МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»
Кафедра систем информационной безопасности

«ДОПУЩЕН К ЗАЩИТЕ»
Зав. кафедрой к.п.н., доцент Р. Ш. Бердибаев
_____ «_____» _____2019 г.
(подпись)

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Нейронная сеть в обеспечении информационной безопасности
Специальность 5В100200 – «Системы информационной безопасности»
Выполнил Жақсыбек Нұржан Группа СИБ-15-2
Научный руководитель Алавердян Егисабет Церуновна

Консультант:
по экономической части:
к.э.н., профессор Аренбаева К.Г.
(ученая степень, звание, Ф.И.О)
(подпись) «11» июня 2019 г.

по безопасности жизнедеятельности:
ст. преподаватель Бекбасаров Н.Н.
(ученая степень, звание, Ф.И.О)
(подпись) «17» июня 2019 г.

по применению вычислительной техники:
к.т.н., доцент Алавердян Е.Ц., Сабикова В.Г.
(ученая степень, звание, Ф.И.О)
(подпись) «12» сентября 2019 г.

Нормоконтролер:
Ст. преподаватель Искакова Н.Б.
(ученая степень, звание, Ф.И.О)
(подпись) «12» июня 2019 г.

Рецензент:
канд. техн. наук, ассоц. проф. Айтхожаева Е.Ж.
(ученая степень, звание, Ф.И.О)
(подпись) «13» 06 2019 г.

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН
Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Институт систем управления и информационных технологий

Кафедра систем информационной безопасности

Специальность 5В100200 – «Системы информационной безопасности»

ЗАДАНИЕ
на выполнение дипломного проекта

Студенту Жақсыбек Нұржан

Тема проекта: Нейронная сеть в обеспечении информационной безопасности

Утверждена приказом по университету № 124 от « 26 » апреля 2019 г.

Срок сдачи законченного проекта « 13 » июня 2019 г.

Исходные данные к проекту (требуемые параметры результатов исследования (проектирования) и исходные данные объекта): проект представляет собой подход к выявлению вредоносного программного обеспечения для ОС Android. Данный подход основан на нейронной сети глубокого обучения

Перечень вопросов, подлежащих разработке в дипломном проекте, или краткое содержание дипломного проекта: дипломный проект включает в себя 5 глав, разделенных на подглавы, каждая из которых освещает определенную тематику, используемую при разработке нейронной сети.

В первой главе дипломного проекта представлены методы анализа Android-приложения и выбор наиболее подходящего.

Во второй главе дипломного проекта представлены методы глубокого обучения и выбор наиболее подходящего.

В третей главе дипломного проекта представлена разработка метода выявления вредоносного программного обеспечения для ОС Android.

Четвертая глава посвящена экономической составляющей дипломного проекта, в которой приведено технико-экономическое обоснование, а также все необходимые расчеты.

Пятая глава посвящена расчетам, необходимым для создания вентиляционных условий в помещении, пригодных к комфортной работе специалистов.

Перечень графического материала (с точным указанием обязательных чертежей):

1 схема работы нейронной сети;

2 схема динамического анализа;

3 схема макета;

4 схема подхода для анализа Android-приложения;

5 график работы нейронной сети.

Основная рекомендуемая литература:

1 Хайкин Саймон. X15 Нейронные сети: полный курс, 2-е изд., испр.: Пер. с англ. – М.: ООО “И.Д. Вильямс”, 2006. – 1104 с.

2 Аксенов С.В., Новосельцев В.Б. Организация и использование нейронных сетей (методы и технологии) / Под общ. ред. В.Б. Новосельцева. – Томск: Изд-во НТЛ, 2006. – 128 с. ISBN 5-89503-285-0.

3 Базовый курс «Как работает сверточная нейронная сеть: архитектура, примеры, особенности» // Stanislav Isakov. – 2018. URL: https://neurohive.io/ru/osnovy-data-science/glubokaya-svertochnaja-nejronnaja-set/ (дата обращения: 17.05.19).

4 Android Developers «Platform Architecture» URL: https://developer.android.com/guide/platform (дата обращения: 11.05.19).

5 Курс «Введение в разработку приложений для встроенных систем на платформе Intel Atom» // Константин Амелин, Олег Граничин, Владимир Кияев, Александр Корявко, Роман Лучин. – 2012 URL: https://www.intuit.ru/studies/courses/10617/1101/info (дата обращения: 03.05.19).

Конструкции по проекту с указанием относящихся к ним разделов проекта

Раздел	Консультант	Сроки	Подпись
Экономический	Арибаева Ж.Г.	04.03.-11.06	[signature]
Безопасность	Бекбаларов Н.Н. кат.	04.05 – 11.06	[signature]
Применение БТ	Сагитжанова Е.Т. Асаверджан Е. ст.	04.05.05.06	[signature]

График
подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
Изучение теории нейронных сетей	07.04. 2019	
Изучение ос Android	21.04. 2019	
Изучение структуры приложения	28.04. 2019	
Изучение Python 3	19.04. 2019.	
Разработка программной модели	29.04. 2019	
Создания нейронной сети	06.05. 2019	
Тестирование нейронной сети	09.05. 2019	
Изучение метода предсказания Android приложения в ходе исследования	21.04. 2019	
Экономическая часть	21.06. 2019	
Безопасность жизнедеят.	21.06. 2019	

Дата выдачи задания «26» октября 2019 г.

Заведующий кафедрой _____ (Бердибаев Р.М.)
(Подпись) (Ф.И.О)

Научный руководитель проекта _____ (Ниязбердин Е.Ш.)
(Подпись) (Ф.И.О)

Задание принял к исполнению студент _____ (Жаулыбаев Нурлан)
(Подпись) (Ф.И.О)

АННОТАЦИЯ

Данный дипломный проект посвящен подходу выявления вредоносного программного обеспечения для ОС Android. В теоретической части описаны существующие методики для анализа Android-приложения. В практической части рассмотрен метод выявления вредоносного программного обеспечения, основанный на представлении Android-приложения в виде изображения с последующим анализом сверточной нейронной сетью.

АНДАТПА

Бұл дипломдық жоба зиян келтіретін бағдарламалық қамтамасыздандыру жүйесін анықтау тәсілдерін жобалагауға арналған. Теориялық бөлім Андроид операциаялық жүйесінің талдайтын бар әдісін сипаттайды. Тәжірибелік бөлімде Android-қосымшасын бейне түрінде ұсынуға негізделген және кейіннен орамды нейрон желісімен талдауымен зиянды бағдарламалық қамтамасыз етуді анықтау әдісі қарастырылған.

ANNOTATION

This thesis project is dedicated to the approach to reveal the malicious software for OC Android. The theoretical part describes the existing methods for analyzing Android application. In the practical part, I considered the method of detection of malicious software based on the representation of the Android application in the form of an image with subsequent analysis of the convolutional neural network.

Алматы 2019

Содержание

Введение

По данным электронного ресурса Statcounter на долю компании Google и ее системы Android приходится около 74% процента рынка мобильных ОС. Ближайший конкурент, это компания Apple с операционной системой iOS располагающей около 23% рынка [1]. На сегодняшний день Android установлен на устройствах самых разных форм факторов, от браслетов до бортовых компьютеров автомобилей. Беря во внимание такую популярность данной ОС совсем неудивительно, что данная система является самой популярной среди разработчиков вредоносного программного обеспечения (ВПО).

Такой стремительный рост ВПО ведет за собой не менее быстрый рост средств защиты информации. На сегодняшний день существует большое количество готовых решений для распознавания ВПО. Обычно они делятся: методы динамического и статического анализа. Статический анализ проводится без запуска программы, его метод основывается на нахождении характерных признаков вредоносной программы в бинарном коде. А динамический анализе направлен на выявление аномального поведения приложения, с помощью которого можно будет утверждать, что приложение вредоносное. Также существует, гибридный анализ, которые сочетает в себе черты как статического, так и динамического.

Согласно отчету компании Kaspersky Lab за 2018 год [2] можно сделать вывод что проблема с защитой Android-устройств остается актуальной, даже не смотря на то что в официальном магазине Android-приложений была введена система Google Play Protect. По заявлению разработчиков, эта система с помощью методов машинного обучения ежедневно проверяет около 50 миллиардов Android-приложений на наличие вредоносного кода. Несмотря на все принятые меры количество ВПО растет в геометрической прогрессии.

Цели дипломного проекта:

1 Провести сравнение методов анализа Android-приложений.

2 Изучить методы глубокого обучения и выбрать наиболее подходящий.

3 Разработать среду анализа Android-приложения

4 Разработать подход выявления ВПО для ОС Android.

5 Реализация программного макета, реализующий данный метод.

1 Анализ предметной области

1.1 Понятие искусственных нейронных сетей. Сети прямого распространения

Искусственные нейронные сети (англ. Artificial Neural Network, ANN), далее ИНС – это вычислительные системы и их программно-аппаратное воплощение, за основу которых легли биологические сети строения мозга. Главным преимуществом которых является – способность к обучению.

На рисунке 1.1 и 1.2 наглядно продемонстрировано заимствование модели построения нейрона у биологического образца [3].

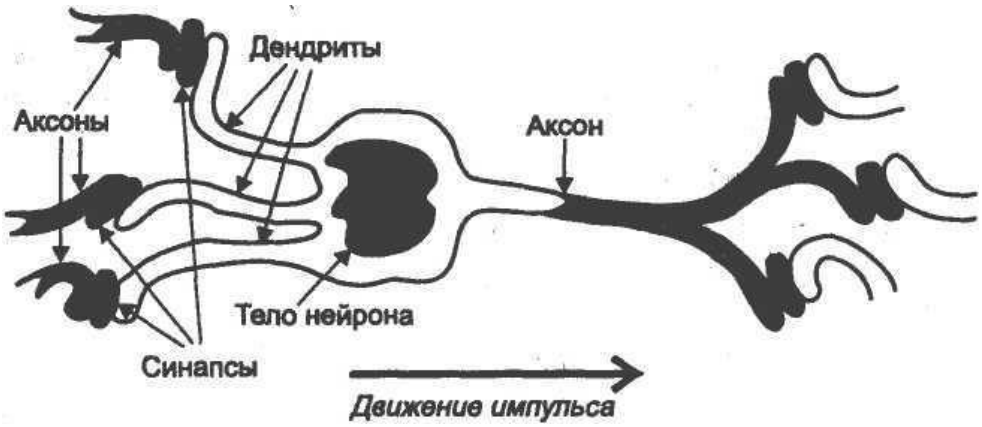

Рисунок 1.1 – Строение биологического нейрона

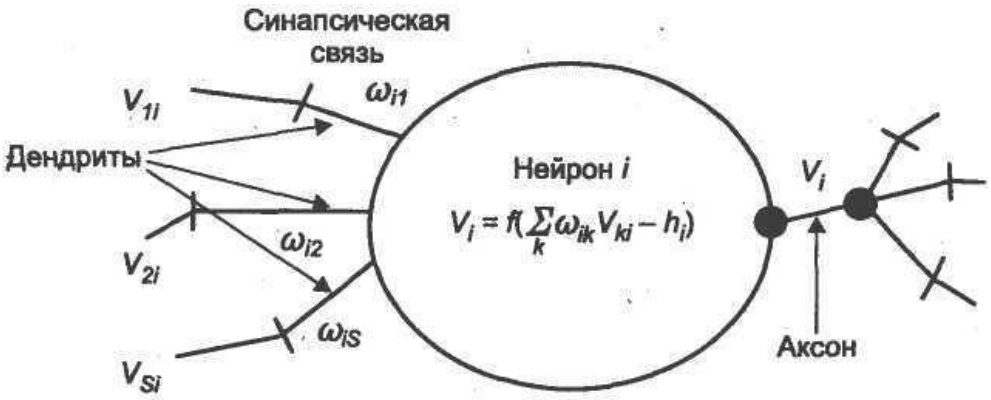

Рисунок 1.2 – Смоделированная математическая модель нейрона

Исходно-базовым элементом простейшей ИНС является искусственный нейрон или сигмовидный нейрон. Его название происходит от используемой в нем функции активации (рисунок 1.3).

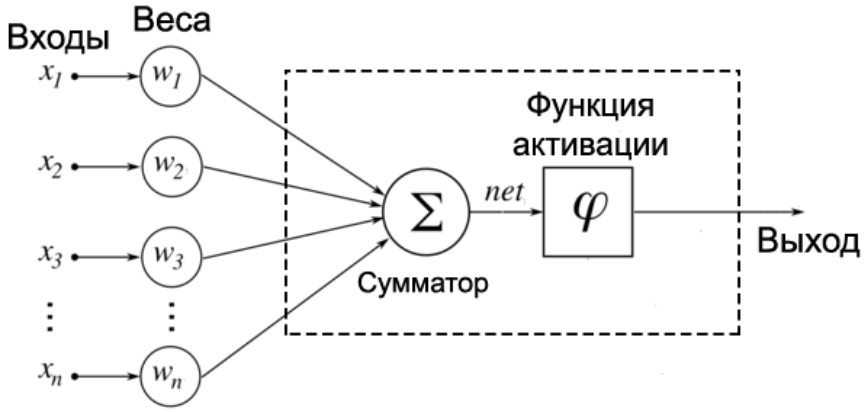

Рисунок 1.3 – Искусственный нейрон

У каждого нейрона есть входы, через которые он принимает сигнал, поступивший сигнал умножается на свои веса. Сигнал первого входа x_1 умножается на соответствующий этому входу вес w_1. В итоге получаем x_1w_1. И так до n-ого входа. В итоге на последнем входе получаем x_nw_n [4].

$$x_1w_1 + x_2w_2 + \cdots + x_nw_n = \sum_{i=1}^{n} x_iw_i \qquad (1.1)$$

Один нейрон способен решать только линейно разделимые задачи, но, если собрать сеть из нескольких нейронов, можно выйти за рамки. В таком случае стоит воспользоваться однослойной сетью – в такой сети, кроме входных и выходного слоев нейронов есть промежуточный слой, именуемый скрытым слоем.

