МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)
01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)
Системное программирование и компьютерные технологии
(направленность (профиль)/специализация

БАКАЛАВРСКАЯ РАБОТА

на тему Разработка алгоритма локализации и классификации объектов на изображении

Студент Д.Д. Фролов _____
(И.О. Фамилия) (личная подпись)
Руководитель Т.Г. Султанов _____
(И.О. Фамилия) (личная подпись)
Консультанты Н.В. Андрюхина _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

АННОТАЦИЯ

Тема: «Разработка алгоритма локализации и классификации объектов на изображении».

Целью выпускной квалификационной работы (ВКР) является исследование методов локализации и классификации объектов на изображении и разработка универсального алгоритма для решения практико-ориентированных задач.

Объектом ВКР является распознавание дорожных знаков.

Предмет исследования – система распознавания дорожных знаков на изображении с использованием методов машинного обучения.

В данной выпускной квалификационной работе исследуются наиболее актуальные и популярные методы локализации и классификации объектов на изображении.

В работе представлен универсальный разработанный подход (алгоритм) к решению задачи локализации и классификации объектов на изображении. Разработана программная реализация и проведено тестирование основных модулей.

Структура ВКР состоит из введения, двух глав, заключения, списка литературы.

Во введении формулируется цель, ставятся задачи и описывается актуальность исследуемой области и практической задачи.

В первой главе рассматривается задача локализации и классификации объектов на изображении, описываются актуальные методы и приводится обоснование выбранного подхода.

Во второй главе разрабатываются основные программные модули, проектируется универсальный алгоритм локализации и классификации объектов, реализуется программное решение и проводится его тестирование.

Бакалаврская работа состоит из пояснительной записки, включая 28 рисунков, 14 источников используемой литературы и 1 приложения.

ABSTRACT

The title of the graduation work is "Development of the algorithm for localization and classification of objects in the image."

The aim of the work is to study the methods of localization of objects in the image and develop a universal algorithm for solving practical problems.

The object of the graduation work is the recognition and localization of road signs.

The subject of the graduation work is system of localization and recognition of road signs on the image using machine learning methods.

The graduation work describes in details main methods of localization and classification of objects in the image. We give full coverage to components of systems of recognition and localization of objects and compare and identify the most accurate and versatile solutions.

The graduation work presents a universal algorithm for solving the problem of localization and classification of objects in the image and describes the software implementation of the algorithm. We also report the results of the software implementation tests.

The graduation work consists of an introduction, two chapters, a conclusion, a list of references.

In the introduction, the goal is formulated, the tasks and the relevance of the studied area are set.

The first chapter deals with the problem of localizing and classifying objects in an image, describes current methods and establishes objectives of the approach chosen.

In the second chapter, basic software modules are developed, a universal object localization algorithm is designed and software solution is tested.

The graduation work consists of an explanatory note on 48 pages, including 28 figures, the list of 14 references including 5 foreign sources and 1 appendix.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 6
Глава 1 АНАЛИЗ МЕТОДОВ ЛОКАЛИЗАЦИИ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ 8
1.1 Выбор актуальной практико-ориентированной задачи локализации объектов на изображении 8
1.2 Анализ задачи локализации объектов на изображении и определение ключевых этапов 8
1.3 Метод Виолы-Джонса 9
1.4 Метод опорных векторов с использованием гистограмм направленных градиентов 11
1.4.1 Определение понятия гистограммы направленных компонентов 11
1.4.2 Теоретическое описание метода опорных векторов 14
1.5. Свёрточные нейронные сети 17
1.5.1 Общие понятия нейронных сетей 17
1.5.3 Функция активации и основы обучения нейронной сети 18
1.5.4 Архитектура свёрточной нейронной сети 20
1.6 Обоснование выбора методов 23
Глава 2 АЛГОРИТМ ЛОКАЛИЗАЦИИ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ.. 27
2.1 Локализация дорожных знаков 27
2.1.1 Формирование и подготовка базы дорожных знаков 28
2.1.2 Обучение HOG&SVM дескрипторов 30
2.2 Классификация дорожных знаков 32
2.2.1 Подготовка обучающей базы данных 32
2.2.2 Проектирование и обучение нейронной сети 33
2.3 Разработка и проектирование алгоритма комплексного распознавания дорожных знаков 37
2.4 Тестирование разработанного программного решения 41
2.4.1 Тестирование и оценка точности модуля локализации 41
2.4.2 Тестирование классифицирующей свёрточной нейронной сети 44

ЗАКЛЮЧЕНИЕ 46
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 47
ПРИЛОЖЕНИЕ А. Листинг кода программной реализации 49

ВВЕДЕНИЕ

Задачи локализации и классификации образов часто встречаются и решаются в повседневной жизни людей. Проблема распознавания образов очень важна в наше время, так как существенное количество сигналов, поступающих единовременно, значительно осложняют жизнь современного человека. Данную проблему обычно называют «информационной перегрузкой».

