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 ВКР-40 461 806-10.27-02-17.81
(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 г.

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 «26» декабря 2016 № 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 машинного обучения

Выводы

2 Используемые алгоритмы и модели

Выводы

3 Проектирование системы

Выводы

Заключение

Список использованных источников

Приложение А Техническое задание

Приложение Б Архитектура сверточной нейронной сети Tiny YOLO

Перечень графического материала

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

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 35 страниц и включает 14 рисунков и 1 таблицы.

Annotation

This document provides an explanatory note to the final work on the bachelor: «Porting a convolutional neural network on the ARM architecture with the limited computing resources and limited memory resources», developed by Melnikov Timofey Alekseyevich, IVT-461.

The possibility of implementing machine learning algorithms on a device with limited computing resources and memory resources is considered in this final. In particular, forward function of a convolutional neural network.

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 технологий и фреймворков машинного обучения

1.1 Исскуственные нейронные сети

1.1.1 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 1 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 [4].

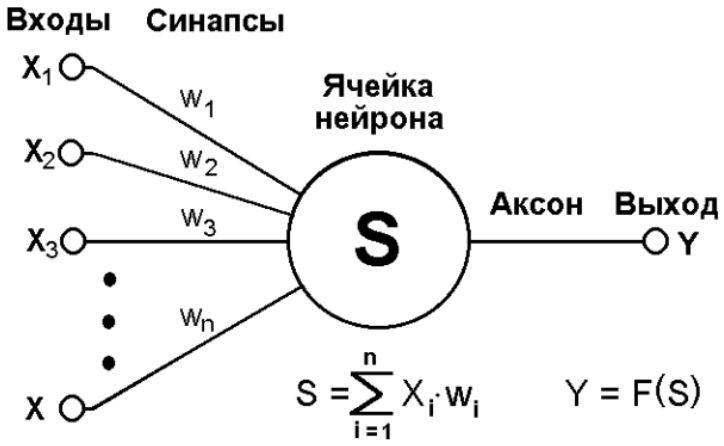

Рисунок 1 — Исскуственный нейрон

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 [5]. 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. Синтез нейронной сети напряму зависит от типа решаемой задачи, поэтому список подробных рекомендаций составить затруднительно. В большинстве случаев оптимальный вариант выбирается эмпирическим методом [6].

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 [7].

1.1.2 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 [8].

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 2 показана структура сверточной нейронной сети. Красный слой на изображении представляет входной слой. Его ширина и высота будут размером изображения, а глубина равна 3-м [9].

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

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

– входные данные [32x32x3];

– сверточный слой [32x32x12];
– RELU-слой;
– объдиняющий слой [16x16x12];
– полносвязный слой [1x1x10].

Входной слой содержит исходные значения пикселей изображения. Сверточный слой 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 выполняет операцию понижения дискретизации по пространственным измерениям (ширина, высота). Такое преобразование приводит к умешьшению размерности пространственных плоскостей. Полносвязный слой 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

