Санкт-Петербургский государственный университет

Направление фундаментальной информатики и информационных технологий
Профиль математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей

Косатый Дмитрий Николаевич

Методы создания гибридных классификаторов на основе каскадов и глубоких нейронных сетей

Магистерская диссертация

Научный руководитель:
к. ф.-м. н., доц. Вахитов А. Т.

Рецензент:
Генеральный директор компании ООО "Биомоделирование" Петров А. Г.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY

Main Field of Study of Fundamental Computer Science and Information Technologies
Area of Specialisation of Software of Computers, Complexes and Networks

Dmitry Kosaty

Methods of creating hybrid classifiers based on cascades and deep neural networks

Master's Thesis

Scientific supervisor:
Associate professor Alexander Vakhitov

Reviewer:
General Director of the LLC Biomodeling Alexander Petrov

Saint-Petersburg
2016

Оглавление

Введение

Задача детекции объектов по изображениям – это важная задача современности. Её применение можно найти в разных областях: робототехнике, системах автовождения, приложениях по отслеживанию и встроенных устройствах [19]. Актуальной на сегодняшний день стоит задача детекции пешеходов [15]. Эта задача заключается в определении положения пешехода и его масштабов на изображении. Также такая задача может формулироваться в терминах класса, когда необходимо определить принадлежность пешехода заранее выбранному классу. В этом случае говорят о задаче классификации, простейшим видом которой является бинарная классификация [1]. В решении задачи детекции пешеходов первостепенными критериями являются качество и скорость. Детектор, удовлетворяющий этим характеристикам, с малой долей риска можно использовать в системах с ограниченной вычислительной мощностью [17].

На сегодняшний день существует множество методов решения задачи детекции. Среди них особо выделяются каскадные алгоритмы и нейронные сети. Каскадные методы получили широкое распространение благодаря высокой скорости обнаружения, позволяющей их применять в приложениях реального времени. Но вместе с тем, сравнивая их с современными решениями, каскадные процедуры проводят обнаружения с большим процентом ложных детекций, тем самым уступая многим другим методам в плане качества [20]. Нейронные сети, в свою очередь, демонстрируют высокую точность обнаружений, значительно сокращая уровень ложно позитивных детекций. Это их основное преимущество. Из недостатков нейронных сетей стоит отметить низкую скорость обнаружения, по которой они проигрывают многим другим подходам, несмотря на размер сети и связность слоёв. Как показано в статье [1] на задаче детекции пешеходов малая скорость работы не позволяет использовать нейросети в реальных приложениях даже с учётом ускорения на GPU-устройствах.

Это хорошо зарекомендовавшие себя подходы, реализацию кото-

рых можно найти во многих программных библиотеках. Среди них можно выделить OpenCV[1], дополнительные инструменты MATLAB[2] и т.д. При этом существует довольно много библиотек, предназначенных для работы исключительно с нейросетями: Caffe[3], Theano[4], Deepnet[5] и Pylearn2[6] и т.п. Сильной стороной перечисленных библиотек можно считать наличие открытого исходного кода, а также быструю C++ реализацию ядра библиотеки, ускоренную на GPU. Следует отметить, что библиотеки Caffe и Theano используют библиотеку cudaDNN[7], а Deepnet и Pylearn2 – библиотеку cuda-convnet [10]. Данные инструментальные средства предоставляют мощные функциональные возможности, удовлетворяющие многим потребностям исследователей в области компьютерного зрения. Но в этом преимуществе кроятся и свои недостатки. Среди них следует отметить отсутствие высокоточных и быстрых методов, отражающих актуальное состояние предлагаемых решений. К этим методам относятся такие, которые фокусируются либо на повышении качества [12], либо на ускорении обнаружений [16]. Кроме того, в случае с MATLAB нельзя говорить о открытом исходном коде, что также можно считать за недостаток.

Вместе с тем в реальных приложениях актуальными остаются подходы, которые сочетают как скорость, так и точность. К таким методам можно отнести получившие сегодня распространение гибридные подходы, сочетающие каскадные методы с глубокими нейросетями. Это интуитивное решение брать лучшее из каждого метода демонстрирует хорошие результаты [17]. Такой подход раскрывает широкие возможности по созданию новых детекторов. Одним из гибридных алгоритмов был разработан в составе исследовательской группы Alex Krizhevsky. Предложенный им подход показывает на сегодняшний день самые высокие

[1]http://docs.opencv.org/2.4/modules/objdetect/doc/objdetect.html

[2]http://www.mathworks.com/products/computer-vision/, http://www.mathworks.com/products/neural-network/

[3]https://github.com/BVLC/caffe

[4]https://github.com/Theano/Theano

[5]https://github.com/nitishsrivastava/deepnet

[6]https://github.com/lisa-lab/pylearn2

[7]https://developer.nvidia.com/cudnn

результаты как в качестве, так и скорости. Данный метод получил название DeepCascade [17]. Сильным преимуществом этого решения по сравнению с другими является то, что оно основано на двух свободно-распространяемых библиотеках Doppia [2] и cuda-convnet2 [11]. Данные библиотеки представляют собой GPU-реализацию быстрого каскадного алгоритма VeryFast и глубокой нейронной сети.

Несмотря на то, что отдельные алгоритмы имеются в составе библиотек с открытым кодом, их интеграция в единую систему не является решенной задачей. Разработка такой библиотеки с открытым кодом позволит использовать ее для прототипирования алгоритмов детекции других видов объектов, например, автомобильных номеров или лиц. Предлагаемая магистерская работа и посвящена разработке библиотеки для детекции объектов на основе интеграции каскадного и нейросетевого подходов.

