САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ И МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Григорьев Артемий Сергеевич

Магистерская диссертация

Сегментация областей интереса на спутниковых изображениях и их локализация для ГИС

Направление 02.04.02

«Фундаментальная информатика и информационные технологии»

Магистерская программа «Вычислительные технологии»

Научный руководитель,
кандидат тех. наук,
доцент
Гришкин В.М.

Санкт-Петербург

2017

Содержание

Введение

В течении последних нескольких десятилетий развился и продолжает очень активно формироваться особый класс автоматизированных комплексов, а именно компьютерные геоинформационные системы (сокращенно ГИС). Эти системы накапливают, обрабатывают, анализируют, обновляют и непосредственно предоставляют пользователям пространственно-координированную информацию об интересующих их объектах, которые расположены на Земле.

Главной формой представления сведений в ГИС являются цифровые карты. Они основаны на различных информационных источников (классические картографические материалы, геодезических измерений и т.д.) Самым важным источником данных для геоинформационных систем являются цифровые изображения, которые получаются в результате космических съемок и аэрофотосъемок земной поверхности. Обработку этих снимков в ГИС можно сформулировать в виде задачи компрессии данных, т.е. преобразование большого массива данных в его некоторое кодированное представление, например, векторной цифровой карты, которая уже имеет намного меньший объем и в свою очередь сохраняет все сведения о зондировании земной поверхности, необходимые для последующего использования в различных прикладных задачах.

Условно процесс обработки аэрокосмических изображений для геоинформационных систем можно разделить на следующие четыре этапа [1]:

- бортовая обработка;
- наземная предварительная обработка;
- тематическая обработка;
- векторизация.

Рассмотрим подробнее третий этап – тематическая обработка. На этой стадии происходит решение задач обработки и анализа изображения, которые решаются в интересах отдельных групп потребителей информации, например,

специалистов в области градостроительства, природопользования, геологии и т.д. Основная цель тематической обработки – формирование слоев картографической информации, которые интересуют определенную группу пользователей в базе данных ГИС.

Одной из практических задач обработки спутниковых снимков и аэрофотоснимков является поиск объектов неприродного происхождения (здания, дороги, автомобили, самолеты, корабли и т.д.). Цель этой задачи – формирование слоев ГИС, которые характеризуют транспортные коммуникации и застройку, при стратегическом планировании городов, а также оценке антропогенных рисков территорий [2]. Обычно для решения задачи поиска неприродных объектов используются аэрофотоснимки среднего и высокого разрешения. Например, для распознавания зданий необходимо разрешение до 6 см, а для поиска дорог можно обойтись снимками с разрешением 20 см [3]. Для обнаружения зданий используются методы, которые сочетают выделение объектов с их двухмерной или трехмерной реконструкцией на основе некоторых моделей [4][5][6][7]. Так как космические средства дистанционного зондирования земной поверхности активно развиваются, то для этих целей стало привлекательным использование спутниковых изображений. Главными преимуществами таких изображений являются большое покрытие и отсутствие проблем с геопривязкой.

В дальнейшем в качестве объекта неприродного происхождения будем рассматривать автотранспортные средства. Проблема локализации автомобилей на спутниковых изображениях очень актуальна на сегодняшний день и привлекло значительное внимание во всем мире в последние годы [8][9][10][11][12].

Обнаружение транспортных средств играет важную роль в широком спектре приложений и имеет множество вариантов использования в коммерческой безопасности, национальной безопасности, а также и в гуманитарных сферах.

В коммерческой безопасности, например, некоторые компании попытались сделать выводы по объему розничного трафика [13] в зависимости от плотности автомобилей на парковке, а отслеживание грузового автотранспорта в режиме реального времени является одной из долгоиграющих целей аналитики спутниковых изображений. Так как спутниковые данные дешевы и достаточно многочисленны, то они могут быть более экономически выгодными, чем встраивание датчиков в дорогу.

В военной области очень ценной была бы программа, которая могла бы обнаруживать наращивание военной техники в нестабильных регионах (обнаружение конвоев транспортных средств, которые хотят пересечь границу).

В гуманитарной сфере было бы полезно, например, определять оптимальные маршруты для оказания медицинской помощи или ликвидации последствий во время стихийных бедствий в малоизвестных регионах на основе наблюдений за движением местных транспортных средств.

Автоматическое обнаружение автотранспортных средств на спутниковых изображениях все еще имеет много проблем из-за их относительно небольшого размера и переменной ориентации объекта (рис.1).

Рис.1. Небольшой размер автотранспортных средств

Кроме того, обнаружение в реальном времени автомобилей на

крупномасштабных аэрофотоснимках с такими сложными объектами фона (рис.2) также увеличивает трудность данной проблемы.

Рис.2. Сложные объекты фона

В данной магистерской диссертации будут подробно рассмотрены некоторые современные методы компьютерного зрения, которые активно используются для решения данной задачи. Также, будут построены каскадные классификаторы при различных параметрах на основе принципов метода Виолы-Джонса, который хорошо работает для детектирования лиц людей в реальном времени, но уже в рамках задачи поиска автомобилей на спутниковых изображениях и аэрофотоснимков. Затем будет проведено исследование по ним и выбор лучшего.

Постановка задачи

В данной магистерской диссертации стоит проблема локализации автомобилей на спутниковых изображениях и аэрофотоснимках. Для ее решения необходимо:

- рассмотреть современные алгоритмы компьютерного зрения, а также их применение в рамках поставленной проблемы;
- построить каскадный классификатор на основе принципов метода Виолы-Джонса;
- показатель полноты классификации должен превышать 80%;
- показатель точности классификации должен превышать 80%;
- исследовать данный подход при различных параметрах.

Глава 1. Обзор литературы

В соответствии с существовавшими методами обнаружения объектов, задача локализации автомобилей может быть разделена на три основные части: поиск областей с данными объектами, выделение признаков и классификация.

1.1. Поиск областей с объектом

Для поиска областей, большинство существующих методов обнаружения транспортных средств использует алгоритм поиска с подвижным (“скользящим”) окном [9][10][14][15]. В данных методах используется либо подвижное окно (рис.3), которое само меняет свой размер после полного прохода по всему изучаемому изображению, либо используется окно фиксированного размера, но уже изменяется разрешение самого изображения от наименьшего к наибольшему, т.е. получается так называемая пирамида изображений (рис.4)

Рис.3. “Скользящее” окно

Это делается потому, что нужный нам объект может иметь различный масштаб на изображении (например, поиск лиц). Таким образом, методы, которые базируются на поиске с подвижным окном имеют высокие затраты по времени из-за генерирования большого количество окон разного масштаба. Но так как

Рис.4. Пирамида изображений

для нашей задачи используются спутниковые изображения, которые имеют одинаковый уровень разрешения, и автомобили на них не сильно отличаются по масштабу друг от друга, то можно задать, например, определенные промежутки изменения размера “скользящего” окна и намного сократить время поиска, не навредив точности распознавания.

По сравнению с методами, основанными на подвижном окне, методы, базирующиеся на region-proposal, предлагают сократить вычислительные затраты. Они основаны на объединение некоторых сегментов изображений, в котором, вероятнее всего, будет включен нужный нам объект. К таким относятся, например, супер-пиксели [16], заметность (saliency) [17], селективный поиск [18] и т.д. Тем не менее, они по-прежнему занимают много времени для поиска областей с распознаваемым объектом. В последнее время методы, разработанные на основе CNN (сверточных нейронных сетях), были широко использованы для этой цели, например, DeepBox [19] и RPN (Region Proposal Networks) [20]. Эти методы могут искать нужные нам регионы с особенностями глубинного обучения, что приводит к многообещающей производительности и скорости поиска.

1.2. Виды дескрипторов и их получение

Перейдем к следующей части – извлечение признаков. Для обнаружения транспортного средства широко используются вручную построенные признаки. Шао и его соавторы в своей работе [14] использовали признаки Хаара и локальные бинарные шаблоны (LBP) для обнаружения транспортного средства. Морандуззо в своей статье [10] использовал дескрипторы SIFT (Scale Invariant Feature Transform). Клакнер [15] и Тюрмер [21] в качестве признаков выбрали гистограмму ориентированных градиентов (HOG), в то время как Лью [9] использовал быстрый бинарный детектор с использованием ICF. Тем не менее, эти построенные вручную признаки недостаточно хорошо отделяют машины от фона в сложных регионах.

В последние годы алгоритмы для обнаружения регионов с искомым объектом на основе CNN достигли больших успехов в задачах с природными изображениями, благодаря своему мощному представлению в пространстве признаков. Наиболее популярным являются следующие нейронные сети: R-CNN (Region-based Convolutional Neural Networks) [22], улучшенная модификация прошлой сети Fast R-CNN [23], Faster R-CNN [20]. Все они достигают самой современной производительности, так как задействуют не только CPU, но и GPU процессоры, которые поддерживают технологию CUDA и фреймворк Caffe.

1.3. Классификация

И наконец последняя часть – классификация. Шао [14] и Морандуззо [10] использовали вручную построенные дескрипторы в методе опорных векторов (SVM) для поиска объектов на изображении. Однако в последнее время алгоритм AdaBoost постепенно замещается SVM из-за его хорошей производительности. Клакнер [15], Тюрмер [21] и Лью [9] использовали как раз таки AdaBoost классификатор вместо SVM для распознавания объекта. Их исследования показывают, что стратегия бустинга, содержащая жесткие негативные примеры и повторно взвешенные примеры для итеративного обучения, значительно улучшают качество классификатора за счет сокращения числа ложных классификаций.