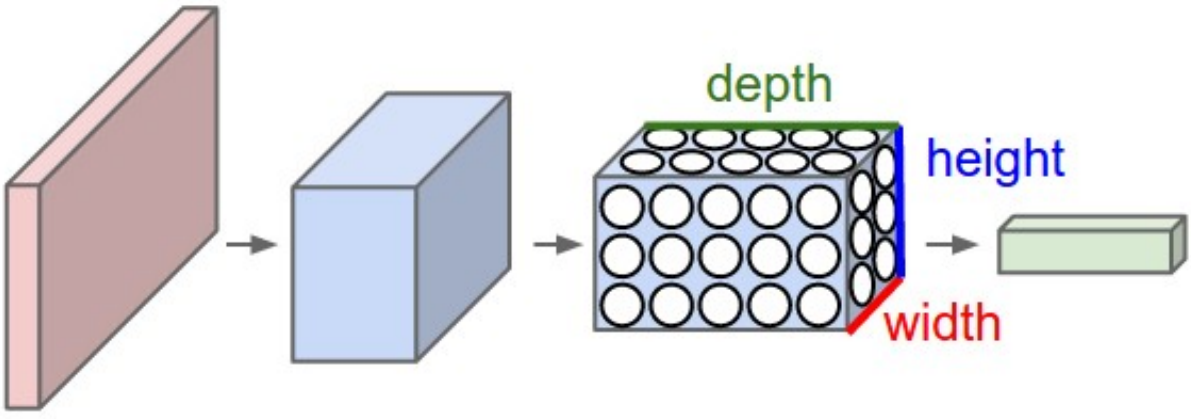

Рисунок 2 — Структура сверточной нейронной сети

являются функцией не только активации входного сигнала, но и параметров (веса, смещения нейронов). Объединяющий и RELU слои реализуют фиксированную функцию [10].

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

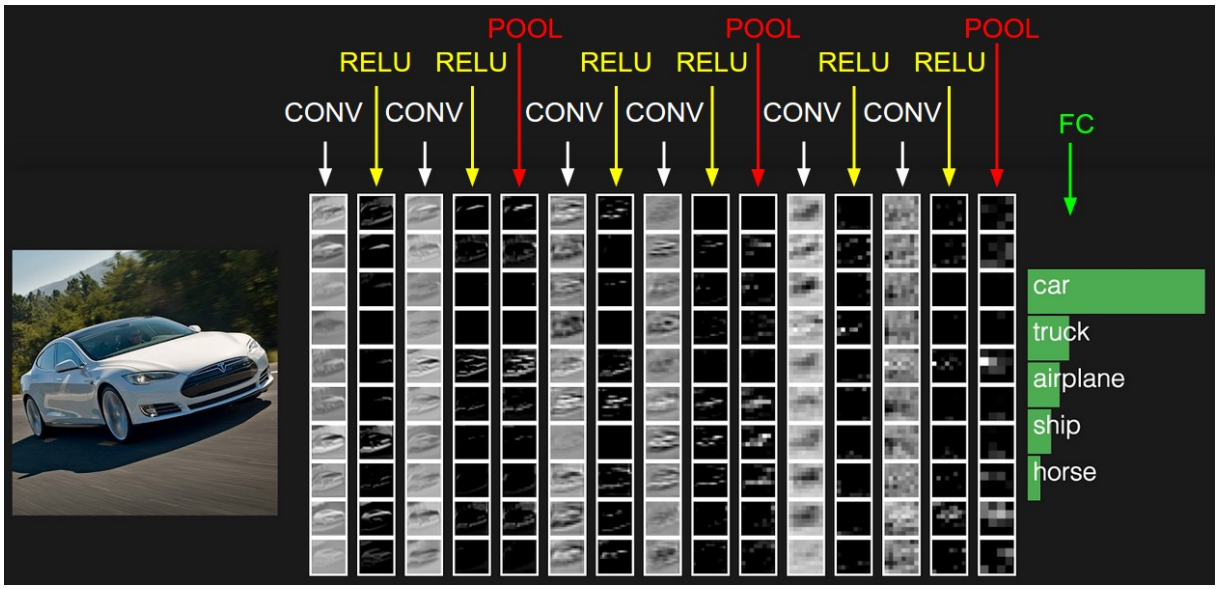

Рисунок 3 — Визуализация работы сверточной нейронной сети

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 4 показано пространственное подключение нейрона, но по полной глубине.

Следующие параметры управляют размером выхода сверточного слоя:

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

– шаг — величина на котороруюю сдвигается фильтр;

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

1.2 Обзор фреймворков машинного обучения

1.2.1 Фреймворк 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) [11].

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

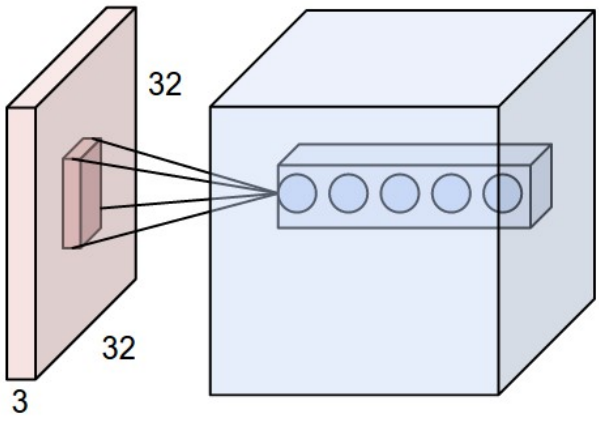

Рисунок 4 — Визуализация свертчного слоя

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 [12].

