Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
Волгоградский государственный технический университет

Факультет Электроники и вычислительной техники
Кафедра Системы автоматизированного проектирования и ПК

Утверждаю
и.о. зав. кафедрой САПРиПК,
д.т.н., проф.
_____ М. В. Щербаков
(подпись) (инициалы, фамилия)
«_____» _____ 2017

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной работе бакалавра на тему
(наименование вида работы)
Портирование сверточной нейросети на ARM архитектуру с ограниченными вычислительными ресурсами и ресурсами памяти

Автор _____ Мельников Тимофей Алексеевич
(подпись и дата подписания) (фамилия, имя, отчество)
Обозначение ВРБ-40 461 806-10.27-10-17.81-81
(код документа)
Группа ИВТ-461
(шифр группы)
Направление 230100 Информатика и вычислительная техника
(код по ОКСО, наименование направления, программы)

Руководитель работы _____ к.т.н. А. В. Катаев
(подпись и дата подписания) (инициалы и фамилия)
Консультанты по разделам:

(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)

Нормоконтролер _____ к.т.н. О. А. Шабалина
(подпись и дата подписания) (инициалы и фамилия)

Волгоград, 2017

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Волгоградский государственный технический университет

Кафедра Системы автоматизированного проектирования и ПК

Утверждаю
и.о. зав. кафедрой САПРиПК,
д.т.н., проф.
_____ М. В. Щербаков
(подпись) (инициалы, фамилия)
«_____» _____ 2017

Задание на выпускную работу бакалавра
(наименование вида работы)
Студент Мельников Тимофей Алексеевич
(фамилия, имя, отчество)
Код кафедры 10.27 Группа ИВТ-461

Тема Портирование сверточной нейросети на ARM архитектуру с ограниченными вычислительными ресурсами и ресурсами памяти
Утверждена приказом по университету от « 1 » ноября 2017 № 1548-ст
Срок представления готовой работы _____
(дата, подпись студента)
Исходные данные для выполнения работы
задание, выданное научным руководителем с кафедры САПРиПК, утвержденное приказом ректора

Содержание основной части пояснительной записки

Введение

1 Обзор технологий и фреймворков

Выводы

2 Реализация СНС с целочисленной арифметикой

Выводы

3 Оценка разработанного решения

Выводы

Заключение

Список использованных источников

Приложение А - Техническое задание

Перечень графического материала

1) 1: Название работы
2) 2: Проблема
3) 3: Цель и задачи
4) 4-7: Аналоги
5) 8: Требования
6) 9: Проектирование. Use Case Diagram
7) 10: Проектирование. Sequence Diagram
8) 11-13: Реализация
9) 14: Выводы

Руководитель работы _____ к.т.н. А. В. Катаев
(подпись и дата подписания) (инициалы и фамилия)

Консультанты по разделам:

(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
Волгоградский государственный технический университет
Кафедра «Системы автоматизированного проектирования и ПК»

Утверждаю
и.о. зав. кафедрой САПРиПК,
д.т.н., проф.
_____ М. В. Щербаков
(подпись) (инициалы, фамилия)
«_____» _____ 2017

Портирование сверточной нейросети на ARM архитектуру с ограниченными вычислительными ресурсами и ресурсами памяти

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ВРБ-40 461 806-10.27-10-17.81-81

Листов 51

Научный руководитель
к.т.н., доцент каф. САПРиПК
_____ А. В. Катаев
«_____» _____ 2017

Нормоконтролер
к.т.н., доцент каф. САПРиПК
_____ О. А. Шабалина
«_____» _____ 2017

Исполнитель
студент группы ИВТ-461
_____ Т. А. Мельников
«_____» _____ 2017

Волгоград, 2017

Аннотация

Документ представляет собой пояснительную записку к выпускной работе бакалавра на тему «Портирование сверточной нейросети на ARM архитектуру с ограниченными вычислительными ресурсами и ресурсами памяти», выполненную студентом группы ИВТ-461, Мельниковым Тимофеем Алексеевичем.

В данной работе рассмотрена возможность реализации алгоритмов машинного обучения, в частности прямой проход сверточной нейронной сети, на устройстве с ограниченными вычислительными ресурсами и ресурсами памяти.

Объём пояснительной записки составил 51 страниц и включает 23 рисунков и 2 таблицы.

Содержание

Введение

Задачи обработки и анализа аналоговой информации являюся одиними из самых сложных в IT-индустрии. Долгое время такие задачи решались алгоритмами, которые требовали огромных аппаратных ресурсов при малой точности результата. На протяжении последних десяти лет стремительно растет и развивается прикладная область математики цель которой, изучение и развитие искусственных нейронных сетей [1]. Актуальность разработок и исследований в данной области оправдывается применением НС в различных сферах деятельности. Это автоматизация процессов анализа объектов, образов, уневерсализация управления, прогнозирование, создание экспертных систем, анализ неформализованной информации и многое другое. В частности, в данной дипломной работе используются нейронные сети для классификации и обнаружения объектов на изображении.

Наиболее существенным недостатком НС является их требовательность к вычислительным ресурсам и ресурсам памяти. Частично данная проблема решается использованием сверточных нейронных сетей, которые, в виду особенностям логики работы, позволяют в разы сократить ресурсы потребляемые нейронной сетью [2].

Однако, не только искусственные нейронные сети являются трендом IT-идустрии, активно развивается коцепция интернета вещей. Диапазон встраиваемых технологий простирается от концепции умных зданий до промышленной консолидации. Совмещение встраиваемых систем и искусственных нейронных сетей позволяет иначе взглянуть на решение нетревиальных задач, таких как автономное управление автомобилем [3].

В связи с вышесказанным целью данной дипломной работы является внедрение фреймворка машинного обучения на ARM-устройство и последующая оптимизация его работы. На основе проделанной работы необходимо сделать вывод о эффективности и рентабельности данного решения.

Для достижения поставленной цели необходимо решить следующие задачи:

– Изучить фреймворки глубокого машинного обучения;

– Разработать консольное приложение для реализации прямого прохода нейронной сети;

– Оптимизировать использование оперативной памяти и реализовать загрузку весов по мере использования;

– Разработать клиент-серверное приложение, демонстрирующее результат работы.

В первом разделе пояснительной записки описаны фрейворки машинного обучения. Далее приведено обоснование выбора фреймворка darknet.

Во втором разделе описаны используемые модели нейронных сетей и алгоритм прямого прохода.

Третей раздел посвящен разворачиванию фреймворка на ARM-устройсто и оптимизации работы алгоритма прямого прохода. Так же описана разработка клиент-серверной части для визуализации работы приложения.

1 Обзор фреймворков машинного обучения

Данные раздел содержит справочную информацию, технические особенности и функциональные возможноти фреймворков глубоко машинного обучения и их сравнение. Раздел содержит обоснование выбора фреймворка Darknet для встраивания и оптимизации на мобильном пк C.H.I.P.

Из всего множества фрейворков были выделены Caffe, Torch7, Darknet, как наиболее зрелые, функционально полные и широко используемые.

1.1 Caffe

Caffe представляет собой фреймворк, разработанный учеными и практиками, с прозрачной и гибкой архитектурой для глубокого обучения и построения эталонных моделей. Фреймворк распространяется под BSD-лицензией и является c++ библиотекой. Так же реализованы python и MATLAB обертки для универсализации обучения и развертывания глубоких моделей. Caffe используется на промышленных компаниях и в медиацинтрах, обрабатывая 40 миллионов изображений в день на Titan GPU (примерно 2.5 милисекунд на изображение).

Caffe поддерживается и разрабатывается университетом Беркли, а именно центром BVLC.

1.1.1 Основные характеристики Caffe

Caffe представляет полный набор инструментов для обучения, тестирования, настройки и разработки моделей с подробной документацией и примерами. Поэтому обучиться использовать фреймворк можно довольно быстро. Возможность использования GPU делает Caffe одним из самых быстрых фреймворков, что позволяет

его использовать в промышленном секторе. Такие показатели достигнуты благодаря особенностям описаным ниже.

Caffe является модульным программным обеспечением. Что позволяет легко добалять новые форматы данных, слои и функции потерь. В фреймворке уже реализовано множество слоев и функций потерь, что позволяет реалзовавать нейронную сеть для задачь различных предметных областей и категорий.

В Caffe представление и реализация разделены. Для описания модели в Caffe используется конфигурационный файл в формате protobuf. Caffe поддерживает сетевые архитектуры в форме произвольно ориентированных ациклических графов. Важной деталей является то, что после создания экземпляра модели Caffe выделяется ровно столько памяти, сколько необходимо для работы сериализованной нейронной сети и для хранения адреса объекта [4].

В Caffe используется полное тестовое покрытие. Каждый модуль имеет собственный набор тестов. Модуль будет принят, только после прохождение всего набора тестов. Это позволяет эффективно оптимизировать модули и гарантирует стабильную работу фреймворка.

Caffe содержит предворительно обученные модели для академических целей и некоммерческого использования. Доступны сверточные НС с архитектурой "AlexNet"и вариации данной НС, обученные на базе данных ImageNet [5]. Так же доступны реккурентные модели [6].

1.1.2 Архитектура Caffe

Caffe сохраняет и передает данные в четырехмерных массивах, которые названы блобами. Блобы представляют унифицированный интерфейс для работы памятью, содержащий пакеты ихображений (или других данных), параметров или обновлений параметров. Блобы скрывают вычислительные издержки смешанной работы CPU и GPU, выполняя синхронихацию по нере необходимости. Память выделяется по требованию (лениво), что позволяет эффективней ее использовать.

Модели сохраняются как буфер, использующий протокол Google (Google Protocol Buffers), который имеет ряд достоинств: минимальный размер строки при сериализации, эффективная сериализация, высокая читабельность в текстовом виде и удобные интерфейсы работы на нескольких языках. Необходимые для обучения огромные массивы данных храняться в базах данных LevelDB. Google Protocol Buffers и LevelDB обеспечивают пропускную способность в 150 Мб/с.

Слой в Caffe представляет собой структуру соответствующую формальному определению слоя: он принимает на вход один или несколько блобов и выдает один или несколько блобов результатом. Caffe предоставляет полный набор типов слоев для глубокого обучения, включая сверточный, pooling слой, inner products слой, нелиности, такие как выпремленная линейная и логическая, слои потерь, таких как softmax и hinge. Настройка слоя требует минимальных усилий в виду композиционного построения сетей.

Caffe обеспечивает функциональность для любого направленного ациклического графа слоев, позволяя корректно выполнять прямой и обратный проход. Модели Caffe — это сквозные системы машинного обучениям.

1.1.3 Приемущества Caffe

От других современных фреймворков глубокого обучения Caffe отличается следующими качествами:

– Реализция полностью основана на C++, что облегчает интеграцию с встраиваемыми системами. CPU режим позволяет использовать фреймворк без специализированного GPU.

– Готовые модели позволяют не тратить время и ресурсы на обучение. Важным пунктом является подробная документация для сериализации и использования моделей.

1.2 Torch7

Torch7 — это универсальный математический фреймворк и библиотека машинного обучения, которая имеет оболочку для языка программирования Lua. Его цель — предоставить гибкую среду для проектирования и обучения моделей глубокого обучения. Гибкость достигается с помощью Lua, так как он является очень легким скриптовым языком. Эффективная реализация низкоуровневых числовых процедур, используя OpenMP и CUDA, позволяет фрейморку достич выской производительности. Фреймворк имеет простой Lua-интерфейс, что позволяет легко подключать его к стороннему программному обеспечению.

1.2.1 Основыне характеристики Torch7

Структура фрейворка имеет три основных преимущества:

– она облегчает разработку численных методов;

– фреймворк легко расширяем (включая использование сторонних библиотек);

– высокая скорость работы фреймворка.