1. Постановка задачи

Цель работы состоит в реализации библиотеки детекции объектов, основываясь на гибридное сочетание актульных существующих решений.

Для её осуществления были поставлены следующие задачи:

- Исследовать актуальные существующие гибридные решения по детекции пешеходов.
- Реализовать гибридный классификатор DeepCascade.
- Создать программную C++ библиотеку, содержащую DeepCascade.
- Произвести обучение детектора пешеходов на изображениях средствами библиотек.
- Провести сравнительное тестирование реализованного гибридного классификатора с методом VeryFast и глубокой нейронной сетью.

2. Обзор

2.1. Существующие методы

2.1.1. Каскадный классификатор

В симбиозном сочетании методов широкое распространение получили каскадные алгоритмы, также извественые, как классификаторы со скользящим окном. Первым методом такого типа был предложен Paul Viola и Michael Jones для задачи обнаружения лиц на фотографиях. Для этой задачи они использовали признаки Хаара – прямоугольные особенности, вид которых подобен вейвлетам Хаара (рис. 1) [14]. Позже актуальным стало применять признаки LBP по причине их более быстрого счёта. Обнаружение объектов каскадными методами основано на этих признаках – вычисляя свёртку признака с изображением, детектор анализирует содержимое.

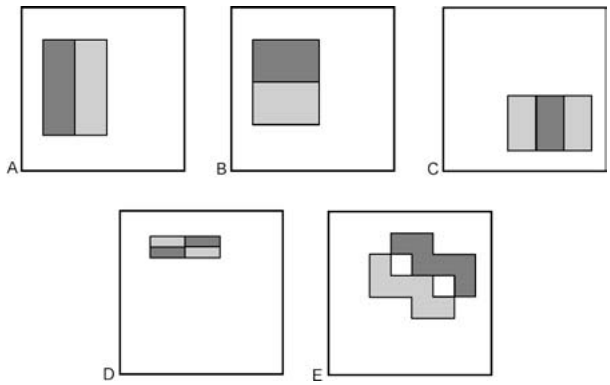

Рис. 1: Признаки Хаара [22]

Основное преимущество данного алгоритма заключается в скорости обнаружении, компромиссном к качеству. Как отмечают авторы, их детектор работает со скоростью 15 кадров в секунду, на половину сокращая уровень ложно позитивных детекций [20]. Это достигается преимущественно за счёт использования интегрального представления изображения вместо пиксельного, что позволяет выполнить вычисление признака за пару операций. Помимо этого, выбор оптимального признака из всевозможного набора осуществляется модификаций алгоритма обучения AdaBoost, что повышает качество обнаружения. Ал-

горитм AdaBoost (англ. Adaptive Boosting) – это алгоритм, усиливающий классификаторы, показавшие плохие результаты на предыдущих этапах (так называемые слабые классификаторы), для последующего объединения их в один итоговый (также именуемый сильным) классификатор [5].

Каскадным этот алгоритм назван из-за использования каскада классификаторов, который в ходе обучения фокусируется на релевантной области с искомым объектом (рис. 2). Этот подход способствует раннему отбрасыванию регионов, где заведомо не должен располагаться объект [21].

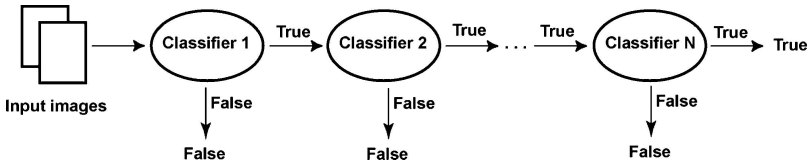

Рис. 2: Архитектура каскадного классификатора [22]

2.1.2. Нейронные сети

Ещё одним методом, который стало модным сочетать с другими подходами, является нейронная сеть. Исследование нейросетей насчитывают обширную историю, которая началась ещё в первой половине 20 века. Их анализ неразрывно связан с разработкой искусственного интеллекта.

Классические нейронные сети представляют собой ориентированный граф с узлами в качестве нейронов, которые расположены по слоям таким образом, что нейроны одного слоя неким образом связаны с нейронами следующего слоя (рис. 4). Структура нейронов соответствует модели МакКалока-Питтса (рис. 3) – линейной комбинации входов $l_1, \ldots, l_N$ и весов $w_1, \ldots, w_N$, по которой на основании отклика функции активации Sum генерируется выход y. Самые первые нейросети имели полносвязную структуру, что отрицательно сказывалось на эффективности. По этой причине в конце 60-х годов прошлого века интерес у научного сообщества к нейросетям угас.

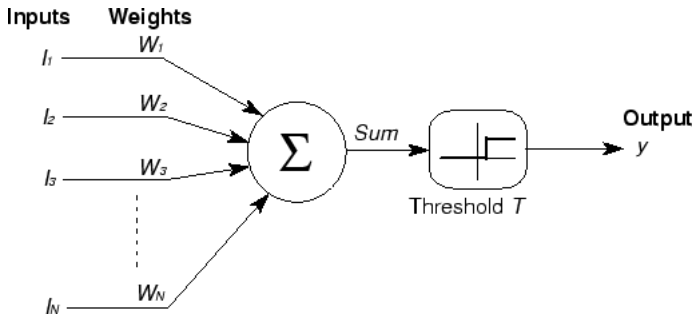

Рис. 3: Модель нейрона МакКалока-Питтса

В 1989 г. Yann LeCun предложил свёрточные нейронные сети, дав вторую жизнь нейросетевой теории. Применены они были к задаче классификации цифр из базы данных рукописных цифр MNIST[8] [7]. Этот вид нейросетей по ряду причин предоставляет более эффективный инструмент для обнаружения, чем классический подход:

- Взамен матричного умножения, используемого в классических нейросетях, вычисление карт признаков проводится с помощью математической операции свёртки;
- В противовес традиционным нейросетям, где преобладает полносвязная структура сети, в свёрточных сетям используется разряженное представление слоёв, где нейроны одного слоя связаны не со всеми нейронами другого слоя;
- Для ускорения вычисления признаков вводится pool-слой, сокращающий размер карт признаков (рис. 4)

Операция свёртки представлет собой линеный оператор вида

$$(K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n), \tag{1}$$

где I – это изображение, K – ядро свёртки [6].

