Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Южно-Уральский государственный университет»

(национальный исследовательский университет)

Факультет математики, механики и компьютерных наук

Кафедра прикладной математики

|  |  |
| --- | --- |
| РАБОТА ПРОВЕРЕНА  Рецензент, заведующий кафедрой МиФА,  д.ф.-м.н, доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_В.Л. Дильман  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. | ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой ПРИМА, д.ф.-м.н., доцент  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Л.А. Прокудина  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. |

Детектирование ключевых точек на лице человека с использованием сверточных нейронных сетей

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

ЮУрГУ– 231300.2016.070.ПЗ ВКР

|  |  |
| --- | --- |
|  | Руководитель работы, , к.ф-м.н, доцент кафедры ПРИМА  \_\_\_\_\_\_\_\_\_\_\_\_\_/С.У.Турлакова |
|  | Автор работы  Студент группы ММиКН-473  \_\_\_\_\_\_\_\_\_\_\_\_\_/А.О. Литвинов  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. |
|  | Нормоконтролер, к.ф-м.н, доцент кафедры ПРИМА  \_\_\_\_\_\_\_\_\_\_\_\_\_/С.У.Турлакова  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. |

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

«Южно-Уральский государственный университет»

(национальный исследовательский университет)

Факультет математики, механики и компьютерных наук

Кафедра прикладной математики

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ Заведующий кафедрой,  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Л.А. Прокудина  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г. |

ЗАДАНИЕ

на выпускную квалификационную работу студента

Литвинову Андрею Олеговичу

Группа ММиКН-473

1. **Тема выпускной квалификационной работы** Детектирование ключевых точек на лице человека с использованием сверточных нейронных сетей

утверждена приказом ректора от 15.04.2016 № 661

1. **Срок сдачи студентом законченной работы «**1» **июня 2016 г.**
2. **Исходные данные в работе**
3. Язык программирования python 2.7
4. Среда программирования Sublime Text 2
5. Библиотеки theano, lasagne, OpenCV
6. **Перечень вопросов, подлежащих разработке**
7. Обзор литературы по решению задачи по тематике нейронных сетей
8. Составление программы расчёта.
9. Решение задачи для заданных начальных условий.
10. **Иллюстративный материал**
11. Общая схема. Демонстрационный плакат – 1 л.
12. Мультимедийная презентация – 10 слайдов.

Общее количество иллюстраций – 12.

1. **Дата выдачи задания** «10» февраля 2016 г.

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /С.У. Турлакова/

(подпись)

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /А.О. Литвинов/

(подпись)

1. **Календарный план подготовки выпускной квалификационной работы**

|  |  |  |
| --- | --- | --- |
| Наименование этапов выпускной квалификационной работы | Срок выполнения этапов работы | Отметка о выполнении |
| 1.Обзор литературы | 10.02.16-03.03.16 | выполнено |
| 2. Изучение численного метода для решения поставленной задачи. Решение тестовой задачи. | 04.03.16-25.04.16 | выполнено |
| 3.Получение результатов, формулировка выводов, структурирование текста | 25.04.16-15.05.16 | выполнено |
| 4.Подготовка текста | 16.05.15-01.06.15 | выполнено |
| 5. Проверка и рецензирование работы руководителем, исправление замечаний. Подготовка доклада и текста выступления. Внешнее рецензирование. | 02.06.15-13.06.15 | выполнено |
| 6. Защита выпускной квалификационной работы. | 01.06.16 |  |

**Руководитель работы**

к.ф-м.н, доцент кафедры ПРИМА \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ С. У. Турлакова

Задание принял к исполнению**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/**А.О. Литвинов

АННОТАЦИЯ

|  |
| --- |
| Литвинов А.О. Детектирование ключевых точек на лице человека с использованием сверточных нейронных сетей.– Челябинск: ЮУрГУ, ММиКН-473, 33 с., 11 ил., библиогр. список – 15 наим., 1 прил., 2 л. плакатов ф. А1. |

В дипломной работе реализована программа детекции ключевых точек лица человека на RGB изображении с помощью сверточной нейронной сети. Исследованы альтернативные методы для решения этой задачи. Также в рамках данной работы были проведены предварительные действия для приведения данных к корректному виду, необходимому для подачи на вход нейронной сети.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 7](#_Toc452909242)

[1 ПОСТАНОВКА ЗАДАЧИ 8](#_Toc452909243)

[2 АЛЬТЕРНАТИВНЫЕ МЕТОДЫ РЕШЕНИЯ 9](#_Toc452909244)

[2.1 Сравнение шаблонов 9](#_Toc452909245)

[2.2 Сравнение эластичных графов 9](#_Toc452909246)

[2.3 Гибкие контурные модели 11](#_Toc452909247)

[2.4 Синтез объектов линейных классов 11](#_Toc452909248)

[2.5 Методы, основанные на геометрических характеристиках лица 12](#_Toc452909249)

[2.6 Метод главных компонент 13](#_Toc452909250)

[3 ДЕТЕКТИРОВАНИЕ ЧЕЛОВЕЧЕСКОГО ЛИЦА НА ИЗОБРАЖЕНИИ ДЛЯ ВХОДА СВЕРТОЧНОЙ СЕТИ 16](#_Toc452909251)

[3.1 Описание метода 16](#_Toc452909252)

[3.2 Интегральное представление изображений 16](#_Toc452909253)

[3.3 Признаки Хаара 17](#_Toc452909254)

[3.4 Построения классификатора на основе алгоритма бустинга и построение каскадной структуры 18](#_Toc452909255)

[4 НАХОЖДЕНИЕ ТОЧЕК ЛИЦА ПО ДЕТЕКТИРОВАННОМУ СЕКТОРУ 20](#_Toc452909256)

[4.1 Структура сверточных нейронных сетей 20](#_Toc452909257)

[4.2 Принцип работы сверточных нейронных сетей 22](#_Toc452909258)

[4.2.1 Convolution слой 22](#_Toc452909259)

[4.2.2 Pooling слой 22](#_Toc452909260)

[4.2.3 Fully-connected 23](#_Toc452909261)

[4.3 Принцип обучения сверточных нейронных сетей градиентными методами 24](#_Toc452909262)

[4.3.1 Вычисление ошибки 24](#_Toc452909263)

[4.3.2 Вычисление градиента 25](#_Toc452909264)

[5 МЕТОДЫ ПРИМЕНЯЕМЫЕ ПРИ ОБУЧЕНИИ 26](#_Toc452909265)

[5.1 Обучение на GPU 26](#_Toc452909266)

[5.2 Увеличение тренировочной выборки 26](#_Toc452909267)

[5.3 Предотвращение переобучения 26](#_Toc452909268)

[5.4 Dropout 27](#_Toc452909269)

[5.5 Перемешивание обучающих примеров 28](#_Toc452909270)

[5.6 Предотвращение паралича сети 28](#_Toc452909271)

[6 АРХИТЕКТУРА СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ В ДАННОЙ РАБОТЕ 29](#_Toc452909272)

[6.1 Общая струтктура 29](#_Toc452909273)

[6.2 Подробности архитектуры 29](#_Toc452909274)

[6.3 Количество выходов, при заданной архитектуре 29](#_Toc452909275)

[6.4 Количество настраиваемых весов, при заданной архитектуре 30](#_Toc452909276)

[7 РЕЗУЛЬТАТЫ 31](#_Toc452909277)

[ЗАКЛЮЧЕНИЕ 32](#_Toc452909278)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 33](#_Toc452909279)

[ПРИЛОЖЕНИЕ 34](#_Toc452909280)

[Текст программы на языке python 34](#_Toc452909281)

ВВЕДЕНИЕ

Искусственные нейронные сети в настоящее время получили широкое распространение в различных сферах человеческой деятельности, таких как распознавание и классификация образов, прогнозирование, управление. Пример конкретных задач – автоматическое управление автомобилем, распознавание изображений, речи, текстовые чат боты. Это произошло благодаря тому, что они имеют возможность синтезировать какие-либо алгоритмы, вычленяя из больших выборок данных произвольные закономерности. Изначально они построены по примеру биологических нейронных сетей – сетей нервных клеток животного. Один из видов искусственных нейронных сетей - сверточные нейронные сети, в основном применяющиеся для работы с изображениями. Причину этого можно увидеть в предпосылках их зарождения. Сверточные сети построены по примеру обработки визуальной информации у живых существ – в мозге есть области, выделяющие какие-либо инвариантные закономерности в изображениях – например линии.

