Санкт-Петербургский государственный университет

Матмематическое обеспечение и администрирование информационных систем
Кафедра системного программирования

Железняков Иван Эдуардович

Применение свёрточных нейросетей для обнаружения объектов дорожной обстановки с малой вычислительной сложностью

Выпускная квалификационная работа

Научный руководитель:
к. ф.-м. н., ст. преп. СПбГУ Салищев С. И.

Рецензент:
к. т. н., доцент ГУАП Рожнин А. Л.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Software and Administration of Information Systems
Software Engineering

Ivan Zheleznyakov

Usage of convolutional neural networks for the detection of road objects with low computational complexity

Graduation Thesis

Scientific supervisor:
PhD, seniour lecturer Sergey Salishev

Reviewer:
PhD, assoc. prof. Alexander Rozhnin

Saint-Petersburg
2019

Оглавление

Введение

В настоящее время уже ни для кого не секрет, что технологии, основанные на методах машинного обучения с использованием нейронных сетей, являются одной из передовых и наиболее перспективной сферой исследований [6]. Оценка дорожной обстановки - одна из важнейших областей использования нейронных сетей как минимум потому, что автомобили с системой автопилота находятся в числе самых ожидаемых технологий нашего будущего.

Автономное управление и помощь водителю транспортных средств является одной из наиболее динамичных областей развития автомобильной электроники и информационных систем в области транспорта. Наиболее точными алгоритмами для анализа дорожной обстановки на сегодняшний день являются алгоритмы на основе сверточных нейросетей. Системы на чипе для автомобильной электроники следующего поколения имеют поддержку специализированных ядер для реализации алгоритмов машинного обучения, в том числе сверточных нейросетей. При этом они имеют ограниченную вычислительную мощность и размер памяти, не позволяющий реализовать наиболее современные алгоритмы с наивысшей точностью и использовать стандартные библиотеки машинного обучения. Для уменьшения вычислительной сложности и размера памяти используется сокращение размерности сетей и уменьшение точности вычислений.

1. Постановка задачи

Целью работы было получение оценок точности классификации, временной сложности и размера памяти для современных архитектур нейросетей для классификации изображений с сокращенной точностью вычислений и размерностью в применении к задаче классификации дорожной обстановки для оценки возможности реализации на системах на чипе для применения в автомобилях следующего поколения.

Для ее достижения были поставлены следующие задачи:

- изучить и установить проект darknet [19] и все необходимые для него технологии;
- изучить архитектуру сетей Yolov3 [15], Yolov3-tiny и квантизованную сеть Yolov3-tiny с классификатором XNOR-net [18] и сравнить их;
- провести оптимизацию обучения, если потребуется;
- обучить сети на наборе данных дорожной обстановки и получить оценки точности, размера памяти и времени инференса.
- оценить возможность внедрения системы на акселератор ?

2. Обзор

2.1. Конфигурация и необходимые требования

Вкратце об используемых технологиях и машине, на которой проводилась работа.

- ОС: Windows 10
- GPU: NVIDIA GeForce MX150 2 GB GPU-RAM
- CMake 3.14 [1] - расширяемая система с открытым исходным кодом для управления процессом сборки приложений в операционной система независимо от компилятора
- CUDA Toolkit 10.0 [17] - среда разработки для создания высокопроизводительных приложений с методами ускорения GPU
- OpenCV 3.4.6 [13] - open-source библиотека для компьютерного зрения и машинного обучения, насчитывающая более 2500 оптимизированных алгоритмов
- cuDNN 7.5.0 для CUDA Toolkit [11] - библиотека, обеспечивающая ускоренную работу GPU в нейронных сетях

2.2. Терминология

Определим некоторые понятия, которые будут приведены в работе. Одна из самых популярных метрик для измерения точности обнаружения является mAP (mean average precision [9]). Для объяснения ее смысла сначала обратимся к матрице ошибок (1).

Таблица 1: Матрица ошибок

	Предсказан класс A: да	Предсказан класс A: нет
Класс A: да	True Positive	False Positive
Класс A: нет	False Negative	True Negative

Точность (precision) показывает, насколько точны предсказания, то есть процент правильно предсказанных классов.

Полнота (recall) - это доля найденных сетью объектов, принадлежащих классу, относительно всех объектов данного класса в выборке.