Второе преимущества достигается за счет выбранных разработчиками технологий. Скрипровый (интерпретируемый) язык с хорошим API-интерфейсом для C обеспечивает фреймворку гибкост в разработке и не накладывает ограничения на его расширяемость. Так как, язык высокого уровня делает процесс разработки программы более простым и понятным, чем язык низкого уровня. К тому же, интерпретируемость позволяет быстро и легко реализовывать различные идеи в интерактивном режиме. Хороший API-интерфейс сохраняет функциональные возможности из разных библиотек, так как становиться прослойкой между универсальной структурой на языке Lua и различными структурами используемых библиотек на языке C.

Высокая скорость работы достигается благодаря компилятору JIT (Just In Time). На данный момет Lua является самым быстрым интерпритируемым языком. Lua разрабатывался для легкого внедрения в приложения, написанные на C. Поэтому представляет большое C-API на основе виртуального стека, для передачи значений между Lua и C. Это унифицирует интерфейс для C/C++ и делает обертывание библиотек тривиальным.

Lua предназначен для использования в качестве мощного, легкого скриптового языка обладающими всеми необходимыми выразительными средствами. Он реализован как библиотека, которая написана на чистом C (точнее на подмножестве ANSI C и C++). Lua сочетает простой процедурный синтаксис с мощными конструкциями описания данных на основе ассоциативных массивов и расширяемой семантики. Lua динамически типизируется, выполняется путем интерпретации байт-кода для виртуальной машины на основе регистров и имеет автоматическое управление памятью с инкрементной сборкой мусора, что делает его идеальным для настройки, написания сценариев и быстрого прототипирования.

Lua предлагает хорошую поддержку объектно-ориентированного программирования, функционального программирования и программирования, управляемого данными. Основным типом Lua является таблица, которая реализует ассоциативные массивы очень эффективным способом. Ассоциативный массив — это массив, который может индексироваться не только числами, но и любыми другими типами данных язка. Таблицы не имеют фиксированного размера, они динамически изменяемы и могут использоваться как "виртуальные таблицы"над другой таблицей, что позволяет имитировать парадигмы объектно-ориентированного программирования. Таблицы являются единственным, но очень мощным механизмом структурирования данных в Lua. Torch7 использует таблицы для простого, равномерного и эффективного представления обычных массивов, таблиц символов, кортежей, очередей и других структур данных. Lua также использует таблицы для представления пакетов [7].

Lua и Python очень схожи как по структурированию данных, так и по стилю программирования. Если говорить о популярности в сообществе, то Python опережает Lua из-за огромного количества поставляемых библиотек. Однако разработчики выбрали Lua по ряду других причин, которые, в виду специфики фреймворка, являются ключевыми. Во-первых, интеграция Lua с C очень проста. За несколько часов любая библиотека на C или C++ может стать библиотекой Lua. Во-вторых, Lua предоставляет эффективные возможности встраивания. Что бы преобразовать прототип в финальный продукт требуется не много дополнительной работы. В-третьих, Lua обладает высокой производительностью благорадя интерпритатору LuaJIT, который выдает производительность на уровне C. Еще одним преимуществом Lua является переносимость. Lua написан на чистом ANSI C, его можно скомпилировать для любых устройств (сотовые телефоны, встроенные процессоры в FPGA, процессоры DSP и др.) [8].

1.2.2 Структуры используемых типов данных

Ключевой сущностью в Torch7 является класс Tensor, поставляемый автономной C-библиотекой Tensor. Данный класс расширяет базовый набор типов Lua, чтобы реализовать эффективную работу с многомерными массивами. Большинство пакетов Torch7 или сторонних пакетов, зависящих от Torch7, реализуют собственный класс Tensor для представления сигналов, изображений, видео и других объектов, что упрощает интегрирование различных библиотек. Библиотека Torch Tensor предоставляет множество классических операций (включая операции линейной алгебры), которые реализованны и оптимизированны на C, используются SSE интсрукции для Intel платформ. Опцианально можно использовать высокопроизовидельные реализации операций линейной алгебры в библиотеке BLAS. Так же данная библиотека поддерживает инструкции OpenMP и вычисления на CUDA GPU.

1.2.3 Пакеты Torch7

На данный момент Torch7 имеет 7 основных пакетов:

– torch: основной пакет Torch7. Обеспечивает фреймворк классом Tensor, облегчает сереализацию и другие базовые функции;

– lap и plot: представляют стандартные функции для создания, преобрзования и визуализации объектов Tensor. Пример работы показан на рисунке 1

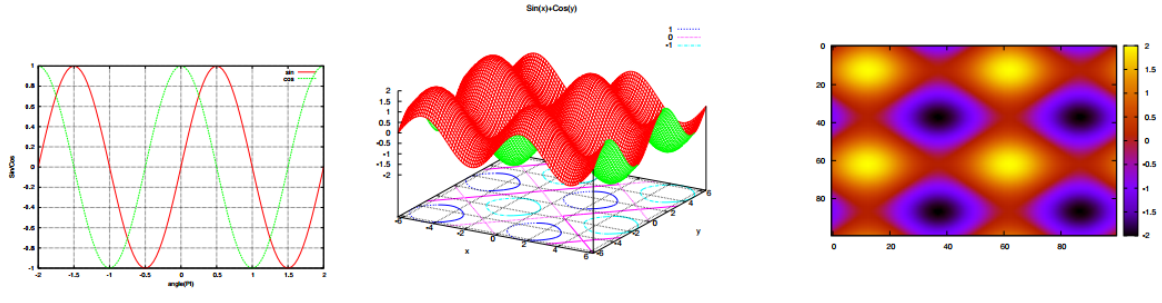

Рисунок 1 — Графики, полученные с помощую пакета plot фреймворка Torch7. Слева: простые синусоидальные функций. В центре: Поверхность, хранящаяся в 2D Tensor. Справа: Матричный график, построенный с использованием карты тепла

– qt: предаставляет интерфейс работы Torch7 с Qt. Реализует конвертацию Tensor в QImage и наоборот. Отлично подходит для быстрого создания интерактивных демонстраци с кроссплатформенным графическим интерфейсом.

– nn: предоставляет набод стандартных модулей для создания нейронной сети. В пакет так же входит набор контейнерных модулей, которые можно использовать для определения произвольно направленных графов. Явное описание графа позволяет избежать сложности с анализатором графов или любого другого компилятора промежуточного уровня.

На практике нейронная сеть представляет собой последовательные графы, либо графы с шаблонными витвлениями и рекурсиями. На рисунке 2 показано создание многослойного перцептрона, используя пакет nn.

Каждый модуль или контейнер имеет стандартные функции для вычисления выходного состояния, обратного распространения производных входов и внутренних параметров. Для нейронной сети, приведенной на рисунке 2, вызов этих функций показан на рисунке 3.

– image: пакет обработки изображений. Данный пакет поставляет стандартные функции работы с изображениями (сохранение, загрузка, маштабирование, вращение, конвертация цветовых пространст, свертка и др.).

– optim: компактный пакет, который обеспечивает фреймворк методами оптимизации. В него входят реалиция наклонного спуска, сопряженного градиента и алгоритма Бройдена — Флетчера — Гольдфарба — Шанно (BFGS).

– unsup: содержит алгоритмы обучения без учителя, такие как K-means, разреженное кодирование и автокодеры.

В дополнение к основным доступен постоянно растущий список сторонних пакетов. К примеру, mattorch, который обеспечивает двухсторонний интерфейс между матричным форматом Matlab и форматом Tensor или parallel, который предоставляет функции разветвления и исполнения Lua-кода на локальных или удаленных машинах, используя механизм сериализации Torch7. Этот список постоянно растет, поскольку Lua упращаяет интерфейс любой билиотеки C.

1.3 Darknet

Darknet является фреймворком машинного обучения с открытым исходным кодом, написанным на C и CUDA. Он прост в установке

```
1  mlp = nn.Sequential()
2  mlp:add(nn.Linear(100,1000))
3  mlp:add(nn.Tanh())
4  mlp:add(nn.Linear(1000,10))
5  mlp:add(nn.SoftMax())
```

Рисунок 2 — Создание многослойного перцептрона, используя пакет nn

и поддерживает вычисления как на центральном процессоре, так и на графическом.

1.3.1 Основные характерискики Darknet

Darknet один из немногих фреймворков машинного обучения, который не имеет обязательных зависимостей. Что позволяет быстро разворачивать его на встраиваемых системах. На ряду с встроенным функционалом, Darket поставляется с двумя опциональными зависимостями:

– OpenCV — для предоставление более широкого спектра поддерживаемых форматов изображений;

– CUDA — для вычислений на GPU.

Обе не являются обязательными для установки фреймворка.

Еще одним важным преимуществом фреймворка является независимость от архитектуры системы. Darknet полностью написан на C, что делает его универсальным, а его интеграцию в встраиваемы системы или в специализированное оборудование простой и понятной.

В оригинальном виде фреймворк, поставляемый разработчиками, представляет консольное приложения для работы с нейронными сетями. С помощью него можно проектрировать, обучать, тестировать нейронные сети типовых топологий. В список функций так же входит визуализация модели классификации и обучение реккурентных моделей. Однако, конфигурация файлов исходных кодов спроектирована специально для предоставляения возможности компиляции необходимых модулей в библиотеку [9]. Поэтому фреймворк можно встраивать как нативную библиотеку в любой пользовательский проект.

```
1  Y = mlp:forward(X)              -- вычисление активации Y = f(X)
2  E = loss:forward(Y,T)           -- вычислить функцию потери E = l(Y,T)
3  dE_dY = loss:updateGradInput(Y,T)       -- вычислить градиент dE/dY = dl(Y,T)/dY
4  dE_dX = mlp:updateGradInput(X,dE_dY)    -- вычилить ошибку, вплодь до dE/dx
5  mlp:accGradParameters(X,dE_dY)  -- вычислить градиенты по весам: dE/dW
```

Рисунок 3 — Вычисление выходного состояния, обратного распространения производных входов и внутренних параметров

Важной особенностью фреймворка является оптимизация работы с памятью и с вычислительными ресурсами. Это позволяет работать с визуальными задачами даже на устройствах с ограничеммыми ресурсами памяти. Darnket имеет две эффективные реализации сверточных нейронных сетей: сети с бинарными весами и XNOR-сети. В сетях с бинарными весами фильтры аппроксимируются двоичными значениями, что приводит к экономи памяти в 32 раза. В XNOR-сетяк как фильтры, так и входные данные для сверточных слоев являются двоичными. XNOR-сети реализуют свертки, используя в основном бинарные опирации. Это приводит к ускорению сверточныз операций в 58 раз и экономии памяти в 32 раза. Данная оптимизация позволяет запускать современный нейронные сети на центральных процессорах в режиме реального времени. Если говорить о точности работы, то классификация модели AlexNet на 2.9 % меньше у сети с бинарными весами оп сравнению с оригинальной реализацией. Метод используемый в сетях с бинарными весами и XNOR-сетех превосходит новейшие сетевые методы бинаризации (BinaryConnect и BinaryNets) на 16 % (тест профодился на классификаю, используя модель ImageNet) [10].

1.3.2 Используемые структуры данных

Ключивыми типами данных в Darknet являются структуры network и layer. Структура layer представляет собой объект для параметров слоя сети. Данная структура имеет общий интерфейс для всях типов слоев, поэтому обладает большим набором параметров. Для расчета выходов и градиента слоев, структура предоставляет два указателя на функции forward и backward соответственно. Реализации данных функций находятся в файлах исходных кодов у каждого типа слоя. Такая модульная структура позволяет быстро добавлять новый типы слоев и компактно реализовывать операции работы с нейронной сетью. В целом, слои представляют двунаправленный связанный список, что соответствует логике работы с нейронными сетями.