Целью выпускной квалификационной работы является исследование возможностей применения сверточных нейронных сетей в области детекции произвольных объектов на RGB изображении. Эта область стала актуальной, в связи с появлением задач, где данный инструмент является вспомогательным а порой ключевым. А именно: досмотр людей на таможне, проект «Умный дом», различные задачи киноиндустрии, 3D анимация человекоподобных моделей и т.д. На сегодняшний день в данной сфере не существует решения дающего результат, хоть как-то близкий к идеальному.

Для достижения поставленной цели необходимо исследовать предметную область, исследовать методы для решения аналогичных задач, исследовать вспомогательные методы.

Также требуется разработать архитектуру нейронной сети для конкретной задачи – для детекции ключевых точек лица человека. Этот тип задачи относиться к классу регрессионных задач, то есть восстановление каких-либо зависимостей. Аналогичные методы решения возможно применять к схожим проблемам – распознавание рукописного текста, номеров автомобилей, дорожных знаков.

1. ПОСТАНОВКА ЗАДАЧИ

Данные получаемые с видеокамеры(кадр) в момент времени обозначим . Каждый кадр есть матрица размерности , каждый элемент принадлежит множеству , где – ширина кадра, - высота кадра, R,G, B – множества в диапазоне от 0 до 255, обозначающие кодировки цветов в формате RGB. Видеопоследовательность есть упорядоченная последовательность кадров (), где – количество кадров в видеопоследовательности.

Сформулируем задачу детектирования точек на лице человека. По имеющемуся кадру предсказать положение точек лица человека, предполагая при этом, что в кадре находится произвольное количество людей, на расстоянии от 0.5 до 2 метров от камеры. Требуется предсказать положение 8 точек:

* Центр подбородка
* Левый край губ
* Правый край губ
* Центр носа
* Левую височную область
* Левый край левого глаза
* Правую височную область
* Правый край правого глаза

Процесс детекции происходит в два этапа:

1. Детектирование человеческих лиц на изображение с помощью каскада Хаара, для последующей передачи выделенных областей на вход сверточной нейронной сети,
2. Детектирование на выделенной области точек лица с помощью сверточной нейронной сети.

В работе необходимо из видеопоследовательности в режиме реального времени выделять точки лица. В качестве исходных данных выступает видеопоследовательность, полученная с неподвижной камеры, с разрешением пикселей в формате RGB глубиной 8 бит на один канал и частотой кадров 30 в секунду.

Заданные входные данные накладывают на возможные методы решения ограничения в плане оптимизации.

1. АЛЬТЕРНАТИВНЫЕ МЕТОДЫ РЕШЕНИЯ

**2.1 Сравнение шаблонов**

Сравнение шаблонов (Template Matching) - заключается в выделении шаблонов областей лица(глаза, рот, брови) и последующим сравнении каким либо образом, например попиксельным. Совпадение областей с шаблонными увеличивает вероятность присутствия на изображении лица или частей лиц. Этот метод был одним из самых первых методов, предназначенных для детекции людей.

Недостаток метода сравнения шаблонов заключается в том, что он требует много ресурсов: памяти – для хранения различных эталонных изображений, так и вычислительных – попиксельное сравнение со всеми шаблонами. В ввиду примитивности алгоритма требуется, чтобы изображения были сняты со строго определенного ракурса, с определенной освещенностью.



Рисунок 1.1 - Шаблонные области лица

2.2 Сравнение эластичных графов

Метод сравнения эластичных графов (Elastic Bunch Graph Matching) представляет лицо в виде графа, где вершины этого графа расположены на ключевых точках лица, таких как края лица, глаза, рот. Каждая грань имеет пометку о расстоянии между ее вершинами.

В каждой такой точке вычисляются коэффициенты разложения по функциям Габора для восьми ориентаций и пяти различных частот.

Функция Габора:

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

где , ,

- коэффициент сжатия, характеризующий эллиптичность функции Габора, - ориентация нормали параллельных полос функции Габора, - сдвиг фаз в градусах, – длина волны множителя-косинуса.

Набор коэффициентов называется *джетом*. Джеты характеризуют локальные области изображений и необходимы для нескольких целей:

1. Нахождение точек соответствия в двух различных изображениях.
2. Сравнения двух областей различных изображений.

Каждый коэффициент для точек из одной области различных изображений характеризуется амплитудой , которая медленно меняется с изменением положения точки, и фазой , которая вращается со скоростью, пропорциональной частоте волнового вектора базисной функции. Поэтому в простейшем случае для поиска на новом изображении точки с аналогичными характеристиками в функции подобия фазу не учитывают:

|  |  |  |
| --- | --- | --- |
|  |  | (2.2) |

Функция подобия с одним джетом в фиксированной позиции и другим с переменной позицией является достаточно гладкой, для того чтобы получить быструю и надежную сходимость при поиске с применением простейших методов таких, как диффузия или градиентный спуск. Более совершенные функции подобия привлекают информацию о фазе. Для различных ракурсов соответствующие ключевые точки отмечаются вручную на тренировочном наборе. Кроме того, чтобы для одного и того же лица представить различные вариации его изображения в одном и том же графе, для каждой точки используются несколько джетов, каждый из которых может соответствовать различным локальным характеристикам данной точки, например, открытому и закрытому глазу. Процесс распознавания неизвестного лица состоит в сравнении графа изображения лица со всеми остальными графами из набора при помощи функции подобия

|  |  |  |
| --- | --- | --- |
|  |  | (2.3) |

Левая сумма характеризует подобие джетов, вычисленное с применением фазочувствительной функции, правая - топографическое соответствие, которое пропорционально квадрату разности расстояний между соответствующими вершинами сравниваемых изображений, - количество вершин, - количество граней, - коэффициент относительной важности топографической информации.

В представленном выше виде метод способен достаточно надежно распознавать при изменениях ракурса до 20 ; при больших углах точность распознавания резко уменьшается, функция подобия оказывается более чувствительной к ракурсу, чем к межклассовым различиям. Дальнейшее развитие метода заключается в извлечении коэффициентов важности на основе анализа обучающей выборки. Для каждого джета симплекс-методом вычисляется коэффициент важности, который затем используется в функции подобия. Коэффициенты важности вычисляются из условия максимизации функции подобия для одного и того же лица и минимизации - для различных лиц. Существуют также более ранние разновидности этого метода, которые не используют изначально определенные ключевые точки и структуры графа. Одни из них используют для сравнения решетки джетов, наложенные на изображение, см. рисунок 1.2. В неизвестном изображении отыскиваются точки соответствия, и затем по найденным точкам строится искаженная решетка и измеряется мера ее искажения для определения наиболее похожего изображения. В других методах точки извлечения джетов изначально образуют решетку, а затем наименее пригодные для распознавания точки отсеиваются в процессе обучения.