Свёрточные нейронные сети, как и классические, для подбора оптимальных параметров весов в слоях сети используют метод обратного распространения ошибки. Это модификаиця метода градиентного спуска, используемая для обучения нейросети. Основная его задача в том,

[8]http://yann.lecun.com/exdb/mnist/

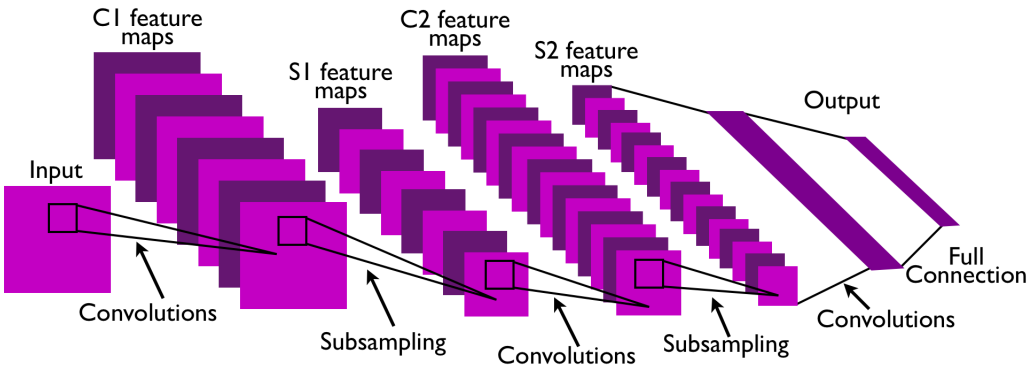

Рис. 4: Типичная свёрточная нейросеть [13]

чтобы минимизировать ошибки классификации. Реализуется это путём распространения ошибок от выходов сети к её входам для корректировки значений весов. Важным условием этого алгоритма является дифференцируемость функций активации нейронов [18].

В 2006 г. Geoffrey Hinton представил мировому сообществу глубокие нейронные сети. Под глубиной следует понимать размер сети от её входов до выходов. Отличительной чертой данной разновидности нейросети является высокая точность обнаружения. Создавая карты признаков из карт признаков предыдущих слоёв, такой подход проводит более детальный анализ объекта.

2.2. Используемые технологии

В основу гибридного детектора DeepCascade положены две свободнораспространяемые библиотеки: Doppia [2] с алгоритмом VeryFast и cuda-convnet [10].

2.2.1. Метод VeryFast

Метод VeryFast – это каскадный алгоритм по типу скользящего окна, использующий HOG-признаки. Как показывает практика, такое сочетание приводит к эффекту размытия по той причине, что каскадные алгоритмы работают с интегральным представлением изображения. Это первый метод, которому удалось совместить каскады с HOG-признаками [16].

HOG-признаки, или признаки гистограмм направленных градиентов (англ. Histogram Oriented Gradient) были предложены Navneet Dalal и Bill Triggs в 2005 году для задачи обнаружения пешеходов на БД INRIA [3]. Они продемонстрировали высокие результаты, адаптировав эти признаки для алгоритма обучения SVM (англ. Support Vector Machine).

Этот метод фокусируется на скорости обнаружения, достигающий, как отмечают авторы, 50 кадров в секунду [16]. Это достигается за счёт сочетания двух высокоэффективных методов: ChnFtrs [9] и FPDW [4], а также во многом благодаря C++/CUDA реализации.

Метод ChnFtrs Данный метод составляет основу алгоритма VeryFast. Детектор ChnFtrs был предложен Piotr Dollar в 2009 году, и основан на идеи "интегральных характеристик канала" (англ. Integral Channel Features). Это прямоугольные признаки, определяющие отклик фильтра по рассматриваемой области изображения. В оригинальной статье были использованы 6 квантованых ориентаций, 1 градиентное значение и 3 LUV цветовых канала, которых достаточно для получения конкурентноспособных результатов (рис. 5). Согласно ChnFtrs алгоритму, после такого представления строится двухуровневое дерево решения, по которому создаётся сильный классификатор в виде линейной комбинации весов. Обучение таких деревьев и их весов проводится алгоритмом дискретного AdaBoost'a.