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;

– подробная документация для сериализации и использования моделей.

1.2.2 Фреймворк 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.

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 [13].

Структура фрейворка имеет три основных преимущества:

– она облегчает разработку численных методов;

– фреймворк легко расширяем (включая использование сторонних библиотек);

– высокая скорость работы фреймворка.

1.2.3 Фреймворк 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 [14].

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 [15]. Поэтому фреймворк можно встраивать как нативную библиотеку в любой пользовательский проект.

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.

1.3 Выводы

Светрочные нейронные сети являются оптимальным решением для решения задачи детекции изображений на маломощном устройсте. Архитектура таких сетей позволяет сокращать изрежки на хранимые в памяти параметры.

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.

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.

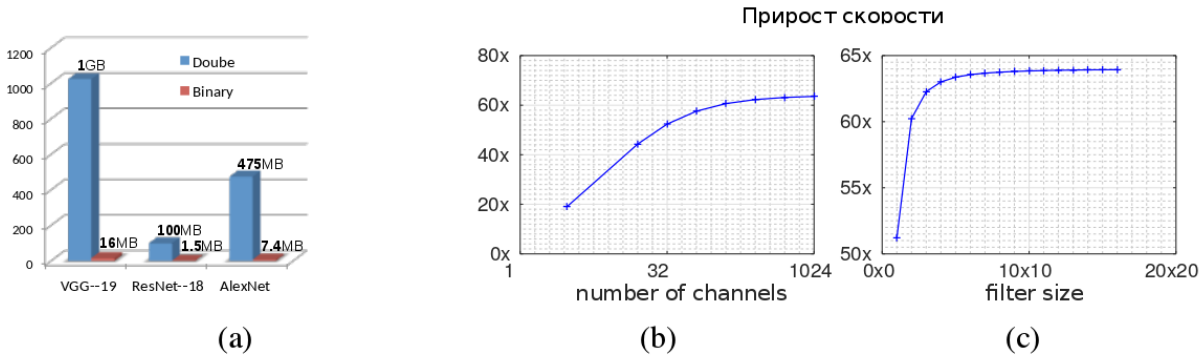

Рисунок 5 — Эффективность использования памяти и вычислений. a – выделяемая память для весов различных архитектур, b — ускорение в зависимости от числа каналов, c — ускорение в зависимости от размера фильтра

2 Используемые алгоритмы и модели

2.1 Обнаружение объектов с применем подхода YOLO

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 6 показана схема работы детекции. Такая модель имеет несколько преимуществ над существующими решениями детекции объектов [16].

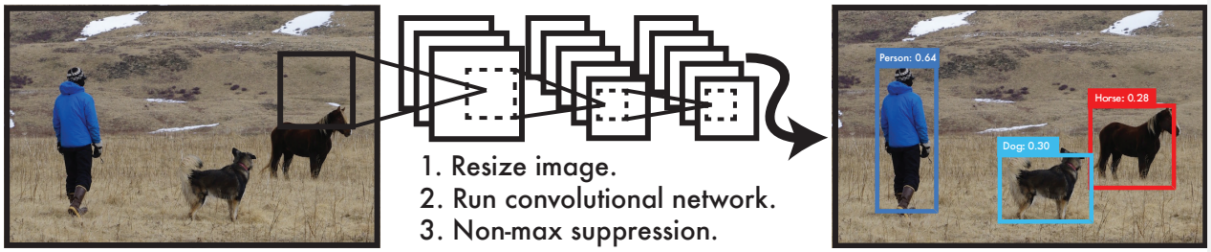

Рисунок 6 — Система обнаружения YOLO

2.2 Архитектура сверточной сети YOLO

Первые слои YOLO являеются сверточным и извлекают признаки объектов. Далее полносвязные слои прогнозируют вероятности и координаты объектов. Сеть имеет 24 сверточных слоя, за которыми следует 2 полносвязных слоя. Для уменьшения количества признаков используются сверточные слои с единичным фильтром. Полная сеть изображена на рисунке 7.

