 10% Полносв 20%	Сверт. 15% Полносв. 20%	Сверт. 20% Полносв. 0%	Сверт. 20% Полносв. 0%

Ядра свертки во всех сетях 3x3. Квадратными скобками обозначены residual-связи вокруг слоев, характерные для сетей типа ResNet.

Критерий ранней остановки, который используется в работе, это итерация, при которой наблюдается минимальная разница между значениями тестовой выборки и выборки для обучения, обычно этот критерий применяется для проверки сети на переобучение. В итоге количество итераций до ранней остановки показывает за сколько итераций сеть удовлетворительно обучилась, и переобучение еще не произошло. Вторым критерием является точность, то есть какой процент изображений классифицировано правильно. Ошибки проверочных и тестовых данных вначале процесса обучения уменьшаются, достигают

минимума, а затем начинают увеличиваться, когда модель проходит большую часть выборки для обучения, при этот как правило резко падает точность.

На рисунке 2.1 представлены результаты тестирования сети Lenet, обрезанной с учетом весов, выигравших нейронов и обрезанных с инициализацией случайным образом.

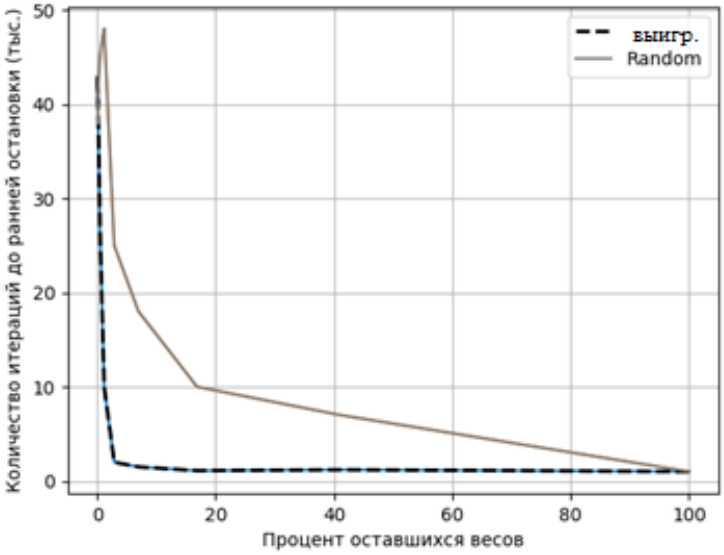


Рисунок 2.1 – Количество итераций до ранней остановки для рандомного обрезания и обрезания весов с учетом выигравших нейронов

Случайная выборка моделирует эффект неструктурированной обрезки, используемой в предшествующих работах [47, с. 8, 48, 49]. Эксперименты на сверточных сетях показали, что чем меньше сеть, тем медленнее обучение и тем ниже точность теста. Это можно увидеть на рисунке 2.2. Видно, что при обрезании до 20 процентов точность сохраняется, при этом рандомное обрезание дает резкое падение точности, в то время как обезание, с учетом выигравших нейронов сохраняет очень долго высокую точность. Таким образом можно увидеть, что существуют меньшие подсети, которые обучаются с самого начала и учатся, по крайней мере, так же быстро, как их более крупные аналоги, при этом достигая аналогичной точности. Следовательно, если начать обучение полной сети, остановить обучение, не достигая переобучения (ранняя остановка), определить подсети, значения весов которых не менялись на протяжении нескольких

предыдущих итераций, исключить их из процесса обучения и заново обучить сеть, то точность распознавания как минимкм не уменьшится.

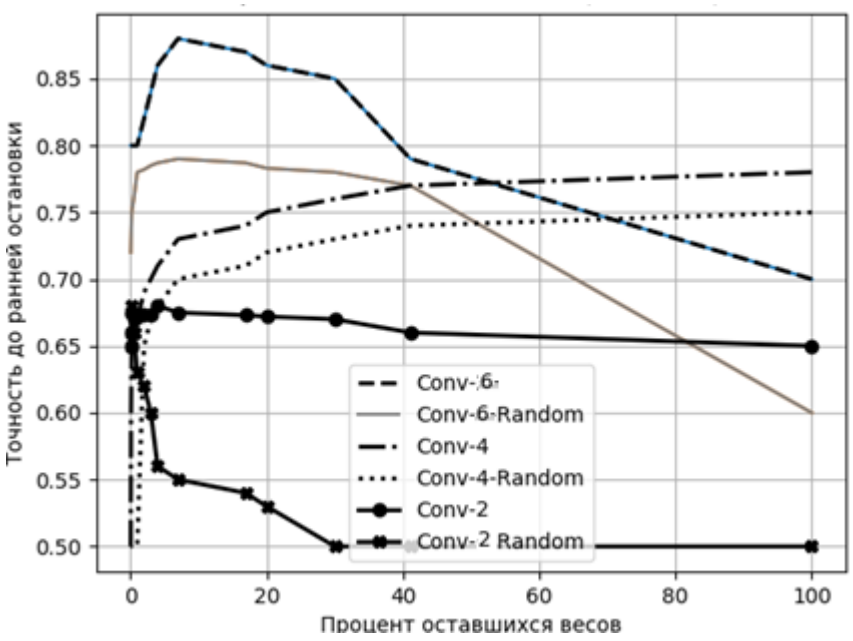


Рисунок 2.2 – Точность до ранней остановки для рандомного обрезания и обрезания весов с учетом выигравших нейронов для сверточных сетей

На основании полученных результатов можно сформулировать **следующую гипотезу**: случайно инициализированная, плотная нейронная сеть содержит подсеть, которая инициализируется так, что - при отдельном обучении можно получить точность теста, такую же, как и для исходной сети после обучения в течение одинакового количества итераций.

Рассмотрим плотную прямую нейронную сеть $f\ (x;\ \theta)$ с начальными параметрами $\theta=\theta_0 \sim D_0$. При оптимизации со стохастическим градиентным спуском (SGD) на тренировочном наборе f достигается минимум функции потери на проверочном множестве k на итерации j с тестовой точностью a. Кроме того, рассмотрим обучение $f\ (x;\ m \in \theta)$ с маской $\mathrm{m} \in \{0;1\}^{|\theta|}$ по параметрам таким, что их инициализация равна $m \in \theta_0$. При оптимизации с SGD на том же обучающем наборе (с фиксированным m), f достигает минимума функции потери на проверочном k_1 на итерации j_1 с тестовой точностью a_1. Согласно гипотезе можно предположить, что существует **m**, для которого $j_1 < j$ (с соразмерным временем тренировки), $a_1 < a$ (с

соразмерной точностью) и $m << \theta$ (гораздо меньшим количеством параметров). Для того, чтобы обнаружить такие подсети необходимо исключать нейроны, веса которых не меняются в процессе обучения и оставлять те веса, которые меняются наиболее сильно. Обозначим эти обучаемые подсети, $f(x; m0)$, инициируем начальные веса, полученными в процессе первоначального обучения, и получим сеть, которая будет успешнее обучаться, поскольку начальные значения весов инициированы не случайным образом, а с учетом весов и связей способных к обучению. Количество таких нейронов зависит от архитектуры сети и находится экспериментально. Мы определяем подходящую подсеть, обучая сеть и сокращая ее веса наименьшей величины. Оставшиеся необрезанными соединения составляют необходимую архитектуру.

Теперь можно сформулировать алгоритм поиска оптимизированных подсетей:

1. Произвольно инициализировать нейронную сеть f (x; ϴ) (где $\theta=\theta_0 \sim D_0$).
2. Обучать сеть в течение j итераций, получив набор параметров θ_j.
3. Сократить p% параметров в θ_j, создав маску m.
4. Установить значения оставшихся параметров равными 0, создав новую сеть f (x; m_0).

Это можно сделать один раз: сеть обучается один раз, p% весов обрезается, и оставшиеся незадействованными веса обнуляются. Можно проводить этот процесс итеративно, то есть многократно обучать, сокращать и обнулять веса за n эпох; каждая эпоха раунд сокращает $p^{1/n}$ % весов, из тех, которые остаются от предыдущей эпохи. Результаты экспериментов показывают, что итеративная обрезка находит подходящие подсети, которые сравнимы по точности с исходной сетью, чем одноразовая обрезка.

В экспериментах данный алгоритм применялся к полносвязным и сверточным нейронным сетям, различной архитектуры с использованием различных стратегий оптимизации (стохастический градиентный спуск, моментум

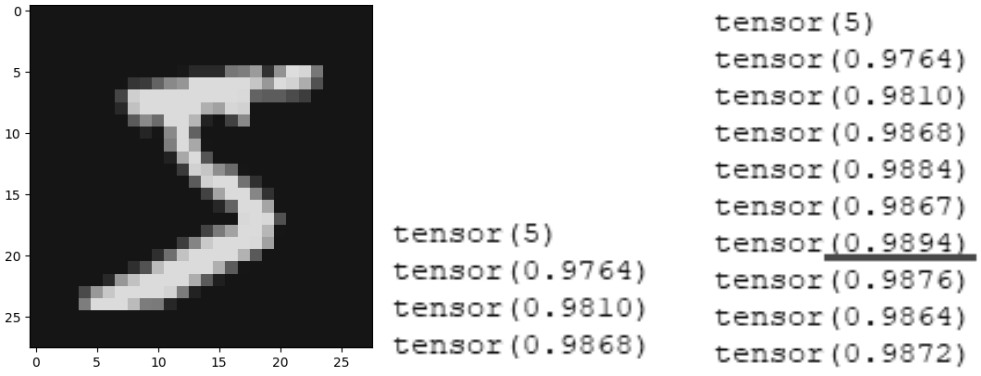
и Адам) с регуляционными методами, такими как dropout, нормализация бача и residual соединениями, характерными для сетей архитектуры ResNet. Подробные архитектуры участвовавших в экспериментах сетей представлены в таблице 1. Когда использовалась рандомная обрезка, особенно в неглубоких сетях, успешно обучаемые подсети получались довольно редко. В более глубоких сетях результаты обрезки для нахождения успешных подсетей оказались чувствительными к скорости обучения: требуется некоторое время, чтобы найти успешно обучаемые подсети. Найденные успешные подсети в основном составляют 10-20% (или меньше) от размера оригинала.

Причем достигая такого размера, они сохраняют, а иногда и превышают точность исходной сети на тестовой выборке. При случайной повторной инициализации выделенные подсети работают намного хуже, то есть структура сама по себе не может объяснить успех такого рода сетей.

Таким образом стохастический градиентный спуск ищет в процессе обучения подмножество хорошо инициализированных весов. Плотные, произвольно инициализированные сети легче обучать, чем разреженные сети, возникающие в результате обрезки, потому что получается больше возможных подсетей, которые будут гораздо успешнее обучаться, чем изначальные сети.

Рассмотрим более подробно результаты экспериментов. Начнем с полносвязной нейронной сети архитектуры Lenet-300-100, обучаемой на наборе MNIST. Набор данных MNIST состоит из 60000 учебных примеров и 10 000 тестовых примеров. Был выбран набор из 5000 примеров из данных Training, а оставшиеся 55 000 примеров были отложены для остальных частей исследования. Набор Training подается на обучение по бачам, размером 60 экземпляров. На каждой эпохе обучения, весь тренировочный набор перемешивается. На рисунке 2.3 представлены результаты распознавания цифры 5, случайно выбранной из тестового датасета сетью, с сигмоидной активационной функцией и с активационной функцией ReLU. В качестве оптимизатора был использован алгоритм Адам. В первой строчке показано какая цифра распознана, а далее выведена динамика изменения точности на тестовой выборке в процессе обучения.

В случае использования сигмоидной активационной функции точность обучения достигает 0,9868 и процесс останавливается, в случае использования функции ReLU точность на тестовой выборке достигает максимума, а затем снижается. Это тот самый момент, когда сеть достигла максимальной точности, а дальше уже идет переобучение. В этот момент необходимо остановить обучение, зафиксировать веса, которые изменялись при последней итерации, а затем всем остальным весам присвоить 0, полученную подсеть записать и затем продолжить обучение на обучающей выборке.



а) б) в)

Рисунок 2.3 – Результат распознавания рукописных цифр нейронной сетью Lenet-300-100: а) – рукописная цифра 5, б) – результат распознавания с сигмоидной активационной функцией, в) – результат распознавания с активационной функцией ReLu, красным отмечен момент переобучения, то есть тот самый момент, когда обучение надо остановить

На рисунке 2.4 представлен процесс обучения обрезанной сети и графики зависимости точности распознавания от количества итераций обучения полной сети с различными активационными функциями и обрезанной согласно предложенной методике сетью с активационной функцией ReLu. Видно, что при обучении точность для полной сети с ReLu сначала растет, причем быстрее, чем у обрезанной, достигает значения 0,988 и потом падает, то есть наступает переобучение. Точность обрезанной сети растет плавно и достигает значения 0,99.

При обрезке, на 0 были заменены 931 нейрон в первом скрытом слое, 30 во втором скрытом слое и 2 в третьем скрытом слое, что составило 55860 параметров, то есть количество параметров сократилось на 21 %.

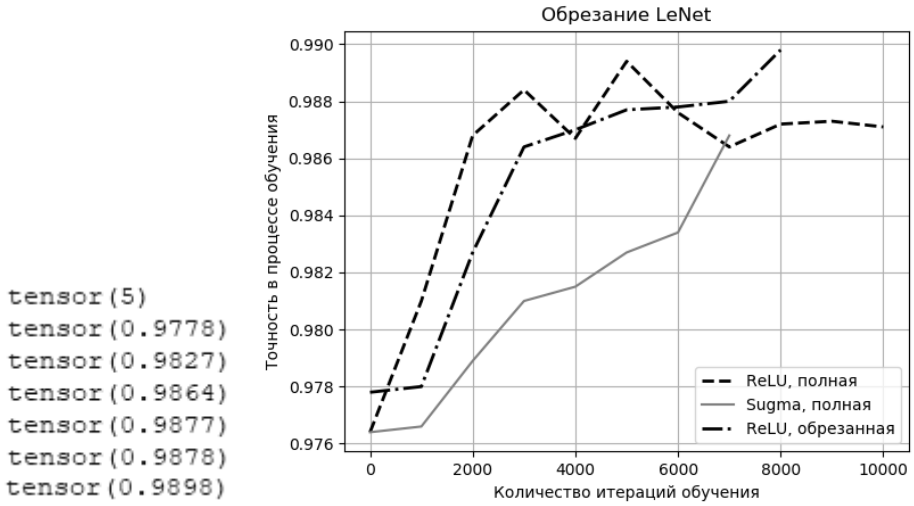


Рисунок 2.4 – Процесс обучения нейросети LeNet полной, с различными активационными функциями и обрезанной

На рисунке 2.5 показано, как меняется точность в процессе обучения после обрезки. Здесь представлены результаты усредненных 5-ти тестов, когда полная сеть училась, затем обрезалась и опять училась.

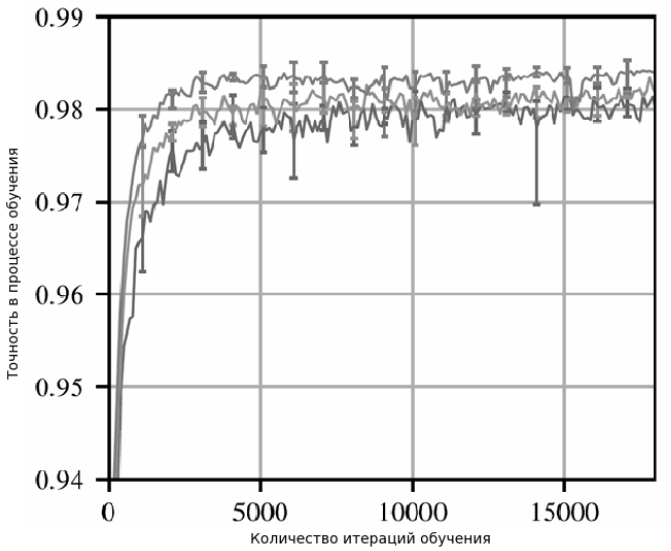


Рисунок 2.5 – Изменение точности для LeNet с разной степенью обрезки. Сверху вниз – 21% обрезки, 51 % обрезки, полная сеть

Причем инициализация весов полной сети происходила каждый раз случайным образом, поэтому результаты были различны для различных тестов. Разброс показан на графике как погрешность относительно среднего значения. Из графика видно, что точность, при обрезании 51±3 % весов увеличивается и достигает максимального значения медленнее, чем при обрезании 21±1 % весов, но быстрее, чем при обучении полной сети. Подобное поведение неоднократно повторялось в ходе экспериментов.

В предыдущих экспериментах обрезание проводилось однократно. Далее представлены результаты для итеративного обрезания, когда цикл обучение – останов- обрезание – обучение повторялся до достижения 3 % оставшихся от первоначальных 100 % весов. При этом на каждой итерации обрезалось одинаковое количество весов.

Для сравнения были проведены эксперименты, в которых после останова веса обрезались случайным образом, без учета успешности предыдущего цикла обучения, как указано в ранее опубликованных работах [47, с. 7, 48, с. 1059, 49, с. 6]. На рисунках 2.6, 2.7 представлены результаты для однократного и итеративного обрезаний, с рандомным обрезание весов и с предложенной стратегией обрезания весов менее успешных нейронов.

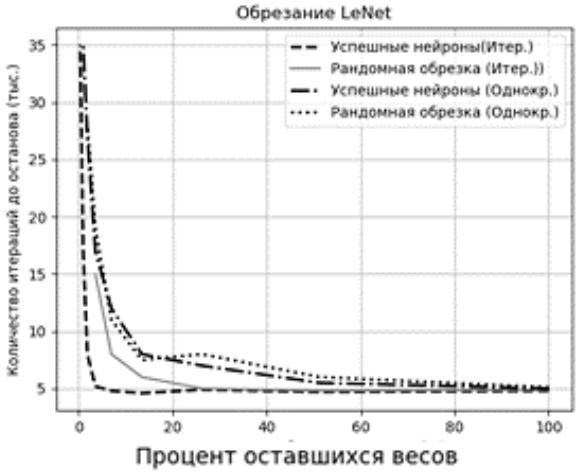


Рисунок 2.6 – Количество итераций до ранней остановки для рандомного обрезания и обрезания весов с учетом выигравших нейронов при использовании однократной и итеративной обрезки

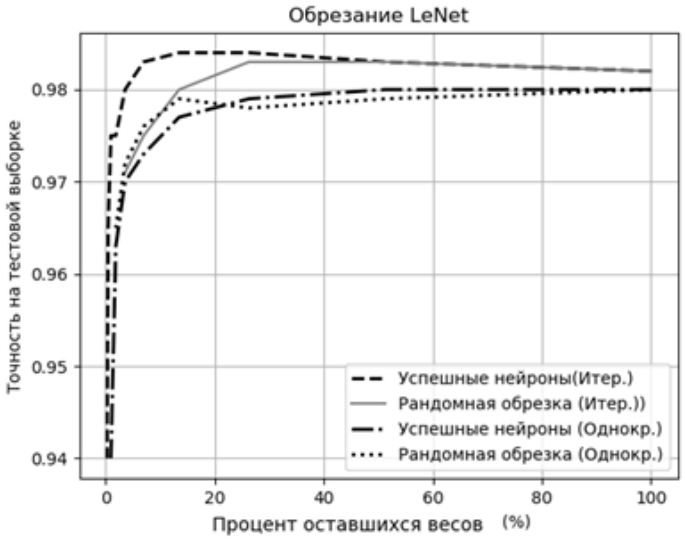


Рисунок 2.7 – Точность до ранней остановки для рандомного обрезания и обрезания весов с учетом выигравших нейронов при использовании однократной и итеративной обрезки

В отличие от обрезки выигравшных нейронов, рандомно обрезанные сети учатся медленнее, чем исходная сеть, и теряют точность тестирования после небольшой обрезки. Средняя точность для итеративной рандомной обрезки падает быстрее (в среднем 21 % за итерацию), чем точность при обрезке неуспешных нейронов (в среднем 2,9 % за итерацию). Этот эксперимент подтверждает гипотезу об итеративном обрезании с учетом успешных нейронов. Такое обрезание не дает значительного падения точности, в то время как случайная повторная инициализация сразу ведет к ухудшению производительности и уменьшению точности. Причем итеративное обрезание дает лучшие результаты, чем однократное. Хотя итеративная обрезка извлекает меньшие подсети, повторное обучение на них проходит успешнее. Они учатся быстрее и достигают более высокой точности как на обучающей, так и на тестовой выборках при меньших размерах сети.

Но, возможно, это свойство характерно только полносвязных сетей? Эксперименты были продолжены на сверточных сетях. Гипотеза была проверена на сверточных сетях Conv-2, Conv-4 и Conv-6, решающих задачу классификации

изображений, с обучающим датасетом CIFAR10 архитектуры которых представлены в таблице 2.1. Они представляют собой уменьшенные варианты VGG. Сети имеют два, четыре или шесть сверточных слоев, за которыми следуют два полносвязных слоя, с макспулингом после каждых двух сверточных слоев. Сети охватывают диапазон от почти полносвязных до традиционных сверточных сетей, с менее чем 1% параметров в сверточных слоях в Conv-2 и почти до двух третей параметров в сверточных слоях для Conv-6. Исходное количество параметров – 20000 для Conv-2, 25000 для Conv-4 и 30000 для Conv-6. Комплект данных CIFAR10 состоит из 50000 обучающих примеров 32x32 (три цветовых канала) и 10000 тестовых примеров. Был случайным образом отобран набор для теста из 5000 примеров из обучающего набора. Размер бача 60. Обучающие данные перемешиваются. Сети Conv-2, Conv-4 и Conv-6 инициализируются с помощью Гауссовой инициализации. Алгоритм оптимизации – Адам. Скорость обучения подбиралась экспериментально для каждой сети. При низких скоростях обучения (например, скоростей обучения 0,001 для Conv-4 и Conv-6, и 0,01 для Conv-2), точность изначально снижалась, а затем повышалась до более высоких уровней. Под скоростью обучения здесь понимается коэффициент определяющий порядок того, как корректируются веса с учётом функции потерь в градиентном спуске. При использовании низкого коэффициента затрачивается много времени на сходимость. При высоких скоростях обучения (например, скоростей обучения 0,005 и 0,008 для Conv-2 и Conv-4) время раннего останова никогда не уменьшалось и оставалось стабильным.

Результаты экспериментов представлены на рисунках 2.8, 2.9. Они в основном повторяют результаты, полученные на полносвязной сети Lenet, когда сеть обрезается, она учится быстрее и тестовая точность повышается по сравнению с исходной сетью. Итеративная обрезка дает увеличение скорости обучения (достижение минимума функции потерь) в среднем в 3,5 раза для Conv-2, 2,5 раза для Conv-4 и 3,5 раз для Conv-6 по сравнению с одноразовой обрезкой. Что же касается точности, то она возрастает при итерационной обрезке в тех же пропорциях.

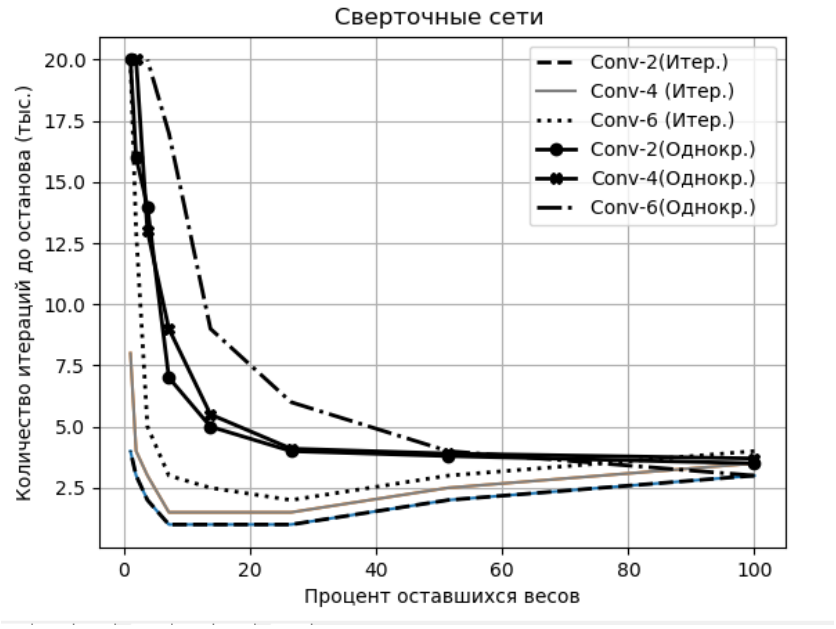


Рисунок 2.8 – Количество итераций до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов при использовании однократной и итеративной обрезки для сверточных сетей с разным количеством сверточных слоев

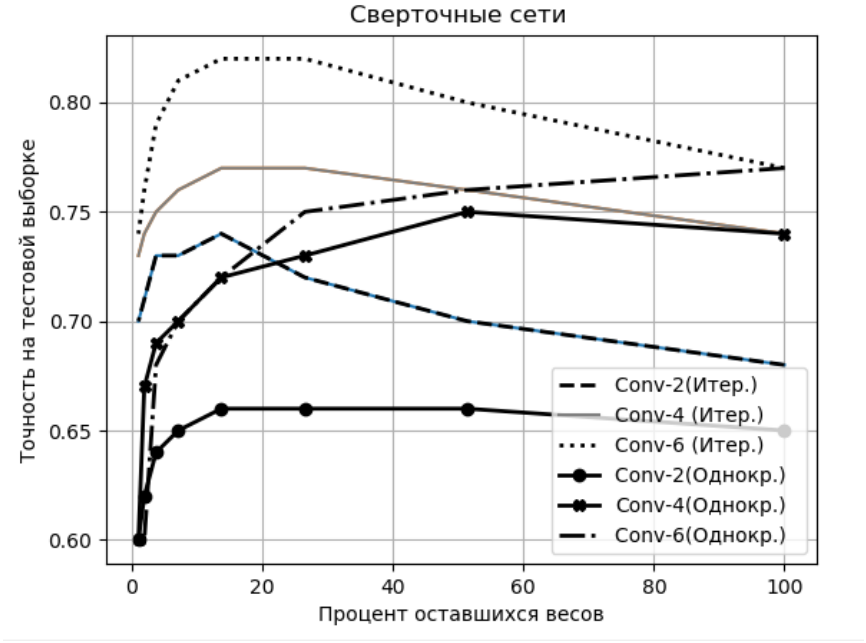


Рисунок 2.9 – Точность до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов при использовании однократной и итеративной обрезки для сверточных сетей с разным количеством сверточных слоев

Как видно из рисунка 2.9 эффект зависит от общего количества параметров сверточных и полносвязных слоев. То, что кривые для Conv-6 лежат выше, связано с изначально лучшей способностью к классификации полной сети с большим количеством сверточных слоев. Однако обрезка не так сильно влияет на Conv-4, как на Conv-2 и Conv-6. Можно сделать вывод, что к большему увеличению точности приводит обрезка сетей с преобладанием либо полносвязных, либо сверточных слоев, в то время как наличие одинакового количества тех и других снижает данный эффект, хотя увеличение точности все равно имеет место.

Дальнейшие эксперименты проводились на сетях с большим количеством сверточных сетей, довольно сложной архитектуры – VGG-19 ResNet -18 [50, 51]. Архитектуры этих сетей также представлены в таблице 2.1. Поскольку эти сети предназначены для классификации изображений, использовалась все та же база данных для обучения CIFAR10. Для Resnet-18 и VGG-19, стратегия обрезания весов была немного изменена: если в предыдущих экспериментах сокращались отдельные веса, которые не менялись за последнюю итерацию обучения, то для этих более глубоких сетей сокращение проводилось глобально, удалялись веса наименьшей величины одновременно для всех сверточных слоев. В этих более глубоких сетях некоторые слои имеют гораздо больше параметров, чем другие. Например, первые два сверточных слоя VGG-19 имеют 1728 и 36864 параметров, в то время как последний имеет 2,35 миллиона. Когда все слои обрезаются одинаково, эти меньшие слои становятся узкими местами, мешая оптимизировать сеть наилучшим образом. Глобальная обрезка позволяет избежать этого недостатка.

Сеть VGG-19 состояла из 5 групп сверточных слоев 3x3, первые, четыре из которых с последующим макспулингом и последний с пулингом усреднения. Сеть имела один последний полносвязный слой, соединяющий результат усредненного пула с выходом. Размер бача 64. Для VGG-19, обучаемой на CIFAR10 был использован следующий режим обучения:160 эпох (112 480 итераций) с SGD с моментумом 0,9 и снижением скорости обучения в 10 раз до 80 с 120 эпох. Эта сеть имеет 20 миллионов параметров.

Сверточные слои сокращались со скоростью 20 % за итерацию, а 5120 в выходном слое не сокращалось, поскольку это затруднило бы классификацию. Скорость обучения 0,01. Resnet-18 состояла из 20 слоев, за каждым сверточным слоем следовало 9 пар остаточных (residual) сверточных слоев, усредненный слой пулинга, и полносвязный слой. Обучающий набор был разделен на 45 000 обучающих примеров и 5000 тестовых примеров. Обучающие данные случайно перемешивались, рандомно применялись четырехпиксельные падинги и кадрирование. Размер бача 128, пакетная нормализация. Обучение – SGD с моментумом 0,9.

Были использованы 3 варианта обучения с разным количеством эпох. Для каждого варианта проводилось обучение со скоростью 0,1, 0,01 и с предварительным обучением в течение 10 тысяч эпох и скоростью 0,01. Использовавшиеся режимы обучения гораздо хуже, чем стандартный для таких сетей, но пришлось пожертвовать точностью, чтобы провести как можно больше экспериментов и исследовать более широкой набор параметров. Для первоначальной инициализации использовалось распределение Гаусса. Сверточные слои сокращались со скоростью 20% за итерацию. Не сокращались 2560 параметров остаточных слоев и 640 параметров в полносвязном выходном слое.

На рисунках 2.10, 2.11, 2.12 представлены результаты измерения точности распознавания на 3-х этапах обучения с различным количеством эпох и разными скоростями обучения.

В третьем случае сеть предварительно обучалась 10 тысяч итераций, и только потом ее начинали обрезать. Из графиков видно, что в этом случае получается самая высокая точность. Чем меньше скорость обучения, тем выше точность.

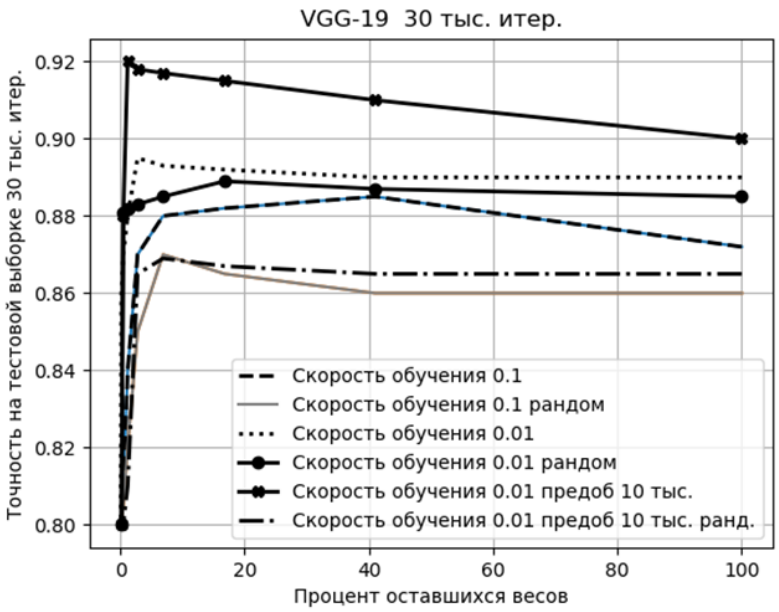


Рисунок 2.10 – Точность до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов на тестовой выборке для VGG-19 обучение 30 тыс. итераций с различной скоростью обучения

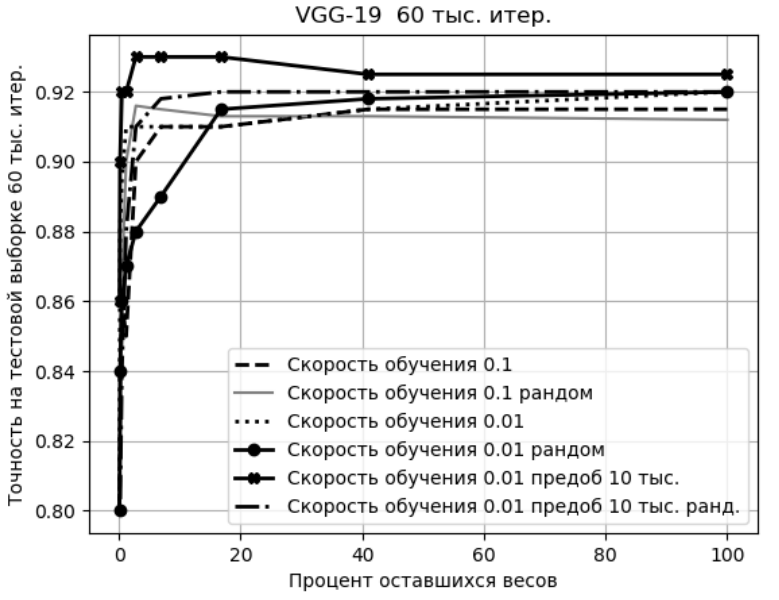


Рисунок 2.11 – Точность до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов на тестовой выборке для VGG-19 обучение 60 тыс. итераций с различной скоростью обучения

Как и в предыдущих случаях для глубокой сверточной сети VGG-19 обрезание с учетом выигравших нейронов дает преимущество по сравнению с рандомной инициализацией.

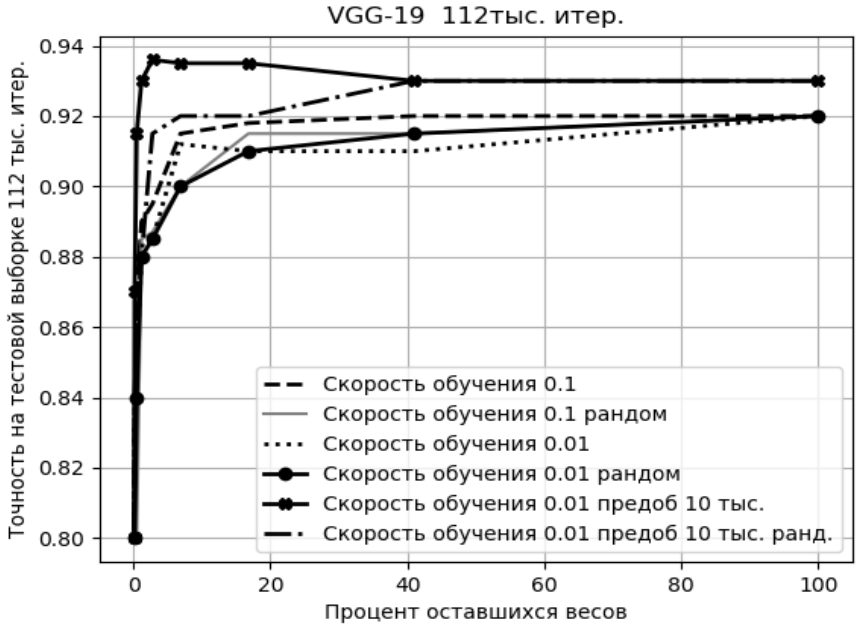


Рисунок 2.12 – Точность до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов на тестовой выборке для VGG 19 обучение 112 тыс. итераций с различной скоростью обучения

Самая большая точность достигается на этапе обучения 112 тысяч итераций, с предварительным обучением на 10 тысяч итераций и скоростью обучения 0,01 и составляет 0,935. Более высокая скорость обучения, не дает увеличения производительности. Причем только в случае предобучения появляется эффект увеличения точности с обрезанием весов. Причем увеличение точности происходит и в случае обрезания весов проигравших нейронов, так и в случае случайного обрезания, однако в последнем случае меньше начальная точность для полной сети. Для остальных режимов обучения при обрезании получается потеря точности. Поэтому в случае использования глубоких нейронных сетей можно рекомендовать сначала предобучить полную сеть, а затем уже начать обрезание.

На рисунках 2.13, 2.14, 2.15 представлены результаты тех же самых экспериментов с сетью ResNet-18.

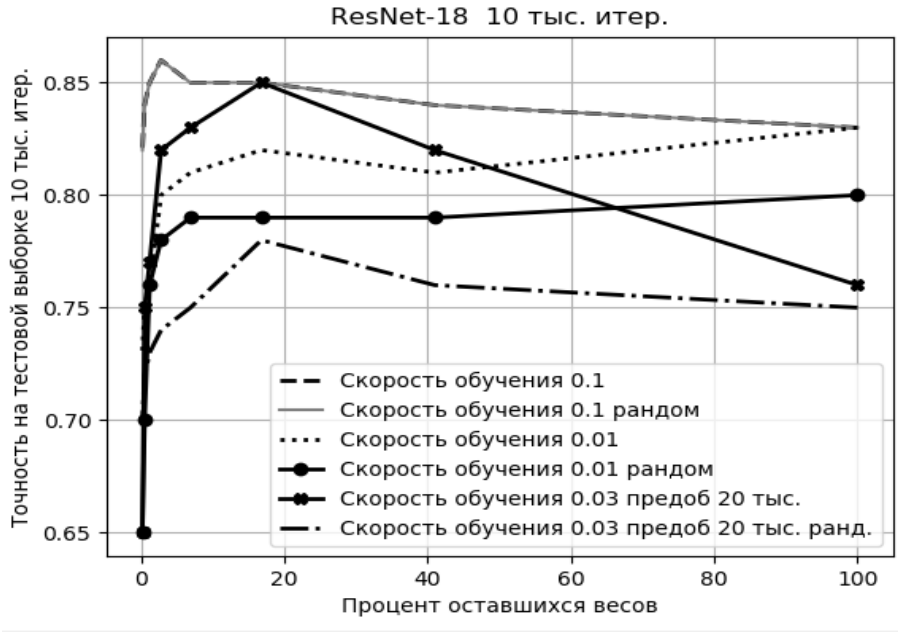


Рисунок 2.13 – Точность до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов на тестовой выборке для ResNet-18 обучение 10 тыс. итераций с различной скоростью обучения

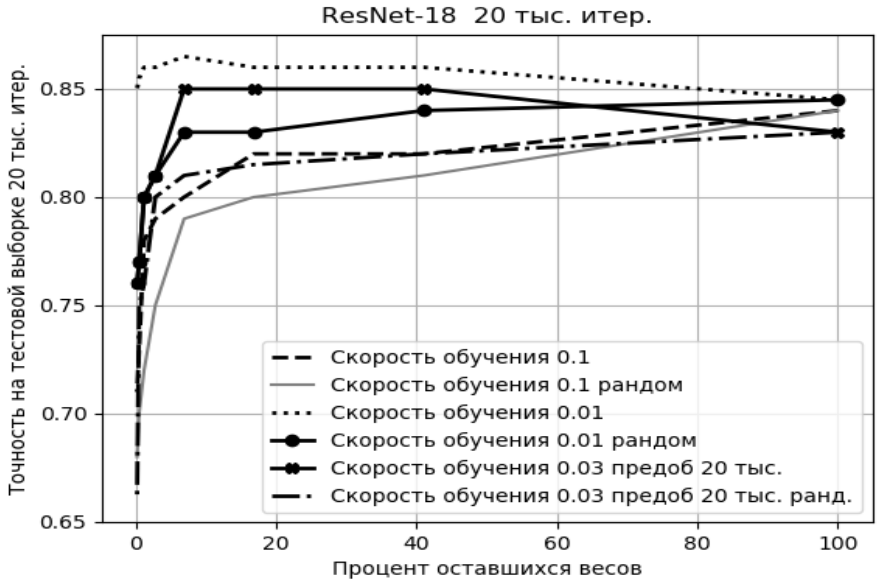


Рисунок 2.14 – Точность до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов на тестовой выборке для ResNet-18 обучение 20 тыс. итераций с различной скоростью обучения

На рисунке 2.15 видно, что в случае ResNet-18 предобучения не дает такого же эффекта, как в случае VGG-19.