Математическое определение:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

IoU (Intersection over Union) измеряет покрытие двух границ: предсказанной классификатором и реальной, которая обрамляет объект на изображении (1).

$$IoU = \frac{overlap}{union}$$

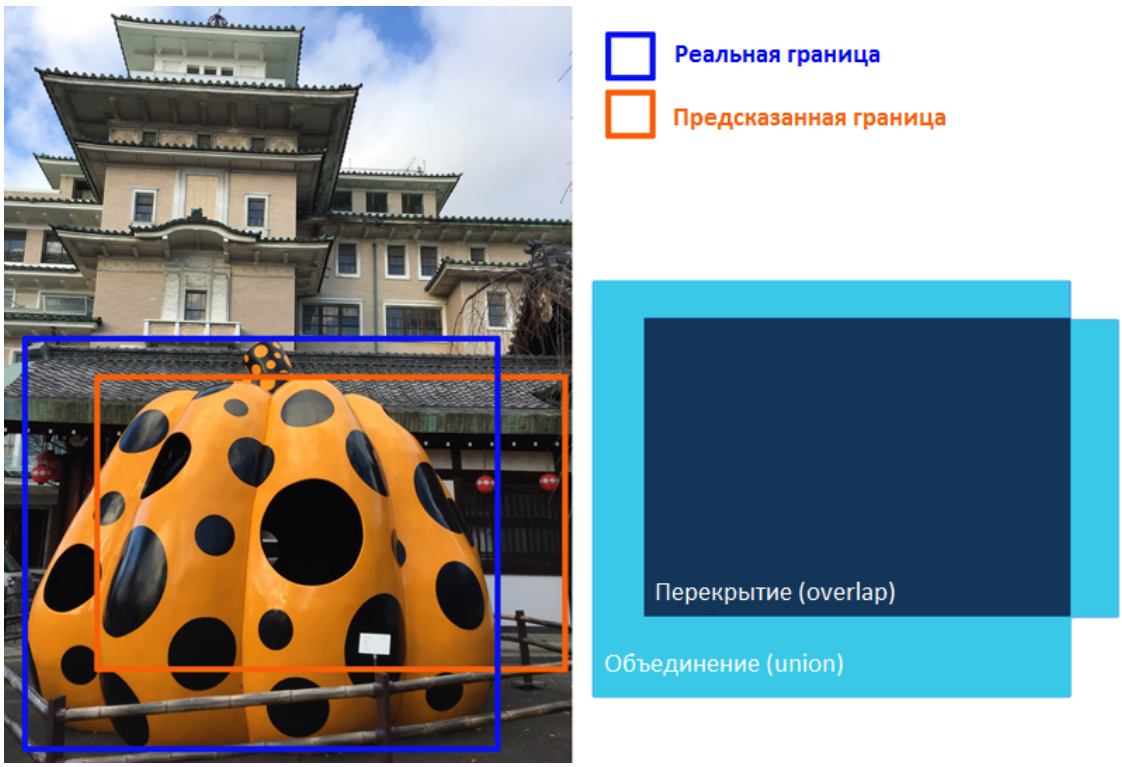

Рис. 1: Пример IoU

Вернемся к определению mAP. Это интегральный параметр, который является средним значением AP (average precision) для каждого класса из обучающей выборки.

$$AP = \frac{1}{11} \sum_{r \in (0, 0.1, \ldots 1)} P(r)$$

$$P - precision$$

$$r - recall$$

Мы будем использовать метрику **mAP@0.5**, которая означает, что объект будет предсказан правильно при $IoU > 0.5$.

2.3. Инструменты исследования, существующие решения

В качестве инструментов исследования были выбраны система обнаружения объектов в реальном времени YOLO (You Only Look Once) с open-source фреймворком Darknet (версия с улучшениями пользователя github AlexeyAB [20]), написанным на C и CUDA. Этот выбор был обоснован факторами, которые приведены ниже.

2.3.1. Скорость работы и точность обнаружения

Yolo v3 почти не уступает по точности таким системам обнаружения как RetinaNet [4], SSD [16], при этом по времени инференса обгоняет многие известные сети в несколько раз (результаты приведены в работе [15]). Более того, в Yolo v3 представлена возможность выбирать оптимальное соотношение скорости выполнения и точности, просто меняя размер модели без переобучения.