Рисунок 1.2 - a) наложенная на изображение эластичная решетка; б) искаженная версия наложенной решетки

2.3 Гибкие контурные модели

Производиться на основе сравнения контуров лица. Контуры обычно получаются из очертаний головы, глаз, губ, носа. Контуры представлены ключевыми точками лица и интерполированными значениями остальных точек, лежащих между ключевых. Для уточнения контуров в различных методах возможно использование априорной информации, так и информация, извлеченной из тренировочной выборки. В тренировочной выборке ключевые точки расставляются вручную. Для поиска контуров нового лица используется алгоритм имитации отжига с функцией с целевой функцией от двух составляющих:

1. Ищется максимум при соответствующей интенсивности пикселов, извлеченных на перпендикулярной контуру линии, аналогичным пикселам из тренировочной выборки
2. Совпадении контура с формой контуров тренировочной выборки

Для сравнения изображений используются значения главных компонент, вычисленные на наборе векторов, представляющих собой координаты ключевых точек. Главной задачей при распознавании по контурам является правильное выделение этих контуров. В общем виде эта задача по сложности сравнима непосредственно с распознаванием изображений.

2.4 Синтез объектов линейных классов

Данный метод позволяет синтезировать новые изображения объекта (и в частности, изображения лица) для разных ракурсов. Имеется тренировочный набор изображений лиц и только одно изображение нового объекта в определенном ракурсе. Тренировочный набор состоит из изображений объектов того же класса (класс лиц в данном случае), что и новый объект, и включает в себя изображения различных лиц, причем для каждого лица имеются его изображения в широком диапазоне ракурсов. Для нового объекта, имеющего изображение в ракурсе , осуществляется линейное разложение на изображения объектов из тренировочного набора в том же ракурсе, с вычислением коэффициентов , где - количество объектов в тренировочном наборе. Синтез изображения в новом ракурсе для нового объекта осуществляется сложением изображений из тренировочного набора в ракурсе с теми же коэффициентами: . Таким образом, метод позволяет синтезировать изображения нового объекта в различных ракурсах по изображению в одном ракурсе без привлечения сложных трехмерных моделей. Данный метод является перспективным для синтеза изображений в новых ракурсах без привлечения сложных трехмерных моделей, однако вопрос о качестве и количестве примеров в тренировочном наборе остается открытым.

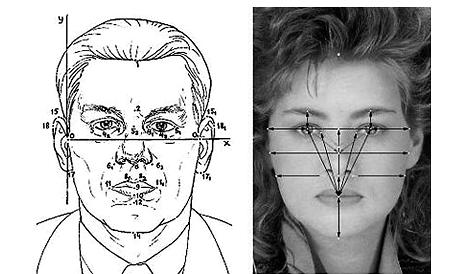
2.5 Методы, основанные на геометрических характеристиках лица

Один из самых первых методов - это анализ геометрических характеристик лица. Изначально он применялся в криминалистике и был там детально разработан. Потом появились компьютерные реализации этого метода. Суть его заключается в выделении набора ключевых точек (или областей) лица и последующем выделении набора признаков. Каждый признак является либо расстоянием между ключевыми точками, либо отношением таких расстояний. В отличие от метода сравнения эластичных графов, здесь расстояния выбираются не как дуги графов. Наборы наиболее информативных признаков выделяются экспериментально.

Ключевыми точками могут быть уголки глаз, губ, кончик носа, центр глаза и т. п. см. рисунок 1.3. В качестве ключевых областей могут служить прямоугольные области, включающие в себя: глаза, нос, рот.

В процессе распознавания сравниваются признаки неизвестного лица с признаками, хранящимися в базе. Задача нахождения ключевых точек приближается к трудоемкости непосредственно распознавания, и правильное нахождение ключевых точек на изображении во многом определяет успех распознавания. Поэтому изображение лица человека должно быть без помех, мешающих процессу поиска ключевых точек. К таким помехам относят очки, бороды, украшения, элементы прически и макияжа. Освещение желательно равномерное и одинаковое для всех изображений. Кроме того, изображение лица должно иметь фронтальный ракурс, возможно, с небольшими отклонениями. Выражение лица должно быть нейтральным. Это связано с тем, что в большинстве методов нет модели учета таких изменений.

Таким образом, данный метод предъявляет достаточно строгие требования к условиям съемки и нуждается в надежном механизме нахождения ключевых точек для общего случая. Кроме того, требуется применение более совершенных методов классификации или построения модели изменений. В общем случае этот метод не является самым оптимальным, однако для некоторых специфических задач все же перспективен. К таким задачам можно отнести документный контроль, когда требуется сравнить изображение лица, полученного в текущий момент, с фотографией в документе. При этом других изображений этого человека не имеется, и, следовательно, механизмы классификации, основанные на анализе тренировочного набора, недоступны.



(а) (б)

Рисунок 1.3 - Идентификационные точки и расстояния: *а)* - используемые при криминалистической экспертизе; *б)* - наиболее часто применяемые при построении автоматизированных систем идентификации

2.6 Метод главных компонент

Метод главных компонент (МГК) включает математический метод, который преобразует ряд вероятностно коррелированных переменных в меньшее количество некоррелированных переменных, называемых основными компонентами. МГК является очень известным подходом, который используется для расчета набора функций для распознавания лица. Это может относиться к любому лицу:

1. Лицо может быть представлено минимальным набором характеристик.

2. Может быть построено с использование небольшой коллекции собственных фотографий. Изображение лица проецируется на некоторое количество шаблонов лиц, называемое собственными лицами, затем вычисляется разница между этими лицами, которая будет считаться отличительной чертой для изображения. Когда набор собственных лиц вычислен, изображения лица может быть реконструировано взвешенной комбинацией собственных лиц. Веса образуют вектор признаков для представления лица и распознавания. Когда подается новое тестовое изображение, веса высчитаны благодяря проекции изображения на вектор собственных лиц. Классификация между изображениями выходит из сравнения расстояний между векторами весов тестового изображения и обучающего множества (входных данных). Как уже отмечено, можно реконструировать исходное изображение из собственных лиц так, что входное изображение должно точно соответствовать оригинальному изображению, используя все собственные лица, извлеченные из оригинальных изображений. МГК является математическим методом, который основан на преобразовании переменных, с помощью которого можно преобразовать набор коррелированных переменных в набор некоррелированных переменных. Переменные, которые некоррелируют друг с другом – основные компоненты. Количество основных компонентов означает, что некоррелированных должно быть меньше или равно количеству исходных переменных. В такого рода трансформации первые главные компоненты будут иметь самый высокий приоритет, который покажет максимальное отклонение. Это поможет вычислить точность данных. Если набор данных распределен нормально, только тогда главные компоненты могут быть независимыми от других переменных. Снижение размеров может привести к потере информации, но какое-то количество информации так или иначе останется.

МГК-АЛГОРИТМ

Пусть имеется такой набор лиц для обучения:

Среднее лицо по набору может быть определено:

|  |  |  |
| --- | --- | --- |
|  |  | (2.4) |

Каждое лицо отличается от среднего вектором признаков:

|  |  |  |
| --- | --- | --- |
|  |  | (2.5) |

Матрица ковариаций строится благодаря вектору:

|  |  |  |
| --- | --- | --- |
|  |  | (2.6) |

Где матрица Множество больших векторов затем подвергают МГК . Получаем вектор весов изображения лица. Лицо проецируется на пространство лиц:

|  |  |  |
| --- | --- | --- |
|  |  | (2.7) |

Для , где это номер собственного лица, используемого для распознавания. Веса формируют вектор .