Во многих междисциплинарных исследованиях также встречается проблема распознавание объектов. В частности, распознавание образов лежит в основе комплексных реализаций систем искусственного интеллекта, а реализация методов и систем распознавания изображений привлекает сегодня большое внимание.

Локализация объектов на изображении и их классификация является огромной проблемой в современной науке, так как нет какого-то универсального решения для всех задач. Однако, существует большое количество реализаций, которые решают данную проблему в конкретном случае.

Автоматизация локализации объектов особенно актуальна в областях жизни человека, где имеется большой поток поступающих сигналов, и не своевременная их обработка может нанести вред здоровью человека.

Одной из актуальных проблем сегодня, является автоматизация управления движением автомобиля и разработка систем помощи водителю, реагирующих на различные сигналы.

Дорожные знаки являются неотъемлемой частью современной дорожной инфраструктуры.

Они предоставляют критически важную информацию и убедительные рекомендации для участников дорожного движения, что, в свою очередь, требует от последних корректировки своего поведения при вождении, с тем чтобы они соблюдали все действующие в настоящее время правила дорожного движения. Без таких полезных рекомендаций мы, скорее всего,

столкнемся с большим количеством ДТП, поскольку водители не получат критических уведомлений о том, насколько быстро они могут безопасно ехать, или не будут проинформированы о дорожных работах, крутых поворотах или школьных переходах. В наше время около 1,3 млн. человек умирают на дорогах каждый год.

Это число было бы намного выше без дорожных знаков, поэтому разработка алгоритмов и систем их обнаружения их обнаружения сегодня особенно актуальна.

Цель выпускной квалификационной работы (ВКР) - исследование методов локализации и классификации объектов на изображении и разработка универсального алгоритма для решения практико-ориентированных задач.

Объект ВКР - распознавание дорожных знаков.

Предмет ВКР: система распознавания дорожных знаков на изображении с использованием методов машинного обучения.

Основные задачи ВКР:

1. Исследовать методы локализации и классификации объектов на изображении.
2. Разработать алгоритм, решающий практико-ориентированную задачу распознавания объекта или объектов на изображении;
3. Протестировать разработанное решение и провести анализ полученных результатов.

Глава 1 АНАЛИЗ МЕТОДОВ ЛОКАЛИЗАЦИИ И КЛАССИФИКАЦИИ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ

1.1 Выбор актуальной практико-ориентированной задачи

Одной из актуальных и интересных задач является разработка системы для функционирования автономного транспортного средства и системы помощи водителя. Некоторые крупные автомобильные компании сегодня активно внедряют подобные системы в свои автомобили.

Задача распознавания дорожных знаков достаточно нетривиальна и не имеет универсальных алгоритмов решения, однако, есть те методы, которые для данной задачи могут стать наиболее подходящими. Для того, чтобы определиться с наиболее эффективными вариантами, необходимо провести теоретический анализ всех наиболее актуальных вариантов.

1.2 Анализ задачи распознавания объектов на изображении и определение ключевых этапов

Задачу комплексного распознавания объектов на изображении обычно разделяют на два этапа:

1. Локализация;
2. Классификация.

На этапе локализации выявляется положение искомого объекта на изображении, а на этапе классификации определяется его принадлежность к некоторому классу. После корректной совместной работы системы на этих этапах можно точно сказать, присутствует ли искомый объект на изображении и к какому подмножеству из искомых он относится. Задача может быть сведена лишь к одному этапу локализации, если нас интересует только нахождение объекта, но не его тип. Примером такой задачи в рамках автоматизации дорожного движения может быть задача локализации автомобиля, без его марки, так как это не имеет значения. Однако, в случае с задачей локализации дорожных знаков такой подход не подойдет, так как нам необходимо не только понимать, что знак присутствует на дороге, но и

реагировать на то, к какому типу он относится. Каждый из этих этапов важен и имеет свои особенности.