Структура network определяет абстрактную модель для хранения внутренних параметров нейронной сети. Как и Caffe, Darknet разделяет представление и реализацию. Это реализуется разделением данных модели на конфигурационный файл, в котором определены внутренние параметры, и на бинарный файл с весами модели. Конфигурационный файл имеет строковый формат и представляет собой описание параметров нейронной сети, параметров обучения, параметров слоев и их последовательность. Формат конфигурационного файла представлен на рисунке 4

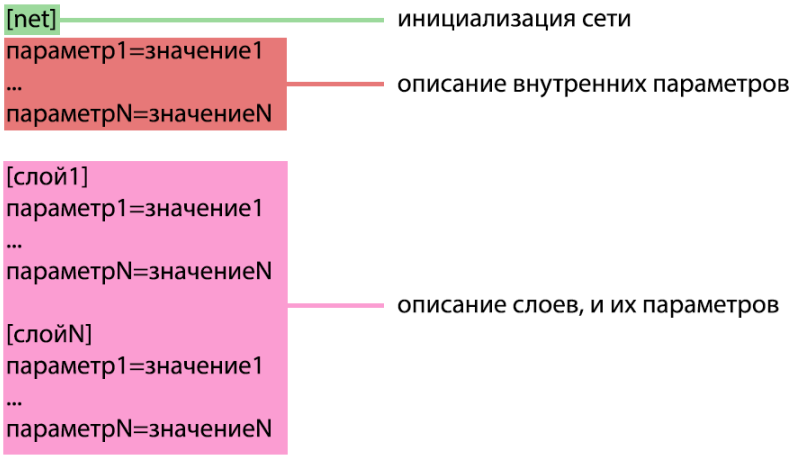

Рисунок 4 — Формат конфигурационного файла нейронной сети

Основной структурой данных в фреймворке является динамический массив. Веса, изображения, строковые таблицы храняться в одномерном массие, который обернут в структуру соответствующего типа данных. Данный подход позволяет сократить издержки работы с памятью.

Выводы

Для использования сверточной нейронной сети на системе с ограниченными вычислительными ресурсами и ресурсами памяти необходимо, что бы фреймворк, поставляющий данные функции удовлетворял следующим условиям:

– высокопроизворительные вычисления;

– оптимизированная работа с памятью;

– минимальное число зависимостей.

Рассмотринные выше фреймворки, используя различные терхнологии и алгоритмы, обеспечивают высокую производительность своих реализаций. Caffe использует библиотеку BLAS (ATLAS, Intel MKL, OpenBLAS) для векторных и матричных вычислений, Lua в совокупности с технологиями SSE, OpenMP позволяют Torch показывать высокую скорость работы, бинаризация ядер в Darknet, позволяет использовать быстрые бинарные операции для расчетов.

Если говорить о оптимизации работы с памятью, то аппроксимация фильтров и входов в Darknet позволяет значительно уменьшить объем выделяемой памяти. На рисунке 5 сравнение бинарной свертки и свертки с двойной точностью.

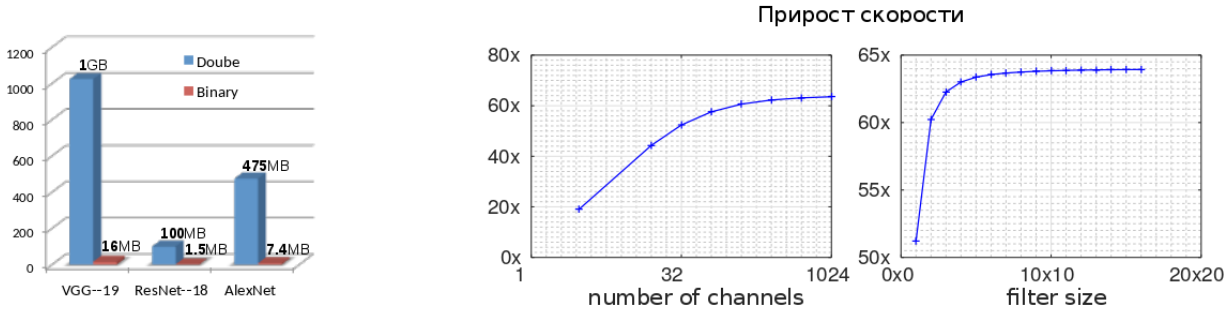

Рисунок 5 — Эффективность использования памяти и вычислений. a – выделяемая память для весов различных архитектур, b — ускорение в зависимости от числа каналов, c — ускорение в зависимости от размера фильтра

Caffe и Torch имеют достаточно большое количество зависимостей. Это объясняется желанием максимально ускорить процессы обучения и прохода нейронных сетей, однако наклабывает ограничения на срециализированное оборудование и оборудование с ограниченными запасами физической памяти.

Суммировав все показатели, можно сделать вывод, что Darknet является лучшим вариантов для разворачивания на маломощном ARM-устройтве.

2 Используемые алгоритмы и модели

2.1 Нейронные сети: основные положения

Основой любой нейронной сети являются однотипные, простые элементы, которые представляют собой упрощенную модель нейронов мозга. Далее по тексту термин “нейрон” используется для определения ячейки нейронной сети — искусственного нейрона. В соответствии с клетками головного мозга, которые могут быть возбужденными или заторможенными, нейрон характеризуется состоянием в момент прохода нейронной сети. Каждый нейрон обладает набором синапсов и одним аксоном. Синапсы являются однонаправленными связями, которые связывают конкретный нейрон с выходами группы других нейронов. В свою очередь, аксон передает сигнал нейрона на синапсы нейронов, расположенных на следующем слое. На рисунке 6 представлен общий вид нейрона. Каждый синапс описывается величиной синаптической связи, иными словами, синапсы характеризуются весом w_i, который является аналогом электрической проводимости в клетках мозга.

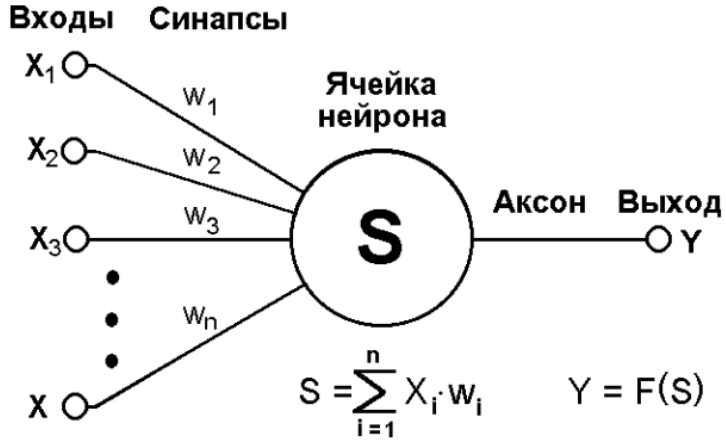

Рисунок 6 — Исскуственный нейрон

Состояние нейрона в момент прохода нейронной сети определяется как взвешенная сумма его входов:

$$s = \sum_{i=1}^{n} x_i w_i \qquad (1)$$

Выходом нейрона является функция от его состояния:

$$y = f(x) \qquad (2)$$

Функция f должна обладать свойством нелинейности. Это необходимо для создания многослойных нейронных сетей. Если в НС используется пороговая функция, то смысла в ее многослойности нет, так как такая сеть эквивалентна сети с одним скрытым слоем и с весовой матрицей единственного слоя [11].

Нелинейная функция f именуется активационной функцией нейрона. На данный момент существует огромное количество видов активационных функций. На рисунке 7 показаны некоторые из них.

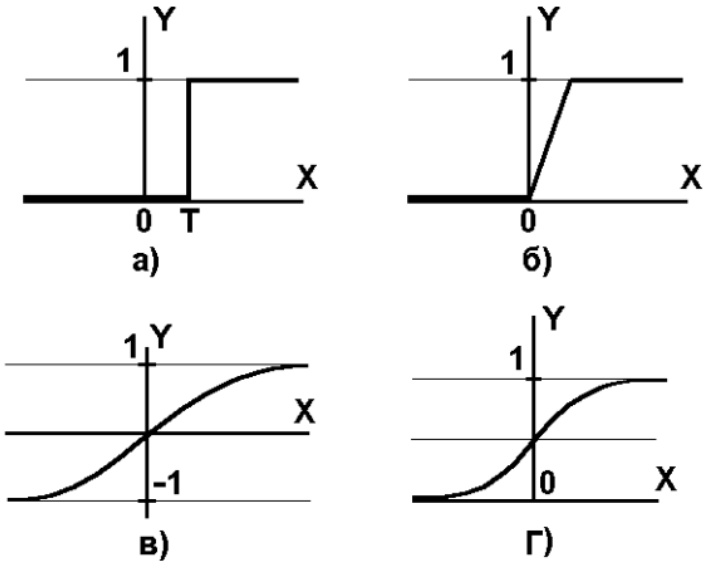

Рисунок 7 — а) функция единичного скачка; б)линейный порог (гистерезис); в) сигмоид – гиперболический тангенс; г) сигмоид – формула (3)

Одной из самых первых используемых активационных функций является логистическая функция или сигмоид (функция имеет

S-образный вид):

$$f(x) = \frac{1}{1 + e^{-\alpha x}} \quad (2)$$

Чем меньше параметр α, тем функция становится более пологой. В пределе при $\alpha = 0$ сигмоид вырождается в горизонтальную прямую в значении 0.5. Если увеличивать a, то сигмоид преобразуется в функцию единичного скачка в точке $x = 0$. Значение данной активационной функции лежит в интервале [0, 1]. Популярность функции обеспечивает простота ее производной, которая используется при обучении НС.

$$f'(x) = \alpha f(x)(1 - f(x)) \quad (2)$$

Логистическая функция дифференцируема на всей оси абсцисс. Это свойство используется в некоторых алгоритмах обучения. Также, сигмоид усиливает слабые сигналы лучше, чем большие, это позволяет избежать перенасыщения от больших сигналов, так как области определения больших сигналов соответствуют пологому наклону функции.

Если говорить про обработку сигналов НС, то, зачастую, они обрабатываются параллельно. Это достигается с помощью объединения большой группы нейронов в слои и соединения определенным образом нейроны разных слоев. Существуют конфигурации, где нейроны одного слоя соединены между собой. Данная конфигурация обрабатывается послойно.

На рис 8 изображен простейшая конфигурация нейронной сети — трехнейронный перцептрон. Пусть нейронной этой НС используют активационную функцию в виде скачка.

На n входов поступают некоторые сигналы, которые распространяются на три нейрона, образующие скрытый слой НС. Каждый нейрон выдает сигнал:

$$y_j = f\left[\sum_{i=1}^{n} x_i w_{ij}\right], j = 1 \ldots 3$$

Из весовых коэффициентов можно составить матрицу W, в которой wij – вес i-того входного сигнала в j-том нейроне. Тогда, процесс прохода НС описывается в матричной форме следующим образом:

$$Y = F(X\,W) \qquad (2)$$

где X — вектор входных сигналов, Y — вектор выходных сигналов, F(XW) — активационная функция, выполняющаяся над каждым элементом вектора XW [12].

Теоретически количество слоев (глубина) и количество нейронов в них (высота), используемых в НС, не ограниченно, но фактически ограничения накладывают вычислительные мощности устройства, на котором выполняется обработка НС. Но чем сложнее НС, тем масштабнее задачи она может решить.

Структура НС зависит от сложности задачи. Оптимальные конфигурации для некоторых типов задачи уже реализованны и описаны, например в [13]. Если же задача не является типовой, то разработчик самостоятельно генерирует модель, в зависимости от сложности задачи, размера обучающей выборки и вычислительных ресурсов. При этом необходимо учитывать основополагающие принципы: качество модели напрямую зависит от количества нейронов сети, плотности связей между ними и количеством слоев; сложность алгоритмов функционирование сети (например, введение нескольких типов синапрос, использование непороговых активационных функций) влиет на производительность НС. Задача поиска оптимальной конфигурации для той или иной задачи является отдельным направлением нейрокомпьютерной науки.

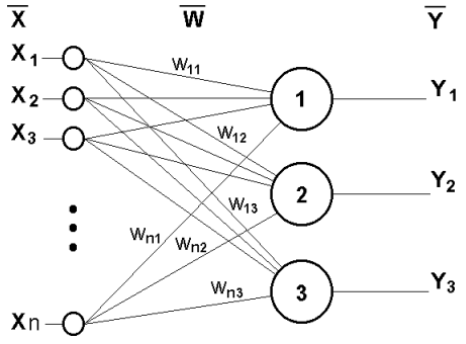

Рисунок 8 — Однослойный перцептрон

Синтез нейронной сети напряму зависит от типа решаемой задачи, поэтому список подробных рекомендаций составить затруднительно. В большинстве случаев оптимальный вариант выбирается эмпирическим методом.

