Использование высокопроизводительной платформы глубокого обучения для ускорения обнаружения объектов

С.О. Степаненко[1], П.Ю. Якимов[1,2]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086
[2]Институт систем обработки изображений РАН - филиал ФНИЦ «Кристаллография и фотоника» РАН, Молодогвардейская 151, Самара, Россия, 443001

Аннотация. Классификация объектов с использованием нейронных сетей крайне актуальна на сегодняшний день. YOLO – это один из самых используемых фреймворков для классификации объектов. Он имеет достаточно высокую точность, но скорость обработки недостаточно высока, особенно в условиях ограниченной производительности вычислителя. В настоящей статье исследуется применение фреймворка NVIDIA TensorRT для оптимизации работы YOLO с целью увеличения скорости обработки изображений. Сохраняя эффективность и качество работы нейронной сети, TensorRT позволяет увеличить скорость обработки, используя оптимизацию архитектуры и вычислений на видеокарте.

1. Введение

Задача обнаружения объектов становится все более популярной. Это связано с тем, что появились мощные вычислительные устройства, и для обнаружения объектов стало возможным использовать нейронные сети, которые могут находить объекты на изображении с большой точностью. Создать свою систему, которая работает на основе искусственной нейронной сети, не является большой проблемой, так как существует большой ряд различных фреймворков, которые упрощают создание нейронной сети, сводя разработку сети к вызову функций. Проблема обнаружения объектов заключается в том, что требуется большая вычислительная мощность, и в реальных задачах, например обработки видеопотока, требуется иметь мощное оборудование. Проблема обнаружения объектов на изображении с помощью нейронных сетей в том, что требуется много вычислительной мощности, особенно, если обрабатывается видеопоток.

Сегодня существует множество решений для обнаружения объектов. Все они используют разные алгоритмы для обнаружения, могут обнаруживать с разной точностью и иметь разную скорость работы. Большинство существующих решений используют CUDA [1] для параллельной обработки данных. С помощью CUDA можно увеличить скорость работы, но также существуют и другие способы ускорить работу. Можно оптимизировать архитектуру нейронной сети, сделав ее работу более быстрой, оставив точность на том же уровне. Но это не всегда легко сделать, особенно, если сеть имеет очень сложную архитектуру. Есть возможность ускорить работу нейронной сети, не тратя много времени на изменение программы. Существует платформа, которая способна ускорить работу нейронной сети, используя алгоритмы оптимизации архитектуры, а также возможности видеокарт NVIDIA для ускорения

вычислений. Эта платформа называется TensorRT [2]. TensorRT предлагает API для создания нейронных сетей, а также позволяет оптимизировать модели многих популярных фреймворках. Это делает ее удобной в использовании во многих случаях тем, что можно ускорить работу программы, не тратя много ресурсов на изменение кода.

2. Технологии инференса сверточных нейронных сетей

Под словом инференс подразумевается получения результата работы нейронной сети, которая была обучена на некотором объеме данных.

В данной статье рассматривается применение платформы TensorRT для ускорения работы алгоритма для обнаружения объектов YOLO [3].

2.1. YOLO

YOLO [4] – это алгоритм классификации и детектирования объектов, использующий для этого сверточные нейронные сети. Плюсы сверточных нейронных сетей для задач этого типа в том, что сверточные нейронные сети могут работать с изображениями, при этом имея более простую архитектуру, чем стандартные нейронные сети. Существует множество реализаций YOLO на различных фреймворках и языках. Стандартная реализация основана на нейронной сети darknet, которая написана на языке программирования C. Работа YOLO начинается с изменения размера входного изображения. Он становится 448x448x3, где 448x448 – размер изображения, 3 – количество цветовых каналов. Сначала изображение пропускается через модифицированную сеть GoogleNet. Это первые 20 слоев сети. На выходе из этой части сети есть 1024 карт признаков размером 14x14. Затем изображения проходит через последовательность сверточных слоев и слоев пулинга. На момент попадания в полно связный слой, имеется 1024 карт признаков размерами 7x7. Пройдя через два полно связных слоя, сеть дает предсказания принадлежности объекта к некоторому классу и положение объекта на изображении [4]. Данная архитектура показана на рисунке 1.