Способы и методы локализации и классификации объектов на изображении условно можно разделить на три основные категории:

1. Методы, основанные на цветовых признаках локализуемого изображения;
2. Методы, основанные на выявлении формы локализуемого изображения и его характерных особенностях;
3. Методы, основанные на алгоритмах машинного обучения.

Для определения наиболее оптимального метода необходимо рассмотреть наиболее популярные и эффективные способы и методы из каждой описанной категории.

1.3 Метод Виолы-Джонса

Метод Виолы-Джонса - это алгоритм, который позволяет в реальном времени обнаруживать объекты на изображении. Он был представлен в 2001 году Полом Виолой и Майклом Джонсом. Хотя алгоритм может идентифицировать разные категории изображений, основной задачей его создания является обнаружение лиц. Существует много реализаций этого метода, в том числе в библиотеке компьютерного зрения OpenCV [8].

Исследование Пола Виолы и Майкла Джонса имеет огромное значение в области компьютерного зрения. Хотя этот метод был первоначально разработан для решения задачи обнаружения лиц, различные исследователи успешно применили его для распознавания и других объектов.

Метод Виолы – Джонса - один из классических методов в задаче локализации объектов, однако во многом уступает современным вариантам по точности работы.

Признаки, используемые алгоритмом, опираются на суммирование пикселей из прямоугольных регионов. Сами признаки несколько напоминают

признаки Хаара, которые ранее также использовались для поиска объектов на изображениях.

Признаки Хаара - это признаки цифрового изображения, используемые в задаче распознавании образов. Признаки Хаара использовались в самом первом детекторе лиц, работающем в реальном времени [9].

Однако признаки, предложенные Виолой и Джонсом, несколько прямоугольных областей и несколько сложнее признаков Хаара. На рисунке 1 показаны четыре различных типа признаков.

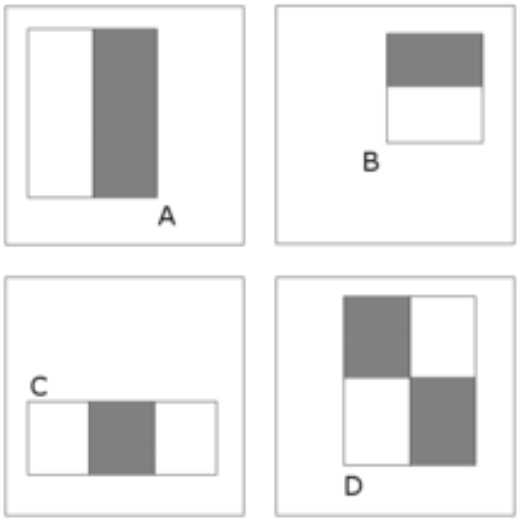

Рисунок 1 – Признаки Виолы-Джонса

Значение каждого признака рассчитывается как сумма пикселей в белом прямоугольнике, из которой вычитается сумма пикселей в черной области.

Прямоугольные признаки более примитивны, чем управляемые фильтры, и, хотя они чувствительны к вертикальным и горизонтальным особенностям изображения, результаты их работы более грубы и обобщены. Однако, когда изображение сохраняется в интегрированном формате (каждый пиксель изображения содержит сумму всех пикселей слева и выше), проверка прямоугольных элементов в определенной позиции выполняется в течение константного времени, что дает прирост в скорости, в сравнении с более точными вариантами.

Высокая скорость расчёта признака не компенсирует большое количество различных возможных признаков, поэтому в алгоритме Виолы-Джонса используется «бустинг» (вариация алгоритма обучения AdaBoost), как для выбора признаков, так и для настройки классификаторов.

В задаче локализации дорожных знаков метод Виолы-Джонса показал себя не самым лучшим образом и выдавал сравнительно большее количество ложных срабатываний, чем методы, используемые в разработке, поэтому было принято решение – отказаться от данного метода в пользу более точного.

1.4 Локализация объектов с помощью гистограмма направленных градиентов и метода опорных векторов

Для решения части исследуемой проблемы, а именно локализации потенциального дорожного знака в рамках данной работы был использован именно метод опорных векторов с использованием гистограмм направленных градиентов, так как он показал большую точность в сравнении с классическим методом Виолы-Джонса.

1.4.1 Определение понятия гистограммы направленных компонентов

Гистограмма направленных градиентов — набор особых точек, которые используются в задаче обработке изображений и иных задачах компьютерного зрения, с целью распознавания и локализации объектов.