Входным слоем является трехканальное изображение с разрешением 448x448. Результатом сети является трехмерная матрица [7x7x30] содержащая информацию о ограничивающих блоках и вероятностей объектов.

2.3 Алгоритм единого обнаружения

Сеть YOLO делит входное изображение на сетку SxS. 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:

$$Pr(Object)*IOU_{pred}^{truth}, \quad (14)$$

где Pr — функция определяющая величину вероятности объекта, IOU_{pred}^{truth} — метрика пересечения между предсказанной локализацией и дейтвительным местоположением объекта [17].

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]. На рисукне 8 изображена визуальная модель сети 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 SxS;

– предсказание ограничивающих блоков B;
– предсказание классов C для объектов в каждом блоке.

2.4 Выводы

Сверточные 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.

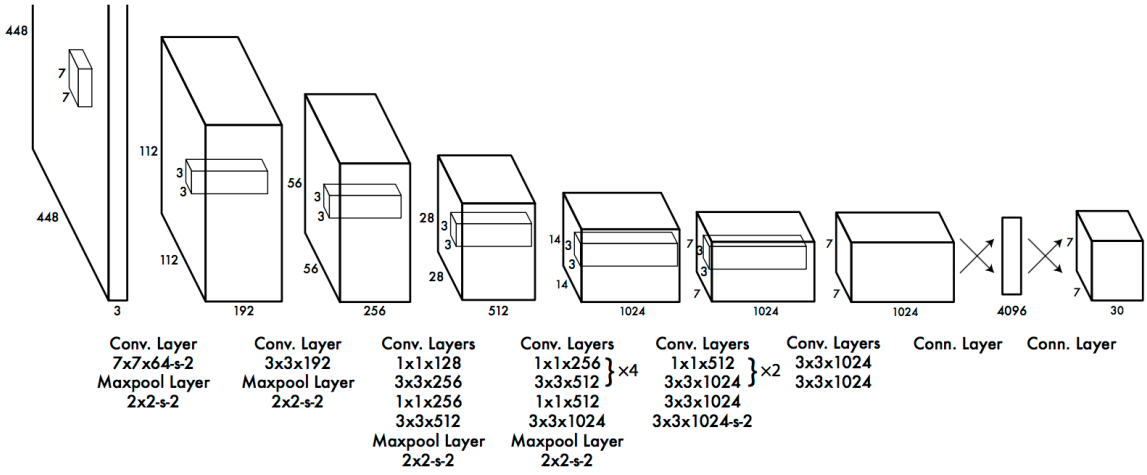

Рисунок 7 — Архитектура нейросети YOLO

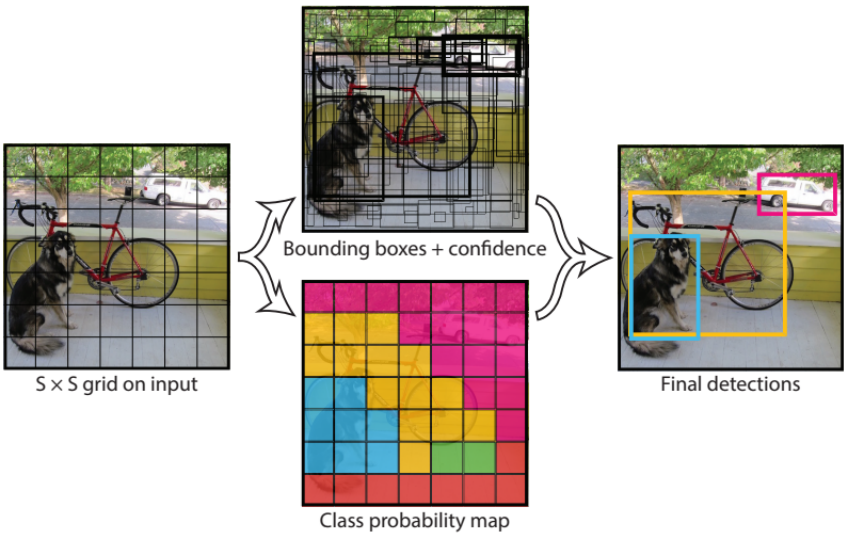

Рисунок 8 — Модель YOLO

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 9 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.