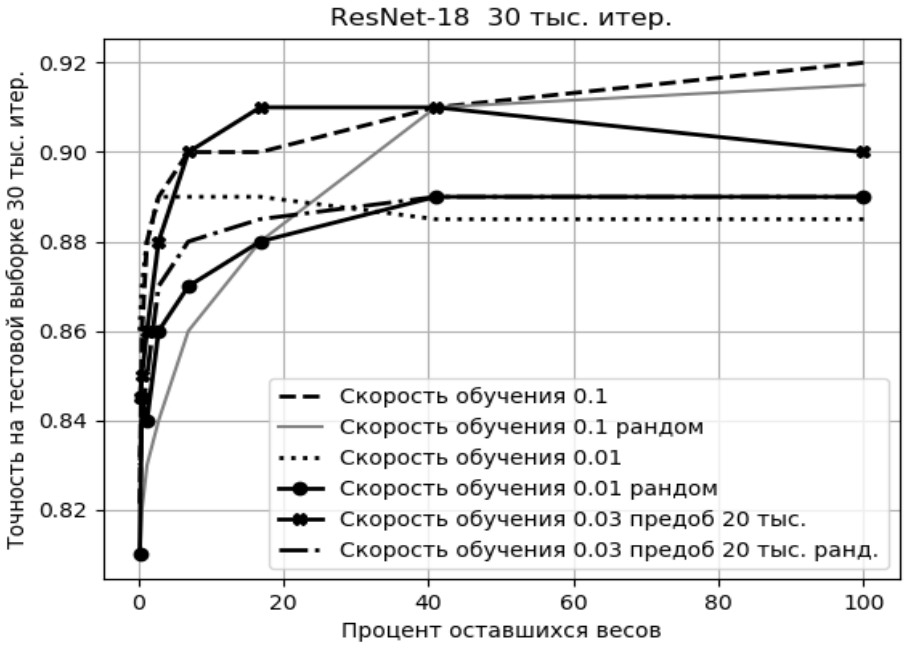


Рисунок 2.15 – Точность до ранней остановки рандомного обрезания и обрезания весов с учетом выигравших нейронов на тестовой выборке для ResNet-18 обучение 30 тыс. итераций с различной скоростью обучения

Наоборот, точность получается больше для меньшей скорости обучения. Только на последнем этапе обучения предобучение дает небольшой выигрыш в точности по сравнению с обучением со скоростью 0,1. Однако общая тенденция, когда обрезанные проигравшие нейроны дают точность лучше, чем полные сети и лучше, чем рандомно обрезанные сети сохраняется.

2.2 Выводы по разделу 2

1. Выдвинута гипотеза о том, что если приравнять к нулю веса нейронов обученной нейронной сети, которые не изменяются в процессе обучения (обрезать сеть), а затем снова начать обучать такую обрезанную сеть, то можно добиться увеличения точности, при уменьшении количества параметров сети. Обрезание нейронной сети можно проводить однократно и итеративно. Данное свойство не зависит от архитектуры сети и методов обучения.

2. Выдвинутая гипотеза была проверена на полносвязной сети LeNet и сверточных сетях Conv-2, Conv-4, Conv-6 с различным количеством сверточных

слоев, глубоких нейронных сетях VGG-19 и ResNet-18 с различными особенностями архитектуры и обучения. В качестве алгоритма обучения применялся стохастический градиентный спуск, с алгоритмом оптимизации Адам. Все исследованные архитектуры нейронных сетей при обрезании, с последующим обучением показали увеличение точности распознавания, причем при обрезании более, чем на 30 %, точность увеличивалась на 2-3% при уменьшении параметров сети на 70 %. С усложнением архитектуры нейронных сетей процент обрезанных весов нейронов без ухудшения точности увеличивался и достигал 80 % (для VGG-19). Такое обрезание дает большее увеличение точности, чем рандомное удаление весов, предложенное в работах [47, с. 9, 49, с. 7].

3. Итеративное обрезание в среднем позволяет сокращать количество параметров на 10 % больше, чем однократное обрезание. Выигрыш в точности распознавания зависит от алгоритма оптимизации, скорости обучения, набора данных для обучения и конкретной модели. Существует некоторый предел, который зависит от архитектуры сети и в среднем составляет 90 %, после которого точность падает.

4. Выдвинутая гипотеза подтверждена экспериментально. Предложенную методику целесообразно использовать в автоматизированных системах, основанных на нейросетевых алгоритмах, для уменьшения ресурсоемкости и увеличения быстродействия без потери точности.

5. В экспериментах были рассмотрены только визуально-ориентированные задачи классификации для небольших наборов данных (MNIST, CIFAR10). Не были рассмотрены большие наборы данных, требующие значительных вычислительных ресурсов для обучения. Обнаруженный эффект, что в более глубоких сетях (Resnet-18 и VGG-19) итеративное обрезание с большей вероятностью приводит к увеличению точности в случае, если полная сеть прошла предобучение, а затем подверглась обрезанию пока не нашел объяснения. Данный факт планируется исследовать в ходе дальнейшей работы.

РАЗДЕЛ 3
РАЗРАБОТКА АСНИ БЕЗОПАСНОСТИ ЛИЧНОСТИ НА ОСНОВЕ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ

В разделе описаны разработанные 2 автоматизированные системы обработки формализованных данных на нейронных сетях типа двухслойный персептрон и рекуррентной сети Элмана и 4 автоматизированные системы обработки изображений. Показано, что функционирование таких систем зависит от методики подготовки данных для обучения, алгоритма обучения, выбранной архитектуры или модели и достигает точности 85-97 %. Для систем, работающих с изображениями, где требуется предварительная обработка, был разработан собственный алгоритм бинаризации и применен детектор Харриса для выделения признаков. Показано, что нейронные сети, состоящие из 2-х, 3-х скрытых слоев не требуют оптимизации и могут быть применены в микроконтроллерных системах или одноплатных мини-компьютерных системах типа Raspberry Pi. Нейронные сети глубокого обучения, с наличием большого количества сверточных слоев требуют оптимизации, и применение оптимизации методом редукции проигравших нейронов позволяет уменьшить количество параметров на 70-80% без потери точности. Результаты исследований внедрены в систему аутентификации и доступа на базе монитора видеодомофона М-480М с системой на кристалле Allwiner A-13 (достигнутая точность распознавания 89 %) и в учебный процесс в лабораторном практикуме по дисциплине «Основы информационной безопасности».

3.1 АСНИ обработки формализованных экспериментальных данных

3.1.1. АСНИ анализа файлов логов на основе нейронной сети и логистической регрессии

Одной из задач по обработке формализованных данных является анализ лог-файлов для обнаружения несанкционированных вторжений. Реализация системы

обнаружения вторжений поиском аномалий в лог файлах нейронной сетью с использованием статистического анализа и библиотеки scikit-learn0.22.1 представлена в данном разделе [52]. Библиотека содержит несколько моделей для статистического анализа. Поэтому необходимо проанализировав задачу, выбрать подходящую модель и подготовить данные для обучения. Для подготовки данных необходимо извлечь характеристики несанкционированного доступа из эталонного лог-файла и использовать их в качестве обучающих примеров для классификации поведения пользователя системы как нормального или неавторизованного. Кроме того, необходимо, чтобы нейронная сеть могла рассчитывать вероятность во время классификации. Это нужно в тех случаях, когда невозможно однозначно определить наличие вторжения и в этом случае устанавливается, например, такое правило: если вероятность классификации составляет 70 % или меньше, попросить о вмешательстве человека. Такие правила могут быть определены для уменьшения количества неудачных обнаружений. В качестве математической модели для реализации автоматизированной системы обнаружения вторжений была использована логистическая регрессия. Результатом выполнения логистической регрессии является предсказанный класс и класс вероятности классификации. Пусть вероятность классификации представлена как числовое значение в диапазоне от 0 до 1. Логистическая регрессия требует обучения с учителем, и поэтому необходим учебный набор данных.

В логистической регрессии, прямая линия, которая разделяет данные, называемая границей решения, она рассчитывается на основе особенностей данных, и так они классифицируются. Используя логистическую регрессию, создается система обнаружения вторжений, которая сможет различать нормальное и аномальное общение с сервером и в случае аномалии сможет определить тип конкретной атаки с вычисленной вероятностью.

Для обучения предпочтительно использовать реальные данные, собранные на конкретном сервере. В данной работе для обучения используется набор данных KDD Cup 1999 Data [53]. Он содержит следы различных вторжений, в среде военной сети США. Этот набор данных содержит большое количество аномальных

сообщений и обычных данных связи, таких как поиск, вторжение и атака. Эти данные были опубликованы под названием «KDD-99» в материалах пятой международной конференции по обнаружению и получению данных. Затем разработанная модель применяется к лог-файлам конкретного целевого сервера.

После загрузки и распаковки архива kddcup.data_10_percent.gz появляется файл формата CSV «kddcup.data_10_percent_corrected», который содержит около 500 000 данных. В нем имя столбца отсутствует, и его трудно будет обрабатывать, поэтому загружаем файл kddcup.names, содержащий имена столбцов, и добавляем строку столбца в первую строку kddcup_train.csv, как показано на рисунке 3.1. В этом файле одна строка представляет данные об одном соединении. «С 1-го по 41-й столбец» – это значение свойства, а «42-й столбец» – метрика (тип соединения = ответ).

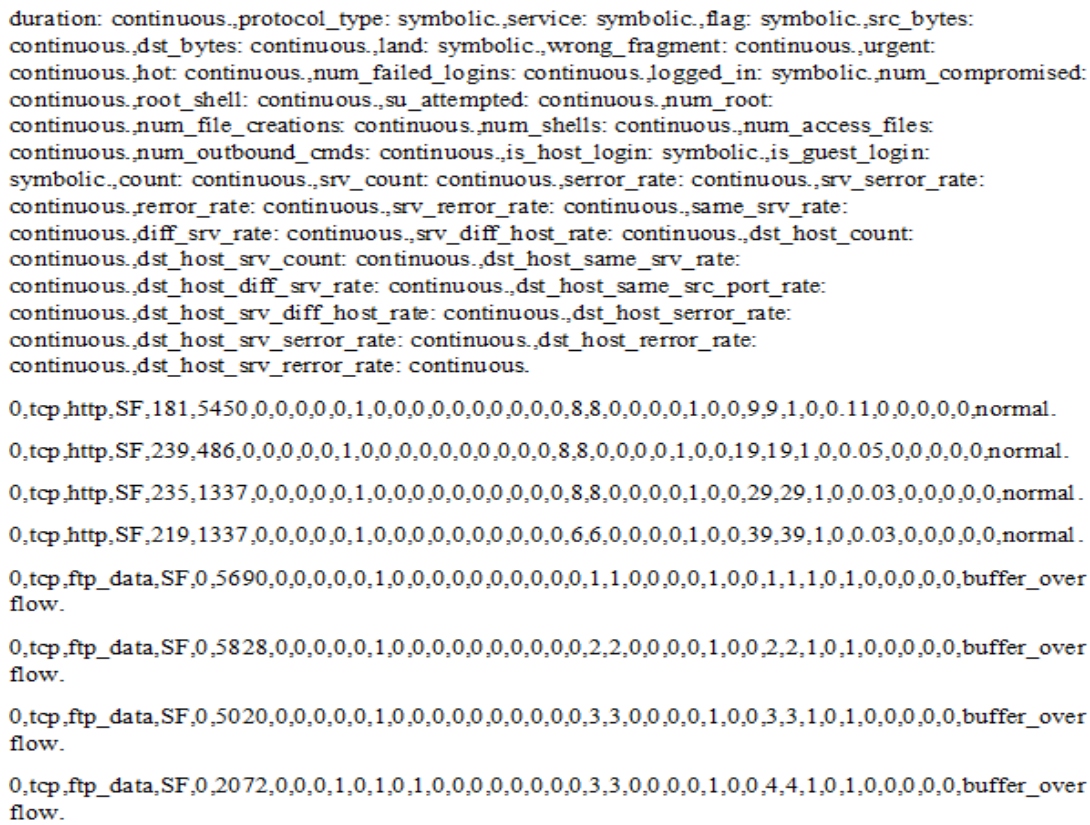


Рисунок 3.1 – Формат файла данных kddcup_train.csv для обучения

В файле есть 23 типа метрик, но для облегчения задачи оставим только 5 типов метрик. То есть будем считать, что набор данных для обучения содержит 5 типов соединений. Эти метрики представлены в таблице 3.1.

Таблица 3.1 – Метрики для обнаружения атаки

Метрика	Описание
normal.	Нормальное общение.
nmap.	Данные во время сканирования Nmap.
Teardrop.	Данные, когда выполняется DoS (teardrop).
buffer_overflow.	Данные указывающие на переполнение буфера.
guess_password.	Данные при выполнении подбора пароля.

Теперь нужно выбрать характерные функции для каждой метрики. Учебные данные «kddcup_train.csv» содержат 41 тип функций, но, если используются

функции, которые не способствуют классификации, точность классификации модели может снизиться. Для выбора функции был использован метод главных компонентов PCA. Аналитическая точность этого метода зависит от характеристик целевых данных, поэтому возможно найти объективно эффективные характеристики. Выбранные автором функции показаны в таблице 3.2.

Таблица 3.2 – Характерные функции для каждой метрики

Метрика	Функция	Описание
nmap.	dst_host_serror_rate	SYN частота ошибок.
nmap.	dst_host_same_src_port_rate	Скорость соединения для одного и того же порта.
Teardrop.	wrong_fragment	Количество неверных фрагментов.
buffer_overflow.	duration	Время подключения к хосту (сек).
buffer_overflow.	logged_in	Успешный вход
guess_password.	dst_host_rerror_rate	REJ частота ошибок.
guess_password	num_failed_logins	Количество неудачных попыток входа.

Метод главных компонент уменьшает количество признаков исходных данных обучения, сохраняя при этом их смысл. Полученный эффект уменьшения количества данных приведет к уменьшению объема вычислений и облегчения визуализации данных.

В случае слишком большого количества обучающих данных может не только увеличится время обучения и объем используемой памяти, но сеть может переобучиться и модель станет неправильно классифицировать данные отличные от обучающих, из-за чрезмерной адаптации.

Кроме данных для обучения необходимо загрузить тестовые данные. На них сеть как раз и проверяет, нет ли переобучения. Эти данные загружаются и обрабатываются так же, как и выборка для обучения. Файл kddcup_test.csv.

Общий алгоритм работы программы для обучения и распознавания вторжений, написанной на языке Python 3.6.5 заключается в следующем.

1. Загрузка обучающей и тестовой выборок данных.
2. Обучение с использованием данных обучения (создание модели).
3. Наблюдение за возможностью обнаружения вторжения с использованием тестовых данных (оценка модели).
4. Вывод результатов оценки.

Использовалась библиотека машинного обучения scikit-learn для реализации логистической регрессии. Команды from sklearn import linear_model и from sklearn import metrics импортируют классы для логистической регрессии и оценки точности. Затем загружаются данные для обучения и нормализуются функцией train.mean. Затем загружаются тестовые данные. Указываются номера используемых столбцов.

```
X_test = df_test.iloc[:, [0, 7, 10, 11, 13, 35, 37, 39]]
X_test = (X_test - X_test.mean()) / X_test.mean()
y_test = df_test.iloc[:, [41]]
```

Определяется модель логистической регрессии. В качестве аргумента C=1e5 используется параметр регуляризации.

```
logreg = linear_model.LogisticRegression(C=1e5)
```

linear_model.LogisticRegression

Наконец загружается модель и проводится обучение.

logreg.fit(X_train, y_train)

Для тестирования модели были взяты реальные файлы логов, полученные с сервера Debian 9 на VestaCP. Они содержат данные о всех пяти видах атак. Поскольку файлы имеют формат, отличный от kddcup.csv (Рисунок 19), (здесь данные отделяются не запятыми, а квадратными скобками и двоеточием), нужно его переформатировать. Для тестового примера это было сделано вручную, а для автоматизированной системы написана программа для обработки лог-файлов. Результаты предварительной обработки сохраняются в файле log.csv.

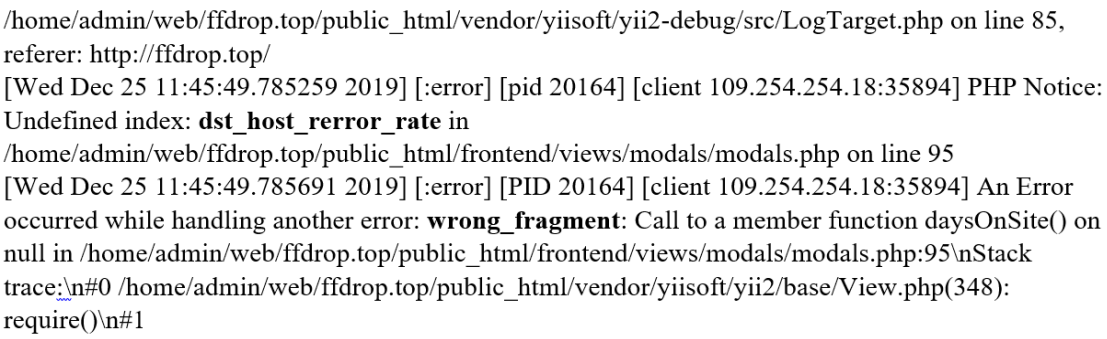

/home/admin/web/ffdrop.top/public_html/vendor/yiisoft/yii2-debug/src/LogTarget.php on line 85, referer: http://ffdrop.top/
[Wed Dec 25 11:45:49.785259 2019] [:error] [pid 20164] [client 109.254.254.18:35894] PHP Notice: Undefined index: **dst_host_rerror_rate** in /home/admin/web/ffdrop.top/public_html/frontend/views/modals/modals.php on line 95
[Wed Dec 25 11:45:49.785691 2019] [:error] [PID 20164] [client 109.254.254.18:35894] An Error occurred while handling another error: **wrong_fragment**: Call to a member function daysOnSite() on null in /home/admin/web/ffdrop.top/public_html/frontend/views/modals/modals.php:95\nStack trace:\n#0 /home/admin/web/ffdrop.top/public_html/vendor/yiisoft/yii2/base/View.php(348): require()\n#1

Рисунок 3.2 – Формат реального лог-файла.

Получаем классификацию вероятностей.

probs = logreg.predict_proba(X_test)

Получаем результат классификации и выводим результат классификации в консоль.

```
y_pred = logreg.predict(X_test)
for predict, prob in zip(y_pred, probs):
print('{0}\t{1}\t{2}'.format(y_test.iloc[idx, [0]].values[0], predict, np.max(prob)))
idx += 1
```

"y_test", "y_pred", "probs" Выводятся по одному на консоль. Пример вывода представлен на рисунке 3.3. Точность классификации составляет 89,9%. Из этих результатов видно, что «normal», «nmap», «teardrop» и «guess_passwd» правильно классифицируются. Однако, глядя на вероятности частичной классификации «nmap», «teardrop» и «guess_passwd», видим, что эти цифры равны «69%, 50%, 38%». Кроме того, можно видеть, что «buffer_overflow» вряд ли правильно

классифицирован и неправильно определяется как «normal» или «guess_passwd». Это говорит о том, что нужно найти более подходящих функции для этих метрик или переобучить нейронную сеть на лог файлах целевого сервера. Необходимо либо пересмотреть функции для метрики «buffer_overflow», либо некорректный набор данных для классификации.

```
train_time   : 10.235417526819267 [sec]
predict_time : 0.023557101303234518 [sec]
score : 0.8996588708933895
------------------------------------------------------------
label              predict          probability
normal.            normal.          0.9998065736764143
normal.            normal.          0.9998065736764143
normal.            normal.          0.9998065736764143
normal.            normal.          0.9917140229898096
nmap.              nmap.            0.695479203
nmap.              nmap.            0.695479203
nmap.              nmap.            0.695479203
nmap.              nmap.            0.695479203
buffer_overflow.   normal.          0.999980104
buffer_overflow.   guess_passwd.    0.333345516
buffer_overflow.   normal.          0.990855252
buffer_overflow.   guess_passwd.    0.333334087
teardrop.          teardrop.        0.999998379
teardrop.          teardrop.        0.500005699
teardrop.          teardrop.        0.500000044
teardrop.          teardrop.        0.999998157
guess_passwd.      guess_passwd.    0.478403815
guess_passwd.      guess_passwd.    0.499999391
guess_passwd.      guess_passwd.    0.38504377
guess_passwd.      normal.          0.999790317
score
```

Рисунок 3.3 – Вывод результата классификации реального файла логов нейронной сетью, обученной на данных kddcup

Если в качестве данных для обучения взять файл логов, непосредственно целевого сервера, на котором необходимо определить наличие атак, то вероятность предсказания атаки заметно повысится. Результат представлен на рисунке 3.4.

Таким образом для уверенного предсказания атаки на сервер необходимо проводить обучение сети на данных с такого же сервера, на который проведена атака, а еще лучше с того же, целевого. При этом в тестовой и обучающей выборке должно быть не менее 10 000 записей.

Автоматизированная система анализа файлов логов серверов с целью обнаружения атак, построенная на основе логистической регрессионной модели обнаруживает признаки сетевых атак с вероятностью 80-90%.

```
train_time   : 12.235417526819267 [sec]
predict_time : 0.013557101303234518 [sec]
score : 0.8796588708933895
------------------------------------------------------------
label              predict          probability
normal.            normal.          0.7998065736764143
normal.            normal.          0.7998065736764143
normal.            normal.          0.7998065736764143
normal.            normal.          0.6917140229898096
nmap.              nmap.            0.895479203
nmap.              nmap.            0.895479203
nmap.              nmap.            0.895479203
nmap.              nmap.            0.895479203
buffer_overflow.   normal.          0.599980104
buffer_overflow.   guess_passwd.    0.733345516
buffer_overflow.   normal.          0.9690855252
buffer_overflow.   guess_passwd.    0.333334087
teardrop.          teardrop.        0.999998379
teardrop.          teardrop.        0.700005699
teardrop.          teardrop.        0.700000044
teardrop.          teardrop.        0.999998157
guess_passwd.      guess_passwd.    0.678403815
guess_passwd.      guess_passwd.    0.699999391
guess_passwd.      guess_passwd.    0.68504377
guess_passwd.      normal.          0.499790317
score
```

Рисунок 3.4 – Результат классификации реального файла логов сервера нейронной сетью, обученной на данных, полученных из логов того же сервера

Поскольку все файлы логов имеют различный формат, то необходимо разработать дополнительное программное обеспечение для подготовки данных для нейросетевого классификатора, использующего логистическую регрессию. Для обучения желательно использовать файлы логов целевого сервера. точность

классификации в значительной степени зависит от точности выбора признаков. Такая система требует дополнительной настройки для работы с серверами разного типа. Использование библиотеки scikit-learn для Python значительно увеличивает объем постоянной памяти, необходимой для работы программы (30 Мб). Объем данных для обработки не должен превышать 30 000, если использовать тип float32. Эффективнее читать только нужные столбцы и строки. При обрезании файлов памяти создается копия DataFrame (пусть и на короткое время, но это может приводить к MemoryError). Методы SciKit-Learn могут напрямую обрабатывать Pandas.DataFrame – не нужно создавать еще одну копию данных в виде NumpyNDArray. После применения, предложенного в разделе 2 метода обрезания незадействованных нейронов, число параметров нейронной сети уменьшилось на 40 %, а скорость работы увеличилась на 30 %.

3.1.2 АСНИ обнаружения радиоканалов утечки информации

В процессе разработки программно-аппаратного модуля для радиоразведки в диапазоне 85-115МГц перед автором встала задача интерполяции данных, выводимых на выходные графики. [54]. Было рассмотрено несколько методов интерполяции с точки зрения качества получаемых графиков и нахождение такого, который предоставит наилучшую возможность их визуального сравнения и обнаружения источника излучения. Для сравнения были взяты спектры, полученные на тех же частотах и при тех же условиях при помощи SDR-приемника, где плавность спектральных линий обусловлена высокой частотой дискретизации. Для сравнительного анализа алгоритмов интерполяции были взяты метод наименьших квадратов, полиномиальная, сплайнами и рекуррентной нейронной сетью. Дискретные значения уровня сигнала для интерполяции можно снимать непосредственно из интерфейса программы (Рисунок 3.5). Можно получить данные для принятого и демодулированного сигналов. Поскольку наибольший интерес имеет демодулированный сигнал, то все эксперименты проводились для этих данных. Наилучшие результаты показали 2 метода – кубическими сплайнами

и рекуррентной нейронной сетью. На рисунке 3.6 представлен результат интерполяции кубическими сплайнами, при отсутствии и наличии сигнала.

Для интерполяции нейронной сетью использовалась рекуррентная нейронная сеть Элмана, так как она предоставляет возможность динамической работы с данными [55, 56, 57]. Однако подобрать подходящую архитектуру оказалось достаточно сложно.

На рисунке 3.7 представлен результат интерполяции нейронной сетью неудачной архитектуры. Видно, что кривая вообще не соответствует начальным данным.

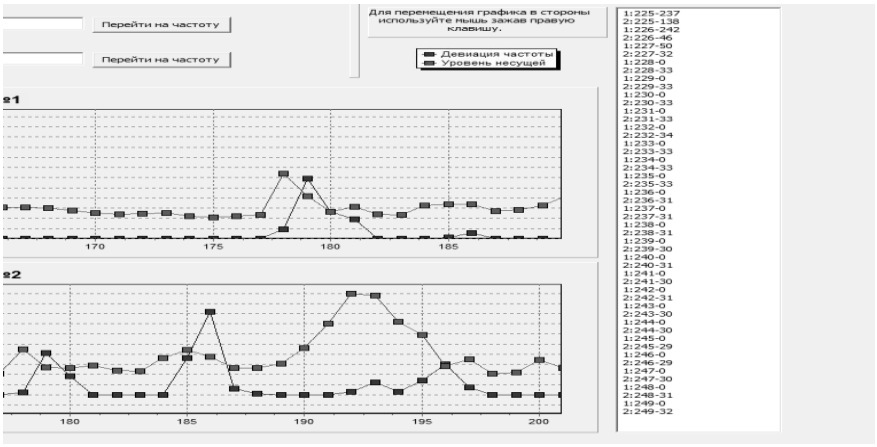


Рисунок 3.5 – Окно для получения дискретных данных для обработки

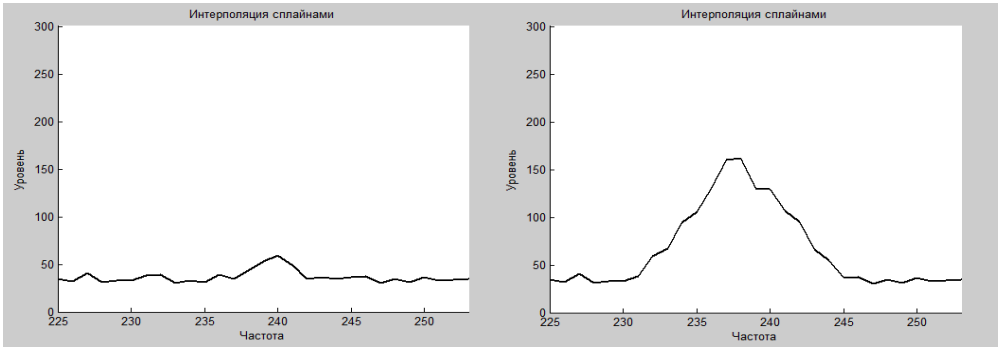


Рисунок 3.6 – Интерполяция кубическими сплайнами

Не смотря на двуслойную архитектуру и прогрессивный метод обучения данная сеть не справлялась с задачей, кроме того, процесс обучения шел около

минуты, что неприемлемо в режиме реального времени. В ходе экспериментальных исследований была подобрана следующая архитектура сети:

1 скрытый рекуррентный слой;

1 выходной слой;

Количество нейронов в обоих слоях – 40;

Функция активации – тангенциальная;

Алгоритм обучения – адаптивный градиентный спуск с импульсом;

Функция потерь – среднеквадратичная (MSE);

Количество эпох – 1000.

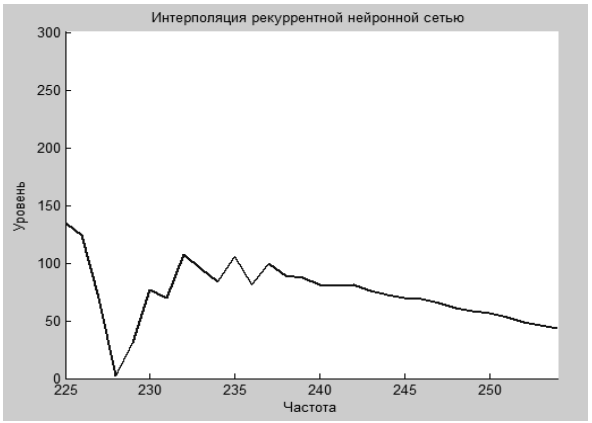


Рисунок 3.7 – Интерполяция рекуррентной нейронной сетью неудачной архитектуры

Сеть обучилась до значения ошибки 0,0001, при этом время обучения заняло 0,13 с. Результат интерполяции представлен на рисунке 3.8.

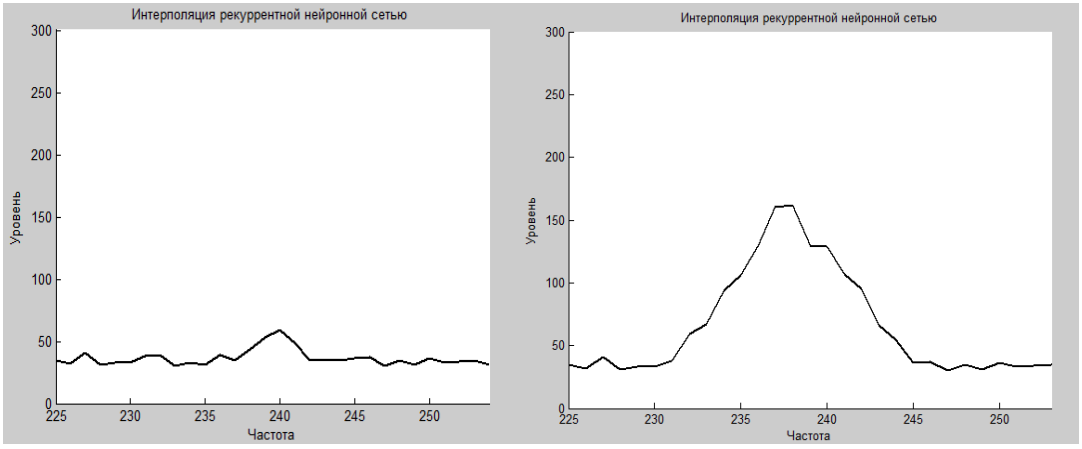


Рисунок 3.8 – Интерполяция рекуррентной нейронной сетью

Результат интерполяции получился очень похожим на результат интерполяции сплайнами. Если сравнить с аппаратной интерполяцией модулем SDRSharp, то данные 2 метода интерполяции ближе всего. Программная реализация осуществлялась виде отдельных функций на языке Python и которые затем импортировались в Delphi, на котором была написана основная программа, как методы объекта PPyObject. Для реализации кубических сплайнов использовались модули defaultdict из библиотеки collections и библиотеки bisect, pylab, math. Сплайны расчитывались функцией buildSpline(dots), в которую в качестве параметра подавались координаты экспериментальных точек.

Результат интерполяции сигнала сплайнами представлен на рисунке 3.9.

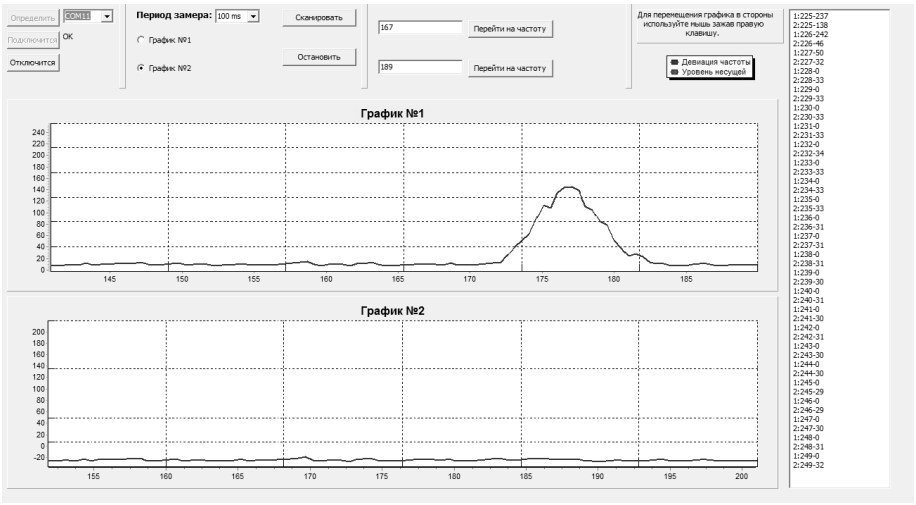


Рисунок 3.9 – Интерполяция демодулированного сигнала радиомикрофона кубическими сплайнами

Для реализации рекуррентной нейронной сети была использована библиотека PyTorch, в которой была создана описанная выше архитектура. Листинг программы представлен в приложении Б. Для обучения использовался сет наиболее характерных данных в количестве 2000 точек. Данные были разбиты на порции по 40 точек и выбирались случайным образом, после этого выборка перемешивалась. Выборка для обучения была разбита на тренировочную, тестовую и validation в соотношении 0.6:0.2:0.2. Обучение длилось около часа на на персональном компьютере с процессором Intel®, Pentium®, CPU G6300 2.7 GHz и

видеоадаптером NVIDIA GeForce CTX 1050 Ti. Видеоадаптер подключался средствами PyTorch. На рисунке 3.10 представлена визуализация процесса обучения.

Видно, что точность в процессе обучения достигает 0.95. Этот результат позволил в дальнейшем интерполировать получаемые данные. Благодаря рекуррентной структуре нейронной сети можно обрабатывать данные потоком. Для обработки берутся по 40 точек, так же, как и для обучения, затем производится интерполяция функцией true_interpolation.

```
[195] test loss: 10.462445378303528, accuracy: 0.7278
[196] train loss: 39.16947162151337, accuracy: 0.762
[196] test loss: 10.488057255744934, accuracy: 0.7222
[197] train loss: 39.196797251701355, accuracy: 0.7634
[197] test loss: 10.502906918525696, accuracy: 0.7222
[198] train loss: 39.395434617996216, accuracy: 0.7537
[198] test loss: 10.354896545410156, accuracy: 0.7472
[199] train loss: 39.331292152404785, accuracy: 0.7509
[199] test loss: 10.367400050163269, accuracy: 0.7389
print(meta_classifier.score(x_test_scaled.reshape(-1, 64), y_test))
>>> 0.95
```

Рисунок 3.10 – Процесс обучения рекуррентной нейронной сети

В процессе работы была изучена зависимость скорости обучения нейронной сети в зависимости от размера этой выборки (batch). Для этого брались различные значения batch и 200 точек входных данных, затем проводилось обучение с различными значениями batch и строился график. Результаты можно увидеть на рисунке 3.11. Оптимальным размером параметра batch были 40 входных значений. При этом скорость обучения составила 15 секунд. Надо сказать, что здесь прослеживается экспоненциальная зависимость скорости обучения от размера батча. Поэтому для обучения на всем сете данных, был выбран размер batch 40. Для нахождения оптимального значения количества нейронов в скрытом слое была исследована зависимость точности интерполяции от количества нейронов в скрытом слое. На рисунке 3.12 представлена эта зависимость. Из графика видно,

что после количества нейронов 25, точность не меняется. Следовательно, в модели можно уменьшить количество нейронов с 40 до 25 без потери точности интерполяции и с несомненным увеличением скорости обучения.

Если первоначальная модель обучалась около часа, то в итоге снижения количества нейронов время обучения уменьшилось до 15 минут. А после обрезания нейронов, у которых не менялись значения весов, методом, представленным в разделе 2, это время сократилось до 3 минут. Интерполяция производится по мере поступления данных из интерфейса программы, то есть только для отображаемых в данный момент точек.

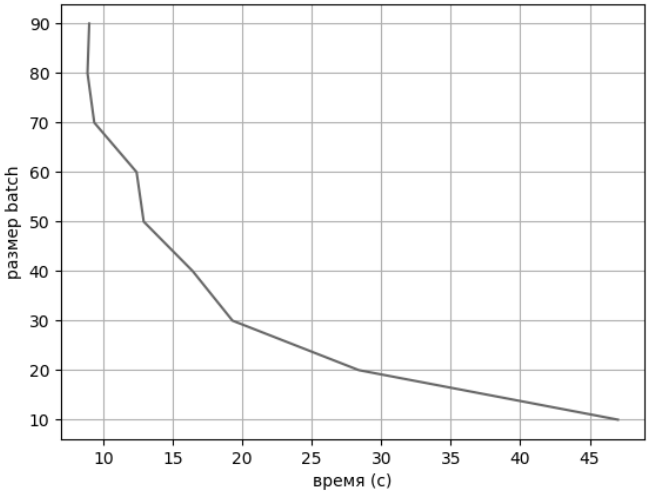


Рисунок 3.11 – Зависимость времени обучения от размера batch

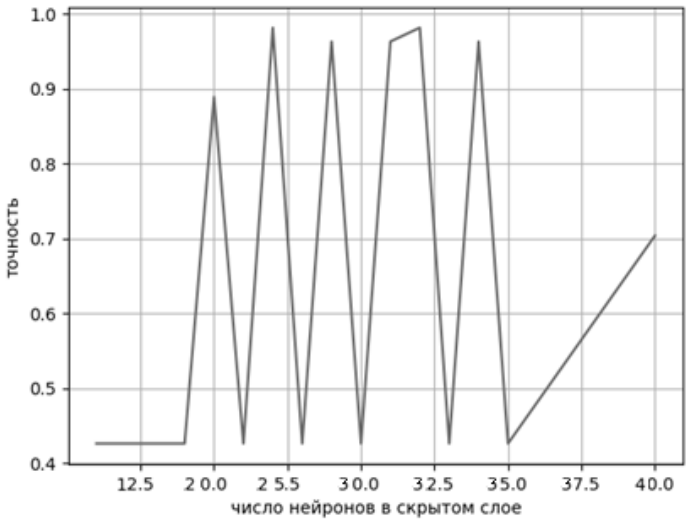


Рисунок 3.12 – Зависимость точности (accuracy) от количества нейронов в скрытом слое

Полученный график представлен на рисунке 3.13. По характеру кривой он заметно не отличается от интерполяции кубическими сплайнами. В процессе тестирования оказалось, что интерполяция необрезанной нейронной сетью происходит с задержкой приблизительно 3 секунды, в то время как при интерполяции сплайнами максимальная задержка 0,5 секунд. После применения обрезания проигравших нейронов задержка нейронной сети уменьшилась до 0,3 с.

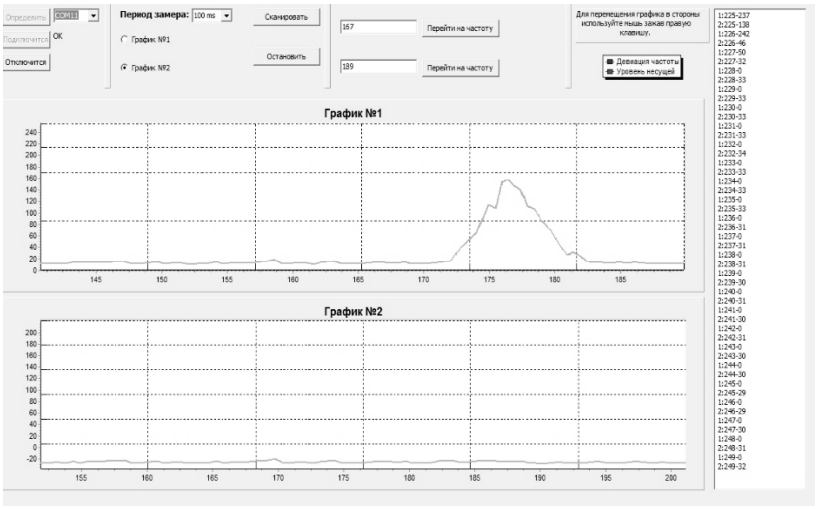


Рисунок 3.13 – Интерполяция демодулированного сигнала радиомикрофона рекуррентной нейронной сетью

В качестве окончательного решения была выбрана интерполяция обрезанной нейронной сетью. И если до оптимизации обрезанием нейронов программа интерполяции рекуррентной нейронной сетью занимала объем постоянной памяти около 1,32 Мб, (без сторонних библиотек), а реализация интерполяции сплайнами занимала объем 197 Кб, то после оптимизации программа интерполяции рекуррентной нейронной сетью заняла объем 192 Кб.