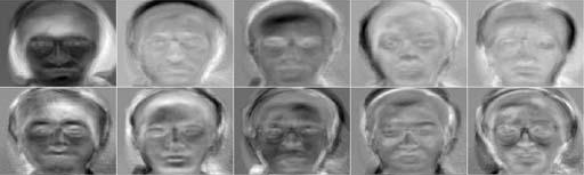


Рисунок 2.4 - Пример собственных векторов(собственных лиц), полученных на обучаемом наборе лиц.

3 ДЕТЕКТИРОВАНИЕ ЧЕЛОВЕЧЕСКОГО ЛИЦА НА ИЗОБРАЖЕНИИ ДЛЯ ВХОДА СВЕРТОЧНОЙ СЕТИ

3.1 Описание метода

Для детектирования лица используется метод, называемый метод Виолы-Джонса, также называемый каскадом Хаара.

Метод был разработан в 2001 г. Майклом Джонсом и Полом Виолой. Этот метод до сих пор не потерял своей актуальности в связи с одним из лучших по соотношению показателей скорости работы и эффективности распознавания. Этот детектор обладает крайне низкой вероятностью ложного срабатывания, порядка 0.00007%. Алгоритм хорошо работает при ориентации детектируемого объекта в пределах 30 градусов. При большем угле, чем 30 градусов, процент обнаружения резко уменьшается. Это не позволяет в стандартной реализации, без каких-либо модификаций, производить детекцию человеческого лица, повернутого под произвольным углом.

Данный метод в общем виде ищет лица и черты лица по общему принципу сканирующего окна.

3.2 Интегральное представление изображений

Для расчета суммарной яркости прямоугольного участка изображения, используют интегральное представление. Это представление используется часто и в других методах, например в фильтрах Хаара, вейвлет преобразованиях, SURF.

Интегральное представление позволяет быстро рассчитывать суммарную яркость произвольного прямоугольника на данном изображении, причем время расчета не зависит от площади прямоугольника. Оно записывается в виде матрицы, размеры которой совпадают с размерами исходного изображения I:

|  |  |  |
| --- | --- | --- |
|  |  | (3.1) |

где – значение интегрального представления в точке , — яркость пиксела исходного изображения.

Каждый элемент матрицы представляет собой сумму интенсивностей пикселей в прямоугольнике от до . Расчет такой матрицы занимает линейное время. Для того чтобы вычислить сумму прямоугольной области в интегральном представлении изображения требуется всего 4 операции обращения к массиву и 3 арифметические операции. Это позволяет быстро рассчитывать признаки Хаара для изображения в процессе обучения и распознавания.

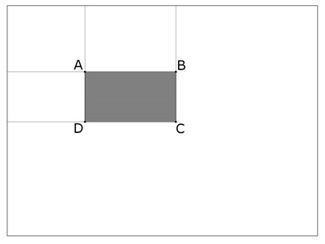


Рисунок 3.1

Для прямоугольного изображения ABCD ее интегральное значение можно выразить через суммы и разности смежных интегральных значений прямоугольников по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (3.2) |

3.3 Признаки Хаара

Целью признаков Хаара является сравнения яркостей в двух прямоугольных областях изображения.

В стандартном методе Виолы–Джонса используются прямоугольные признаки, см. рисунок 3.2. Эти признаки называются примитивами Хаара.

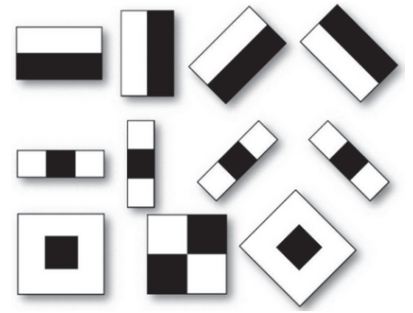


Рисунок 3.2 - Примитивы признаков Хаара

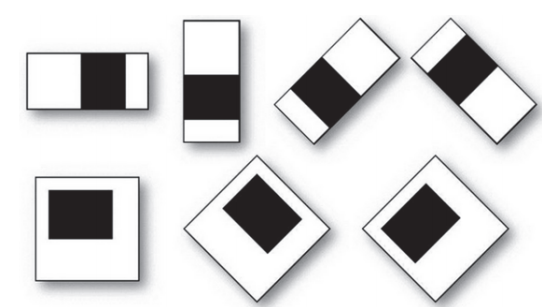


Рисунок 3.3. - Дополнительные признаки Хаара

В расширенном методе Виолы–Джонса, представленном в библиотеке OpenCV, используются дополнительные признаки см. рисунок 3.3.

Вычисляемым значением такого признака будет:

|  |  |  |
| --- | --- | --- |
|  |  | (3.3) |

где – значение интегрального представления, закрываемое светлой частью признака, а – значение интегрального преставления, закрываемое темной частью признака. Признаки Хаара дают значение перепада яркости между и .

3.4 Построения классификатора на основе алгоритма бустинга и построение каскадной структуры

Бустинг (boosting) — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов. Представляет собой жадный алгоритм построения композиции алгоритмов. Изначально понятие бустинга возникло в работах по вероятно почти корректному обучению в связи с вопросом: возможно ли, имея множество плохих (незначительно отличающихся от случайных) алгоритмов обучения, получить хороший.

В течение последних 10 лет бустинг остаётся одним из наиболее популярных методов машинного обучения, наряду с нейронными сетями и машинами опорных векторов. Основные причины — простота, универсальность, гибкость (возможность построения различных модификаций), и, главное, высокая обобщающая способность.

Простое голосование базовых классификаторов:

|  |  |  |
| --- | --- | --- |
|  |  | (3.4) |

– классифицируемый объект, – базовые классификаторы

Композиция может быть лучше базовых если они лучше случайного классификатора и достаточно различны.

Каскадная структура повышает скорость обнаружения, фокусируя свою работу на наиболее информативных областях изображения.

Структура каскадного детектора приведена на рисунке 3.4. Каскад состоит из слоев, которые представляют собой классификаторы, обученные с помощью процедуры бустинга.

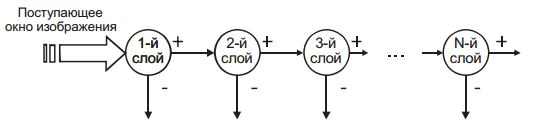


Рисунок 3.4. - Структура каскадного детектора

1. НАХОЖДЕНИЕ ТОЧЕК ЛИЦА ПО ДЕТЕКТИРОВАННОМУ СЕКТОРУ

4.1 Структура сверточных нейронных сетей

Сверточные нейронные сети содержат три типа слоев:

1. Convolutional(сверточные). В операции свертки используется лишь ограниченная матрица весов небольшого размера, которую «двигают» по всему обрабатываемому слою (в самом начале по входным данным), формируя после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. То есть для различных нейронов выходного слоя используются общие веса — матрица весов, которую также называют набором весов или ядром свертки. Она построена таким образом, что графически кодирует какой-либо один признак, например, наличие наклонной линии под определенным углом. Тогда следующий слой, получившийся в результате операции свертки такой матрицей весов, показывает наличие данной наклонной линии в обрабатываемом слое и ее координаты, формируя так называемую карту признаков (feature map). В сверточной нейронной сети количество ядер, как правило много: они кодируют признаки исходных данных.

При этом в зависимости от метода обработки краёв исходной матрицы результат может быть меньше исходного изображения , такого же размера или большего размера .