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 10 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.

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 11 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;

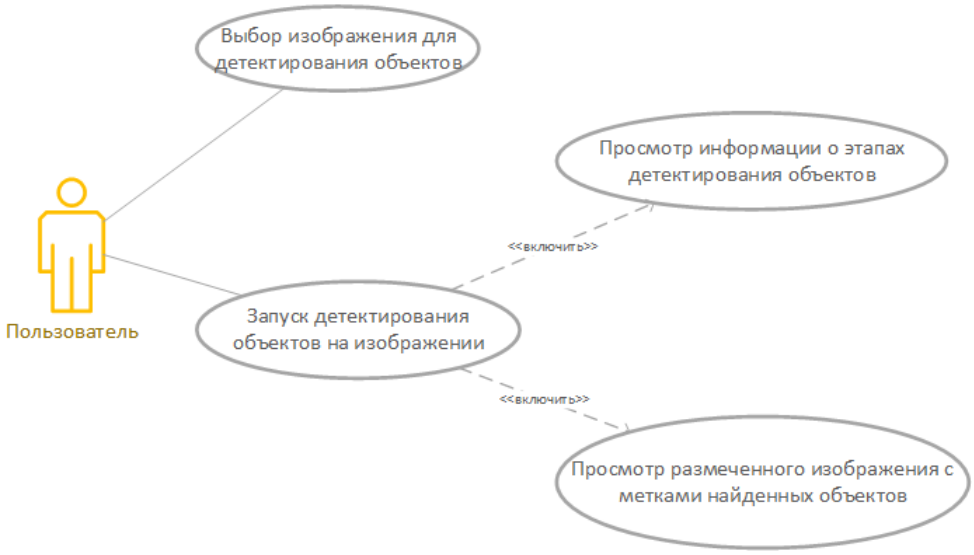

Рисунок 9 — Диаграмма вариантов использования клиенского приложения

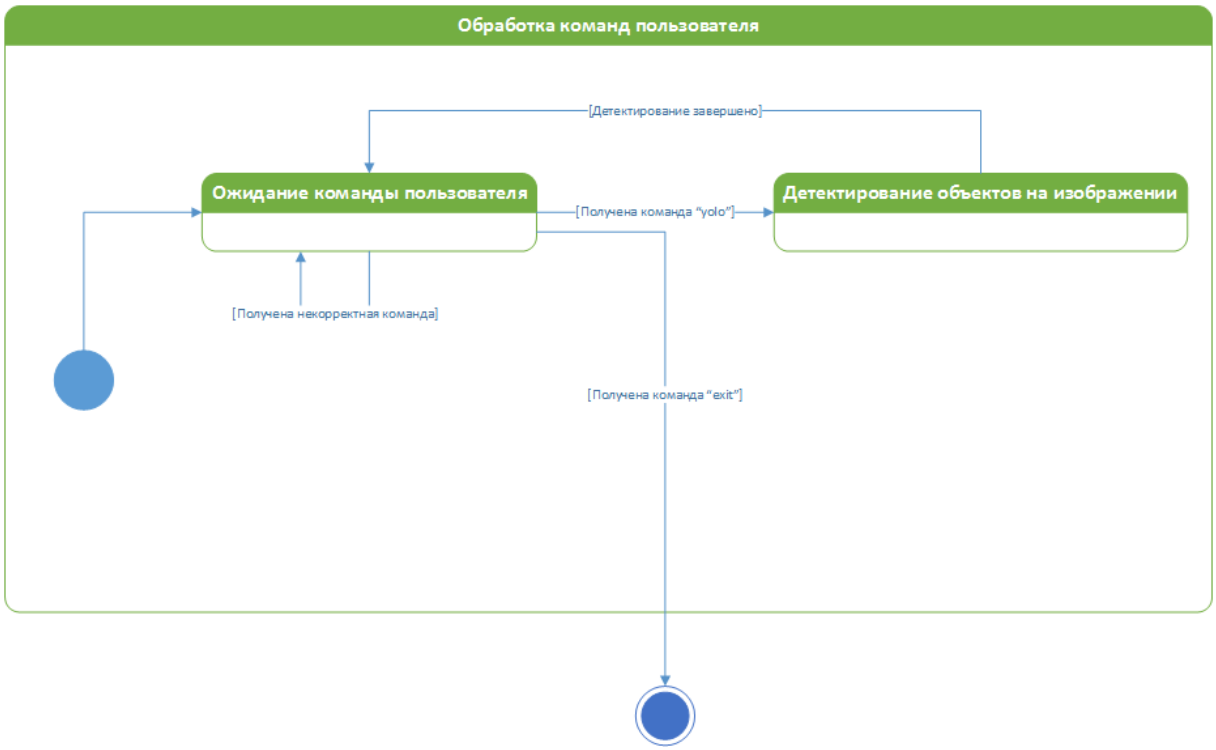