На рисунке 3.14 для сравнения представлен тот же сигнал на частоте 82 МГц, полученный в среде SDRsharp, использующей аппаратную интерполяцию.

При сравнении графиков видно, что полученные в результате интерполяции сплайнами и нейронной сетью кривые более плавные и по характеру идентичны интерполяции, полученной за счет значительной частоты дискретизации и разрядности АЦП.

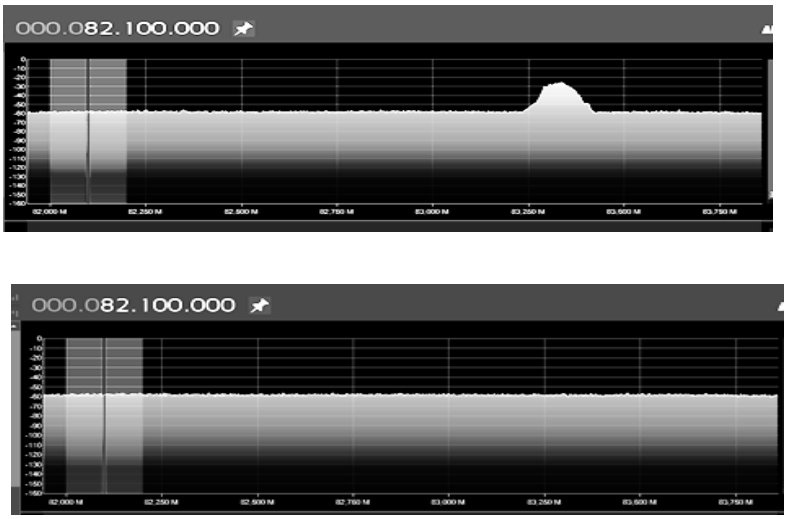


Рисунок 3.14 – Интерполяция в программе SRDsharp дает менее плавную форму кривых

Предложенная в разделе 2 методика дает возможность обрезать рекуррентную сеть таким образом, что она превосходит по затратам оперативной и постоянной памяти, а также скорости работы интерполяцию сплайнами.

3.2 АСНИ обработки изображений

Для экспериментального исследования принципов создания автоматизированных систем для обработки изображений и снижения их затратности были созданы несколько прототипов автоматизированных систем.

3.2.1 АСНИ распознавания формы предметов

В промышленном производстве с применением автоматизированных и роботизированных систем часто возникает проблема распознавания формы предметов. Например, на конвейере нужно различать некондиционные изделия, отличающиеся от кондиционных только формой. Или для сортировки мусора необходимо сепарировать предметы прямоугольной и округлой формы. Или в системах безопасности личности, например, определения на сцене наличия опасных предметов находить предметы, напоминающие по форме оружие. Для автоматизации процесса можно применять анализ коэффициентов формы. Предмет

фотографируется или снимается на камеру, после предварительной обработки изображения производится сегментация и расчёт коэффициента формы. Затем автоматизированная система принимает решение об отнесении предмета к тому или иному классу и действии, которое необходимо совершить с этим предметом, затем дает команду манипулятору произвести действие. Существующие коэффициенты формы представлены в таблице 3.3. Различие только в том, по какой формуле рассчитывается коэффициент.

Таблица 3.3 – Коэффициенты формы

№ п/п	Название коэффициента	Формула для расчета	Функции пакета Matlab, рассчитывающие коэффициенты
1	Коэффициент округлости	$k_{\text{окр}} = \frac{4\pi S}{P^2}$, где S – площадь объекта; P – периметр объекта.	bwboundaries выделение краев и затем 4*pi*area/ perimeter^2
2	Коэффициент заполнения	$k_{\text{зап}} = \frac{S}{hl}$ где h и l – размеры описанного вокруг объекта прямоугольника.	regionprops признак Extent
3	Эксцентриситет эллипса	$e = \sqrt{1 - (\frac{b}{a})^2}$, где b и a большая и малая полуось эллипса.	regionprops признак Eccentricity
4	Коэффициент компактности	$k_{\text{комп}} = \frac{S}{S_1}$, где S_1 – площадь выпуклого многоугольника, в который вписан объект	regionprops признак Solidity
5	Отношение периметров	$k_{\text{пер}} = \frac{P_{\text{объекта}}}{P_{\text{прямоуг}}}$, где $P_{\text{объекта}}$ – периметр объекта, $P_{\text{прямоуг}}$ – периметр описанного вокруг объекта прямоугольника.	bwboundaries выделение краев и затем boundarylength/perimeter

Для определения формы предметов «Круг», «Квадрат» и «Треугольник» использовались все коэффициенты, кроме эксцентриситета эллипса.

Распознавание формы предмета на изображении производилось при помощи нейронной сети типа LVQ с двумя скрытыми слоями: конкурирующим и линейным. Предмет должен был быть отнесен к трем основным классам – «Круг», «Квадрат» и «Треугольник».

Для обучения использовались специально созданные тестовые изображения, относящиеся к данным классам (Рисунок 3.15). Подсчитывалось процентное соотношение классов на изображения, для задания параметров обучения.

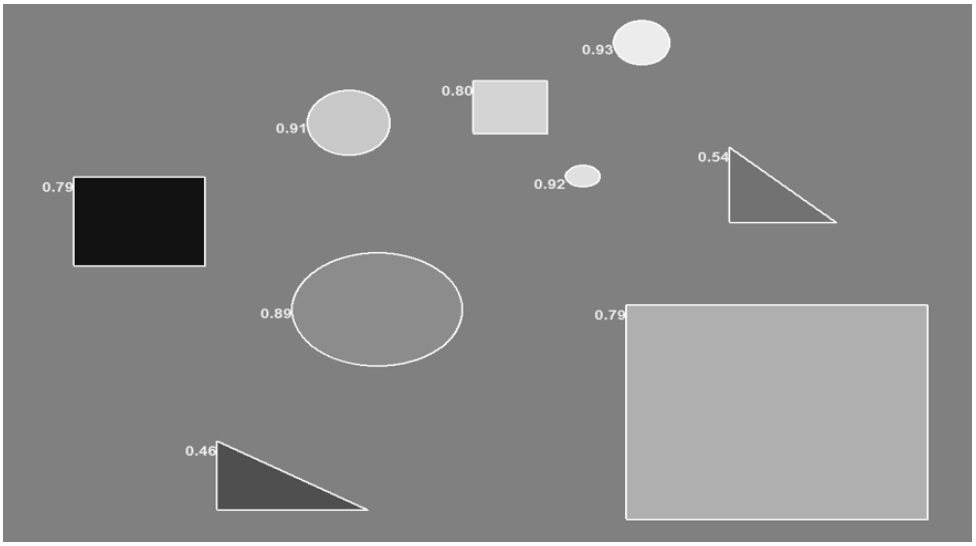


Рисунок 3.15 – Значения выделенных признаков выведены на тестовое изображение

Затем выделялись признаки либо функцией regionprops, либо bwboundaries для выделения границ, а затем расчет по формулам, представленным в таблице 3.3. Затем создавалась матрица признаков, которая и являлась входным вектором для нейронной сети. Выходным вектором являлась многомерная матрица из нулей и единиц. Единицы указывают на выбранные классы.

Для создания и обучения сети использовалась функция newlvq. Например в newlvq(minmax(x),4,[.4 .3 .3],0.1); Процентные доли входных векторов в каждом классе равны 0.4, 0.3 и 0.3 соответственно.

Выделены 4 кластера, (кластеров должно быть больше, чем классов, а классов всего 3, количество классов задано выходными данными. Параметр скорости настройки весов равен 0.1, он подбирался экспериментально.

Для визуализации с подписями была написана небольшая программа, использующая координаты центров масс выделенных объектов. На рисунке 3.16 видно, что для распознавания круга коэффициент округлости должен быть более 0,8, а треугольника – менее 0,6. В полностью автоматизированных системах, когда оператор не видит изображения для выбора объектов для обучения можно воспользоваться функцией [L num]=bwlabel(f,8); Процесс обучения длится секунды, что характерно для сетей с конкурирующим слоем.

Для проверки работоспособности сети было подготовлено несколько фотографий предметов различной формы, которые требовали предварительной обработки. Признаки вычисляются только у полутонового или бинарного изображения, поэтому вначале изображение из полноцветного преобразовывалось в полутоновое. Затем производилось улучшение методом эйквализации гистограммы (Рисунок 3.16).

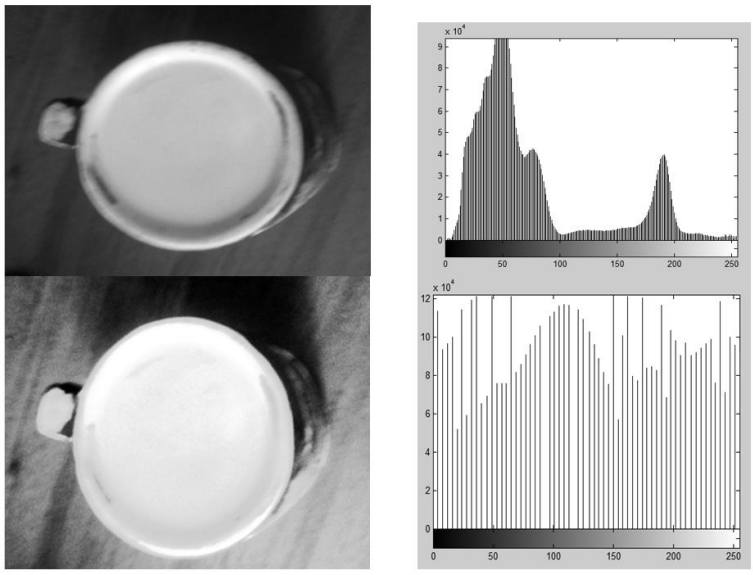


Рисунок 3.16 – Результаты улучшения изображения методом эйквализации гистограммы

Затем выделялись признаки и подавались на обученную нейронную сеть, которая определяла форму предмета (Рисунок 3.17).

В последующих экспериментах предварительно обученной нейронной сети предъявлялись изображения с несколькими предметами различной формы. В этом случае определялась форма только светлых предметов и, в некоторых случаях, окаймляющего квадрата. Изучив матрицу извлеченных признаков, можно прийти к выводу, что в данном случае функция regionprops не определяла признаки темных предметов, на которых находились светлые.

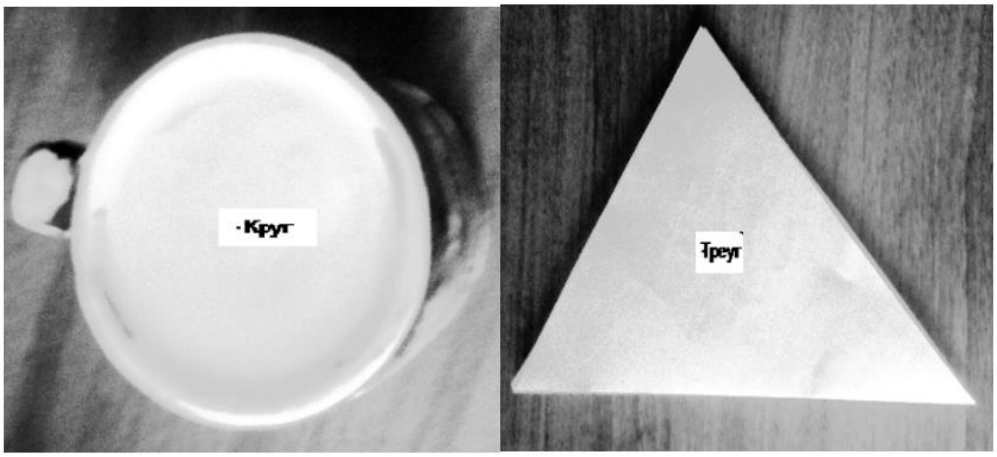


Рисунок 3.17 – Нейронная сеть LVQ правильно классифицирует форму предмета на фотографии

Таким образом при данном методе сегментации (наращивание регионов) признаки хорошо выделяются только из нескольких светлых предметов на темном фоне или одиночных светлых, или темных предметов на контрастном фоне, а при наложении друг на друга темных и светлых предметов извлечения признаков не происходит. Для более детального изучения распознавания формы предметов данной автоматизированной системой на вход подавались по 10 фотографий предметов различной формы, светлых на темном фоне, темных на светлом фоне и групп предметов светлой и темнойформы на нейтральном фоне. Фиксировался процент предметов, с правильно распознанной формой. Результаты экспериментов представлены в таблице 3.4. Под нейтральным фоном подразумевается серый, с неярко выраженной текстурой. Из проведенных экспериментов можно сделать

выводы, что нейронная сеть распознает LVQ форму предметов в 100 случаев, если правильно выделены признаки формы. Хотя функция regionprops может выделять сразу 3 признака, для 100% распознавания формы предмета оказалось достаточно одного. Для выделения признаков функцией bwboundaries результаты оказались аналогичными. Но в этом случае приходилось определять границы, а потом рассчитывать коэффициенты по формулам. При этом нет необходимости переучивать нейронную сеть, натренированную на одних признаках, определять форму по другим признакам.

Таблица 3.4 – Результаты распознавания предметов различной формы

№ п/п	Характеристика изображения	Процент распознавания формы предметов	Особенности распознавания формы предметов
1	Круглый светлый предмет на темном фоне	100	
2	Квадратный светлый предмет на темном фоне	100	
3	Треугольный светлый предмет на темном фоне	100	
4	Круглый темный предмет на светлом фоне	100	
5	Квадратный темный предмет на светлом фоне	100	
6	Треугольный темный предмет на светлом фоне	90	
7	Несколько круглых светлых предметы на фоне темных треугольных и квадратных	50	Распознавались только светлые изображения и окаймляющий квадрат
8	Несколько квадратных темных предметов на фоне круглых светлых	50	Распознавались только темные изображения и окаймляющий квадрат
9	Несколько светлых предметов разной формы на нейтральном фоне	100	
10	Несколько темных предметов разной формы на нейтральном фоне	90	Не распознавались в основном треугольные предметы
11	Несколько темных предметов разной формы на светлом фоне	100	
12	Несколько отдельных темных и светлых предметов разной формы на нейтральном фоне	80	Распознавались все светлые и некоторые темные предметы

Все неудачные случаи распознавания связаны с неправильной сегментацией и выделением признаков. Данную проблему можно решить, используя сверточные нейронные сети, однако они будут занимать гораздо больше памяти как постоянной, так и оперативной. В этом случае можно воспользоваться предложенным в разделе 2 методом снижения количества параметров. В этом случае, написанная на языке С, программа с предложенными методами сегментации 8-ми связным наращиванием регионов или выделением границ, выделением признаков по одному из 4-рех коэффициентам формы и двуслойной нейронной сети архитектуры LVQ, будет занимать приблизительно 20-30 Кбайт, что позволит использовать дешевые микроконтроллеры для промышленной реализации подобных систем.

3.2.2 АСНИ аутентификации по отпечаткам пальцев

Общий алгоритм аутентификации человека по отпечаткам пальцев представлен блок-схемой на рисунке 3.18 [58]. Причем каждый этап может быть реализован различными способами.

Цель проведенных экспериментов состоит в выборе наиболее подходящих алгоритмов для каждого этапа и создание собственного алгоритма для автоматизированной системы аутентификации по отпечаткам пальцев, с применением нейронных сетей на последнем этапе.

Рисунок 3.18 – Блок-схема общего алгоритма аутентификации

3.2.2.1 Получение изображения отпечатка

Современные компьютерные системы не позволяют получить графический файл отсканированного отпечатка пальца. Это связано с политикой безопасности операционной системы. Протокол обмена между ПК и сканером зашифрован, и поэтому вытащить оттуда изображение отпечатка не представляло возможности.

Из аппаратных средств имелись в наличии: сканер Microsoft 1033 и самодельный сканер отпечатков на основе непосредственно оптического сканера DY50_MAIN_V3, платы Arduino UNO с микроконтроллером Atmega329 (Рисунок

3.19). Оба сканера показали хорошее качество сканов отпечатков, однако Microsoft 1033 оказался более удобным в работе, поэтому все эксперименты проводились со сканами отпечатков, снятых при помощи данного сканера.

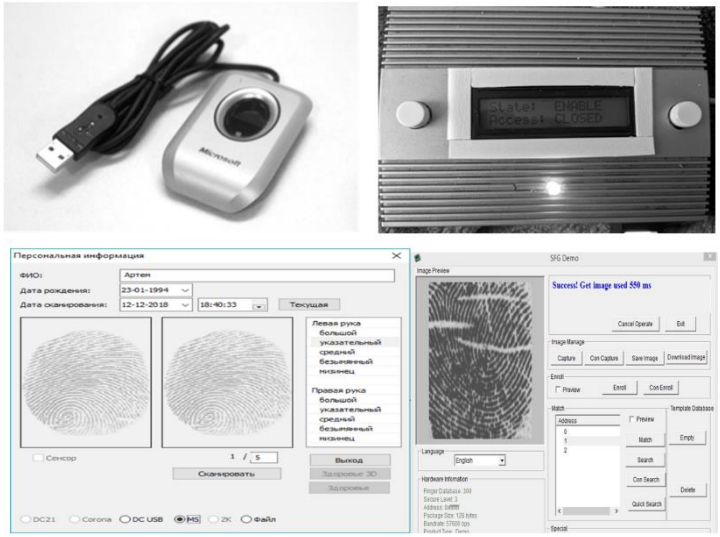


Рисунок 3.19 – Оптические сканеры отпечатков Microsoft 1033 и DY50_MAIN_V3 и их интерфейсы

3.2.2.2 Предварительная обработка

Со сканера на вход процедуры предварительной обработки поступает полноцветное изображение отпечатка, которое преобразовывается в полутоновое. Далее производится улучшение преобразованного изображения, применяется фильтрация, бинаризация и скелетизация, которые будут описаны в следующих подразделах. Подробный алгоритм процесса представлен на рисунке 3.20. Один из наиболее распространенных дефектов изображений со сканера– слабый контраст. Для изменения контрастности применялись различные методы улучшения изображения. Реализация алгоритмов осуществлялась в среде Matlab R2015a, с использованием библиотеки Image Processing [59].

Эйквализация гистограммы. Эйквализация гистограммы приводит ее к равномерному закону распределения.

Рисунок 3.20 – Блок-схема алгоритма предварительной обработки

Если имеется полутоновое изображение, с разрешением M×N пикселей. Количество уровней квантования яркости пикселей составляет J. На каждый уровень яркости, согласно формуле 3. 1, должно выпадать пикселей:

$$n_{aver} = \frac{N \cdot M}{J}. \tag{3.1}$$

Если x, y – случайные величины, описывающие изменение интенсивности пикселей на изображениях, $w_x(x)$ – плотность распределения интенсивности на исходном изображении, $w_y(y)$ – нормальная плотность распределения. Преобразование плотностей распределения $y = f(x)$, которое позволило бы получить необходимую плотность вычисляется по формуле (3.2) [31].

$$w_y(y) = \begin{cases} \frac{1}{y_{max} - y_{min}}, y_{min} \le y \le y_{max} \\ 0, \text{в противном случае} \end{cases}. \tag{3. 2}$$

Если $F_x(x)$ и $F_y(y)$ интегральные законы распределения случайных величин x и y, то из условия вероятностной эквивалентности следует, что $F_x(x) = F_y(y)$. Распишем интегральный закон распределения по формуле (3.3).

$$F_x(x) = F_y(y) = \int_{y_{min}}^{y} w_y(y)dy = \frac{y - y_{min}}{y_{max} - y_{min}}. \quad (3.3)$$

Получаем, что

$$y = (y_{max} - y_{min})F_x(x) + y_{min}\ . \quad (3.4)$$

Для оценки интегрального закона распределения $F_x(x)$, необходимо сначала построить гистограмму исходного изображения, затем нормализовать полученную гистограмму, разделив величину каждого уровня квантования на общее количество пикселей $N \cdot M$. В этом случае уровни квантования яркости можно рассматривать как приближенное значение функции плотности распределения $w_x^*(x), 0 \le x \le 255$.

Таким образом, значение интегральной функции распределения можно представить, как сумму следующего вида:

$$F_x^*(x) = \sum_{j=0}^{x} w_x^*(j). \quad (3.5)$$

Данную оценку можно использовать для вычисления новых значений интенсивности.

Результат выполнения метода выравнивание гистограммы над полутоновым изображением представлен на рисунке 3.21.

Контрастирование с гамма-коррекцией. Это применение гамма коррекции. Описывается гамма функция степенной функцией по формуле (3.6).

$$V_{out} = AV_{in}^{\gamma}, \quad (3.6)$$

где A служит коэффициентом, а входные V_{in} и выходные V_{out} значения – неотрицательные вещественные числа.

а) б)

Рисунок 3.21 – Изображение отпечатка, полученное с помощью выравнивания гистограммы: а) полутоновое изображение, б) улучшенное изображение

В общем случае, если A = 1, то входные и выходные значения находятся в пределах от 0 до 1 [31]. Результат обработки изображения методом контрастирования с гамма-коррекцией над полутоновым изображением представлен на рисунке 3.22.

а) б) в)

Рисунок 3.22 – Изображение отпечатка, полученное с помощью контрастирования с гамма-коррекцией: а) gamma < 1, б) gamma = 1, в) gamma > 1

Поиск границ повышения контраста на изображении. Значительного улучшения изображения можно добиться выделением краев с использованием

функции протяженности точек PSF. Эта функция описывает зависимость распределения освещенности от координат в плоскости изображения.

Если есть предмет I (x,y), то каждая его точка изображается в виде функции $h(x' - V_{x,y'} - V_y)$, т.е. PSF смещается в точку с координатами (V_x, V_y), а изображение всего предмета будет представлять собой сумму этих изображений, которое описано в формуле (3. 7).

$$I'''(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(x,y)h(x' - V_x, y' - V_y)dxdy. \quad (3.\ 7)$$

Если V принять за единицу, то данное выражение преобразуется в свертку, представленную в формуле (3. 8).

$$I'''(x,y) = I(x,y)h(x,y''). \quad (3.\ 8)$$

Функция изображения является сверткой функции предмета с функцией рассеяния точки [31]. Рассмотренная функция протяженности точек описывается функцией edgetaper. Эта функция размывает контуры исходного изображения I, используя функцию протяженности точки PSF. Результирующее изображение J представляет собой взвешенную сумму исходного изображения и его размытой версии [31].

Результат выполнения поиска границ повышения контраста на изображении представлен на рисунке 3.23.

Контрастно-ограниченная адаптивная эйквализация гистограммы. Изображение разбивается на прямоугольные блоки. В каждом блоке вычисляется гистограмма распределения яркости.

Пусть M – число строк в блоке, N – число столбцов в блоке, α – параметр, задающий ограничение. Предельное значение гистограммы вычисляется в соответствии с формулой (3. 9).

а) б)

Рисунок 3.23 – Изображение отпечатка, полученное с помощью поиска границ повышения контраста: а) полутоновое изображение, б) улучшенное изображение

$$hist_{lim} = hist_{min} + \alpha(MN - hist_{min}), \quad (3.9)$$

где $hist_{min} = \frac{MN}{L+1}$, L – максимальное значение яркости.

После ограничения для каждой области формируется своя LUT (функция преобразования яркости) на основании эквализации гистограммы распределения яркости. Для входного изображения вычисляется значение яркостей выходного изображения в соответствии с формулой (3. 10).

$$LUT(k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n}, \quad (3.\ 10)$$

где LUT(k) – значение яркости выходного изображения, соответствующее яркости r_k входного изображения, $k \in [0, L]$; r_j – яркость элемента входного изображения, n_j – число элементов входного изображения со значением r_j, n – общее число элементов изображения; $p_r(r_j)$ – вероятность появления элемента входного изображения с яркостью r_j [31].

Результат выполнения контрастно-ограниченной адаптивной эквализации гистограммы представлен на рисунке 3.24. Такое преобразование позволяет равномернее распределить уровни яркости по частоте, в основе преобразования

лежит предположение, что в контрастных черно – белых изображениях значения яркости распределены равномерно.

а) б)

Рисунок 3.24 – Изображение отпечатка, полученное с помощью, контрастно-ограниченной адаптивной эквализации гистограммы: а) полутоновое изображение; б) улучшенное изображение

По результатам экспериментов был выбран метод улучшения на основе вычисления уровня контраста изображения, результаты которого приведены в подразделе 3.2.2.6 (Таблица 3.6), а также равномерности распределения контраста по всему образу отпечатка пальца.

3.2.2.3. Бинаризация и скелетизация изображения

Для выбора метода бинаризации были рассмотрены следующие методы: *Бинаризация отсечением по порогу яркости.* Метод реализовывался согласно:

$$f'(m,n) = \begin{cases} 0, f(m,n) \leq t; \\ 1, f(m,n) > t, \end{cases} \quad (3.\ 11)$$

где $f(m,n)$ – яркость пикселя на исходном изображении, $f(m,n) \in [0.2^k - 1]$, $f'(m,n)$ – значение пикселя результирующего изображения, $f'(m,n) \in$

$[0,1], t$ – порог бинаризаци. Любая точка изображения, для которой выполняется условие $f(m,n) > t$, называется точкой объекта, а в противном случае – точкой фона [60].

Результат бинаризации методом отсечения по порогу яркости представлен на рисунке 3.25.

а) б) в) г)

Рисунок 3.25 – Бинаризация отпечатка отсечением по порогу яркости: а) по порогу 0,3, б) по порогу 0,5, в) по порогу 0,7, г) по порогу 0,9

Метод Отцу. Вычисляется порог *t*, минимизирующий среднюю ошибку от принятия решения о принадлежности пикселей изображения объекту или фону. Значения яркостей пикселей изображения рассматриваются как случайные величины, а их гистограмма – как оценка плотности распределения вероятностей. Если плотности распределения вероятностей известны, то можно определить оптимальный порог для сегментации изображения на два класса c_0 и c_1 (объекты и фон). Исследования основывались на следующем: изображение представляется с помощью L уровней яркости; hi – число элементов изображения, имеющих яркость i, i = 0, 1, ..., L-1; H – общее число пикселей на изображении; гистограмма изображения является нормализованной и ее можно рассматривать как распределение вероятностей, согласно формуле (3.12).

$$p_i = \frac{h_i}{H}, i = 0,1, \dots L-1; \sum_{i=0}^{L-1} p_i = 1\,. \qquad (3.12)$$

Элементы изображения делятся на два класса c_0 и c_1 с помощью порогового значения t, где класс c_0 содержит пиксели с яркостями из множества (0, 1, ..., t), а класс c_1 – пиксели с яркостями из множества (t, t+1, ..., L – 1). Вероятности каждого из этих двух классов и средние значения их яркости описываются выражениями, согласно формулам (3.13) [60].

$$P_0 = \sum_{i=0}^{t} p_i = P_t, P_1 = \sum_{i=t+1}^{L-1} p_i = 1, \mu_0 = \sum \frac{ip_i}{P_0} = \frac{\mu_t}{P_t};$$

$$\mu_1 = \sum_{i=t+1}^{L-1} \frac{ip_i}{P_1} = \frac{\mu_T - \mu_t}{1 - P_t}, \quad (3.13)$$

где $\mu_T = \sum_{i=0}^{L-1} ip_i$ означает среднюю яркость всего изображения.

Результат бинаризации методом Отцу представлен на рисунке 3.27.

Метод Ниблейка. В данном методе для каждого пикселя изображения используется свое значение порога. Величина порога определяется на основе вычисления локального среднего и локального среднеквадратического отклонения. Значение порога в точке с координатами (m, n) вычисляется в соответствии с формулой (3. 14).

$$t(m,n) = \mu(m,n) + k \cdot \sigma(m,n), \quad (3.14)$$

где $\mu(m,n)$ – среднее, а $\sigma(m,n)$ – среднеквадратичное отклонение в локальной окрестности точки изображения (m, n) [60]. Результат бинаризации методом Ниблейка представлен на рисунке 3.26.

Итерационный алгоритм бинаризации с обработкой краев изображения. Данный метод был разработан автором, как альтернатива методу Отцу [61].

Математической основой данного алгоритма является метод итерационного приближения к порогу бинаризации, характерный для численных методов. Задается начальное значение для итераций, как половина суммы всех яркостей изображения. Суммируются яркости и считается количество пикселей, имеющих яркость выше и ниже этого значения. Для каждой суммы находится половина и устанавливается новое значение порога, как среднее из этих двух значений.

Затем итерации повторяются пока разница между верхним средним значением и нижним не достигнет нуля. Затем полученное значение нормализуется по всему количеству пикселей изображения и получается порог бинаризации.

Рисунок 3.26 – Бинаризация отпечатка методами Отцу, Ниблейка и итерационным алгоритмом с обработкой краев изображения

На рисунке 3.27 представлена блок-схема разработанного алгоритма бинаризации. Программная реализация представлена в приложении В. Результат бинаризации разработанным алгоритмом представлен на рисунке 3.27.

Метод отсечения по порогу яркости показал не лучший результат, из-за того, что он не может рассчитать реальный порог изображения, а бинаризует его вслепую. Метод Ниблейка дает сильный краевой эффект и заниженный порог бинаризации, из-за которого изображение значительно затемнено, края отпечатка не являются четкими. Итерационный алгоритм не хуже алгоритма Отсу нашел порог бинаризации, изображение получилось даже более четким, значит глобальный порог бинаризации рассчитан точнее.

Окончательный выбор метода бинаризации осуществлялся по результатам расчета интенсивности пикселей, представленного в подразделе 3.2.2.6 (Таблица 3.7).

Скелетизация – это выделение всех точек бинарного изображения с анализом окрестности пикселя. Были исследованы такие морфологические методы как, построение остова, эрозия и дилатация для получения скелета изображения [62].

Построение остова. Эта морфологическая операция принимает значение *skel*, выполняющая эрозию объекта с учетом ряда условий для сохранения 8-связности остова.

Рисунок 3.27 – Блок-схема итерационного алгоритма пороговой бинаризации

В итоге последовательного применения данной операции можно построить остов, представляющий собой связную линию толщиной в 1 пиксель, проходящую по середине объекта.

Метод утоньшения объекта основан на простом анализе окрестности каждой из его точек. Image Processing выполняет эрозию объекта с сохранением 8-

связности участков остова. Результат применения операции утончение показан на рисунке 3.28.

Рисунок 3.28 – Скелетизация построеним остова изображения, утончением и дилатацией изображения

При утолщении объекта выполняется операция дилатации, которая утолщает объект с учетом 4-связности участков фона. Размер структурного элемента, равен 3×3 пикселей. Анализ окрестности выполняется согласно формуле (3. 15).

$$K = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x + j, y + i) \cdot h(i, j), \qquad (3.\ 15)$$

где K – величина оценки окрестности, используемая как индекс; (x, y) – координаты исследуемого пикселя; f(x, y) – значение пикселя на бинарном изображении; h(i, j) – маска, элементы которой задают вес точек [63].

Выбор морфологической операции для построения скелета основывался на внешнем анализе полученного отпечатка. Результат анализа представлен в подразделе 3.2.2.6.

3.2.2.4. Выделение особых точек

Бинаризация в сочетании со скелетизацией обеспечивает простую основу для выделения особых точек. Для нахождения этих точек, применялись метод Харриса и метод выделения по локальным признакам.

Метод Харриса Поиск особых точек осуществляется с помощью нахождения углов на изображении. Они формируются из двух или более граней, и грани обычно определяют границу между различными объектами и / или частями одного и того же объекта.

В зависимости от количества пересекаемых граней существуют разные виды уголков: L-, Y- (или T-), и X- связные, которые представлены на рисунке 3.29.

Рисунок 3.29 – Виды угловых детекторов

В данной работе используется L-связный угловой детектор Харриса. Этот детектор рассматривает изображение как окно W (размер окна зависит от размера самого изображения) с точкой в центре ⟨x,y⟩, а также его сдвига на ⟨u,v⟩, который изображен на рисунке 3.30 [63, 64].

Тогда взвешенная сумма квадрата разностей между окном W и окном, сдвинутым на (u,v) (см. формулу (3.16)) равна [63]:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2, \qquad (3.16)$$

где I(x,y) – яркость в точке ⟨x,y⟩ окна W, w(x,y) – весовая функция окна.

Далее вычисляется матрица вторых моментов M по окну вокруг каждого пикселя согласно формуле (3. 17) [63].

$$M = \sum_{(x,y)} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \quad . \qquad (3.\ 17)$$

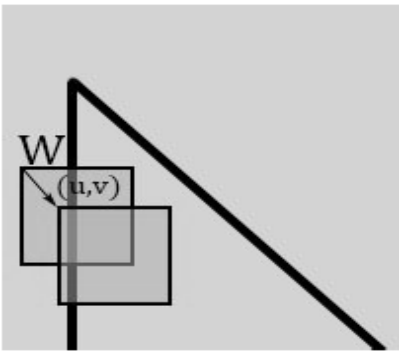


Рисунок 3.30 – Используемое окно

Обозначим λ_1 и λ_2 собственные значения матрицы M для разных точек изображения (Рисунок 3.31), тогда величина E(x, y) будет пропорциональна каждому из них. При этом будет реализовываться один из следующих случаев:

Если $\lambda_1 \approx 0$ и $\lambda_2 \approx 0$, то рассматриваемый пиксель находится в области равномерной интенсивности и не имеет локальной точечной особенности.

Если $\lambda_1 \approx 0$, а λ_2 большое положительное число, то интенсивность изображения имеет перепад вдоль одного направления, что свидетельствует об обнаружении края.

Если λ_1 и λ_2 являются большими положительными числами, то обнаружена точечная особенность типа угол.

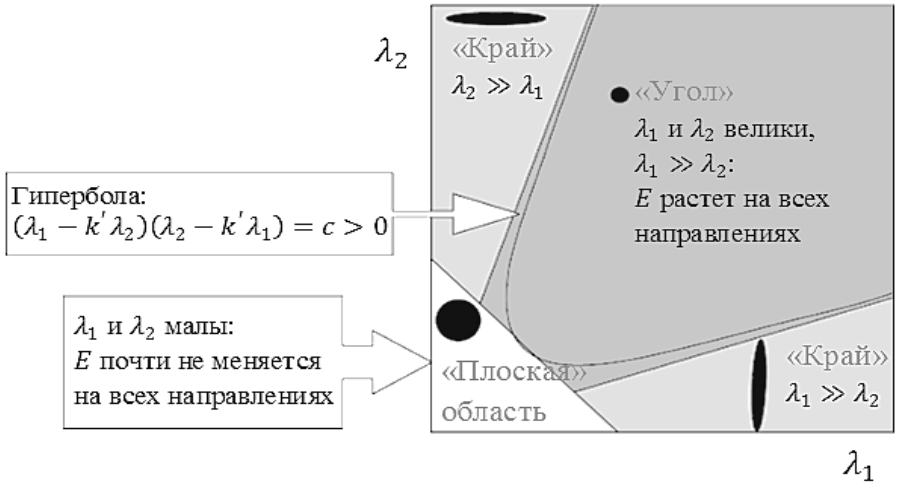


Рисунок 3.31 – Расположение собственных значений

Вместо непосредственного подсчёта значений λ_1 и λ_2 можно вычислить функцию отклика по формуле (3.18).

$$R = detM - k(trM)^2, \quad (3.18)$$

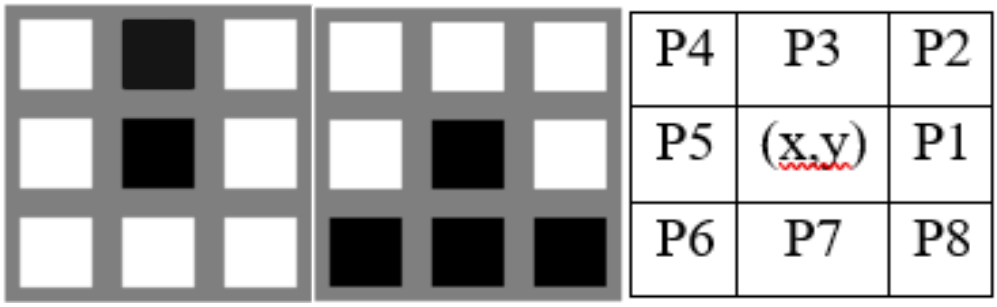
где $detM = \lambda_1\lambda_2, trM = \lambda_1 + \lambda_2$; λ_1 и λ_2 – собственные значения матрицы M; k – эмпирически подбираемый параметр со значениями порядка 0,04-0,06 [64]. Затем нужно рассмотреть те точки, в которых мера силы угла больше некоторого порога $R \geq 0$ получаем угол, при отрицательной мере получаем край, при нулевой – «плоскую» область. Далее находятся локальные максимумы функции отклика в окрестности заданного радиуса и выбираются в качестве особых точек.

Метод выделения по локальным признакам. Для выделения особых точек изображение разбивается на блоки 3x3 пикселей и к 8-ми соседним точкам применяется формула (3.19).

$$cn(p) = \frac{1}{2}\sum_{i=1\ldots8}|val(p_i) - val(p_{i-1})|. \quad (3.19)$$

Здесь p – значение для каждого пикселя, находящегося рядом с пикселем центрального блока, $val \in \{0,1\}$ – бинарное значение интенсивности пикселя.

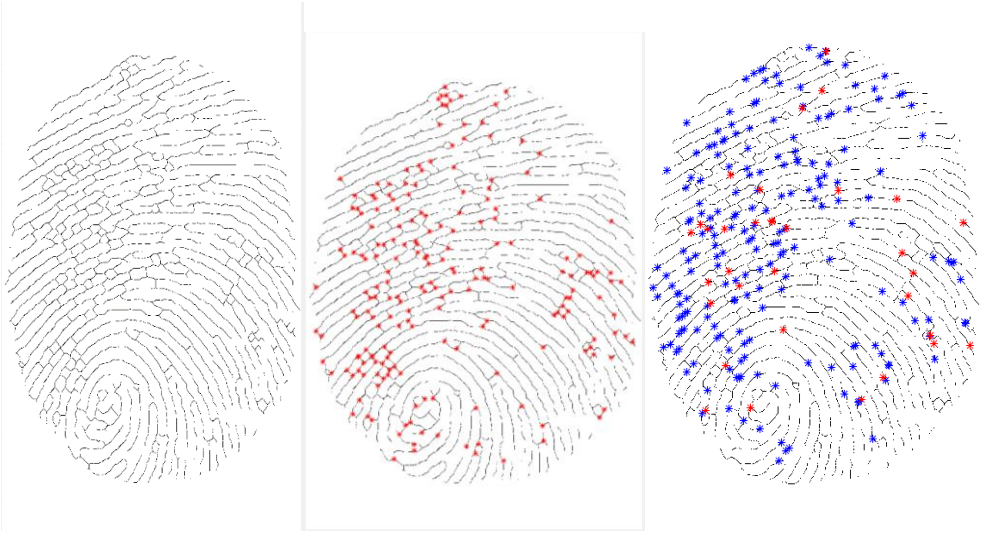
После этого подсчитывается число чёрных (ненулевых) пикселей, находящихся вокруг центра. Пиксель в центре считается минуцией, если он сам ненулевой и соседний с ним пиксел один (минуция «окончание»), если три (минуция «ветвление»). Координаты обнаруженных минуций изображения будут представлять собой уникальный набор значений, принадлежащий конкретному отпечатку. На рис. 3.32 показан пример соответствующих минуций [65].



а) б) в)

Рисунок 3.32 – 8-связная область смежных пикселей: a) окончание, b) ветвление, c) блоки 3x3

Результаты выделения особых точек предложенными методами представлен на рисунке 3.33.



а) б) в)

Рисунок 3.33 – а) скелет изображения, б) выделенные особые точки по Харрису, в) выделенные особые точки по локальным признакам

3.2.2.5. Алгоритмы распознавания

В окрестности каждой контрольной точки производится сравнение ближайших точек, если точка располагается на допустимом удалении, то эти точки считаются совпавшими. Смещение контрольных точек может достигать до 10 % от длины рамки изображения отпечатка пальца. Распознавание выполняется по следующим алгоритмам.