1. Pooling, subsampling (операция субдискретизации). После каждого сверточного слоя, может быть pooling слой. Pooling слой берет небольшие прямоугольные блоки из сверточного слоя и вычисляет один выход из этого блока. Есть несколько способов выполнить эту операцию, например, взяв среднее, максимальное или какую-либо линейную комбинацию нейронов в блоке.
2. Fully-Connected(персептрон, MLP, Полносвязная сеть).После нескольких convolutional и max-pooling слоев следуют полносвязные слои. Fully connected слой принимает все нейроны из предыдущего слоя (будь то fully connected, pooling или convolutional) и соединяет его с каждым нейроном, который он имеет. Fully connected слои не могут быть расположены где угодно, то есть не может быть сверточных слоев после того, как идет fully connected слой.

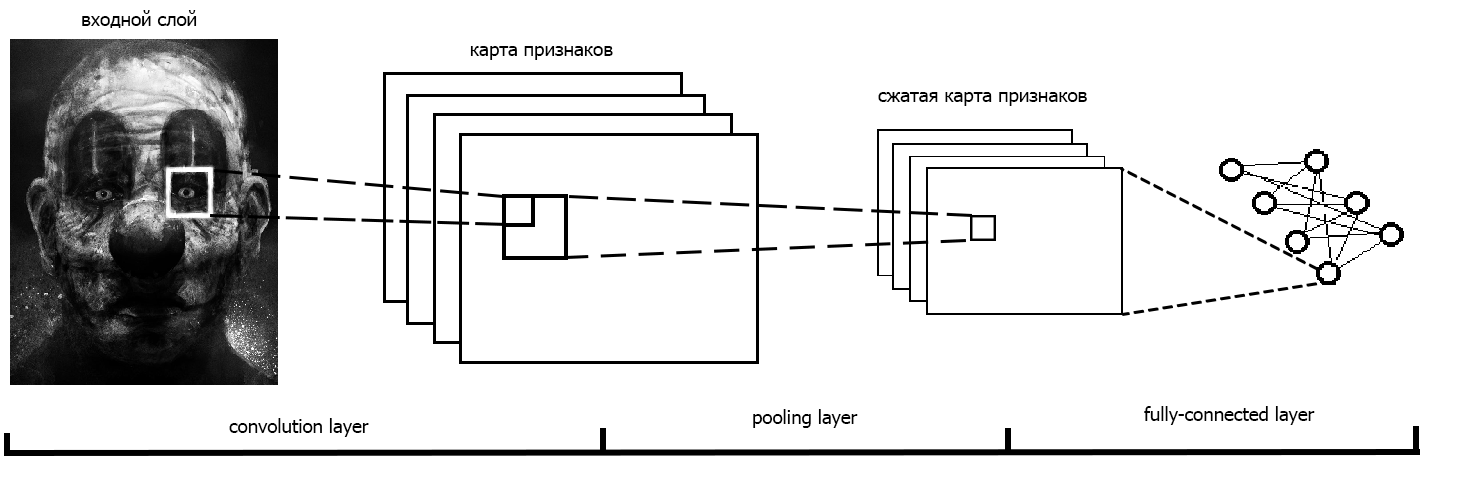


Рисунок 4.1 – общая структура сверточной нейронной сети

Первые два типа слоёв (convolutional, pooling), чередуясь между собой, формируют входной вектор признаков для fully-connected. Fully-connected слой по вектору признаков выдает обобщенный результат. Сеть можно обучать с помощью градиентных методов.

Применение нейронных сетей не ограничивается двумерным случаем. Возможно точно таким же образом построить одно или трехмерные сверточные нейронные сети; convolutional фильтры будут просто иметь размеры, определенные надлежащим образом, и pooling слои поменяют размерность. Например, возможно использовать одномерные сверточные сети на аудио данных.

4.2 Принцип работы сверточных нейронных сетей

4.2.1 Convolution слой

Свёрточный слой реализует идею локальных рецептивных полей, то есть каждый выходной нейрон соединен только с определённой областью входной матрицы и таким образом моделирует некоторые особенности человеческого зрения.

Этот слой описывается следующий формулой:

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

Здесь - выход слоя , - функция активации, - коэффициент сдвига, символом ∗ обозначена операция свёртки входа с ядром .

При этом за счёт краевых эффектов размер исходных матриц изменяется.

|  |  |  |
| --- | --- | --- |
|  |  | (4.2) |

Здесь - карта признаков (выход слоя ), - функция активации, - коэффициент сдвига для карты признаков , - ядро свёртки номер , - карты признаков предыдущего слоя.

4.2.2 Pooling слой

Слои этого типа выполняют уменьшение размера входной карты признаков. Это можно делать разными способами, рассмотрим метод выбора максимального элемента (max-pooling) - вся карта признаков разделяется на ячейки размера, из которых выбираются максимальные по значению. Формально слой может быть описан следующим образом.

|  |  |  |
| --- | --- | --- |
|  |  | (4.3) |

Здесь - выход слоя , - функция активации, - коэффициенты, - операция выборки локальных максимальных значений.

Использование этого слоя позволяет улучшить распознавание образов с изменённым масштабом (уменьшенных или увеличенных).

4.2.3 Fully-connected

Последний из типов слоёв это слой многослойного перцептрона (MLP), его можно описать следующим соотношением.

|  |  |  |
| --- | --- | --- |
|  |  | (4.4) |

Здесь - выход слоя , - функция активации, - коэффициент сдвига, - матрица весовых коэффициентов.

4.3 Принцип обучения сверточных нейронных сетей градиентными методами

4.3.1 Вычисление ошибки

* Для выходного (Fully-connected)

Ошибка рассчитывается следующим образом.

|  |  |  |
| --- | --- | --- |
|  |  | (4.5) |

Здесь - ожидаемый (учебный) выход, - реальный выход, - производная функции активации по её аргументу

* Для скрытых слоёв Fully-connected

Ошибка имеет следующий вид.

|  |  |  |
| --- | --- | --- |
|  |  | (4.6) |

Здесь - ошибка слоя , - производная функции активации, - состояние (не активированное) нейронов слоя , - матрица весовых коэффициентов слоя .

* Ошибка на выходе convolutional слоя

Формируется путём простого увеличения размера матриц ошибки следующего за ним pooling слоя.

|  |  |  |
| --- | --- | --- |
|  |  | (4.7) |

Здесь - ошибка слоя , - производная функции активации, - состояние (не активированное) нейронов слоя , - операция увеличения размера матриц.

* Ошибка на выходе pooling слоя

Рассчитывается путём выполнения ”обратной свёртки” карт признаков следующего за ним свёрточного слоя, то есть над каждой картой признаков выполняется свёртка с соответствующим ”перевернутым” ядром, при этом за счёт краевых эффектов размер исходных матриц изменяется. Далее над получившимися картами вычисляются несколько частичных сумм по числу ядер свертки, в соответствии с матрицей смежности polling и convolutional слоёв.

|  |  |  |
| --- | --- | --- |
|  |  | (4.8) |

Здесь - ошибка слоя , - производная функции активации, - состояние (не активированное) нейронов слоя , - ядра свёртки.

4.3.2 Вычисление градиента

Процедура вычисления градиента функции ошибки. Обучение сводиться к её минимизации в пространстве весов сети.

* Градиент для ядра свёртки можно посчитать как свёртку матрицы входа convolutional слоя с "перевёрнутой" матрицей ошибки для выбранного ядра.

|  |  |  |
| --- | --- | --- |
|  |  | (4.9) |

Здесь - ошибка слоя , - вход слоя , - ядра свёртки.

* Градиент для сдвига для convolutional слоя вычисляется как сумма значений соответствующей матрицы ошибки.

|  |  |  |
| --- | --- | --- |
|  |  | (4.10) |

Здесь - ошибка слоя

* Градиент для коэффициентов pooling слоя вычисляется следующим образом.

|  |  |  |
| --- | --- | --- |
|  |  | (4.11) |

Здесь - выход слоя , - ошибка слоя , - операция выборки локальных максимальных значений.