Рисунок 10 — Диаграмма деятельности серверного приложения

– 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.

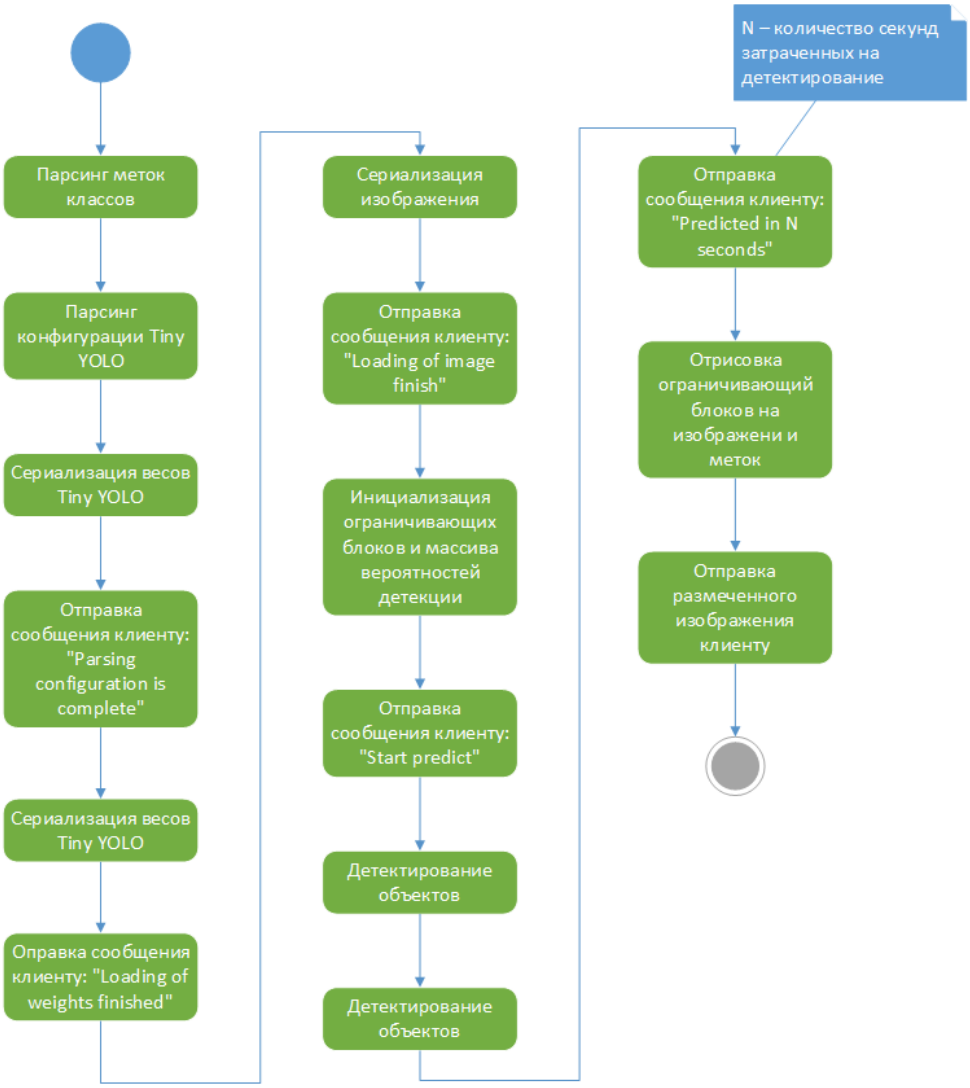

Рисунок 11 — Диаграмма последовательности алгоритма детектирования объектов

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 1 показано количество занимаемой памяти сетью Tiny YOLO.