Сравнение по заданному порогу. Решение об идентификации принимается по расчету матриц изображений идентифицирующего и эталонного отпечатков, представляющих координаты контрольных точек. Если разница значений будет меньше заданного порога, то отпечатки считаются совпавшими, в противном случае будет не совпадение.

Расчет матриц двух изображений отпечатков вычисляется по формуле 3.20.

$$S(X,Y) = \left(\left|X_{i,j} - Y_{i,j}\right|\right) < \text{заданного порога}, \qquad (3.20)$$

где X – матрица идентифицируемого изображения отпечатка, Y – матрица эталонного изображения отпечатка.

Распознавание нейронной сетью. В качестве усовершенствования алгоритма распознавания внедрялись нейронные сети, было опробовано несколько архитектур [66]. Для экспериментов были отобраны девять архитектур нейронных сетей с различными параметрами. Это, три сети типа FF (однонаправленная сеть, обучаемая с применением алгоритма обратного распространения с заданным количеством итераций 1000, и целевой ошибкой 0,001) с разным количеством скрытых слоев и нейронов, RBF1 - радиально-базисная сеть с нулевой ошибкой и RBF2,3 – радиально-базисные сети с ненулевой ошибкой и разным значением параметра ширины активационной функции spread – 0,2, 0,7, три сети типа GRNN (обобщенная регрессионная нейронная сеть), с разными значениями параметра spread – 0,2, 0,7, 1.

Параметры сетей были подобраны в ходе экспериментов. Сети обучались на векторах из 200 признаков, выделенных из отпечатка пальца одного человека, представленных в виде матрицы 200x2. На обученную сеть подавались признаки, извлеченные из другого отпечатка того же человека и признаки, извлеченные из отпечатка другого человека. Те признаки, разница между которыми не превышала 30 пикселей считались распознанными правильно.

Результаты распознавания отпечатков пальцев нейронными сетями различной архитектуры представлены в таблице 3.5.

Полученные результаты сравнивались с результатами распознавания алгоритмом простого сравнения векторов признаков двух изображений. Пороговое значение 30 было найдено экспериментально, и соответствовало заданному критерию для алгоритма простого сравнения выделенных признаков.

Кроме того, выходные вектора признаков на которых проводилось обучение и полученные в процессе распознавания выводилась на график для визуального контроля. Количество правильно распознанных признаков отпечатков одного человека алгоритмом простого сравнения – 95,7 %, количество распознанных признаков отпечатков разных людей – 0,005.

Таблица 3.5 – Результаты распознавания отпечатков пальцев нейронными сетями различных архитектур

№	Тип нейронной сети	Количество правильно распознанных признаков 2-х отпечатков одного человека, %	Количество распознанных признаков 2-х отпечатков разных людей, %
1	FF1	21,7	0,010
2	FF2	47,5	0,010
3	FF3	16,4	0,020
4	RBF1	99,5	0,000
5	RBF2	90,1	0,010
6	RBF3	79,4	0,010
7	GRNN1	89,5	0,005
8	GRNN2	77,5	0,005
9	GRNN3	68,5	0,010

Лучшие результаты показала только одна нейронная сеть – RBF1 (радиально-базисная сеть с нулевой ошибкой). Таким образом, в ходе работы проведен сравнительный анализ результатов распознавания отпечатков пальцев нейронными сетями типов RBF, GRNN, FF с различными параметрами. Показано, что использование алгоритма, основанного на применении нейронной сети RBF с нулевой ошибкой, дает возможность эффективного сравнения признаков и классификации отпечатков пальцев.

3.2.2.6. Окончательный выбор алгоритма

Исходя из преобразованного изображения, полученного в результате осуществления различных методов улучшения, был подсчитан уровень контраста преобразованного изображения. Для расчета использовалась среднеквадратическая контрастность, определяемая как стандартное отклонение яркости пикселя

изображения I, от средней яркости B растрового изображения размерами M × N, которая вычислялась по следующим формулам (3.21, 3.22) [31].

$$\text{Contrast} = \sqrt{\frac{1}{\text{NMP}}\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{p}[[I(i,j,k) - B]^2]}, \quad (3.21)$$

$$\text{где } B = \frac{1}{\text{NMP}}\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{p}[I(i,j,k)], \quad (3.22)$$

где M – число строк на изображении, N – число столбцов на изображении, I – улучшенное изображение.

В таблице 3.6 представлены результаты вычислений. По результатам расчетов видно, что наибольший уровень контраста дает метод контрастно-ограниченной адаптивной эквализацией гистограммы. По результатам бинаризации проводился расчет интенсивности белых и черных пикселей изображения, полученного различными методами бинаризации. Результаты представлены в таблицах 3.6, 3.7.

Таблица 3.6 – Результаты расчетов уровня контраста

Методы улучшения изображения	контрастность
Выравнивание гистограммы	10,8071
Контрастирование с гамма-коррекцией	11,7923
Выделение краев с использованием функции протяженности точек	10,3203
Контрастно-ограниченная адаптивная эквализация гистограммы	11,8002

Таблица 3.7 – Результаты расчетов интенсивности пикселей

Интенсивность пикселей	Методы бинаризации						
	Отсечение по порогу яркости				Метод Отцу	Метод Ниблейка	Итерац. метод
	0,3	0,5	0,7	0,9			
белых	252293	535284	822552	116614	725859	377139	720338
черных	1819817	1536826	1249558	905962	1346251	1694971	1351772

Практическая значимость проведенного расчета заключается в нахождении метода бинаризации, оптимального для используемых изображений отпечатка пальца. Интенсивность белых и черных пикселей итерационного метода практически не отличается от интенсивности метода Отцу, а по сравнению с другими оставшимися методами, будут являться средними значениями. Поэтому результат разработанного итерационного метода бинаризации с обработкой краев изображения, будет характеризоваться получением более качественного папиллярного узора отпечатка пальца.

На этапе скелетизации были рассмотрены три морфологические операции: построение остова объекта, утончение и утолщение объекта. Выбиралась та операция морфологической обработки, которая подчеркивала бы лучше остальных геометрические и топологические свойства рассматриваемого изображения. Как при построении остова, так и при утончении линий объекта, в итоге сами линии имеют небольшой разрыв между собой. Появляются при этом дополнительные линии, никак не связанные с основными линиями отпечатка.

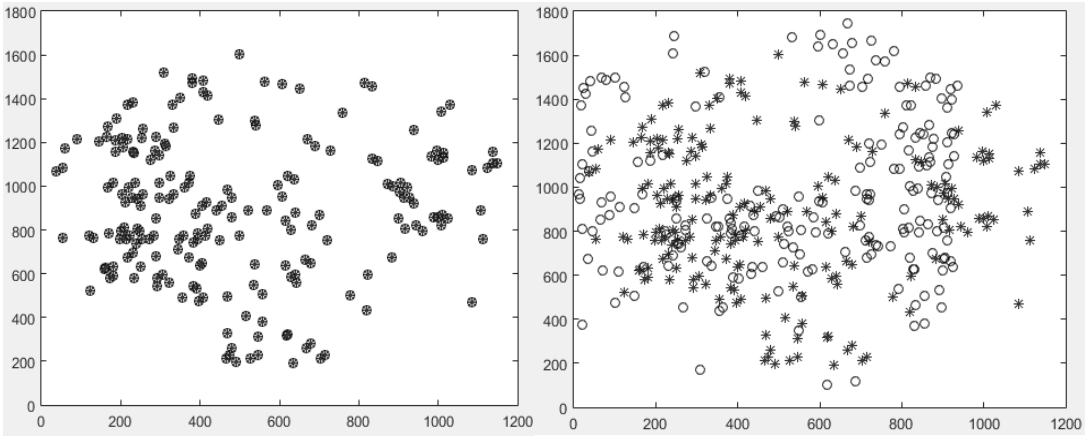
Морфологическая операция сохранила связность и структуру папиллярных линий, тем самым упростив задачу выделения особых точек.

Перед тем как реализовывать метод Харриса, проводилась качественная предварительная обработка. Преобразованное изображение к скелету, привело к появлению ярко-выраженных углов. Благодаря этому, большинство углов, расположенных на отпечатке, были хорошо заметны для алгоритма. Это в значительной степени способствовало нахождению особых точек на изображении отпечатка пальца. Нахождение особых точек по локальным признакам (окончание и разветвление), во многом зависело от качества скелетезации. Недостаточная скелетизация, как правило, приводила к получению лишних линий, однако благодаря морфологической операции утолщение объекта, решило данную проблему. Изучив результаты работу исследуемых алгоритмов выделения особых точек, а также применив их на практике, можно выделить основные достоинства и недостатки, которые представлены в таблице 3.8.

Таблица 3.8 – Достоинства и недостатки методов выделения признаков

Методы выделения	Достоинства	Недостатки
Харриса	Быстродействие, не зависит от качества скелетизации	Чувствительность к шуму, зависимость от масштаба изображения
По локальным признакам	Выделяет не один, а несколько признаков	Зависимость от качества скелетизации, возможность получения ложных минуций, медленный в работе

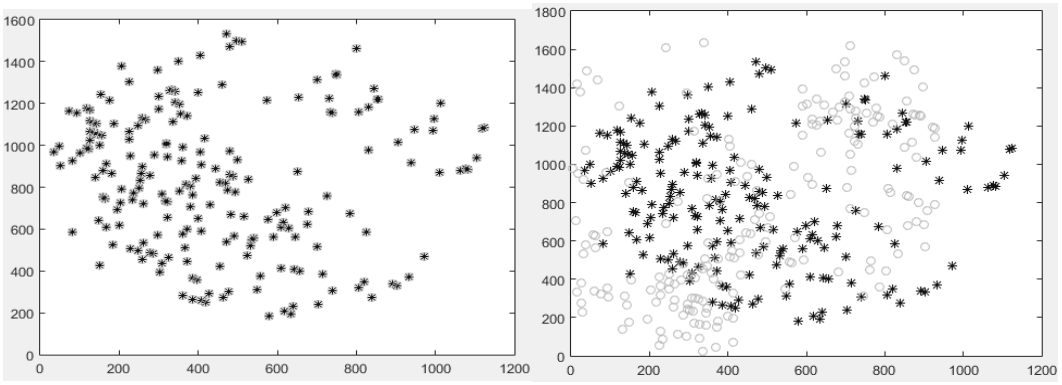
Результаты сравнения особых точек двух отпечатков, найденные детектором Харриса, отображаются на графике (Рисунок 3.34) в виде специально заданных фигур: звездочки характеризуют особые точки эталонного отпечатка, окружности – идентифицирующего. Если произошло соединение фигур это означает, что данные точки совпали, и наоборот, если не соединились, а расположены рядом друг с другом, то это значит, что нет совпадения.



а) б)

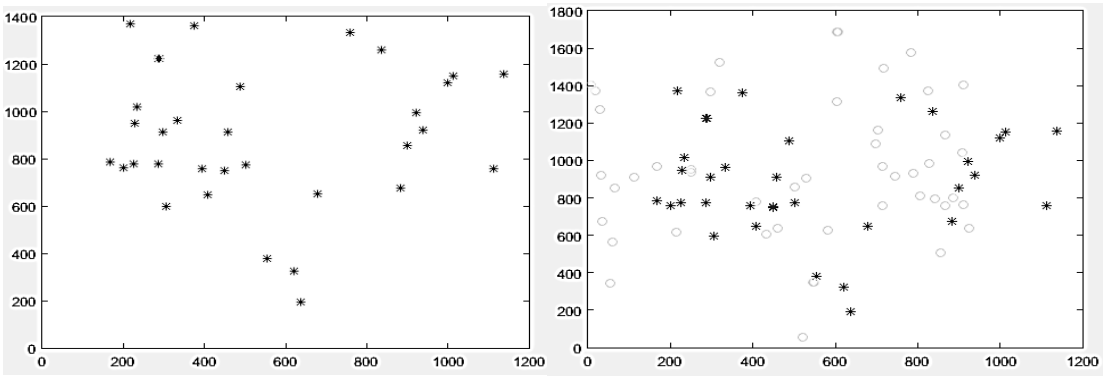
Рисунок 3.34 – Сравнение найденных особых точек детектором Харриса: а) одного и того же человека, б) разных людей

Результат сравнения особых точек двух отпечатков, найденных по локальным признакам представлен на рисунках 3.35 и 3.36



а) б)

Рисунок 3.35 – Сравнение ветвлений: а) одного и того же человека, б) разных людей



а) б)

Рисунок 3.36 – Сравнение окончаний: а) одного и того же человека, б) разных людей

По результатам выполнения программы аутентификации по отпечаткам пальцев были выбраны следующие методы предварительной обработки: контрастно-ограниченная адаптивная эквализация гистограммы, итеративный алгоритм бинаризации с обработкой краев изображения, скелетизация методом дилатации.

Выделение особых точек проводилось по методу Харриса и по локальным признакам, при этом оба показали удовлетворительные результаты и имеют свои достоинства и недостатки, которые должны учитываться при разработке системы аутентификации. Например, при наличии высокого уровня шума, для изображений, получаемых со сканера, нужно применять метод выделения по локальным

признакам, в том случае если имеет большое значение быстродействие системы – метод Харриса. Сравнение двух отпечатков проводилось по заданному порогу и при помощи нейронной сети RBF, оба показали удовлетворительные результаты по качеству и по затратности.

3.2.3 АСНИ биометрической аутентификации по лицу

Процесс аутентификации по лицу можно разделить на две отдельные задачи: 1) поиск лица человека на кадре; 2) распознавание найденного изображения нейронной сетью.

Существует большое количество методов поиска лица на изображении [67]. В качестве способа поиска расположения лица на фотографии для автоматизированной системы аутентификации был выбран метод Виолы-Джонса, использующий «признаки Хаара» [68]. А далее найденное лицо (а если оно не одно, то каждое по очереди) подаётся на нейронную сеть. Поскольку каждый пиксель изображения необходимо подавать на один нейрон входного слоя нейронной сети, а в случае даже выделенного изображения лица, это 10 000 нейронов, то в системах распознавания лиц используются сверточные нейронные сети. Изображение, пропускается через серию свёрточных, нелинейных слоев, слоев объединения и полносвязных слоёв, и генерируется вывод. Выводом может быть класс или вероятность классов, которые лучше всего описывают изображение. Однако, такой подход затрудняет добавление новых лиц для аутентификации. Ведь если добавлять новое лицо сеть нужно заново переучивать. И чем больше лиц в итоге, тем дольше длится новое обучение. Чтобы это избежать нужно сделать нейронную сеть, которая будет просто сравнивать лица на двух поступающих на вход изображениях, а на выходе будет давать их «похожесть» в процентах. Для исследований разработано приложение для распознавания лиц, которое, осуществляет распознавание лиц студентов и преподавателей кафедры радиофизики и инфокоммуникационных технологий.

3.2.3.1. Поиск лица человека на изображении

Для поиска лица на изображении рассматривались 2 метода: цветовая сегментация и Виолы-Джонса. На рисунке 3.37 представлен результат выделения лица методом цветовой сегментации.

Рисунок 3.37 – Выделение лица, методом цветовой сегментации

Видно, что лицо выделяется, но часто вместе с другими частями тела, если они находятся близко к лицу. Метод Виолы-Джонса дал гораздо лучшие результаты. Изображения для обработки этим методом сначала преобразуются в интегральном представлении, то есть учитываются яркости соседних пикселей, находящихся левее и выше данного элемента. Затем вычисляются признаки Хоара. В работе применялся расширенный метод Виолы – Джонса, использующий дополнительные признаки Хоара. Это незначительно удлиняет время обнаружения лица на изображении, но значительно повышает качество поиска. В процессе поиска признаков применяется бустинг (от англ. boost – улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на данной части изображения. Выбирается размер ячейки 8x8 пикселей, окно сканирования 24x24 ячейки, которое начинает последовательно двигаться по изображению, с шагом в 1 ячейку. В каждом окне вычисляется приблизительно 200 000 вариантов расположения признаков Хоара, за счет изменения масштаба признаков и их положения в окне сканирования, сканирование производится последовательно для различных масштабов окна (изменяется размер ячейки), затем признаки поступают

на вход классификатора, который определяет относится ли фрагмент изображения к области лица, для быстрого отбрасывания окон, где не найдено лицо, применяются каскады признаков. После того как на изображении найдено человеческое лицо и получены его координаты, область с лицом вырезается, приводится к формату 100x100 пикселей и далее подаётся на нейронную сеть [68].

Для получения изображения использовались видеокамеры, установленные на двух компьютерах: ПК Acer Aspire TC-780 (DT.B8DME.007) с веб-камерой Logitech HD WebCam C270 и ноутбук Asus VivoBook Max X541NA (X541NA-GO123) со встроенной веб-камерой. При этом базы лиц для распознавания создавались и на ноутбуке, и на ПК. В качестве библиотеки работы с веб камерой была использована библиотека компьютерного зрения OpenCV, которая распространяется свободно в условиях лицензии BSD [69].

3.2.3.2. Разработка архитектуры нейронной сети

При проектировании топологии нейронной сети желательно точно знать количество людей, которые будут проходить аутентификацию, поскольку в выходном слое количество нейронов должно быть равно количеству лиц, которые сеть способна распознать, плюс один, если сеть никого не распознала. Но тогда возникают сложности с добавлением новых лиц для аутентификации, так как это требует полного переучивания сети каждый раз, что замедляет работу приложения. Чтоб этого избежать был применен следующий способ аутентификации: вместо того, чтобы нейронная сеть распознавала лицо человека, она сравнивает лица с двух изображений – одно из базы данных, другое только что поступившее с камеры. Все изображения в базе данных заранее подписаны. Считается, что если лица на изображениях из базы и с камеры совпадают на 75 % и более, то это лица одного и того же человека. Для более надежного распознавания, в базе данных хранится несколько фотографий одного человека. Все изображения обрабатываются в оттенках серого, что позволяет уменьшить количество

информации в три раза. Изображения из тестовой базы данных представлены на рисунке 3.38.

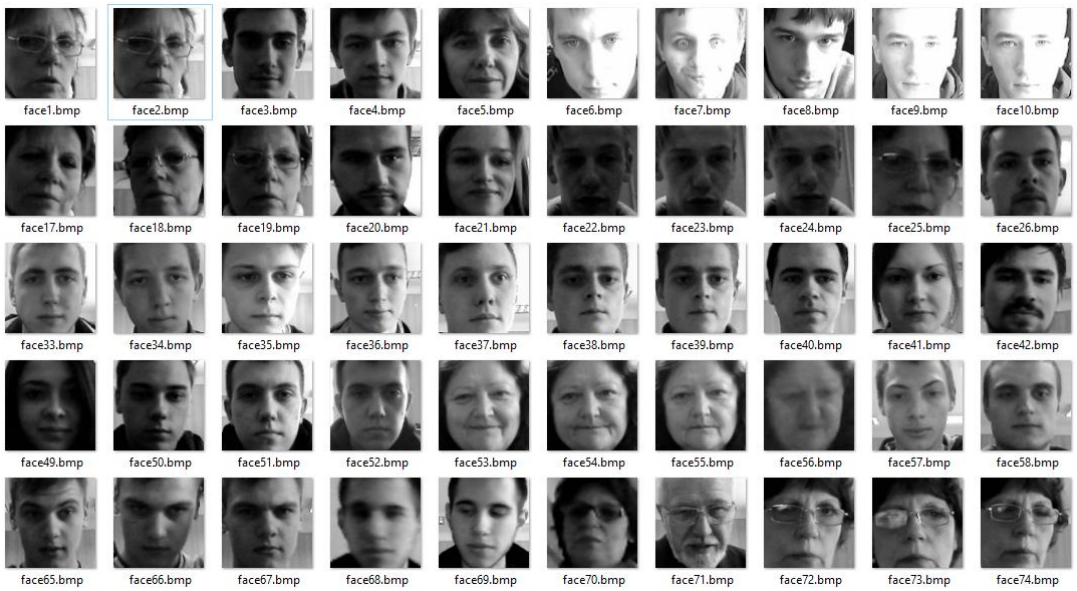


Рисунок 3.38 – Внешний вид базы изображений для тестирования системы

Архитектура разработанной нейронной сети представлена на рисунке 3.39.

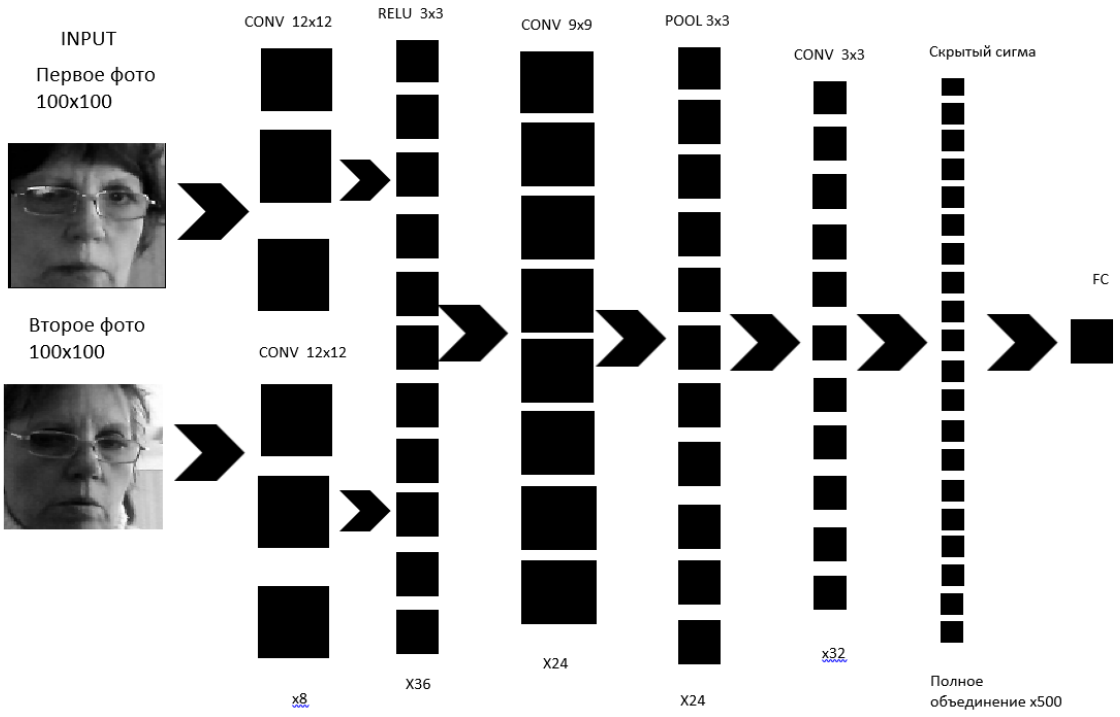


Рисунок 3.39 – Архитектура нейронной сети

В нее входят следующие слои:

INPUT (входные данные) [100×100×1] (ширина, высота, цветовые каналы, градации серого).

Слой CONV. Ядро свертки - [12×12]. Слой RELU maxpooling.

Слой POOL понижающий дискретизацию пространственных размеров до [16×16×12]. Логика работы такова: если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробной картинки.

Полносвязный скрытый нейронный слой.

Полносвязный слой soft-max для определения нужного класса.

На выходе – карта признаков и свойств, которые наиболее характерны для определенного класса. Классов условно два: совпадает и не совпадает. Оба класса выдают число от 0 до 1, как вероятность. В сумме, вероятности обоих классов будут равны 1, поэтому классом «не совпадает» можно пренебречь, подразумевая, что его значение будет равно 1 – вероятность первого класса. Свёрточная нейронная сеть, слой за слоем, трансформирует исходное изображение. В частности, слой свёртки и полносвязный скрытый слой выполняют преобразования, которые являются не только функцией от входного активационного объема, но и параметров (веса и смещения нейронов). С другой стороны, блок линейной ректификации и слой пулинга реализуют фиксированную функцию без параметров. Слои CONV и FC обучаются с помощью градиентного спуска SGD.

В самой свёрточной нейронной сети оба изображения пропускаются через слои свёртки параллельно друг другу и независимо, и только в конце подаются вместе на нейроны.

3.2.3.3. Программная реализация и обучение нейронной сети

Чтобы проверить работоспособность вышеперечисленных алгоритмов была написана программа на алгоритмическом языке C#, которая способна распознавать

человеческие лица на видеоряде от веб-камеры и сравнивать их с фотографиями, уже имеющиеся в базе.

В качестве библиотеки работы с веб-камерой была взята библиотека компьютерного зрения OpenCV, которая распространяется в условиях лицензии BSD. Из этой библиотеки была взята реализация алгоритма Виолы – Джонса.

Схематическое описание интерфейса представлено на рисунке 3.40.

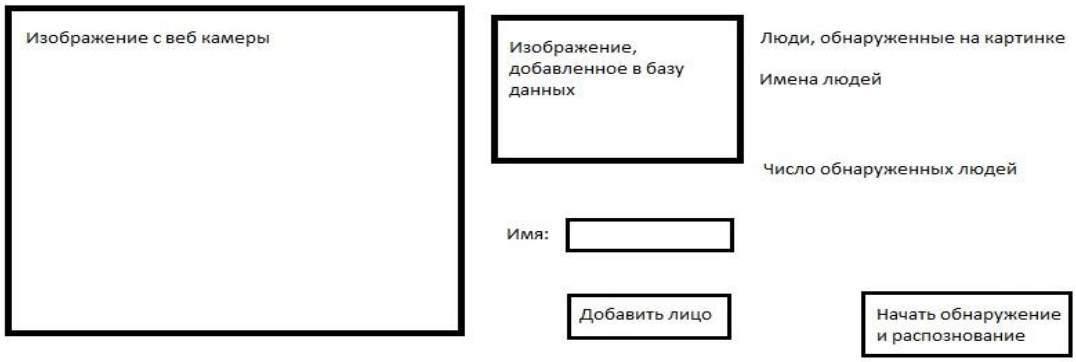


Рисунок 3.40 – Схема интерфейса программы.

Блок-схема алгоритма распознавания представлена на рисунке 3.41. Код фрагмента программы для распознавания представлен в приложении Г.

Рисунок 3.41 – Блок-схема программы.

В качестве алгоритма обучения использовался алгоритм обратного распространения ошибки с градиентным спуском [70]. Изначально все веса инициируются случайным образом. Изображение из обучающей выборки пропускается через всю сеть. На выходе получается случайный ответ. Вычисляется функция потери. Для оценки потери использовалась функция кросс-энтропии, которая значительно ускорила обучение, поскольку выходные нейроны являются сигмоидами. Затем идет перенастройка нейронных связей, минимизирующая функцию потери градиентным спуском. В качестве набора для обучения использовалась база изображений лиц, созданная в университете Карнеги-Меллон [71]. В базе 41368 изображений 68 человек. Изображения в серии для каждого человека отличаются небольшими деталями, ракурсом и выражением лица. Указана «похожесть» изображений относительно остальных в процентах.

3.2.3.4. Методика тестирования автоматизированной системы распознавания лиц

Для тестирования приложения для распознавания лиц при помощи нейронной сети было создано 4 базы данных из лиц студентов и преподавателей кафедры радиофизики и инфокоммуникационных технологий из 2, 10, 25 и 50 изображений. База из 2-х лиц автора и научного руководителя применялась на ранних этапах тестирования, для понимания работы алгоритма. Основные измерения проводились на базе из 50 изображений лиц, еще 2 (10 и 25 изображений) были задействованы, когда выяснилось, что скорость работы алгоритма сильно зависит от количества изображений в базе. В работе использовались 2 компьютера и соответственно 2 видеокамеры: ПК Acer Aspire TC-780 (DT.B8DME.007) с веб-камерой Logitech HD WebCam C270 и ноутбук Asus VivoBook Max X541NA (X541NA-GO123) со встроенной веб-камерой, операционной системой Windows 10. При этом базы создавались на ноутбуке и на ПК [72].

Сначала осуществлялась регистрация в базу данных. Студент или преподаватель садился фронтально на расстоянии 0,5 м от камеры и регистрировался, записывая свое имя, когда приложение захватит его лицо (выделит в красный квадрат). Регистрация осуществлялась на протяжении месяца.

После того, как, базы изображений были сформированы, были проведены тесты на распознавание. В них принимали участие 30 человек из тех, кто был зарегистрирован в базе и 20 человек из тех, кто не был зарегистрирован. Лицо считалось обнаруженным, когда программа выдавала сообщение «Число обнаруженных лиц 1» и лицо считалось распознанным, если над ним появлялась надпись с именем. Одно и тоже лицо предъявлялось алгоритму по нескольку раз, при разном освещении, с разными прическами и другими меняющими внешность факторами. Эксперименты проводились в дневное время в 9:00 до 13:00 с разным уровнем освещенности, с включенным освещением и выключенным. В процессе тестирования испытуемые применяли маскировку (борода, очки шапки), меняли прическу, одежду, мимику лица. База изображений для тестирования системы состояла из 50 изображений лиц: 17 женских и 33 мужских студентов (41) и преподавателей (9). Изображения лиц сохранялись в папке TrainedFaces в виде файлов в формате bmp. Там же находился и текстовый файл с именами, зарегистрированных людей, который можно редактировать.

3.2.3.5. Результаты распознавания

В работе использовались стандартные метрики, которые применяются в системах аутентификации и имеют названия: TAR (True Accept Rate), TRR (True Reject Rate), FAR (False Accept Rate), FRR (False Reject Rate).

Результаты тестирования для базы данных из 50 лиц приведены в таблице 3.9. Количество лиц, отнесенных к одной из четырех категорий, делится на количество лиц, предъявленных алгоритму в эксперименте, и умножается на 100 %. Системе предъявлялись лица, имеющиеся в базе данных и отсутствующие в базе. При предъявлении, лица, имеющиеся в базе данных, составляли 50 % от

общего количества предъявляемых лиц, поскольку для систем аутентификации важно, как чтобы осуществлялся допуск зарегистрированного пользователя, так и отказ в допуске незарегистрированного. Из таблицы 3.9 видно, что результат распознавания сильно зависит от освещенности.

Таблица 3.9 – Результаты тестирования автоматизированной системы

Описание эксперимента	FAR (%) не имеется в базе данных, но распознано, как имеющееся	FRR (%) имеется в базе данных, но не распознано	TAR (%) имеется в базе данных и распознано правильно	TRR (%) не имеется в базе данных и не распознано	Всего предъявлено лиц из базы/ не из базы
Предъявление лиц при одинаковых условиях с условиями регистрации	7	8	42	43	40/20
Предъявление лиц в условиях разной освещенности	0	64	3	33	40/20
Увеличение расстояния от объекта до видеокамеры до 3м	5	8	42	45	40/20

На рисунке 3.42 показан результат распознавания лица, зарегистрированного при той же освещенности.

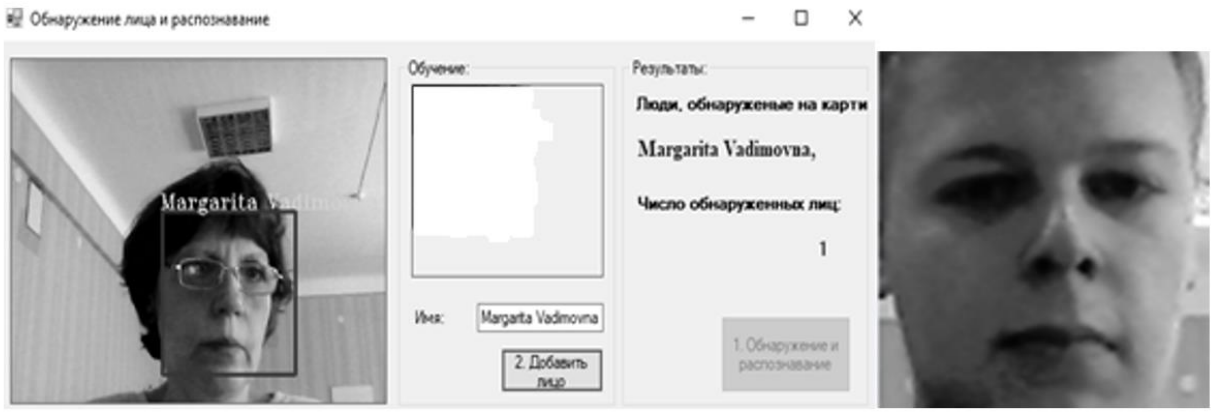


Рисунок 3.42 – Слева-процесс распознавания лица приложением, справа- лицо, которое распознавалось системой, как зарегистрированный пользователь и увеличило FAR системы

Лицо студента Токаржевского Романа, из 5-ти незарегистрированных в базе лиц, которых алгоритм распознал, как зарегистрированных, троих, он определил, как Токаржевского. Очевидно это лицо обладает особыми антропометрическими свойствами. Поскольку распознавание проводилось при разном освещении, то изображения в базе данных получались разной яркости. На рисунке 3.43 представлены 2 лица из базы данных и их гистограммы яркости по трем цветовым каналам.

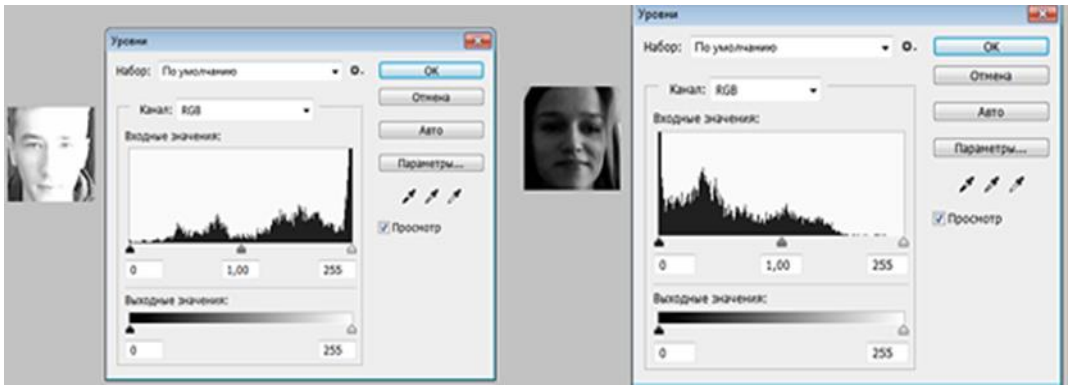


Рисунок 3.43 – Разница гистограмм изображений, зарегистрированных при разном освещении

Видно, что разброс яркости очень велик и поэтому алгоритм очень плохо распознает лица, занесенные в базу данных при разном, с условиями предъявления для распознавания, освещении. Увеличение расстояния от объекта до веб-камеры с 0,5 до 3 метров существенно не повлияло на результат распознавания, пока алгоритм мог различать лицо на сцене он распознавал его так же, как и на близком расстоянии.

Предъявлялись и несколько лиц одновременно, результат был таким же, как и для одного лица. Алгоритм распознавал столько лиц, сколько помещалось на сцене. В процессе тестирования вносились изменения во внешность, для того чтобы проверить как влияет изменение биометрических характеристик на распознавание. За время, прошедшее с момента регистрации, два студента отрастили бороды. Алгоритм не распознал их, как присутствующих в базе, хотя в базе присутствовали лица, зарегистрированные с бородами. Однако и

незарегистрированных 2 лица алгоритм не распознал, как зарегистрированных. Таким образом борода увеличивает FAR и не влияет на FRR. А вот легкая небритость, на результаты распознавания не влияла совсем. Очки в тонкой оправе практически не влияли на результаты распознавания, в то время как очки в массивной оправе резко снижали TAR и увеличивали FRR

Результаты для головных уборов светлых и темных оказались разными. Очевидно, алгоритм отождествляет темный головной убор с прической, а светлый – с частью лица (Рисунок 3.44).

Рисунок 3.44 – Нетипичные результаты распознавания лиц в головных уборах

Для темных головных уборов TAR – 40 %, в то время как для светлых – 5 %. Были подготовлены фотографии лиц, которые имеются в базе данных для того, чтобы проверить можно ли ввести в заблуждение систему при помощи фотографии. Причем это были как фотографии, сделанные фотоаппаратом, так и распечатанные на бумаге изображения лиц, имеющихся в базе. Только 2 фотографии, из имеющихся в базе данных были распознаны правильно. Но ни одно лицо с фотографии, сделанной фотоаппаратом не было правильно распознано (Рисунок 3.45).

Рисунок 3.45 – Алгоритм выделяет лицо на фотографии, но не распознает его, справа – фотография, сделанная фотоаппаратом, слева – распечатанное изображение из базы данных

А вот когда предъявлялось изображение лица с экрана мобильного телефона, то случаи ложного допуска имели место, хотя большинство лиц, имеющихся в базе данных не были распознаны. Скорее всего это результат разной яркости изображений на экране мобильного телефона и захваченного веб-камерой (Рисунок 3.46).

Рисунок 3.46 – Изображение лица с экрана мобильного телефона распознается неправильно.

Небольшое (до 10 %) изменение ракурса лица при распознавании не сильно влияло на результаты, что вообще характерно для нейросетевых алгоритмов. А вот когда поворот головы составлял более 10 %, то алгоритм не распознавал лиц вообще, ни имеющихся в базе данных, ни тех, которых в базе не было. Но при

сохранении фронтального расположения лица, и вращении его вправо-влево результаты распознавания были такими же, как и при прямом расположении.

Изменение мимики не влияло на результаты распознавания совсем. Студенты выражали на лицах различные эмоции – грусть, радость, удивление, страх. Алгоритм прекрасно распознавал лица и результаты не отличались от результатов с нейтральным выражением (Рисунок 3.47).

Рисунок 3.47 – Изменение мимики никак не влияет на результаты распознавания

В процессе того, как объем базы изображений лиц увеличивался стало заметно, что скорость работы алгоритма зависит от количества изображений в базе. Поэтому были проведены эксперименты с базами лиц разных объемов. Лицо предъявлялось в камеру и время, прошедшее с этого момента, до того, как появлялась надпись о том, что лицо обнаружено и распознано считалось временем распознавания. Время засекалось секундомером для 20 предъявленных лиц, а затем результат усреднялся.

Из таблицы 3.10 видно, что зависимость эта нелинейная и объясняется тем, что алгоритм сравнивает предъявляемое для распознавания лицо со всеми изображениями, имеющимися в базе. По результатам экспериментов были предложениы возможные шаги по улучшению системы:

1) Выравнивание яркости захваченных камерой и хранящихся в базе изображений;

Таблица 3.10 – Зависимость скорости работы алгоритма с изменением объёма базы изображений

Объем базы изображений лиц (шт.)	10	25	50
Скорость распознавания лица (с)	1	3	4

2) Возможность поиска и идентификации не по одному изображению лица, а по нескольким в различных ракурсах;

3) Фильтрация маскирующих признаков, которые влияют на работу алгоритма;

4) Организация поиска в базе более быстрым способом, чем поочередное сравнение, что особенно актуально для больших, по объему баз изображений.

Предложенная автоматизированная система аутентификации с распознаванием лиц функционирует удовлетворительно, при неизменной освещенности сцены для записи изображений лиц в базу данных и для предъявления лиц на распознавание.

Основные характеристики:

TAR (True Accept Rate) – лицо имеется в базе данных и распознано правильно – 42%, TRR (True Reject Rate) – лицо не имеется в базе данных и не распознано – 43%, FAR (False Accept Rate) – лицо не имеется в базе данных, но распознано, как имеющееся – 7%, FRR (False Reject Rate) – лицо имеется в базе данных, но не распознано – 6%.

Результаты распознавания зависят от освещения в помещении, изменение ракурса более, чем на 10%, таких маскирующих признаков, как борода, очки в массивной оправе, прическа, закрывающая лицо, светлый головной убор. Не зависят от изменения ракурса менее, чем на 10%, изменения мимики, наличие темного головного убора. Алгоритм невозможно «обмануть», предъявляя фотографию или изображение с экрана мобильного телефона вместо лица

человека. Скорость распознавания лица зависит от объема базы записанных изображений, причем нелинейно убывает с возрастанием объема базы.

К несомненным достоинствам алгоритма можно отнести тот факт, что не требуется переобучение системы, при добавлении нового лица в базу. Система заняла на диске 19,5 МБ вместе с базой данных (18,5 без базы данных). После обрезки методом, описанным в разделе 2, автоматизированная система заняла 1 МБ без потери точности.