2.3.2. Актуальность и новизна

Производители чипов для автоэлектроники делают новые чипы с акселератором сверточных нейронных сетей [14], но они заточены на старых сетях, например, AlexNet. Хотелось бы рассмотреть относительно новую систему обнаружения Yolo v3 и оценить, способна ли она лечь в основе современных чипов. Также разработчики Darknet для сравнения Yolo v3 с другими сетями использовали в основном набор данных COCO [10], в этой же работе в качестве обучающих данных представлен датасет cityscapes [2].

2.3.3. Ограниченность вычислительной мощности и размера процессора

Как уже было сказано, Yolo v3 имеет несколько моделей для разных задач. Принято решение в рамках данной работы использовать модель tiny, которая уступает по точности остальным моделям, но работает

быстрее и требует меньшего размера памяти.

2.3.4. XNOR-net

Одним из преимуществ версии фреймворка AlexeyAB является то, что в ней некоторые модели реализованы с технологией XNOR-net. XNOR-сети аппроксимируют конволюционные вычисления, используя в основном двоичные операции. Как утверждают авторы, данный подход может привести к значительному ускорению производительности и экономии памяти.

2.4. Архитектура YOLO v3

Одно из основных преимуществ модели YOLO заключается в его названии - You Only Look Once. Графическая интерпретация архитектуры инференса YOLO представлена на рисунке 2.

Рис. 2: Архитектура инференса YOLO

Входящее изображение по умолчанию изменяет размер до 448x448 и пропускается через модифицированную архитектуру GoogLeNet [5], на выходе получаем некоторый набор feature map, затем происходит несколько конволюций, в ходе которого происходит изменение пространственной размерности. В результате получаем тензор 7x7x30 (в случае 20 классов). В этом тензоре хранится вся детекция. На исходное изображение накладывается сетка 7x7. Каждой ячейки сетки сопоставлен

30-мерный вектор. Первые 10 чисел содержат информацию о координатах и размерах двух обрамляющих прямоугольников для ячейки, а также некоторая уверенность того, что прямоугольники правильно нашли детектируемый объект. Остальные 20 чисел соответствуют объектам классификации и показывают уверенность того, что центр объекта лежит внутри ячейки вне зависимости от различных обрамляющих прямоугольников. Далее оценка уверенности для каждой ячейки умножается на вероятность каждого класса, чтобы получить окончательную оценку.

Текущая версия сверточной нейросети Yolo v3 содержит 75 сверточных слоев. В модели tiny их количество уменьшено до 16, что значительно сокращает время инференса и размер памяти, но увеличивает время обучения.

3. Набор данных для дорожной обстановки

В качестве обучающих данных был выбран датасет cityscapes, содержащий записи уличных сцен в 50 городах. Всего в наборе данных представлено 30 классов, из которых по причинам оптимизации было отобрано 9. Каждому из 4000 изображений датасета (3300 из которых - обучающие данные, 700 - тестирующие) соответствует json-файл, внутри которого содержится информация о высоте и ширине изображения, а также маркировка объектов. Обозначение классов представлено следующим образом:


```
{
    "imgHeight": 1024,
    "imgWidth": 2048,
    "objects": [
        {
            "label": "car",
            "polygon": [
                [
                    281,
                    430
                ],
                [
                    196,
                    429
                ],
                [
                    181,
                    436
                ],
...
        }
        "label": "person",
```

```
"polygon": [
...]
},
...
```

То есть каждый объект представлен координатами вершин многоугольника, в котором он находится. Yolo требует иную разметку. С каждый изображением в той же самой директории должен находиться txt-файл с идентичным названием, внутри которого находятся идентификатор объекта и координаты прямоугольника, в котором он содержится, для каждого объекта новая строка:

```
<object-class> <x_center> <y_center> <width> <height>
```

где `<object-class>` - целое число от 0 до (кол-во классов - 1);
`<x_center> <y_center> <width> <height>` - координаты центра прямоугольника, в котором находится объект, его ширина и высота, значения в промежутке от 0.0 до 1.0. Для маркировки датасета cityscapes в соответствии с yolo был написан скрипт на C# с использованием библиотеки Newtonsoft [12] для .NET-платформы с удобной реализацией работы с файлами формата json. Переразметка проводилась следующим образом. Для каждого объекта по каждой из вершин многоугольника находим минимумы и максимумы по X и по Y. Собственно, пары (maxX, maxY), (maxX, minY), (minX, maxY), (minX, minY) образуют координаты вершин прямоугольника, по которым уже легко находим его центр, высоту и ширину.