Все рассмотренные выше сети относились к сетям прямого распространения, т.е. сигнал проходил строго от входа к выходу. Такие сети широко используется для таких задач как: кластеризация, прогнозирование и распознавание. На рисунке 1.4 представлена искусственная нейронная сеть прямого распространения:

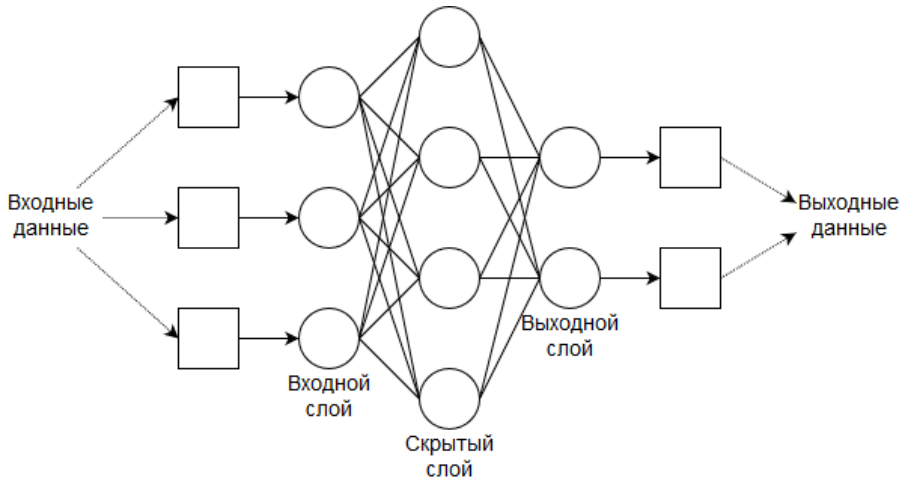

Рисунок 1.4 – Схема сети прямого распространения

1.2 Глубокие нейронные сети

Глубокая нейронная сеть (англ. Deep Neural Network, DNN) – это практически любая сеть, имеющая больше одного скрытого слоя. Теоретически вычислительные способности глубокой нейронной сети неограничены, вопрос лишь в количестве скрытых слоев и ресурсоемкости оборудования. Простой пример глубокой нейросети – это сеть прямого распространения с несколькими скрытыми слоями h_1 и h_2 являющиеся также полносвязными слоями (рисунок 1.6).

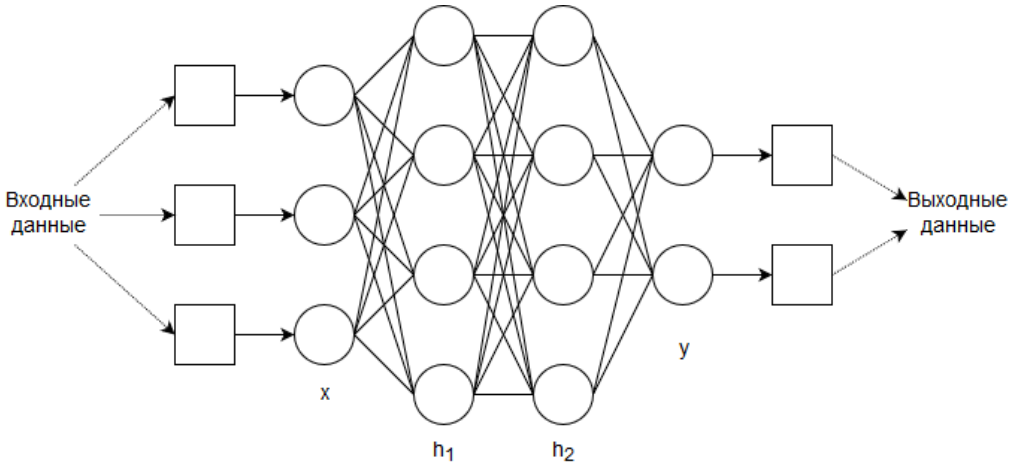

Рисунок 1.6 – ИНС прямого распространения

Основной проблемой сети прямого распространения является то что при большом количестве скрытых слоев и нейронов, процесс обучения, называемым обратным распространением ошибки (англ. backpropagation) начинает работать медленно. Это связано с тем что каждый нейрон любого слоя связан со всеми нейронами предыдущих и последующих слоев.

Чаще всего к глубоким нейросетям относят:

1 Нейронные сети прямого распространения.

2 Рекуррентные нейронные сети.

3 Сверточные нейронные сети.

4 Капсульные нейронные сети.

Существует огромное количество глубоких нейронных сетей, но вышеуказанные являются самыми изученными и популярными. На рисунке 1.7 представлены схемы и строения самых популярных нейронных сетей [5].

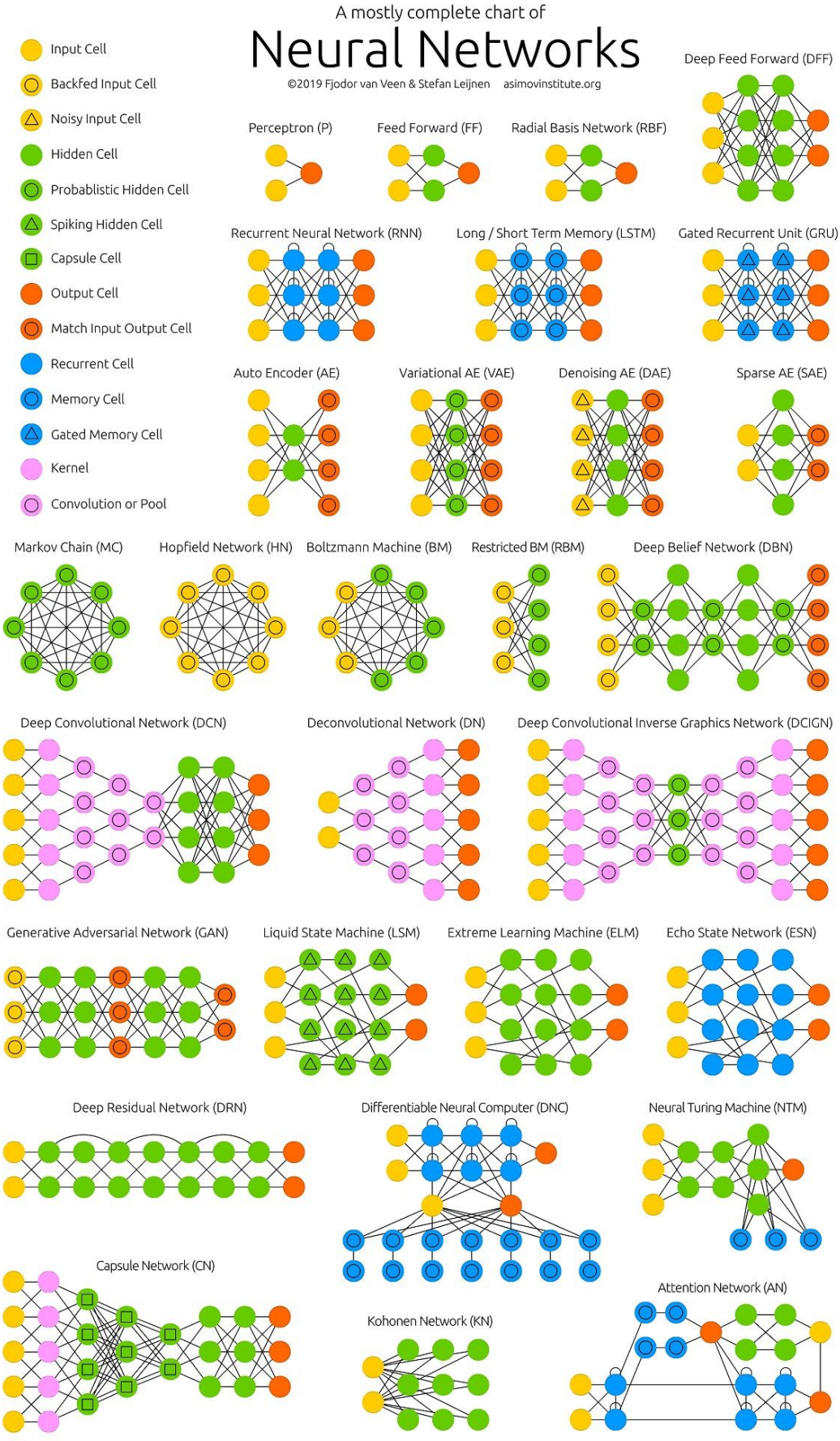

Рисунок 1.7 – Популярные архитектуры ИНС

1.3 Сети с обратными связями. Рекуррентные нейронные сети. сеть с долгой краткосрочной памятью

Рекуррентная нейронные сеть (англ. Recurrent Neural Networks, RNN) – в сетях такого типа между элементами образуются направленные последовательности. Данное особенность позволяет строить обратные связи и сохранять информацию. Рекуррентные сети широко используются в машинном переводе.

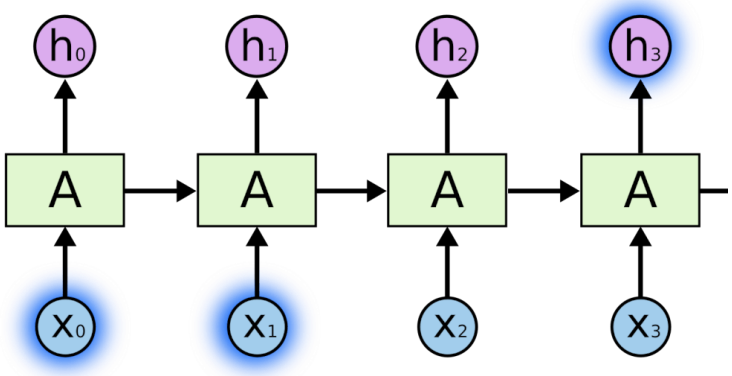

Рисунок 1.5 – Схема рекуррентной нейронной сети

Рекуррентная нейронная сеть использует раннее полученные данные на основе которых занимается решения поставленной задачи.

В некоторых случаях для решения той или иной задачи требуется обратиться лишь к предыдущей информации. Например: «рыба в воде», для того чтобы нейросети предсказать какое слово следует за «рыба в…» необходимо обратиться только к предыдущему слову.

Но в большинстве случаев требуется гораздо больше информации. Например: «Я живу в Великобритании… Я свободно говорю на...» В данном случае нейросеть понимает, что после «говорю на…» следует название языка. Но это вырвано из контекста, возникает необходимость в раннем упоминании Великобритании. Таким образом возникает разрыв [6]. Разрыв продемонстрирован на рисунке 1.6.

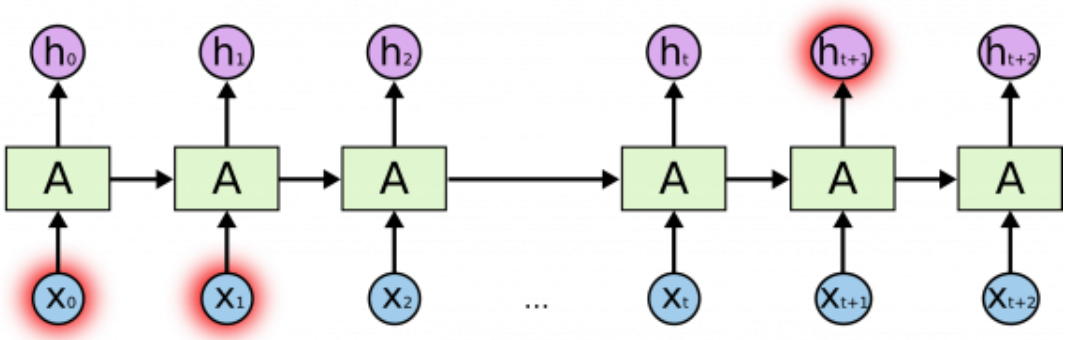

Рисунок 1.6 – Разрыв в рекуррентной сети

К сожалению, по мере увеличения этого разрыва РНС теряют связь между информацией.

Медленно, но верно мы подошли к следующему типу нейронных сетей – сети с долгой краткосрочной памятью, которая не страдает проблемами рекуррентных сетей.

Сети с долгой краткосрочной памятью (англ. Long Short Term Memory, LSTM) – данный тип нейронной сети основан на рекуррентной нейронной сети, способной обучаться долгосрочным зависимостям. Главная их особенность – это запоминание информации на долгие периоды времени, вследствие чего их почти не нужно обучать.

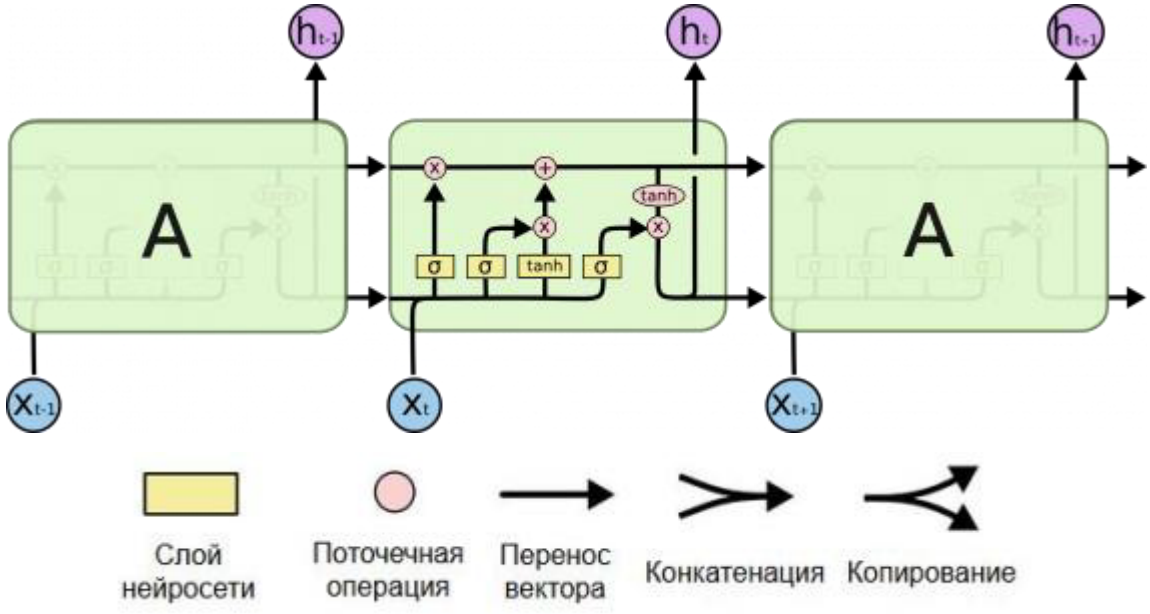

Рисунок 1.7 – Схема LSTM

Структура LSTM представляет собой цепочку, представленный выше модуль LSTM имеет четыре уровня, каждый из которых взаимодействует особым образом.