3.2.4. АСНИ видеонаблюдения по распознаванию предметов повышенной опасности

В автоматизированных системах видеонаблюдения нейросетевые алгоритмы обнаруживают объекты специфического размера и формы (оружие, взрывчатку), игнорируя несоответствующие объекты. Они могут отслеживать эти объекты, принимая во внимание возможные последствия (нож зажат в руке специфическим хватом), и генерировать сигнал тревоги. При принятии решения о подаче тревоги, алгоритм принимает во внимание факторы, которые закладывает разработчик, или предоставляет пользователю самостоятельно выбирать угрозы из имеющегося списка [73]. Для реализации проекта автоматизированной системы были выбраны глубокие сверточные нейронные сети, которые позволяют значительно уменьшить объем характерных признаков изображения, применять распараллеливание для увеличения скорости обработки, ведь распознавание должно происходить в режиме близком к реальному времени [74].

3.2.4.1. Алгоритм распознавания и архитектура нейронной сети

Для распознавания были выбраны четыре типа объектов повышенной опасности: нож, пистолет, автомат и граната. Эти объекты образовали четыре класса, по которым должно проводиться распознавание. Алгоритм позволяет увеличивать количество классов, но тогда для обработки изображений потребуются более мощные средства и оптимизация самой программы.

Задачами, решаемыми в данной работе, были следующие. Первая, каким образом и по каким признакам различить опасный предмет от безопасного. В некоторых случаях даже натренированный оператор не может точно определить, какой именно предмет на видео. Предложенным решением является классификация предмета с определенной вероятностью, и в случае, когда эта вероятность выше установленного порога, утверждать, что предмет относится к одному из четырех классов. Этот порог предполагалось найти экспериментально на этапе обучения системы. Вторая задача, для корректной работы большинства нейронных сетей необходимо нормализовать входные данные, то есть подавать на вход изображения определенного размера. Были рассмотрены три варианта решения этой задачи:

– изначально настроить камеру на съемку кадров определенного размера, однако функция съемки в заданном разрешении доступна не для всех видеокамер и данное обстоятельство усложнит развертывание системы на объекте, уже оснащенном видеосистемой;

– произвести пост обработку изображения, однако это замедлило бы процесс распознавания и увеличило расход оперативной памяти вычислительной системы;

– использовать специальную сверточную архитектуру нейронной сети, которая позволяет пропорционально увеличивать или уменьшать количество выходных ячеек сетки. Как показали проведенные исследования, данный подход оказался наиболее перспективным, что позволило предложить следующий алгоритм распознавания:

1) Активация камеры.
2) Из видеопотока выбирается каждый 10-й кадр.
3) Предварительная обработка изображения.
4) Изображение подается на вход нейронной сети.
5) Нейронная сеть обнаруживает или не обнаруживает принадлежность объекта(ов) на кадре к одному из четырех классов (в случае обнаружения, указывается вероятность, с которой выделенный на кадре предмет соответствует выбранному классу).

6) Система оповещает оператора видео или аудио сигналом.

Захваченные кадры проходят следующую предварительную обработку: из полноцветного изображение переводится в полутоновое, производится фильтрация шумов и улучшение по алгоритму Винера.

В основу нейронной сети легла усовершенствованная, для увеличения скорости работы и уменьшения требования к производительности аппаратной платформы, архитектура YOLO 3-й версии фреймворка Darknet -53, CUDA Toolkit 10.2, с использованием методов якорей и K-Means [75]. В стандартную архитектуру были внесены следующие изменения:

– исключен сверточный слой 13 x 13, отвечающий за распознавание крупных объектов;

– добавлено 6 якорей детектирования для увеличения точности распознавания;

– добавлено 3 выходных сверточных слоя для предсказания ограничивающей рамки трехмерного тензорного кодирования, наличия объекта и одного из классов. Поскольку данная сеть является глубокой и ресурсоемкой, в дальнейшем проводилась редукция проигравших нейронов по методу, описанному в разделе 2.

3.2.4.2. Обучение нейронной сети

Обучение производилось на 2000 изображениях опасных предметов. На подготовленных образцах присутствовал либо один объект, соответствующий одному из четырех классов, либо несколько объектов из разных классов, либо несколько объектов одно и того же класса, либо объект, имеющий сходства с любым из распознаваемых классов, последний вариант был включен, чтобы алгоритм не распознавл сторонние объекты, как соответствующие одному из классов. Изображения размечались вручную (Рисунок 3.48). Затем данные распределялись на обучающую выборку, используемую для корректировки весов (80 %), и тестовую выборку (20 %), с помощью которой проверялось качество распознавания на разных этапах, и отсутствие переобучения сети (наличие

изменения градиента). На нейронную сеть в случайном порядке подавались изображения из группы для тренировки, затем выдавался ответ, распознан или не распознан объект.

Рисунок 3.48 – Процесс разметки обучающей выборки

Результат работы алгоритма сравнивался с реальным классом и затем производилась корректировка весов по алгоритму Левенберга-Марквардта [76]. Для определения оптимального вероятностного порога распознавания проводилось несколько таких циклов обучения на одних и тех же данных [77].

3.2.4.3. Тестирование автоматизированной системы видеонаблюдения

В разработке GUI приложения на языке программирования Python 3.6.5 применялась библиотека Tensorflow 2.0.2.

Процесс распознавания представлен на рисунке 3.49. Захваченный кадр разграничивается сеткой SxS ячеек. Каждая ячейка имеет от 3 до 6-ти якорей.

Для каждого якоря рассчитываются координаты, вероятность наличия объекта и вероятность принадлежности к каждому из классов. После формирования всех якорей происходит процесс фильтрации, по окончанию которого остаются только те, у которых значение вероятности выше, заданного при

обучении порога. Предполагаемый объект повышенной опасности выделяется рамкой с указанием вероятности распознавания в процентах.

Первичное тестирование проводилось на видео, с изображением различных предметов, в том числе и повышенной опасности для определения слабых мест алгоритма.

Рисунок 3.49 – Процесс распознавания с разграничивающей сеткой и тремя якорями (слева), результат распознавания ножа, зажатого нехарактерным хватом (справа)

Для окончательных тестов были взяты достоверные макеты объектов повышенной опасности. Предметы предъявлялись в четырех положениях: лежали на столе, были видны полностью в руке, зажаты характерным хватом, когда видно 80 и 60% предмета. Тестирование проводилось при разном освещении – нормальном и тусклом (отличие от нормального приблизительно на 40 %).

Для тестирования использовался стенд:

CPU – Intel Core i7-5700HQ, GPU – NVidia GeForce GTX 960M (2 Гб GDDR5), RAM – 12 Гб DDR3L-1600 МГц, Web Cam – HD (1080*720)

Помимо прочих факторов менялось и расстояние до объекта, поскольку при увеличении расстояния уменьшается и количество пикселей, которые захватывает распознаваемый предмет. Тестирование проводилось на расстояниях 3 и 5 метров,

опасные предметы предъявлялись системе в фас и в профиль. Результаты в процентном выражении представлены в таблице 3.11. Из таблицы 3.11 видно, что сеть распознавала опасные предметы на расстоянии в 3 метра в среднем в 96,84 % случаев и на расстоянии 5 метров в 89,06 % случаев. Полученный результат не хуже, чем в существующих решениях [78].

Предложенный алгоритм распознавания работает удовлетворительно и как ожидалось, лучшие результаты получены при условиях, приближенных к идеальным (освещение нормальное, расстояние малое).

Таблица 3.11 – Результаты распознавания предметов повышенной опасности в процентах

Эксперимент	Фас				Профиль			
	Нож	Пистолет	Автомат	Граната	Нож	Пистолет	Автомат	Граната
3 м. нормальное освещение								
На столе	100	100	100	97,5	92,5	100	100	97,5
В руке	100	97,5	100	97,5	97,5	97,5	100	97,5
Видно 80%	97,5	97,5	100	97,5	92,5	100	100	100
Видно 60%	95	92,5	95	92,5	90	95	97,5	92,5
5 м. нормальное освещение								
На столе	97,5	97,5	100	95	95	100	100	97,5
В руке	100	97,5	97,5	95	97,5	97,5	100	92,5
Видно 80%	97,5	97,5	97,5	95	92,5	100	97,5	95
Видно 60%	97,5	95	97,5	95	85	95	95	95
3 м. тусклое освещение								
На столе	95	100	100	97,5	92,5	100	100	97,5
В руке	100	100	97,5	92,5	97,5	100	100	92,5
Видно 80%	95	95	100	97,5	90	97,5	100	95
Видно 60%	95	95	95	95	87,5	95	97,5	92,5
5 м. тусклое освещение								
На столе	92,5	92,5	95	95	92,5	92,5	95	92,5
В руке	95	92,5	97,5	92,5	95	95	95	92,5
Видно 80%	90	90	90	87,5	80	92,5	90	90
Видно 60%	87,5	90	90	87,5	82,5	92,5	92,5	90

Разработанная система способна обрабатывать изображения с частотой ~30 кадров в секунду, что вполне достаточно для работы в реальном времени. До редукции параметров методом удаления проигравших нейронов модель занимала объем постоянной памяти 98 МБ, а вся библиотека Tensorflow 206 МБ, после редукции модель стала занимать 12 МБ.

Был разработан интерфейс и дописаны классы, отвечающие за выбранные опасные предметы. Подготовлены данные для обучения, проведено обучение на статических изображениях опасных предметов, в ходе обучения осуществлялась корректировка параметров.

Предложено решение трудности распознавания ножа (узкое лезвие, недостаточное освещение), путем введения признака «характерный хват».

Установлено, что действенность предложенного алгоритма распознавания зависит от качества изображения на носителе, а такие факторы как размытие, шум, нечеткость и т.д. отрицательно сказываются на его работе. Все предметы лучше распознавались в фас (97,3 % при нормальном освещении и 94,2 % при тусклом освещении) и хуже в профиль (96,4 % при нормальном освещении и 93,6 % при тусклом освещении).

В реальных условиях, автоматизированная система видеонаблюдения по распознаванию предметов повышенной опасности, на основе нейронной сети должна быть обучена с носителя, например, телекамеры, установленного на объекте и предполагаемых опасных предметах при разных условиях предъявления, что предусмотрено логикой программы. По результатам тестирования намечены пути усовершенствования системы: увеличение количества и разнообразия данных для обучения и выравнивание освещенности захваченных кадров. Для сокращения затратности алгоритма проводилась обрезка, по методу, предложенному в разделе 2. В результате количество параметров нейронной сети уменьшилось на 30 %, и система стала работать на 20 % быстрее.

3.2.3.4 Внедрение системы распознавания лиц в монитор видеодомофона М-480М

Результаты разработки системы распознавания лиц сверточной нейронной сетью были внедрены в монитор М-480М, разработанный на предприятии ООО «фирма МДЛ». Для обработки изображения использовалась система на кристалле Allwiner A-13, с рабочей частотой 1 ГГц, 1 ядерный, ядро Cortex A8, объемом ОЗУ 512 МБ, ПЗУ – 4 ГБ, графическим чипом MALI 400MP. На рисунке 3.50 представлен модуль с установленной Allwiner A-13 и ПЗУ.

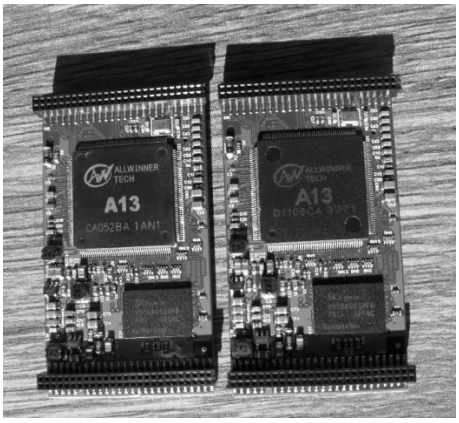


Рисунок 3.50 – модуль с установленной Allwiner A-13 и ПЗУ

В процессе работы каждые 0,5 секунд модуль записывал изображение сцены и в итоге постоянное запоминающее устройство переполнялось большим объемом записанных кадров. Было предложено проводить запись только при попадании в кадр лица. Затем система определяет наличие лица в базе данных и записывает эти кадры в разные папки. Для распознавания попадающих в кадр лиц и использовалась разработанная в разделе 3.2.2.3 нейросетевая система, оптимизированная методом, представленным в разделе 2. Процесс распознавания представлен на рисунке 3.51.

В результате тестовых испытаний алгоритм показал точность 89 % и скорость распознавания 1 лицо в 0,5 с при записи 5-6 лиц в базу данных, что характерно для охранных систем на одну квартиру или частный дом. В охранной стстеме для дома

или офиса база лиц увеличивается до 100-150 и это уменьшает скорость распознавания до 1 лицо в 4-4,5 с.

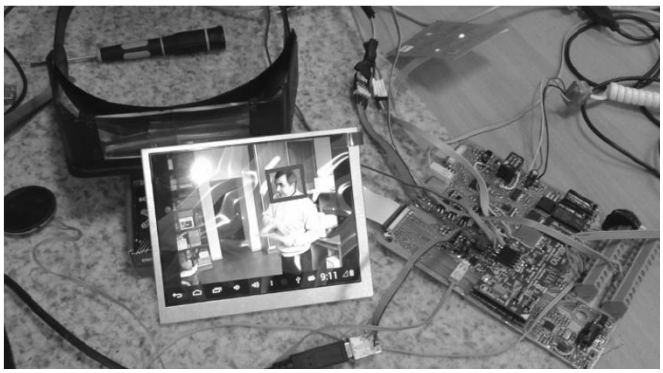


Рисунок 3.51 – Процесс распознавания лица на мониторе

3.3 Выводы по разделу 3

1. Разработаны 2 автоматизированные системы обработки формализованных данных (анализа файлов логов и определения каналов утечки информации) на нейронных сетях типа двухслойный персептрон и рекуррентной сети Элмана и 4 автоматизированные системы обработки изображений (распознавания формы предметов на основе сети LVQ, аутентификации пользователей по отпечаткам пальцев на основе сети RBF и по лицу на основе сверточной сети с пулингом, распознавания опасных предметов в режиме реального времени на основе сверточной сети глубокого обучения).

2. Показано, что успешность работы таких систем зависит от методики подготовки данных для обучения, алгоритма обучения, выбранной архитектуры или модели и достигает точности 85-97 %.

3. В случае систем, работающих с изображениями большое значение, имеют алгоритмы предварительной обработки изображений, как для улучшения качества, так и для выделения признаков, подающихся на нейронную сеть. В частности, для системы распознавания по отпечаткам пальцев был разработан собственный алгоритм бинаризации и применен детектор Харриса для выделения признаков.

4. Автоматизированные системы на нейронных сетях, состоящие из 2-х, 3-х скрытых слоев не требуют оптимизации, поскольку не занимают значительных

вычислительных ресурсов и могут быть применены в микроконтроллерных системах или одноплатных мини-компьютерных системах типа Raspberry Pi.

5. Автоматизированные системы глубокого обучения для распознавания изображений, с наличием большого количества сверточных слоев требуют уменьшения количества параметров. Применение метода редукции проигравших нейронов позволяет уменьшить количество параметров на 70-80 % без потери точности.

6. Результаты исследований внедрены в систему аутентификации и доступа на базе монитора видеодомофона М-480М с системой на кристалле Allwiner A-13. Достигнута точность распознавания 89 % и скорость распознавания 0,5 с/лицо. при записи 5-6 лиц в базу данных, 4,5 с/лицо при записи 150 лиц в базу данных.

РАЗДЕЛ 4
УГРОЗЫ АСНИ БЛ ОСНОВАННЫХ НА НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЯХ

В разделе рассмотрены атаки на нейросетевые классификаторы. Разработано приложение, позволяющее проводить как целевые, так и нецелевые атаки с заданием значения точности распознавания, выбором и контролем оптимизируемой метрики. Проведено исследование 9-ти методов генерации состязательных примеров для ненаправленных и направленных атак. Сгенерированные в результате направленных и ненаправленных атак состязательные изображения предъявлялись поисковым системам Bing, Google и Yandex для распознания, которые оказались устойчивы к направленным атакам типа «черный ящик». Алгоритмы Google и Yandex показали уязвимость к ненаправленным атакам. Предложены методы защиты от атак на нейросетевые алгоритмы, на основе анализа полученных результатов по каждой атаке. Сделан вывод о необходимости дополнять автоматизированные системы аутентификации инструментами поиска уязвимостей и предусматривать защитные меры.

4.1 Угрозы для автоматизированных нейросетевых классификаторов

Применение нейросетевых алгоритмов в автоматизированных системах позволяет повысить их надежность и производительность. Однако, особенности функционирования нейронных сетей делают их уязвимыми к целенаправленным атакам. В особенности это касается классификаторов образов [79]. Достаточно заново обучить нейронную сеть, но используя обратное распространение ошибки не для изменения весовых коэффициентов, а для редактирования точек изображения. Сгенерированные фальшивые изображения будут некорректно классифицироваться нейронной сетью, которая их создала.

Для исследования атак на нейросетевые классификаторы была разработана программа, которая модифицирует предварительно обученную модель нейронной сети таким образом, что полученное изображение будет распознаваться как другой

предмет, хотя визуально останется прежним. Программа имела возможность регулирования количества итераций преобразования, а, следовательно, количества измененных пикселей. Исходные и модифицированные изображения подавались на известные классификаторы, которые компании предоставляют для свободного пользования в сети Интернет. Одновременно для сравнения подавались изображения, модифицированные при помощи дифференциального генетического алгоритма [80]. Исследованы возможности применения алгоритма градиентного спуска с искажениями для введения в заблуждение известных нейросетевых классификаторов и уязвимостях автоматизированных систем аутентификации к данному виду атак.

Упомянутые выше алгоритмы в литературных источниках имеют различные названия и модификации, поэтому автор воспользовался собственной классификацией [79, с. 51].

Для создания фальшивых изображений использовались:

- градиентный спуск с добавлением статического белого шума по всему изображению (Gradient Descent with static White Noise GDWN);

- градиентный спуск с добавлением ограниченного количества искажающих данных (Gradient Descent with a limited amount of Distortion Data GDDD);

- заплатка на основе дифференциального генетического алгоритма (DifferentialGeneticAlgorithmPatch DEAP);

Последний алгоритм не был реализован в программе, использовалась заплатка, описанная в работе [81], которая неплохо зарекомендовала себя в предыдущих исследованиях.

Для классификации использовалась предварительно обученная, на наборе данных MNISTв ImageNet, нейронная сеть MobileNetKeras с 'channels_first' форматом данных (каналы, высота, ширина), используемая совместно с библиотеками Keras 2.1.5 и TensorFlow 2.2.2 [82]. Данная сеть обнаруживает 1000 различных типов объектов. Модель MobileNet была выбрана, так как она использует меньше оперативной памяти, чем подобные решения Keras и других производителей. Размер ввода по умолчанию для этой модели – 224x224.

Эксперименты проводились на персональном компьютере с процессором Intel®, Pentium®, CPUG6300 2.7 GHz и видеоадаптером NVIDIAGeForceCTX 1050 Ti. Процесс создания фальшивого изображения представлен на рисунке 4.1.

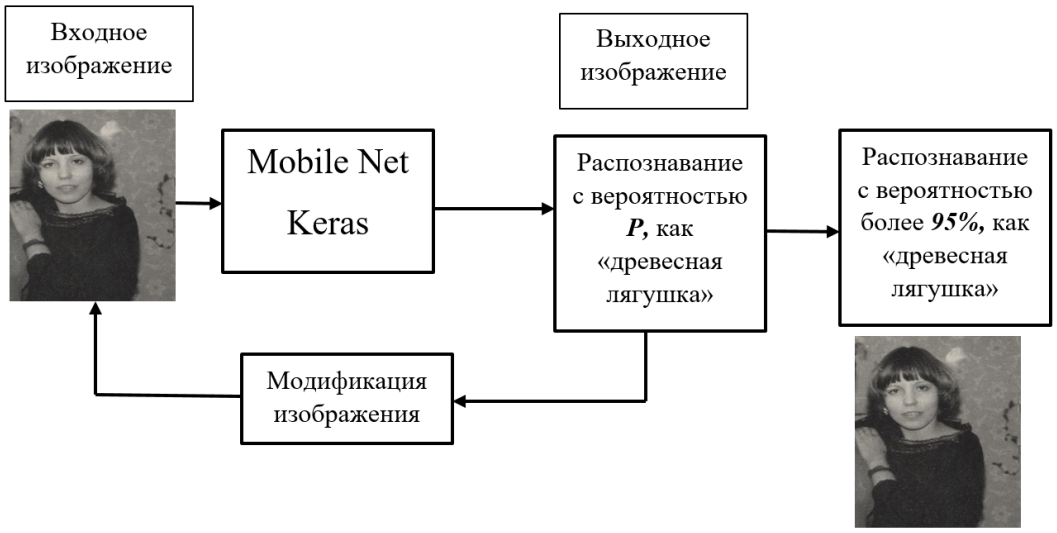


Рисунок 4.1 – При генерации фальшивого изображения методом обратного распространения ошибки, пиксели изменяются так, чтобы результат стал немного ближе к нужному ответу на каждой итерации

Фотография, которую сеть классифицирует как «девушка» с вероятностью 95%, после модификации алгоритмами GDWN и GDDD должна классифицироваться, как «древесая лягушка», с не менее чем с 95% вероятностью. Изображение для классификации должно быть похожим на исходное изображение не только на глаз, но и по значению хеш-суммы (максимальная разница в 2 бита с использованием p-хэша).

Сгенерированные изображения не обязательно должны быть близки к базовому изображению для работы в локальных системах, но для классификаторов, используемых в компьютерных сетях, это может иметь значение, так как при загрузке проверяются хеш-суммы. Вероятность, с которой сеть будет распознавать девушку как лягушку можно настраивать в программе.

В идеале можно получить уверенность 99 % и разницу в 0 бит. Однако позволяя регулировать пиксели без каких-либо ограничений, изменения в

изображении могут стать достаточно значительными, и человек сможет их увидеть, как обесцвеченные или волнистые области. Чтобы предотвратить подобные искажения, в алгоритм добавлено ограничение: ни один пиксель в фальшивом изображении не может быть изменен более чем на 0,01% от исходного. Это условие настраивается в программе переменными max_change_above и max_change_below. Предмет, за который принимает девушку нейронная сеть можно изменять, изменяя номер изображения для обучения в базе MNIST. Для древесной лягушки это 31, для кольчуги – 52, для банана – 63, для тостера – 859. Тестовое изображение может быть цветным или в оттенках серого, в формате tif, png или jpeg. На изображении должен быть один предмет, если предметов больше одного, то классификация производится некорректно.

Сначала на обученную сеть подается исходное изображение, которое распознается как «девушка», с вероятностью 99,97521 %. Затем производится обработка изображения по алгоритму GDWN, то есть накладывается белый шум, а затем значения яркости пикселей корректируются градиентным спуском, пока не получится нужный результат (Рисунок 4.2). На второй фотографии визуально девушка, но сеть MobileNet, распознает ее как древесную лягушку с вероятностью 99,965894 %.

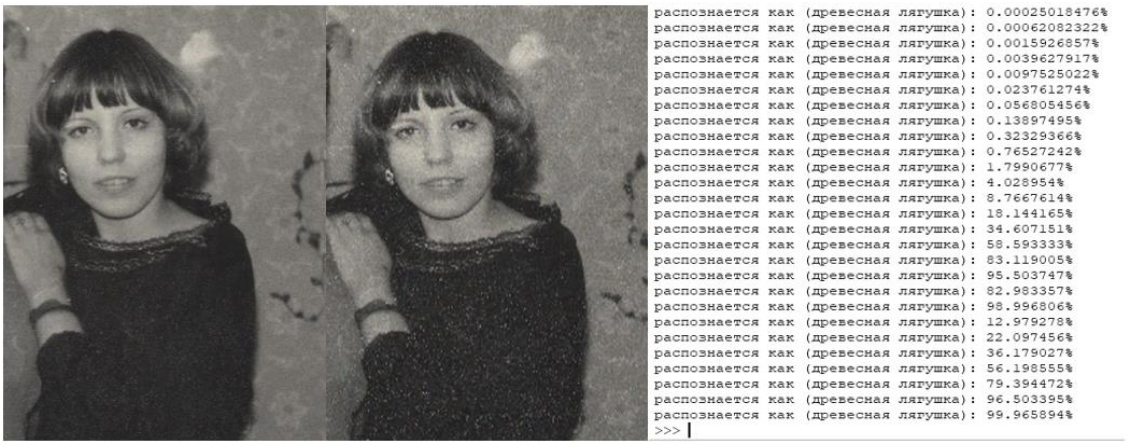


Рисунок 4.2 – Слева – исходное изображение, распознаваемое нейронной сетью MobileNet, как «девушка», в центре – сгенерированное алгоритмом GDWN изображение, распознаваемое нейронной сетью, как «древесная лягушка», справа – процесс преобразования изображения.

На рисунке видно, что полученное фальсифицированное изображение (состязательный пример) немного зашумлено, появились артефакты характерные для алгоритма GDWN. Затем производится обработка изображения по алгоритму GDDD, то есть не накладывается белый шум, а Keras рассчитывает градиент на основе входного изображения и текущего предсказанного класса, в переменную model_input_layer возвращается изображение, которое модифицируется на каждой итерации. Максимальное значение модифицируемых пикселей, ограничено 0.01%, как и в предыдущем случае.

Результаты преобразования в целом аналогичны преобразованиям алгоритмом GDWN, получалось меньше артефактов на выходных изображениях.

Когда изображение прогонялось через нейронную сеть для модификации несколько раз, количество итераций модификации сокращалось нелинейно. Усредненные результаты эксперимента представлены на рисунке 4.3.

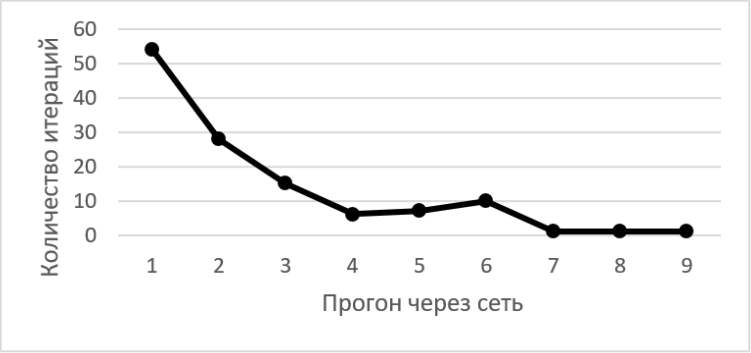


Рисунок 4.3 – Количество итераций алгоритма GDDD в зависимости от количества прогонов через сеть MobileNet

Видно, что после 7-го прогона модификации не происходят, поскольку достигнуто пороговое значение 0,01 %. При выставлении других пороговых значений, выходные изображения в итоге распознаются сетью MobileNet как целевые с соответствующей меньшей вероятностью, при неизменном внешнем виде.

Изображение заплатки для модификации алгоритмом DEAP было взято из работы «Заплатка для фальсификации», модифицированное изображение должно

распознаваться как «тостер» [81]. Заплатка накладывалась на исходное изображение в различных местах и с разными размерами (Рисунок 4.4).

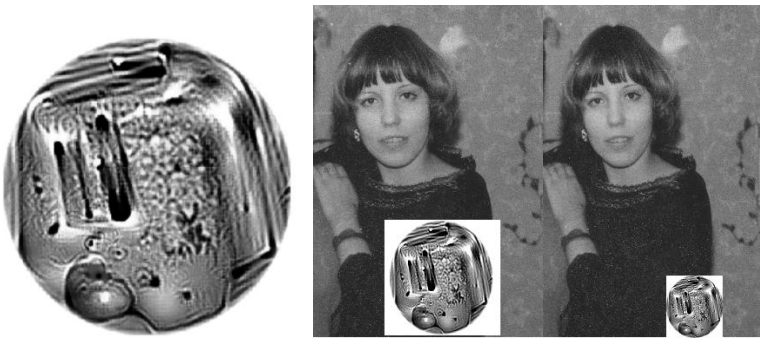


Рисунок 4.4 – Слева – заплатка для модификации изображения в класс «тостер», справа – исходное изображение с наложенной заплаткой различных размеров.

Место накладывания влияло на результаты распознавания только если заплатка накладывалась непосредственно на лицо, при этом вероятность распознавания девушки, как «тостер» резко приближалась к 100%. Во всех остальных случаях наблюдалось увеличение процента неправильного распознавания при увеличении размеров заплатки (Рисунок 4.5).

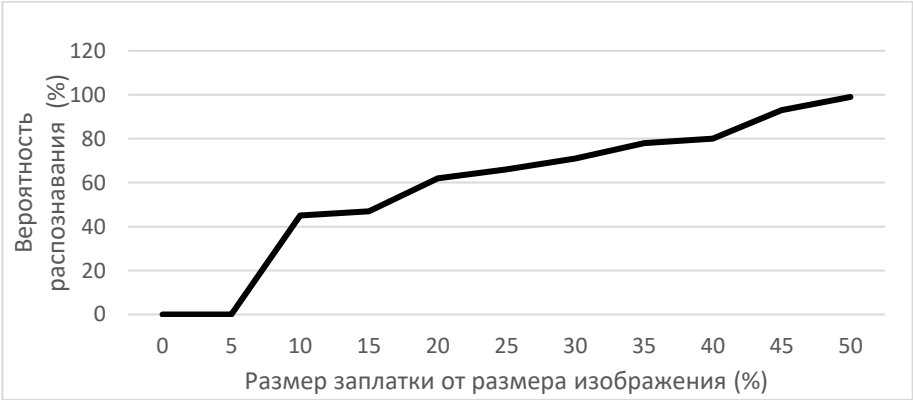


Рисунок 4.5 – Зависимость результатов распознавания тестового изображения как «тостер» в зависимости от размера заплатки по отношению к размеру изображения, сеть MobileNet

Для исследования распознавания фальсифицированных изображений сторонними ресурсами были выбраны 12 on-line общедоступных сервисов для

распознавания предметов и лиц, поиск по изображениям от Google и Yandex. Два последних ресурса не удалось ввести в заблуждение, они уверенно распознавали все модифицированные изображения, как «девушка» (Рисунок 4.6).

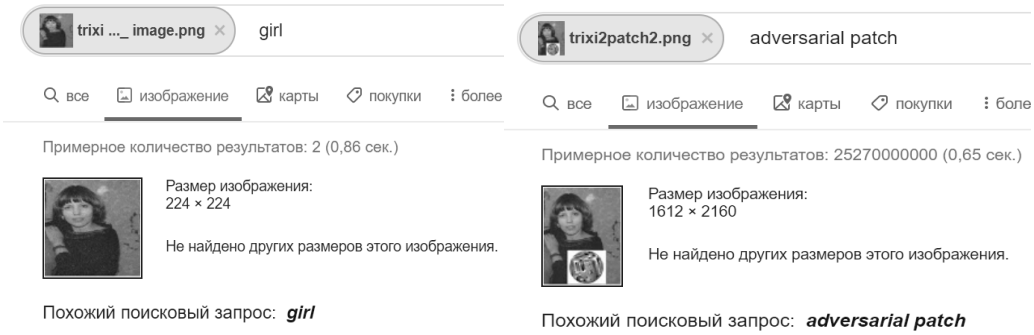


Рисунок 4.6 – Google распознает все модифицированные изображения как «girl», кроме изображения с заплаткой

Поиск изображений Google распознавал изображения с заплаткой как «Adversonalpatch», в то время как Яндекс распознавал, как «девушка». Это обусловлено различными подходами к выделению признаков для классификации и структурами нейронных слоев, а также небольшой популярностью данной тематики в российском секторе Интернета. Среди остальных доступных в Интернете 10-ти сайтов по распознаванию лиц «обмануть» удалось только 3. На рисунках 4.7, 4.8 представлены результаты распознавания модифицированных изображений на сайтах для автоматического генерирования подписей к фотографиям и распознавания образов.

Видно, что модифицированное изображение распознается как «зеленая лягушка», «кольчуга», а изображение с заплаткой как «тостер». В то время как немодифицированное изображение распознавалось как «женщина» или «портрет». В таблице 4.1 представлены результаты по всем протестированным ресурсам. Главным критерием являлось, распознает ли нейронная сеть изображение как «девушка» или хотя бы как «человек».

Рисунок 4.7 – Результаты распознавания, модифицированного алгоритмом GDWN и исходного изображений на сайте neuronus.com

Рисунок 4.8 – Результаты распознавания, исходного и модифицированного алгоритмами GDWN, DGAP и GDDD на сайте imagga.com

Сервисы, предназначенные для поиска похожих изображений, оказалось ввести в заблуждение труднее, чем сервисы для распознавания лиц. Легче всего вводились в заблуждение сервисы для классификации изображений. Сгенерированные нейронной сетью состязательные примеры распознаются неправильно той сетью, которой они были сгенерированы с вероятностью 99 % (при этом в процессе модификации эту вероятность можно менять).

Таблица 4.1 – Результаты распознавания модифицированных изображений на on-line ресурсах

№	Ресурс	GDWN	GDDD	DGAP
1	neuronus	Распозналось как «зеленая лягушка»	Распозналось как «зеленая лягушка»	Распозналось как «тостер»
2	imagga	Распозналось как «кольчуга»	Распозналось как «зеленая лягушка»	Распозналось как «тостер»
3	tineye	Не распозналось	Не распозналось	Распозналось как «Adversionalpatch»
4	images.google	Распозналось как «девушка»	Распозналось как «девушка»	Распозналось как «Adversionalpatch»
5	yandex/images	Распозналось как «девушка»	Распозналось как «девушка»	Распозналось как «девушка»
6	betaface	Распозналось неправильно	Распозналось неправильно	Распозналось неправильно
7	how-old.net	Распозналось как «девушка» с вероятностью 27%	Распозналось как «девушка» с вероятностью 31%	Не распозналось
8	pimeyes	Не распозналось как лицо	Не распозналось как лицо	Распозналось неправильно
9	pic.sogou	Распозналось как «девушка»	Распозналось как «девушка»	Распозналось как «девушка»
10	bing	Распозналось как «девушка»	Распозналось как «девушка»	Распозналось как «девушка
11	kairos	Распозналось как «девушка»	Распозналось как «девушка»	Не распозналось как человек
12	aidemos.microsoft	Распозналось как «девушка»	Распозналось как «девушка»	Не загружалось

На сторонних нейросетевых ресурсах, работающих как нейросетевые классификаторы, они так же распознаются неправильно в 90 % случаев. Из 5-ти протестированных сервисов по поиску похожих изображений ввести в заблуждение не удалось ни один. Сервисы для распознавания лиц в большинстве случаев не распознают модифицированные изображения как лицо человека или распознают неправильно, например, как мужчину или ребенка.

4.2 Уязвимости АСНИ распознавания лиц

После того, как были исследованы атаки на простые классификаторы, встал вопрос можно ли заставить автоматизированную систему, распознающую лица, принимать лицо одного человека, за лицо другого человека. Для этого, во-первых, использовались более мощные сверточные нейронные сети, во-вторых, был расширен класс атак за счет использования библиотеки foolbox, разработанной Бренделем Виландом и Йонасом Раубером в 2017 году [83].

Использовались следующие виды атак (слева – название атаки в библиотеке, справа – название атаки из литературы):

1. GradientAttack – L-BFGS атака [41, с. 5];
2. GradientSignAttack – Быстрый метод градиентного знака [80, с. 685];
3. BasicIterativeMethod – Метод базовых итераций [84];
4. CarliniWagnerL2Attack – Атака Carlini&Wagner [85];
5. SinglePixelAttack – Однопиксельная атака [40, с. 7];
6. DeepFoolAttack – DeepFool [86];
7. GaussianBlurAttack – Атака, применяющая гауссовское размытие [87];
8. AdditiveGaussianNoiseAttack – Атака, добавляющая гауссовый шум [88];
9. ContrastReductionAttack – Атака, понижающая контрастность [89].

Суть 1, 2 атак уже была описана в разделе 1.4. Атаки 7, 8 и 9, как понятно из названия, изменяют контрастность и резкость изображения, накладывают Гауссов шум. Поэтому остановимся на менее известных методах.

Метод базовых итераций (BIM). Разработан для данных, которые передаются только через устройства (например, камеры, датчики). Поэтому был использован быстрый метод градиентного знака с тонкой оптимизацией для проведения нескольких итераций. В каждой итерации обрезались значения пикселей, чтобы избежать большого изменения каждого пикселя:

$$Clip_{x,\xi}\{x'\} = min\{255, x + \xi, max\{0, x - \epsilon, x'\}\}.$$

Здесь $Clip_{x,\xi}\{x'\}$ ограничивает изменение сгенерированного состязательного изображения в каждой итерации. Состязательные примеры генерируются в несколько итераций.

Нейронную сеть удалось ввести в заблуждение с помощью изображения, полученного с камеры мобильного телефона. Также выяснилось, что быстрый метод градиентного знака оказался устойчивым к фототрансформации, в то время как итерационные методы не могут противостоять фототрансформации [80].

Атака C&W Карлини и Вагнер применили целенаправленную атаку против защитной фильтрации. Согласно их исследованиям, C&W's эффективна для большинства существующих систем обнаружения состязательных примеров.

Сначала они определили новую функцию g, такую, что:

$$min\ |\eta|_p + c * g(x + \eta):$$

$$x + \eta \in [0,1]^n,$$

где $g(x') \geq 0$ тогда и только тогда, когда $f(x') \geq l'$. Были предложены 7 вариантов функции g. Одной из эффективных функций, оцениваемых их экспериментами, может была

$$g(x') = max(max_{i \neq l'}(Z(x')_i) - Z(x')_t, -\kappa),$$

где Z обозначает функцию softmax, κ – константа.

Во-вторых, вместо использования L-BFGS с ограничениями для поиска минимальных возмущений, авторы представили новый вариант w, которое должно удовлетворять равенство:

$$\eta = \frac{1}{2}(tanh(w) + 1) - x.$$

Было выполнено 20 итераций такой для поиска оптимального с с помощью бинарного поиска, после чего Карлини и Вагнер обнаружили, что если градиенты $|\eta|_p$ и $g(x+\eta)$ не совпадают по размерности, то трудно найти подходящую константу с во всех итерациях градиентного поиска и получить оптимальный результат. Из-за этого для двух из предложенных функций не нашлось оптимальных решений для состязательных примеров.

В-третьих, в статье обсуждались три измерения расстояний возмущения: l_0, l_2и l_∞.Авторы представили три вида атак, основанных на метрике расстояния: l_0 атака, l_2 атака и l_∞ атака.

l_2 атака может быть описана как:

$$min\,|\frac{1}{2}(tanh(w)+1)|_2 + c * g\left(\frac{1}{2}(tanh(w)+1)\right).$$

Авторы показали, что фильтрация в сети не может помочь в защите от атаки l_2.

l_0 атака была проведена итеративно, так как l_0 не дифференцируется. В каждой итерации несколько пикселей считаются тривиальными для генерирования состязательных примеров и удаляются. Значение пикселей определяется градиентом l_2. Итерация останавливается, если оставшиеся пиксели не могут сгенерировать состязательный пример.

l_∞ атака была также итерационной атакой:

$$min\ \text{с} * g(x+\eta) + \sum_i[(\eta_i - \tau)^+].$$

Для каждой итерации они уменьшали τ в 0.9 раз, если $\eta_i < \tau$, то τ считалось оценкой l_∞ [85].

Однопиксельная атака чтобы избежать проблемы измерения количества измененных пикселей, можно генерировать состязательные примеры, модифицируя только один пиксель. Тогда проблема оптимизации становится:

$$min\,J(f(x'), l'):$$
$$|\eta|_0 \leq \epsilon_0,$$

где $\epsilon_0 = 1$ для изменения только одного пикселя. Новое ограничение усложняет оптимизацию задачи. Для поиска оптимального решения использовали дифференциальную эволюцию (DE), один из эволюционных алгоритмов. Дифференциальная эволюция не требует вычисления градиентов и может быть использована для недифференцируемых функций. Предложенный метод был протестирован на наборе данных CIFAR-10 с использованием трех нейросетей: сверточная сеть (AllConv), сеть в сети (NiN) и VGG16. Их результаты показали, что 70.97% изображений успешно ввели в заблуждение глубокие нейронные сети хотя бы с одним целевым классом. Средняя точность составила 97,47 % [40].