Очевидно, что функционирование нейронной сети напрямую зависит от величины синаптических связей между нейронами. Поэтому, после нахождения конфигурации нейронной сети, разработчик должен найти оптимальные значения всех переменных весовых коэффициентов (некоторые веса могут быть постоянными).

Описанный процесс называется обучений нейронной сети, он является ключевым при сознании НС. От того, насколько хорошо он будет выполнен, зависит качество решений поставленных задачь перед нейронных сетей. На этапе обучение кроме качества поиска весов важное место занимает такой параметр как время обучения. Эти два параметра обратно пропорциональны: чем лучше подобраны веса, тем больше затрачено времени на обучение.

Сущетвует два варианта обучения: с учителем и без него. В первом случае, при обучении предоставляются как входны сигналы, так и желаемые выходные. Далее обучение представляет собой алгоритм подгонки весов, таким образом, что бы желаемы выходные сигнали совпадали с выходными сигналами НС. Во втором случае, выходы генерируется нейронной сетью, а при обучении учитываются только входные и производные от них сигналы.

Существующие алгоритмы машинного обучения делятся на два типа: детерминистские и стохастические. В первом случае подбор оптимальных весом представляет собой четкую последовательность, во втором — подчинен некоторому случайному процессу.

Необходимо сказать, что среди классификаций НС важное место занимают бинарные и аналоговые сети. Первые используют двоичные сигналы, в результате чего выход каждого из нейров принимает одно из двух значений: логический ноль ("заторможенное"состояние) или логическая единица ("возбужденное"состояние). К такой классификации относится перцептрон, описанные выше. Его активационная функция является пороговой, значение которой либо 0 либо 1. В аналоговых

сетях выходное значение нейронов может быть непрерывным, это реализуется использованием в качестве активационных непрерывные функции, например сигмоид.

Еще одна классификация разделяет НС на синхронные и асинхронные. Первый случай предполагает изменение состояния только одного нейрона в каждый момент времени. Во втором случае изменние происходит одновременно у группы нейронов, как правило, у всего слоя. Ход времени в НС представлен последовательным выполненим однотипных действий над нейронами. В данной главе будут рассмотрены только синхронные НС.

Обычно, сети классифицируют по числу слоев. На 9 показана НС полученная добавлением еще одного слоя, состоящего из двух нейронов, в НС, изображенную на 8. Слои, которые не являются входными и выходными, называются скрытыми.

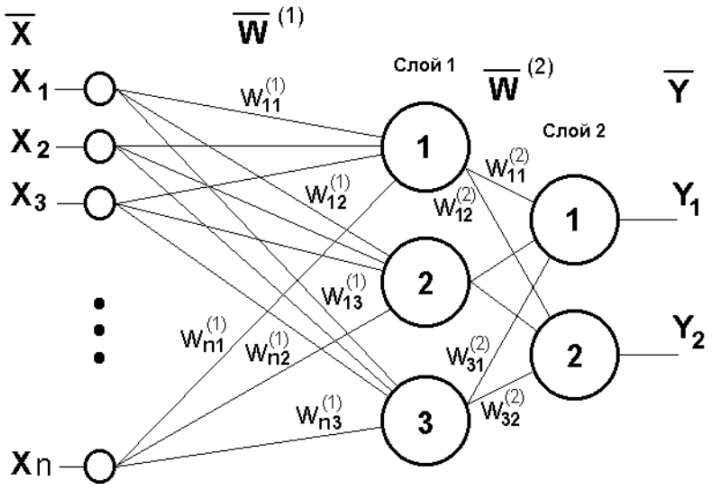

Рисунок 9 — Двухслойный перцептрон

Бывают случае, когда нелинейность используется еще и в синаптических связях. Большинство современных сетей используют формулу (1) для вычисления значения нейронна, однако, для эффективного решения некоторых задач используется другая запись, например:

$$s = \sum_{i=1}^{n} x_i^2 w_i \qquad (2)$$

или даже

$$s = \sum_{i=1}^{n} x_i^2 x_{((i+1) \bmod n)} w_i \qquad (2)$$

Главное, что бы разработчик понимал, какие цели он преследует при наделении нейрона подобной связью и какие ограничения на нейрон накладываются. Введение таких нелинейность увеличивает вычислительную мощность НС, другими словами, позволяет уменьшить число нейронов и связей без потери качества работы[8].

При обучении НС учитывается не только время процесса и качество обучения. Помимо этих параметров необходимо подобрать пороговое значение Т. Из рисунка 7 видно, что, в общем случает, Т может принимать произвольное значение. То же самое относиться и к центральной точке сигмоиды, положение которой изменяется по оси X влево или вправо. В общем случии каждая активационная функция имеет параметр, который необходимо подобрать при обучении. В связи с этим формула (1) должа выглядеть следующим образом:

$$s = \sum_{i=1}^{n} x_i w_i - T \qquad (11)$$

Что бы добавить данное смещение, необходимо добавить еще один вход, которые меет синаптическую связь со всеми нейронами слоя. На этот вход всегда "возбужденный"сигнал. Присвоим такому входу номер 0. Тогда

$$s = \sum_{i=1}^{n} x_i w_i - T \qquad (11)$$

где $w_0 = -T, x_0 = 1.$

Из чего следует, что отличие формулы (1) от формулы (12) в способе нумерации входов.

Все задачи, которая решает НС можно свести к классификации. Грубо говоря задача НС обпределить к какому классу принадлежит группа входных сигналов, находящихся в n-мерном пространстве. С

математической точки зрения процесс представляет собой разбиение гиперпространство гиперплоскостями на области.

К каждой области принадлежит отдельный класс. Максимальное число классов для НС перцептронного типа не превышает 2^m, где m — число выходов сети. Однако существует ограничение на формы гиперплоскостей, иначе говоря, не все нейронные сети могут разделить n-мерное пространство на необходимое количество классов.

Например, однослойный перцептрон, с одним нейроном, изображенный на рисунке 10 не способен разделить двумерное пространство на две полуплоскости так, что бы классифисировать входные сигналы на классы A и B (см. 1).

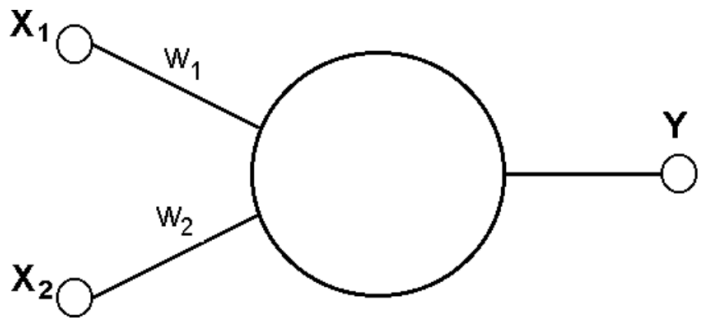

Рисунок 10 — Однонейронный перцептрон

Таблица 1 — Классификация XOR

x1	x2	A	B
0	0	•	
0	1		•
1	0		•
1	1	•	

Сеть, представленная на рисунке 10 описывает следующие уравнение:

$$x_1 w_1 + x_2 w_2 = T \qquad (11)$$

Данное уравнение является прямой, которая не способна разделить плоскольсть таким образом, что бы группа входных сигналов x_1, x_2 пренадлежали необходимым классам. На рисунке 11 показана работа НС.

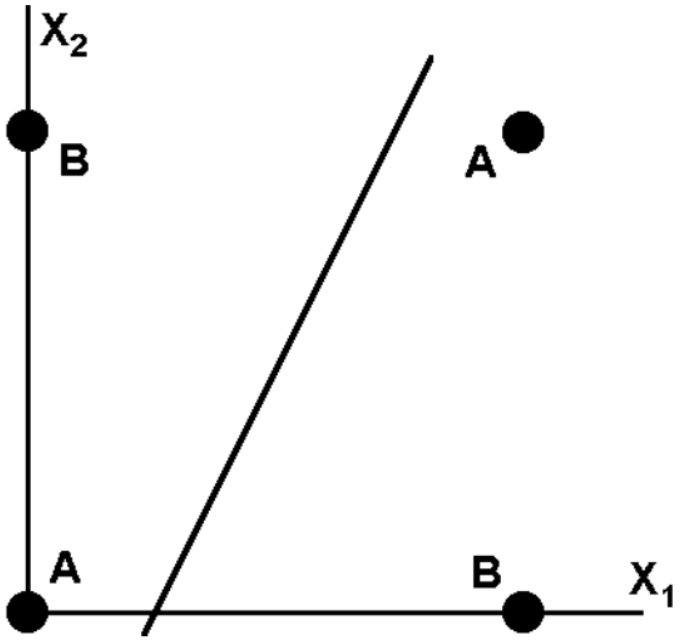

Рисунок 11 — Визуальное представление работы НС с рисунка 10

Таблица 1 является таблицей истинности для логической функции исключающего ИЛИ. Невозможность реализовать данную функцию, используя однонейронный перцептрон, получила название проблемы исключающего ИЛИ.

Задачи, которые не решаются однослойной сеть, называются линейно неразделимыми [13]. Для решения таких задачь используются нейронные сети сбольшим количеством скрытый слоев. Однако, и в таких случаях корректное разделение на классы не гарантируется. Как было сказано раньше, конфигурация НС это итеративный эмпирический процесс.

После обзора теоритических основ нейроной сети, можно более подробродно рассмотреть алгоритм обучения с учителем, на основу взят перцептрон, изображенный на рисунке 8.

Алгоритм выглядит следующим образом:

1) Проинициализировать весовые коэффициэны случайными значениями.

2) Подать на вход вектор вхожных сигналов, вычислить выходные сигналы.

3) Если выход совпарает с желаемыми значениями, перейти на шаг 4. Иначе вычислить разницу желаемым и полученным значение НС:

$$\delta = Y_l - Y \tag{11}$$

Изменить веса в соответствии с формулой:

$$w_{ij}(t+1) = w_{ij}(t)+\nu\delta x_i \tag{11}$$

где t и $t+1$ — номера текущей и следующей итерации; ν — коэффициент скорости обучения; i — номер входа; j — номер нейрона в слое.

Веса будут увеличины, если $Y_l > l$, тем самым ошибка уменьшится. В обратном случае они будут уменьшены, и Y соответственно тоже уменьшится, приближаясь к Y_l.

4) Повтор шага 2, пока не будет достигнута желаемая точность.

На втором шаге на вход НС подаются все входные вектора из обучающей выборки в случайном порядке. Число итераций зависит от сложности задачи и конфигурации нейронной сети. Определить точное количество итераций, необходимых для корректного обучения определить невозвожно [14].

2.2 Сверточные нейронные сети

Сверточные нейронные сети имеют схожие характеристики с обычными нейронными сетями. Они состоят из нейронов, которые имеют обучаемые веса. Каждый нейрон преобразует входные данные в выходной сигнал, который, возможно, будет изменен нелинейностью. Каждая такая сеть имеет функцию потерь на последнем полносвязным слоем.

Изменения заключаются в том, что архитектура сверточных сетей построена на явном предположении, что входной слой представляет собой изображения. Это предположение вносит особые свойства в архитектуру сети. Функция прямого прохода становится более

эффективной для реализации и значительно уменьшаются количество параметров сети.

Проблема регулярных нейронных сетей заключается в невозможности маштабирования. Например, в CIFAR-10 изображения имеют размер 32x32x3 (3 цветовых канала), поэтому каждый полносвязный нейрон в первом скрытом слое будет иметь 32 * 32 * 3 = 3072 веса. Данное количество весов является приемлемым для нейронной сети, но полносвязная структура не маштобируется. Например, если на вход подается изображение с размером 200x200x3, то каждый полносвязный нейрон будет иметь 200 * 200 * 3 = 120000 весов. Поэтому посносвязная структура сети использует огромные вычислительные ресурсы и ресурсы памяти. Большое количество параметров быстро приведет к переобучению.