1.4 Сверточные нейронные сети

Сверточные нейронные сети (англ. Convolutional Neural Network, CNN) – сеть глубокого обучения по структуре схожая с сетью прямого распространения. Этот тип нейронных сетей изначально разрабатывался для работы с изображениями, в итоге это позволило оптимизировать работу нейронной сети для более высокой производительности. Основные различия представлены на рисунке 1.8.

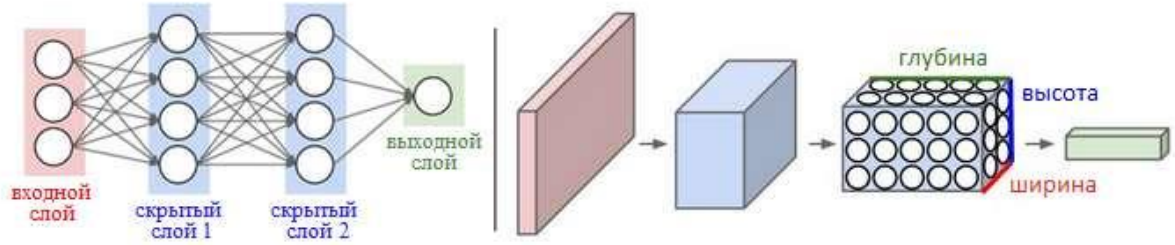

Рисунок 1.8 – Разница между обычной и сверточной нейросетью

На рисунке слева изображена сеть прямого распространения с двумя скрытыми слоями. На входе она принимает вектор значений и на их основе она получает вектор выходных значений. Справа представлена сверточная нейронная сеть, в которой нейроны организованы не в последовательные слои, а в 3 измерения: ширине, высоте и глубине. Каждый слой сверточной сети преобразует входной 3D объём значений функций активации. На этой схеме шириной и высотой будут размеры изображения, а глубиной будет палитра RGB.

Сверточная сеть представляет собой некоторую последовательность слоев, причем каждый слой преобразует один объём в другой при помощи функции. Данная сеть имеет следующие слои: сверточный слой (англ. Convolutional Layer), слой пулинга (англ. Pooling Layer) и полносвязный слой. Также необходимо упомянуть о отдельном слое вычисляющую функцию активации ReLU (формула 1.2),

$$\mathrm{ReLU}(x) = \max(0, x) \tag{1.2}$$

Далее приведу пример простейшей сверточной нейронной сети. В которой набор данных CIIFAR-10 состоящий из 60000 тысяч RGB изображений разрешением 32x32, поделенных на 10 классов.

[INPUT > CONV > RELU > POOL > FC],

где:

INPUT – входной слой 32x32x3, содержит значение пикселей.

CONV – сверточный слой 32x32x12 (12 фильтров), каждый нейрон вычисляет взвешенную сумму. Соединен только с локальными областями.

RELU – слой элементарной функции ReLU, не изменяет объем. Фильтрует только значимые признаки.

POOL – слой пулинга, исполняет операцию понижающей дискретизации (англ. downsampling) над пространственными измерениями (ширина и высота), преобразую исходный объем (16x16x12).

FC – полносвязный слой, вычисляет значение классов, преобразуя входные данные 1x1x10 (10 – количество классов CIFAR-10).

Сверточная нейронная сеть являются золотой серединой между биологическим прототипом сетей и обычным многослойным персептроном. В настоящее время сверточные нейронные сети в подавляющем большинстве случаев удачно распознают изображения. Их скорость в среднем на 10-15% выше сетей с другой архитектурой [7].

Современные сверточные сети невероятно популярны, основной причиной их успеха послужило то что они имеют небольшое количество параметров, по сравнению с более ранними версиями. Также их алгоритм обучения остается таким же и основывается на обратном распространении ошибки.

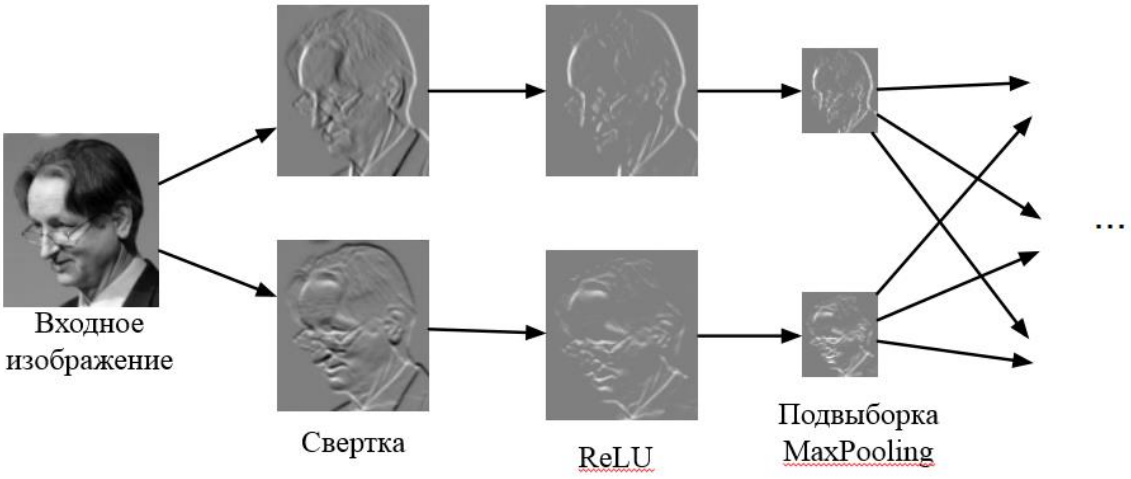

Рисунок 1.9 – Визуализация свертки и подвыборки

Однако, у свертоных нейронных сетей есть один серьезный конструктивный недостаток. Например: на картинке человеческое лицо, глаза, губы, нос и рот. Для сверточной сети сам по себе этот факт может послужить серьезным основанием полагать что на картинке изображено лицо. Для сверточной сети ориентационные и относительные пространственные отношения между компонентами не очень важны.

Рисунок 1.10 – Недостаток сверточной нейросети

Давайте проверим действительно сверточная сеть не различает эти два изображения. Компания Google не раскрывает архитектуру нейронной сети занимающийся поиском изображений, но с помощью данного примера можно предположить о том, что эта сеть является сверточной. При загрузке намеренно искаженного изображения, поисковик безошибочно показал что это Ким Кардашян и любезно предложил скачать не искаженное изображение в более высоком качестве (рисунок 1.11).

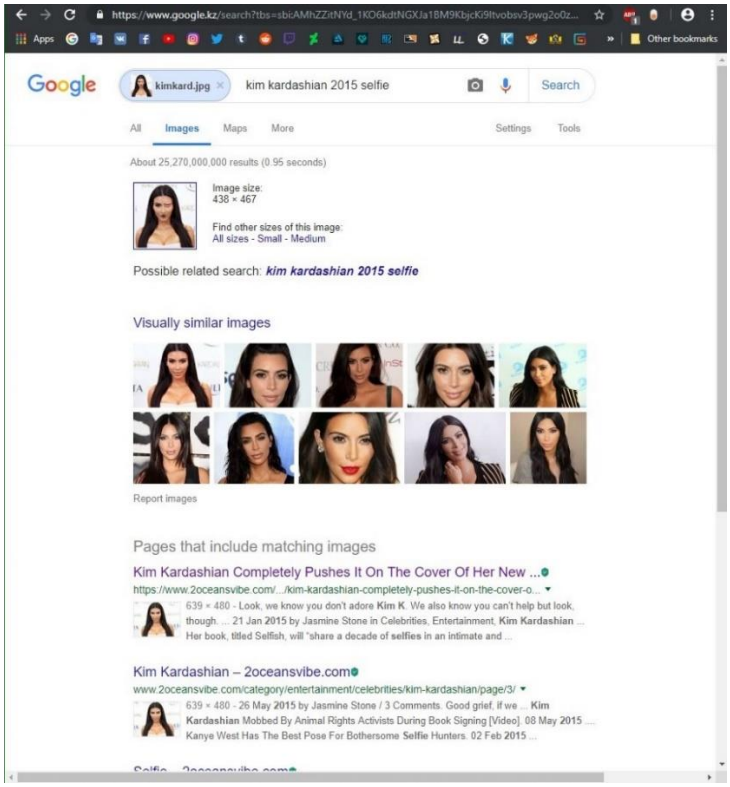

Рисунок 1.11 – Поиск по искаженному изображению

Сверточная сеть недоумевает, ведь на обеих картинках изображено лицо. Сверточный слой данной сети умеет определять наличие компонентов, но не может определить их расположение относительно других компонентов.

1.5 Сравнение глубоких нейронных сетей

Таблица 1.1 – Сравнение глубоких ИНС

Архитектура DNN	Основной слой/элемент	Особенности	Применение
Нейронная сеть прямого распространения (FFNN)	Полносвязный слой	Все скрытые слои являются полносвязными. Обучение значительно замедляется при увеличении нейронов и скрытых слоев	Аппроксимация всевозможных функций с не большим количеством входных параметров и скрытых слоев
Рекурсивная нейронная сеть (RNN)	Рекуррентный слой	Наличие обратных связей между нейронами. Способны связывать предыдущую информацию с текущей задачей. Не способна справляться с долговременными зависимостями	Распознавание речи и рукописного текста
Сеть долго срочной краткосрочной памяти (LSTM)	LSTM слой	Модификация RNN, разработанная специально для работы с долговременными зависимостями. Имеют LSRM слой, состоящий из трех фильтров, позволяющих защищать и контролировать состояние ячейки (какую информацию оставить, а какую отбросить)	Распознавание речи, рукописного текста, жестов. Сжатие текста на естественном языке. Захват изображений
Сверточная нейронная сеть (CNN)	Сверточный слой	Архитектурно спроектирована для распознавания изображений. Имеют сверточный слой, представляющий из себя набор фильтров входного объема изображения. Не требует предобучения без учителя, что ускоряет обучение и снижает вероятность попасть на локальный минимум	Распознавание изображений. Распознавание видеопоследовательности. Анализ данных захвата движения. Обработка естественного языка

API последовательности могут иметь большую длину, следовательно будет большое количество входных данных. Для этих целей больше всего подойдет сверточная нейронная сеть (CNN).

Многие исследователи, занимающиеся проблемой выявления вредоносных Android-приложений, используют именно сверточные нейронные сети и имеют высокую точность [8,9]. Также сверточные сети хороши в распознавании изображений, как в следствии мы имеем возможность представить последовательность пар $\{API_i, PL_j\}$ как единичный пиксель, как результат, все Android-приложение – в качестве изображения.

1.6 Общее описание операционной системы Android

Google Android – это операционная система с открытым программным кодом, основанная на ядре Linux, созданная для широкого круга форм-факторов. На рисунке 1.12 представлена ее компонентная модель:

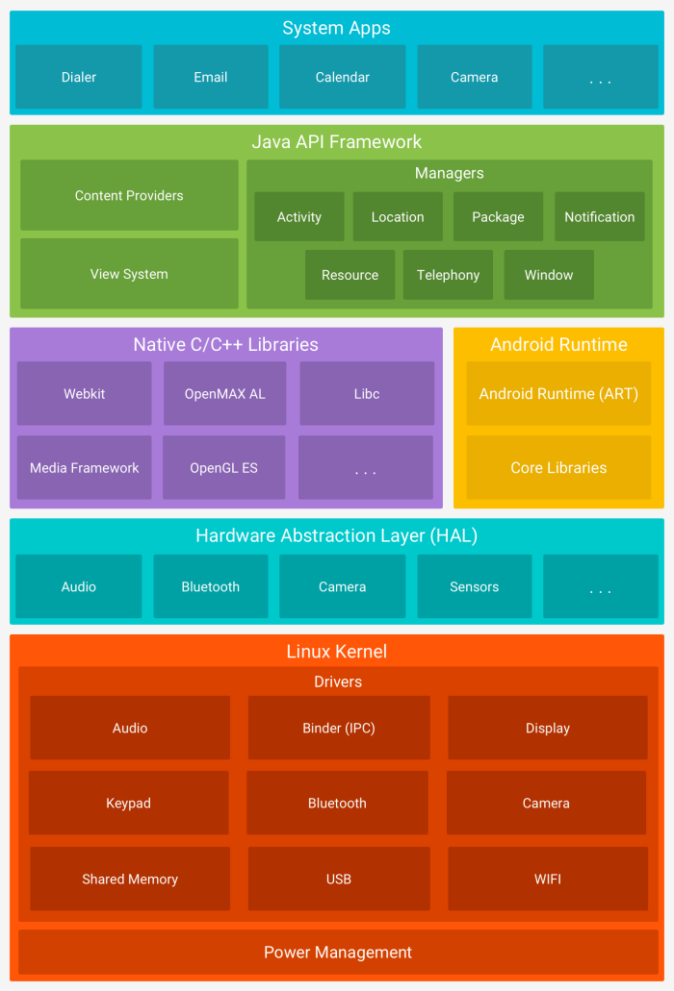

Рисунок 1.12 – Компонентная модель ОС Android

Коротко об основных компонентах программной архитектуры ОС:

1 Ядро Linux (англ. Linux Kernel) – обеспечивает функционирование системы и отвечает за безопасность, управление памятью, энергосистемой и

процессами, а также предоставляет сетевой стек и модель драйверов [10]. Благодаря ядру Linux упрощается процесс создание драйверов под Android, потому что их создание проходит под уже хорошо известное ядро Linux.

2 Слой аппаратных абстракций (англ. Hardware Abstraction Layer) – это стандартные алгоритмические интерфейсы, представляющие аппаратные возможности устройства высокоуровневой структуре Java API. Библиотека состоит из модулей, таких как: Аудио, Bluetooth, Камера и т.д. Каждый раз, когда высокоуровневый интерфейс API обращается к аппаратной части, система Android загружает модуль для данного компонента.