DeepFool Атака DeepFool ставит целью найти наименьшее расстояние от исходного входа до состязательного примера. Для преодоления нелинейности в высшем измерении проводится итеративная атака с линейным приближением. Было обнаружено, что минимальное возмущение аффинного классификатора – это расстояние до разделяющей аффинной гиперплоскости:

$$\mathcal{F} = \{x: w^T x + b = 0\}.$$

Тогда возмущение аффинного классификатора f может быть записано как:

$$\eta^*(x) = -\frac{f(x)}{|w|^2} w.$$

Если f – бинарный дифференцируемый классификатор, то можно использовать итерационный метод для приближения возмущения с учетом того, что f становится более линейным вокруг x_i при каждой итерации. Минимальное возмущение вычисляется как:

$$arg\ min\ |\eta_i|_2:$$
$$f(x_i) + \nabla f(x_i)^T \eta_i = 0.$$

Этот результат также может быть распространен на многоклассовый классификатор путем поиска ближайших гиперплоскостей. Атака также может быть расширена до более общей нормы $l_p, p \in [0, \infty)$. DeepFool обеспечивает меньшее возмущение по сравнению с FGSM и JSMA. По сравнению с JSMA, DeepFool также уменьшает интенсивность возмущений вместо количества выбранных функций [85].

Для проверки эффективности атак была выбрана библиотека keras_vggface. В ней были отобраны 3 обученные модели для распознавания лиц из библиотеки: vgg16, resnet50 и senet50 [90, 91, 92]. Все 3 модели имеют разную архитектуру. vgg16- классическая сверточная сеть с чередованием сверточных и агрегирующих слоев, активационной функцией ReLU в промежуточных слоях и softmax на выходе. resnet50 – более продвинутая архитектура с 50 слоями с весами и пропускными (residual) соединениями (для решения проблемы исчезающего градиента в очень глубоких сетях). Модель senet50 – сеть с применением новой технологии «свертка-развертка». Кроме обычных сверточных слоев здесь имеются слои “fire modules”, которые делают обратную операцию – расширение, за счет этого на выходе получаются большие блоки, за счет чего увеличивается точность распознавания и скорость работы. Для выделения лица на изображении в данной библиотеке используются три сверточных сети: P-Net, R-Net и O-Net, которые позволяют не только выделить область лица, но и его основных частей (нос, рот и т.д.).

Более продвинутые сети быстрее должны генерировать состязательные примеры для атак. С другой стороны, поскольку они с большей точностью распознают лица их должно быть сложнее обмануть. В любом случае успешность атаки должна зависеть от архитектуры сети. Можно выдвинуть гипотезу, что сети с лучшими характеристиками труднее обмануть, а состязательные примеры, сгенерированные более продвинутыми сетями, будут легче обманывать сети с упрощенной архитектурой.

Входом для всех моделей являются изображения лица размером 224×224 с вычитанием среднего изображения лица (вычисленного из обучающего множества) – это критично для стабильности алгоритма оптимизации.

Набор данных содержит 3,31 миллиона изображений 9131 человека, в среднем 362,6 изображения на каждого человека. Изображения загружаются из поисковой системы Google Image Search и имеют большие различия в позе, возрасте, освещении, этнической принадлежности и профессии (например, актеры, спортсмены, политики). Лицо описывается данной моделью как вектор длиной

2048. Затем длина вектора нормируется, например, до длины 1 или единичной нормы с использованием векторной нормы L2 (Euclidean distance from the origin). Это называется «дескриптор лица». Расстояние между дескрипторами лиц (или группами дескрипторов лиц, называемыми «шаблоном субъекта») вычисляется с использованием косинусного сходства [93].

Дескриптор лица извлекается из слоя, смежного со слоем классификатора. Это приводит к дескриптору размерности 2048, который затем нормализуется L2.

Так как входом для всех моделей являются изображения лица размером 224×224 с вычитанием среднего изображения лица, нужно предварительно обрабатывать изображения. Для получения изображения лица используется нейросеть MTCNN. Она состоит из трех отдельных сетей: P-Net, R-Net и O-Net (Рисунок 4.9):

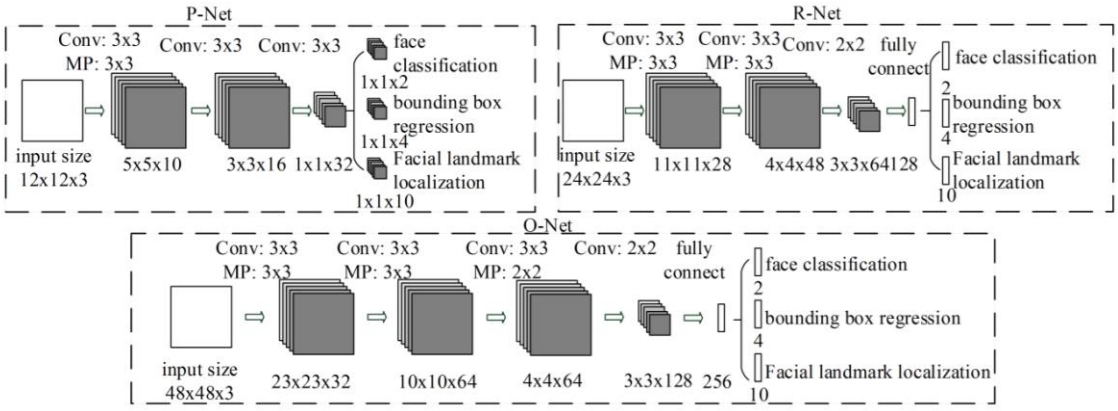


Рисунок 4.9 – Структура MTCNN

Для каждого изображения, которое мы передаем, сеть создает пирамиду изображений, несколько копий этого изображения в разных размерах. В P-сети по каждому масштабируемому образу пробегает ядро 12x12, ищущее лицо. В процессе сканирования, оно смещается в сторону (или вниз) на 1 пиксель, и продолжает делать это до тех пор, пока не пройдёт через всё изображение. Внутри каждого из этих 12x12 ядер выполняется 3 свертки с 3x3 ядрами. После каждого слоя сверток реализуется слой, умножающий каждый отрицательный пиксель на определенное число 'alpha', определяемое в процессе обучения. Кроме того, после этого слоя вставляется агрегирующий слой.

После третьего слоя свертки сеть разбивается на два слоя. Активации из третьего слоя передаются в два отдельных слоя свертки, а слой softmax после одного из этих слоев свертки (softmax присваивает каждому результату десятичные вероятности, а вероятности складываются до 1. В этом случае он выдает 2 вероятности: вероятность того, что в области есть лицо, и вероятность того, что лица нет). Свертка 4-1 выводит вероятность того, что лицо находится в каждом ограничивающем прямоугольнике, а свертка 4-2 выводит координаты ограничивающих полей.

R-Net имеет похожую структуру, но с еще большим количеством слоев. Он берет на вход ограничивающие прямоугольники P-Net и уточняет их координаты. Аналогично, R-Net в конце концов разделяется на два слоя, выдавая два выхода: координаты новых ограничивающих прямоугольников и вероятность нахождения лица в каждом ограничивающих прямоугольников.

Наконец, O-Net принимает в качестве входа ограничивающие прямоугольники R-Net и помечает координаты лицевых опорных точек. Выход O-Net разбивается на 3 слоя в конце, выдавая 3 различных вывода: вероятность нахождения лица в прямоугольнике, координаты ограничивающего прямоугольника и координаты лицевых ориентиров (расположение глаз, носа и рта). Вычитание среднего изображения лица осуществляется при помощи функции preprocess_input из модуля keras_vggface.utils. Так как модели обучены на разных наборах, в качестве параметра необходимо передавать версию модели [91].

Для проверки эффективности различных атак на нейронные сети была написана программа на языке python с использованием библиотек:

tkinter – для создания интерфейса,

numpy – для обработки массивов,

PIL – для формирования изображений,

MTCNN – для определения места расположения лица,

keras_vggface – для распознания лиц,

foolbox – для произведения атак на нейронные сети, а также задания критериев и оптимизируемых расстояний,

time – для измерения времени выполнения атак.

Схематическое описание интерфейса представлено на рисунке 4.10, а сам интерфейс – на рисунке 4.11.

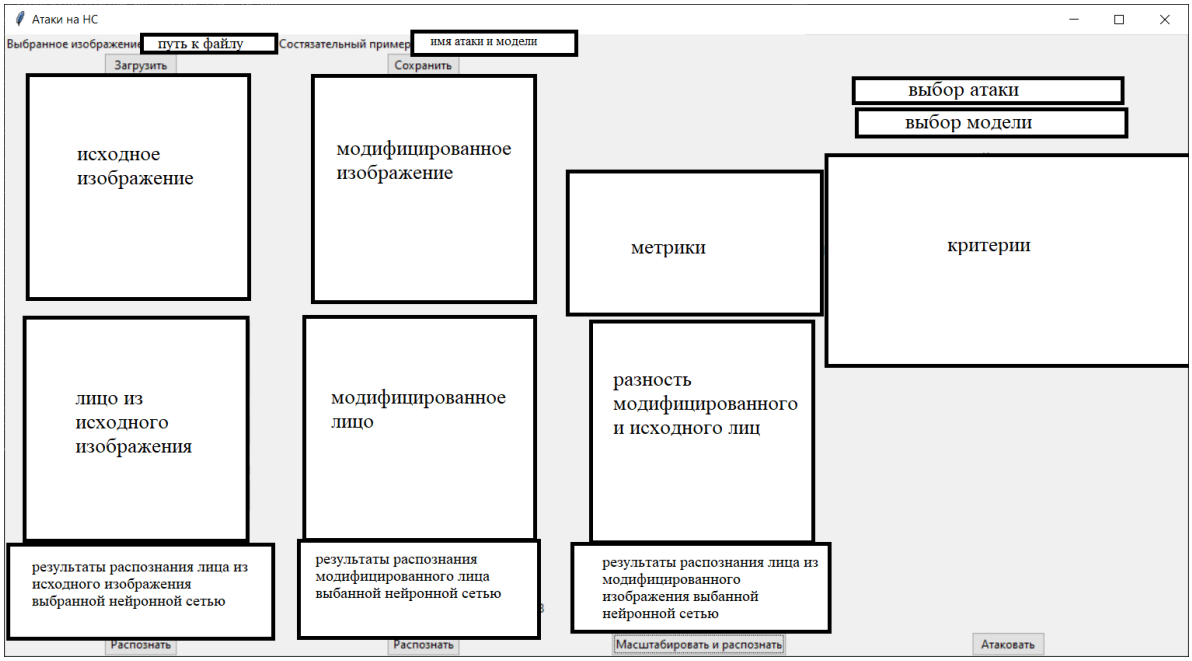


Рисунок 4.10 – Схема интерфейса программы

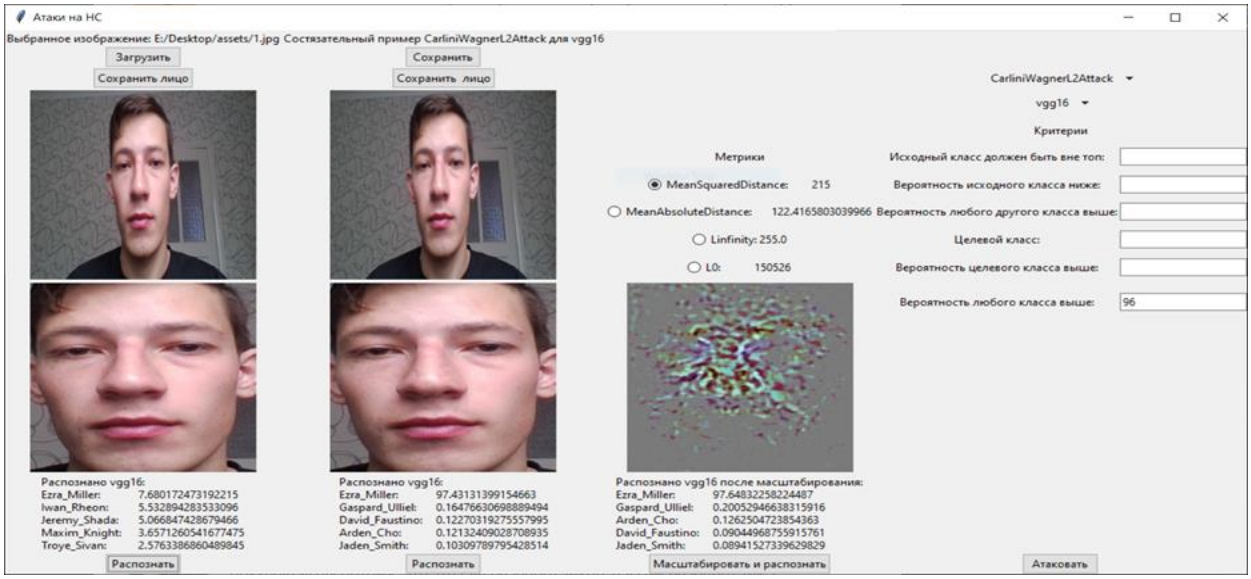


Рисунок 4.11 – Интерфейс программы

Для начала необходимо загрузить изображение человека, воспользовавшись кнопкой загрузить и появившимся диалоговым окном. После чего программа

попытается обнаружить лицо и выведет на экран исходное изображение и изображение лица в отведенные места. Далее можно выбрать модель нейронной сети и нажать кнопку «Распознать», расположенную в одной колонке с исходным изображением. Тогда программа выведет 5 наиболее вероятных варианта ответа и их вероятности, а также имя выбранной модели в поле над кнопкой. После выбора модели можно выбрать атаку из списка, задать ей критерии (можно задавать несколько, также есть критерий по умолчанию) и оптимизируемую метрику в соответствующих полях, и нажать кнопку «Атаковать». После этого справа от изображения лица появится модифицированное лицо (результат атаки), над ним исходное изображение с замененным лицом (которое можно сохранить при помощи кнопки «Сохранить» в формате .bmp, т.к. этот формат не приводит к потерям информации). Правее модифицированного лица будет располагаться изображение попиксельной разности между изображениями лиц, а над ним – значения метрик.

Изображения лиц можно сохранить при помощи кнопок «Сохранить лицо» в соответствующих колонках. Теперь можно использовать кнопку «Распознать» под модифицированным лицом (работает, как и предыдущая кнопка «Распознать», но для модифицированного лица) и кнопку «Масштабировать и распознать» (распознает лицо из модифицированного изображения, результаты могут отличаться от предыдущей кнопки, т.к. при вставке модифицированного лица ему, в большинстве случаев, приходится менять разрешение).

На рисунке 1 приложения Е представлен результат проведения GradientAttack (L-BFGS). Данная атака распределяет возмущение равномерно, т.к. большинство разности пикселей серые. Также видно пятно из цветных пискелей в областях глаз и носа и по контурам лица. Это может добавлять изображению заметные дефекты, как на нижней половине рисунка в виде разноцветных пятен.

Состязательные примеры атаки GradientSignAttack (Рисунок Е.2) отличаются от состязательных примеров предыдущей. Возмущение распределено неравномерно, многие пиксели разности белые. Также в разности можно заметить гораздо большее количество цветных пикселей, т.е. возмущение распределено

неравномерно даже по цветным каналам. Верхняя половина рисунка типична для моделей resnet50 и senet50, а нижняя – для vgg16. В верхней половине из трех базовых цветов наименее представлен зеленый, который лучше всего воспринимается человеческим глазом. В нижней половине основной цвет – синий, который хуже всего воспринимается человеком. Таким образом, данная атака делает состязательный пример менее распознаваемым человеком. Также можно заметить продолжительные черные линии, особенно в нижней половине. Они являются главным недостатком данного метода при рассмотрении человеком, т.к. их можно заметить на представленном состязательном примере.

Состязательные примеры BasicIterativeMethod (Рисунок Е.3) похожи на предыдущие, однако большая часть разности сконцентрирована в областях глаз и рта. Также уменьшено количество черных линий и их продолжительность, благодаря чему их уже практически невозможно заметить на представленных состязательных примерах.

Состязательные примеры CarliniWagnerL2Attack (Рисунок Е.4) похожи на GradientAttack равномерным распределением возмущения, однако пятно с цветными пикселями сконцентрировано в центре лица и дефекты практически неразличимы.

Атака SinglePixelAttack (Рисунок Е.5) изменяет всего один пиксель. Поэтому разность представлена белым квадратом с одним черным пикселем. В нижней половине рисунка показан увеличенный фрагмент состязательного примера, на котором четко виден черный пиксель слева от глаза. Состязательные примеры данной атаки легко различимы при небольшом разрешении, например, 224*224.

DeepFoolAttack (Рисунок Е.6) используется только в ненаправленных атаках. Возмущение распределяется равномерно, как и в GradientAttack и CarliniWagnerL2Attack, однако цветные пиксели распределены равномерней, благодаря чему распознать состязательный пример еще сложнее.

GaussianBlurAttack (Рисунок Е.7) также используется только в ненаправленных атаках. Может быть применена в качестве атаки «черного ящика». Состязательный пример может быть выдан за низкокачественную фотографию.

Разность повторяет изображение лица, т.к. каждый пиксель изменяется в соответствии со значениями соседних, т.е. соседние пиксели и составляют разность.

AdditiveGaussianNoiseAttack (Рисунок Е.8) используется только в атаках, не направленных на конкретный класс. Разность между исходным изображением и состязательным примером для этой атаки равномерна, т.к. представляет собой гауссов шум. На состязательном примере легко просматривается наличие шума, неестественную природу которого практически невозможно определить.

ContrastReductionAttack (Рисунок Е.9) которая также используется только в атаках «черного ящика». Состязательный пример может выглядеть как нормальной изображение, т.к. никому неизвестна исходная контрастность.

GradientAttack, CarliniWagnerL2Attack и DeepFoolAttack меняют практически все пиксели. GradientSignAttack и BasicIterativeMethod изменяют только часть, а SinglePixelAttack – в соответствии с названием, всего один. GaussianBlurAttack, AdditiveGaussianNoiseAttack и ContrastReductionAttack также затрагивают почти все пиксели, однако они работают с ними не просто как с массивом, а как с изображением: размывают, добавляют шум и меняют контрастность соответственно. Из-за этого данные атаки плохо описываются существующими метриками, т.к. они не учитывают перемещения пикселей и равномерность возмущения. К примеру, если увеличить яркость каждого пикселя или сдвинуть изображение в любую сторону, метрики увеличатся, однако это не сделает изображение менее естественным. Также нужно отметить, что несмотря на то, что состязательный пример SinglePixelAttack обладает наилучшими метриками, его проще заметить, чем остальные.

В качестве критериев успешности атак были выбраны:

- критерий по умолчанию, определяемый класс после атаки должен отличаться от исходного;

- указание места, не выше которого должен находиться исходный класс;

- ограничение вероятности исходного класса сверху;

- ограничение вероятности любого класса, кроме исходного снизу;

- ограничение вероятности любого конкретного класса снизугр

- ограничение вероятности любого класса, включая исходный, снизу.

Программой предусмотрено, что перед выполнением атак можно указать оптимизируемую метрику из списка:

– MeanSquaredDistance – среднеквадратичное отклонение;

– MeanAbsoluteDistance – средний модуль отклонения;

– Linfinity – максимальное отклонение;

– L0 – количество измененных элементов (цветов пикселей).

После выполнения атаки все метрики выводятся на экран.

4.3 Методика компьютерных экспериментов

Для тестирования были созданы 3 группы по 5 изображений людей. Первая предназначалась для тестирования эффективности направленных атак и состояла из изображений людей, не использованных для тренировок нейронных сетей. Вторая и третья предназначались для тестирования ненаправленных атак. Вторая состояла из изображений людей, включенных в набор VGGFace (vgg16), а третья – в набор VGGFace2 (resnet50 и senet50). После чего для первой группы выполнялась каждая атака, допускающая данный критерий на каждую модель с критерием «Вероятность любого класса выше», установленным на 1 % выше текущей максимальной вероятности и оптимизацией метрики среднеквадратичного отклонения (т.к. смена оптимизируемой метрики практически не влияет на результат, но увеличила бы время тестирования в 4 раза). После этого атака повторялась с критерием, установленным на 1 % выше результата последней атаки. Так продолжалось до тех пор, пока атака могла выполняться успешно, т.е. завершаться без ошибок и лицо с модифицированного изображения могло быть распознано. При переходе к следующей атаке, критерий сбрасывался. После выполнения каждой атаки в файл записывалось время выполнения атаки (только вызова функции «attack»), метрики, вероятности исходного и полученного классов

(в исходном изображении, после атаки и после масштабирования). На самом деле атака занимала намного больше времени, чем вызов самой функции, т.к. ее требовалось еще загрузить, сконвертировать данные в необходимый для нее формат и выгрузить результат для отображения, а также подсчитать метрики и разность, однако это время хорошо отражает соотношение производительности атак. Если будет нужно провести атаки на множестве предварительно сконвертированных изображениях без подчета метрик, то время поиска каждого состязательного примера будет соответсвовать измеренному в данном эксперимент. Для второй и третьей группы процедура проходила точно так же, только с другим критерием – исходный класс должен был быть вне топ-1.

Если же по каким-то причинам атака завершалась успешно, однако критерий не выполнялся, то она повторялась с увеличенным на 1 значением критерия (максимальным значением было выбрано 16, после чего атака прекращалась).

После этого было проведено тестирование на переносимость атак. Для этого было необходимо, чтобы все нейронные сети могли правильно обработать одно изображение. Поиск совпадающих меток в наборах VGGFace и VGGFace2 не дал результатов, после чего пришлось искать метки, совпадающие только по фамилии. Таким образом, были найдены метки, соответствующие одному человеку: Terrence_Howard и Terence_Howard (в наборе VGGFace метка записана с двумя «r», а в VGGFace2 – с одной). Однако найденную в интернете фотографию по данному запросу обе сети определили правильно, из чего можно сделать вывод, что метки соответствуют одному человеку. Было выбрано изображение, для которого возможно выполнить наибольшее количество атак. Для него были выполнены ненаправленные атаки каждым методом на каждую модель, после которых в файл записывались вероятности исходного класса при распознании каждой нейронной сетью.

Протестировать направленные атаки, таким образом, не удалось, т.к. при критерии «вероятность любого класса выше» атаки повышали вероятность классов, отсутствующих в другом наборе данных. А в случае указания конкретного

класса очень сильно понижалась, что не позволило собрать достаточное количество данных за приемлемое время.

4.4 Результаты тестирования на классификаторах

Результаты тестирования успешности ненаправленных атак приведены в таблице 4.2. Из таблицы 4.2 видно, что наиболее успешными методами атаки являются C&W и GaussianBlur, а наименее – SinglePixel и ContrastReduction. Остальные методы показали примерно одинаковый процент успешности. Также видно, что процент успешности атак практически не зависит от выбранной модели. Поскольку состязательные примеры генерировались теми же сетями, на которые направлена атака (атака белого ящитка), то это не удивительно, далее были проведены эксперименты по переносимости атак, когда состязательные примеры, сгенерированные одной сетью, подавались на другую сеть, в этом случае примеры, сгенерированные более слабой сетью, не могли атаковать более новые и мощные сети.

Немаловажным фактором является скорость выполнения атак. Все эксперименты проводились на ноутбуке с процессором Intel Core i7-7500U, 12 Гб ОЗУ и видеокартой Nvidia GeForce GTX 950M. Результаты занесены в таблицу 4.3. Как видно из таблицы 4.3, GaussianBlur является самой быстрой атакой, а C&W – самой медленной. ContrastReduction показала время 50,07 в своем единственном успешном эксперименте.

Далее для оценки разницы между исходным изображением и состязательным примером замеряны метрики MeanSquaredDistance (среднеквадратичное отклонение), MeanAbsoluteDistance (средний модуль отклонения), и L0 (количество измененных элементов (цветов пикселей)), (Linfinity у всех атак оказалась равна маскимальному значению – 255). Результаты для каждой атаки представлены в таблице 4.4.

Таблица 4.2 – Успешность ненаправленных атак («+» означает, что атака прошла успешно)

	L-BFGS	FGSM	BIM	C&W	Single Pixel	Deep Fool	Gauss. Blur	Gauss. Noise	Contr. Reduct.
vgg16	+	+	+	+		+	+	+	+
							+		
	+	+	+	+		+	+	+	
				+			+		
	+	+	+	+		+	+	+	
resnet50	+	+	+	+		+	+		
				+			+	+	
				+			+		
	+	+	+	+		+	+	+	
senet50	+	+	+	+		+	+	+	
	+	+	+	+		+	+	+	
	+	+	+	+			+		
				+			+		
Результат (%)	53,33	53,33	53,33	80,0	0,0	46,67	86,67	46.67	6,67

Таблица 4.3 – Время выполнения ненаправленных атак (в секундах)

	Gradient	FGSM	BIM	C&W	Deep Fool	Gaussian Blur	Gaussian Noise
1	2	3	4	5	6	7	8
vgg16	1,65	1,89	26,79	243,44	8,14	1,67	44,18
						1,86	
	1,03	1,30	22,49	207,37	8,08	1,50	31,82
				199,93		1,78	
	1,81	1,93	27,41	229,12	10,47	1,80	48,16
resnet50	8,86	9,58	39,77	350,95	29,21	1,85	
				351,05		1,52	42,81
				323,77		1,75	
	9,90	10,71	38,01	321,83	25,96	1,93	59,44
senet50	8,44	7,75	42,28	290,06	22,65	1,54	39,93
	8,26	8,19	41,93	331,10	20,11	1,21	25,15

Продолжение таблицы 4.3

1	2	3	4	5	6	7	8
	11,74	10,42	48,43	397,10		1,76	
				343,20		2,34	
Среднее значение	6,46	6,47	35,89	299,08	17,80	1,73	41,64

Здесь же представлена разность между вероятностью исходного класса до и после масштабирования, поскольку устойчивость к масштабированию имеет значение. Из таблицы 4.3 видно, что GaussianBlurAttack наиболее устойчива к масштабированию.

Таблица 4.4 – Усредненные результаты для ненаправленных атак

	Gradient	FGSM	BIM	C&W	Deep Fool	Gauss Blur	Gauss Noise	Contrast Reduc-tion
Успешность (в проц.)	53,33	53,33	53,33	80,0	46,67	**86,67**	46,67	6,67
Время (в сек.)	6,46	6,47	35,89	**299,08**	17,80	**1,73**	41,64	50,07
Среднеквадратичное расстояние	**58,88**	133,50	168,50	169,25	142,00	210,31	122,86	197
Средний модуль отклонения	**117,98**	127,07	127,13	125,92	130,15	130,66	125,14	172,82
Количество измененных пикселей	150,52	**75,73**	103,09	150,526	150,52	150,53	150,53	150,50
Мас-штаби-рова-ние (раз-ность в проц.	-13,13	-11,80	-26,17	-23,21	-18,18	**0,81**	-15,95	7,98
Пере-носи-мость (в проц.)	69,30	81,90	98,82	97,13	96,56	23,85	46,63	**2,91**

Для проверки атак на переносимость все атаки, кроме C&W и ContrastReductionAttack выполнялись с критерием «исходный класс вне топ-5». C&W и ContrastReductionAttack выдавали ошибку при таком критерии, поэтому для них он был «исходный класс вне топ-1». Остальным атакам также не удавалось убрать исходный класс из топ-5, однако они выполнялись до конца и значительно понижали вероятность исходного класса. SinglePixelAttack выполнить не удалось, несмотря на всю перспективность данной атаки ни один состязательный пример не попал вкласс ниже топ-1. Обобщенные результаты представлены в таблице 4.4 (атака ContrastReduction представлена результатами своего единственного успешного эксперимента).

Таким образом, для ненаправленных атак наиболее успешной является GaussianBlurAttack, если нужна скорость надежность и устойчивость к масштабированию, или GradientAttack, если нужно получить состязательный пример, максимально похожий на оригинал. В случае GradientAttack состязательный пример похож на исходное изображение низкого качества. При этом при попытках его цифровой обработки терялись свойства состязательности. Если же нужно изменить минимальное количество пикселей (в случае неудачи SinglePixelAttack), то лучше всего использовать GradientSignAttack. Если нет доступа к атакуемой нейросети (атака черного ящика), то лучше всего провести ContrastReductionAttack на одной сети, получить состязательный пример и проверить его на другой нейросети. В случае неудачи лучшим выбором будет GaussianBlurAttack.

Аналогичные эксперименты были проведены для направленных атак. Результаты успешности занесены в таблицу 4.5.

Видно, что по затратам времени обработка изображений для направленных атак незначительно отличается от обработки для ненаправленных атак. По-прежнему атака Карлини-Вагнера наиболее затратна по времени, а минимальное время требует градиентный спуск.

Таблица 4.5 – Успешность направленных атак («+» означает, что атака прошла успешно)

	Gradient	FGSM	BIM	C&W	Single Pixel
vgg16	+	+	+	+	
				+	
		+		+	
	+	+	+	+	
				+	
resnet50		+		+	
				+	
	+	+	+	+	
senet50	+	+	+	+	+
	+	+	+	+	+
Результат (%)	33,33	46,67	33,33	66,67	13,33

Замеры времени выполнения приведены в таблице 4.6.

Таблица 4.6 – Время выполнения направленных атак (в секундах)

	Gradient	FGSM	BIM	C&W	Single Pixel
1	2	3	4	5	6
vgg16	2,41	1,63	25,86	215,25	
				233,03	
		1,97		208,26	
	2,49	27,31	27,07	123,02	
				205,14	
resnet50		10,42		279,81	
				342,74	
	9,76	9,81	40,1	294,22	

Продолжение таблицы 4.6

1	2	3	4	5	6
senet50	7,84	8,65	38,15	372,55	79,04
	8,0	9,03	46,69	148,06	76,16
Среднее значение	6,10	9,83	35,57	242,21	77,60

Усредненные результаты для направленных атак представлены в таблице 4.7. Лучшие результаты показала атака C&W, если нужна надежность или высокая вероятность целевого класса. Однако она значительно проигрывает в скорости всем остальным методам. Для получения состязательного примера с лучшими метриками в случае неудачи SinglePixelAttack лучше всего использовать FGSM. Если же нужно получить масштабируемый состязательный пример, то лучше всего подходит GradientAttack или BIM.

Таблица 4.7 – Усредненные результаты для направленных атак

	Gradient	FGSM	BIM	C&W	Single Pixel
Успешность (в проц.)	33,33	46,67	33,33	**66,67**	13,33
Время (в сек.)	22,94	19,57	21,05	**33,20**	7,47
Среднеквадратичное расстояние	**6,10**	**9,83**	35,57	242,21	77,60
Средний модуль отклонения	125,80	121,14	**78,80**	**87,20**	151,00
Количество измененных пикселей	122,58	126,72	126,78	126,60	**0,00**
Масштабирование (разность в проц.)	150,527	79,814	92,286	150,20	**3,00**
Переносимость (в проц.)	**-12,77**	-18,06	**-14,37**	-19,79	-28,58

Исследование было бы неполным, если бы ограничилось лишь разработанной программой. Сгенерированне состязательные примеры были проверены на известных сторонних поисковых системах.

4.5 Результаты тестирования на поисковых АСНИ БЛ

Для исследования распознавания фальсифицированных изображений сторонними ресурсами были выбраны поисковые системы Bing, Google и Yandex. Для тестирования направленной атаки было выбрано изображение с лучшей вероятностью распознавания состязательного примера (результат атаки C&W на vgg16). Для тестирования ненаправленных атак было выбрано первое изображение из набора vgg16 (Erza Miller), т.к. на нем удалось выполнить почти все атаки. В эксперименте использовались только изображения лица, чтобы алгоритмы поисковых систем не обнаруживали похожие изображения по фону.

Направленная атака не принесла успеха. Состязательный пример определился просто как человек, т.е. ввести в заблуждение алгоритмы не удалось (Рисунок 4.12).

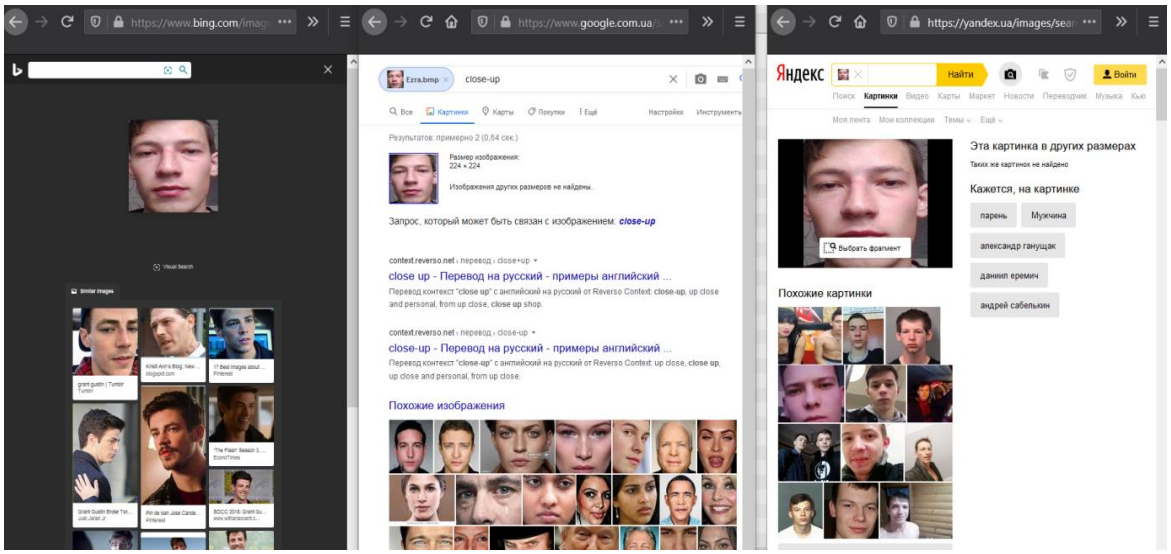


Рисунок 4.12 – Результат направленной атаки в Bing, Google и Yandex

Атака, не напраленная на определенный целевой класс была более успешной. Критерий – исходный класс вне топ-5. То есть нейронная сеть модифицировала изображение лица так, чтобы правильный результат не входил в 5 распознанных классов. При этом не устанавливался процент успешности, с которым она должна распознать лицо.

Результаты поиска по оригинальному изображению приведены на рисунке 4.13. Все 3 поисковика правильно определили человека на фотографии.

В качестве примера результат поиска по состязательным примерам, созданным алгоритмом AdditiveGaussianNoiseAttack показан на рисунке 4.14. Bing определил правильно, Google заметил только улыбку, а Яндекс распознал как Виктора Цоя. То есть при добавлении специально сгенерированного шума можно заставить нейронную сеть как минимум не опознать человека. Причем при добавлении случайного шума или измениния яркости изображение в большинстве случаев распознвется правильно.

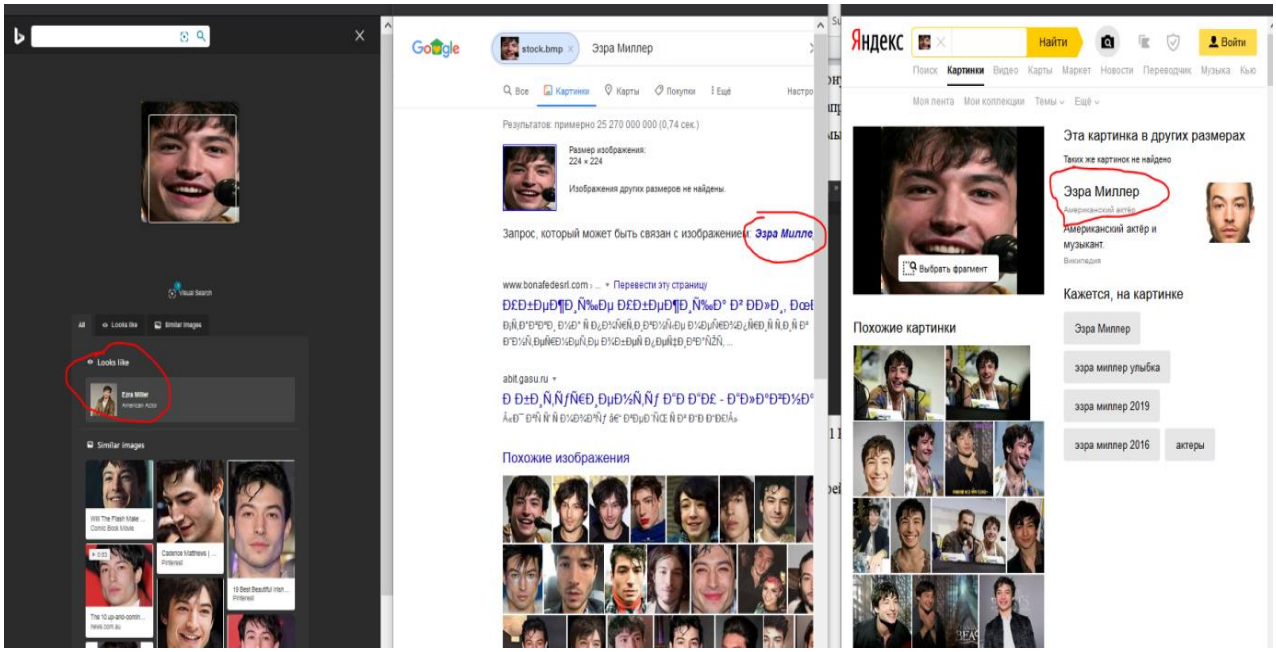


Рисунок 4.13 – Поиск по исходному изображению уверенно распознает Эрза Миллер

На рисунке 4.15 представлены результаты распознавания исходного изображения с поднятой на 50 % яркостью. Bing и Яндекс распознали изображение правильно, а Google обнаружил на картинке всего лишь «близкий план».

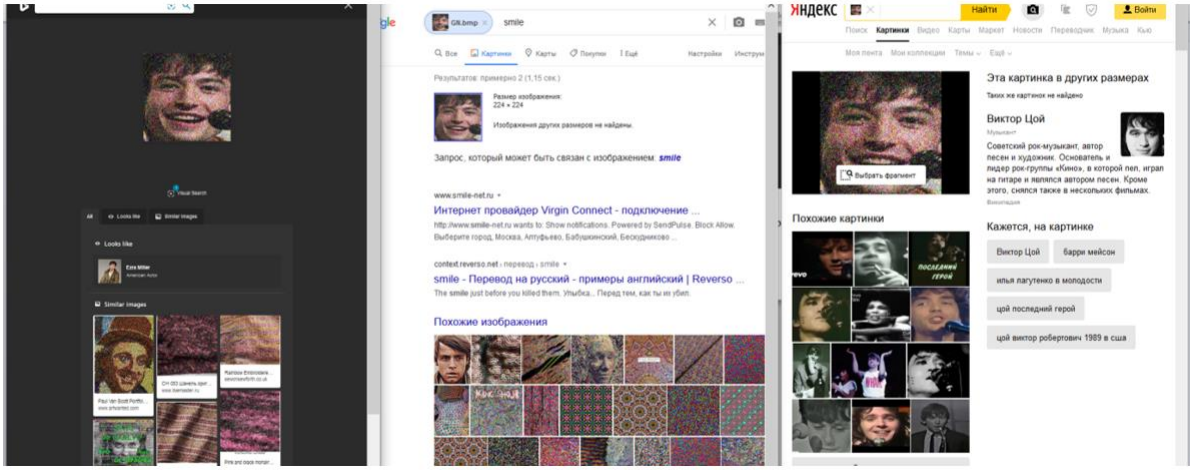


Рисунок 4.14 – Поиск по состязательному примеру AdditiveGaussianNoiseAttack, Эрза Миллер распознается только поисковой системой Bing

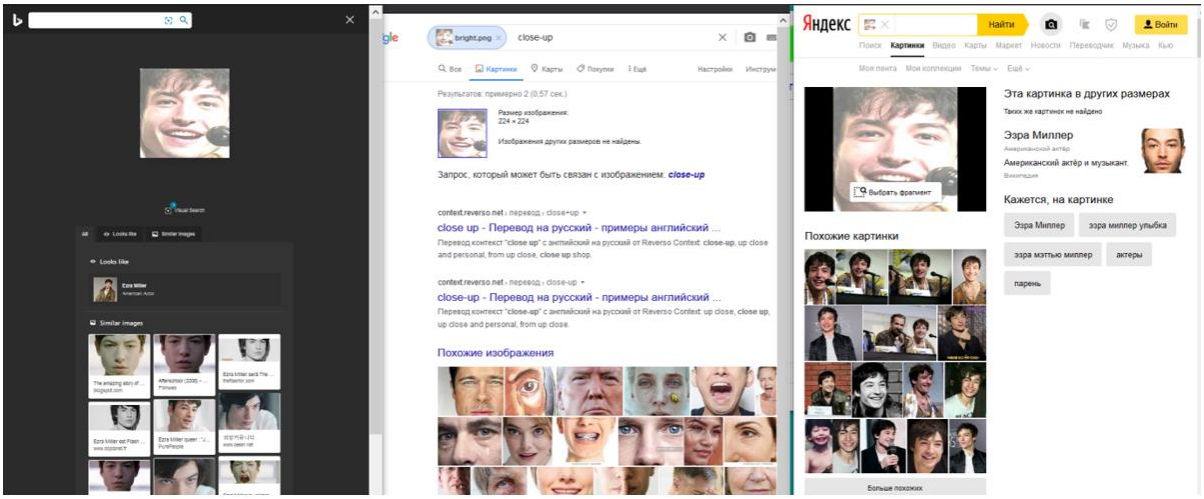


Рисунок 4.15 – Поиск по изображению с измененной на 50 % яркостью

Результаты тестирования сгенерированных состязательных примеров в поисковых сервисах приведены в таблице 4.8.