В сверточных нейронных сетях учитывается тот факт, что на вход подается изображение, поэтому архитектура таких сетей оптимальней использует ресурсы памяти. В частности, в отличие от обычных сетей, сверточные используют нейроны, имеющие 3 измерения: ширина, высота, глубина (в CIFAR-10 ширина — 32, высота — 32, глубина — 3). Нейроны в слое подключены только к малой области предыдущего слоя. На изображении 12 показана структура сверточной нейронной сети. Красный слой на изображении представляет входной слой. Его ширина и высота будут размером изображения, а глубина равна 3-м.

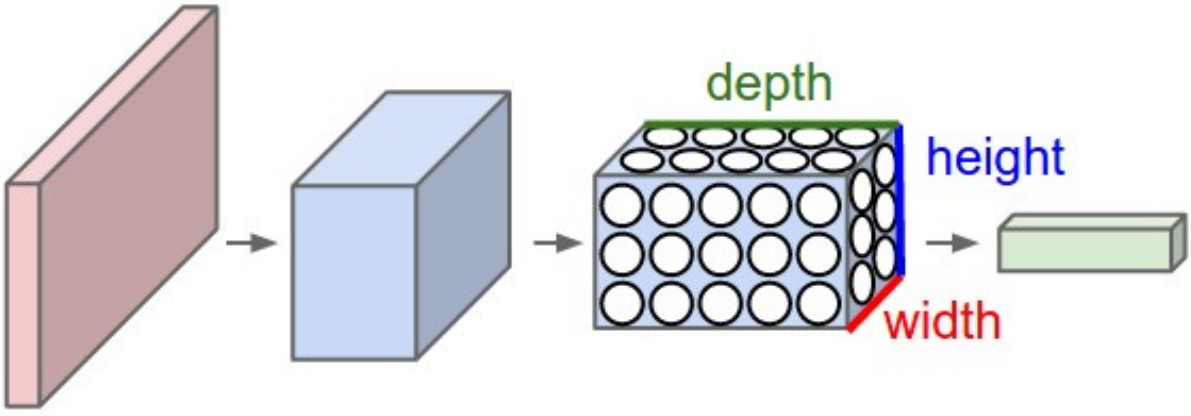

Рисунок 12 — Структура сверточной нейронной сети

Для построение архитектуры сверточной нейронной сети используются 3 основных типа своев: сверточный (convolution), слой объединения (pooling) и полносвязный [15].

Для решение задачи классификации на размеченной базе CIFAR-10 может использоваться следующая архитектура:

– Вход [32x32x3]. Содержит исходные значения пикселей изображения.

– Сверточный слой [32x32x12]. Вычисляет выходы нейронов, которые подключены к локальным областям на входе. Каждый нейрон имеет на выходе значение, вычесленное для небольшой области изображения. Это приводит к увеличению размерности слоя. В данном случае, слой будет иметь 12 фильтров, поэтому глубина слоя увеличится по отношению к входному слою.

– RELU-слой. Проводит активацию сверточного слоя. В таком слое применяется пороговая функция активации. В нашем случает используется функция $max(0, x)$. Размер слое остается неизменным.

– Объдиняющий слой [16x16x12]. Выполняет операцию понижения дискретизации по пространственным измерениям (ширина, высота). Такое преобразование приводит к умешьшению размерности пространственных плоскостей.

– Полносвязный слой [1x1x10]. Вычисляет оценки классов. Наждый из 10 значений соответствуют оценке класса, среди категорий CIFAR-10. Как и в случае с обычными нейронными сетями, каждый нейрон этого слоя связан со всеми нейроннами предыдущего слоя.

Таким образом сверточная нейронная сеть преобразует исходные значение пикселей изображения в итоговые оценки классов. В такой сети, некоторые слои содержат параметры, а другие нет. В частности, сверточный и полносвязный слой выполняют преобразования, которые являются функцией не только активации входного сигнала, но и параметров (веса, смещения нейронов). Объединяющий и RELU слои реализуют фиксированную функцию.

В итоге, можно сделать следующие выводы:

– Архитектура сверточной нейронной сети в простейшем случае представляет собой список слоев, которые преобразуют изображения в выходные сигланы (например, вероятности классов изображений).

– Существуют несколько разлизных типов слоев (Сверточный, объединяющий, RELU и полносвязный являются самыми популярными).

– Каждый слой принивает 3-х мерный массив сигналов и преобразует его в выходной 3-х мерный массив сигналов через дифференцируемую функцию.

– Каждый слой может иметь или не иметь параметров.

На рисунке 13 показан результат работы сверточной нейронной сети. На данном изображении визуализированны выделенные признаки нейронной сетью на каждом из этапов прямго прохода.

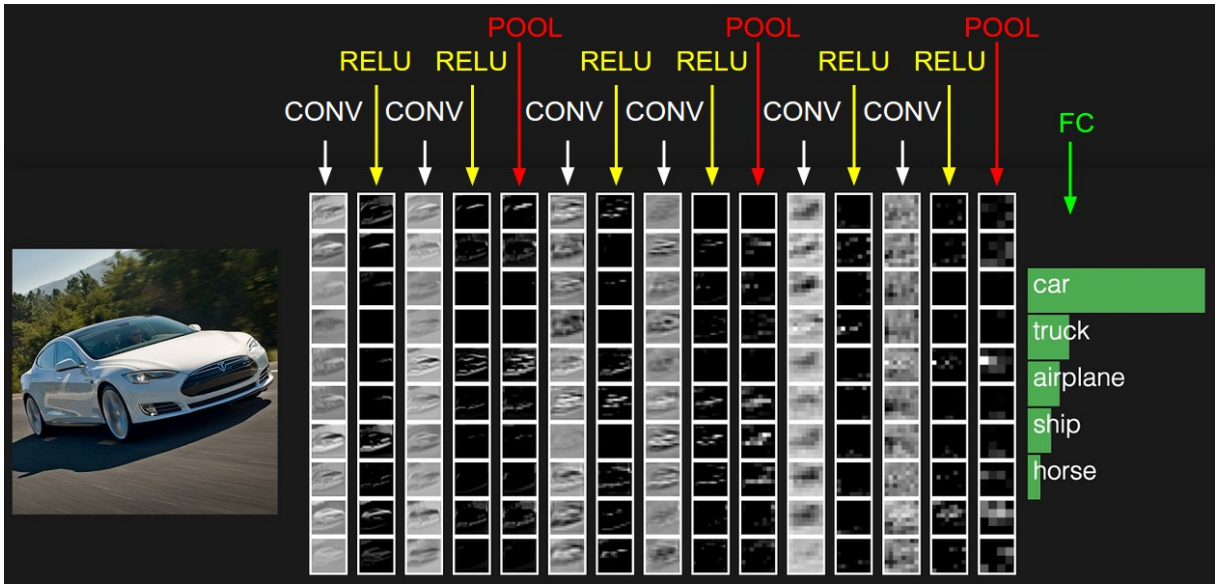

Рисунок 13 — Визуализация работы сверточной нейронной сети

Сверточный слой является основным стоительным блоком сверточной нейронной сети. На него затрачивается основная часть вычислительных ресурсов.

Для начала разберем сверточный слой без привязки к биологическому нейрону. Параметры слоя состоят из набора обучаемых фильтров. Фильтр является малым относительно входного пространства (по ширине и высоте), однако он проходит через всю глубину входного объема. Например, стандартный фильтр для первого сверточного слоя имеет размер 5x5x3 (Ширина и высота по 5 пикселей). Во время прямого прохода мы перемещаем фильтр по входному пространству и вычисляем

свертку между локальными значениями входного пространства и значениями фильтра. При этом вычисляется двумерная карта активации, которая генерирует выход в каждом пространственном положении. На протяжении этого процесса сеть, активирует функции, которые представляют собой какую-либо визуальную информации. От различный линий на первом слое, и конкретные объекты изображения на конечном слое. В итоге, нейронная сеть, представленная выше, будет иметь 12 фильтров, каждый из который сгенерирует двумерную карту активаций.

При работе с высокоразмерными входами, такими как изображение, как было показано выше, не целесообразно связывать нейроны текущего слоя со всеми нейронами предыдущего слоя. Вместого этого в сверточных нейронных сетях каждый нейрон подключается только к локальной области входного объема. Пространственная протяженность этой связности является гирерпараметром, которые называется восприимчивым полем нейрона (размер фильтра). Важной особенносью является то, что соединения нейронов локальны в пространстве (по ширине и высоте), но всегда полны по всей глубине входного объема.

Например, если входным слоем является изображение 32x32x3 и размер фильра равен 5x5, то каждый нейрон в сверточном слое будет имет размер 5x5x3. В общей сложности 5*5*3 = 75 весов (и параметр смещения).

На рисунке 14 показано пространственное подключение нейрона, но по полной глубине [16].

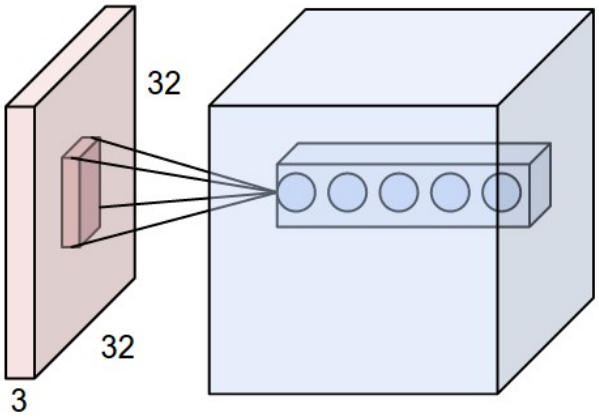

Рисунок 14 — Визуализация свертчного слоя

Следующие параметры управляют размером выхода сверточного слоя:

– Глубина — гиперпараметр, соответствующий числу фильтров. Каждый из фильтров обучается на поиск конкретного признака объекта.

– Шаг — величина на которорую сдвигается фильтр.

– Нулевое заполнение — размер размещений нулей вокруг границ входного пространства.

2.3 Обнаружение объектов с применем подхода YOLO

2.3.1 Преимущества YOLO

Люди смотрят на изображение и сразу узнают, какие объекты находятся на изображении, где они находятся, и как они взаимодействуют. Человеческая визуальная система является быстрой и точной, что позволяет нам выполнять сложные задачи, такие как вождение автомобиля. Быстрые и точные алгоритмы обнаружения объектов позволят компьютерам управлять автомобилями без специальных датчиков. Такие алгоритмы используются для передачи вспомогательным устройствам информации о событиях в реальном времени, что расширяет потенциял для универсальных, гибких роботизированных систем.

На данный момент, системы обнаружения перепрофилируют классификаторы. Чтобы обнаружить объект, эти системы используют классификатор для этого объекта и оценивают его работу в различных местах изображения и в различных маштабах. Такие системы используют подход скользящего окна, где классификатор запускается в равномернорасположеных точках по всему изображению.

Основная идея подхода YOLO стоиться на том, что обнаружение объекта определяется как проблема с одной регрессией, от пикселов изображения до координат ограничительной рамки и вероятностей классов. Такой подход позволяет за один проход сети определить, акие объекты присутствуют на изображении и где они находятся.

Архитектура YOLO представляет собой сверточную нейронную сеть, которая одновременно предсказывает локализацию объекта и класс найденного объекта. На рисунке 15 показана схема работы детекции. Такая модель имеет несколько преимуществ над существующими решениями детекции объектов [17].