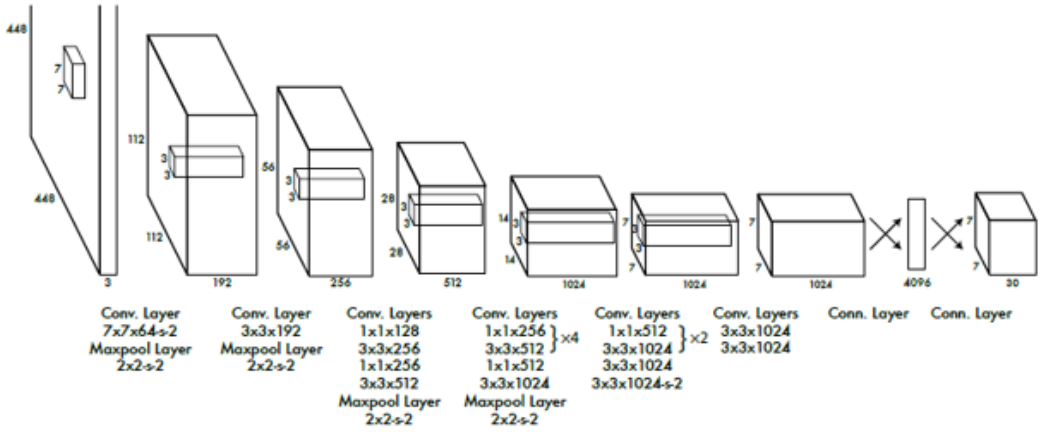

Рисунок 1. Архитектура нейронной сети YOLO.

Для определения границ объекта в алгоритме YOLO сначала накладывается сетка размером SxS. Далее для каждой сетки происходит предсказание объекта. Для каждого элемента сетки создается вектор размером 5*B+C, где B – количество границ, которые предсказывает элемент сетки, C – количество классов, которые способна предсказать сеть, 5 – определяет количество объектов, которые могут быть найдены. Первые 5*B значений вектора показывают координаты центра границы внутри ячейки сетки, высоту и ширину и вероятность того, что граница определена верно. Остальные C значений показывают вероятности нахождения центра объекта в центре данной ячейки. В итоге имеется S*S*B границ объектов с вероятностями классов. Затем вектор сортируются по убыванию и к ним применяется алгоритм Non maximal suppression. Так происходит со всеми классами. В итоге просматриваются все границы. Для каждой границы учитывается максимальная вероятность по классам и, если она положительная, то граница наносится на изображение [5].

2.2. TensorRT

TensorRT – платформа глубокого обучения от компании NVIDIA [6]. На данный момент существует пять версий TensorRT. Каждая новая версия способна работать с большим числом типов слоев нейронной сети и с большим количеством математических операций. TensorRT позволяет использовать реализованные парсеры для многих популярных фреймворков. К ним относятся: TensorFlow, Caffe2, PyTorch, Mxnet, Microsoft Cognitive Toolkit, Chainer. Tensorflow же имеет встроенную поддержку TensorRT версии 3.0 [2]. В случае, когда сеть создана на этих фреймворках, использовать TensorRT очень просто. Достаточно лишь использовать готовый парсер. Для TensorFlow процесс создания сети с использованием парсера TensorRT показан на рисунке 2. Если сеть не создана на этих фреймворках, возможно использовать API TensorRT для переноса модели сети.

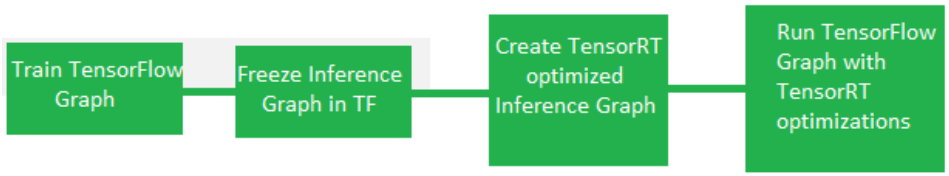

Рисунок 2. Процесс создания сети с использованием TensorFlow на TensorRT.

