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

Согласовано

_____
(должность гл. специалиста предприятия)

_____ _____
(подпись) (инициалы, фамилия)
«_____» _____ 2017

Утверждаю
Зав. кафедрой САПР и ПК, д.т.н.,
??.
_____ 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 ВСТАВИТЬ КОД-81
(код документа)
Группа ИВТ-461
(шифр группы)
Направление ??.??.?? Автоматизированные системы управления
(код по ОКСО, наименование направления, программы)

Руководитель работы _____ 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 _____ ????? ???????????
(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
Зав. кафедрой САПР и ПК, д.т.н.,

??.

_____ М. В. Щербаков
(подпись) (инициалы, фамилия)
«_____» _____ 2017

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

Тема Портирование сверточной нейросети на ARM архитектуру с ограниченными вычислительными ресурсами и ресурсами памяти
Утверждена приказом по университету от «??» ?????? 201? № ????–ст
Срок представления готовой работы _____
(дата, подпись студента)
Исходные данные для выполнения работы
задание, выданное научным руководителем с кафедры САПР и ПК, утвержденное приказом ректора

Содержание основной части пояснительной записки

Что-то там раз

Что-то там два

Перечень графического материала

1) Графический материал раз
2) Графический материал два

ВСТАВИТЬ КОД-81

Руководитель работы _____ 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
Зав. кафедрой САПР и ПК, д.т.н.,

??.

_____ М. В. Щербаков
(подпись) (инициалы, фамилия)
«_____» _____ 2017

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

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ВСТАВИТЬ КОД-81

Листов 26

Научный руководитель
старший преподаватель САПР и ПК
_____ А. В. Катаев
«_____» _____ 2017

Нормоконтролер
?????, ????
_____ ???????????
«_____» _____ 2017

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

Волгоград, 2017

ВСТАВИТЬ КОД-81

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 26 страниц и включает 5 рисунков и 0 таблицы.

ВСТАВИТЬ КОД-81

Содержание

ВСТАВИТЬ КОД-81

Введение

Задачи обработки и анализа аналоговой информации являюся одиними из самых сложных в IT-индустрии. Долгое время такие задачи решались евристическими 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 (НС). 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 многие другие применения. В частности, в данной дипломной работе используются нейронные сети для классификации и детектирования 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 потребляемые нейронной сетью ресурсы.

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 задачь, таких как автономное управление автомобилем.

В связи с вышесказанным целью данной дипломной работы является внедрение фреймворка машинного обучения на embedded систему C.H.I.P. и последующая его оптимизация. На основе проделанной работы необходимо сделать вывод о эффективности и рентабельности данного решения.

ВСТАВИТЬ КОД-81

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 сделать 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 устройстве C.H.I.P. и оптимизации работы алгоритма прямого прохода. Так же описана разработка клиент-серверной части для визуализации работы приложения.

ВСТАВИТЬ КОД-81

1 Обзор фреймворков машинного обучения

Данные раздел содержит справочную информацию, технические особенности и функциональные возможноти фреймворков глубоко машинного обучения и их сравнение. Так же 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 обертки для python и MATLAB 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). Одно из преимуществ Caffe это разделение модели данных от реализации. Что позволяет использовать приложения на разных платформах.

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

ВСТАВИТЬ КОД-81

1.1.1 Основные характеристики Caffe

Caffe представляет полный набор инструментов для обучения, тестирования, настройки и разработки можелей с подробной документацией и разобранными примерами. Поэтому процесс обучение использования фреймворка занимает короткий период. 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. Важным 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.[1]

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[2]. Так же доступны реккурентные модели[3].

ВСТАВИТЬ КОД-81

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 достоинст: минимальный размер строки при сериализации, эфективная 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 слой 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 обучения.[1]

ВСТАВИТЬ КОД-81

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 тратитб время и ресурсы на обучение. Важным пунктом является подробная документация для сериализации и использовании 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. [4]

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.[5]

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.

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 дает 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.).

ВСТАВИТЬ КОД-81

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 делает интегрирование различных библиотек тривиальной задачей. Библиотека Torch Tensor предоставляет множество классическ их операций (включая операции линецной алгебры), который реализованны и оптимизированны на C, используются SSE интсрукции для Intel платформ. Опцианально можно использовать высокопроизовидельные реализации BLAS/Lapack операций линейной алгебры. 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

– qt: предаставляет интерфейс работы Torch7 с Qt. Реализует конфертацию 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 имеют шаблонные витвления и рекурсии. На рисунке 2 показано создание многослойного перцептрона.