* Градиент для коэффициента сдвига для pooling слоя вычисляется как сумма значений соответствующей матрицы ошибки.

|  |  |  |
| --- | --- | --- |
|  |  | (4.12) |

Здесь - ошибка слоя

* Градиент для весов MLP выглядит следующим образом.

|  |  |  |
| --- | --- | --- |
|  |  | (4.13) |

Здесь - ошибка слоя , - вход слоя , - матрица весовых коэффициентов слоя .

1. МЕТОДЫ ПРИМЕНЯЕМЫЕ ПРИ ОБУЧЕНИИ

5.1 Обучение на GPU

Важную роль в процессе обучения нейронных сетей занимает время ее обучения, так как современные сети как правило очень сложны и без какой-либо оптимизации они могут обучаться годами. Один из способов ускорить обучение – это распараллеливание процессов.

В сверточных сетях есть независимые операции(например вычисление по различным фильтрам соответствующим им future map)

В глубоком обучении активно используется обучение на видеокартах, так как они обладают множеством ядер.

GPU используются для обучения глубоких нейронных сетей с помощью намного более крупных обучающих последовательностей в более сжатые сроки, с использованием меньшей инфраструктуры ЦОД. GPU также используются, чтобы воспроизводить эти учебные модели машинного обучения для выполнения задач классификации и прогнозирования на облаке. При этом графические процессоры позволяют работать с данными большего объема и с более высокой производительностью, потребляя меньше энергии и на базе меньшей инфраструктуры.

К числу тех, кто впервые применил графические ускорители для решения задач машинного обучения, относятся многие крупные веб-компании и социальные сетевые серверы, наряду с научно-исследовательскими институтами высокого ранга в области обработки и анализа данных и машинного обучения. Благодаря тысячам вычислительных ядер и увеличению производительности приложений в 10-100 раз по сравнению с CPU, GPU стали процессорами, которые выбирают специалисты по обработке данных для работы с данными большого объема.

5.2 Увеличение тренировочной выборки

Зачастую в глубоком обучении наибольшую выгоду приносит простое увеличение тренировочной выборки.

5.3 Предотвращение переобучения

Переподгонка, переобучение, (overfitting, overtraining,) —явление, возникающее при решении задач обучения “c учителем”, когда ошибка на тренировочной выборке оказывается значительно меньше, чем ошибка на тестовой выборке. Переобучение возникает при избыточно большой подгонке под тренировочную выборку или при использовании избыточно сложной модели.

Для того, чтобы избежать переобучения, требуется использовать дополнительные методы:

1. Перекрёстная проверка (*Cross-validation*)

Процедура оценивания обобщающей способности алгоритмов, обучаемых по какой-либо тренировочной базе данных.

Выбирается множество разбиений исходной выборки на две подвыборки исходных тренировочных данных: на новую тренировочную и тестовую. Для каждого разбиения выполняется обучение алгоритма на тренировочной подвыборке, затем оценивается его средняя ошибка на объектах тестовой подвыборки. Оценкой скользящего контроля называется средняя по всем разбиениям величина ошибки на контрольных подвыборках.

Если выборка независима, то средняя ошибка скользящего контроля даёт несмещённую оценку вероятности ошибки. Это отличает её от средней ошибки на обучающей выборке, которая может оказаться смещённой (заниженной) оценкой вероятности ошибки.

Перекрестная проверка является стандартной методикой предотвращения переобучения и тестирования алгоритмов регрессии, классификации и прогнозирования.

1. L1 и L2 регуляризация

Регуляризация в машинном обучении— метод добавления некоторой дополнительной информации к условию с целью предотвратить переобучение или решить некорректно поставленную задач. Эта информация представляет штраф за избыточную сложность модели. В случае L1 это добавление к штрафующей функции модуля весов, в L2 – добавление квадратов весов.

1. Ранняя остановка обучения

5.4 Dropout

При неограниченных вычислительных ресурсах лучшим способом регуляризации моделей фиксированного размера является усреднение предсказания по всем возможным настройкам параметров. Комбинация моделей почти всегда увеличивает качество работы, но в больших сетях использование комбинации нескольких сетей непомерно дорого. Комбинирование нескольких сетей наиболее полезно, когда сети имеют различную архитектуру или обучаются на различных тренировочных данных. Обучение сразу нескольких сетей довольно дорогостоящая и долгая процедура из-за очень большого количества параметров сети, требующая большого количества вычислительных ресурсов. Более того, обучение больших сетей требует большого количества обучающих данных, которых всегда достаточно. Но даже если натренировать большие сети, то их применение будет также требовать больших вычислительных ресурсов.

Dropout – технология позволяющая решить обе этих задачи. Она предотвращает переобучение и позволяет эффективно комбинировать сети с различными архитектурами. Термин «dropout» означает удаление некоторых узлов в нейронных сетях.

Под удалением узлов понимается временное исключение входящих и исходящих связей нейрона. Выбор нейрона, который будет удален, случаен. В простейшем случае вероятность исключения любого узла одинакова и выбирается либо, исходя из конкретной задачи.

Применение dropout к нейронным сетям создает несколько «разреженных» сетей внутри одной. Каждая разряженная сеть содержит только оставшиеся после отсечения узлы. Сеть, состоящая из n нейронов, может образовать разреженных сетей. Эти сети имеют общие веса, поэтому общее число параметров будет или меньше. Для каждого представления каждого тренировочного случая, выбирается разреженная сеть и обучается. Таким образом, обучение нейронной сети с использование dropout можно свести к обучению набора из разряженных сетей с общими весами, где каждая сеть обучается гораздо реже, чем все в совокупности. При тестировании сети невозможно усреднить выход каждой сети из-за их экспоненциального количества. Но на практике хорошо работает другой метод усреднения. Идея в том, чтобы использовать единственную сеть без dropout. Веса данной сети являются сжатыми весами тренированных сетей. Если какой-то нейрон имел вероятность включения в сеть p – при обучении то, во время тестирования вес будет умножен на данную вероятность.

Использование dropout в качестве усреднения ведет к улучшению качества работы сети, уменьшению ошибки на широком спектре задач классификации по сравнению с другими методами регуляризации.

5.5 Перемешивание обучающих примеров

Так как данные в выборке могут быть определены изначально в некоторой определенной последовательности(например в выборке данных мужчин и женщин, данные могут быть отсортированы по полу), то получая данные, по порядку их следования, модель лучше обучается по последним данным.

5.6 Предотвращение паралича сети

Слишком большие значения весовых коэффициентов, вызывающие «паралич» нейронной сети или чрезмерное уменьшение шага обучения, увеличивающее время обучения до чрезмерно большого значения. Кроме того, при некоторых условиях обучение нейронной сети может превратиться в неустойчивый процесс, когда, выучивая одно, сеть при этом забывает другое

Способы избежать паралича сети:

1. Уменьшить размер шага градиента, соответственно увеличится время обучения.

2.В слоях с большими весами отказаться от зависимости

|  |  |  |
| --- | --- | --- |
|  |  | (5.1) |

3.Эвристические правила для предотвращения роста весов.