3 Среда выполнения Android (англ. Android Runtime) – это среда исполнения для устройств, использующих Android 5.0 или новее, каждое приложение запускается отдельных процессом со своим собственным экземпляром среды исполнения Android. Это позволяет запускать большое количество приложений одновременно на «слабых» устройствах с маленьким объёмом памяти. Достигается это путем компиляции приложения во время его установки.

4 Набор библиотек C/C++ (англ. Native C/C++ Libraries) – многие компоненты системы, такие как: слой аппаратных абстракций, среда выполнения построены и нуждаются в библиотеках, написанных на C и C++. Система Android предоставляет Java API доступ к библиотекам. Например: можно получить доступ к OpenGL ES через Java OpenGL API, для того чтобы добавить функции графического редактора 2D и 3D графики в приложении.

5 Java API Фреймворк (англ. Java API Framework) – полный спектр возможностей доступен через API, написанный на языке Java. Данные API-интерфейсы образуют модули, необходимые для создания Android - приложений. Тем самым упрощая процесс создания приложений на Android.

6 Системные приложения (англ. System Apps) – операционной система Android предоставляется с установленным набором приложений. К системным приложениям относятся: Контакты, Камера, Календарь, Калькулятор и т.д.

1.7 Структура Android-приложения

Приложения для ОС Android в большинстве случаев пишется на языке программирования Java, с помощью инструмента Android SDK. Android SDK – это программа для разработки программного обеспечения для платформы Android, содержит примеры проектов с исходным кодом, инструментами разработки, эмулятором и необходимыми библиотеками для создания приложений.

Android SDK компилирует Java-код и все сопроводительные файлы в единый файл APK (Android Application Package). Процесс компиляции представлен на рисунке 1.13:

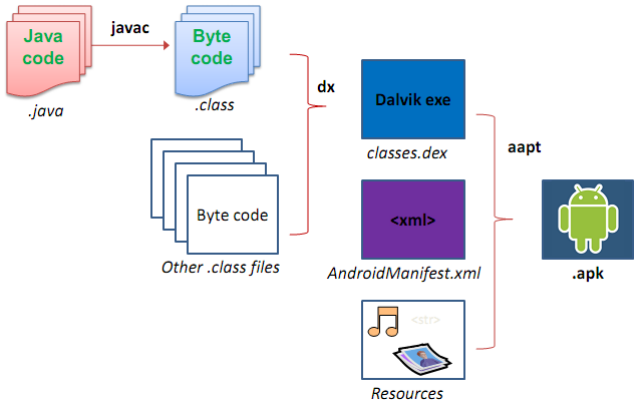

Рисунок 1.13 – Компиляция Android-приложения

Java-код при компиляции преобразуется в файл формата .class. Следующим шагом с помощью утилиты dx входящий в инструмент Android SDK служит конвертация файлов .class в файл .dex (Dalvik Executable).

Структура APK-файла показана на рисунке 1.14:

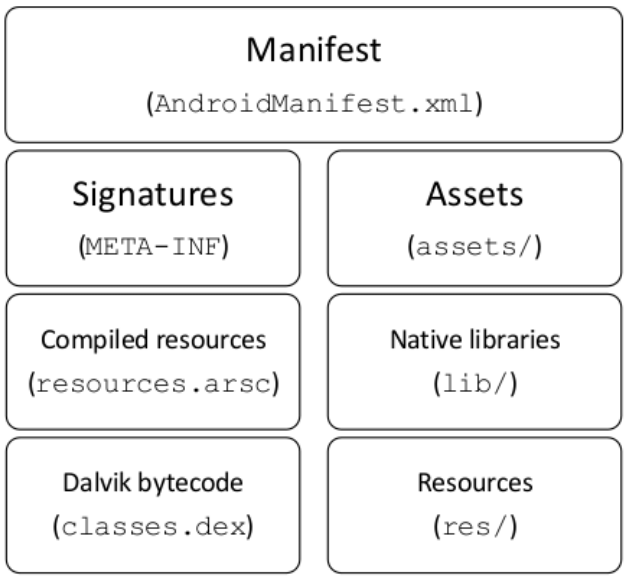

Рисунок 1.14 – Содержание APK-файла

AndroidManifest.xml – файл манифеста содержит важную информацию о запуске и выполнении приложения. Этот файл содержит разрешения, необходимые для доступа к API-интерфейсу.

META-INF – это файл метаданных JAR, содержит в себе контрольные суммы SHA-1, файлы сертификата RSA или DSA.

resources.arsc – скомпилированный xml-файл содержащий данные о ресурсах необходимых программе.

classes.dex – байт-код выполняемый в Dalvik VM или в Android Runtime.

assests/ - папка ресурсов приложения

lib/ - папка с нативными библиотеками C/C++, присутствует если приложение написано на C/C++.

res/ - папка с основными ресурсами, является «начинкой» приложения, графические элементы приложения.

Вывод

В данной главе рассмотрено понятие нейронной сети, были проведены параллели с биологическим прототипом. Было показаны типы нейронных сетей такие как: сети прямого распространения, рекуррентные сети, сети кратко долгосрочной памяти и сверточной сети. При детальном сравнении данных нейронных сетей было принято решение о построении своего подхода на основе сверточной нейронной сети исходя из ряда ее преимуществ перед другими нейронными сетями. Сверточная нейронная сеть заточена под распознавание изображений.

Также в данной главе описывалась структура ОС Android. Было показана компонентная модель ОС. Подробно рассмотрены компоненты apk файла Android-приложения.

2 Анализ используемых средств и ПО для реализации программного макета

2.1 Описание метода

Основной идеей данного метода является – представление Android-приложения в виде изображения для последующего анализа сверточной нейронной сетью, изображение представляет собой последовательности пар API вызова и советующему ему уровню защиты.

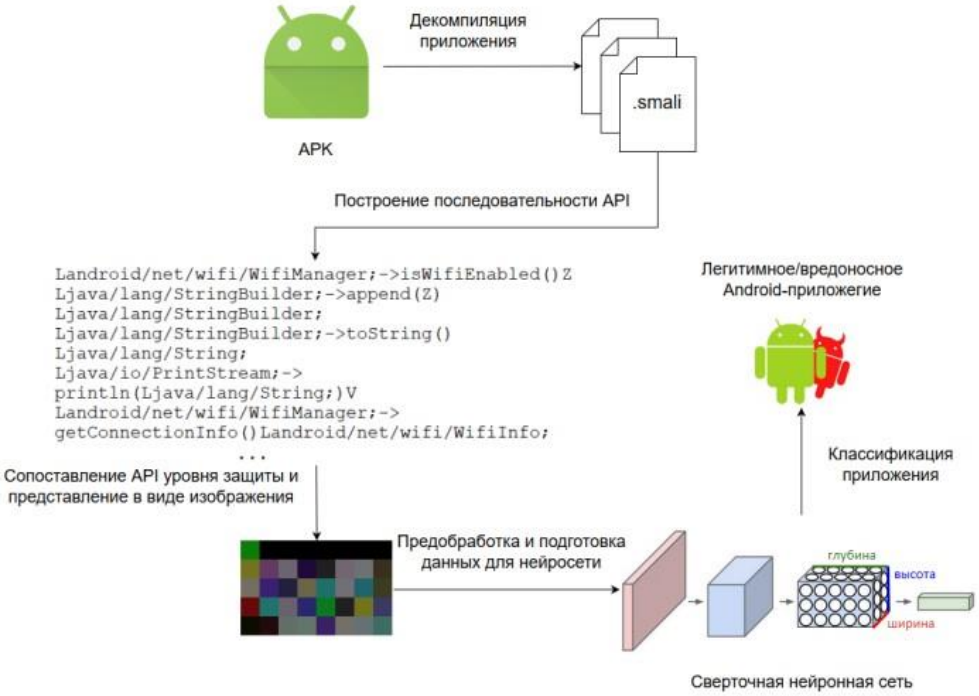

Рисунок 3.1 – Схема метода

Сначала декомпилируется Android-приложение, байт-код classes.dex дизассемблируется в код smali. Далее производится анализ полученного кода и

построение графа потока управления, по которому и выполняется построение последовательности API вызовов. Полученные API вызовы сопоставляются с уровнями защиты.

Ключевым моментом является представление последовательности пар API вызовов в виде изображения. Для дальнейшего анализа сверточной нейронной сетью

Последний шаг — это анализ изображения нейронной сетью и классификация приложения на легитимные и вредоносные.

2.2 Оборудование для среды анализа Android-приложения

Для процесса подбора гиперпараметров и обучению нейронной сети использовался мой рабочий сервер (рисунок 2.1 и таблица 2.1)

Рисунок 2.1 – HP ProLiant ML350 Gen9

Таблица 2.1 – Характеристики оборудования

CPU	Intel E5-2650v3 2.3 GHz
GPU	NVIDIA Quadro P4000 8GB
RAM	DDR4 32GB
SSD	240GB
OS	Ubuntu 16.04.4 LTS

2.3 Python. TensorFlow и Keras.

В данной работе я использовал язык программирования Python потому что этот язык программирования является основным для библиотек TensorFlow и Keras. Также Python очень мощный язык программирования позволяющий использовать эффективные высокоуровневые структуры данных и предлагающий эффективный подход к объектно-ориентированному программированию. Интуитивно понятный синтаксис делает его одним из наиболее простых в изучении языков программирования (рисунок 2.2).

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Рисунок 2.2 – Python

Изначально мною был использован инструмент Google Colab, который представляет собой удобный и легкий инструмент для создания простых

нейронных сетей. Но позже я отказался от данного инструмента в пользу более мощных библиотек TensorFlow и Keras.

TensorFlow – это библиотека для машинного обучения, целью которой является решения задач построения и тренировки нейронной сети. Фреймворк TensorFlow позволяет использовать концепцию графического представления для вычислительных задач. Ниже представлен простой пример сложения a и b, результат, которого c (рисунок 2.3).

```
# Импорт TensorFlow
import tensorflow as tf

# Определение a и b в качестве плейсхолдеров
a = tf.placeholder(dtype=tf.int8)
b = tf.placeholder(dtype=tf.int8)

# Определение сложения
c = tf.add(a, b)

# Инициализация графа
graph = tf.Session()

# Запуск графа
graph.run(c, feed_dict={a: 5, b: 4})
```

Рисунок 2.3 – Решение тестовой задачи

Keras – основной программный пакет для реализации нейронной сети. Данный пакет представляет собой высокоуровневый API интерфейс для конструирования нейронной сети. В моем подходе он выполняется поверх TensorFlow. Ниже представлен шаблон сверточной нейронной сети с учетом терминологии Keras (рисунок 2.4).

```
INPUT -> [[CONV -> RELU]*N -> POOL?]*M -> [FC -> RELU]*K -> FC
```

Рисунок 2.4 – Шаблон сверточной нейронной сети

2.4 Представление пары API и уровни защиты

Прогресс с каждой новой версией Android колоссален, количество доступных API с каждой новой версией растет в геометрической прогрессии. На данное время — это около сотни тысяч. Логичнее всего было бы закодировать API вызов в RGBA изображение, занять 3 канала пикселя API, а альфа канал использовать для упорядочения уровня защиты. Так прозрачные пиксели будут менее опасными, а менее прозрачные – более опасными.

У предложенного метода существует большой недостаток, при длинных API вызовах, состоящих из нескольких сотен строк, потребуется изображение 100x100 пикселей что негативно скажется на количество необходимых нейронов для обработки.

Для осуществление данной задачи необходимо модернизировать наш подход для представления пары {API, PL} в качестве RGBA пикселя. Был выбран метод преобразования из RGBA в RGB с применением фона. Например, я выполняю преобразование, в качестве фона выбрал черный цвет (#000000), в данном случае более прозрачные пиксели будут преобразованы в более темные цвета. На рисунке 3.2 показан пример преобразования

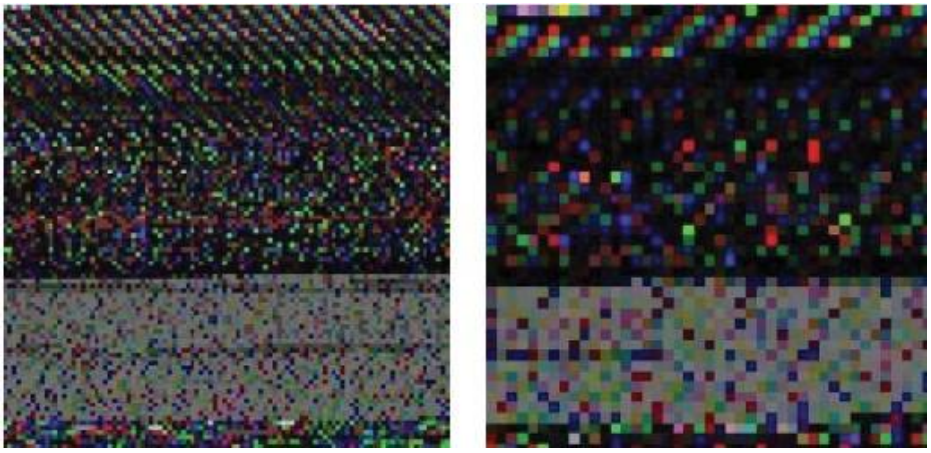

Рисунок 3.2 – Пример преобразованного вредоносного приложения

Для уплотнения API запросов использовалась не криптографическая хэш-функция MurmurHash3 [11].

Уровень защиты	Числовое значение	Краткое описание
NONE	1	Присваивается Android API, которым не было поставлено в соответствие никакое разрешение
UNKNOWN	2	Присваивается Java и Javax API вызовам
NORMAL	3	Соответствует группе разрешений Android SDK с уровнем защиты normal
SIGNATURE	4	Соответствует группе разрешений Android SDK с уровнем защиты signature
DANGEROUS	5	Соответствует группе разрешений Android SDK с уровнем защиты dangerous

Рисунок 3.3 – Уровни защиты

Преобразование уровня защиты в альфа диапазон от 0 до1 выполняется по формуле 3.1.

$$\mathrm{P}L^{\alpha} = \frac{PL_i}{PL_{dangerous}} \qquad (1.2)$$

где, $PL_i = \{1,\ldots,5\}$. $PL_{dangerous}=5$, а PL^{α} представляет значение PL_i в диапазоне от 0 до 1.