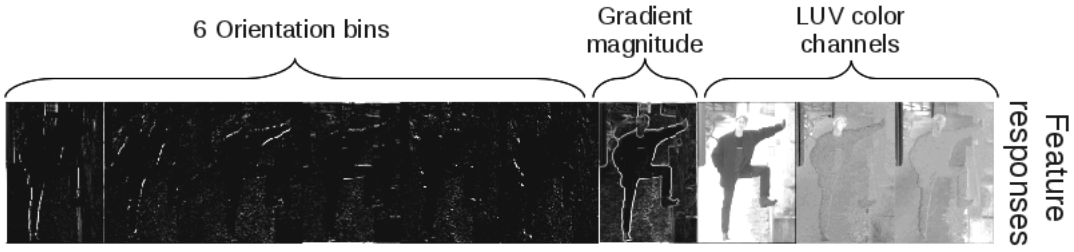

Рис. 5: Интегральные характеристики канала [16]

Авторы метода VeryFast для обучения использовали настройки, с которыми метод ChnFtrs показал наилучшие результаты: обученный описанным выше спобом сильный классификатор содержил 2 тыс. сла-

бых классификаторов, где признаки проивольным образом извлекались из пула размером в 30 тыс. прямоугольников; первые итерации обучения проводились с 5 тыс. негативных образцов, после чего их количество увеличилось вдвое, добавляя при этом 5 тыс. дополнительных негативных примеров.

Авторами метода VeryFast были реализовано две версии метода ChnFtrs: CPU и GPU, из которых большего внимания заслуживает последняя.

Метод FPDW Ключевой особенностью VeryFast метода, во многом по которой достигается такая высокая скорость работы, является то, что авторами был перенесён процесс изменения размера изображения с процедуры тестирования на этап обучения. Для достижения этого ими была заимствована идея детектора FPDW – классификатора, разработанного Piotr Dollar в 2010 году (англ. Fastest Pedestrian Detector in the West). Это модификация алгоритма ChnFtrs до многомасштабного детектора.

Если классические детекторы, работающие с множеством масштабов, обучаются на единственной модели для одного канонического (или, проще говоря, заранее выбранного) масштаба, вслед за чем изменяют размер изображения в N раз (рис. 7), то идея FPDW детектора состоит в изменении размера изображения в N/K раз, также обучаясь на одной модели на единственном масштабе (рис. 8). Каждое таким образом изменённое изображение участвует в свёртке с признаком, которые после используются в свёртке вместе с оставшимися $N - N/K$ масштабами. Коэффициент K был выбран авторами эмпирическим способом с целью сокращения количества изменений размера изображений и вычисления признаков, и равен ~ 10.

Согласно алгоритму VeryFast, выбирается некий канонический классификатор, который преобразуется в K классификаторов для других масштабов. Исходя из этого, возможно провести обучение N/K (~ 5) классификаторов: по одному для каждого набора масштабов 0.5, 1, 2 и т.д. На этапе тестирования производится преобразование N/K в N классификаторов (по одному на масштаб). Вслед за этим вычисляются

Рис. 6: Наивный подход

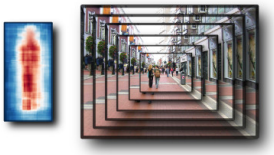

Рис. 7: Традиционный подход

Рис. 8: Детектор FPDW

Рис. 9: Детектор VeryFast

Рис. 10: Различный подходы по обнаружению объектов с множеством масштабов [16]

интегральные характистики канала исходного изображения, за которым N классификаторами определяется отклик для каждого масштаба (рис. 9).

Этот алгоритм имеет сходства с наивным подходом, при котором проводится обучение столько моделей, сколько имеем масштабов, т.е. порядка 50 моделей (рис. 6), но в разы эффективнее его.

2.2.2. Библиотека cuda-convnet

Это C++/CUDA реализация свёрточной (прямого направления) нейронной сети, обучение которой проводится методом обратного распространения ошибки. Ядро этой библиотеки написано C++ и ускорено графическим ускорителем CUDA, когда как интерфейс представляет собой скрипты на языке Python. К системным требованиям относится поддержка графических процессоров "поколения Fermi" [10].

Интерфейс этой библиотеки располагает Python-модулями для следующих задач:

- Обучение нейросети.
- Демонстрация её работы: карт признаков любого слоя, наглядный тест.
- Проверка градиента.
- Написание конфигурационных файлов с архитектурой нейросети, а также задание особым образом её внешнего вида: проинициализировать значения весов не нормальным распределением с

нулевым математическим ожиданием и указанным стандартным отклонением, а предварительно обученнымии весами.

Отличительной чертой этой библиотеки является введение так называемого dropout-слоя (англ. dropping out units), предовращающего возникновение переобучения нейросети [8]. Процесс переобучение – это явление, при котором обученная на тренировочных данных нейросеть показывает плохие результаты на тестовых образцах. Эта проблема может возникнуть при использование полносвязных слоёв с множеством параметров, которые усложняю модель. При рассмотрении после полносвязного слоя dropout-слоя, на котором с вероятностью 0.5 отбрасываются нейроны, становится возможным эффективное обучение нейросети методом обратного распространения ошибки.

3. Гибридный классификатор DeepCascade

3.1. Архитектура

3.1.1. Алгоритм

Согласно описанной в статье [17] идеи, гибридный классификатор DeepCascade строится на основе двух подходов – метода VeryFast и глубокой нейронной сети DNN. Его псевдо-алгоритм можно записать следующим образом:

Algorithm 1 DeepCascade

input: the images dataset $\mathcal{D}(I, M)$, where $\mathcal{I}$ – the images set, $\mathcal{M}$ – the markings set
output: the labels set $\mathcal{L}$

```
1:  for all image ∈ 𝒟 do
2:      detected_objects ← VeryFast(image)
3:      for all detected_object ∈ detected_objects do
4:          resized_object ← ResizeTo32x32(detected_object)
5:          for all real_object ∈ real_objects do        ▷ real_objects ∈ ℳ
6:              if detected_object ∩ real_object > 1/2 MAX(detected_object, real_object) then
7:                  positive_samples ← resized_object
8:              else
9:                  negative_samples ← resized_object
10: ℒ ← DNN(positive_samples, negative_samples)          ▷ DNN – Deep Neural Network
```

Дадим словесное пояснение приведённому алгоритму:

1. Каждое изображение из базы данных обрабатывается методом VeryFast, детектируя пешеходов и возвращая набор детекций в виде прямоугольных областей (см. строки 1–2).