6 АРХИТЕКТУРА СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ В ДАННОЙ РАБОТЕ

6.1 Общая струтктура

1. input\_shape=( 3, 100, 100) – входные данные
2. conv1\_num\_filters=32, conv1\_filter\_size=(3, 3) – 32 фильтра, размерности 3x3
3. pool1\_pool\_size=(2, 2) – pooling размерностью 2x2 с шагом 2
4. dropout1\_p=0.1 – исключение 10% весов
5. conv2\_num\_filters=64, conv2\_filter\_size=(2, 2) – 64 фильтра, размерности 2x2
6. pool2\_pool\_size=(2, 2) – pooling размерностью 2x2 с шагом 2
7. dropout2\_p=0.2 – исключение 20% весов
8. conv3\_num\_filters=128, conv3\_filter\_size=(2, 2) – 128 фильтров, размерности 2x2
9. pool3\_pool\_size=(2, 2) – pooling размерностью 2x2 с шагом 2
10. dropout3\_p=0.3 – исключение 30% весов
11. hidden4\_num\_units=500 – 500 нейронов в полносвязном слое персепртрона
12. dropout4\_p=0.5 – исключение 50% весов
13. hidden5\_num\_units=500 – 500 нейронов в полносвязном слое персепртрона

6.2 Подробности архитектуры

(input) – входное изображение в формата RGB(3 канала), разрешения 100x100

(conv, pool, dropout) – трижды повторяется, для вычленения признаков изображения, с слоями dropout, для предотвращения переобучения.

(hidden, hidden) – обычная полносвязная сеть, для обобщения результатов, на основании признаков, выделенных в сверточной части этой нейронной сети.

(out) – выходные 8 точек, которые мы предсказываем, вида (x, y)

Активационная функция между слоями ReLU: max(0, x)

6.3 Количество выходов, при заданной архитектуре

input (3, 100, 100) produces 30000 outputs

conv1 (32, 98, 98) produces 307328 outputs

pool1 (32, 49, 49) produces 76832 outputs

conv2 (64, 48, 48) produces 147456 outputs

pool2 (64, 24, 24) produces 36864 outputs

conv3 (128, 23, 23) produces 67712 outputs

pool3 (128, 12, 12) produces 18432 outputs

hidden4 (500) produces 500 outputs

hidden5 (500) produces 500 outputs

output (16) produces 16 outputs

6.4 Количество настраиваемых весов, при заданной архитектуре

1. 32 фильтра, размерности 3x3 – 288 весов
2. 64 фильтра, размерности 2x2 – 256 весов
3. 128 фильтров, размерности 2x2 – 512 весов
4. Результатом сверточной части сети 18432 выход, связь с 500 нейронами полносвязного слоя обеспечивают 9216000 весов
5. Связь между двумя полновязными слоями с 500 нейронов в каждом – 500\*500=250000 нейронов
6. Связь с выходным слоем с 16 нейронами – 8000 весов

Итого 9475056 весов

1. РЕЗУЛЬТАТЫ

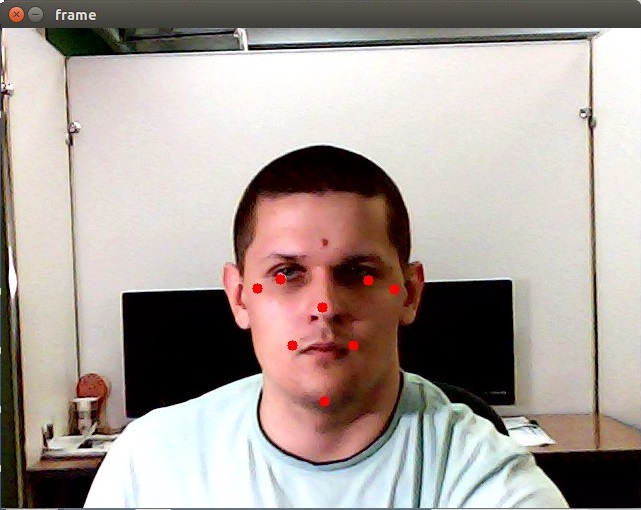


Рисунок 7.1

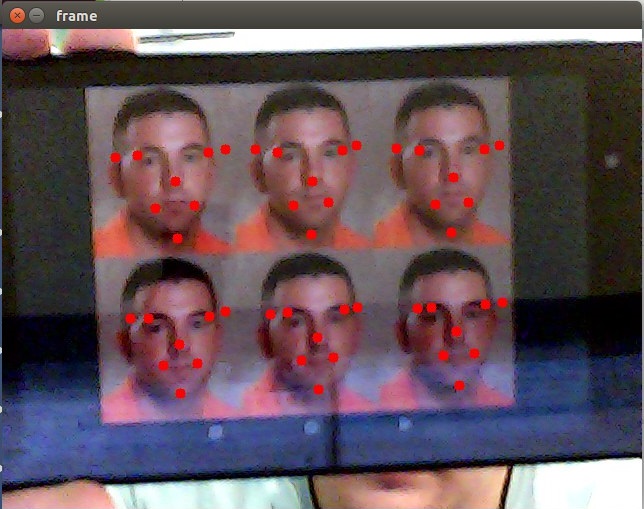


Рисунок 7.2

ЗАКЛЮЧЕНИЕ

Разработан и протестирован комплекс алгоритмов для распознавания точек лица человека.

Был использован достаточно эффективный метод Виолы-Джонса, который дал приемлемый результат. Обучение признаков Хаара не производилось, так как это весьма длительный процесс и сети Интернет есть уже обученные признаки. Выполнялось их использование.

Произведен подбор архитектуры нейронной сети, процесс подготовки данных: сбор обучающей выборки – порядка 2000 изображений различных людей. Выполнены манипуляции по преподготовке данных для обучения – оказалось, что для обучения нейронной сети достаточно размерности изображений 100х100, соответственно были отмасштабированы все изображения.

Написана программа по забору изображений с веб камеры с помощью OpenCV.

Итого был реализован программный комплекс состоящий из пяти этапов:

1. С помощью OpenCV получаем изображение с веб камеры.
2. Методом Виолы-Джонса детектируем лица на полученном изображении
3. Приводим детектированные участки изображений с лицами к размерности 100х100
4. Подаем на вход нейронной сети полученные изображения и получаем от нее детектированные точки
5. Отрисовываем полученные точки на исходном изображении и выводим пользователю полученный результат

Тест полученного решения проводился в режиме реального времени. Установлено, что приемлемый уровень качества возможен при присутствии перед камерой количества лиц не более шести. При шести лицах в кадре происходит обработка 7 кадров в секунду. При одном лице – 25 кадров в секунду.

Возможно существенное улучшение работы алгоритма за счет следующих подходов:

1. Уменьшение “глубины” нейронной сети без потери качества.
2. Распараллеливание предсказания точек лица, по областям определенным методом Виолы-Джонса
3. Использование более качественных детекторов лиц, чем метод Виолы-Джонса.
4. Расширение обучающей выборки для получения лучшего результата.
5. Использование языков более низкого уровня для предсказания по предобученной модели определенной архитектуры