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

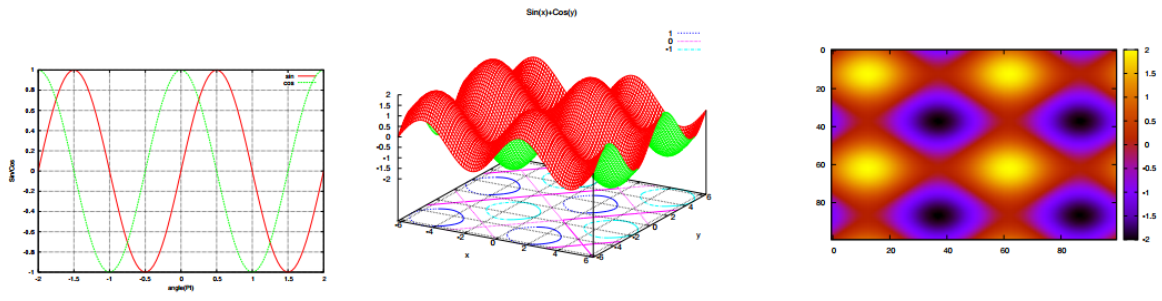


Рисунок 1 — Графики, полученные с помощую пакета plot фреймворка Torch7. Слева: простые синусоидальные функций. В центре: Поверхность, хранящаяся в 2D Tensor. Справа: Матричный график, построенный с использованием карты тепла

форматом Tensor или parallel, который предоставляет функции разветвления и исполнения Lua-кода на локальных или 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 и поддерживает вычисления как на центральном процессоре, так и на графическом процессоре.

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.

```
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

ВСТАВИТЬ КОД-81

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

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

```
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

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).

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 обладает общим интерфейсом для всях типов слоев, поэтому обладает большим набором параметров. Для расчета выходов и графиента слое, 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 архитектуре нейронных сетей.

Структура network определяет абстрактную модель для хранения внутренних параметров нейронной сети. Как и Caffe, Darknet разделяет представление и реализацию. Это реализуется разделением данных о 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

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

ВСТАВИТЬ КОД-81

1.4 Сравнение фреймворков машинного обучения

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 или 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.

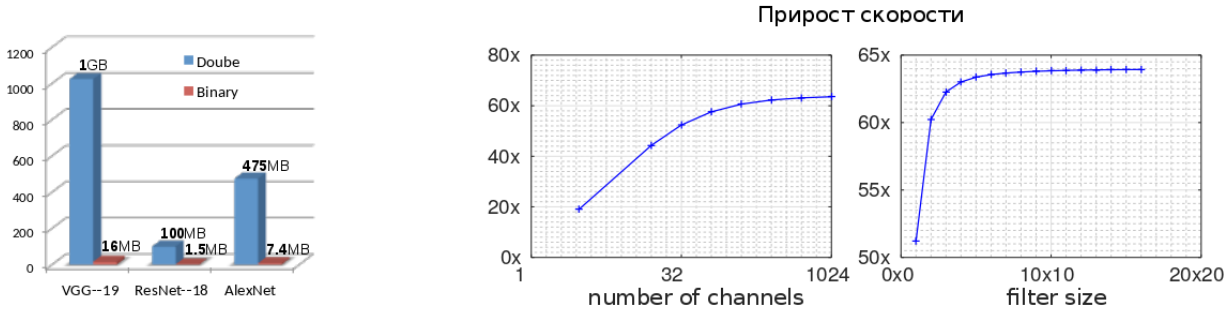
Если говорить о оптимизации работы с памятью, то аппроксимация фильтров и входов в Darknet позволяет значительно уменьшить объем выделяемой памяти. На рисунке 5 сравнение бинарной свертки и свертки с двойной точностью.

Caffe и Torch имеют достаточно большое количество зависимостей. Это объясняется желанием максимально ускорить

Рисунок 4 — Формат конфигурационного файла нейронной сети

процессы обучения и прохода нейронных сетей, однако наклабывает ограничесние на срециализированное оборудование и оборудование с ограниченными запасами физической памяти.

Суммировав все показатели, можно сделать вывод, что Darknet



(a) (b) (c)

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

ВСТАВИТЬ КОД-81

2 Используемые алгоритмы и модели

2.1 Теоретические основы нейронных сетей

2.1.1 Перцептрон - основа нейронных сетей

2.1.2 Алгорим обратного распространения ошибки

2.1.3 Сверточные нейронные сети

2.1.4 Обнаружение объектов с применем подхода YOLO

2.2 Оптимизация работы с памятью

2.2.1 Бинаризация весов

2.2.2 Оптимизация работы со слоями

ВСТАВИТЬ КОД-81

3 Проектирование системы

ВСТАВИТЬ КОД-81

Заключение

ВСТАВИТЬ КОД-81

Список использованных источников

1 https://arxiv.org/pdf/1408.5093.pdf

2 J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In ICML, 2014

3 A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012

4 http://ronan.collobert.com/pub/matos/2011_torch7_nipsw.pdf

5 http://www.lua.ru/doc/1.html

6 https://pjreddie.com/media/files/papers/xnor.pdf

ВСТАВИТЬ КОД-81

Приложение А
Техническое задание