2. Полученная детекция сопоставляется с разметкой изображения, т.е. с истинным положением и размером объекта (см. строки 3–5).

3. Если детекция будет пересекаться с разметкой более чем на половину, то изображение рассматривается как положительное, иначе как отрицательное (см. строки 6–9).

4. Для каждой детекции выполняется изменение входного изображения до размера 32x32 (см. строку 4).

5. По набору положительных примеров создаётся позитивная база, а по множеству отрицательных – негативная (см. строки 7, 9).

6. Проводится обработка глубокой нейронной сетью образцов с предыдущего шага (см. строку 10)

   (a) Обучение нейросети осуществляется на основе данных, которые были помечаны в исходной базе как тренировочные.

   (b) Тестирование нейросети проводится на основе данных, помеченных в исходной базе как тестовые.

Наглядная демонстрация псевдо-алгоритма приведена на рис 11. На рисунке схематично изображена последовательность работы гибридного классификатора, где по входному изображению методом VeryFast распознаётся пешеход (красным цветом выделены детекции). На основании этих детекций и исходной разметки (помечена синим цветом) осуществляется сопоставление верно определённых и ложно сдетектированных пешеходов. По каждой строится своё множество образцов, которое в итоге подаётся на вход нейронной сети. Таким образом, нейронная сеть работает не с исходными изображения, которые имеются в базе, а с теми окнами, который выдал метод VeryFast.

Использование метода VeryFast и нейросети в такой последовательности обоснованы тем, что каскадный детектор будет выступать в роли фильтра, пропуская нейросети только релевантные окна. Тем самым сужается область потенциальных областей с искомым объектом, что эффективно сказывается на скорости работы нейросети.

3.1.2. Топология глубокой нейронной сети

Топология нейросети была выбрана следующего вида рис 12. Как показано в [10] такая последовательность слоёв даёт хорошие результаты.

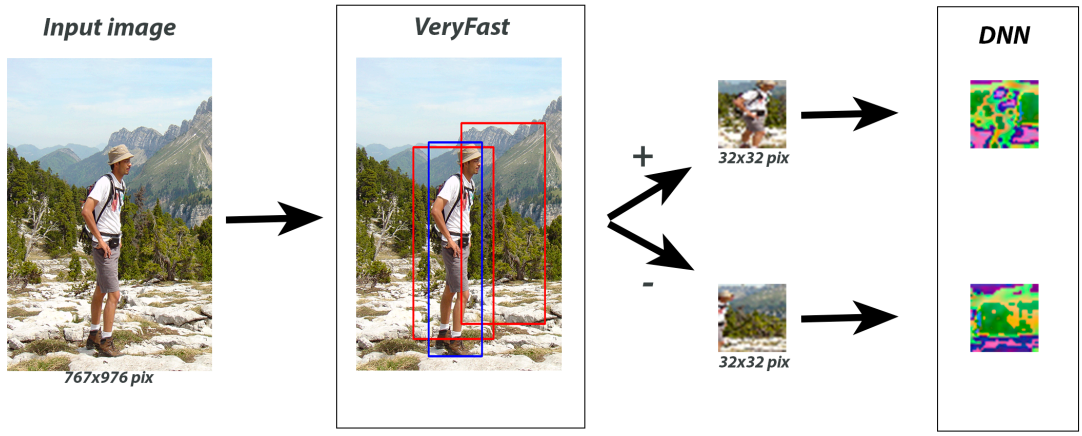

Рис. 11: Архитектура гибридного классификатора

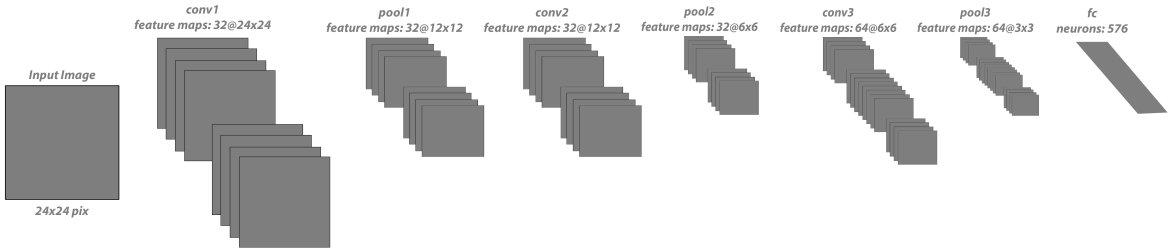

Рис. 12: Топология глубокой нейронной сети

Как видно из рисунка, топология нейросети представлет собой последовательность трёх свёрточных и подвыборочных слоёв, замыкаемая полносвязным слоем. Условно её можно представить в виде:

$$input \to (conv1 \to pool1) \to (conv2 \to pool2) \to (conv3 \to pool3) \to fc$$

В обозначениях приведённого изображения "conv$N$", определяет N-ый свёрточный слой, "pool$N$" характеризует N-ый слой подвыборки, и, наконец, "fc" выражает полносвязный слой. Для каждого N-го слоя (кроме последнего) изображены карты признаков, полученных с предыдуего $N-1$ слоя, с шаблоном "$M@mxm$", где M – количество карт признаков и mxm – их пиксельный размер. В последнем полносвязном слое условно изображено множество нейронов, по взвешенной комбинации которых строится выходное значение нейросети.