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Hubel, D. H.; Wiesel, T. N., Receptive fields and functional architecture of monkey striate cortex // The Journal of Physiology 195 (1) - 1968-03-01 - 215–243.
2. LeCun, Yann; Léon Bottou; Yoshua Bengio; Patrick Haffner, Gradient-based learning applied to document recognition// PROC OF THE IEEE November - 1998.
3. Viola P., Jones M.J., Rapid object detection using a boosted cascade of simple features // IEEE Conf. on Computer Vision and Pattern Recognition. – Kauai, Hawaii, USA, 2001. – V. 1. – P. 511–518. 2.
4. Viola P., Jones M.J. Robust realtime face detection // International Journal of Computer Vision. – 2004. – V. 57. – № 2. – P. 137–154.
5. Дата обновления: 01.05.2016. URL: <http://www.deeplearningbook.org/> (дата обращения 15.05.2016)
6. Дата обновления: 22.11.2006. URL: <http://cogprints.org/5869/1/cnn_tutorial.pdf> (дата обращения 15.05.2016)
7. Дата обновления: 04.02.2016. URL: <http://cs231n.github.io/> (дата обращения 15.05.2016)
8. Дата обновления: 12.12.2015. URL: <http://neuralnetworksanddeeplearning.com/chap6.html> (дата обращения 15.05.2016)
9. Дата обновления: 04.03.2016. URL: <http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork/> (дата обращения 15.05.2016)
10. Дата обновления: 24.02.2014. URL: <http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/> (дата обращения 15.05.2016)
11. Дата обновления: URL: 07.11.2015. <http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/> (дата обращения 15.05.2016)
12. Дата обновления: URL: 09.05.2016. <http://mechanoid.kiev.ua/ml-lenet.html> (дата обращения 15.05.2016)
13. Дата обновления: URL: 12.02.2016. http://mechanoid.kiev.ua/neural-net-backprop.html (дата обращения 15.05.2016)
14. Дата обновления: URL: 21.04.2016. http://mechanoid.kiev.ua/neural-net-backprop2.html (дата обращения 15.05.2016)
15. Дата обновления: URL: 11.03.2015. http://oxozle.com/2015/04/11/metod-raspoznavaniya-lic-violy-dzhonsa-viola-jones (дата обращения 15.05.2016)

ПРИЛОЖЕНИЕ

Текст программы на языке python

import sys

sys.path.append("/home/lao/release/lib/")

import os

import cv2

from PIL import Image, ImageFilter

import glob

import numpy as np

import pandas as pd

from pandas.io.parsers import read\_csv

from sklearn.utils import shuffle

import lasagne

import matplotlib.pyplot as plt

from sklearn.metrics import mean\_squared\_error

from math import sqrt

from lasagne import layers

from lasagne.updates import nesterov\_momentum, adagrad

from nolearn.lasagne import NeuralNet

import cPickle as pickle

import theano

from IPython.display import display

from nolearn.lasagne import BatchIterator

from PIL import ImageDraw, ImageFont

import time

FTRAIN = '/home/lao/detectKeypoint/dataHelenScale/train/'

FTEST = '/home/lao/detectKeypoint/dataHelenScale/test/'

class AdjustVariable(object):

def \_\_init\_\_(self, name, start=0.03, stop=0.001):

self.name = name

self.start, self.stop = start, stop

self.ls = None

def \_\_call\_\_(self, nn, train\_history):

if self.ls is None:

self.ls = np.linspace(self.start, self.stop, nn.max\_epochs)

epoch = train\_history[-1]['epoch']

new\_value = float32(self.ls[epoch - 1])

getattr(nn, self.name).set\_value(new\_value)

def float32(k):

return np.cast['float32'](k)

class EarlyStopping(object):

def \_\_init\_\_(self, patience=100):

self.patience = patience

self.best\_valid = np.inf

self.best\_valid\_epoch = 0

self.best\_weights = None

def \_\_call\_\_(self, nn, train\_history):

current\_valid = train\_history[-1]['valid\_loss']

current\_epoch = train\_history[-1]['epoch']

if current\_valid < self.best\_valid:

self.best\_valid = current\_valid

self.best\_valid\_epoch = current\_epoch

self.best\_weights = nn.get\_all\_params\_values()

elif self.best\_valid\_epoch + self.patience < current\_epoch:

print("Early stopping.")

print("Best valid loss was {:.6f} at epoch {}.".format(

self.best\_valid, self.best\_valid\_epoch))

nn.load\_params\_from(self.best\_weights)

raise StopIteration()

def LoadNet():

with open('net1.pickle', 'rb') as f:

return pickle.load(f)

pointInSample = [0, 8, 16, 36, 45, 30, 48, 54]

net1 = NeuralNet(

layers=[

('input', layers.InputLayer),

('conv1', layers.Conv2DLayer),

('pool1', layers.MaxPool2DLayer),

('dropout1', layers.DropoutLayer), # !

('conv2', layers.Conv2DLayer),

('pool2', layers.MaxPool2DLayer),

('dropout2', layers.DropoutLayer), # !

('conv3', layers.Conv2DLayer),

('pool3', layers.MaxPool2DLayer),

('dropout3', layers.DropoutLayer), # !

('hidden4', layers.DenseLayer),

('dropout4', layers.DropoutLayer), # !

('hidden5', layers.DenseLayer),

('output', layers.DenseLayer),

],

input\_shape=(None, 3, 100, 100),

conv1\_num\_filters=32, conv1\_filter\_size=(3, 3), pool1\_pool\_size=(2, 2),

dropout1\_p=0.1, # !

conv2\_num\_filters=64, conv2\_filter\_size=(2, 2), pool2\_pool\_size=(2, 2),

dropout2\_p=0.2, # !

conv3\_num\_filters=128, conv3\_filter\_size=(2, 2), pool3\_pool\_size=(2, 2),

dropout3\_p=0.3, # !

hidden4\_num\_units=500,

dropout4\_p=0.5, # !

hidden5\_num\_units=500,

output\_num\_units=len(pointInSample) \* 2,

output\_nonlinearity=None,

update=nesterov\_momentum,

update\_learning\_rate=theano.shared(float32(0.03)),

update\_momentum=theano.shared(float32(0.9)),

on\_epoch\_finished=[

AdjustVariable('update\_learning\_rate', start=0.03, stop=0.0001),

AdjustVariable('update\_momentum', start=0.9, stop=0.999),

EarlyStopping(patience=100),

],

batch\_iterator\_train=BatchIterator(batch\_size=10),

regression=True,

max\_epochs=800,

verbose=1,

eval\_size=0.1

)

net1 = LoadNet()

def Predict(image):

imResize = cv2.resize(image,(width, height))

imResize = np.divide(imResize, 255.)

for i in range(0, 3):

for j in range(0, width):

for k in range(0, height):

imNew[0, i, j, k] = imResize[j, k, i]

pred = net1.predict(imNew)

pred = np.subtract(pred, -1)

pred = np.multiply(pred, 0.5)

pred = pred.reshape(len(pred[0]) / 2, 2)

return pred

cap = cv2.VideoCapture(0)

cascPath = "/home/lao/detectKeypoint/haarcascade\_frontalface\_alt.xml"

faceCascade = cv2.CascadeClassifier(cascPath)

# Capture frame-by-frame

from multiprocessing import pool

from multiprocessing.dummy import Pool as TreadPool

# Our operations on the frame come here

width = 100

height = 100

originalWidth = 640

originalHeight = 480

imNew = np.zeros((1, 3, width, height))

imNew = imNew.astype(np.float16)

while(True):

ret, frame = cap.read()

im = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

####rect

faces = faceCascade.detectMultiScale(

frame,

scaleFactor=1.1,

minNeighbors=5,

minSize=(30, 30),

flags=cv2.CASCADE\_SCALE\_IMAGE

)

# Draw a rectangle around the faces

for (x, y, w, h) in faces:

x1 = int(x - 0.2 \* w)

x1 = 0 if x1 < 0 else x1

x2 = int(x + 1.2 \* w)

x2 = originalWidth if x2 > originalWidth - 1 else x2

y1 = int(y)

y1 = 0 if y1 < 0 else y1

y2 = int(y + 1.3 \* h)

y2 = originalHeight if y2 > originalHeight - 1 else y2

#cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

imRect = im[y1 : y2, x1 : x2]

pred = Predict(imRect)

for i in range(0, len(pointInSample)):

pass

cv2.circle(frame, (int(pred[i][0] \* (x2 - x1) + x1 ), int(pred[i][1] \* (y2 - y1) + y1 )),

5, (0,0,255), -1)

#end rect

frame = np.fliplr(frame)

cv2.imshow('frame', frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

# When everything done, release the capture

cap.release()

cv2.destroyAllWindows()