4. Адаптация системы

Во фреймворке Darknet реализованы некоторые инструменты для получения информации об обучении. В рамках данной работы в исходный код проекта были внесены изменения.

- По умолчанию Darknet сохраняет полученные во время обучения веса каждые 1000 итераций и перезаписывает их в отдельный файл каждые 100 итераций. Это сделано для того, чтобы можно было прервать обучение и продолжить через некоторое время, а также для подсчета mAP для разных весов, чтобы избежать переобучения. Реализована опция, благодаря которой можно считать mAP для сохраненных весов во время обучения и запоминать наиболее точные из них.

- Реализована возможность периодического вывода на консоль времени, прошедшего с начала обучения.

- Во время обучения строится график, показывающий изменения mAP и функции потери. Но если прервать обучение и продолжить его через некоторое время с сохраненными весами, предыдущая информация об этих значениях пропадет. Теперь же есть возможность записывать временные mAP и работать с ними даже после паузы в обучении.

5. Оптимизация

Первым делом нужно было посмотреть первоначальные результаты обучения сетей Yolo v3 и Yolo v3-tiny с произвольными весами. В таблице 2 представлено сравнение.

Таблица 2: Сравнение

Сеть	Количество итераций	Среднее значение функции потери	Максимальное значение mAP	Размер весов (МБ)
Yolo v3	5000	90	13%	250
Yolov3-tiny	7000	102	10%	40,2
Yolov3-tiny XNOR	7000	125	6%	39,8

Несмотря на то, что сеть Yolo v3 показала наилучший результат точности, выбор был сделан в пользу двух других сетей, так как размер памяти очень важен в оценке возможности использовать нейросети в системах на чипе.
Было проведено еще 3000 итерации, в ходе которых особых улучшений не было. Встал вопрос об оптимизации обучения.

Разработчики Darknet и YOLO предоставляют предобученные веса на различных наборах данных (COCO (таблица 3), ImageNet [8] (таблица 4)). Их использование вместо произвольных весов значительно улучшило скорость и качество обучения.

Наличие большого количества классов уменьшает точность обнаружения и увеличивает время инференса. Некоторые классы были удалены:

- занимающие много места на изображении и перекрывающие другие (напр. sky, road, building);
- не являющиеся слишком важными для обнаружения на дорожной обстановке (напр. parking, bridge, rail track).

Таблица 3: Обучение сетей с использованием предобученных весов на наборе данных COCO

Сеть	Количество итераций	Среднее значение функции потери	Максимальное значение mAP
Yolov3-tiny	7000	82	15%
Yolov3-tiny XNOR	7000	108	8%

Таблица 4: Обучение сетей с использованием предобученных весов на наборе данных ImageNet

Сеть	Количество итераций	Среднее значение функции потери	Максимальное значение mAP
Yolov3-tiny	7000	70	18%
Yolov3-tiny XNOR	7000	95	11%

В таблице 5 представлены оставшиеся классы и их отношение к общему количеству объектов из обучающей выборки.

Из конкретных алгоритмов оптимизации работы нейронных сетей среди ADAM, RMSProp и Adagard был выбран первый. Данные алгоритмы содержат в себе принцип более слабого обновления весов для типичных признаков и уделяет больше внимания классам, которые сеть тяжело распознает. Проведенные исследования [7] [3] показывают, что он лучше остальных справляется с выходом из локальных минимумов функции потери, показывает наибольшую производительность при работе с с имеющимися предобученными весами.

Таблица 5: Классы из обучающего набора данных

Класс	Количество объектов	Отношение к общему количеству
person	21413	14,3%
car	31822	21,2%
truck	582	0,3%
bus	483	0,3%
motorcycle	888	0,5%
bicycle	4904	3,2%
pole	52748	35,2%
traffic sign	24976	16,6%
traffic light	11898	7,9%

6. Результаты обучения

Каждое принятое по оптимизации решение сопровождалось повторным переобучением сетей Yolov3-tiny и Yolov3-tiny с классификатором XNOR, требующим от 5 до 10 тысяч итераций для выявления улучшений. В среднем 1000 итераций Yolov3-tiny обучается 4 часа, Yolov3-tiny(XNOR) 3 часа 43 минуты. Финальное обучение (18000 итераций) для сети Yolov3-tiny составило 72 часа 35 минут, для квантизованной Yolov3-tiny(XNOR) - 64 часа 50 минут.