1.4. Итоги обзора

Принимая во внимание все три части по детектированию объекта на изображении можно сделать следующие выводы.

Методы обнаружения на основе CNN показывают прекрасную производительность. R-CNN использует алгоритм селективного поиска для поиска регионов, содержащий искомый объект, а затем извлекает глубокие

дескрипторы для дальнейшей классификации с помощью SVM. Для того чтобы увеличить скорость и точность обнаружения, Fast R-CNN использует многозадачную сеть для классификации и bounding box regression, комбинируя получение признаков и классификацию в один процесс. Тем не менее, их алгоритм для поиска области с нужным объектом все еще занимает много времени. Поэтому, нейронная сеть Faster R-CNN уже использует технологию RPN и классификатор на основе CNN для дальнейшего поиска регионов-кандидатов. Faster R-CNN – это комплексный метод, обеспечивающий обнаружение объекта в режиме реального времени с современной производительностью. Тем не менее, для задачи поиска автомобилей на спутниковых изображениях он имеет некоторые недостатки. Алгоритм RPN не подходит для небольших объектов, из-за грубой карты признаков, которую он использует. Также, не стоит оставлять без внимания методы, которые используют принцип “скользящего” окна, а в качестве классификатора могут выступать SVM или AdaBoost.

Глава 2. Современные программные комплексы и методы для поиска объектов на изображении

В данной главе будут подробно рассмотрены два современных подхода к поиску объектов на изображении: DetectNet и YOLO (You only look once). По последнему будет приведен отчет по применению данной нейронной сети к задаче поиска автомобилей на спутниковых изображениях. Так же будет рассмотрено веб-приложение DIGITS разработанное компанией NVidia для тренировки моделей глубокого обучения.

2.1. DIGITS и сеть DetectNet

DIGITS – это полностью интерактивный инструмент, разработанный компанией NVidia для решения различных общих задач в области глубокого обучения. С его помощью исследователи-аналитики могут:

- подготавливать данные,
- определять сверточные сети,
- параллельно обучать несколько моделей,
- наблюдать за процессом обучения в режиме реального времени,
- выбирать лучшую модель.

При этом данное веб-приложение полностью избавляет пользователя от программирования и отладки, и можно сделать основной упор именно на обучение различных моделей сети и поиск лучшей для конкретной задачи.

В новой версии DIGITS 4 реализован новый подход к задаче обнаружения объектов на изображении. Он позволяет обучать нейронные сети для поиска объектов и определять ограничивающие прямоуголники вокруг них [24]. Одним из них является новая сетевая архитектура DetectNet, разработанная инженерами Nvidia. На рис.5 продемонстрирован результат работы данной сети для поиска транспортных средств на аэрофотоснимках.

Рис.5. Применение сети DetectNet для поиска транспортных средств

2.1.1 Сеть DetectNet: формат данных

Рассмотрим подробнее, что использует сеть DetectNet в качестве входных данных для задачи классификации изображений.

С одной стороны, для тренировки обучающей выборки можно использовать изображения небольшого размера, которые содержат один объект, и классовые метки (целочисленный идентификатор класса). С другой стороны, для поиска объектов на изображении необходимо больше информации для обучения. Поэтому для сети DetectNet уже используются изображения большего размера, содержащие не один, а несколько объектов, для каждого из которых классовая метка дополняется информацией о расположении углов ограничивающего их прямоугольника. Так как в этой ситуации количество объектов на изображениях обучающей выборки могут варьироваться, то может быть затруднено определение функции потерь.

Для решения этой проблемы сеть DetectNet использует фиксированный трехмерный формат метки. Благодаря этому, можно использовать изображения с различным размером и любым числом присутствующих на них объектов.

Рассмотрим схему обработки (рис.6) для обучения сети DetectNet изображений с разметкой из обучающей выборки.

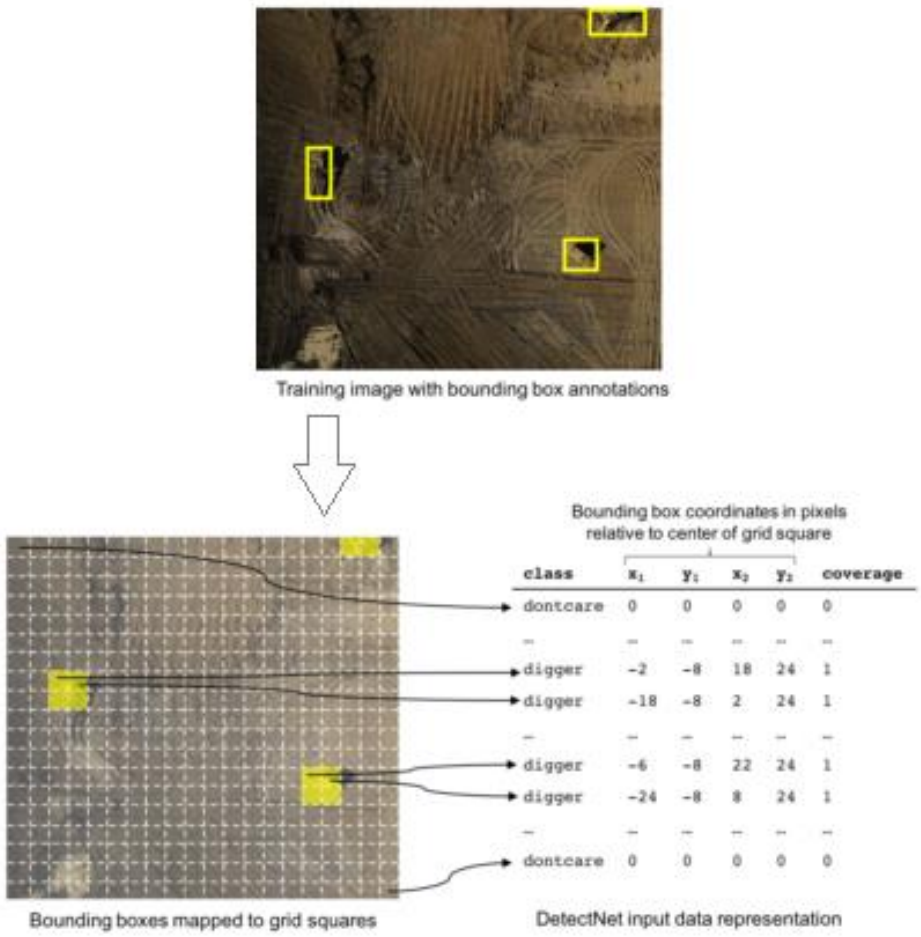

Рис.6. Схема представления входных данных DetectNet

На первом этапе, на исходное изображение накладывается сетка, которая имеет фиксированный размер клеток. Этот размер выбирается так, чтобы он был немного меньше самого маленького объекта, который нужно распознать.

На втором этапе, каждая ячейка сетки дополняется следующей информацией:

- класс объекта, который находится в ячейках сетки,
- координаты пикселя углов ограничивающего прямоугольника.

Если ни один распознаваемы объект не попал в ячейку сетки, то используется особый класс "dontcare", чтобы соблюдался фиксированный формат данных.

Также в данный формат данных добавляется еще одно значение, которое называется "coverage". Оно может принимать два значения: «0» или «1». "Coverage" нужно для того, чтобы указать, есть ли в ячейки сетки объект. Для случая, когда в одну ячейку сетки попадают несколько объектов, сеть DetectNet выбирает объект, занимающий наибольшее количество пикселей. Но бывают и случаи, когда два объекта имеют одинаковое количество пикселей. Тогда выбирается тот, который имеет наименьшую ординату (OY) ограничивающего прямоугольника. Для задачи поиска транспортных средств на спутниковых изображениях и аэрофотоснимке такой выбор объектов как правило не принципиален.

В итоге можно сделать вывод, что основной целью обучения сети DetectNet является предсказание для каждой ячейки сетки о наличие объекта в ней и вычислении относительных координат углов прямоугольника, ограничивающего данный объект.

2.1.2 Сеть DetectNet: архитектура

В данном параграфе будет рассмотрена архитектура сети DetectNet, которую можно использовать в DIGITS.

Всего нейронная сеть DetectNet имеет 5 частей, которые определены в файле модели сети фреймворка Caffe.

Рассмотрим два этапа: обучение и проверка.

На рис. 7 показана схема архитектуры сети на первом этапе.