На рисунке 3.4 показан процесс преобразования из RGBA в RGB.

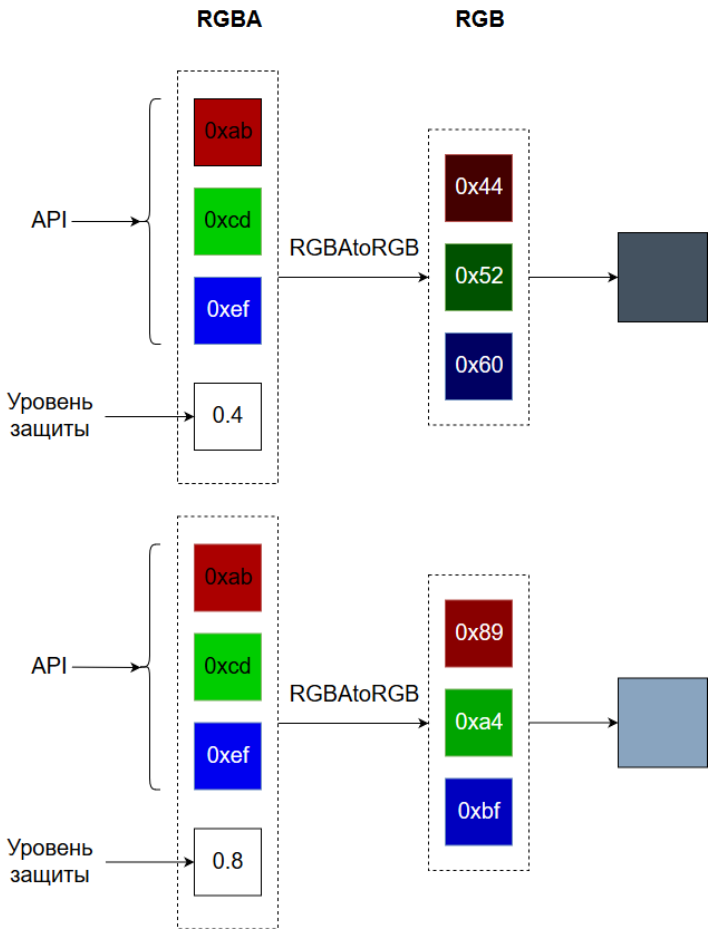

Рисунок 3.4 – Из RGBA в RGB

Непосредственно для всех последующих операций я использовал пакет Androguard [12], инструмент для реверс-инжиниринга и анализа apk файлов Android. На рисунке 3.4 представлена блок smali кода, полученный при помощи Androguard. Данный блок состоит из четырех инструкций, каждый из них представляет определенную операцию.

```
0 (0x00000000) invoke-direct v0, Ljava/lang/Object;-><init>()V
1 (0x00000006) invoke-virtual v0, v1, Landroid/net/http/SslError;->addError(I)Z
2 (0x0000000c) iput-object v2, v0, Landroid/net/http/SslError;->mCertificate Landroid/net/http/SslCertificate;
3 (0x00000010) return-void
```

Рисунок 3.4 – Пример API запроса в Androguard

Всего в smali около 218 инструкций и только 10 из них используют API вызовы (рисунок 3.5).

Опкод	Мнемоника
0x6e	invoke-virtual
0x6f	invoke-super
0x70	invoke-direct
0x71	invoke-static
0x72	invoke-interface
0x74	invoke-virtual/range
0x75	invoke-super/range
0x76	invoke-direct/range
0x77	invoke-static/range
0x78	invoke-interface/range

Рисунок 3.5 – Инструкции связанные с API

Список составлен для того чтобы отсеивать все остальные инструкции. Для построения последовательности API предлагается следующий алгоритм (рисунок 3.6)

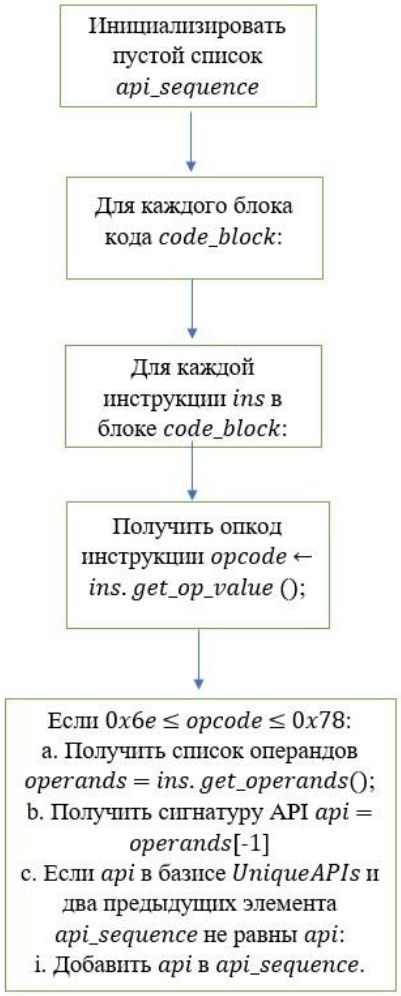

Рисунок 3.6 – Алгоритм

Далее следует сопоставление разрешения API вызову. Для данного этапа использовалась программа PScout [13].

```
Permission:android.permission.CHANGE_WIFI_STATE
1676 Callers:
…
<android.net.wifi.WifiManager: boolean requestBatchedScan(an-
droid.net.wifi.BatchedScanSettings)> ()
<android.net.wifi.WifiManager: boolean saveConfiguration()> ()
<android.net.wifi.WifiManager: boolean setWifiEnabled(boolean)>
<android.net.wifi.WifiManager: void disableEphemeralNet-
work(java.lang.String)> ()
```

Рисунок 3.7 – Отчет в PScout

Выходные данные имеют формат Java, необходимо преобразовать в smali (рисунок 3.8).

```
def to_smali_type(java_type):
    ret = None
    narrays = 0

    # проверить, является ли java_type массивом
    if java_type.endswith('[]'):
        narrays = java_type.count('[]')
        java_type = java_type.rstrip('[]')

    # если это класс
    if '.' in java_type:
        smali_type = 'L' + java_type.replace('.', '/') + ';'
    # если это простой тип
    elif java_type in PRIMITIVE_TYPES:
        smali_type = PRIMITIVE_TYPES[java_type]

    # если это массив - добавить [ перед типом
    ret = ('[' * narrays) + smali_type
    return ret
```

Рисунок 3.8 – Преобразование в Java

где, PRIMITIVE_TYPES – словарь сопоставления [14]. Пример преобразования показан на рисунке 3.9.

Java:

```
android.net.wifi.WifiManager: boolean setWifiEnabled(boolean)
```

Smali:

```
Landroid/net/wifi/WifiManager;->setWifiEnabled(Z)Z
```

Рисунок 3.8 – Преобразование в Java

Преобразование RGBA в RGB выглядит следующим образом (рисунок 3.9).

```
def rgba2rgb(self, rgba_color):
    alpha = rgba_color[-1]
    rgb_bg = [0, 0, 0]  # черный фон
    rgb_color = [
        ceil((1 - alpha) * rgb_bg[0] + alpha * rgba_color[0]),
        ceil((1 - alpha) * rgb_bg[1] + alpha * rgba_color[1]),
        ceil((1 - alpha) * rgb_bg[2] + alpha * rgba_color[2])
    ]
```

Рисунок 3.9 – Функция преобразования RGBA-RGB

2.5 Построение сверточной нейронной сети и реализация программного макета

Входные данные для нейросети это трехмерный тензор $W \times H \times D$, где W – ширина изображения, H –высота изображения, а D – глубина изображения.

Обучение происходит с учителем, подается на вход некоторое изображение X, представляющее собой преобразованное Android-приложение и подается метка y={0,1}, где 0 – вредоносное, а 1- легитимное. Обучение происходит в несколько эпох, циклов. После каждого цикла проходит тестовая выборка для выявления эффективности обучения.

Как показывает практика наиболее удачным шаблоном для сверточной сети является шаблон побеждавший в популярном соревновании ImageNet Large Scale Visual Recognition Challenge (OLSVRC). Далее показан шаблон с учетом терминологии Keras (рисунок 3.10)

```
INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL -> DROP]*2 -> FLAT -> [FC -> RELU -> DROP] -> FC
```

Рисунок 3.10 – Архитектура сверточной сети на Keras

где, слой DROP (от англ. Dropout) является слоем исключения, который представляет метод регуляции нейросети для предотвращения переобучения сети. Достигается это путем исключения нейронов параметром rate в слое DROP из проходящих вычислений.

Слой FLAT (от англ. Flatten) – данный слой преобразует входной объем. Например: 32x32x12 в 1x1x12288 (32 * 32 * 12 = 12288).

Архитектура имеет два CONV слоя расположенных перед POOL слоем. Таким образом расположение и наличие сразу двух CONV слоем является признаком глубоких сверточных сетей. Поскольку несколько подряд расположенных CONV слоев создают более сложные комбинации перед слоем POOL. Рекомендуется использовать множественные CONV вместо больших пространственных размеров входных фильтров. Это обусловлено