Обе из рассматриваемых сетей стали лучше обучаться (6).

Таблица 6: Результаты финального обучения (1)

Сеть	Среднее значение функции потери	Максимальное значение mAP
Yolov3-tiny Yolov3-tiny XNOR	32 73	24% 14%

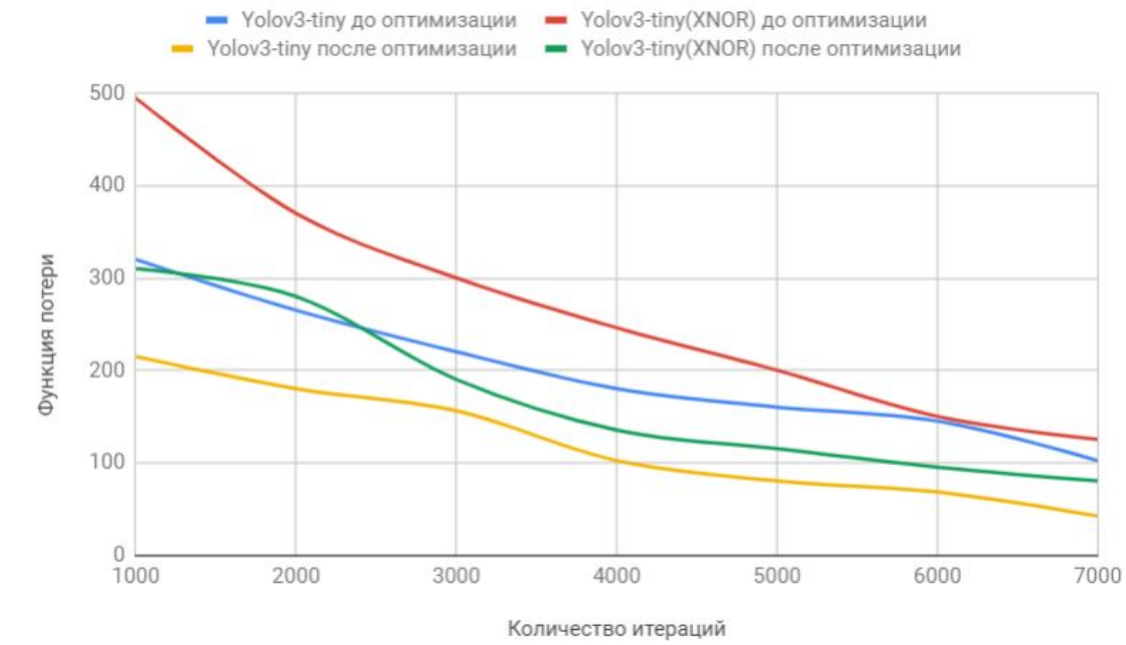

Рис. 3: График изменения функции потери

Были подсчитаны значения средней точности обнаружения по отдельным классам для каждой из сетей (7). Лучше всего сети находят большие объекты (car, bus) и людей (person), хуже справляются с обнаружением небольших объектов (pole, traffic light).

Таблица 7: AP для каждого класса

Класс	Yolov3-tiny	Yolov3-tiny (XNOR)
person	24,8%	17,2%
car	53,9%	38,2%
truck	12,5%	6,7%
bus	35,7%	21,8%
motorcycle	13,5%	7,6%
bicycle	18,1%	14,2%
pole	9,5%	6,8%
traffic sign	11,8%	7,4%
traffic light	8,8%	3,3%

Квантизованная сеть справилась со своей основной задачей: увеличение скорости инференса и уменьшение объема памяти (8). Но ка-

чество обучения при этом стала ниже. Это объясняется несколькими причинами: во-первых, часть техник обучения XNOR пока не реализованы в открытом доступе; во-вторых, по словам автором, модель Tiny не самым лучшим образом оптимизирована для XNOR.