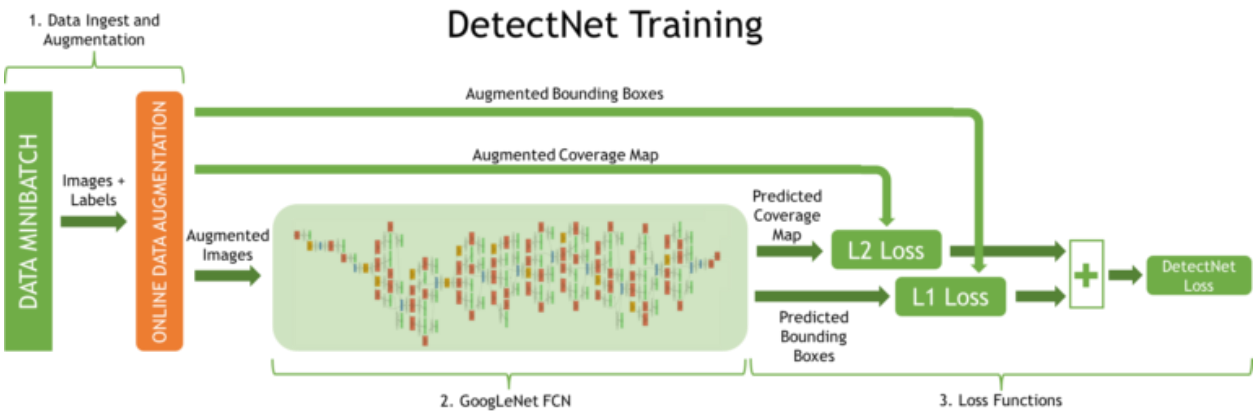

Рис.7. Архитектура DetectNet для обучения

На данной схеме можно выделить три основных процесса:

1. Изображения и метки объектов из обучающей выборки поступают на вход слоя данных. Затем, преобразующий слой “на лету” дополняет данные.
2. Полностью сверточная нейронная сеть (FCN) извлекает признаки, а также предсказывает классы объектов и прямоугольники, ограничивающие их, по ячейкам сетки.
3. Функции потерь, параллельно, считают ошибку по задачам предсказания покрытия объекта и углов прямоугольника, ограничивающего его, по ячейкам сетки.

На рис. 8 показана схема архитектуры сети на втором этапе.

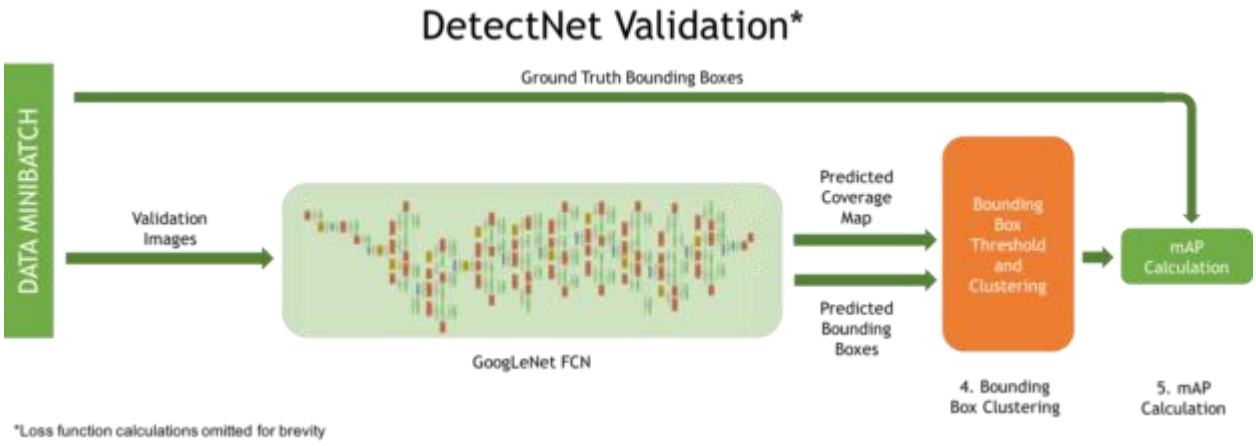

Рис.8. Архитектура DetectNet для проверки

На данном этапе появляются два дополнительных процесса:

1. Кластеризация прямоугольников, которые ограничивают искомые объекты, для получения окончательного набора.
2. Подсчет метрики mAP (mean Average Precision). Она нужна для измерения эффективности построенной модели по всей тестовой выборке.

Размер ячейки сетки для обучающих меток можно изменить, задавая шаг для слоя `detectnet_groundtruth_param`:

```
detectnet_groundtruth_param {
    stride: 16
    scale_cvg: 0.4
    gridbox_type: GRIDBOX_MIN
    min_cvg_len: 20
    coverage_type: RECTANGULAR
    image_size_x: 1024
    image_size_y: 512
    obj_norm: true
    crop_bboxes: false
  }
```

В параметрах данного слоя также можно указать и размер изображений, использующихся для обучения (`image_size_x, image_size_y`). Если указать данные параметры, то все изображения, которые будут попадать на вход сети DetectNet, будут случайным образом обрезаться по заданным размерам. Это полезно в том случае, если обучающая выборка состоит из очень больших по размеру изображений, а объекты на них, которые нужно распознать, очень маленькие.

Параметры слоя, который дополняет входные данные «на лету», находятся в `detectnet_augmentation_param`:

```
detectnet_augmentation_param {
    crop_prob: 1.0
    shift_x: 32
    shift_y: 32
    scale_prob: 0.4
    scale_min: 0.8
    scale_max: 1.2
    flip_prob: 0.5
    rotation_prob: 0.0
    max_rotate_degree: 5.0
    hue_rotation_prob: 0.8
    hue_rotation: 30.0
    desaturation_prob: 0.8
    desaturation_max: 0.8
  }
```

Данная процедура играет очень важную роль во время обучения сети DetectNet. Благодаря ей полученный детектор объектов становится высокочувствительным и точным. Параметры слоя, пример которого показан выше, определяют различные случайные преобразования над обучающей выборкой, такие как смещение, отражение и т.п. Данные преобразования данных приводят к тому, что DetectNet не обрабатывает два раза одно и тоже изображение.

Структура подсети FCN, которая используется в DetectNet, основана на сети GoogLeNet без слоя входных данных, заключительного слоя и выходных слоев [25]. Благодаря данному подходу снижается время обучение сети DetectNet и улучшается ее точность. FCN – это полная сверточная нейронная сеть, слои которой связаны не полностью. Следовательно, можно принимать на вход изображения с любым размером и считать отклик, используя технику «скользящего» окна с шагом. В результате, GoogLeNet без заключительного слоя – нейронная сеть с подвижным окном размера 555x555 px и шагом в 16 px.

Для создания окончательной функции потерь и оптимизации сеть DetectNet использует линейную комбинацию двух независимых друг от друга функции потерь:

$$\frac{1}{2N}\sum_{i=1}^{N}\left|coverage_i^t - coverage_i^p\right|^2$$

$$\frac{1}{2N}\sum_{i=1}^{N}\left[\left|x_1^t - x_1^p\right| + \left|y_1^t - y_1^p\right| + \left|x_2^t - x_2^p\right| + \left|y_2^t - y_2^p\right|\right]$$

В первой функции вычисляется квадратичная ошибка по всем ячейкам сетки исходных данных между настоящим и предсказанным покрытием объекта. Во второй находится средняя ошибка между настоящими и предсказанными углами прямоугольника, ограничивающего искомый объект, по всем ячейкам сетки.

2.1.3 Сеть DetectNet: выходные данные

В последних слоях сети DetectNet происходят следующие два процесса: фильтрация и кластеризация набора сгенерированных прямоугольников, ограничивающие объект, для ячейки сетки. Фильтрация осуществляется при помощи порогового метода по значению предсказанного покрытия объекта. Кластеризация производится с использованием критерия эквивалентности прямоугольников. С помощью данного критерия происходит объединение фигур подобного местоположения и размера. За сходство прямоугольников, отвечает переменная `eps`. Если ее значение равно нулю, то прямоугольники объединяются, а если стремится к бесконечности, то все прямоугольники попадают в один кластер. После объединения происходит пороговая фильтрация по параметру `gridbox_rect_thresh`, которая отделяет малые кластеры, а по остальным рассчитываются средние прямоугольники, которые и попадают в выходные данные. Данный метод кластеризации реализован с помощью Python в фреймворке Caffe через интерфейс «Python Layers».

В сети DetectNet для вычисления и вывода метрики mAP, также используется интерфейс «Python Layers». Для предсказанного прямоугольника, который ограничивает искомый объект, и настоящего прямоугольника считается параметр IoU (отношение площади пересечения этих прямоугольников к их сумме). Этот параметр используется для разделения прямоугольников на TP (True Positive) или FP (False Positive) и FN (False Negative). В итоге, метрика mAP в данной сети рассчитывается как произведение точности (precision) и меры полноты (recall), где

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

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

Данная метрика является удобной характеристикой чувствительности к отбрасыванию ложных результатов, обнаружению объектов обучающей выборке, так и к точности найденных прямоугольников, которые

ограничивают искомый объект. [26]

2.1.4 Эффективность обучения

Сеть DetectNet является достаточно эффективной и точной благодаря тому, что использует FCN.

Обучение сети DetectNet в DIGITS 4 на выборке из 307 обучающих и 24 тестовых изображений (1536x1024px) с Nvidia Caffe 0.15.7 и cuDNN RC 5.1, занимает около 63 минуты, используя одну видеокарту Titan X.

Время распознавания объектов данной сетью занимает 41 мс (примерно 24 fps) на изображениях размером 1536x1024px (размер сетки 16 px, одна видеокарта TitanX, Nvidia Caffe 0.15.7, cuDNN RC 5.1).