Таблица 1 — 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 использользовался 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 12 показаны различия между оригинальным алгоритмом прямого прохода и видоизмененным.

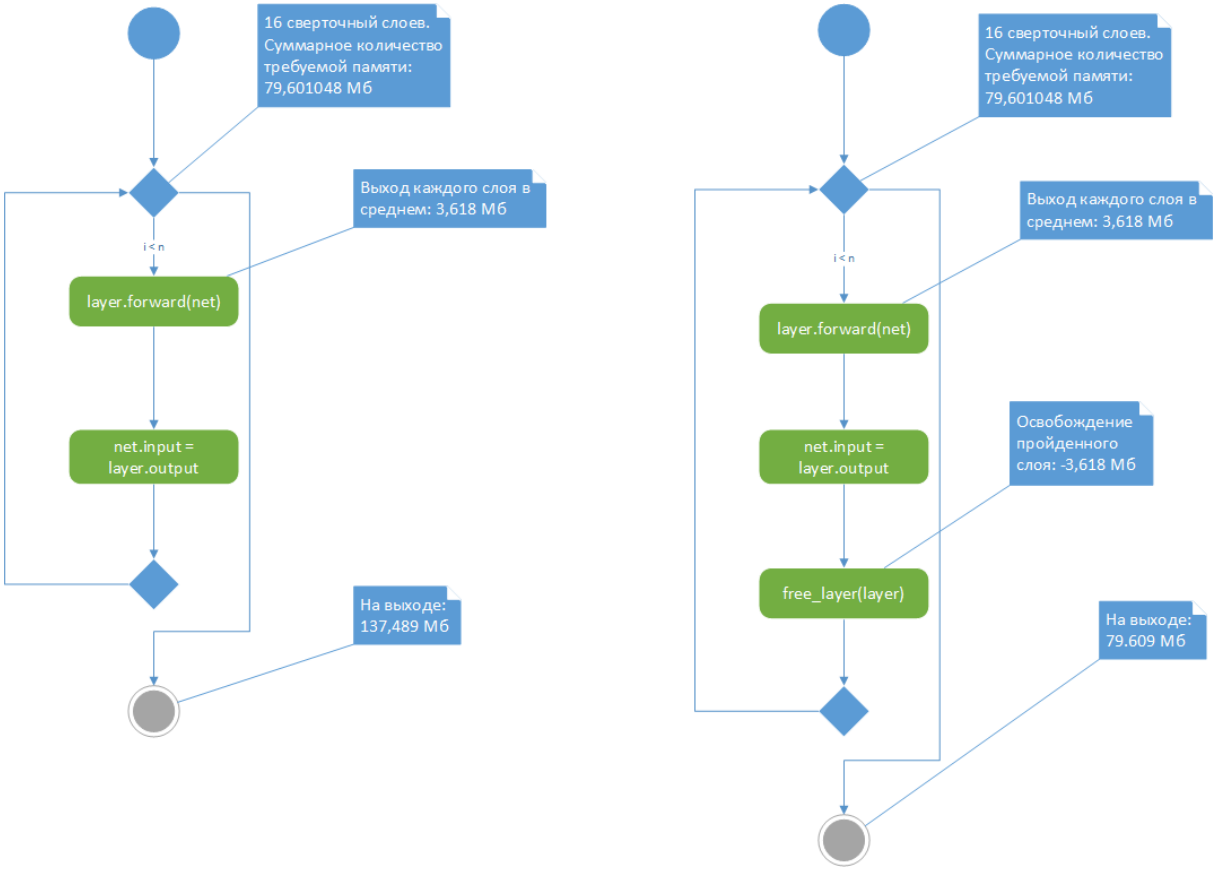

Рисунок 12 — Различие между оригинальным алгоритмом прямого прохода и видоизмененным