тем что при наличии множественных слоев пулинга возможно сократить количество входных параметров без изменения входных данных. На рисунке 3.11 показана выше представленная архитектура в среде Keras.

```
model = Sequential()
model.add(Conv2D(…))
model.add(Activation('relu'))
model.add(Conv2D(…))
model.add(Activation('relu'))
model.add(MaxPooling2D(…))
model.add(Dropout(…))

model.add(Conv2D(…))
model.add(Activation('relu'))
model.add(Conv2D(…))
model.add(Activation('relu'))
model.add(MaxPooling2D(…))
model.add(Dropout(…))

model.add(Flatten())
model.add(Dense(…))
model.add(Activation('relu'))
model.add(Dropout(…))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

Рисунок 3.11 – Реализация архитектуры сверточной сети на Keras

При использовании класса Sequential модель нейронной сети задается последовательным добавлением слоев. В последнем полносвязном слое (Dense) num_classes обозначает количество классов, в нашем случае два, легитимные и вредоносные. Визуальная схема нейронной сети представлена на рисунке 3.12.

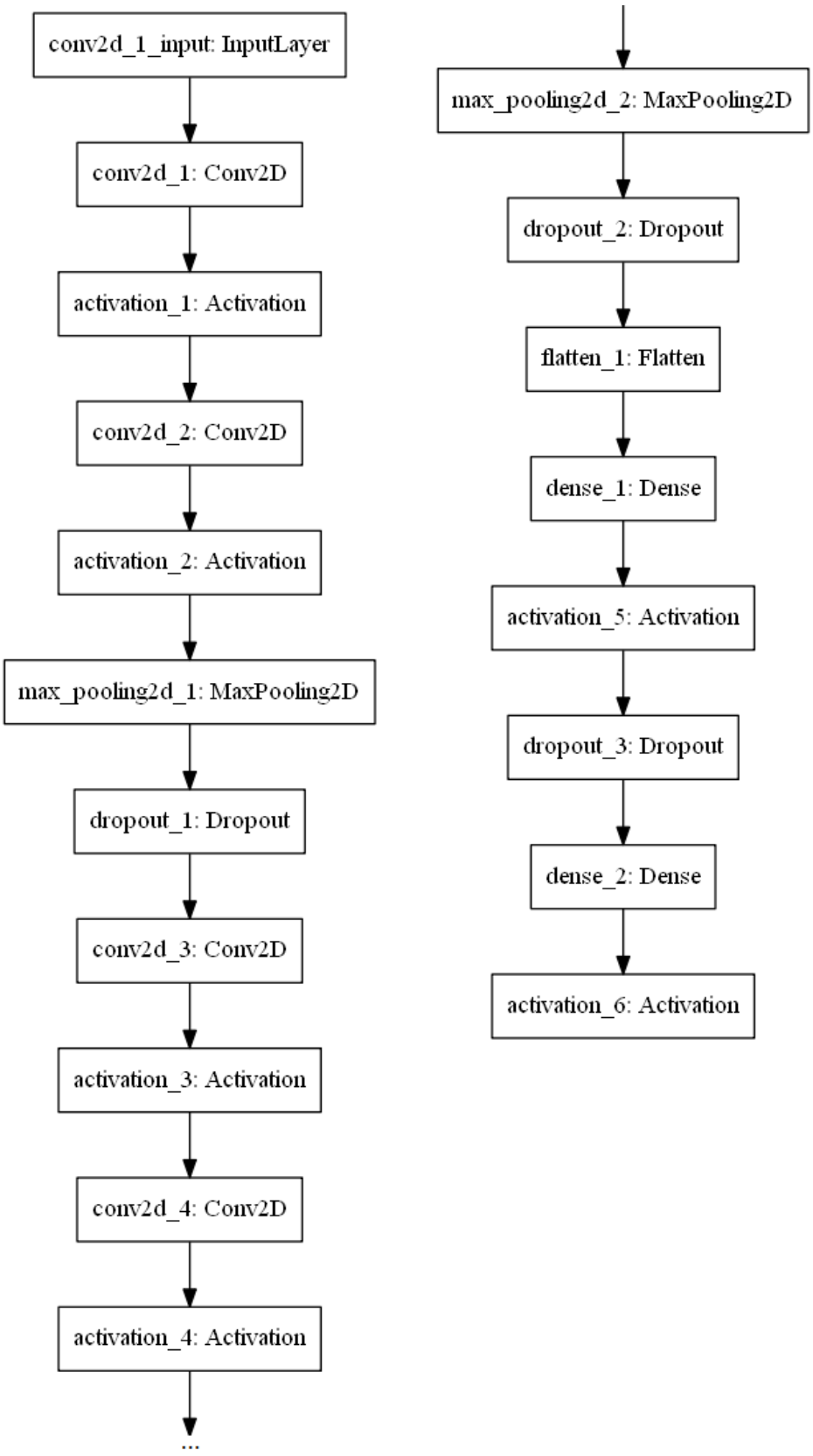

Рисунок 3.12 – Схема сверточной нейронной сети

Слои Conv2D, MaxPooling2D, Dense соответствуют слоям CONV, POOL и FC.

1 INPUT (InputLayer)	- размер изображения (input_shape).
2 CONV (Conv2D)	- количество фильтров (filters); - пространственный размер фильтра или размер восприимчивого поля (kernel_size); - шаг фильтра (strides); - нулевое смещение по краям изображения (padding).
3 POOL (MaxPooling2D)	- Пространственная протяженность (pool_size); - шаг (strides)
4 FC (Dense)	- количество нейронов (units)
5 Activation	- функция активации (activation)
6 DROP (Dropout)	- процент исключаемых нейронов (rate)
7 FLAT (Flatten)	- коэффициент скорости обучения (learning_rate) – определяет, как сильно корректируются значения параметров (весов и смещений) нейросети; - функцию потери (loss) – позволяет оценить отклонение значений, выдаваемых нейросетью от требуемых; - количество эпох (epochs) – количество полных прогонов элементов выборки через алгоритм обучения; - размер партии (batch_size) – количество элементов выборки, после прохождения которых происходит корректирование параметров нейросети, т.е. выполнение алгоритма обратного распространения - оптимизатор алгоритма обучения (optimizer) и его параметры и некоторые другие

Рисунок 3.13 – Гиперпараметры сверточной сети, в скобках указаны обозначения Keras

Финальным шагом преобразования API в изображение является построение файла с набором данных. Был выбран формат данных CIFAR-10 [15]. Набор данных представляет словарь Python (dict)

В качестве источника легитимных Android-приложений был взят набор PlayDrone APKs [16]. Для проекта я взял произвольно 320 приложений (таблица 3.1).

Таблица 3.1 – Количество легитимных приложений по категориям

Категория	Кол-во	Категория	Кол-во
Action	57	Personalization	52
Communication	45	Racing	43
Media	22	Photography	51
Music	27	Tools	23

А в качестве источника вредоносных приложений был использован проект Android Malware Dataset [17]. Для проекта я взял произвольно 341 приложение (таблица 3.2).

Таблица 3.2 – Количество вредоносных приложений по категориям

Категория	Кол-во	Категория	Кол-во
Ad	75	Leech	59
Bankbot	23	Spy	62
Fake Booster	39	Stealer	39
Fake Clock	44		

Таким образом у меня получилось 661 приложение из них 320 – легитимные, а 341 – вредоносные.

При использовании нейронной сети в качестве анализатора вредоносных приложений удалось достичь результата в 94% процента точности.

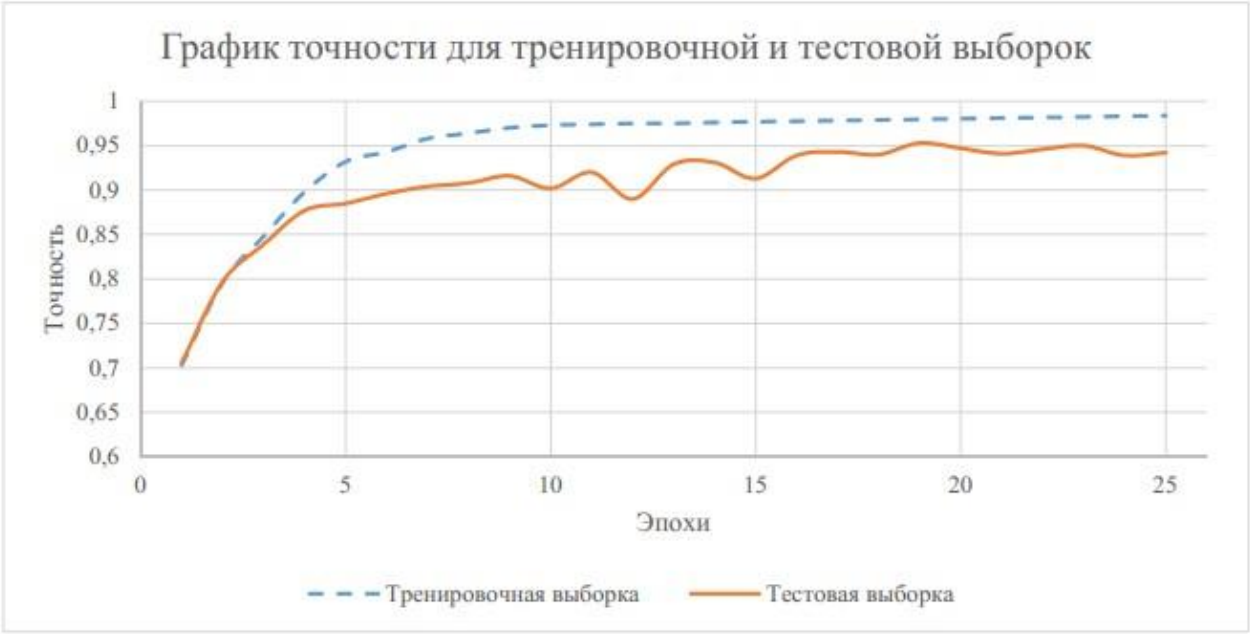

Рисунок 3.14 – График тренировочной и тестовой выборки

При наличии более мощных аппаратных ресурсов и больших размерах выборки возможно сократить количество циклов (эпох) обучения

Вывод

В данной главе был подробно описан метод по выявлению вредоносных Android-приложений с помощью сверточной нейронной сети. Также было описано оборудование для тестирования и обучения нейросети, и программное обеспечение в виде языка программирования Python 3, фреймворка TensorFlow и Keras.

Для представления данных нейросети был осуществлен метод по представлению пар API запросов в виде RGB изображения. Выборка из легитимных и вредоносных Android-приложений была собрана из открытых библиотек Playdrone-apk и Android Malware Dataset.

Также было подробно описан процесс разработки сверточной нейронной сети под анализ Android-приложений.

3 Технико-экономическое обоснование

В данном дипломном проекте рассматривается разработка программного продукта для компаний, в которых есть отдел информационной безопасности.

Основная идея разрабатываемого программного обеспечения – автоматизированный поиск уязвимостей с помощью искусственных нейронных сетей.

Технико-экономическое обоснование содержит:

- определение трудоёмкости разработки программного продукта (ПП);
- расчет затрат на разработку ПП;
- определение возможной цены разработанного ПП;
- оценку социально – экономических результатов функционирования ПП.

В стоимость разработки включаются следующие расходы:

- заработная плата разработчиков;
- отчисления на социальные нужды;
- материальные затраты;
- затраты на спецоборудование для разработки ПО;
- амортизационные расходы и т.д.

3.1 Трудоёмкость разработки ПП

Для определения трудоёмкости разработки ПП необходимо в первую очередь составить перечень всех основных этапов выполняемых работ. Следует выделить среди всех этапов логическое упорядочивание последовательности отдельных работ и выявление возможностей их параллельного выполнения с целью существенного сокращения общей длительности проведения разработки ПП.

Форма разделения работ по этапам с учетом трудоёмкости их выполнения приведена в таблице 4.1.

Таблица 4.1 – Распределение работ по этапам, оценка их трудоёмкости

Этапы разработки ПП	Вид работы на данном этапе	Трудоемкость разработки ПП, чел. час.
1 Планирование	Поиск и изучение сопутствующей литературы, изучение подобных программ и постановка задач.	32
2 Анализ требований	Создание технического задания.	24
3 Разработка алгоритма	Разработка алгоритма работы программы (блок схема).	32
4 Кодирование	Создание алгоритма решения на языке	64

	программирования.	

Продолжение таблицы 4.1

5 Отладка	Поиск и устранение ошибок, недоработок.	24
6 Тестирование	Проверка ПП при большом числе входных данных.	24
7 Проектирование	Разработка дизайна и функциональных характеристик программы.	48
8 Документация	Создания технической и пользовательской документации.	16
9 Внедрение	Настройка ПП под определенные условия использования.	8
10 Сопровождение ПП	Улучшение и оптимизация ПП после передачи в эксплуатацию.	72
Итого:		344

3.2 Расчет затрат на разработку ПП

Определение затрат на разработку ПП производится на основе существующей сметы, которая включает следующие статьи:

Статья «Оборудование» состоит из перечня оборудования, необходимого для разработки ПП. Расчет затрат на оборудование производится по форме, приведенной в таблице 4.2.

Статья «Материальные затраты» состоит из основных и вспомогательных материалов, энергии, необходимых для разработки ПП. Расчет затрат на материальные ресурсы производится по форме, приведенной в таблице 4.3.

Таблица 4.2 – Расчет затрат на оборудование

Наименование оборудования	Единица измерения	Количество	Цена за ед., тг	Сумма, тг
Ноутбук HP OMEN 17-w109ur	Штук	1	197 000,00	197 000,00
Роутер TP-LINK TL-WR820N	Штук	1	10 500,00	10 500,00
МФУ XEROX WorkCentre 3225DNI	Штук	1	89 900,00	89 900,00
Итого:				297 400,00

Таблица 4.3 – Расчет затрат на материальные ресурсы

Наименование материального ресурса	Ед. измерения	Количество	Цена за ед., тг.	Сумма, тг
Компьютерная мышь Defender Dacota MS-155 Nano	Штук	1	3 000,00	3 000,00
Бумага офисная А4 (500 штук) SVETOCOPY	Упаковка	1	1 400,00	1 400,00
Тетрадь (96 листов) "Иммитация дерева" Маяк	Штук	2	210,00	420,00
Блокнот А5, 50Л "STANDART" (HERLITZ)	Штук	2	400,00	800,00
Ручки "SUPER" GRIP-G	Штук	2	120,00	240,00
Итого:				5 860,00

Общая сумма затрат на оборудование и материальные ресурсы ($З_м$) определяется по формуле (4.1):

$$З_м = \sum_{i=1}^{n} P_i \times Ц_i, \quad (4.1)$$

где P_i – расход i-го вида материального ресурса, натуральные единицы;
$Ц_i$ – цена за единицу i-го вида материального ресурса, тг;
i – вид материального ресурса;
n – количество видов материальных ресурсов.

$З_м$ = 297 400,00 + 5 860,00 = 303 260,00(тенге).

Затраты на программное обеспечение представлены в таблице 4.4:

Таблица 4.4 – Затраты на программное обеспечение

Наименование	Название продукта	Кол-во	Стоимость единицы, тг	Общая стоимость, тг
Операционная система	Microsoft Windows 10 SP1	1	45000,00	45 000,00
Редактор кода	Sublime Text	1	30320,00	30 320,00

Итого:	75 320,00

Затраты на интернет. По тарифу КазахТелеком цена интернета составляет 4500 тг. за месяц. Так как для разработки ПП требуется 43 дня, имеет смысл рассчитать стоимость за два месяца:

Затраты на интернет:

4 500 * 2 = 9 000 тг.

3.3 Расчет затрат на электроэнергию

Затраты на электроэнергию. Так как для разработки ПП используется в основном электрооборудование, то следует рассчитать затраты на электроэнергию по специальной форме, приведенной в таблице 4.5.

Таблица 4.5 – Затраты на электроэнергию

Наименование оборудование	Паспортная мощность, кВт	Коэффициент использования мощности	Время работы оборудования для разработки ПП, ч	Цена электроэнергии, $\frac{тг}{кВт \cdot ч}$	Сумма, тенге
Ноутбук	0,3	0,7	168	18,32	646,33
Роутер	0,06	0,7	168	18,32	129,27
МФУ	0,7	0,9	36	18,32	415,50
Освещение	0,3	0,7	168	18,32	646,33
Итого затраты на электроэнергию					1 837,43
НДС 12%					220,49
Общая сумма					2 057,92

По тарифу АлматыЭнергоСбыт цена электроэнергии составляет 18,32 $\frac{тг}{кВт \cdot ч}$ (для юридических лиц).

Общая сумма затрат на электроэнергию $З_э$ рассчитывается по формуле (4.2):

$$З_э = \sum_{i=1}^{n} M_i \times K_i \times T_i \times Ц \ , \qquad (4.2)$$

где M_i – паспортная мощность i-го электрооборудования, кВт;

K_i –коэффициент использования мощности i-го электрооборудования (принимается K_i = 0,7 , 0,9);

T_i – время работы i-го оборудования за весь период разработки ПП ч (из таблицы 1);

Ц – цена электроэнергии, тг/кВт×ч;

i – вид электрооборудования;