Преимущество в использовании TensorRT заключается в том, что данная платформа способна ускорить работу нейронной сети, используя алгоритм упрощения архитектуры сети, не меняя функционала самой сети, и используя возможности видеокарт от NVIDIA для ускорения вычислений.

Для упрощения архитектуры сети, TensorRT анализирует граф, представляющий модель сети. Если в графе содержатся элементы, которые повторяются, TensorRT объединяет их. В итоге размер сети становится меньше.

Ускорение на видеокарте происходит из-за возможности работы с «Тензор Ядрами». Данные ядра представляют возможность использовать для вычислений тип данных половинной точности float16. При использовании CUDA этого нельзя достичь. CUDA позволяет работать с типом данных float32. Скорость работы возрастает из-за более быстрой передачи данных и более быстрых вычислений с таким типом данных. Данный тип ускорения возможен лишь на малом количестве моделей видеокарт, в которых есть поддержка данной технологии.

3. Реализация YOLO

Для сравнения скорости работы были взяты реализации алгоритма YOLO с использованием платформы TensorRT и без использования на одном наборе данных и на одинаковых обученных моделях.

3.1. Реализация YOLO без использования платформы TensorRT.

3.1.1. Darknet

Для сравнения производительности в данной статье была рассмотрена одна из реализаций алгоритма YOLO на нейронной сети darknet [3]. Были произведены запуски YOLO на GPU, для чего понадобилось установить CUDA 10.0, а также OpenCV. В качестве модели была взята готовая модель YOLOv2. Перед запуском необходимо произвести сборку проекта. Это можно сделать, запустив команду make из папки с проектом. После установки появится исполняемый файл, который необходимо запускать. Для запуска достаточно указать команду с нужным набором опций. Команда для запуска программы имеет следующий вид: ./darknet detect path_to_cfg_file path_to_weights_file.

Darknet позволяет также производить обработку видео из файла и из веб камеры.

3.1.2. Darkflow

Еще одна реализация алгоритма YOLO на языке Python, которая использует Tensorflow. Для того, чтобы запустить данную программу, необходимо установить CUDA 9.0, Tensorflow 1.0,

numpy, OpenCv версии 3.0 и выше. Для того, чтобы запустить Darkflow и другие реализации на одном компьютере, необходимо иметь сразу CUDA 10.0 и CUDA 9.0. Чтобы сменить версию CUDA, которую используют приложения, достаточно обновить переменные окружения. Darkflow имеет возможность обрабатывать видеопоток. Перед запуском необходимо запустить скрипт установки. После установки можно запускать программу, используя команду:
flow --model path_to_cfg_file --load path_to_weights_file --imgdir path_to_folder_with_images –gpu percent, где percent – число от 0 до 1, показывающее процент использования видеокарты. 0 – 0% использования. 1 – 100% использования. Без указания опции --gpu будет происходить обработка на центральном процессоре. Скорее всего в таком случае скорость обработки будет значительно ниже, чем в случае обработки на видеокарте.

3.2. Реализация YOLO с применением платформы TensorRT.
В данной статье представлена реализация YOLO вместе с TensorRT 5.0 [7]. Перед запуском программы необходимо установить все зависимости. Для работы программы необходимы: CUDA 10.0, TensorRT 5.0, OpenCV 3.4.0. Для запуска нужны файлы весов обученной модели и файлы конфигурации сети. Их можно найти на официальном сайте разработчиков YOLO. Для примера была взята обученная модель YOLOv2. Данная модель способна обнаруживать 80 классов объектов. Сначала необходимо установить проект, используя make. После произвести настройку, указав пути к всем зависимостям и файлам весов и конфигурации. Затем стоит выбрать тип данных, с которым будет происходить работа. Возможен выбор Float32, Float16 и Int8. В случае, если на видеокарте не поддерживаются тензор ядра, программа может запуститься лишь на Float32. Есть возможность обрабатывать изображения не по одному, а сразу батчами. Обработка видео возможна лишь, если дополнительно использовать DeepStream SDK. DeepStream SDK разработан компанией NVIDIA для обработки потоковых данных. Он использует TensorRT, CUDA, Video Codec SDK. На данный момент максимальная версия Deepstream SDK версии 3.0. Без использования DeepStream SDK обработку видео можно произвести, изменив исходный код, сделав возможным извлечение кадров из видеопотока. Такую возможность предоставляет библиотека OpenCV.Для запуска необходимо ввести следующую команду:
trt-yolo-app
Для данной команды доступны следующие опции:

- Batch_size – количество изображений, которые одновременно обрабатываются
- Decode – Принимает True или False. Показывает, нужно ли декодировать изображения. По умолчанию true.
- Seed – параметр для генератора случайных чисел.

После окончания работы в папку сохранятся файлы с обработанными изображениями. Для работы с видео можно использовать возможность OpenCV, которая выделяет кадры из видео потока. Еще один вариант использования обработки видео – использовать deepstream, библиотеку от NVIDIA для работы с потоками. Deepstream использует библиотеки для ускорения обработки потоков, а также использует TensorRT с CUDA.

4. Экспериментальные исследования

Для экспериментального исследования были произведены запуски двух реализаций алгоритма YOLO на одинаковой обученной модели YOLOv2. Были взяты 2416 изображений в качестве входных данных. На выходе, изображения, где удалось сделать обнаружить объекты, сохранялись в отдельную папку. В реализации без TensorRT время обработки каждого изображения записывалось в файл, а затем подсчитывалось среднее время обработки. При запуске реализации с применением TensorRT, время обработки каждого изображения не учитывалось, а сразу считалось среднее. Все эксперименты производились на компьютере с характеристиками, представленными в таблице 1.
Среднее время обработки набора изображений каждой реализацией представлено в таблице 2.

Таблица 1. Основные характеристики компьютера.

GPU	CPU	Память
NVIDIA GeForce GT 710	AMD FX-4300	4 ГБ памяти
NVIDIA GeForce GTX 950	Intel Core i5-6500	8 ГБ памяти

Таблица 2. Время работы YOLO в секундах.

GPU	Darkflow	Darknet	Yolo с TensorRT
NVIDIA GeForce GT 710	2.28	0.78	0.2
NVIDIA GeForce GTX 950	0.98	0.21	0.02

Самой медленной оказалась реализация алгоритма YOLO на tensorflow. Время работы данной реализации больше в 2.92 раза для GT710 и в 4.7 раз для GTX 950, чем время работы стандартной реализации алгоритма на сети darknet. Платформа TensorRT оказалась самой быстрой. Время работы алгоритма, реализованного на платформе TensorRT, оказалось в 3.9 раз меньше для GT 710 и в 10.5 раз меньше для GTX 950, чем время работы на сети darknet. Учитывая, что запуск алгоритма на платформе TensorRT осуществлялся с типом данных Float32, запустив его на Float16, время работы будет еще меньше. К сожалению, имеющаяся видеокарта не способна производить такие вычисления, поэтому эксперименты производились только с типом данных Float32.

Для реализации на TensorRT было сравнено время работы при различных размерах батча. Размеры батча изменялись от 1 до 16. После 16 не получалось выделить память на видеокарте. На рисунках 3 и 4 изображено время работы для различных размеров батчей для двух разных видеокарт.

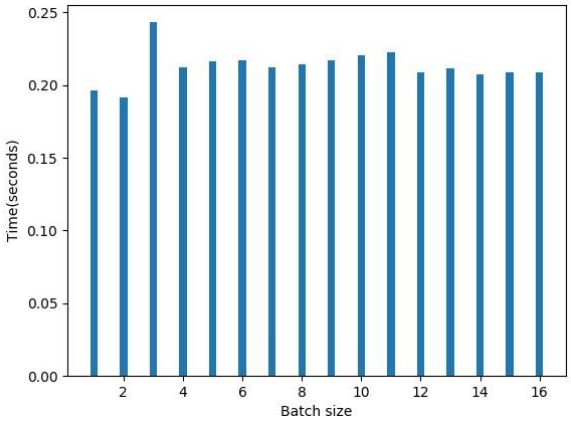

Рисунок 3. Время работы алгоритма при различных размерах батчей для NVIDIA GeForce GT 710.