2.2. Сеть YOLO

2.2.1 Особенности и архитектура

Еще одной эффективной и достаточной быстрой нейронной сетью является сеть YOLO (You Only Look Once), разработанная Джозефом Редмоном и его командой. Данная сеть наиболее проста, в отличии от Faster R-CNN. Так как сеть DetectNet, которая была детальна рассмотрена в прошлом параграфе, базируется на сети YOLO, то и архитектуры у них схожие, поэтому еще раз рассматривать подробно архитектуру сети YOLO в данной работе мы не будем, а сделаем в этом параграфе акцент на ее применении в задаче поиска автотранспортных средств на спутниковых изображениях, которая была рассмотрена в [27]. Наиболее подробно о сети YOLO и ее новой версии можно прочитать в статьях [28][29].

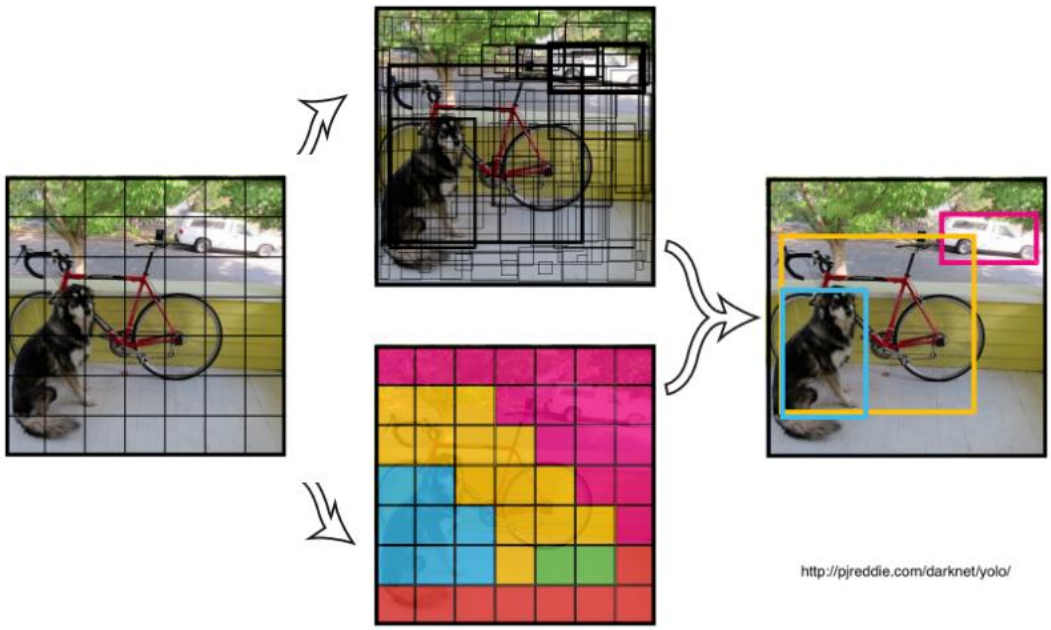

Рис.9. Применение сети YOLO для распознавания объектов на изображении

Рассмотрим вкратце как работает данная сеть (рис. 9). На изображение накладывается сетка (в данном случае 7x7), которая делит его на несколько ячеек, в каждой ячейке сетки отдельно применяется классификатор, и после этого делается предсказание о том, где находится центр искомого объекта и где находится прямоугольник, ограничивающий его. Работает сеть YOLO намного быстрее, чем R-CNN, Fast R-CNN и Faster R-CNN, при этом точность почти не падает [30].

2.2.2 Применение сети YOLO для поиска автомобилей на спутниковых снимках

Для обучения и тестирования нейронной сети YOLO в рамках поставленной задачи Адам в своей статье [27] использовал открытый набор данных COWC (Cars Overhead With Context) [31].

Обучающая выборка состояла из 2418 изображений размером 416x416 пикселей (рис.10). Всего на этих изображениях было 13303 уникальных автомобилей.

Рис.10. Примеры изображений из обучающей выборки.

Обучение было произведено для 1600 эпох (полный проход через все обучающие данные), что заняло 4 дня на одной видеокарте NVIDIA Titan X GPU.

Для тестирования использовались 23 изображения размером 4000x4000 пикселей из набора данных штата Юта. Всего в общей сложности данные изображения содержали 25980 автомобилей. В результате обработки получились следующие итоги (на рис.11 и рис.12 приведены 2 достаточно показательных примера из 23-ех [32]). В качестве меры оценки качества распознавания была выбрана мера F_1, которая вычисляется следующим способом:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

По этим двум примерам видно, что сеть YOLO достаточно хорошо F_1=0,95 справилась для изображения (рис.11) с городской местностью и где автомобили располагались не сильно плотно друг другу. Но для изображения (рис.12) с загородной местностью, на котором автомобили расположены очень плотно друг к другу, результаты распознавания были намного хуже F_1=0,67.

Рис.11. Пример 1 из тестовой выборки [32].

Рис.12. Пример 2 из тестовой выборки [32].

Глава 3. Метод Виолы-Джонса

В прошлой главе были подробно рассмотрены две нейронные сети для поиска объектов на изображении, последняя из которых показала достаточно хороший результат в рамках задачи поиска автотранспортных средств на спутниковых изображениях. В данной главе будет детально рассмотрен метод разработанный Виолой и Джонсом [33][34] в 2001 году, которой очень хорошо показал себя в задаче поиска лица и тела человека на изображениях и в видеопотоке в реальном времени, на основе которого в дальнейшем будет построен каскадный классификатор для задачи, поставленной в этой работе.

3.1. Основные принципы

В алгоритме Виолы-Джонса можно выделить четыре основных этапа:

1. Перевод изображения в интегральное представление.
2. Использование признаков Хаара в качестве дескрипторов.
3. Использование алгоритма AdaBoost в качестве классификатора.
4. Использование каскада из классификаторов.

Обучение данных классификаторов происходит очень медленно, но детектируются искомые объекты очень быстро. Данный метод работает на принципе «скользящего» окна.

Рассмотри каждый из этих этапов подробнее.

3.2. Интегральное представление изображения

Для того, что сократить количество операций для вычисления признаков Хаара, исходное изображение переводится в свое интегральное представление. Данное представление позволяет довольно быстро рассчитать суммарную яркость прямоугольника на изображении независимо от того, какого размера будет данный прямоугольник. Интегральное представление

изображения – это матрица, размер которой равен исходному изображению, а значение ее элементов рассчитывается следующим способом:

$$L(x,y) = \sum_{i=0,j=0}^{i\le x, j\le y} I(i,j),$$

где $I(i,j)$ – это яркость пикселя изначального изображения.

Каждый элемент матрицы $L(x,y)$ – есть сумма пикселей в прямоугольнике (0,0)(0,x)(x,y)(0,y). Расчет такой матрицы занимает линейное время, которое пропорционально количеству пикселей в данном изображении. Поэтому интегральное представление рассчитывается полностью за один проход.

Продемонстрируем наглядно вычисление площади прямоугольника в интегральном изображении на следующем примере (рис.13).

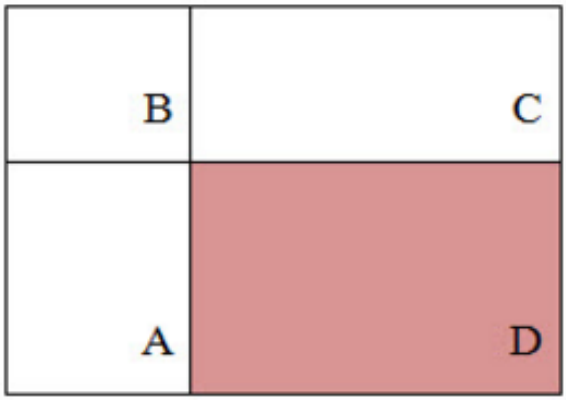

Рис.13. Нахождение площади прямоугольника ABCD в интегральном представлении изображения.

Пусть в интегральном изображении нужно найти площадь прямоугольника с вершинами ABCD. Данное вычисление можно свести всего к трем операциям:

$$S(ABCD) = L(D) - L(A) - L(C) + L(B)$$

3.3. Признаки Хаара

Исторически сложилось, что алгоритмы, которые работают только с интенсивностью изображения, имеют очень большую вычислительную сложность. Папагеоргиу в свое работе [35] рассмотрел множество признаков, основанные на вейвлетах Хаара. Виола и Джонс адаптировали данную идею и разработали признаки Хаара.

В классическом методе Виолы-Джонса используются прямоугольные признаки Хаара, которые выглядят следующим образом рис.14.

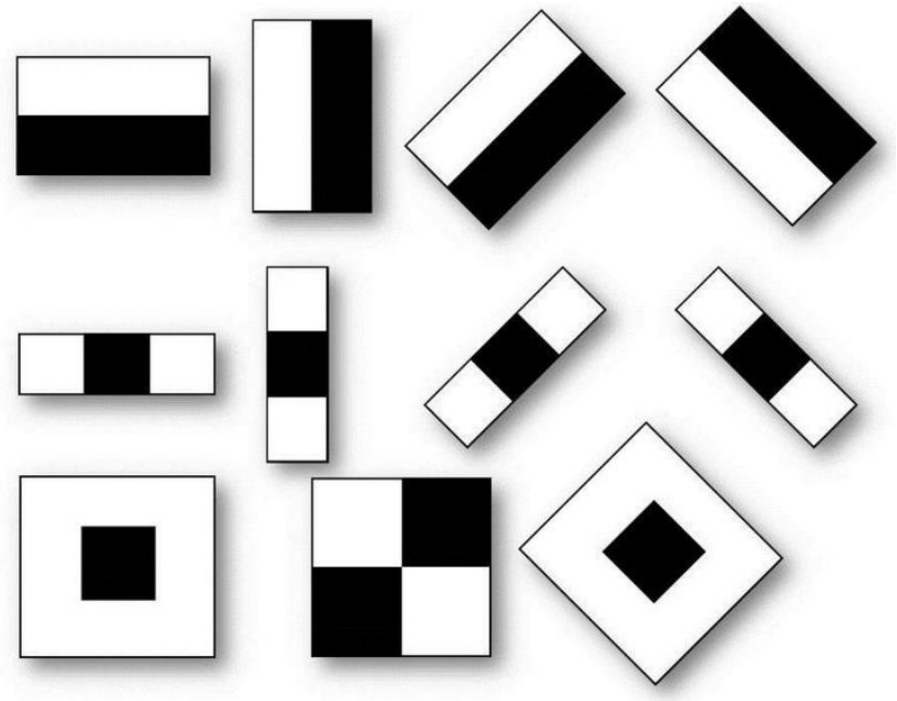

Рис.14.Прямоугольные признаки Хаара.