Как видно из таблицы не все методы генерирования состязательных примеров позволили ввести в заблуждение сторонние системы. Нейросетевые алгоритмы поисковой системы Bing наиболее устойчивы к атакам, а алгоритмы google можно ввести в заблуждение простым изменением яркости. Также видно,

что методы, показавшие лучшую переносимость на другие нейронные сети, продемонстрировали лучшие результаты и на нейронных сетях поисковых систем.

Таблица 4.8 – Результаты тестирования в автоматизированных поисковых сервисах

	Grad.	FGSM	BIM	C&W	Deep Fool	Gauss Blur	Gauss Noise	Contr. Reduction	Пов. ярк.
Bing								+	
Google						+	+	+	+
Yandex						+	+	+	

4.6 Методы защиты от угроз для нейросетевых алгоритмов

Рассмотрев результаты, можно сделать вывод, что направленную атаку на нейронную сеть выполнить очень сложно. Оригинальные изображения все 3 тестируемые сети определяли на 98-100 % соответствующими своей метке. К этому результату удалось приблизиться состязательному примеру C&W лишь в одном эксперименте. Однако состязательные примеры данной атаки показали худшие результаты в масштабировании, поэтому при смене разрешения изображения в процессе предварительной обработки они теряют свои свойства. Т.е. в реальных условиях даже при доступе к выходному слою нейросети и при возможности указать целевой класс любым, а не каким-то конкретным, атаку выполнить не удалось.

Ненаправленные атаки были намного успешнее, в большинстве случаев им удавалось ввести в заблуждение нейросеть. Этот вид атак менее опасен, чем направленные, но проблема остается актуальной.

Наиболее универсальным способом защиты является использование еще одной нейронной сети, обученной обнаруживать состязательные примеры, однако

этот способ требует повышения вычислительной сложности. Также в будущем возможно появление состязательных примеров и для этой нейронной сети.

Рассмотрим меры, которые можно принять, чтобы минимизировать риск описанных атак. Для успешной защиты нужно знать слабые места каждой атаки. SinglePixelAttack наиболее сложна в реализации и хуже остальных переносит масштабирование. Также можно считать отдельно четные и нечетные пиксели и отдельно распознавать их. Если результат будет сильно отличаться от оригинала, то очевидно, что изображение является состязательным примером.

ContrastReductionAttack просто понижает контрастность, поэтому перед обработкой изображений можно повышать их контрастность. К примеру, онлайн сервис https://www.imgonline.com.ua/auto-contrast.php неплохо справился с восстановлением изображения из состязательного примера.

BIM, C&W и DeepFool плохо переносятся на другие сети, поэтому для защиты от них рекомендуется уменьшить обратную связь с атакующим. К примеру, установить ограничение на количество попыток входа или защиту от роботов, типа капчи. Также нужно сообщать владельцу о блокированных попытках входа.

Если нейросеть является частью антивируса, то не следует предоставлять пользователю доступ к выходному слою, а выводить только класс угрозы без вероятности (именно так и происхдит, например, для антивируса Касперского).

GaussianBlurAttack и AdditiveGaussianNoiseAttack слишком сильно изменяют изображение и могут быть легко обнаружены по внешнему виду. Чтобы убедиться в этом, сформируем изображение для поиска дефектов из исходного и состязательных примеров (Рисунок 4.16).

Рисунок 4.16 – Изображение, состоящее из исходного и состязательных примеров (слева-направо): GradientAttack, FGSM, GaussianBlurAttack и AdditiveGaussianNoiseAttack

В первой строчке состязательные примеры для vgg16, во второй – для resnet50, и в третьей – для senet50. Загрузим это изображение в инструмент Forensically, для анализа изображений и воспользуемся вкладками ErrorLevelAnalysis (Рисунок 4.17) и LuminanceGradient (Рисунок 4.18). Видно, что состязательные примеры AdditiveGaussianNoiseAttack содержат высокий уровень шума, а GaussianBlurAttack – наоборот, низкий.

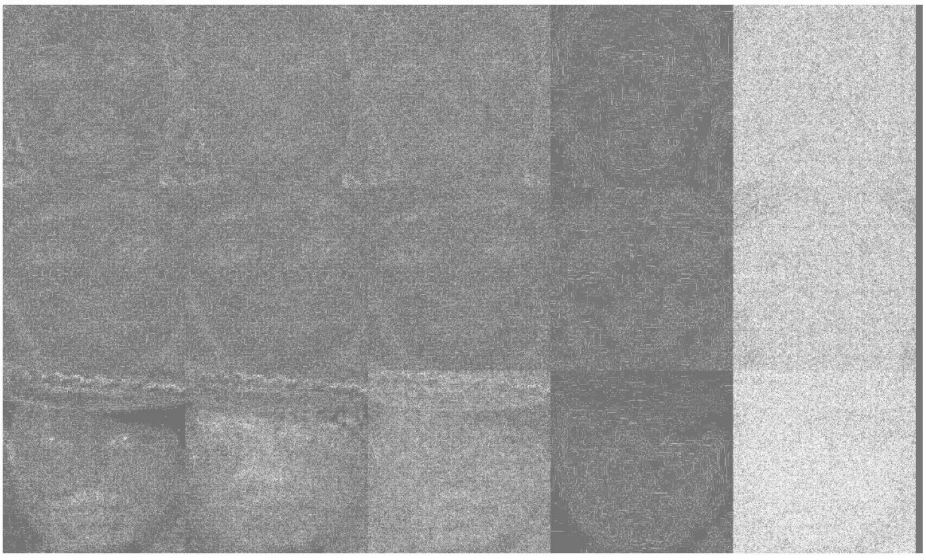


Рисунок 4.17 – Инструмент Error Level Analyzer

По средней яркости этих изображений уже можно обнаружить состязательные примеры. Также видно, что состязательные примеры GradientAttack и FGSM тоже повышают уровень яркости.

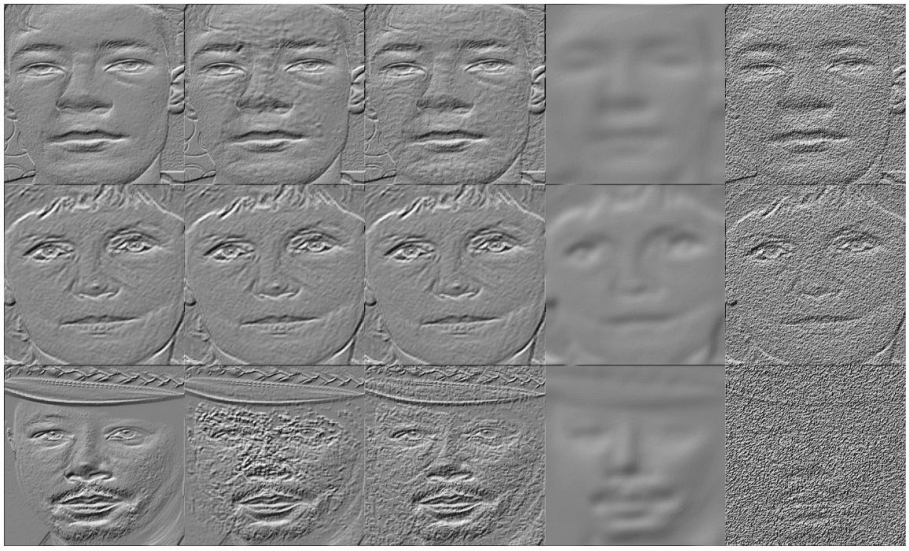


Рисунок 4.18 – Инструмент Luminance Gradient

Инструмент LuminanceGradient показывает отличия состязательных примеров от исходного изображения в областях глаз и носа для GradientAttack и по всему лицу для FGSM. Таким образом, данные атаки могут быть обнаружены уже существующими алгоритмами. GradientAttack и FGSM также могут быть замечены благодаря обработке изображений несколькими нейросетями или с разной предварительной обработкой (например, масштабирование) с последующим сравнением результатов.

4.7 Выводы по разделу 4

Для реализации атак на модели vgg16, resnet50 и senet50 разработано приложение с пользовательским интерфейсом, позволяющее проводить как целевые, так и нецелевые атаки с заданием значения точности распознавания, выбором и контролем оптимизируемой метрики. Приложение позволяет менять критерии успешности атак и предоставляет возможность записи времени проведения атак в файл. С помощью данного приложения проведено исследование

9-ти методов генерации состязательных примеров для ненаправленных атак и направленных атак. Сгенерированные в результате направленных и ненаправленных атак состязательные изображения предъявлялись поисковым системам Bing, Google и Яндекс для распознания.

По результатам исследования сделаны следующие выводы:

1. Для ненаправленных атак лучше всего подходят атаки GaussianBlurAttack, если нужна скорость надежность и устойчивость к масштабированию, или GradientAttack, если нужно получить состязательный пример, максимально похожий на оригинал. Если же нужно изменить минимальное количество пикселей (в случае неудачи SinglePixelAttack), то лучше всего использовать GradientSignAttack.

2. Для направленных атак лучше всего подходит C&W, если нужна надежность или высокая вероятность целевого класса. Однако данная атака значительно проигрывает в скорости всем остальным методам. Для получения состязательного примера с лучшими метриками в случае неудачи SinglePixelAttack лучше всего использовать FGSM. Если же нужно получить масштабируемый состязательный пример, то лучше всего взять GradientAttack или BIM.

3. Модели улучшенной архитектуры, например, senet50, наиболее устойчивы к состязательным примерам GaussianBlurAttack, AdditiveGaussianNoiseAttack и ContrastReductionAttack сгенерированных нейронными сетями более простой архитектуры.

4. Поисковые системы Bing, Google и Yandex устойчивы к направленным атакам типа «черный ящик». Алгоритмы Bing наиболее устойчивы к ненаправленным атакам, а алгоритмы Google и Yandex можно обмануть простым изменением яркости. Состязательные примеры, сгенерированные методами, не основанными на градиенте, лучше вводят в заблуждение поисковые системы, однако данные методы предназначены только для ненаправленных атак.

5. Ненаправленные атаки в большинстве случаев осуществляются легче и успешнее, чем направленные.

6. Предложены методы защиты от атак на нейросетевые алгоритмы, на основе анализа полученных результатов по каждой атаке, в частности подсчет четных и нечетных значений яркости пикселей, обучение на состязательных примерах, повышение контрастности и фильтрация, уменьшение обратной связи с атакующим, но ни один из предложенных методов не является надежным и универсальным.

7. Поскольку распознавание изображений находит все более широкое применение в автоматизированных системах аутентификации и безопасности, разработчикам стоит тестировать системы на возможность подобного «обмана» со стороны злоумышленников. Необходимо держать в секрете архитектуру используемых нейронных сетей, поскольку, как показали эксперименты, злоумышленник может сгенерировать фальшивые изображения, используя аналогичную модель. При накладывании фальсифицирующей заплатки наличие знаний о структуре модели необязательны, однако в этом случае процесс верификации не должен быть полностью автоматизирован, поскольку заплатка в отличие от машинного зрения, хорошо различима человеческим. Целесообразно дополнять автоматизированные системы аутентификации инструментами поиска подобных уязвимостей и предусматривать защитные меры.

ЗАКЛЮЧЕНИЕ

В диссертационной работе дано решение актуальной научно-технической задачи совершенствования АСНИ безопасности личности на основе искусственных нейронных сетей путем улучшения технологий обработки данных, что позволяет повысить быстродействие систем и степень достоверности принятия решений, а также снизить уровень уязвимости нейросетевых алгоритмов. Основные результаты работы состоят в следующем.

1. Разработан метод сокращения количества параметров нейросетевых алгоритмов редукцией проигравших нейронов, который позволяет снизить количество параметров нейронных сетей до 30 % и повышает скорость работы и ресурсоемкость программных решений.

2. Разработаны архитектурные решения и программная реализация двух систем обработки формализованных данных (для анализа файлов логов сервера и обнаружения радиоканалов утечки информации) и четырех систем обработки изображений (определение формы предмета, распознавание по отпечатку пальца, по лицу, распознавание опасных предметов).

3. Исследованы уязвимости АСНИ БЛ на основе нейросетевых алгоритмов путем создания состязательных примеров и проведения 9-ти типов направленных и ненаправленных атак на классификаторы изображений, в том числе поисковики Bing, Google, Yandex. Предложены методы защиты от подобных атак.

4. Анализ полученных экспериментальных данных подтверждает возможность внедрения нейросетевых технологий в автономные автоматизированные системы другого назначения, например, промышленного и широкого назначения за счет уменьшения ресурсоемкости разработанных решений без понижения эффективности и качества.

5. Методика аутентификации пользователей и программное приложение для распознавания лиц на основе нейронной сверточной сети, методика проверки цифровых документов на подлинность, компьютерное приложение для защиты цифровых документов от редактирования внедрены в учебный процесс кафедры

радиофизики и инфокоммуникационных технологий путем использования в лекционных курсах и лабораторных практикумах по дисциплинам «Пакеты прикладных программ для обработки изображений», «Нейронные сети», «Основы информационной безопасности».

6. Результаты диссертационных исследований внедрены в систему аутентификации и доступа на базе монитора видеодомофона М-480М с системой на кристалле Allwiner A-13. Достигнута точность распознавания 89 % и скорость распознавания 0,5 с/лицо. при записи 5-6 лиц в базу данных, 4,5 с/лицо при записи 150 лиц в базу данных, что позволило увеличить скорость работы и уменьшить ресурсоемкость системы.

ПЕРЕЧЕНЬ УСЛОВНЫХ СОКРАЩЕНИЙ

АСНИ БЛ – автоматизированные системы научных исследований безопасности личности

LVQ – Learning Vector Quantization

RBF – Radial Basis Function

GRNN – General Regression Neural Network

FC –Full Connection

BSD – Berkeley Software Distribution

TAR – True Accept Rate

TRR – True Reject Rate

FRR – False Reject Rate

FAR – False Accept Rate

SGD – Stochastic Gradient Descent

FGSM – Fake Gradient Stochastic Momentum

СПИСОК ЛИТЕРАТУРЫ

1. Hertz, J. Introduction to the theory of neural computation / J. Hertz, A. Krogh, R. G. Palmer // Lecture Notes Volume I the Santa FE Institute studies in the sciences of complexity. - [Электронный ресурс] ResearchGate 2001. – Режим доступа: https://www.researchgate.net/publication/200033871_Introduction_To_The_Theory_Of_Neural_Computation, свободный.

2. Твилдиани, Г. М. Особенности аппроксимации алгебраических функций нейронными сетями персептронного и RBF типов/ Г. М. Твилдиани // Вест. студ. науч. общ. ГОУ ВПО «Донецкий национальный университет». — 2017. — в. 9, т 1, Естественные и технические науки. — С-159-164

3. Филатова, Т. В. Применение нейронных сетей для аппроксимации данных/Т. В. Филатова // Вест. Том. гос. ун-та. — 2004. — № 284. — С. 121–125.

4. Павлова, А. И., Лончакова Сравнительный анализ применения нейронных сетей для аппроксимации функцией/ А. И. Павлова, О. Ю. Лончакова // ScienceTime. — 2015. — №5 (17). —. С 118–121.

5. Jorgensen, M. The Use of Precision of Software Development Effort Estimates to Communicate Uncertainty/ M. Jorgensen // Software Quality Days. TheFutureofSystemsandSoftwareDevelopment. Springer. — 2016. — pp. 156–168.

6. Kingma, J. Methods for Stochastic Optimization. / J. Kingma, D. P. Ba // Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego. — 2017.

7. Боровиков М Нейронные сети Statistica neural Networks/ М. Боровиков. М.: Телеком. — 2008. – 392 с.

8. Калацкая, Л.В. Организация и обучение искусственных нейронных сетей /Авт.-сост. Л. В. Калацкая, В. А. Новиков, В. С. Садков. Мн.: БГУ. — 2003. – 321 с.

9. Медведев, В. С. Нейронные сети/В. С Медведев., В. Г. Потемкин. MATLAB 6. М.: Диалог – МИФИ. — 2002. – 496 с.

10. Нейросеть на Python: пошаговое руководство [Электронныйресурс] сайт. — Режим доступа:https://proglib.io/p/neural-nets-guide/, свободный.

11. Жвакина, А. В. Искусственные нейронные сети в медицинской диагностике / А. В. Жвакина // 54-я научная конференция БГУИР. —. 2018. – С.182–183.

12. Москалев, Н. С. Виды архитектур нейронных сетей / Н. С. Москалев // Молодой ученый. — 2016. — №29. — С. 30-34.

13. Wang, Y. Fuzzy clustering analysis/ Y. Wang// ICIC Express letters. — 2012. —Vol. 2, №4. — pp. 331-337.

14. Спицын, В. Г. Алгоритмы для классификации отпечатков пальцев на основе применения фильтра габора, вейвлет/преобразования и многослойной нейронной сети / В. Г. Спицын, Фан НгокХоанг // Известия Томского политехнического университета. – 2012. – Т. 320. – № 5. – С. 60–61.

15. Глушков, С.В. Метод идентификации морских объектов / С. В. Глушков, Н. Н. Жеретинцева, Ю. В. Шемчук // сборник трудов ДВТУ, 2007. –Вып. №146. – С. 221-225.

16. Горбунов, А. А. Применение глубоких нейронных сетей для классификации больших объемов астрономических данных /А. А. Горбунов, Е. А. Исаев, В. А. Самодуров// Радиофизика и радиоастрономия. — 2017. –Т. 22, № 4, – С. 270–275.

17. Воробьев, Е. В. Классификация текстов с помощью сверточных нейронных сетей / Е. В. Воробьев, Е. В. Пучков // Молодой исследователь Дона. — 2017, №6 (9), – С.2-7.

18. Сердюков, В. И. Использование элементов искусственного интеллекта для повышения надежности технических изделий / В. И. Сердюков, Н. А. Сердюкова, С. И. Шишкина // Вестник машиностроения. — 2017. — № 10. — С. 29-32.

19. Колесников, А. А. Использование технологии машинного обучения при решении геоинформационных задач / А.А. Колесников, П. М. Кикин, Е. В. Комиссарова, Касьянова Е.Л. // Интеркарто — 2018.— Т. 24, №2— С. 371-384.

20. Телипенко, Е. В. Оценка риска банкротства предприятия на основе нейросетевых технологий/ Е. В. Телипенко, М. Р. Яворский // Экономика и предпринимательство. — 2014. - № 7. - С. 509-514.

21. Казачков, Е. А. Обнаружение и классификация малоразмерных объектов на изображениях, полученных радиолокационными станциями с синтезированной апертурой / Е. А. Казачков, С. Н. Матюгин, И. В. Попов, В. В. Шаронов. ‑ Вестник Концерна ВКО «Алмаз. – Антей». — 2018. —№ 1 — с. 93-99.

22. Tajbakhsh, N. Computer-Aided Pulmonary Embolism Detection Using a Novel Vessel-Aligned Multi-planar Image Representation and Convolutional Neural Networks/ N. Tajbakhsh, M. B. Gotway, J. Liang.// Proceedings of 18th International Conference on Medical Image Computing and Computer-Assisted Intervention – MICCAI — 2015, Munich, Germany, October 5-9, —Volume 2, — pp. 62-69.

23. Kovshov, E. Automated remote computer monitoring of environmentally hazardous products. / E. Kovshov, A. Kosach In Proc. The 14th International Conference Application of Contemporary Non-Destructive Testing in Engineering — 2017. — pp. 95-102.

24. Сорокин, В. Е. Применение аппарата нейронных сетей для решения первичной задачи адаптивной компенсации искажений GPS сигналов/ В. Е. Сорокин // Сборник научных трудов, ДонИЖТ. — № 40 . — Донецк: ДонИЖТ. — 2016. — С. 18-26.

25. Ежов, А. А. Сеть Хемминга и ее применение для решения задачи распознавания подписей. /А. А. Ежов, А. С. Новиков // Известия ТулГУ, Технические науки. —2016. —вып 11.2.1. — С-56-63

26. Nielsen, M. Neural Networks and Deep Learning / M. Nielsen [Электронный ресурс] Online book Neural Networks and Deep Learning, 2017. — Режим доступа: URL:http://neuralnetworksanddeeplearning.com/index.html

27. Гонсалес, Р. Цифровая обработка изображений. / Р. Гонсалес, Р. М. Вудс. – М: Техносфера, 2012. – 1104 с.

28. Крысова, И. В. Методы распознавания графических образов для решения задач автоматизированного проектирования / И. В. Крысова, И. Л. Чулкова. — Вестник СибАДИ. — 2013. — выпуск 5 (33) —С. 234-240.

29. Брагин, А. В. Способ распознавания формы объектов доменных структур магнитооптических материалов на базе нейронных сетей прямого распространения / А. В. Брагин, М. В. Герасимов, Р. Р. Навлётов, Д. В. Пьянзин [Электронный ресурс]. Журнал электроники ISSN 1684-1719, N10, 2018 — Режим доступа: https://www.Electronics_(magazine)/index.php?id=1684-1719, свободный.

30. Dosovitskiy, A. Learning to generate chairs with convolutional neural networks / A.Dosovitskiy, J.T.Springenberg, and T.Brox,// CVPR, 2015, — pp. 1538–1546. [Электронный ресурс] arxiv.org/abs/1411.5928. — Режим доступа: https://arxiv.org/abs/1411.5928, свободный.

31. Roy, A. Evolutionary methods for generating synthetic masterprinttemplates: Dictionary attack in fingerprint recognition / A. Roy, N. Memon, J. Togelius, and A. Ross. In International Conferenceon Biometrics, pages 1–8, 2018 [Электронный ресурс]. arxiv.org/abs/1705.07386 — Режим доступа: https://arxiv.org/abs/1705.07386 , свободный.

32. Русакова, А. А. Оценка эффективности работы сети Хопфилда в задаче распознавания образов. [Электронный ресурс]. Научная электронная библиотека «киберленинка» 2017 — Режим доступа: https://cyberleninka.ru/article/n/otsenka-effektivnosti-raboty-seti-hopfilda-v-zadache-raspoznavaniya-obrazov, свободный.

33. Руденко, О. В. Нейросетевоераспознавание в технических системах зерноперерабатывающей и пищевой промышленности / О. В.Руденко, С. В. Усатиков // Современные проблемы науки и образования. 2011. № 3 [Электронный ресурс]. Электронный журнал – Режим доступа: https://www.science-education.ru/ru/article/view?id=4668, свободный.

34. Kovaci, K. Computer Vision Systems in Road Vehicles: A Review/ K Kovaci E Ivanjko H Gold, [Электронный ресурс]. — arXiv preprint arXiv: 1310.0315, 2013. — Режим доступа: https://arxiv.org/ftp/arxiv/papers/ 1310/1310.0315.pdf, свободный.

35. Mishra, S Deep 3D Convolutional Neural Network for Automated Lung Cancer Diagnosis / S. Mishra, Naresh K. Chaudhary, P. Asthana, A. Kumar , [Электронный ресурс] arXiv preprint arXiv: 1906.01054, 2019. — Режим доступа: https://arxiv.org/abs/ 1906.01054, свободный.

36. Papernot, N. Practical black-box attacks against machine learning/ N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, A. Swami in Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security. ACM, 2017, pp. 506–519. [Электронный ресурс].- Режим доступа: https://arxiv.org/abs/1602.02697/, свободный.

37. Baluja, S. Adversarial transformation networks: Learning to generate adversarial examples/ S. Baluja, and I. Fischer arXiv preprint arXiv: 1703.09387, 2017. [Электронный ресурс].- Режим доступа: https://www.arxiv-https://www.arxiv-vanity.com/papers/1703.09387/, свободный.

38. Szegedy, C. Intriguing properties of neural networks/ C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus arXiv preprint arXiv:1312.6199, 2013. [Электронный ресурс].- Режим доступа: https://www.arxiv-vanity.com/papers/1312.6199/, свободный.

39. Goodfellow I. J. Explaining and harnessing adversarial examples/ I. J. Goodfellow, J. Shlens, and C. Szegedy, arXiv preprint arXiv: 1412.6572, 2014. [Электронный ресурс].- Режим доступа: https://arxiv.org/abs/1412.6572, свободный.

40. Su, J. One pixel attack for fooling deepneural networks / J. Su, D. V. Vargas, and S. Kouichi, [Электронныйресурс] arXiv preprint arXiv: 1710.08864, 2017. — Режим доступа: https://arxiv.org/abs/ 1710.08864, свободный.

41. Papernot, N. The limitations of deep learning in adversarial settings, in Security and Privacy / N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, (EuroS&P), 2016 IEEE European Symposium on. IEEE, 2016, pp. 372–387.

[Электронный ресурс]. — Режим доступа: https://arxiv.org/abs/1511.07528, свободный.

42. Messer, K. Xm2vtsdb: The extended m2vts database/ K. Messer, J. Matas, J. Kittler, J. Luettin, and G. Maitre, Second international conference on audio and video-based biometric person authentication. – vol. 964. – 1999. – pp. 965–966. – [Электронный ресурс]. — Режим доступа: http:// viewdoc/summary ?doi= 10.1.1.35.468, свободный.

43. Li, Hao Pruning filters for efficient convnets/ Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, Hans Peter Graf. arXiv preprint arXiv:1608.08710, 2016. [Электронный ресурс]. — Режим доступа: https://arxiv.org/abs/1608.08710, свободный.

44. Liu, Zhuang Rethinking the value of network pruning/ Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, Trevor Darrell. In International Conference on Learning Representations, 2019. [Электронный ресурс]. — Режим доступа: URL: https://openreview.net/forum?id=rJlnB3C5Ym, свободный.

45. Baldi, Pierre Understanding dropout/ Pierre Baldi and Peter J Sadowski In Advances in neural information processing systems, pp. 2814–2822, 2013. [Электронный ресурс]. — Режим доступа: https://papers.nips.cc/paper/4878-understanding-dropout, свободный.

46. Molchanov D. , Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks/ D. Molchanov D, A. Ashukha, D. Vetrov. arXiv preprint arXiv:1701.05369, 2017. [Электронный ресурс]. . — Режим доступа: https://arxiv.org/abs/1701.05369, свободный.

47. Tien-Ju, Yang Designing energy-efficient convolutional neural networks using energy-aware pruning/ Yang Tien-Ju, Chen Yu-Hsin, and Vivienne Sze arXiv preprint, 2017. [Электронный ресурс]. — Режим доступа: https://arxiv.org/abs/ 1611.05128, свободный.

48. Li, Wan Regularization of neural networks using dropconnect/ W. Li, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus. In International Conference on Machine Learning,

pp. 1058–1066, 2018. [Электронный ресурс].- Режим доступа: http://proceedings.mlr.press/v28/wan13.html, свободный.

49. Molchanov, P. Pruning convolutional neural networks for resource efficient transfer learning./ P. Molchanov, S. Tyree, T. Karras, T. Aila, J. Kautz. arXiv preprint arXiv:1611.06440, 2016. [Электронный ресурс].- Режим доступа: https://arxiv.org/abs/1611.06440, свободный.

50. Simonyan, K. Very deep convolutional networks for large-scale image recognition/K. Simonyan, A. Zisserman arXiv preprint arXiv: 1409.1556, 2014. [Электронный ресурс].- Режим доступа: https://arxiv.org/abs/1409.1556, свободный.

51. He, Kaiming Deep residual learning for image recognition/ Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun// In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016. [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1512.03385, свободный.

52. Scikit-learn. MachineLearninginPython [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/stable/, свободный.

53. KDDCup 1999 Data [Электронный ресурс]. - Режим доступа: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, свободный.

54. Бабичева, М. В. Автоматизация средств обнаружения радиоканалов утечки информации / М. В. Бабичева, А. С. Попов, А. В. Яновский // Вестник Донецкого национального университета. Серия Г: Технические науки. – 2020. – № 1. – С. 9-13.

55. Денисенко, А. Н. Применение различных методов восстановления непрерывных сигналов по их дискретным значениям /А. Н. Денисенко, В. Н. Исаков // Радиотехника. 2001. – № 10. – С. 16-20.

56. Crochiere, R E. Interpolation and decimation of digital signals: a tutorial review / R. E. Crochiere, L. R. Rabiner// Proceedings of the IEEE. 2008. – Vol. 69. – No. 3. – Pp. 300-331.

57. Schafer, R. W. A digital signal processing approach to interpolation / R. W Schafer, L. R Rabiner // Proceed-ingsofthe IEEE. 2003. – Vol. 61. – No. 6. – Pp. 692-702.

58. Рыканов, А. С. Анализ методов распознания отпечатков пальца / А.С. Рыканов // Системы обработки информации. – 2010. – № 6. – с. 164-171

59. Список функций Image Processing Toolbox [Электронный ресурс]. – Режим доступа: http://matlab.exponenta.ru/imageprocess/book3/index.php, свободный.

60. Янковский, А. А. Критерии выбора метода бинаризации при обработке изображений / А. А. Янковский, А. Н. Бугрий – 2010. – с. 53-56

61. Бабичева, М. В. Итеративный алгоритм пороговой бинаризации для обработки отпечатков пальцев в биометрических системах доступа / М. В. Бабичева, А. С. Юрченко // Вестник Донецкого национального университета Серия Г. Технические науки, 2018. – № 3. – С. 41-46.

62. Гудков, В. Ю. Скелетизация бинарных изображений и выделение особых точек для распознавания отпечатков пальцев/ В. Ю. Гудков, Д. А. Клюев // Компьютерные технологии, управление, радиоэлектроника – 2015. – № 3. – с. 11-17

63. Xia, Gui-Song Accurate Junction Detection and Characterization in Natural Images./ Gui-Song Xia, Julie Delon, Yann Gousseau // International Journal of Computer Vision (IJCV), 2014. –106 (1). –pp.31-56.

64. Бабичева, М. В. Выделение особых точек отпечатков пальцев детекторами углов /М. В. Бабичева, А. С. Юрченко // Вестник Донецкого национального университета Серия Г. Технические науки, 2019. – № 2. – С. 10-15.

65. Кухарев, Г. А. Биометрические системы: Методы и средства идентификации личности человека / Г. А. Кухарев – СПб.: Политехника. – 2001. – 240 с.

66. Бабичева, М. В. Нейронные сети в системах для научных исследований / М. В. Бабичева, В. В. Данилов, А. С. Юрченко // Материалы конференции Материалы IV Международной научной конференции «Донецкие чтения 2019». – Том 1, Часть 2. – С. 164-165.

67. Кудряшов, П. П. Алгоритмы обнаружения лица человека для решения прикладных задач анализа и обработки изображений /П. П. Кудряшов. – Издательство физико-математической литературы. – М. – 2007. -Т.2.

68. Viola, P. Rapid object detection using a boosted cascade of simple features / P. Viola, M. Jones // Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition. December 2001, vol. 1, pp. 1063–6919.

69. Bradski, G. Learning OpenCV. Computer Vision with the OpenCV Library / G. Bradski, A. Kaebler – O'REILLY. – 2008. -- 556 с.

70. Осовский С. Нейронные сети для обработки информации / С. Осовский. – М.: Финансы и статистика, 2002. – 344 с.

71. WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set [Электронный ресурс]. – Режим доступа: https://archive.ics.uci.edu/ml/datasets/WISDM+Smartphone+and+Smartwatch+Activity+and+Biometrics+Dataset, свободный.

72. Бабичева, М.В. Распознавание лиц в режиме реального времени сверточной нейронной сетью / М. В. Бабичева, А. С. Шевченко // Вестник Донецкого национального университета. Серия Г: Технические науки. – 2018. – № 2 (60). – С. 67-71.

73. Грубо, И.И. Системы охранной сигнализации. Технические средства. /И. И. Грубо. – М: Солон-Пресс, 2017. —220 с.

74. Рутковская, И. Д. Нейронные сети, генетические алгоритмы и нечеткие системы: пер. с польск. И. Д. Рудинского. /Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия – Телеком, 2006. – 452 с.

75. Image Net Classification. Classifying with hre-trained models [Электронный ресурс].–Режим доступа:https://pjreddie.com/darknet /imagenet/#reference , свободный.

76. Николенко, С. Глубокое обучение / С. Николенко, А. Кадурин, Е. Архангелькая. – СПб.: Питер, 2018. – 480 с.

77. Бабичева, М. В. Автоматизированная система видеонаблюдения по распознаванию предметов повышенной опасности / М.В. Бабичева, В. В. Данилов

«Сборник научных трудов // Донецкий институт железнодорожного транспорта» . – 2020. – № 56. – С. 20-25

78. Цветков, А. А Алгоритмы распознавания объектов/ А. А. Цветков, Д. К. Шорох, М. Г. Зубарева, С. В. Юрсков, А. В. Шуклин, А. Л. Хамуш, И. Б. Ануфриев // Технические науки: проблемы и перспективы: материалы IV Междунар. науч. конф. (г. Санкт-Петербург, июль 2016 г.). — СПб.: Свое издательство. – 2016. — С. 20-28.

79. Бабичева, М. В. Атаки на нейросетевые классификаторы/ М.В. Бабичева // Вестник ДонНУ. Серия Г: Технические науки. – 2019. – № 2. – С. 51-55.

80. Stefan, S. Optimizing a production process by a neural network/genetic algorithm approsh / S. Stefan // EgngnApplic. Artif. Intell. - 2016. – Vol. 9, № 6. - P. 681-689.

81. Brown, Tom B. Adversarial Patch / T. B. Brown, M. Dandelion, R. Aurko , M. Abadi, J. Gilmer. [Электронныйресурс] arXiv preprint arXiv:1712.09665, 2017. — Режим доступа: https://arxiv.org/abs/1712.09665, свободный.

82. Kerasdocumentation. Applications [Электронный ресурс] .— Режим доступа: https://keras.io/applications/, свободный.

83. Brendel, Wieland Decision-based adversarial attacks: Reliable attacks against black-box machine learning models/ W. Brendel, J. Rauber, Jonas, M. Matthias. In International Conference on Learning Representations, [Электронныйресурс] arXiv preprint arXiv: 1712.04248, 2018. — Режим доступа: https://arxiv.org/abs/1712.04248, свободный.

84. Sabour, S. Adversarial manipulation of deep representations / S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, [Электронныйресурс] ICLR conference 2016, 2016. — Режим доступа: http://www.cs.toronto.edu/ ~fleet/research/Papers/iclr 2016 _conference.pdf, свободный.

85. Carlini, N. Adversarial examples are not easilydetected: Bypassing ten detection methods / N. Carlini and D. Wagner, [Электронныйресурс] arXiv preprint arXiv: 1705.07263, 2017. — Режим доступа: https://arxiv.org/abs/ 1705.07263, свободный.

86. Moosavi-Dezfooli, S. Deepfool: a simple and accurate method to fool deep neural networks / S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, [Электронныйресурс] arXiv preprint arXiv: 1511.04599, 2016. — Режим доступа: https://arxiv.org/abs/ 1511.04599, свободный.

87. Zhao, Z. Generating natural adversarial examples / Z. Zhao, D. Dua, and S. Singh, [Электронныйресурс] arXiv preprint arXiv: 1710.11342, 2017. — Режим доступа: https://arxiv.org/abs/ 1710.11342, свободный.

88. Tabacof, P. Exploring the space of adversarial images / P. Tabacof and E. Valle, [Электронныйресурс] arXiv preprint arXiv: 1510.05328, 2016. — Режим доступа: https://arxiv.org/abs/ 1510.05328, свободный.

89. Sharif , M. Accessorize toa crime: Real and stealthy attacks on state-of-the-art face recognition / M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, 2016 [Электронный ресурс] ReserchGate. — Режим доступа: https://www.researchgate.net/publication/309448167_Accessorize_to_a_Crime_Real_and_Stealthy_Attacks_on_State-of-the-Art_Face_Recognition, свободный.

90. Rozsa, A. Adversarial diversity and hard positive generation /A. Rozsa, E. M. Rudd, and T. E. Boult, [Электронныйресурс] arXiv preprint arXiv: 1605.01775, 2016. — Режим доступа: https://arxiv.org/abs/ 1605.01775, свободный.

91. Parkhi, O. M. Deep Face Recognition / O. M. Parkhi, A. Vedaldi, A. Zisserman, 2016 [Электронный ресурс] robots. — Режим доступа: http://www.robots.ox.ac.uk:5000/~vgg/publications/2015/Parkhi15/parkhi15.pdf

92. Qiong, Cao GFace2: A dataset for recognising faces across pose and age/ Q.Cao, L.Shen, W. Xie, O. M. Parkhi, A. Zisserman IEEE Conference on Automatic Face and Gesture Recognition, [Электронныйресурс] arXiv preprint arXiv: 1710.08092, 2018. — Режим доступа: https://arxiv.org/abs/ 1710.08092, свободный.

93. Liu, Y. Delving into transferableadversarial examples and black-box attacks / Y. Liu, X. Chen, C. Liu, and D. Song, [Электронныйресурс] arXiv preprint arXiv: 1611.02770, 2017. — Режим доступа: https://arxiv.org/abs/ 1611.02770, свободный.

ПРИЛОЖЕНИЕ А

ДОКУМЕНТЫ, ПОДТВЕРЖДАЮЩИЕ ВНЕДРЕНИЕ РЕЗУЛЬТАТОВ ДИССЕРТАЦИИ

УТВЕРЖДАЮ:

Директор

Общества с ограниченной ответственностью

«ФИРМА «МДЛ»

Белокобыльский М. А.

«____» _____ 2021 г.

г. Донецк

АКТ

о внедрении алгоритма распознавания лиц сверточной нейронной сетью оптимизированной редукцией нейронов, разработанного в кандидатской диссертации Бабичевой М. В.

Составлен комиссией: Щербаков С.М. – главный конструктор; члены комиссии: Гильман И. Я. –начальник отдела НСИ, Селедцов С.А. – инженер-программист в том, что разработанный в рамках кандидатской диссертационной работы Бабичевой М. В. алгоритм планируется к внедрению на предприятии ФИРМА «МДЛ» в систему аутентификации доступа на базе монитора видеодомофона М-480М с системой на кристалле Allwiner A-13. Основные положения приведены в описании алгоритма на 10 страницах.

Председатель комиссии: Щербаков С. М.

Члены комиссии: Гильман И. Я.

Селедцов С. А.

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ:
Проректор по научной и
инновационной деятельности
д-р техн. наук, профессор
_____ В.В. Данилов
«16» 03 2018 г.

АКТ № 01.18/12.1-34
внедрение результатов научно-исследовательской работы
Моделирование защищенных инфокоммуникационных систем
шифр Г-18/39 № госрегистрации 0118D000013
(название НИР, шифр, № госрегистрации)

1. Комиссия в составе:
Председатель: Дубровина В.А., заведующая НИЧ
Члены комиссии: Кошка Т.В., заведующая УЧ
Фоменко М.В., ученый секретарь НИЧ
Кишкань Л.О., заведующая отделом ОНИР НИЧ

в период с 12.03.18 **по** 16.03.18 **рассмотрела материалы внедрения результатов указанной** научно-исследовательской работы (НИР) в учебный процесс.

2. Комиссия установила:
Результаты НИР (методика аутентификации пользователей и программное приложение для распознавания лиц на основе нейронной сверточной сети) внедрены в учебный процесс кафедры радиофизики и инфокоммуникационных технологий путем
(способы, методы, методики, новые знания, выводы, результаты расчетов, рекомендации и т.п
использования в лабораторном практикуме по дисциплине «Основы информационной безопасности» - Лабораторная работа № 4 «Методы биометрической аутентификации»
2018 г.