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

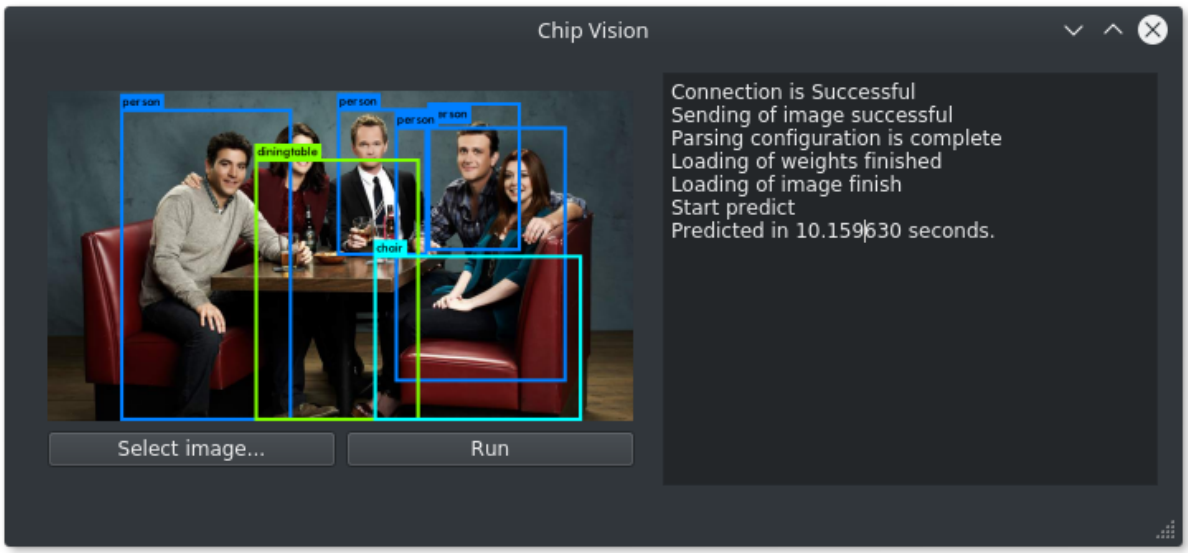

Рисунок 13 — Пример работы детекции объектов

3.4 Выводы

На рисунке 14 показно время на детектирование изображений различных разрешений.

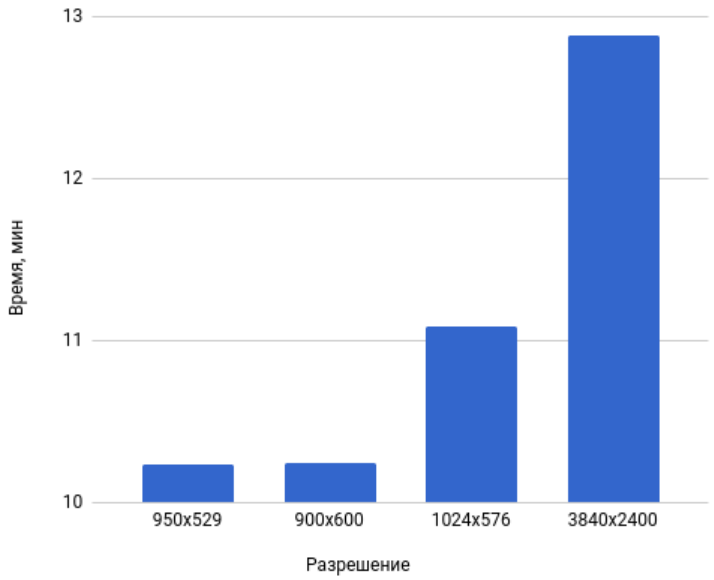

Рисунок 14 — Диаграмма зависимости времени детекции от разрешения изображения

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

Результаты тестирования приложения говорят о возможности реализации алгоритмов машинного обучения на маломощных устройствах.

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

5 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.

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

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

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

9 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.

10 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).

11 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).

12 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).

13 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).

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

15 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). 28.05.2017).

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

17 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).

18 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