Данные признаки еще называются примитивами Хаара.

В расширенном методе Виолы-Джонса, эти признаки дополнились еще несколькими модификациями рис.15.

Вычисляются данные признаки следующим путем:

$$F = X - Y$$

где X – это сумма значений яркостей пикселей изображения закрываемых белой областью, а Y – черной областью. Собственно, для этого и используется интегральное представление изображения, рассмотренное выше.

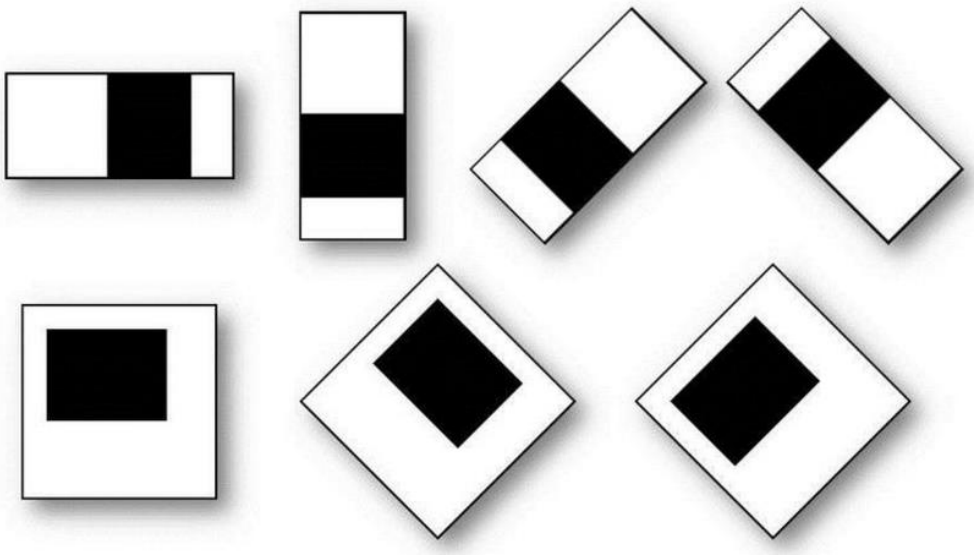

Рис.15.Дополнительные признаки Хаара.

Количество различных вариаций признаков, рассчитанное для окошка 24x24, составляет примерно 160,000+, и очень многие из них бесполезны для детектирования того или иного объекта. Чтобы выбрать полезные признаки используется алгоритм AdaBoost, который будет рассмотрен далее.

3.4. Алгоритм AdaBoost

Бустинг (от англ. boosting – повышение, усиление, улучшение) – это комплекс методов, который способствует повышению точности аналитических моделей. Модель называется «сильной», если допускает малое число ошибок классификаций. «Слабая» модель же, наоборот, делает большое количество ошибок и не позволяет надежно разделить классы. Поэтому бустинг – это такая процедура, которая в результате последовательного построения композиции алгоритмов машинного обучения стремится устранить недостатки всех предыдущих алгоритмов в каждом новом алгоритме.

Наиболее совершенным алгоритмом бустинга является AdaBoost (Adaptive Boosting), предложенный в 1999 г. Фройндом и Шапиром [36]. Алгоритм AdaBoost адаптивный потому, что каждый следующий

классификатор строится по тем объектам, которые были неверно классифицированы прошлыми. Данный алгоритм чувствителен к шуму в данных и выбросам, а также он менее подвержен переобучению.

Рассмотрим работу алгоритма AdaBoost на задаче построения бинарного классификатора.

Дано:

$(x_1, y_1), \cdots, (x_m, y_m)$, где $x_i \in X, y_i \in Y = \{-1, +1\}$.

Задаем начальные веса:

$D_1(i) = \frac{1}{m}, i = 1, \dots, m.$

Для всех $t = 1, \dots, T$:

- Находим классификатор:

  $h_t : X \to \{-1, +1\}$, минимизирующий взвешенную ошибку классификации: $h_t = \arg\min_{h_j \epsilon H} \varepsilon_j$,

  где $\varepsilon_j = \sum_{i=1}^{m} D_1(i)[y_i \neq h_j(x_i)]$

- Если $\varepsilon_t \geq 0.5$, то останавливаемя.
- Выбираем $a_t \in R$, который обычно равен $a_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$, где ε_t – это взвешенная ошибка классификатора h_t.
- Обновляем веса:

$$D_{t+1}(i) = \frac{D_t(i)e^{-a_t y_i h_t(x_i)}}{Z_t}$$

  где Z_t – это нормализующий параметр, который выбирается так, чтобы $\sum_{i=1}^{m} D_1(i) = 1$

Строим результирующий «сильный» классификатор:

$$H(x) = sign\left(\sum_{t=1}^{T} a_t h_t(x)\right)$$

Плюсы AdaBoost:

- Довольно простая реализация.

- Очень хорошая обобщающая способность, которая может улучшаться по мере увеличения числа базовых алгоритмов.
- Возможность обнаружить объекты, которые являются шумовыми выбросами.
- Собственные накладные расходы данного алгоритма невелики.

Минусы AdaBoost:

- Требует довольно длинные обучающие выборки.
- Иногда строятся неоптимальные наборы базовых алгоритмов. Для этого можно временами возвращаться к раннее построенным алгоритмам и обучать их заново.
- Иногда случается переобучение при наличии довольно большого уровня шума в данных, т.е. экспоненциальная функция, которая отвечает за потери, слишком сильно увеличивает веса «самых трудных» объектов, на которых ошибаются многие «слабые» алгоритмы. Чаще всего именно эти объекты оказываются шумовыми выбросами. В результате чего, алгоритм AdaBoost начинает настраиваться на шум, что и ведёт к переобучению. Данная проблема решается путём удаления этих шумовых выбросов или применения таких функций потерь, которые менее «агрессивны».
- Построению довольно больших и громоздких композиций, состоящих из сотен «слабых» алгоритмов. Данные композиции исключают возможность содержательной интерпретации, а также требуют больших объёмов памяти для хранения «слабых» алгоритмов и довольно больших временных затрат на вычисление классификаций.

После построения классификатора, на основе данного алгоритма, Виоле и Джонсу удалось сократить количество вычисляемых признаков для окошка 24x24 с 160,000 до 6000. Но и это довольно большое число признаков, для

детектирования объектов в реальном времени. Поэтому было решено использовать вместо одного «сильного» классификатора каскад из классификаторов.

3.5. Каскад классификаторов

Каскад из «сильных» классификаторов – это некое дерево принятия решений, в котором каждый узел построен таким образом, чтобы распознавать практически все интересующие объекты на изображении и отклонять области, в которых этих объектов нет. Кроме того, узлы данного дерева принятия решений устроены таким образом, что чем ближе узел расположен к корню дерева, тем меньшее число дескрипторов он содержит, а следовательно, требует меньше времени на принятие решения.

Данная каскадная модель классификаторов очень хорошо подходит для обработки изображений, на которых содержится сравнительно небольшое детектируемых объектов. В этом случае классификатор построенный таким образом может быстрее принять решение о том, что данная область не содержит объект, и перейти к следующей.

Каскад из классификаторов можно представить в виде следующей схемы, изображенной на рис.16. Опишем ее:

- На вход подается одна из областей обрабатываемого изображения и к ней применяется первый самый простой «сильный» классификатор из каскада.
- Если он принял решение, что искомого объекта там точно нет, то эта область отбрасывается и на вход подается следующая.
- Если первый классификатор принял решение, что скорее всего объект присутствует в данной области, то она подается на следующий классификатор, который более приспособленный и уже сложнее прошлого.
- Данный процесс продолжается до тех пор, пока область с

изображением не поступит в обработку на последний самый сложный «сильный» классификатор из каскада, который примет окончательное решение о том есть ли там объект или нет.