3.2. Обучение

3.2.1. Предварительное обучение

Для обучения нейронной сети начальные значения весов были заданы из нейросети, предварительно обученной на наборе данных CIFAR-10[9]. Её архитектура соответствует изображенной на рисунке топологии, с той лишь разницей, что на выходе полносвязного слоя предобученной сети будет проиводится классификация 10 меток, когда как у нашей нейросети всего два класса: пешеход и не пешеход. Таким образом, значения весов были взяты с первых трёх слоёв предобученной сети, а полносвязный слой задан из нормального распределения $N(0, \sigma^2)$.

Это распространённая практика обучения нейронных сетей, потому что позволяет использовать веса, которые уже обучены на выявление определённых закономерностей. Более того, такой вид инициализации более предпочтителен, чем задание начальных значений из нормального распределения, что также можно встретить на практике [17].

3.2.2. Создание данных

Для того, чтобы более качественно обучить нейронную сеть, из экземпляров изображений, пропущенных методом VeryFast, было создано в 10 раз больше образцов. За эту процедуру создаются крошечные изображения размера 24x24, которые получены произвольным обрезанием исходного изображения вокруг рамки с пешеходом. Такое расширение тренировочного множества положительно сказывается на обучении, поскольку делает эти наборы более разнообразными, а также предовращает возникновения процесса переобучения [10].

[9]https://www.cs.toronto.edu/ kriz/cifar.html

4. Программная реализация

В ходе работы над реализацией метода DeepCascade были разработаны два модуля, каждый из которых отвечает задаче, решаемой гибридным классификатором: модуль, детектирующий пешеходов, что отвечает методу VeryFast, и модуль, классифицирующий пешеходов, что соответствует глубокой нейронной сети.

4.1. Детектор

Данный модуль реализован на языке программирования C++ 11 версии. Были использованы все библиотеки, которые требуются для библиотеки Doppia [16]: STL, Boost и т.д. В качестве системы сборки была выбрана кроссплатформенная CMake.

Для более удобной работы с методом VeryFast в рамках библиотеки Doppia было написано собственное API, состоящее их двух функций:

- метода *init*, принимающего в качестве входных параметров аргументы командной строки (опцию с конфигурационным файлом метода VeryFast) и инициализирующего детектор:

```
int init(int argc, char *argv[]);
```

- процедуры *compute*, которая по входному изображению возвращает множество прямоугольных детекций:

```
std::vector<cv::Rect> compute(cv::Mat image);
```

Предложеное API может быть полезно не только в рамках данного проекта, но и для смежных задач, как детекция объектов на видео, поскольку упрощает работу с изображениями, подаваемыми на вход каскадному алгоритму. Проект можно найти по адресу `https://bitbucket.org/dkosaty/objects_detection`. Для работы требуется загрузить библиотеку Doppia и заменить модуль "objects_detection" на предложенный.

Опишем программный интерфейс, используемый для оценки окон, пропускаемых методом VeryFast:

- Detector – абстрактный класс, интерфейс которого используют:
  - VeryFastDetector – непосредственно, главный детектор.
  - PseudoDetector – тестовый детектор, который выполняет проверку правильного подсчёта оценок статистическим вычислителем.
- StatisticsComputer – вычислитель оценочных параметров, как true positive, false positive, true negative и false negative, а также точности, полноты и accuracy.
- PreLearner – подготовитель данных, полученных после работы метода VeryFast на всей базе изображений, которые затем используются глубокой нейронной сетью. В его функции входит сжатие детекций до размера 32x32 и сохранение их в качестве отдельных изображений.
- LibUtilities – набор вспомогательных инструментов для работы с исходной базой изображений, как чтение картинок и извлечение разметок из аннотационных файлов.

Проект расположен по адресу `https://bitbucket.org/dkosaty/test_objects_detection`.

Введение класса PseudoDetector обосновано тем, что по верно и ложно позитивным оценкам создаются соответствующие образцы для нейросети, что крайне важно для получения высоких результатов гибридным методом. Нейросеть получает на вход только те окна, которые пропустил VeryFast.

4.2. Классификатор

Данный модуль реализован на на языке программирования Python версии 2.7. По аналогии с модулем, детектирующим пешеходов, данная программа включает в себя классификатор со статистическим вычислителем. В функции классификатора предусмотрена процедура обучения,

которая реализуется библиотекой cuda-convnet. Статистический вычислитель работает с метками классов $\in [0, 1]$, в то время, как аналогичная программа модуля-детектор работает с разметками. Реализацию данного классификатора доступна по ссылке `https://bitbucket.org/dkosaty/alexnet_test`.

В качестве вспомогательного набора скриптов был реализован сериализатор изображений, полученных от метода VeryFast. В его задаче входит преобразовать входные изображения в формат batch-файлов, пригодные для использования библиотекой cuda-convnet. Для проверки правильности его работы был создан десериализатор, выполняющий обратное преобразование: по наборе batch-файлов извлекать множество изображений. Реализованные скрипты могут быть применены для любой другой базы изображений. Реализация сериализатора расположена на сайте `https://bitbucket.org/dkosaty/alexnet_serializer`, десериализатора – `https://bitbucket.org/dkosaty/alexnet_deserializer`.

Во всех представленных проектах с IT-хостинга `bitbucket.org` автор фигурировал под именем *dkosaty*.

По мимо тех инструментов, которые используются в каждой из библиотек Doppia и cuda-convnet, была задействована библиотека компьютерного зрения для обработки изображений: чтения/записи, изменения размера и т.д.