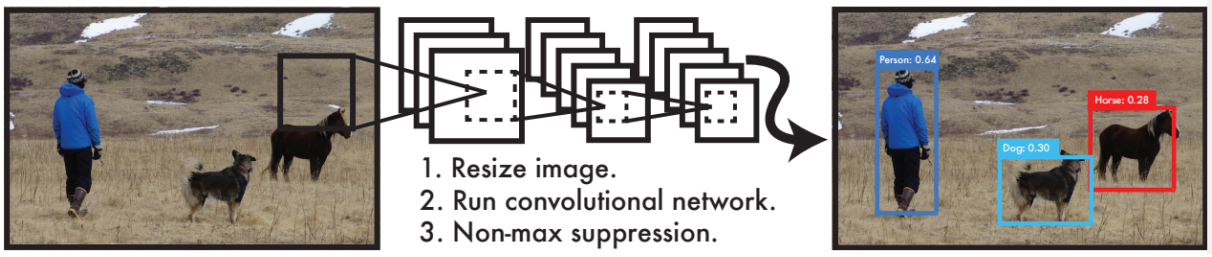

Рисунок 15 — Система обнаружения YOLO

Во-первых, нейронная сеть YOLO является очень быстрой. Посколько детекция расскатривается как проблема регрессии, не нужен сложный конвейвер. Кроме того, YOLO превосходит более чем в два раза среднюю точность обнаружения по сравнению с другими системами реального времени.

Во-вторых, YOLO учитывает весь контекст изображения. В отличии от методов, основанных на использовании окон и регионов, YOLO при обучении учитывает не только конкретный объект, но и контекстуальную информацию, которая существенно влияет на точность детекции.

В-третьих, YOLO обобщает предсавления объектов. При обучении на естественных изображениях и тестировании на художественных работах YOLO превосходит методы детекции, такие как DPM и R-CNN.

2.3.2 Алгоритм единого обнаружения

Сеть YOLO делит входное изображение на сетку $S x S$. Если центр объекта попадает в ячейку сетки, эта ячейка сетки отвечает за обнаружение этого объекта.

Каждая ячейка сетки предсказывает B ограничивающих рамок и оценки доверия для этих рамок. Эти оценки доверия отражают уверенность модели в том, что в рамке содержится объект. Формально, величина доверия определяется следующим образом:

$$Pr(Object)*IOU_{pred}^{truth}, \quad (1)$$

где Pr — функция определяющая величину вероятности объекта, IOU_{pred}^{truth} — метрика пересечения между предсказанной локализацией и дейтвительным местоположением объекта.

Если объекта нет в данной ячейке, то оценка доверия должна быть равна нулю. В обратном случае, величина доверия равна пересечению между предстказанным местоположением и действительным местоположением объекта.

Каждый ограничивающий блок состоит из 5 прогнозов: x, y, w, h и величина доверия. Координаты (x, y) представляют собой центр блока относительно границ ячейки сетки. Ширина и высота предсказываются относительно всего изображения. Величина доверия представляет собой IOU между предсказанным блоком и дейтвительным местоположением объекта.

Каждая ячейка сетки также пресказывает вероятности условного класса C. Эти вероятности обусловлены ячейкой сетки, содержащей объект [18].

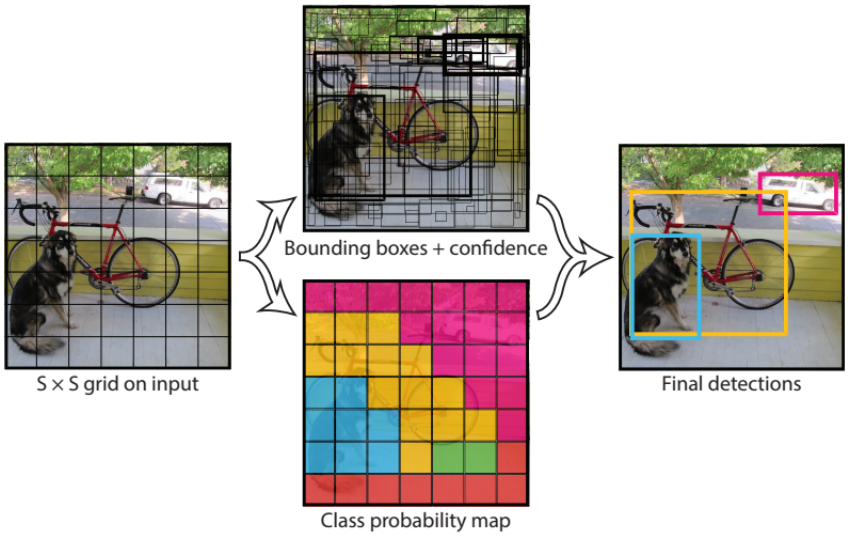

Рисунок 16 — Модель YOLO

Во время тестирования условные вероятности классов и идивидуальные оценки доверия каждого блока. Эта оценка интерпритируется как вероятность того, что данный класс имеется в блоке и насколько предсказанный блок подходит для данного объекта.

Суммировав вышесказанное, алгоритм работы YOLO выглядит следующим образом:

– Разделение изображения на сетку $S x S$;
– Предсказание ограничивающих блоков B;
– Предсказание классов C для объектов в каждом блоке.

На рисукне 16 изображена визуальная модель сети YOLO.

Выводы

Сверточные нейронные сети являются наиболее оптимальным вариантом для решение задачи детектирования изображений на маломощном устройстве. Это обусловлено спецификой архитектуры данного типа нейронных сетей. Сверточные и объединяющие слои позволяют троекратно сократить затраты вычислений и оперативной памяти при прямом проходе за счет уменьшения размерности признаков.

Сверточной сети YOLO позволяют реализовать алгоритм детектирование объектов с минимальными затратами вычислительных ресурсов. Важным приемуществом данного подхода является целостность архитектуры. При использовании YOLO достаточно один раз совершить прямой проход сети по изображению, что бы детектировать объекты на нем, что значительно уменьшает количество вычислений по сравнению с методами сколзящего окна [19].

3 Проектирование системы

3.1 Архитектура системы

Для полноценного функционирования система разделена на две части: клиентскую и серверную.

Клиентское приложение запускается на персональном компьютере и имеет GUI-интерфейс для общения с пользователем. В свою очередь серверное приложение, запускаемое на ARM-устройстве выполняет детекцию изображения и возвращает результат клиентскому приложению в виде размеченного изображения. Взаимодействие между клиенстким и серверным приложением осуществляется посредством TCP-сокетов.

3.1.1 Проектирование клиентского приложения

Клиенское приложение необходимо для демонстрации вычислений сверточной нейронной сети, которая производится на ARM-устройстве. Оно представляет собой GUI-интерфейс, в котором пользователю предоставлются следующие возможности:

- Подключение к серверному приложению на ARM-устройстве;
- Выбор изображения для детекции;
- Запуск детекции на сервере;
- Просмотр информации о этапах работы сервера;
- Завершение работы серверного приложения.

На рисунке 17 показана диаграмма вариантов использования клиенского приложения.

Что бы получать информацию о выполненных этапах детекции изображения на сервере, запущен дополнительный поток, который ожидает сообщения от сервера и выводит сообщение в текстовый браузер.

В приложении 2 технического задания изображена диаграмма последовательности, на которой показана какая информация поступает от серверного приложения и в какой последовательности.

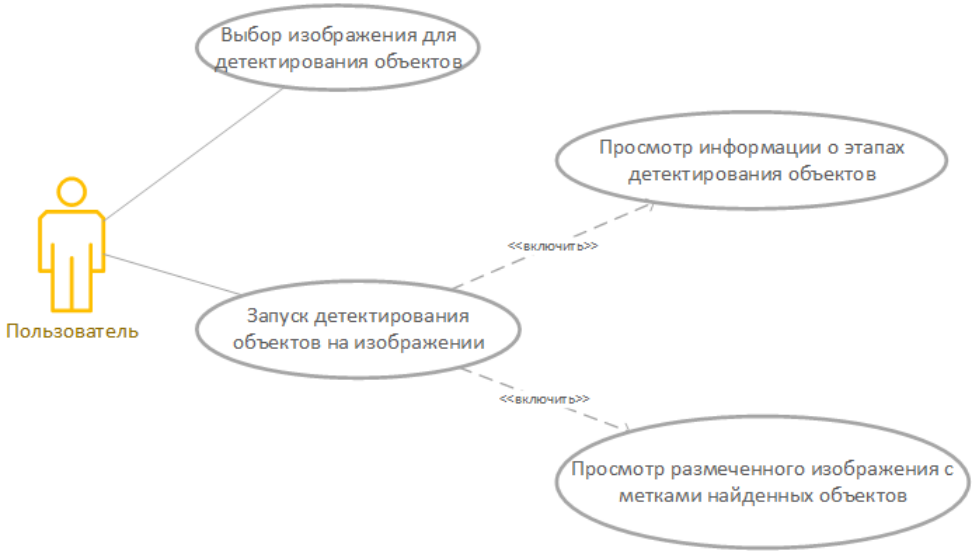

Рисунок 17 — Диаграмма вариантов использования клиенского приложения

3.1.2 Проектирование серверного приложения

Серверное приложение выполняется на ARM-устройстве. Оно реализует взаимодействие с клиентской частью, используя сокеты, и детекцию объектов на изображении с использованием API фреймворка darknet.

После подключения клиента, серверное приложения ожидает команды от клиентского приложения. Для запуска детекции изображения используетса команда "yolo". На рисунке 18 изображена диаграмма деятельности, на которой показны функции, выполняемые серверным приложением в зависимости от поступившей команды клиетского приложения.

Завершение работы серверного приложение происходит при получении команды "exit". При этом серверное приложение закрывает дискриптор сокета, через который совершался обмен сообщениями.

Для обмена сообщениями между клиентским и серверным приложением реализован общий интерфейс передачи данных. В него

входят функции приема и отправки текстовых сообщений и изображений.

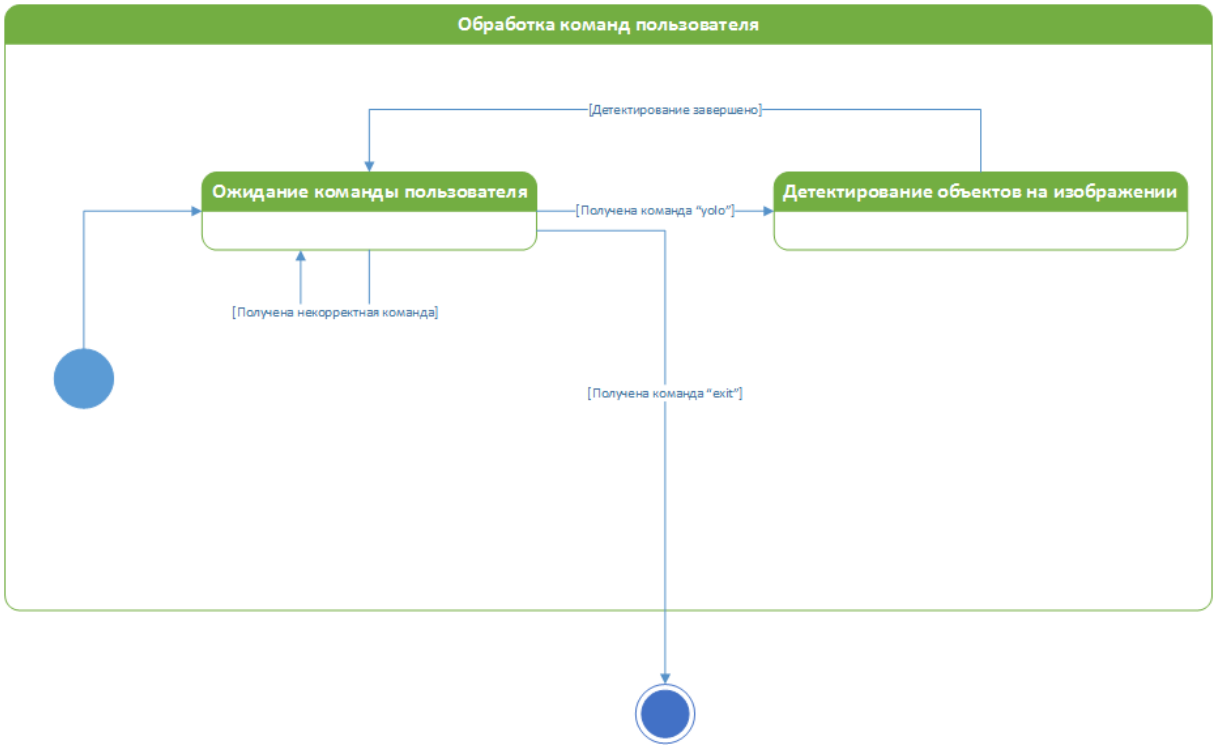

Рисунок 18 — Диаграмма деятельности серверного приложения

3.1.3 Проектирование алгоритма прямого прохода сверточной нейронной сети

Для реализации детектирование объектов на изображении используется API фреймворка darknet. В время работы детекции изображения серверное приложение отсылает клиенту информацию о пройденых этапах детекции. На рисунке 19 изображена диаграмма последовательности, на которой показаны этапы детекции.