n – количество электрооборудования.

$$З_э = 646,33 + 129,27 + 415,50 + 646,33 = 1\ 837,43 \text{ тг.}$$

С учетом НДС сумма составит:

$$З_э = 1\ 837,43 + (1\ 837,43 * 12\%) = 2\ 057,92 \text{ тг.}$$

3.4 Затраты на оплату труда

Для разработки программного обеспечения создана группа из трех человек. В которую входят: аналитик, разработчик и тестировщик.

Трудоемкость разработки составляет 344 часа / 8 = 43 рабочих дня.

В статью «Затраты на оплату труда» входят расходы на оплату всех работников, занятых разработкой ПП. Затраты на оплату труда рассчитываются по форме, приведенной в таблице 4.6.

Таблица 4.6 – Затраты на оплату труда

Категория работника	Квалификация	Трудоемкость разработки ПП, час	Часовая ставка, тг/ч	Сумма заработной платы за месяц, тг
Аналитик	Инженер-аналитик	344	852,27	293 180,88
Разработчик	Инженер-проектировщик	344	1 704,54	586 361,76
Тестировщик	Инженер-программист	344	681,81	234 542,64
Итого затраты на оплату труда:				1 114 085,28

Общая сумма затрат на оплату труда $З_т$ определяется по формуле (4.3):

$$З_{тр} = \sum_{i=1}^{n} ЧС_i \times T_i, \quad (4.3)$$

где $ЧС_i$ – часовая ставка i-го работника, тенге;

Ti – трудоемкость разработки ПП, чел×ч;

i – категория работника;

n – количество работников, занятых разработкой ПП.

Часовая ставка работника может быть рассчитана по формуле (4.4):

$$ЧС_i = \frac{ЗП_i}{ФРВ_i}, \quad (4.4)$$

где ЗПi – месячная заработная плата i-го работника, тг;

ФРВi – месячный фонд рабочего времени i-го работника, час.

$ЧС_{аналитик}$ = 150000,00 / 8 * 22 = 852,27 (тенге/час).

$ЧС_{разработчик}$ = 300000,00 / 8 * 22 = 1 704,54 (тенге/час).

$ЧС_{тестировщик}$ = 120000,00 / 8*22 = 681,81 (тенге/час).

З тр = (852,27 + 1704,54 + 681,81) * 344 = 1 114 085,28 тенге.

3.5 Расчет затрат по социальному налогу

Социальный налог. В статью «Социальный налог» включена сумма, рассчитываемая как 9,5% от затрат на оплату труда всех работников $З_т$, занятых разработкой ПП. При расчете необходимо учесть, что пенсионные отчисления (10% от $З_т$) не облагаются социальным налогом. Социальный налог можно рассчитать по формуле (4.5):

$$С_н = (ФОТ - ПО) * 0,095 \quad (4.5)$$

где ПО – отчисления в пенсионный фонд, они составляют 10% от ФОТ.

$$ПО = 1\,114\,000,00 * 0,1 = 111\,408,52 \text{ тенге}$$
$$С_н = (1\,114\,085,28 - 111\,408,52) * 0,095 = 92\,254,29 \text{ тенге}$$

3.6 Амортизация основных фондов и прочие затраты

Амортизация основных фондов.

В статье «Амортизация основных фондов» рассчитывается сумма амортизационных отчислений от стоимости оборудования и программного обеспечения (ПО), которые использовались в ходе разработки ПП. Амортизационные отчисления рассчитываются по форме, приведенной в таблице 4.7.

Таблица 4.7 – Амортизация основных фондов

Наименование оборудования и ПО	Стоимость оборудования и ПО, тг	Годовая норма амортизации, %	Эффективный фонд времени работы оборудования и ПО, ч/год	Время работы оборудования и ПО для разработки ПП, ч	Сумма, тенге
Ноутбук HP OMEN 17-	197 000,00	25	2016	344	4 104,17

w109ur					
Роутер TP-LINK TL-WR820N	10 500,00	20	2016	344	175,00

Продолжение таблицы 4.7

МФУ XEROX WorkCentre 3225DNI	89 900,00	20	2016	344	1 498,33
ОС Microsoft Windows 10 SP1 лицензия	45 000,00	20	2016	344	750,00
ПП Sublime Text	30 320,00	20	2016	344	505,53
Итого амортизация основных фондов					7 032,50

Общая сумма амортизационных отчислений определяется по формуле (4.6):

$$З_{АМ} = \sum_{i=1}^{n} \frac{Ф_i \times Н_{Ai} \times Т_{НИРi}}{100 \times Т_{эф}}, \quad (4.6)$$

где Фi – стоимость i-го ОФ, тг;

$Н_{Ai}$ – годовая норма амортизации i-го ОФ, %;

$Т_{НИР}$ –время работы i-го ОФ за весь период разработки ПП, ч;

$Т_{эф}$ – эффективный фонд времени работы i-го ОФ за год, ч/год;

i – вид ОФ;

n – количество ОФ.

$З_{АМ}$ = 197 000,00* 25 * 168 / 100 * 2016 = 4 104,17 (тенге).

$З_{АМ}$ = 10 500,00* 20 * 168 / 100 * 2016 = 175,00 (тенге).

$З_{АМ}$ = 89 900,00 * 20 *168 / 100 * 2016 = 1 498,33 (тенге)

$З_{АМ}$ = 45 000,00 * 20 *168 / 100 * 2016 = 750,00 (тенге).

$З_{АМ}$ = 30 320,00* 20 * 168 / 100 * 2016 = 505,53 (тенге).

3.7 Полная сметная стоимости разработки ПО

На основе всех представленных расчетов необходимо оформить смету расходов на разработку ПП согласно форме, которая приведена в таблице (4.8). На рисунке 4.1 продемонстрирована диаграмма рабочих расходов.

Таблица 4.8 – Полная смета затрат разработки ПП

№	Наименование статьи затрат	Всего, тг

1	Затраты на оборудование	297 400,00
2	Затраты на материальные средства	5 860,00
3	Затраты на программные средства	75 320,00
4	Затраты на электроэнергию	2 057,92

Продолжение таблицы 4.7

5	Затраты на оплату труда	1 114 085,28
6	Затраты на социальные налоги	92 254,29
7	Амортизация основных фондов	7 032,50
8	Прочие расходы (интернет)	9 000,00
ИТОГО:		1 603 009,07

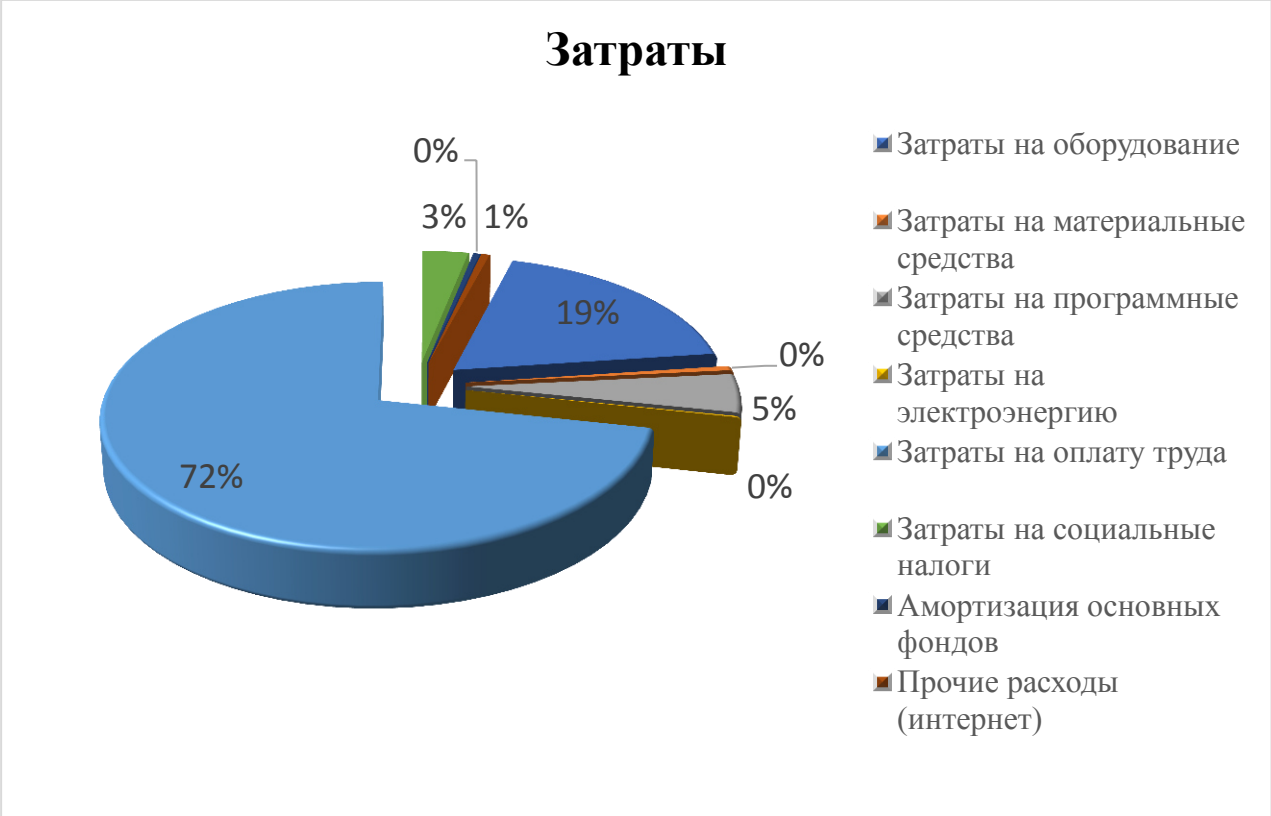

Рисунок 4.1 – Диаграмма затрат

3.8 Определение возможной (договорной) цены ПП

Величина возможной (договорной) цены ПП устанавливается на основе эффективности, качества и сроков её выполнения на уровне, отвечающем экономическим интересам заказчика (потребителя) и исполнителя.

Договорная цена $Ц_д$ для прикладных ПП рассчитывается по формуле (4.7):

$$Ц_д = З_{НИР}\left(1+\frac{Р}{100}\right) \quad (4.7)$$

где $З_{НИР}$ – затраты на разработку ПП, тг;

Р – средний уровень рентабельности ПО, (%). Данный параметр принят равным 20%.

$$Ц_Д = 1\ 603\ 009{,}07 * (1+20/100) = 1\ 923\ 610{,}88 \text{ тг.}$$

$$\text{Прибыль} = 1\ 603\ 009{,}07 * 0{,}2 = 320\ 601{,}81$$

Далее определяется цена реализации с учетом налога на добавленную стоимость (НДС), ставка (НДС) устанавливается законодательно. Налоговым Кодексом РК. На 2019 год ставка НДС установлена в размере 12%.

Цена реализации с учетом НДС рассчитывается по формуле (4.8):

$$Ц_Р = Ц_Д + Ц_Д * НДС \quad (4.8)$$

$$Ц_р = 1\ 923\ 610{,}88 + 1\ 923\ 610{,}88 * 0{,}12 = 2\ 154\ 444{,}18 \text{ тг.}$$

Рассчитанную возможную цену ПП можно округлить до 2 154 444,20 тенге.

Вывод

В данном разделе были произведены расчеты затрат на приобретение необходимого оборудования, материальных ресурсов и программного обеспечения для разработки сканера уязвимостей, включая расчет затрат на оплату труда.

Смета затрат на программное обеспечение составила 1 603 009,07 тенге. Возможная договорная цена программного продукта, с учетом НДС составила 2 154 444,18 тенге. Рентабельность (прибыль) программного обеспечения составляет 320 601,81 тенге.

4 Безопасность жизнедеятельности

4.1 Анализ условий труда

Организация безопасной корпоративной сети предполагает собой наличие различного компьютерного оборудования и программного обеспечения. Для организации рабочего процесса сотрудников необходимо обеспечить их определенными удобствами: просторным рабочим пространством, обеспечивающим комфортное совершение определенных действий и перемещений в ходе рабочего процесса; комфортными креслами, снижающими нагрузку на позвоночник; аспирационными системами, обеспечивающими вентиляцию воздуха и поддерживающими оптимальную комнатную температуру.

Рабочее помещение расположено на 6 этаже бизнес центра. Планировка помещения представлена на рисунке.

Рабочее помещение рассчитано для работы 2-х сотрудников (мужчины – 2), имеющих служебные места, включая меня.

Характеристики кабинета: длина L = 5 метров, ширина W= 4 метров, высота H = 3 метра. В кабинете имеется лишь одно окно, что является недостаточной мерой обеспечения вентиляционного процесса. Также присутствует вторая дверь, примыкающая к помещению с серверами. Исходные данные указаны в таблице 5.1.

Используемое оборудование и его характеристики:

- два персональных компьютера;
- принтер.

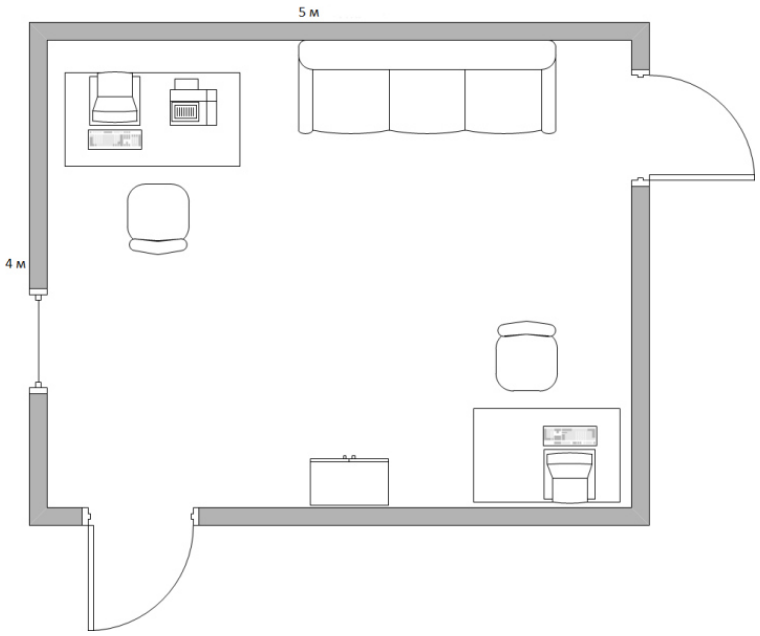

Рисунок 5.1 – Рабочее помещение

Таблица 5.1 – Исходные данные

Город		Алматы
Параметры помещения (L x W x H), м		5 x 4 x 3
Данные по оборудованию	кол-во, шт	2
	мощ. $P_{об}$, кВт/ч	0,65
	КПД, η	0,95
Данные по источникам света	мощ. N ос.уст., Вт/м^2	60
	вид ист. св.	лампы накаливания
Число сотрудников, из них	мужчины	2
	женщины	0
Окна	кол-во	1
	площадь 1 окна, м^2	1.5
	расположение	ЮЗ
	вид	жалюзи, метал. переплеты, одинарные, загрязнение незначительное
Расчетное время суток, ч.	14-15	
Температура в помещении, ^{0}C	летом	25
	зимой	19
Вид положения работы	Сидя	

4.2 Расчет тепловых нагрузок в помещении

В помещениях различного назначения действуют в основном тепловые нагрузки, возникающие снаружи помещения (наружные); а также тепловые нагрузки, возникающие внутри зданий (внутренние).

Наружные тепловые нагрузки характеризуются определенными составляющими:

– теплопоступления или теплопотери в результате разности температур снаружи и внутри здания через стены, потолки, полы, окна и двери.

– разность температур снаружи здания и внутри него летом является положительной, в результате чего имеет место приток тепла снаружи вовнутрь помещения; и наоборот – зимой эта разность отрицательна и направление потока тепла меняется;

– теплопоступления от солнечного излучения через застекленные площади; данная нагрузка проявляется в форме ощущаемого тепла;

– теплопоступления от инфильтрации.

В зависимости от времени года и времени суток наружные тепловые нагрузки могут быть положительными. Теплопоступления и теплопотери в результате разности температур определяются по формуле 1.1:

$Q_{огр} = V_{пом} * X_о * (t_{Нрасч} – t_{Врасч})$, Вт (1.1), где

$V_{пом}$ – объем помещения, $м^3$;

$V_{пом} = 5*4*3=60$ $м^3$;

$X_о$ – удельная тепловая характеристика, $Вт/м^3*^0С$;

$X_о= 0,33$ $Вт/м^3*^0С$;

$t_{Нрасч}$ – наружная температура (параметр А). Для холодного периода – средняя температура самого холодного месяца в 13 часов, для теплого периода – средней температуре самого жаркого месяца в 13 часов.

$t_{Врасч}$ – внутренняя температура, выбирается с учетом комфортных условий или технологических требований, предъявляемых к производственным процессам.

Для теплого времени года:

$t_{Нрасч} = 29,4$ $^0С$

$t_{Врасч} = 25$ $^0С$

$Q_{огр}= 60*0,33*4,4=87,1$ Вт

Для холодного времени года

$t_{Нрасч} = -9$ 0С

$t_{Врасч} = 19$ 0С

$Q_{огр} = 60*0,33*|-28| = 554,4$ Вт