Исходный метод DeepCascade использовал библиотеку cuda-convnet2 – усовершенствованный аналог библиотеки cuda-convnet. Основное их отличие в том, что cuda-convnet2 работает исключительно с более мощными видеокартами NVIDIA "поколения Kepler" [11]. В собственной реализации метода DeepCascade была использована библиотека cuda-convnet, что обосновано технической проблемой недоступности соответствующей cuda-convnet2 аппаратуры.

5. Эксперименты

Все приведённые эксперименты проводились на машине с процессором Intel Core-i5, графической видеокартой NVIDIA GeForce 720M и на операционной системе Ubuntu 14.04 LTS.

5.1. Набор данных

В качестве базы образцов был выбран набор данных INRIA[10], представляющий собой прогуливающихся людей (в том числе и пешеходов) на фоне городских и пригородных сцен. Этот набор был предоставлен исследователями Dallal и Triggs в 2005 году, на которой они продемонстрировали новый метод детекции пешеходов HOG+SVM [3]. Он разбит на тренировочные и тестовые выборки, в каждой из которых содержится множество положительных и отрицательных примеров. Размер учебной выборки составляет 1832 изображения, из которых 614 с пешеходами и 1218 без них, а объём проверочной выборки равен 741 изображению, где 288 с пешеходами и 453 – без.

5.2. Критерии оценивания

Основными критериями по оценке работы гибридного классификатора DeepCascade были выбрана критерии качество и скорость. Характеристика качества оценивается 3 параметрами: точностью (precision), полнотой или чувствительность (recall или sensitivity), и accuracy. Эти величины строятся на основе таких оценок, как true positive (TP), false positive (FP), true negative (TN) и false negative (FN).

Распишем более подробно о каждой из них. Для этого введём гипотезу H_0, утверждающую, что на изображении присутствует искомый объект. В такое случае возможны следующие четыре ситуации:

- True positive: классификатор говорит, что на изображении присутствует искомый объект, и он там на самом деле есть. В этом случае говорят, что гипотеза H_0 верно принята;

[10] http://pascal.inrialpes.fr/data/human/

- False positive: классификатор говорит, что на изображении присутствует искомый объект, но его на самом деле там нет. Говорят, что гипотеза H_0 неверно принята, что называют ошибкой 2-го рода;

- True negative: классификатор говорит, что на изображении искомый объект отсутствует, и его на самом деле там нет. Тогда говорят, что гипотеза H_0 верно отвергнута;

- False negative: классификатор говорит, что на изображении искомый объект отсутствует, а он там на самом деле есть. Гипотеза H_0 неверно отвергнута, что называют ошибкой 1-го рода.

Исходя из этого, точность определяется выражением $\frac{TP}{TP+FP}$, что можно интерпретировать, как долю объектов определённого класса, который детектор верно определил, ко всем объектам, которые детектор определил как относящимся к данному классу. Таким же образом вводится метрика полноты отношением вида $\frac{TP}{TP+FN}$, что представляет собой долю объектов определённого класса, который детектор верно определил, ко всем объектам, которые представлены в выборке из данного класса. Наконец, accuracy можно вычислить через соотношение $\frac{TP+TN}{TP+FP+TN+FN}$, и понимать как долю объектов, по которой детектор принял верное решение, т.е. количество объектов, по которым детектор принял правильное решение, делённое на размер выборки.

Величину скорости для метода VeryFast будем оценивать параметром FPS (англ. Frame Per Second), также известным как кадровая частота. Пусть t_i – это время, с которым детектор обрабатывал i-ый кадр. Тогда скорость работы с этим кадром будет $s_i = \frac{1}{t_i}$. Проводя замеры скорости на N кадрах, будем понимать под FPS среднюю скорость обработки N кадров в единицу времени, т.е. $\text{FPS} = \frac{1}{N}\sum_{i=1}^{N} s_i$. Нейросеть будем оценивать параметром SPW (англ. Second Per Window), или количеством секунд, задействованных ею на обработку 1 окна. Окном будем считать отдельный участкок изображения, как в случае с DeepCascade, или весь кадр.

5.3. Сравнительное тестирование

Был осуществлён сравнительный анализ гибридного классификатора на базе данных INRIA по критериям качества и скорости как непосредственного его самого, так и его составных частей. В таблице 1 приведёны результаты работы метода VeryFast:

Data	pos	neg	frames	windows	TP	FP	TN	FN	precision(%)	recall(%)	accuracy(%)	FPS
Train	614	1218	1832	3175	1044	2131	920	201	32.8819	83.8554	45.7169	53
Test	288	453	741	1115	470	645	322	124	42.1525	79.1246	50.7367	37

Таблица 1: Результаты метода VeryFast

Приведённое количество кадров составляется из числа положительных и негативных изображений базы INRIA. Количество окон же определяется тем числом детекций, которые выдал метод, т.е. TP и FP, соответственно, как истинную и негативную детекцию. Остальные оценки TN и FN не содержат детекций метода, поэтому в качестве окон не рассматриваются. Как видно из оставшихся параметров, метод VeryFast выдаёт высокую скорость, но его точность не демонстрирует высоких результатов.

Было проведено обучение и тестирование нейронной сети на образцах, сгенерированных методом VeryFast. Результаты представлены в таблице 2.

Библиотека cuda-convnet требует одинаковое число положительных и негативных образцов как для обучения, так и для тестирования. Поэтому было рассмотрена тренировочная выборка из 1880 окон (940 каждого класса), а тестовая – из 940 (470 каждого класса).