Предложения к дальнейшему внедрению НТП, другие замечания _____

С актом ознакомлены:

Декан факультета _____ (ФИО, подпись, дата)
Зав. кафедрой _____ (ФИО, подпись, дата)
Руководитель НИР _____ (ФИО, подпись, дата)

Председатель комиссии _____ В.А. Дубровина
Члены комиссии:
_____ Т.В. Кошка
_____ М.В. Фомсико
_____ Л.О. Кишкань

ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ:
Проректор по научной и
инновационной деятельности
д-р физ.-мат. наук, профессор
_____ В.В. Данилов
05 2018 г.

АКТ 12.1-34

внедрение результатов научно-исследовательской работы

Моделирование защищенных инфокоммуникационных систем

шифр Г-18/39 № госрегистрации 0118D000013

(название НИР, шифр, № госрегистрации)

1. Комиссия в составе:
Председатель: Дубровина В.А., заведующая НИЧ
Члены комиссии: Кошка Т.В., заведующая УЧ
Фоменко М.В., ученый секретарь НИЧ
Кишкань Л.О., заведующая отделом ОНИР НИЧ

в период с 12.03.18 по 16.03.18 рассмотрела материалы внедрения результатов указанной научно-исследовательской работы (НИР) в учебный процесс.

2. Комиссия установила:
Результаты НИР (методика проверки цифровых документов на факт редактирования, база примеров цифровых документов для проверки, компьютерное приложение для защиты цифровых документов от редактирования при помощи стеганографического алгоритма создания цифровых водяных знаков)
(способы, методы, методики, новые знания, выводы, результаты расчетов, рекомендации и т.п.) внедрены в учебный процесс кафедры радиофизики и инфокоммуникационных технологий
путем использования в лабораторном практикуме по дисциплине «Основы информационной безопасности» *(лабораторный практикум, методические указания, конспект лекций, учебное пособие и т.д.*
Лабораторная работа № 12 «Проверка подлинности цифровых документов», Лабораторная работа № 13 «Защита цифровых документов при помощи цифровых водяных знаков» 2018 г.
Указать название, год внедрения) предоставить выписку из протокола заседания кафедры/факультета, подтверждающие использование НТП в учебном процессе)

Предложения к дальнейшему внедрению НТП, другие замечания _____

С актом ознакомлены:
Декан факультета _____ (ФИО, подпись, дата)
Зав. кафедрой _____ (ФИО, подпись, дата)
Руководитель НИР _____ (ФИО, подпись, дата)

Председатель комиссии _____ В.А. Дубровина
Члены комиссии:
_____ Т.В. Кошка
_____ М.В. Фоменко
_____ Л.О. Кишкань

Ученый секретарь

ПРИЛОЖЕНИЕ Б

ФРАГМЕНТ ПРОГРАММНОЙ РЕАЛИЗАЦИИ ИНТЕРПОЛЯЦИИ РЕКУРРЕНТНОЙ НЕЙРОННОЙ СЕТЬЮ

Фрагмент программной реализации интерполяции рекуррентной нейронной сетью

```
import torch
from torch.utils.data import TensorDataset, DataLoader
import torch.nn as nn
import torch.nn.functional as F
from torch.optim import Adam
from sklearn.datasets import load_digits
import numpy as np
import matplotlib.pyplot as plt


# загрузка данных из файла
x, y = load_digits(n_class=10, return_X_y=True)
# разделим выборку на тренировочную и тестовую
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
                                        shuffle=True, random_stat
scaler = StandardScaler()
scaler.fit(x_train)
#нормализация данных
x_train_scaled = scaler.transform(x_train)
x_test_scaled = scaler.transform(x_test)
class SimpleCNN(nn.Module):
  def __init__(self):
    super(SimpleCNN, self).__init__()
    self.tan1 = nn.Tanh1(in_channels=1, out_channels=3, kernel_size=5, stride=1, padding=2)
```

```
  self.tan1_s = nn. Tanh1 (in_channels=3, out_channels=3, kernel_size=5, stride=2, padding=2)
  self.tan2 = nn. Tanh2 (in_channels=3, out_channels=6, kernel_size=3, stride=1, padding=1)
      self.tan2_s = nn. Tanh2 (in_channels=6, out_channels=6, kernel_size=3, stride=2, padding=1)
      self.out3 = nn.Lenear(in_channels=6, out_channels=10, kernel_size=3, stride=1, padding=1)
      self.cout3_s = nn.Lenear(in_channels=10, out_channels=10, kernel_size=3, stride=2, padding=1)


      self.flatten = nn.Flatten()
      # гордый полносвязный слой
      self.fc1 = nn.Linear(10, 10)


    # функция для пропускания данных через модель
    def forward(self, x):
      x = F.tanh(self.tan1(x))
      x = F.tanh(self.tan1_s(x))
      x = F.tanh(self.tan2(x))
      x = F.tanh(self.tan2_s(x))
      x = F.tanh(self.lin3(x))
      x = F.tanh(self.lin3_s(x))
      x = self.flatten(x)
      x = self.lin3(x)
      x = F.lenear(x)
      return x


  batch_size = 40 # размер батча
  learning_rate = 1e-3 # шаг оптимизатора
```

```
    epochs = 1000 # количество эпох


    x_train_tensor = torch.tensor(x_train_scaled.reshape(-1, 1, 8, 8).astype(np.float32))
x_test_tensor = torch.tensor(x_test_scaled.reshape(-1, 1, 8, 8).astype(np.float32))
y_train_tensor = torch.tensor(y_train.astype(np.long))
y_test_tensor = torch.tensor(y_test.astype(np.long))
train_dataset = TensorDataset(x_train_tensor, y_train_tensor)
train_loader = DataLoader(train_dataset, batch_size=batch_size)
test_dataset = TensorDataset(x_test_tensor, y_test_tensor)
test_loader = DataLoader(test_dataset, batch_size=batch_size)
simple_cnn = SimpleCNN().cuda() # подключение gpu
criterion = nn.CrossEntropyLoss()
#функция оптимизации
optimizer = MSE(simple_cnn.parameters(), lr=learning_rate)
#обучение методом градиентного спуска с импульсом:
for epoch in range(epochs):
  simple_cnn.train()
  train_samples_count = 0
  true_train_samples_count = 0
  running_loss = 0

  for batch in train_loader:
    x_data = batch[0].cuda()
    y_data = batch[1].cuda()

    y_pred = simple_cnn(x_data)
    loss = criterion(y_pred, y_data)
```

```
    optimizer.zero_grad()
    loss.backward()
    optimizer.step() # обратное распространение ошибки
    running_loss += loss.item()
        y_pred = y_pred.argmax(dim=1, keepdim=False)
        true_interpolation = (y_pred == y_data).sum().item() # количество верно определенных данных в текущем батче
        true_train_samples_count += true_classified
        train_samples_count += len(x_data) # размер текущего батча
      train_accuracy = true_train_samples_count / train_samples_count
      # прогоняем тестовую выборку
      simple_cnn.eval()
      test_samples_count = 0
      true_test_samples_count = 0
      running_loss = 0
      for batch in test_loader:
        x_data = batch[0].cuda()
        y_data = batch[1].cuda()
        y_pred = simple_cnn(x_data)
        loss = criterion(y_pred, y_data)
        loss.backward()
        running_loss += loss.item()
        y_pred = y_pred.argmax(dim=1, keepdim=False)
        true_interpolation = (y_pred == y_data).sum().item()
        true_test_samples_count += true_interpolation
        test_samples_count += len(x_data)
      test_accuracy = true_test_samples_count / test_samples_count
      print(f"[{epoch}] test loss: {running_loss}, accuracy: {round(test_accuracy, 4)}")
```

```
    base_model = PytorchModel(net_type=SimpleCNN, net_params=dict(),
optim_type=MSE,
                              optim_params={"lr": 1e-3}, loss_fn=nn.CrossEntropyLoss(),
                              input_shape=(1, 8, 8), batch_size=32, accuracy_tol=0.02,
                              tol_epochs=10, cuda=True)
    base_model.fit(x_train_scaled, y_train) # обучение модели
    preds = base_model.predict(x_test_scaled)
    true_interpolation = (preds == y_test).sum()
    test_accuracy = true_interpolation / len(y_test) # accuracy
    print(f"Test accuracy: {test_accuracy}")
    Interpolation(
      base_estimator=PytorchModel(accuracy_tol=0.02, batch_size=32,
        cuda=True, input_shape=(1, 8, 8),
        loss_fn=CrossEntropyLoss(),
        net_params={},
        net_type=<class '__main__.SimpleCNN'>,
        optim_params={'lr': 0.001},
        optim_type=<class 'torch.optim.mse.MSE'>,
        tol_epochs=10),
      bootstrap=True, bootstrap_features=False, max_features=1.0,
      max_samples=1.0, n_estimators=10, n_jobs=None,
      oob_score=False, random_state=None, verbose=0,
      warm_start=False)
pylab.plot(xlist, [Interpolation(x) for x in xlist])
pylab.show()
```

ПРИЛОЖЕНИЕ В

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛОЖЕННОГО АЛГОРИТМА БИНАРИЗАЦИИ

Программная реализация алгоритма бинаризации итерационным алгоритмом с обработкой краев изображения, представленном в разделе 3.2.2.3.

```
L=double(I_dil); // преобразование типов матрицы изображения
[nx ny]=size(L);  //определение размера изображения
obj = 0;
back=L(1,1) + L(1,ny) + L(nx,1) + L(nx,ny); //определение краев изображения
for i=2:1:nx-1
for j=2:1:ny-1
obj = obj + L(i,j);
end end;
k =1;
back_no = 4;   // установка размера краев
obj_no = nx * ny - back_no;
back_m = back / back_no;
obj_m = obj / obj_no;
th(k) =round((back_m + obj_m) / 2);
k=k+1;
obj_no =0;
obj = 0;
back_no = 0;
back = 0;
for i=1:1:nx
for j=1:1:ny
if (L(i,j) >= th(k-1))
obj = obj + L(i,j);
obj_no = obj_no +1;
```

```
else
back = back + L(i,j);
back_no = back_no + 1;
end  end  end
back_m = back / back_no;
obj_m = obj / obj_no;
th(k) =round((back_m + obj_m) / 2);
while (th(k)~=th(k-1))
k=k+1;
obj_no =0;
obj = 0;
back_no = 0;
back = 0;
for i=1:1:nx
for j=1:1:ny
if (L (i,j) >= th(k-1))
obj = obj +L(i,j);
obj_no = obj_no +1;
else
back = back + L(i,j);
back_no = back_no + 1;
end end end
back_m = back / back_no;
obj_m = obj / obj_no;
th(k) =round((back_m + obj_m) / 2);
end
res_th=th(k);
nor_th = (res_th/256);
binaryim = im2bw(I_dil,nor_th);
```

ПРИЛОЖЕНИЕ Г

ФРАГМЕНТ ПРОГРАММЫ ДЛЯ РАСПОЗНАВАНИЯ ЛИЦ

Содержимое файла EigenObjectRecognizer.cs автоматизированной системы распознавания лиц, описанной в разделе 3.2.3.

```
using System;
using System.Diagnostics;
using Emgu.CV.Structure;
namespace Emgu.CV
{
   [Serializable]
  public class EigenObjectRecognizer
  {
    private Image<Gray, Single>[] _eigenImages;
    private Image<Gray, Single> _avgImage;
    private Matrix<float>[] _eigenValues;
    private string[] _labels;
    private double _eigenDistanceThreshold;
     /// Получить собственные векторы, которые образуют собственное пространство
     public Image<Gray, Single>[] EigenImages
    {
      get { return _eigenImages; }
      set { _eigenImages = value; }
    }
     /// Получить или установить метки для соответствующего учебного изображения
     public String[] Labels
    {
      get { return _labels; }
```

```
        set { _labels = value; }
      }
       public double EigenDistanceThreshold
      {
        get { return _eigenDistanceThreshold; }
        set { _eigenDistanceThreshold = value; }
      }
       /// Получение среднего изображения.
       public Image<Gray, Single> AverageImage
      {
        get { return _avgImage; }
        set { _avgImage = value; }
      }
       /// Получить собственные значения каждого учебного изображения
       public Matrix<float>[] EigenValues
      {
        get { return _eigenValues; }
        set { _eigenValues = value; }
      }
      private EigenObjectRecognizer()
      {
      }
       /// Создание объекта распознавателя с помощью конкретных tranning данных и параметров, он всегда будет возвращать наиболее похожий объект
      /// <param name="images">Образцы, используемые для обучения, каждый из них должен быть одного размера. Изображения нормированы по гистограмме</param>
      /// <param name="termCrit">Критерии обучения распознавателей</param>
        public EigenObjectRecognizer(Image<Gray, Byte>[] images, ref MCvTermCriteria termCrit)
```

```
    : this(images, GenerateLabels(images.Length), ref termCrit)
{
}

private static String[] GenerateLabels(int size)
{
  String[] labels = new string[size];
  for (int i = 0; i < size; i++)
    labels[i] = i.ToString();
  return labels;
}
  /// <param name="images">Образцы, используемые для обучения, каждый из них должен быть одного размера. Рекомендуется, чтобы изображения были нормированы по гистограмме</param>
  /// <param name="labels">Метки, соответствующие изображениям</param>
  /// <param name="termCrit">Критерии обучения распознавателей</param>
  public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, ref MCvTermCriteria termCrit)
  : this(images, labels, 0, ref termCrit)
{
}
  /// Создание распознавателя объектов с использованием данных и параметров передачи данных
  /// <param name="labels">Метки, соответствующие изображениям</param>
  /// <param name="eigenDistanceThreshold">
  /// Собственный порог расстояния (0, ~ 1000).
```

```
        /// Чем меньше число, тем более вероятно, что рассмотренное изображение будет рассматриваться как непризнанный объект.
        /// Если порог равен 0, распознаватель всегда будет рассматривать рассматриваемое изображение как один из известных объектов.
        /// <param name="termCrit">Критерии обучения распознавателей</param>
        public EigenObjectRecognizer(Image<Gray, Byte>[] images, String[] labels, double eigenDistanceThreshold, ref MCvTermCriteria termCrit)
            {
              Debug.Assert(images.Length == labels.Length, "The number of images should equals the number of labels");
            Debug.Assert(eigenDistanceThreshold >= 0.0, "Eigen-distance threshold should always >= 0.0");
            CalcEigenObjects(images, ref termCrit, out _eigenImages, out _avgImage);
              /*
              _avgImage.SerializationCompressionRatio = 9;
              foreach (Image<Gray, Single> img in _eigenImages)
                //Задайте коэффициент сжатия для наилучшего сжатия.
img.SerializationCompressionRatio = 9;
              */
              _eigenValues = Array.ConvertAll<Image<Gray, Byte>, Matrix<float>>(images,
               delegate(Image<Gray, Byte> img)
                    {
                 return new Matrix<float>(EigenDecomposite(img, _eigenImages, _avgImage));
               });
            _labels = labels;
            _eigenDistanceThreshold = eigenDistanceThreshold;
          }
```

```
        /// Вычисление собственного изображения для конкретного учебного
изображения
        /// <param name="eigenImages">Полученные собственные
изображения</param>
        /// <param name="avg">Полученное среднее изображение</param>
        public static void CalcEigenObjects(Image<Gray, Byte>[] trainingImages, ref
MCvTermCriteria termCrit, out Image<Gray, Single>[] eigenImages, out Image<Gray,
Single> avg)
            {
      int width = trainingImages[0].Width;
      int height = trainingImages[0].Height;
        IntPtr[] inObjs = Array.ConvertAll<Image<Gray, Byte>,
IntPtr>(trainingImages, delegate(Image<Gray, Byte> img) { return img.Ptr; });
        if (termCrit.max_iter <= 0 || termCrit.max_iter > trainingImages.Length)
          termCrit.max_iter = trainingImages.Length;
        int maxEigenObjs = termCrit.max_iter;

        #region initialize eigen images
        eigenImages = new Image<Gray, float>[maxEigenObjs];
        for (int i = 0; i < eigenImages.Length; i++)
          eigenImages[i] = new Image<Gray, float>(width, height);
        IntPtr[] eigObjs = Array.ConvertAll<Image<Gray, Single>,
IntPtr>(eigenImages, delegate(Image<Gray, Single> img) { return img.Ptr; });
        #endregion
        avg = new Image<Gray, Single>(width, height);
        CvInvoke.cvCalcEigenObjects(
          inObjs,
            ref termCrit,
          eigObjs,
          null,
```

```
                avg.Ptr);
            }
             /// Разложение изображения как собственные значения, используя
собственные векторы
        /// <param name="src">Изображение, подлежащее разложению</param>
        /// <param name="eigenImages">Собственные изображения</param>
        /// <param name="avg">Средние изображения</param>
        /// <returns>Собственные значения разложенного изображения</returns>
             public static float[] EigenDecomposite(Image<Gray, Byte> src, Image<Gray,
Single>[] eigenImages, Image<Gray, Single> avg)
            {
              return CvInvoke.cvEigenDecomposite(
                src.Ptr,
                Array.ConvertAll<Image<Gray, Single>, IntPtr>(eigenImages,
delegate(Image<Gray, Single> img) { return img.Ptr; }),
                avg.Ptr);
            }
             #endregion /// Учитывая собственное значение, восстановить
проецируемое  <param name="eigenValue">Собственные значения</param>
        /// <returns>Проецируемое изображение</returns>
        public Image<Gray, Byte> EigenProjection(float[] eigenValue)
            {
              Image<Gray, Byte> res = new Image<Gray, byte>(_avgImage.Width,
_avgImage.Height);
              CvInvoke.cvEigenProjection(
                Array.ConvertAll<Image<Gray, Single>, IntPtr>(_eigenImages,
delegate(Image<Gray, Single> img) { return img.Ptr; }),
                eigenValue,
                _avgImage.Ptr,
                res.Ptr);
```

```
            return res;
        }
    /// Получение эвклидового собственного расстояния между <paramref
name="image"/> и любым другим изображением в базе данных
        /// <param name="image">Изображение, которое нужно сравнить с
учебными изображениями</param>
        /// <returns>Массив собственного расстояния от каждого изображения в
обучающих изображениях</returns>
        public float[] GetEigenDistances(Image<Gray, Byte> image)
        {
          using (Matrix<float> eigenValue = new
Matrix<float>(EigenDecomposite(image, _eigenImages, _avgImage)))
            return Array.ConvertAll<Matrix<float>, float>(_eigenValues,
              delegate(Matrix<float> eigenValueI)
              {
                return (float)CvInvoke.cvNorm(eigenValue.Ptr, eigenValueI.Ptr,
Emgu.CV.CvEnum.NORM_TYPE.CV_L2, IntPtr.Zero);
              });
        }
    /// Учитывая <paramref name = "image" />, который нужно изучить, найти в
базе данных самый похожий объект, верните индекс и собственное расстояние
    /// <param name="image">Изображение для поиска из базы данных</param>
    /// <param name="index">Индекс самого похожего объекта</param>
        /// <param name="eigenDistance">Собственное расстояние самого
похожего объекта</param>
        /// <param name="label">Метка конкретного изображения</param>
        public void FindMostSimilarObject(Image<Gray, Byte> image, out int index,
out float eigenDistance, out String label)
        {
          float[] dist = GetEigenDistances(image);
```

```
            index = 0;
            eigenDistance = dist[0];
            for (int i = 1; i < dist.Length; i++)
            {
              if (dist[i] < eigenDistance)
              {
                index = i;
                eigenDistance = dist[i];
              } }
            label = Labels[index]; }
        /// Распознать изображение и вернуть его ярлык
        /// <param name="image">Изображение, подлежащее распознаванию</param>
        /// String.Empty, если не признано; Обозначение соответствующего
изображения, в противном случае
                  public String Recognize(Image<Gray, Byte> image)
                {        int index;
            float eigenDistance;
                  String label;
            FindMostSimilarObject(image, out index, out eigenDistance, out label);
            return (_eigenDistanceThreshold <= 0 || eigenDistance <
_eigenDistanceThreshold )  ? _labels[index] : String.Empty;
          }   }   }
```

ПРИЛОЖЕНИЕ Д

ФРАГМЕНТ ПРОГРАММЫ ДЛЯ РАСПОЗНАВАНИЯ ОПАСНЫХ ПРЕДМЕТОВ

Фрагмент программы для распознавания опасных предметов в режиме реального времени, описанной в разделе 3.2.4.

```
import numpy as np
import tensorflow as tf
slim = tf.contrib.slim

_BATCH_NORM_DECAY = 0.9
_BATCH_NORM_EPSILON = 1e-05
_LEAKY_RELU = 0.1

_ANCHORS = [(10, 13), (16, 30), (33, 23), (30, 61), (62, 45), (59, 119),
            (116, 90), (156, 198), (373, 326)]

def darknet53(inputs):
    inputs = _conv2d_fixed_padding(inputs, 32, 3)
    inputs = _conv2d_fixed_padding(inputs, 64, 3, strides=2)
    inputs = _darknet53_block(inputs, 32)
    inputs = _conv2d_fixed_padding(inputs, 128, 3, strides=2)

    for i in range(2):
        inputs = _darknet53_block(inputs, 64)
    inputs = _conv2d_fixed_padding(inputs, 256, 3, strides=2)

    for i in range(8):
        inputs = _darknet53_block(inputs, 128)
    route_1 = inputs
    inputs = _conv2d_fixed_padding(inputs, 512, 3, strides=2)
```

```
    for i in range(8):
        inputs = _darknet53_block(inputs, 256)
    route_2 = inputs
    inputs = _conv2d_fixed_padding(inputs, 1024, 3, strides=2)

    for i in range(4):
        inputs = _darknet53_block(inputs, 512)
    return route_1, route_2, inputs

def _conv2d_fixed_padding(inputs, filters, kernel_size, strides=1):
    if strides > 1:
        inputs = _fixed_padding(inputs, kernel_size)
    inputs = slim.conv2d(inputs, filters, kernel_size, stride=strides,
                         padding=('SAME' if strides == 1 else 'VALID'))
    return inputs

def _darknet53_block(inputs, filters):
    shortcut = inputs
    inputs = _conv2d_fixed_padding(inputs, filters, 1)
    inputs = _conv2d_fixed_padding(inputs, filters * 2, 3)

    inputs = inputs + shortcut
    return inputs
def _fixed_padding(inputs, kernel_size, *args, mode='CONSTANT', **kwargs):

    pad_total = kernel_size - 1
    pad_beg = pad_total // 2
    pad_end = pad_total - pad_beg
```

```
    if kwargs['data_format'] == 'NCHW':
        padded_inputs = tf.pad(inputs, [[0, 0], [0, 0],
                                        [pad_beg, pad_end],
                                        [pad_beg, pad_end]],
                               mode=mode)
    else:
        padded_inputs = tf.pad(inputs, [[0, 0], [pad_beg, pad_end],
                                        [pad_beg, pad_end], [0, 0]], mode=mode)
    return padded_inputs
def _yolo_block(inputs, filters):
    inputs = _conv2d_fixed_padding(inputs, filters, 1)
    inputs = _conv2d_fixed_padding(inputs, filters * 2, 3)
    inputs = _conv2d_fixed_padding(inputs, filters, 1)
    inputs = _conv2d_fixed_padding(inputs, filters * 2, 3)
    inputs = _conv2d_fixed_padding(inputs, filters, 1)
    route = inputs
    inputs = _conv2d_fixed_padding(inputs, filters * 2, 3)
    return route, inputs
def _get_size(shape, data_format):
    if len(shape) == 4:
        shape = shape[1:]
    return shape[1:3] if data_format == 'NCHW' else shape[0:2]


def _detection_layer(inputs, num_classes, anchors, img_size, data_format):
    num_anchors = len(anchors)
    predictions = slim.conv2d(inputs, num_anchors * (5 + num_classes), 1,
                              stride=1, normalizer_fn=None,
                              activation_fn=None,
                              biases_initializer=tf.zeros_initializer())
```

```
shape = predictions.get_shape().as_list()
grid_size = _get_size(shape, data_format)
dim = grid_size[0] * grid_size[1]
bbox_attrs = 5 + num_classes
if data_format == 'NCHW':
    predictions = tf.reshape(
        predictions, [-1, num_anchors * bbox_attrs, dim])
    predictions = tf.transpose(predictions, [0, 2, 1])
predictions = tf.reshape(predictions, [-1, num_anchors * dim, bbox_attrs])
stride = (img_size[0] // grid_size[0], img_size[1] // grid_size[1])
anchors = [(a[0] / stride[0], a[1] / stride[1]) for a in anchors]
box_centers, box_sizes, confidence, classes = tf.split(
    predictions, [2, 2, 1, num_classes], axis=-1)
box_centers = tf.nn.sigmoid(box_centers)
confidence = tf.nn.sigmoid(confidence)
grid_x = tf.range(grid_size[0], dtype=tf.float32)
grid_y = tf.range(grid_size[1], dtype=tf.float32)
a, b = tf.meshgrid(grid_x, grid_y)
x_offset = tf.reshape(a, (-1, 1))
y_offset = tf.reshape(b, (-1, 1))
x_y_offset = tf.concat([x_offset, y_offset], axis=-1)
x_y_offset = tf.reshape(tf.tile(x_y_offset, [1, num_anchors]), [1, -1, 2])
box_centers = box_centers + x_y_offset
box_centers = box_centers * stride
anchors = tf.tile(anchors, [dim, 1])
box_sizes = tf.exp(box_sizes) * anchors
box_sizes = box_sizes * stride
detections = tf.concat([box_centers, box_sizes, confidence], axis=-1)
classes = tf.nn.sigmoid(classes)
predictions = tf.concat([detections, classes], axis=-1)
```

```
        return predictions
    def _upsample(inputs, out_shape, data_format='NCHW'):
        if data_format == 'NCHW':
            inputs = tf.transpose(inputs, [0, 2, 3, 1])
        if data_format == 'NCHW':
            new_height = out_shape[3]
            new_width = out_shape[2]
        else:
            new_height = out_shape[2]
                    new_width = out_shape[1]
     inputs = tf.image.resize_nearest_neighbor(inputs, (new_height, new_width))
        if data_format == 'NCHW':
            inputs = tf.transpose(inputs, [0, 3, 1, 2])
        inputs = tf.identity(inputs, name='upsampled')
        return inputs
    def yolo_v3(inputs, num_classes, is_training=False, data_format='NCHW',
reuse=False):
        img_size = inputs.get_shape().as_list()[1:3]
        if data_format == 'NCHW':
            inputs = tf.transpose(inputs, [0, 3, 1, 2])
        inputs = inputs / 255
        batch_norm_params = {
            'decay': _BATCH_NORM_DECAY,
            'epsilon': _BATCH_NORM_EPSILON,
            'scale': True,
            'is_training': is_training,
            'fused': None,
        }
```

ПРИЛОЖЕНИЕ Е
РЕЗУЛЬТАТЫ АТАК НА НЕЙРОННЫЕ СЕТИ

Атаки проводились при помощи разработанного приложения, описанного в разделе 4.1.

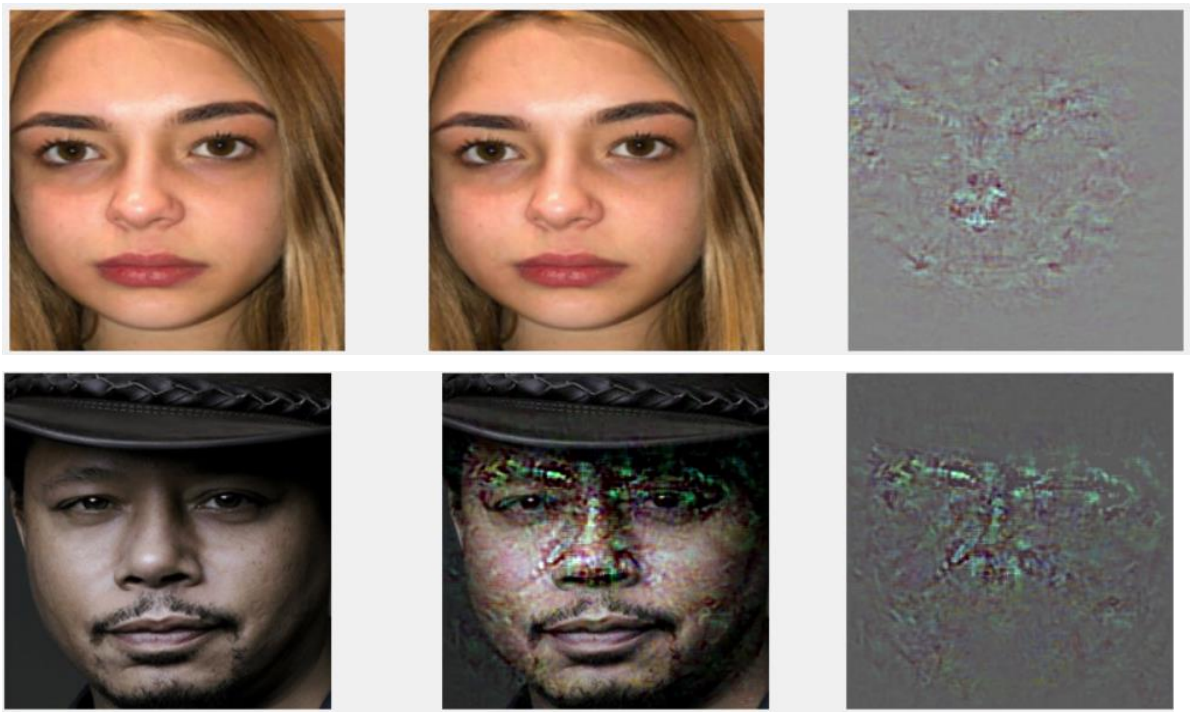


Рисунок Е.1 – Результат атаки GradientAttack

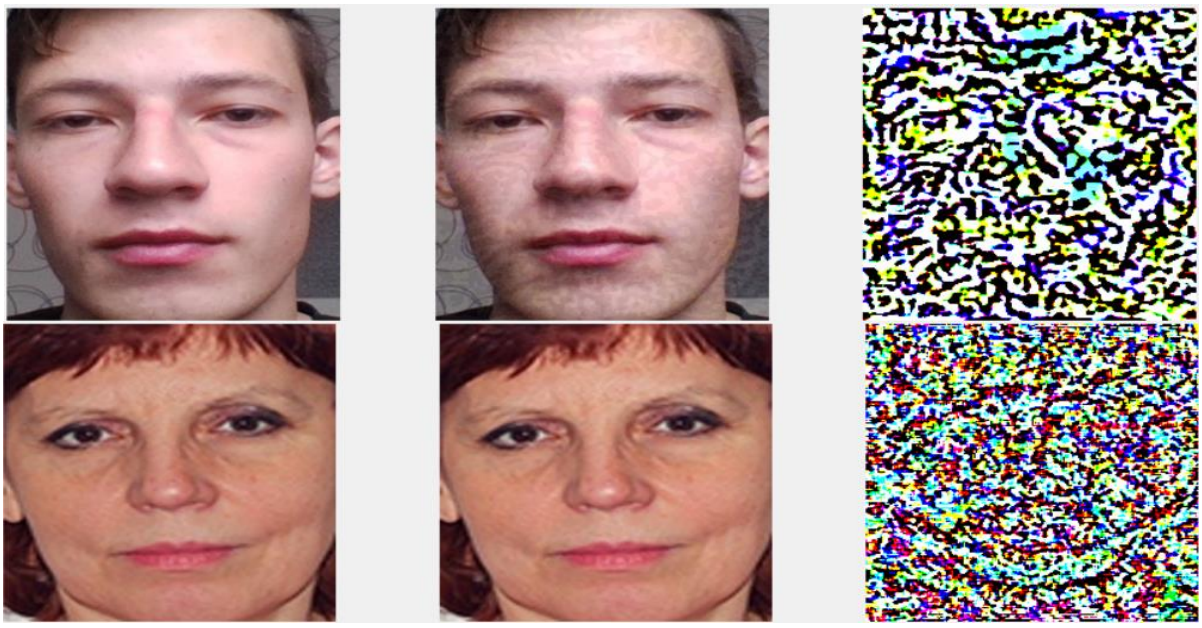


Рисунок Е.2 – Результат атаки GradientSignAttack

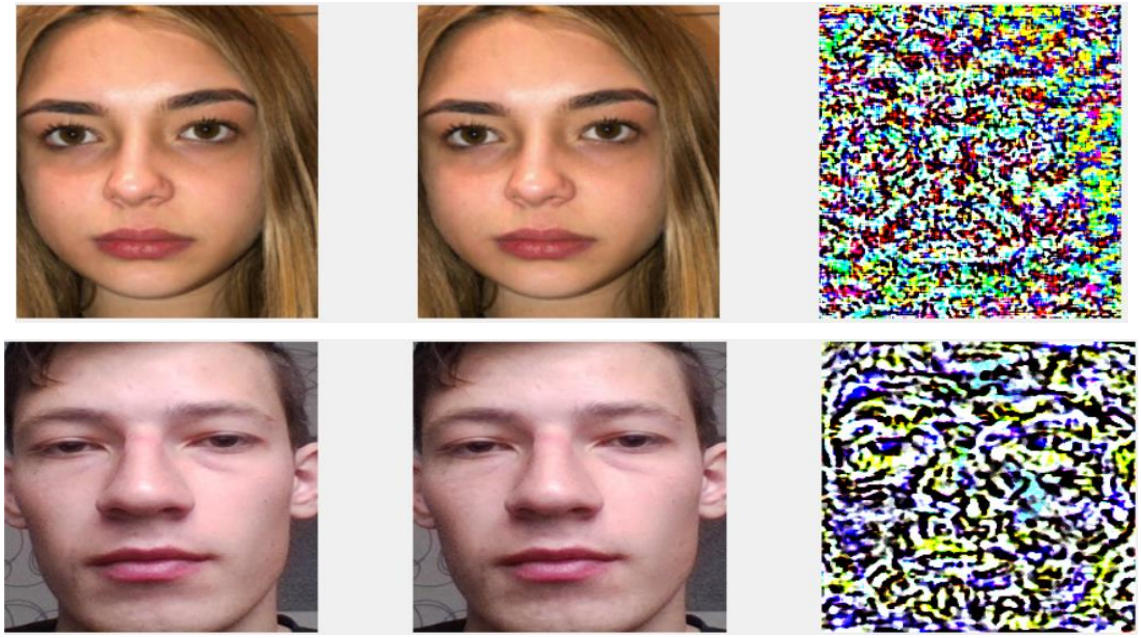


Рисунок Е.3 – Результат атаки BasicIterativeMethod

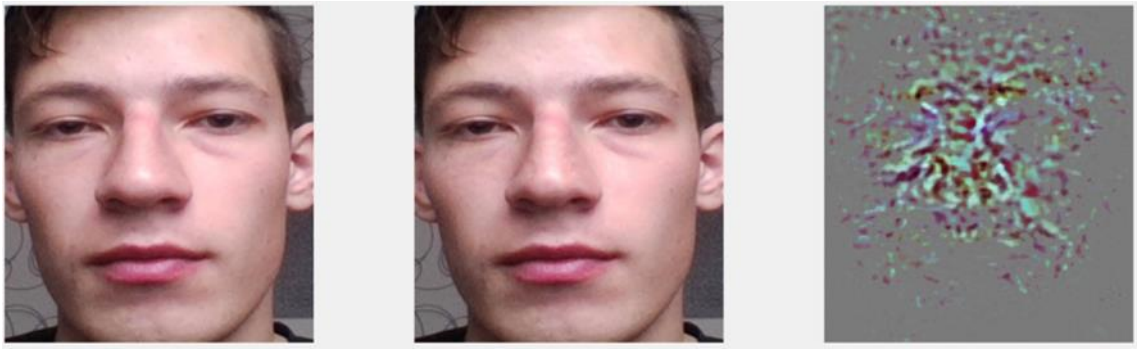


Рисунок Е.4 – Результат атаки CarliniWagnerL2Attack

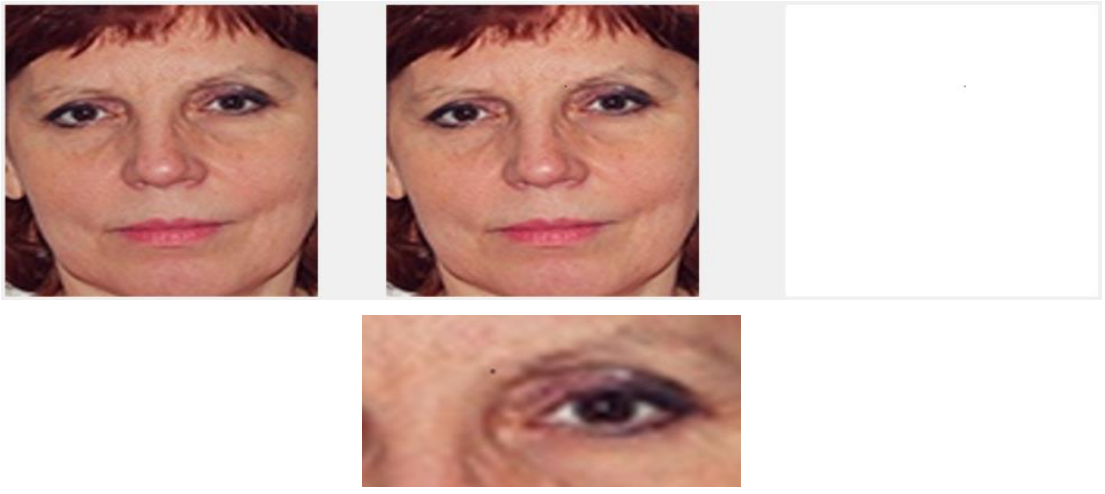


Рисунок Е.5 – Результат атаки SinglePixelAttack

Рисунок Е.6 – Результат атаки SinglePixelAttackDeepFoolAttack

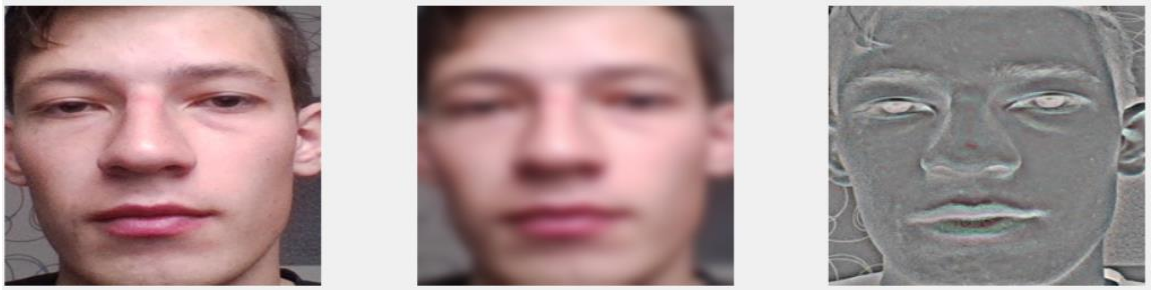


Рисунок Е.7 – Результат атаки GaussianBlurAttack

Рисунок Е.8 – Результат атаки AdditiveGaussianNoiseAttack

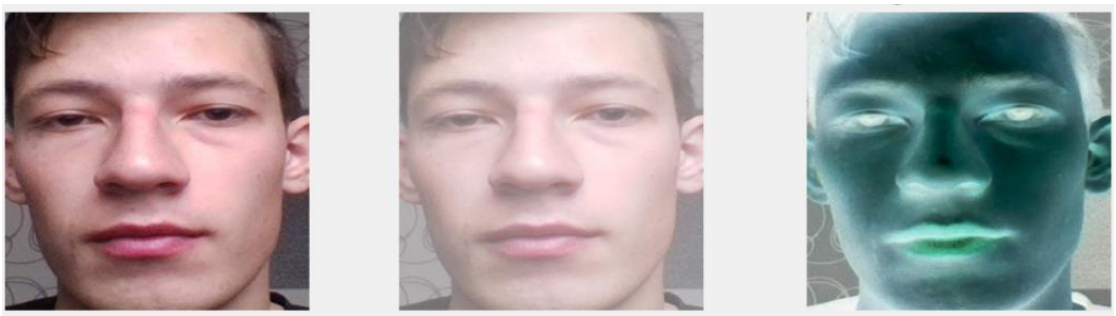


Рисунок Е.9 – ContrastReductionAttack