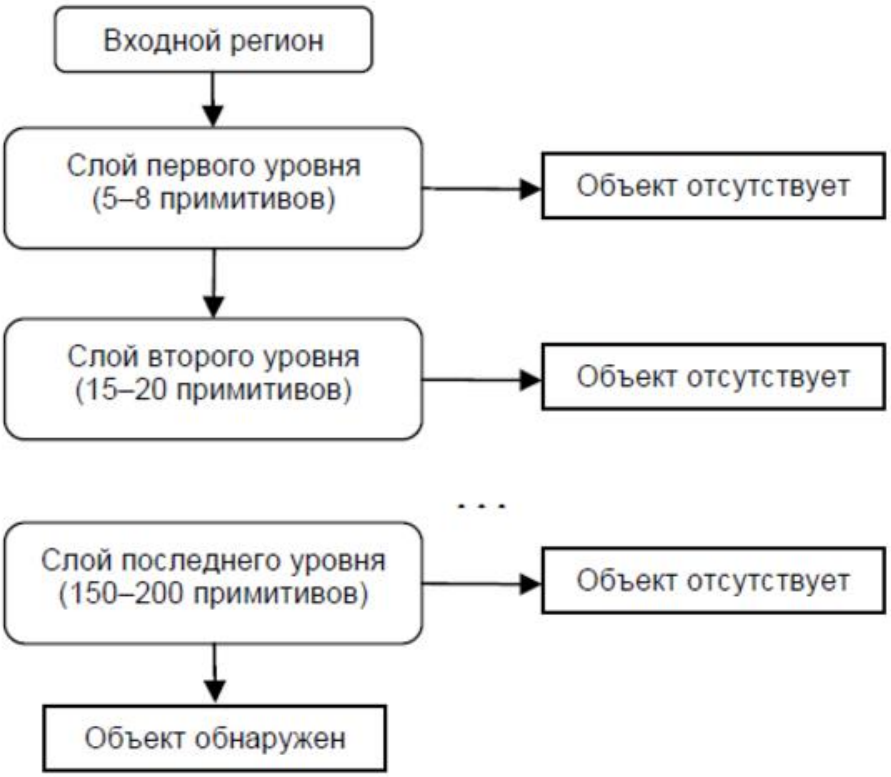

Рис.16.Пример каскадной модели «сильных» классификаторов.

Для тренировки такого каскадного классификатора потребуются следующие действия:

1. Установление значения уровня ошибок для каждого классификатора из каскада. Данные параметры называются detection и false positive rates. Нужно чтобы уровень detection был высоким, а уровень false positive rates низким.
2. Добавление дескрипторов до тех пор, пока параметры для обучаемого классификатора из каскада не достигнут поставленного уровня.
3. Если false positive rates высокий, то добавляется следующий слой.
4. Ложные обнаружение, найденные на текущем слое, используются как отрицательные на следующем слое.

В итоге благодаря данному подходу, Виоле и Джонсу удалось получить достаточно хороший и быстрый классификатор для детектирования лиц.

Глава 4. Построение каскадного классификатора для локализации автомобилей и его тестирование

В данной главе будут построены каскадные классификаторы для поиска автомобилей на спутниковых изображениях и аэрофотоснимках на основе метода Виолы-Джонса при различных параметрах, начиная от выбора размера подвижного окна и кончая выбором конкретного типа алгоритма AdaBoost. По данным классификаторам будет проведено тестирование на тестовой выборке, на основе которого будет выбран лучший относительно меры F_1. Для полученного классификатора будет проведено исследования на различных изображениях, которые были использованы в качестве спутниковых снимков, и сделан вывод о том, как он справляется с поставленной задачей.

4.1. Подготовка обучающей и тестовой выборки

В качестве набора данных из спутников снимков была выбрана открытая база COWC (Cars Overhead With Context) [31]. Эта база состоит из большого количества аннотированных автомобилей. Всего представлено шесть регионов городского типа: Торонто - Канада, Селуин - Новая Зеландия, Постдам и Файинген – Германия, Коламбус и Юта – США.

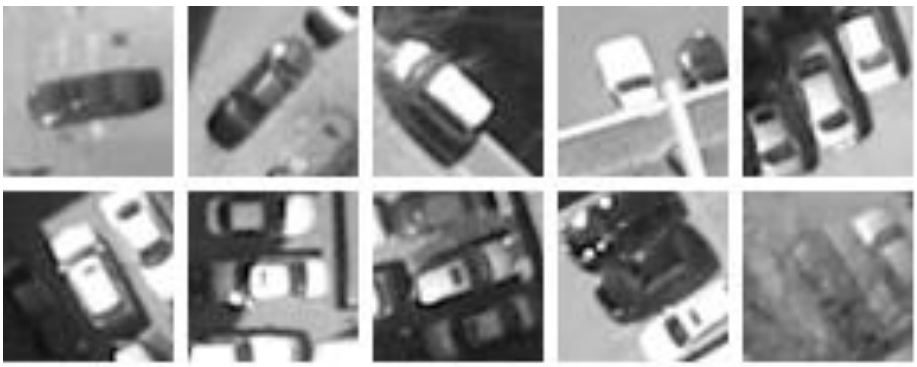

Рис.17. Примеры изображений из положительной выборки.

В качестве исследования мной был выбран набор данных принадлежащий Торонто. Все изображения в данном датасете имеют размер 256x256 пикселей с разрешением 15см на пиксель. Из этих изображений была нарезана положительная выборка, состоящая из 10000 изображений 48x48 пикселей, содержащих автомобиль (рис.17). В качестве отрицательной выборки, не содержащей автомобиль, сначала были нарезаны 20000 изображений 50x50 пикселей с достаточно сложными объектами фона (рис.18), затем 18000 изображений 190x190 пикселей с сложными объектами и простыми.

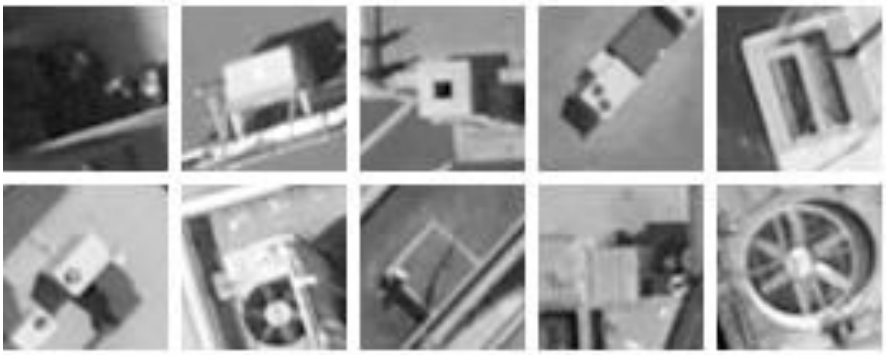

Рис.18. Примеры изображений из отрицательной выборки.

Из полученных положительной и отрицательных выборок были составлены обучающая и тестовая:

- Обучающая выборка состоит из 7000 положительных и 17000 отрицательных примеров.
- Тестовая выборка состоит из 1500 положительных и 500 отрицательных примеров.

4.2. Структура каскадного классификатора

Изначально изображение переводилось из RGB в градацию серого.

В качестве дескрипторов мной были выбраны локальные бинарные шаблоны (LBP) [37], которые, как мне кажется, должны хорошо подойти для

поставленной задачи. Данные признаки реализованы в библиотеке OpenCV. Рассмотрим подробнее, как вычисляются данные признаки.

Локальные бинарные шаблоны – это некие бинарные коды определенной разрядности, которые используются для классификации в компьютерном зрении. Данные признаки инвариантны, как и к изменениям в условиях освещения, так и к поворотам изображения.

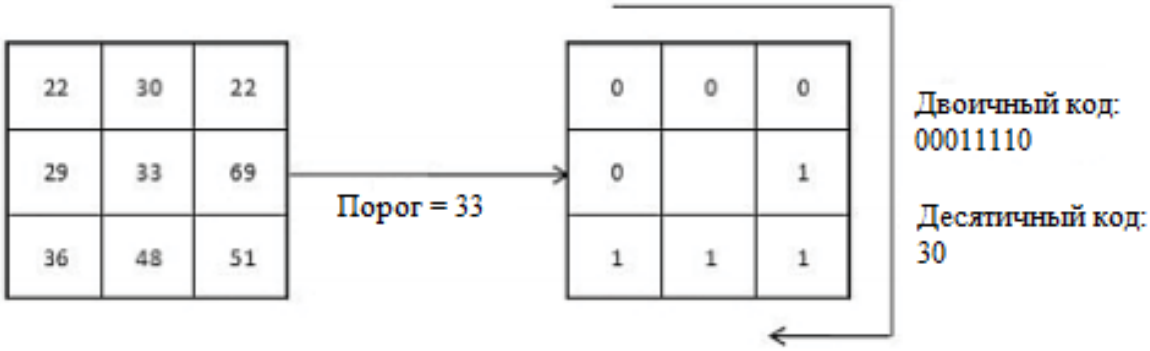

Рис.19. Базовый ЛБШ.

ЛБШ описывает окрестность пикселя изображения в двоичном представлении. Чтобы вычислить в некоторой точке изображения базовый ЛБШ используются значения интенсивности восьми пикселей окружающих данную точку, а в качестве порога принимается значение интенсивности в этой точке. Если интенсивность пикселя больше или равна порогу, то они принимают значение «1», оставшиеся «0» (рис.19). В результате чего, получается восьмиразрядный бинарный код, описывающий окрестность данной точки.

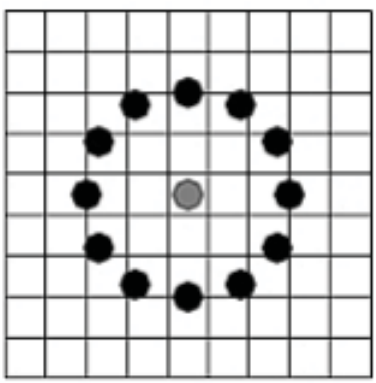

Рис.20. Расширенный ЛБШ.