Data	pos	neg	frames	TP	FP	TN	FN	precision(%)	recall(%)	accuracy(%)	SPW
Train	940	940	1880	381	61	515	195	86.199095	66.145833	77.777778	0.88
Test	470	470	940	220	83	205	68	72.607261	76.388889	73.784722	0.88

Таблица 2: Результаты гибридного классификатора

Итоговое тестирование было проведено над глубокой нейронной сетью по всей базе INRIA. Результаты продемонстрированы в таблице 3. Аналогичным образом было рассмотрено 1152 тренировочных (576 образцов каждого класса) и 576 тестовых примеров (288 каждого класса). Обучение было проведено на 250 примерах.

Data	pos	neg	frames	TP	FP	TN	FN	precision(%)	recall(%)	accuracy(%)	SPW
Train	576	576	1152	572	5	571	4	99.133449	99.305556	99.218750	0.9
Test	288	288	576	263	41	247	25	86.513158	91.319444	88.541667	0.9

Таблица 3: Результаты глубокой нейронной сети

Как видно из результатов, показатели глубокой нейросети выше, чем у гибридного классификатора, что можно объяснить высоким процентом ошибок каскадного метода, что приводит к шумам в размеченных наборах. Нередка ситуация, когда в качестве ложной детекции фигирирует позитивный образец, что снижает качетство нейросети.

Заключение

В ходе выполнения исследовательской работы были выполнены следующие задачи:

- Исследованы существующие актуальные гибридные решения по обнаружению объектов.
- Реализован гибридный классификатор DeepCascade.
- Произведено обучение детектора пешеходов на изображениях средствами библиотек.
- Проведено сравнительное тестирование разработанного гибридного классификатора.

Список литературы

[1] Angelova A., Krizhevsky A., Vanhoucke V. Pedestrian detection with a Large-Field-Of-View deep network // IEEE International Conference on Robotics and Automation. — 2015.

[2] Benenson R. Doppia library // bitbucket.org, веб-сервис для хостинга IT-проектов. — 2015. — URL: `https://bitbucket.org/rodrigob/doppia/` (online; accessed: 1.05.2016).

[3] Dalal N, Triggs B. Histograms of Oriented Gradients for Human Detection // IEEE Conference on Computer Vision and Pattern Recognition. — 2005.

[4] Dollar P., Belongie S., Perona P. The Fastest Pedestrian Detector in the West // British Machine Vision Conference. — 2010.

[5] Freund Y., Schapire R. A decision-theoretic generalization of on-line learning and an application to boosting // Journal of Computer and System Science. — no. 55.

[6] Goodfellow I., Bengio Y., Courville A. Deep learning. — 2016. — Book in preparation for MIT Press. URL: `http://www.deeplearningbook.org`.

[7] Handwritten Digit Recognition with a Back-Propagation Network / Y. LeCun, B. Boser, J. Denker et al. // Neural Information Processing Systems Conference. — 1990. — No. 2. — P. 396–404.

[8] Improving neural networks by preventing co-adaptation of feature detectors / G. Hinton, N. Srivastava, A. Krizhevsky et al. // Journal of Computing Research Repository. — 2012.

[9] Integral Channel Features / P. Dollar, Z. Tu, P. Perona, S. Belongie // British Machine Vision Conference. — 2009.

[10] Krizhevsky A. cuda-convnet library // code.google.com, веб-сервис для хостинга IT-проектов. — 2014. — URL: https://code.google.com/p/cuda-convnet/ (online; accessed: 1.05.2016).

[11] Krizhevsky A. cuda-convnet2 library // github.com, веб-сервис для хостинга IT-проектов. — 2014. — URL: https://github.com/akrizhevsky/cuda-convnet2 (online; accessed: 1.05.2016).

[12] Krizhevsky A., Sutskever I., Hinton G. ImageNet Classification with Deep Convolutional Neural Networks // Neural Information Processing Systems Conference. — 2012. — No. 25. — P. 1097–1105.

[13] LeCun Y., Kavukcuoglu K., Farabet C. Convolutional Networks and Applications in Vision // International Symposium on Circuits and Systems. — No. 253–256. — 2010.

[14] Papageorgiou C., Oren M., Poggio T. A general framework for object detection // International Conference on Computer Vision. — 1998. — P. 555–562.

[15] Pedestrian Detection: A Benchmark / P. Dolla, C. Wojek, B. Schiele, P. Perona // IEEE Conference on Computer Vision and Pattern Recognition. — 2009.

[16] Pedestrian detection at 100 frames per second / R. Benenson, M. Mathias, R. Timofte, L. Van Gool // IEEE Conference on Computer Vision and Pattern Recognition. — 2012. — P. 2903–2910.

[17] Real-Time Pedestrian Detection With Deep Network Cascades / A. Angelova, A. Krizhevsky, V. Vanhoucke et al. // British Machine Vision Conference. — 2015.

[18] Rumelhart D., Hinton G., Williams R. Learning Internal Representations by Error Propagation // Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1. — 1986. — P. 318–362.

[19] Ten Years of Pedestrian Detection, What Have We Learned? / R. Benenson, M. Omran, J. Hosang, B. Schiele // Conference on Computer Vision for Road Scene Understanding and Autonomous Driving. — 2014. — P. 613–627.

[20] Viola P., Jones M. Rapid object detection using a boosted cascade of simple features // IEEE Conference on Computer Vision and Pattern Recognition. — 2001.

[21] Viola P., Jones M. Robust Real-time Object Detection // International Journal of Computer Vision. — 2001.

[22] Viola P., Jones M., Snow D. Detecting Pedestrians Using Patterns of Motion and Appearance // International Journal of Computer Vision. — 2005. — no. 2. — P. 153–161.