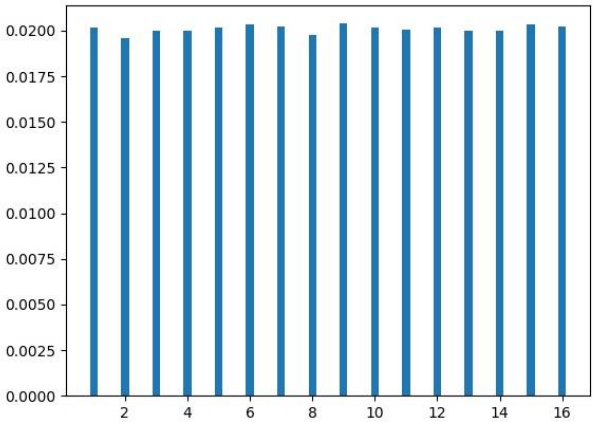

Рисунок 4. Время работы алгоритма при различных размерах батчей для NVIDIA GeForce GTX 950.

Различия во времени работы незначительны для обоих видеокарт. Разница между худшим и лучшим результатами примерно в 1.27 раз. Это может быть связано с различными причинами, и определить заранее оптимальный размер сложно. Приходится делать это экспериментально.

5. Заключение

В статье были рассмотрены 3 реализации алгоритма YOLO, для сравнения производительности. Одна из этих реализаций использует платформу TensorRT. Сама платформа способна ускорять алгоритм, работая с той же точностью. Данная возможность может быть использована на практике в обработке видеопотока, где скорость обработки является важным показателем. При использовании TensorRT время работы сократилось почти в 4 раза, по сравнению с стандартной реализацией алгоритма, при условии, что не была использована возможность видеокарт производить вычисления, используя тензор ядра, из-за того, что имеющаяся видеокарта не может производить данные вычисления.

6. Литература

[1] CUDA [Electronic resource]. – Access mode: https://developer.nvidia.com/cuda-gpus (01.11.2018).
[2] Официальный сайт TensorRT [Электронный ресурс]. – Режим доступа: https://developer. nvidia.com/tensorrt (01.11.2018).
[3] YOLO: Real-Time Object Detection [Electronic resource]. – Access mode: https://pjreddie.com /darknet/yolo/ (01.11.2018).
[4] Redmon, J. You Only Look Once: Unified, Real-Time Object Detection / J. Redmon, S. Divvala, R. Girshick, A. Farhadi // You Look Only Once, 2015. – 10 p.
[5] Redmon, J. YOLO9000: Better, Faster, Stronger / J. Redmon, A. Farhadi // University of Washington, Allen Institute for AI, 2017. – 9 с.
[6] TensorRT integration speeds up tensorflow inference [Electronic resource]. – Access mode: https://devblogs.nvidia.com/tensorrt-integration-speeds-tensorflow-inference/ (01.11.2018).
[7] Реализация YOLO с TensorRT [Электронный ресурс] – Режим доступа: https://github.com/vat-nvidia/deepstream-plugins/ (01.11.2018).

Благодарности

Работа выполнена при финансовой поддержке гранта РФФИ № 17-29-03112 офи_м.

Using high-performance deep learning inference platform to accelerate object detection

S.O. Stepanenko[1], P.Y. Yakimov[1,2]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086
[2]Image Processing Systems Institute of RAS - Branch of the FSRC "Crystallography and Photonics" RAS, Molodogvardejskaya street 151, Samara, Russia, 443001

Abstract. Object classification with using of neural networks is extremely current today. YOLO is one of the most used frameworks for object classification. It has fairly high accuracy but the processing speed is not high enough, especially in conditions of limited performance of a computer. This article researches the using of NVIDIA TensorRT to optimize YOLO work with the aim of increasing of image processing speed. Saving efficiency and quality of the neural network work TensorRT enables to increase the processing speed using an optimization of the architecture and an optimization of calculations on a GPU.