Таблица 8: Результаты финального обучения (2)

Сеть	Среднее время инференса для одного изображения (мс)	Размер весов (МБ)
Yolov3-tiny	15,5	34,1
Yolov3-tiny XNOR	10,3	31,8

Рис. 4: Оригинальное изображение

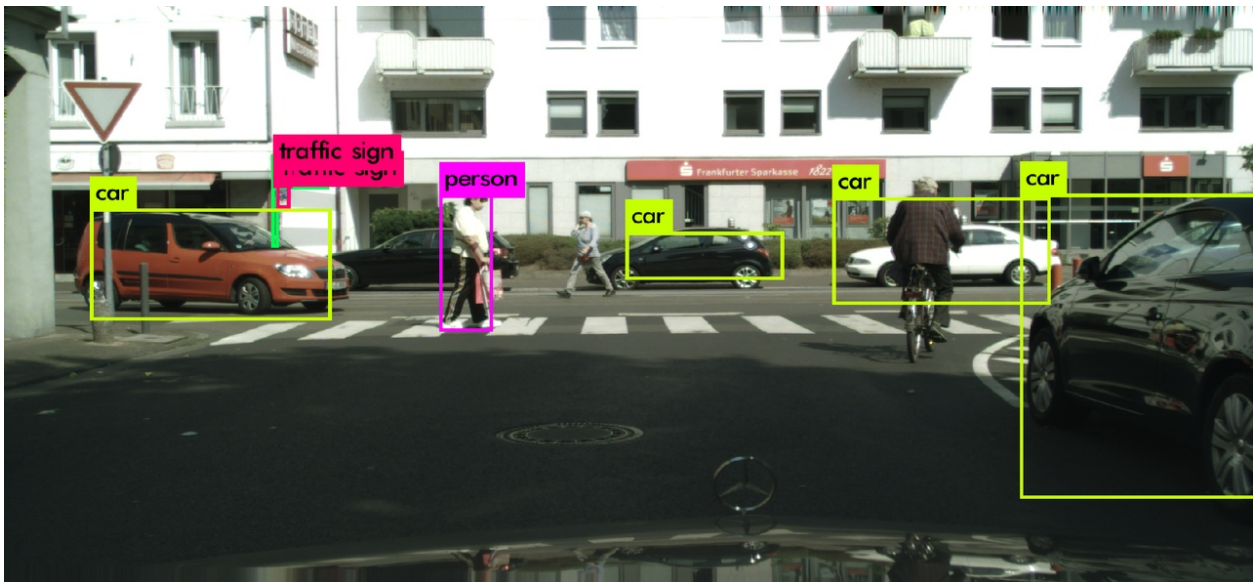

Рис. 5: Обнаружение yolov3-tiny(XNOR)