Избыточная теплота солнечного излучения в зависимости от типа стекла почти до 90% поглощается средой помещения, остальная часть отражается. Максимальная тепловая нагрузка достигается при максимальном уровне излучения, которое имеет прямую и рассеянную составляющие. Интенсивность излучения зависит от ширины местности, времени года и времени суток.

Теплопоступление от солнечного излучения через остекление определяется по формуле 1.2:

$Q_р = (q^I F^I_o + q^{II} F^{II}_o) * βс.з$ (1.2), где

q^I, q^{II} – тепловые потоки от прямой и рассеянной солнечной радиации, Вт/м^2;

F^I_0, F^{II}_0 – площади светового проема, облучаемые и необлучаемые прямой солнечной радиацией, м^2 ;

βс.з. – коэффициент теплопропускания. Для штор-жалюзи с металлическими пластинами:

βс.з. = 0,15

При отсутствии наружных затеняющих козырьков, ребер и т. д. для периода облучения остекления солнцем, когда его лучи проникают через окно в помещение $F^I_0 = F^{II}_0 = F_0 = 0$, (1.3):

$$Q_р = q^{II} F_0 * βс.з = q_{вр} * K^T_1 * K_2 * βс.з * n * S_{0;}$$

$q_{вп}$; $q_{вр}$ – тепловые потоки от рассеянной радиации, Вт/м^2. Для широты в 43^{0}СШ (Алматы) после полудня в 14-15 ч. при расположении ЮЗ:

$q_{вр}$ = 97 Вт/м^2;

$F_o = nS_o = 1·1,5 = 1,5$ м^2 – площадь светового проема (n – число окон; S_o – площадь 1 окна);

K_1 – коэффициент затемнения остекления переплетами (K^T_1 – для проемов в тени).

$K^T_1 = 1,28$;

K_2 – коэффициент загрязнения остекления.

$K_2 = 0,95$.

Тогда:

$Q_р = 97*1,28*0,95*0,15*1,5 = 26,5$ Вт

Внутренние нагрузки в жилых, офисных или относящихся к сфере обслуживания помещениях слагаются в основном из тепла:

- выделяемого людьми;
- выделяемого лампами и осветительными, электробытовыми приборами;
- выделяемого компьютерами, печатающими устройствами фотокопировальными машинами пр.;

Теплопоступления от людей зависит от интенсивности выполняемой работы и параметров окружающего воздуха. Тепло, выделяемое человеком, складывается из ощутимого (явного), то есть передаваемого в воздух помещения путем конвекции и лучеиспусканий, и скрытого тепла, затрачиваемого на испарение влаги с поверхности кожи и из легких.

Летом при 24 ^{0}С один мужчина выделяет явного тепла 67 Вт, а общего – 102 Вт. Женщина выделяет 85% от нормы тепловыделений взрослого мужчины. Тогда выделение явного тепла в помещении составит:

$Q_л^я = 67*2 = 134$ Вт

А выделение общего тепла:

$Q_л^о$ = 102*2 = 204 Вт

Зимой при 18 ^{0}С один мужчина выделяет явного тепла 89 Вт, а общего – 104 Вт. Тогда выделение явного тепла в помещении составит:

$Q_з^я$ = 89*2 = 188 Вт

А выделение общего тепла:

$Q_з^о$ = 104*2 = 208 Вт

Теплопоступление от осветительных приборов, оргтехники и оборудования рассчитывается следующим образом. Теплопоступление от ламп определяется по формуле (5):

$$Q_{осв} = \eta \cdot N_{осв} \cdot F_{пол}, \ \ Вт \qquad (1.4)$$

где η – коэффициент перехода электрической энергии в тепловую (для лампы накаливания η=0,92-0,97);

$N_{осв}$ – установленная мощность ламп (N=60 Вт/м^2);

$F_{пол}$ – площадь пола:

$F_{пол}$ = 5*4=20 м^2

Тогда:

$Q_{осв}$ = 0,94*60*20 = 1128 Вт

Тепло, выделяемое производственным оборудованием, определяется по формуле (6):

$$Q_{об} = N_{уст} \cdot K \qquad (1.5)$$

$Q_{об}$ = 0,65 * 2 * 0,95 * 10^3 = 1,24 кВт.

Теплопритоки, возникающие за счёт находящейся оргтехники – это 30% мощности оборудования:

$Q_{орг}$ = 1 * 0,3 * 0,3 * 10^3 = 0,09 кВт

4.3 Расчет теплового баланса помещения

На основании выполненных расчетов составим баланс теплопоступлений в помещении:

$$Q_{изб} = Q_р + Q^я + Q_{осв} + Q_{об} + Q_{орг} + Q_{огр}$$

Лето: $Q_{изб}^Л$ = 26,5 + 134 + 1128 + 1240 + 90 + 87,1 = 2,705 кВт

Зима: $Q_{изб}^З$ = 26,5 + 188+ 1128 + 1240 + 90 + 554,4 = 3,226 кВт

Так как тепловой баланс для лета больше зимнего теплового баланса, то рассчитаем теплонапряженность воздуха по формуле:

$$Q_н = \frac{Q_{изб.лето} \times 860}{V_{пом}}$$

$$Q_н = \frac{2{,}705 \cdot 860}{60} = 38{,}7 \text{ ккал/м3} \ .$$

При $Q_н$ >20ккал/м^3, Δt =8 °С,

при $Q_н$ <20ккал/м^3, Δt =6 °С.

Определение количества воздуха, необходимое для поступления в помещение:

$$L = \frac{Q_{изб} \times 860}{C \times \Delta t \times \gamma}$$

$$L = \frac{2{,}705 \cdot 860}{0{,}24 \cdot 8 \cdot 1{,}206} = 1004{,}6 \text{ м3/час}$$

где C=0,24 ккал/(кг · °C) – теплоемкость воздуха,
γ=1,206 кг/м^3 – удельная масса приточного воздуха.

Определение кратности воздухообмена:
$N = L/V_{пом} = 1004{,}6/60 = 16{,}7$ час$^{-1}$

4.4 Выбор кондиционера. Схема расположения

Исходя из полученных результатов, для удаления лишнего тепла и очистки воздуха нужно использовать вентиляционную систему, которая способна обеспечить требуемую подачу воздуха L=1004,6 (м3/ч). В данном случае подойдет Кондиционер Daikin FAQ100B/RR100BV Nord-40T. Данный кондиционер способен обеспечить подачу воздуха до 1380 м3 /ч.

Технические характеристики:
- мощность (охлаждение): 3.56 кВт;
- мощность (обогрев): 3.56 кВт;
- потребляемая мощность при охлаждении: 2600 Вт;
- потребляемая мощность при обогреве: 2800 Вт;
- обслуживаемая площадь: 80 м2;
- уровень шума внутреннего блока: 45 дБ;
- уровень шума внешнего блока: 53 дБ;
- цвет: белый.

Характеристики подключения:
- вентиляция: 1380 м3/час;
- класс энергоэффективности при охлаждении/обогреве: A++/A++;
- электропитание, В/Гц/Ф:220 Вт;
- энергопотребление в режиме ожидания не более 1 Вт.

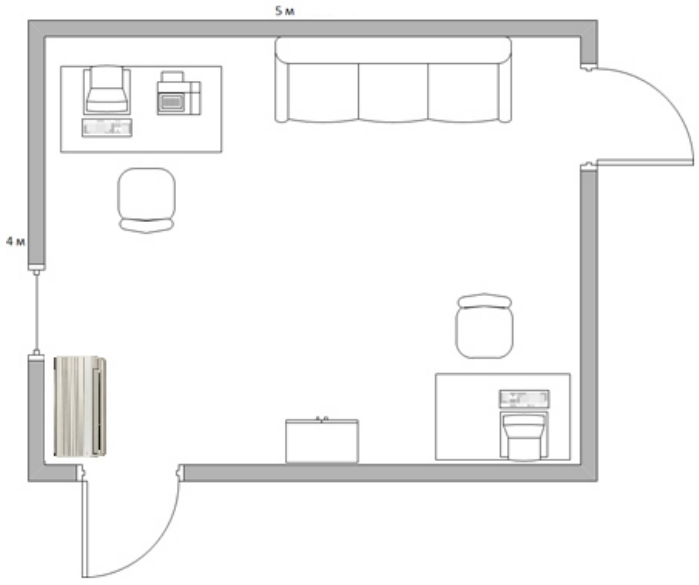

Рисунок 5.2 – Расположение кондиционера

Вывод

В данном разделе дипломного проекта были рассмотрены и рассчитаны воздушные показатели для благоприятных условий труда, а именно, тепловые нагрузки в помещении, наружные и внутренние. Исходя из расчетов, была выбрана модель кондиционера с соответствующими характеристиками. По расчетам можно наблюдать, что избыток тепла летом составляет 2,705 кВт, что делает необходимым установку достаточно мощной системы кондиционирования. Для создания хороших условий труда необходим один кондиционер с подачей воздуха не менее 1004 м3 /ч, в текущем случае был выбран кондиционер Daikin FAQ100B/RR100BV Nord-40T с подачей воздуха до 1380 м3 /ч.

Заключение

В данном дипломном проекте были представлены результаты метода выявления вредоносных Android-приложений с помощью выработанного подхода основанного на методе глубокого обучения сверточной нейронной сети. Судя по данным ресурса Statcounter доля ОС Android на рынке портативный устройств составляет 74%. Поэтому защита данной ОС посредством нейронных сетей остается актуальной.

Были рассмотрены и кратко описаны основные виды нейронных сетей глубокого обучения такие как: рекуррентная сеть, сети долгой краткосрочной памяти и сверточная нейронная сеть. Исходя из достоинства быстро распознавать изображения было принято решение использовать сверточную нейронную сеть.

Также была рассмотрена структура ОС Android, ее компоненты. Вкратце изложено из чего состоит Android-приложение и какой его компонент при декомпиляции будет использован.

Основываясь на выбранном подходе и среде анализа Android-приложений был реализован метод анализа на вредоносное программное обеспечение путем представления пар API запросов в виде изображения и последующим анализом сверточной нейронной сетью.

Эффективность в обнаружении вредоносного приложения данным методом составляет 94%. При наличии более мощных аппаратных ресурсов возможно повысить скорость обучения и существенно увеличить темп анализа Android-приложений.

Список литературы

1 Mobile Operating System Market Share Worldwide [Электронный ресурс]
// StatCounter.com. – URL: http://gs.statcounter.com/os-market-share/mobile/worldwide. (дата обращения: 05.05.2019)

2 Отчет компании Kaspersky Lab // SucureList.com – URL: https://securelist.com/kaspersky-security-bulletin-2018-statistics/89145/ (дата обращения: 24.05.19).

3 Барский А. Б. Б26 Нейронные сети: распознавание, управление, принятие решений. — М.: Финансы и статистика, 2004. — 176 с: ил. — (Прикладные информационные технологии). ISBN 5-279-02757-X.

4 Курс «Основы Нейронных Сетей » // NeuralNet.Info – URL: neuralnet.info/chapter/основы-инс (дата обращения: 12.05.19).

5 Курс «Neural Network Zoo» // The Asimov Institute – URL: http://www.asimovinstitute.org/neural-network-zoo/ (дата обращения: 12.05.19).

6 Базовый курс «Как работает LSTM » // Stanislav Isakov – URL https://neurohive.io/ru/osnovy-data-science/lstm-nejronnaja-set/ (дата обращения: 17.05.19).

7 Статья «Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество» // Habr.com Иван Голиков – URL: https://habr.com/ru/post/348000/ (дата обращения: 19.05.19)

8 N. Peiravian, X. Zhu. Machine Learning for Android Malware Detection Using Permission and API Calls [Текст] // ICTAI '13 Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence. – Ноябрь 2013. –С. 300-305.

9 R. A. Nix. Applying Deep Learning Techniques to the Analysis of Android APKs [Электронный ресурс] // digitalcommons.lsu.edu: Louisiana State University Digital Commons, 2016. – URL: https://digitalcommons.lsu.edu/cgi/viewcontent.cgi?article=5443&context=gradschool_theses. – (дата обращения: 10.04.2018)

10 Курс «Введение в разработку приложений для встроенных систем на платформе Intel Atom» // Константин Амелин, Олег Граничин, Владимир Кияев, Александр Корявко, Роман Лучин. – 2012 URL: https://www.intuit.ru/studies/courses/10617/1101/info (дата обращения: 03.05.19).

11 Open Source MurMurHash3 – URL: https://github.com/aappleby/smhasher/wiki/MurmurHash3 (дата обращения: 10.04.2018)

12 Open Source Androguard – URL: https://github.com/androguard/androguard (дата обращения: 17.05.2019)

13 Open Source PScout – URL:https://github.com/zyrikby/PScout (дата обращения: 17.05.2019)

14 Open Source TyperMethodsAndFields – URL:https://github.com/JesusFreke/smali/wiki/TypesMethodsAndFields (дата обращения: 17.05.2019)

15 Курс «CIFAR-10» // Alex Krizhevsky – URL: https://www.cs.toronto.edu/~kriz/cifar.html (дата обращения: 17.05.2019)

16 Playdrone-apk-e8 [Электронный ресурс] // archive.org: Internet Archive. – URL: https://archive.org/details/playdrone-apk-e8. – (дата обращения: 25.05.2019).

17 Android Malware Dataset [Электронный ресурс] // amd.arguslab.org. – URL: http://amd.arguslab.org/. – (дата обращения: 24.05).