Для реализации прямого прохода нейронной сети использовались следующие функции, реализованные в фреймворке darknet:

– read_data_cfg — осуществляет парсинг меток классов;

– parse_network_cfg — осуществляет парсинг конфигурации нейронной сети;

– load_weights — сериализует веса нейронной сети;

– load_image_color — сериализует изображение для детекции;

– network_predict — осуществляет прямой проход сериализованной нейронной сети;

– draw_detections — осуществляет отрисовку ограничивающих объекты блоков и меток объектов.

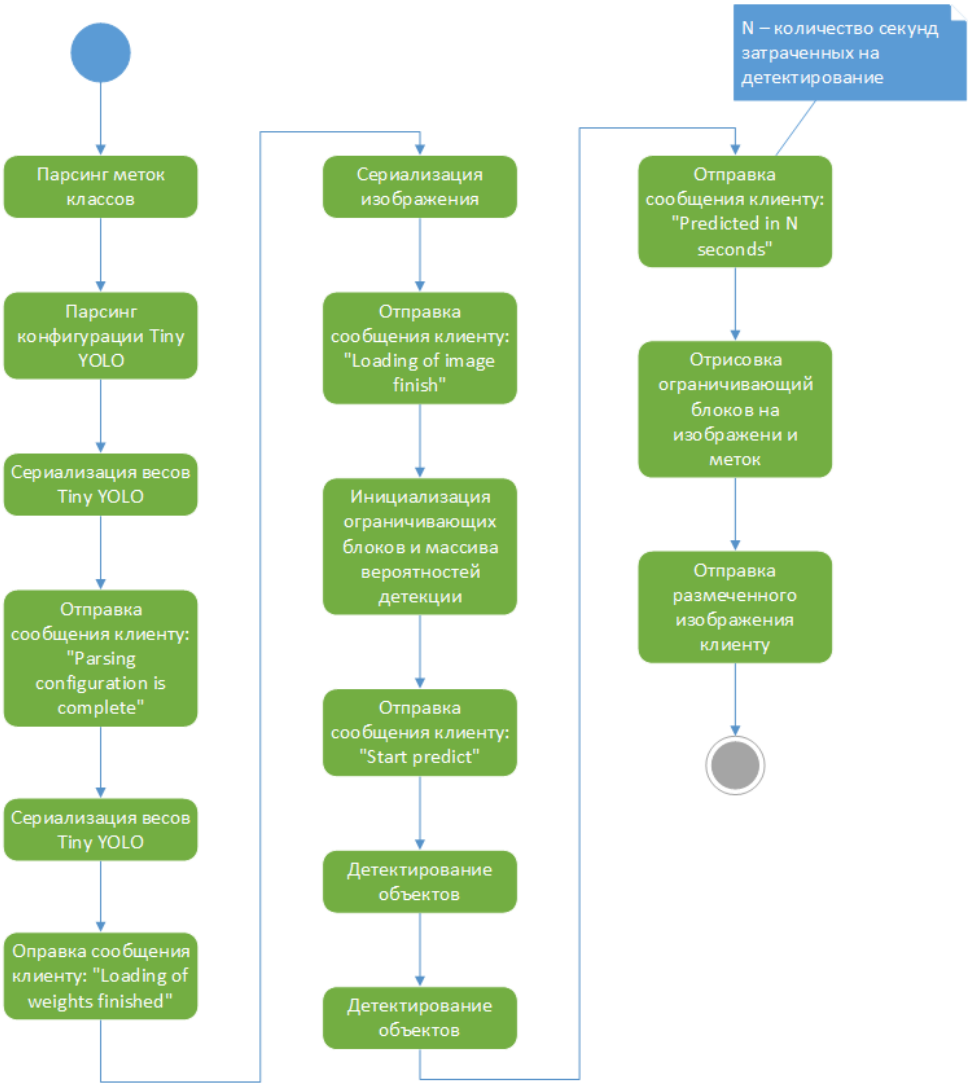

Рисунок 19 — Диаграмма последовательности алгоритма детектирования объектов

3.2 Особенности реализации прямого прохода сверточной нейронной сети

Для уменьшения затрат оперативной памяти при прямом проходе сверточной нейронной сети были произведены следущие мероприятия:

– Для реализации прямого прохода использовалась сверточная нейронная сеть Tiny YOLO;

– После прохождения очередного слоя, память затраченная на его сереализацию очищается.

Архитектура сверточной нейронной сети Tiny YOLO в два раза меньше оригинальной сети YOLO. Соответсвенно занимаемая память на хранение сети уменьшилась в два раза. В приложении Б изображена архитектура Tiny YOLO. В таблице 2 показано количество занимаемой памяти сетью Tiny YOLO.

Таблица 2 — Количество памяти, необходимое для сериализации весов Tiny YOLO

Имя слоя	Размер слоя	Необходимое количество памяти, Мб
Data	448x448x3	2,408 448
Conv1	224x224x64	12,845 056
Pool1	112x112x64	3,211 264
Conv2	112x112x192	9,633 792
Pool2	56x56x192	2,408 448
Conv3	56x56x128	1,605 632
Conv4	56x56x256	3,211 264
Conv5	56x56x256	3,211 264
Conv6	56x56x512	6,442 528
Conv7	28x28x256	0,802 816
Conv8	28x28x512	6,442 528
Conv9	28x28x256	0,802 816
Conv10	28x28x256	0,802 816
Conv11	28x28x256	0,802 816
Conv12	28x28x512	6,442 528
Conv13	28x28x256	0,802 816
Conv14	28x28x512	6,442 528
Conv15	28x28x512	6,442 528
Conv16	28x28x1024	3,211 264
Fc17	1x1x4096	0,016 384
Fc19	1x1x1470	0,005 880

Суммарное количество памяти, необходимо для сериализации Tiny YOLO составляет 79,601 Мб. Это является приемлемыми затратами для мобильного пк C.H.I.P., который использовался как ARM-устройство в данной работе. Однако, при прямом проходе каждого сверточного и объединяющего слоя генерируюся выходные сигналы. Поэтому, ресурсов C.H.I.P. не достаточно для осуществления прямого прохода.

Для того, что бы оптимизировать работу с оперативной памятью при выгрузке слоев, был видоизменена функция прямого прохода нейрнных сетей в фрейворке darnket. Сам фремворк интегрируется посредством компилирования исходных файлов фреймворка в исполняемый файл серверного приложения. На рисунке 20 показаны различия между оригинальным алгоритмом прямого прохода и видоизмененным.

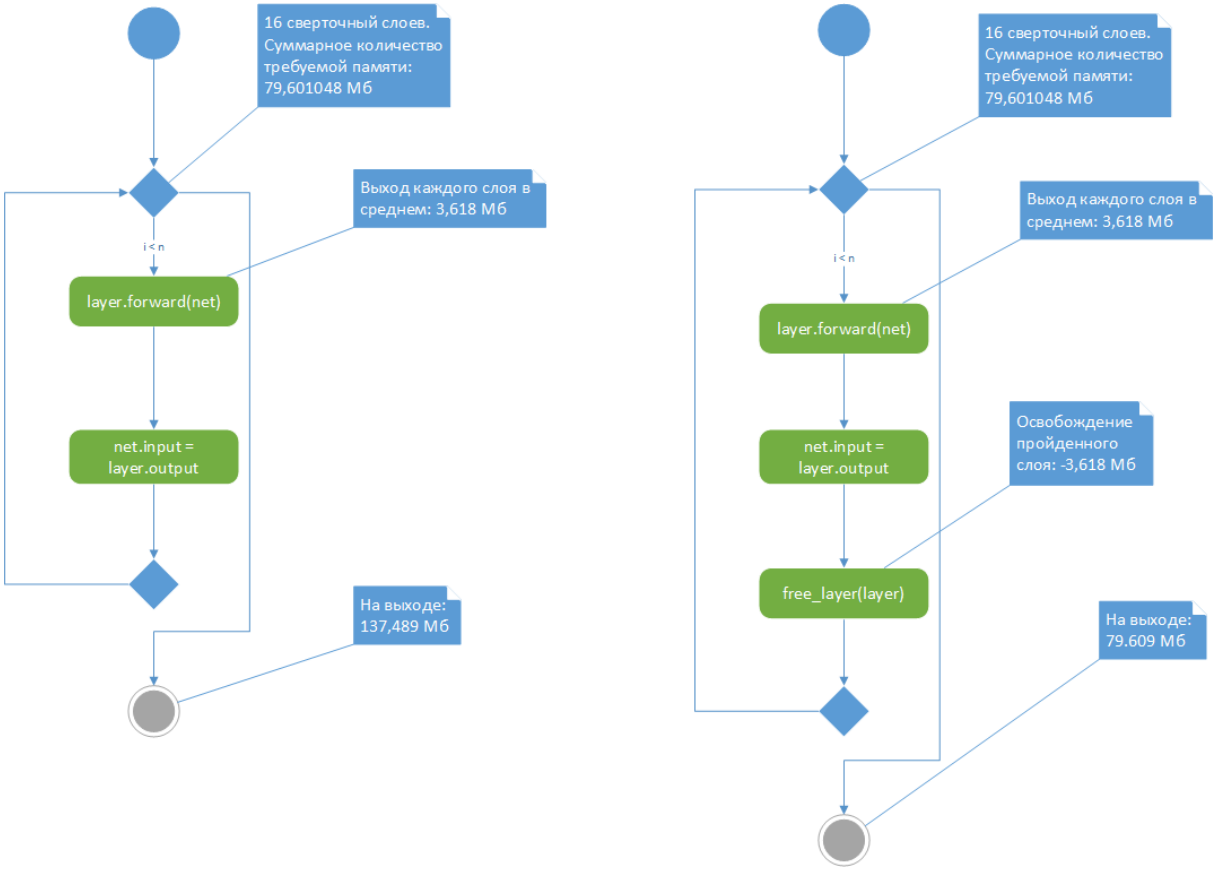

Рисунок 20 — Различие между оригинальным алгоритмом прямого прохода и видоизмененным

Удаление слоя после его прохода, позволяет уменьшать затраты оперативной памяти на каждой итераци вычислений и позволяет

осуществлять детектирование объектов на маломощном устройстве C.H.I.P.

3.3 Требования к входным/выходным данным приложения

Входными данными для клиентского приложения является изображения в форматах png и jpg.

Входными данными для серверного приложения являются файл с метками объектов, на детекцию которых обучена нейронная сеть формата data, конфигурационный файл формата cfg, изображение для детектирования в форматах png и jpg, веса нейронной сети в формате weights. В приложении 1 технического задания описаны форматы конфигурационных файлов нейронной сети.

Выходными данными системы является изображение с отображением меток детектированного изображения, данные о этапах детектирования и время выполнения детектирования.

На рисунке 21 показан результат работы детектирования изображения. Слева на изображении отмечены прямоугольником обнаруженные объекты. Справа отображена информация об основных этапах детектирования данного изображения.