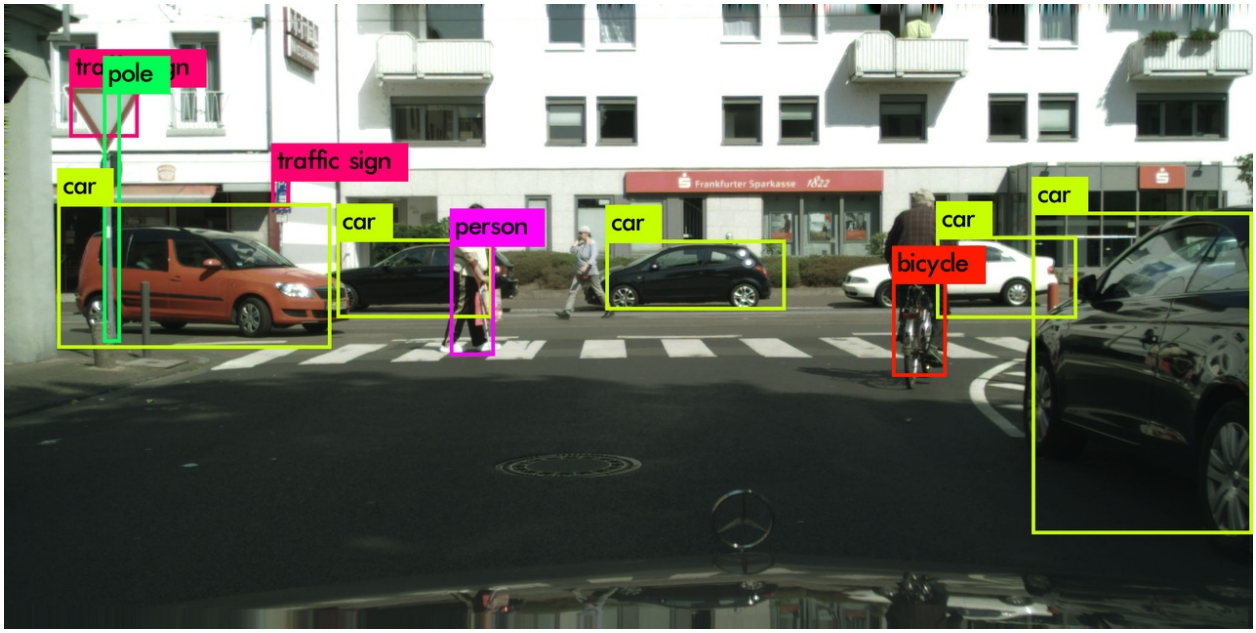

Рис. 6: Обнаружение yolov3-tiny

Заключение

В ходе выполнения данной выпускной квалификационной работы были достигнуты следующие результаты:

- проведен обзор фреймворка Darknet и системы обнаружения YOLO;
- рассмотрены и сравнены модели Yolov3 и Yolov3-tiny;
- адаптирован набор данных cityscapes для работы с YOLO;

- произведена оптимизация обучения на сетях Yolov3-tiny и квантизованной Yolov3-tiny;
- внесены изменения в исходный код проекта Darknet, связанные с получением информации обучения;
- получены данные для оценки возможности реализации модели на современных чипах.

Решения по оптимизации уменьшили значение функции потери в 2-3 раза, увеличили точность mAP на 8-14% и уменьшили размер весов на 10 МБ. Полученные данные могут быть использованы для оценки внедрения в системы на чипах для автомобильной электроники.

Список литературы

[1] CMake. — URL: https://cmake.org/.

[2] The Cityscapes Dataset for Semantic Urban Scene Understanding / Marius Cordts, Mohamed Omran, Sebastian Ramos et al. — arXiv, 2016.

[3] Dozat Timothy. Incorporating Nesterov Momentum into Adam.

[4] Focal Loss for Dense Object Detection / Tsung-Yi Lin, Priya Goyal, Ross Girshick et al. — arXiv, 2017.

[5] Going Deeper with Convolutions / Christian Szegedy, Wei Liu, Yangqing Jia et al. — arXiv, 2014.

[6] Goodfellow Ian, Bengio Yoshua, Courville Aaron. Deep Learning (Adaptive Computation and Machine Learning series). — The MIT Press, 2016.

[7] Kingma Diederik P., Ba Jimmy. Adam: A Method for Stochastic Optimization. — arXiv, 2014.

[8] Krizhevsky Alex, Sutskever Ilya, Hinton Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks.

[9] Manning Christopher D., Raghavan Prabhakar, Schütze Hinrich. Chapter 8: Evaluation in information retrieval. — Introduction to Information Retrieval, 2009.

[10] Microsoft COCO: Common Objects in Context / Tsung-Yi Lin, Michael Maire, Serge Belongie et al. — arXiv, 2014.

[11] NVIDIA. CUDNN. — URL: https://developer.nvidia.com/cudnn.

[12] Newtonsoft. JSON.Net. — URL: https://www.newtonsoft.com/json.

[13] OpenCV. — URL: https://opencv.org/.

[14] RENESAS. V3H System-on-Chip. — URL: https://www.renesas.com/eu/en/solutions/automotive/soc/r-car-v3h.html.

[15] Redmon Joseph, Farhadi Ali. YOLOv3: An Incremental Improvement. — arXiv, 2018.

[16] SSD: Single Shot MultiBox Detector / Wei Liu, Dragomir Anguelov, Dumitru Erhan et al. — arXiv, 2015.

[17] Scalable Parallel Programming with CUDA / John Nickolls, Ian Buck, Michael Garland, Kevin Skadron. — Queue - GPU Computing, 2008.

[18] XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks / Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, Ali Farhadi. — arXiv, 2016.

[19] pjreddie. Darknet. — URL: https://github.com/pjreddie/darknet.

[20] Алексей Бочковский. Darknet. — URL: https://github.com/AlexeyAB/darknet.