Также, для более гибкого анализа текстурных особенностей изображения строятся расширенные ЛБШ (рис.20) с произвольным числом точек и радиусом, благодаря круговой окрестности и билинейной интерполяции значений интенсивности пикселей.

Некоторые виды бинарных кодов несут в

себе больше информации, чем другие. Речь идет о равномерных локальных шаблонах – шаблоны которые содержат не более трех серий «0» и «1» (например, 001110000). РЛБШ определяют важные локальные особенности изображения, примеры которых показаны на рис. 21. Также, они обеспечивают экономию памяти.

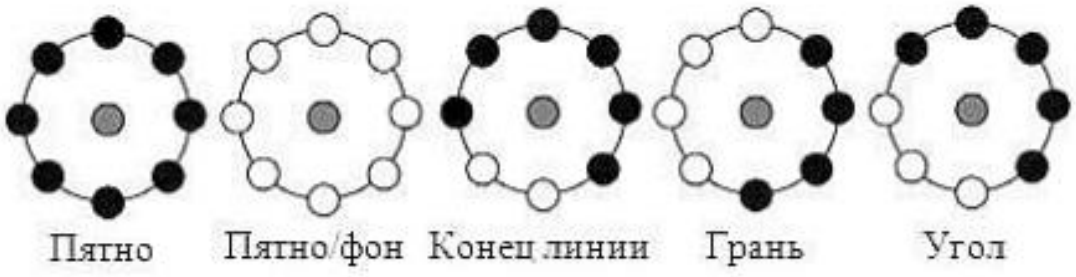

Рис.21. Локальные особенности, детектируемы РЛБШ.

ЛБШ вычисляются для всех пикселей изображения. Далее, строится гистограмма, в которой всем РЛБШ соответствует отдельный столбец. Также, учитывается и число неравномерных ЛБШ, значение которого записывается в последний столбец гистограммы.

Так как для построения данного классификатора были выбраны ЛБШ признаки, то перевод изображения в его интегральное представление не требуется.

В качестве алгоритма классификации был выбран AdaBoost, как и в методе Виола-Джонса. В дальнейшем мной было применено четыре типа этого алгоритма [38], которые также реализованы в OpenCV: дискретный AdaBoost (DAB), вещественный AdaBoost (RAB), LogitBoost (LB), гладкий AdaBoost (GAB).

Также был применен каскадный подход для построения классификатора, с целью сокращения времени распознавания объекта.

4.3. Обучение и тестирование классификаторов

Все исследования были проведены на процессоре Intel Core i7 3635QM Ivy Bridge 2400 МГц.

Для начала я провел обучение на выборке из 3000 положительных (размер подвижного окна был выбран 48x48) и 7000 жестких отрицательных (размер 50x50) примеров. В качестве бустинг-классификатора был выбран GAB (Gentle AdaBoost). Каскад состоял из 8 этапов. Но данный подход показал себя с плохой стороны, как и по скорости обучения и детектирования, так и по качеству (рис.22).

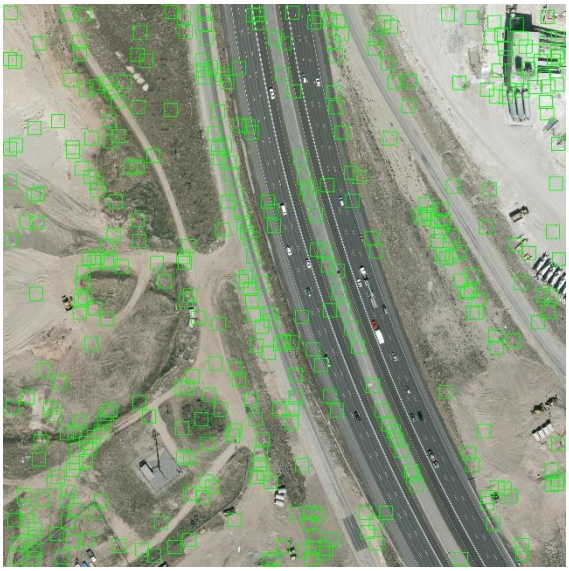

Рис.22. Пример обработанного изображения.

Детектирование автомобилей на данном снимке (размер 2048x2048 пикселей) заняло около 1 мин и было получено очень большое кол-во TN.

Поэтому в качестве отрицательной выборки далее были использованы изображения (190x190 пикселей), содержащие, как и сложные объекты фона, так и простые.

Далее на выборке из 4000 положительных и 10000 отрицательных примеров были построены 4 классификатора с разным размером «скользящего» окна: 30x30, 36x36, 40x40 и 48x48. В качестве бустинг-классификатора все они использовали GAB. Каскады у трех классификаторов состояли из 16-ти этапов, у одного из 13-ти, так как заданная точность была

достигнута раньше. Время обучения каждого классификатора занимало от 10ч до 25ч, в зависимости от выбора размера окна.

Полученные классификаторы были протестированы на тестовой выборке из 1500 положительных и 500 отрицательных примера. Результаты данного тестирования приведены в табл.1.

Номер	Тип AdaBoost	Размер окна, пиксели	Кол-во этапов в каскаде	TP	FN	FP	F_1
1	GAB	48x48	16	1413	87	83	0.95
2	GAB	40x40	16	1407	93	93	0.94
3	GAB	36x36	13	1371	129	102	0.91
4	GAB	30x30	16	1178	322	23	0.86

Табл.1. Выбор размера подвижного окна.

Лучше всего по качеству себя показал первый классификатор, но скорость детектирования была намного ниже второго, который почти не уступает по качеству первому. Результаты остальных двух классификаторов были хуже второго. Поэтому для дальнейшего исследования было выбрано подвижное окно размером 40x40 пикселей.

Следующим этапом исследования было построение классификаторов с разным типом бустинг-алгоритма, с уже выбранным ранее размером «скользящего» окна.

На той же самой обучающей выборке было обучено еще три классификатора, с другими типами AdaBoost: DAB, RAB и LB. Каскады состояли из 16 этапов. Время обучения занимало от 14ч до 40ч. Дольше всего обучался DAB.

Полученные классификаторы были протестированы на тестовой выборке. Результаты приведены в таб.2.

Номер	Тип AdaBoost	Размер окна, пиксели	Кол-во этапов в каскаде	TP	FN	FP	F_1
2	GAB	40x40	16	1407	93	93	0.94
5	DAB	40x40	16	1126	374	314	0.79
6	RAB	40x40	16	1314	186	151	0.88
7	LB	40x40	16	1152	348	297	0.78

Табл.2. Выбор типа алгоритма AdaBoost.

Лучше всего себя показал классификатор с алгоритмом типа GAB, который был выбран изначально.

Далее, был изменен параметр `max_false_alarm_rate` (максимальный желаемый уровень ложных срабатываний для каждого этапа каскада), который изначально был равен 0.5. Было обучено еще 2 классификатора, с другими значениями этого параметра, а именно 0.4 и 0.3. Результаты приведены в табл. 3.

Изменение данного параметра не привело к заметному улучшению качества распознавания.

Номер	Тип AdaBoost	Max _false _alarm _rate	Размер окна, пиксели	Кол-во этапов в каскаде	TP	FN	FP	F_1
2	GAB	0.5	40x40	16	1407	93	93	0.94
8	GAB	0.4	40x40	16	1396	104	85	0.93
9	GAB	0.3	40x40	16	1388	112	82	0.93

Табл.3. Изменение параметра `max_false_alarm_rate`.

И последним шагом было увеличение обучающей выборки.

На выборке из 7000 положительных и 17000 отрицательных примеров был построен классификатор с размером «скользящего» окна 40x40. В качестве бустинг-классификатора использовался GAB, выбранный ранее. Каскад состоял из 16-ти этапов. Время обучения этого каскада из классификаторов заняло 28ч.

Тестирование было проведено на той же тестовой выборке. Результаты приведены в табл.4.

Номер	Тип AdaBoost	Размер окна, пиксели	Кол-во этапов в каскаде	TP	FN	FP	F_1
2	GAB	40x40	16	1407	93	93	0.94
10	GAB	40x40	16	1415	85	73	0.95

Табл.4. Сравнение классификаторов, обученных на разных выборках.

Благодаря увеличению обучающей выборки мера F_1 увеличилась до значения 0.95.

Рис.23. Примеры обработанных снимков.

Рис.24. Фрагмент обработанного снимка.

Далее, в качестве примера, полученный классификатор был применен к двум изображениям с разной плотностью автомобилей (рис.23). Со своей задачей он справился довольно хорошо, при сравнительно небольшой обучающей выборки. Данный классификатор, например, смог обнаружить автомобили спрятанные в тени здания (рис.24), но не смог найти автомобиль за деревом.

Если возвратиться к снимку (рис. 22), к которому был применен первый обученный классификатор, и применить последний, то можно увидеть заметную разницу в качестве распознавания (рис.25). На первом изображении слишком много ложных срабатываний и почти отсутствуют положительные. На втором же почти все автомобили были обнаружены, но также были и незначительные ложные срабатывания из-за того, что изображения с данным типом местности не присутствовали в обучающей выборки.

Рис.25. Сравнение двух классификаторов.