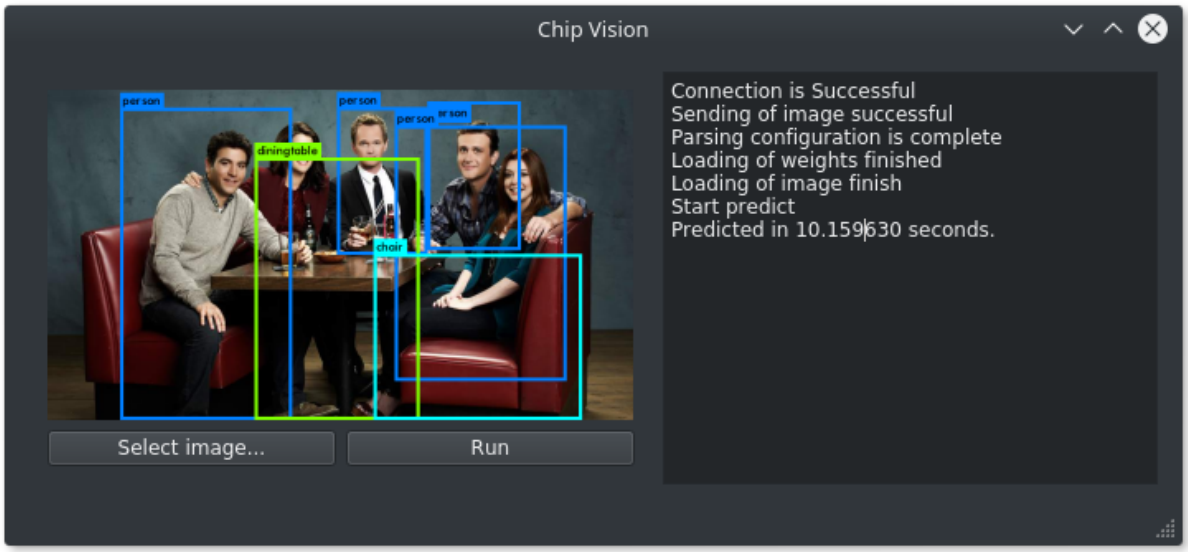

Рисунок 21 — Пример работы детекции объектов

3.4 Тестирование разработанного приложения

На рисунке 22 показно время на детектирование изображений различных разрешений.

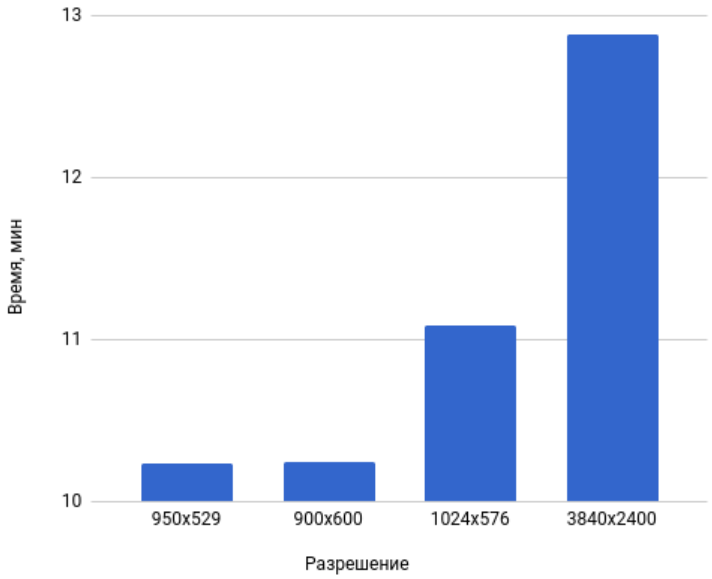

Рисунок 22 — Диаграмма зависимости времени детекции от разрешения изображения

В среднем время на детектирование объектов состовляет 10,5 минут. С увеличением размерности изображение время увеличивается, это связано с затратами на изменение размера изображения.

Заключение

На основании проделанной рабоыт можно сделать следующие выводы:

– Были проанализированны фреймворки глубинного машинного обучения;

– Были проанализированны подходы к детекции объектов на изображении;

– Разработан алгоритм детектирования объектов на изображении для маломощного ARM-устройства;

– Программно реализованно серверное приложение, выполняющее детектирование объектов на маломощном ARM-устройстве;

– Программно реализованно клиентское приложение, которое демонстратирует результаты работы серверного приложения.

Разработанная система позволяет сделать вывод о возможности реализации систем машинного обучения на маломощных устройствах. Однако, для реализации требовательных к вычислительным ресурсам и ресурсам памяти алгоритмов машинного необходимо выполнять следущие ограничения:

– использование алгоритмов и архитектур с наименьшим количеством геперпараметров;

– оптимиированная работа с памятью;

– отсутствие зависимостей.

Список использованных источников

1 A Quick Introduction to Neural Networks [Electronic resource]. — Mode of access : https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/ (date of access 12.03.2017).

2 Advantages and disadvantages of using artificial neural networks [Electronic resource]. — Mode of access : http://www.sciencedirect.com/science/article/pii/S0895435696000029 (date of access 12.03.2017).

3 The Advantages of an Embedded System [Electronic resource]. — Mode of access : https://www.techwalla.com/articles/the-advantages-of-an-embedded-system (date of access 12.03.2017).

4 Caffe: Convolutional Architecture for Fast Feature Embedding [Electronic resource]. — Mode of access : https://arxiv.org/pdf/1408.5093.pdf (date of access 21.05.2017).

5 ImageNet classification with deep convolutional neural networks [Electronic resource]. – Mode of access : https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-d (date of access 25.05.2017).

6 Decaf: A deep convolutional activation feature for generic visual recognition [Electronic resource]. – Mode of access : http://proceedings.mlr.press/v32/donahue14.pdf (date of access 25.05.2017).

7 Torch7: A Matlab-like Environment for Machine Learning [Electronic resource]. – Mode of access : https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-d (date of access 27.05.2017).

8 Справочное руководство по языку Lua 5.1 [Электронный ресурс]. – Режим доступа : http://www.lua.ru/doc/1.html (дата обращения 23.05.2017).

9 Darknet: Open Source Neural Networks in C [Electronic resource]. — Mode of access : https://pjreddie.com/darknet/ (date of access 22.03.2017).

10 XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks [Electronic resource]. – Mode of access : https://pjreddie.com/media/files/papers/xnor.pdf (date of access 30.05.2017).

11 Монахова, Е. Д. "Нейрохирурги"с Ордынки / Е. Д. Монахова // PC Week/RE. – 2013. – № 9. — С 25-28.

12 Уоссермен, Ф. Нейрокомпьютерная техника: Теория и практика / Ф. Уоссермен ; перевод Зуев Ю. А., Точенов В. А. – Москва : Мир, 1992. – 184 с.

13 Итоги науки и техники. Серия Физические и математические модели баз данных и нейронных сетей / Российская акад. наук, Всероссийский ин-т науч. и технической информ. – Москва : ВИНИТИ, 2007. – 351 с.

14 Нейронная сеть – введение [Электронный ресурс]. – Режим доступа : http://robocraft.ru/blog/algorithm/558.html (дата обращения 28.05.2017).

15 Neural Network Architectures — Mode of access : https://medium.com/towards-data-science/neural-network-architectures-156e5 (date of access 11.04.2017)

16 Deep Learning with Limited Numerical Precision / Suyog Gupta [etc.] // ICML'15 Proceedings of the 32nd International Conference on International Conference on Machine Learning : Lille, France, July 06-11, 2015. / International Conference on Machine Learning. — Lille, France, 2015. — Volume 37. — Pages 1737-1746.

17 YOLO: Real-Time Object Detection [Electronic resource]. – Mode of access : https://pjreddie.com/darknet/yolo/ (date of access 16.04.2017).

18 You Only Look Once: Unified, Real-Time Object Detection [Electronic resource]. – Mode of access : https://pjreddie.com/media/files/papers/yolo_1.pdf (date of access 11.05.2017).

19 YOLO9000: Better, Faster, Stronger [Electronic resource]. – Mode of access : https://pjreddie.com/media/files/papers/YOLO9000.pdf (date of access 17.05.2017).

Приложение А
Техническое задание

Приложение Б — Архитектура сверточной нейронной сети Tiny YOLO

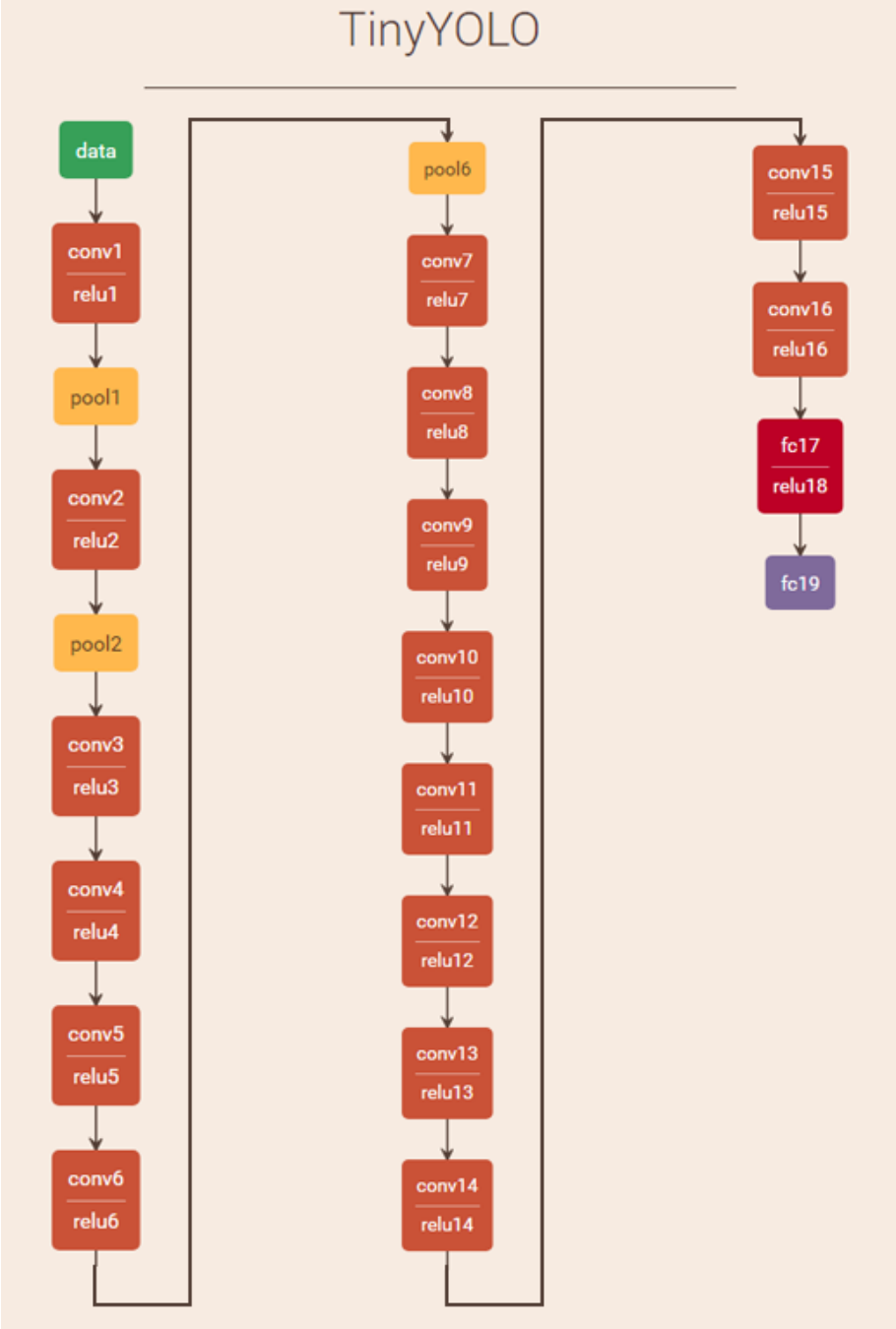

Рисунок Б.1 — Конфигурация сверточной нейронной сети Tiny YOLO