4.4. Вывод

Исследовав данный подход к локализации автотранспортных средств на аэрокосмических изображениях, было выяснено, что оптимальный размер «скользящего» окна для разрешения 15см на пиксель должен составлять 40x40 пикселей. Также, было установлено, что из четырех типов AdaBoost алгоритмов, реализованных в OpenCV, лучше всего справляется с поставленной задачей GAB. Изменение параметра, который отвечает за максимальный желаемый уровень ложных срабатываний для каждого этапа каскада, особо не повлиял на качество распознавания.

Выводы

В ходе выполнения данной магистерской диссертации были подробно рассмотрены две нейронный сети DetectNet и YOLO, а также веб-сервис DIGITS, который, как мне кажется, очень хорошо может помочь в исследовании и решении задач компьютерного зрения. Дальше было рассмотрено применение сети YOLO в рамках поставленной задачи. Данный подход показал очень хороший результат качества классификации по мере F_1, значение которой было равно 0.92~0.94. Далее, был рассмотрен метод Виолы-Джонса, который неплохо себя зарекомендовал в задаче поиска лиц людей на изображениях и видеопотоке. На основе его принципов был построен каскадный классификатор для локализации автомобилей на спутниковых изображениях и аэрофотоснимках. Для решения поставленной задачи была произведена модификация данного подхода. В качестве дескрипторов использовались не признаки Хаара, а локальные бинарные шаблоны, которые как мне кажется очень хорошо подходят для данной задачи в виду формы детектируемого объекта и их инвариантности относительно небольшого поворота. Также, в ходе проведенного исследования, которое заключалось в построении каскадных классификаторов при различных параметрах, было получено самое лучше и оптимальное решение. Вместо стандартного DAB, использовавшегося в методе В.-Д., было принято решение использовать GAB. В добавок ко всему, для увеличения скорости распознавания, были выбраны определенные рамки размера распознаваемых объектов, в виду их небольших изменений (все автомобили почти одинакового размера) на изображении и однородности данных. Точность и полнота полученного классификатора равнялась 0.94 и 0.95 соответственно, а мера F_1 0.95, хотя обучающая выборка по-прежнему еще была сравнительно небольшой. То есть данный подход не уступает по показателю F_1 нейронной сети YOLO.

Заключение

В рамках проведенной работы были выполнены все поставленные задачи:

- рассмотрены современные алгоритмы компьютерного зрения, а также их применение в рамках поставленной проблемы;
- построен каскадный классификатор на основе принципов метода Виолы-Джонса,
- была произведена модификация данного подхода: в качестве признаков использовались ЛБШ, вместо DAB использовался GAB, определение рамок размера объекта при детектировании;
- показатель полноты классификации превышает 80% и равен 95%;
- показатель точности классификации превышает 80% и равен 94%;
- исследован данный подход при различных параметрах обучения классификатора.

Список литературы

[1] Гашников М.В., Глумов Н.И., Мясников В.В., Мясников Е.В., Сергеев В.В., Чернов А.В. Компьютерная обработка картографических и спутниковых изображений, 2010.

[2] Шокин Ю.И., Москвичев В.В., Ничепорчук В.В. Методика оценки антропогенных рисков территорий и построения картограмм рисков с использованием геоинформационных систем // Вычисл. технологии. 2010. Т.15. №1. С. 120-131.

[3] Mayer H. Automatic object extraction from aerial imagery - a survey focusing on buildings // Computer Vision and Image Understanding. 1999. 74(2): pp. 138– 149.

[4] Hu J., You S., and Neumann U. Integrating lidar, aerial image and ground images for complete urban building modeling // in Third International Symposium on 3D Data Processing, Visualization, and Transmission, pp. 184–191, June 2006.

[5] Elaksher A. and Bethel J. Automatic generation of high quality 3D urban buildings from aerial images // Journal of Urban and Regional Information Systems Association. 2008. 20(2): pp 5-14.

[6] Lin C. and Nevatia R. Building detection and description from a single intensity image // Computer Vision and Image Understanding. 1998. Vol. 72, No. 2, pp. 101-121(21).

[7] Ding M. Automated, 3D, Airborne Modeling of Large Scale Urban Environments. Master's Thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Dec. 2007.

[8] Leitloff J., Rosenbaum D., Kurz F., Meynberg O., Reinartz P. An Operational System for Estimating Road Traffic Information from Aerial Images. Remote Sens. 2014;6:11315–11341.

[9] Liu K., Mattyus G. Fast Multiclass Vehicle Detection on Aerial Images // IEEE Geosci. Remote Sens. Lett. 2015;12:1–5.

[10] Moranduzzo T., Melgani F. Automatic Car Counting Method for

Unmanned Aerial Vehicle Images // IEEE Trans. Geosci. Remote Sens. 2014; 52:1635–1647.

[11] Moranduzzo T., Melgani F. Detecting Cars in UAV Images With a Catalog-Based Approach // IEEE Trans. Geosci. Remote Sens. 2014; 52: 6356–6367.

[12] Chen Z., Wang C., Luo H., Wang H. Vehicle Detection in High-Resolution Aerial Images Based on Fast Sparse Representation Classification and Multiorder Feature // IEEE Trans. Intell. Transp. Syst. 2016; 17: 2296–2309.

[13] Crawford, J. Beyond supply and demand: Making the invisible hand visible // Re-Work Deep Learning Summit, San Francisco. 2016.

[14] Shao W., Yang W., Liu G., Liu J. Car detection from high-resolution aerial imagery using multiple features // IEEE Int. Geosci. Remote Sens. Symp. 2012; 53: 4379–4382.

[15] Kluckner S., Pacher G., Grabner H., Bischof H. A 3D Teacher for Car Detection in Aerial Images // Proceedings of the IEEE 11th International Conference on Computer Vision; Rio de Janeiro. 2007; pp. 1–8.

[16] Van de Sande K.E.A., Uijlings J.R.R., Gevers T., Smeulders A.W.M. Segmentation as selective search for object recognition // Proceedings of the International Conference on Computer Vision; Barcelona, Spain. 2011; pp. 1879–1886.

[17] Alexe B., Deselaers T., Ferrari V. What is an object? // Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); San Francisco, CA, USA. 2010; pp. 73–80.

[18] Uijlings J.R.R., Sande K.E.A.V.D., Gevers T., Smeulders A.W.M. Selective Search for Object Recognition. Int. J. Comput. Vis. 2013; 104: 154–171.

[19] Kuo W., Hariharan B., Malik J. DeepBox: Learning Objectness with Convolutional Networks // Proceedings of the IEEE International Conference on Computer Vision; Los Alamitos, CA, USA. 2015; pp. 2479–2487.

[20] Ren S., He K., Girshick R., Sun J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv. 2016.

[21] Tuermer S., Kurz F., Reinartz P., Stilla U. Airborne Vehicle Detection in Dense Urban Areas Using HoG Features and Disparity Maps // IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 2013; 6: 2327–2337.

[22] Girshick R., Donahue J., Darrell T., Malik J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation // IEEE Trans. Pattern Anal. Mach. Intell. 2015; 38: 1.

[23] Girshick R. Fast R-CNN // Proceedings of the IEEE International Conference on Computer Vision; Los Alamitos, CA, USA. 2015.

[24] Deep Learning for Object Detection with DIGITS. https://devblogs.nvidia.com/parallelforall/deep-learning-object-detection-digits

[25] Christian S., Wei L., Yangqing J., Pierre S., Scott R., Dragomir A., Dumitru E., Vincent V., Rabinovich A.; Going Deeper with Convolutions. 2014.

[26] Hoiem D., Chodpathumwan Y., and Dai Q., Diagnosing Error in Object Detectors // ECCV. 2012.

[27] Car Localization and Counting with Overhead Imagery, an Interactive Exploration. https://medium.com/the-downlinq/car-localization-and-counting-with-overhead-imagery-an-interactive-exploration-9d5a029a596b

[28] Redmon J., Farhadi A. YOLO9000: Better, Faster, Stronger, 2016.

[29] Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection, 2016.

[30] YOLO. https://pjreddie.com/darknet/yolo/

[31] Cars Overhead With Context. http://gdo-datasci.ucllnl.org/cowc/

[32] YOLT2_COWC. https://photos.google.com/share/AF1QipOaySrRVsnGOCtTxR9TqH7xWzP_bAfgybVW1rCG7mHuJ43E8lWpKOCcuml1NZd6AQ?key=aXVKVVo4cWdrSklDSklQZEd5LU5ZY1FLb2dHeXN3

[33] Viola P., Jones M.J. Rapid Object Detection using a Boosted Cascade of Simple Features // proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001), 2001.

[34] Viola P., Jones M.J., Robust real-time face detection // International

Journal of Computer Vision, vol. 57, no. 2, 2004., pp.137–154.

[35] Papageorgiou, Oren, Poggio, A general framework for object detection // International Conference on Computer Vision, 1998.

[36] Freund Y., Schapire R.E. A Short Introduction to Boosting // Shannon Laboratory, USA, 1999., pp. 771-780.

[37] Liao S., Zhu X., Lei Z., Zhang L., Stan Z. Li. Learning Multi-scale Block Local Binary Patterns for Face Recognition // International Conference on Biometrics (ICB), 2007, pp. 828-837.

[38] AdaBoost. https://en.wikipedia.org/wiki/AdaBoost