Гистограмма направленных градиентов является переведенным понятием с английского Histogram of Oriented Gradients (HOG). Поэтому, когда в данной пояснительной записке упоминается аббревиатура HOG, имеется в виду гистограмма направленных градиентов.

HOG основана на подсчете количества направлений градиента в определенных областях изображения. Метод, реализуемый HOG схож с другими, но характерной особенностью является то, что при нахождении гистограммы значения равномерно вычисляется на плотной сетке

распределенных ячеек и использует нормализацию перекрывающегося локального контраста для увеличения точности [10].

Впервые гистограмма направленных градиентов была описана в июне 2005 года Биллом Триггсом и Навнитом Далалом в их работе на CVPR.

CVPR - это одна из главнейших ежегодных конференций по компьютерному зрению, проходящая обычно в Солт-Лейк-Сити, штат Юта, США.

Билл и Навнит в своей работе описали алгоритм нахождения пешеходов на статичных изображениях [19], но в последствии алгоритм был расширен для нахождения людей и различных других объектов в видеопотоке.

В своей работе Билл и Навнит предположили, что внешний вид и форма объекта на определенном участке изображения можно описать с использованием градиентов интенсивности (направлением интенсивности контуров). Для получения данных дескрипторов необходимо разделить изображение на маленькие связанные друг с другом ячейки и рассчитать HOG в каждой из них. Скомбинировав получившиеся диаграммы получается дескриптор (шаблон), который в дальнейшем можно использовать для локализации. Для получения наибольшей точности было принято решение использовать не одно изображение, а несколько изображений с общим объектом на каждом из них (тот, который необходимо локализовать), такое объединение принято называть блоком. Также для увеличения точности локальные гистограммы нормализуют по контрасту, что позволяет получить большую инвариантность к освещению и другим шумам.

Дескриптор HOG обладает рядом преимуществ, по сравнению с другими дескрипторами. Так как HOG работает локально (для каждой ячейки), то метод не зависит от геометрических и фотометрических преобразований объекта, и позволяет обнаружить его в любом случае (в любом месте), однако метод чувствителен к изменению ориентации объекта.

Во многих детекторах объектов первым шагом является нормализация по цвету, гамма-коррекция и иные маски, однако для дескриптора HOG в этом нет необходимости, так как последующая нормализация в ячейках приводит к тому же результату.

В общем виде реализация детектора HOG состоит из следующих этапов: вычисление градиента, группировка направлений, блоки дескрипторов, нормализация блоков, SVM-классификатор.

На первом шаге формирования детектора HOG в каждой ячейке рассчитывается значение градиентов. Наиболее распространённым методом расчета градиента является использование одномерной дифференцирующей маски в различных положениях. Для работы данного метода требуется произвести цветовую или яркостную фильтрацию с использованием следующих ядер: $-1, 0, 1$ и $-1, 0, 1\ ^{T}$.

На втором шаге вычисляются локальные гистограммы. Для каналов гистограммы направлений на основании каждого пикселя в ячейке проводится взвешенное голосование, основанное на значении градиентов. Ячейки могут быть как прямоугольной, так и круглой формы, каналы гистограммы распределяют равномерно либо от 0 до 180, либо от 0 до 360 градусов, в зависимости от того, какой градиент вычисляется.

Третьим шагом является формирование блоков дескрипторов. Для учета изменений яркости и контрастности градиенты необходимо нормировать, а именно объединить в более крупные связанные блоки. В общем виде, дескриптор HOG – вектор, состоящий из нормированных гистограмм ячеек. Как правило каждая ячейка входит более чем в один конечный дескриптор. В практике применяются два основных типа блока - R-HOG (прямоугольные) и C-HOG (круглые), однако из них чаще всего применяются именно квадратные. Блоки R-HOG – квадратные сетки, которые можно охарактеризовать тремя ключевыми параметрами: количеством ячеек, количеством пикселей на ячейку, количеством каналов на

гистограмму. В своей работе Далал и Триггс отметили оптимальными блоки размером 16 на 16, ячейки 8 на 8, при 9 каналах на гистограмму.

На четвертом шаге производится нормализация блоков. Далал и Триггс в своей работе исследовали четыре следующих способа нормализации:

- L2-норма: $f = \frac{v}{\sqrt{v_2^2 + e^2}}$;
- L2-hys: L2-норма ограничивается сверху (значения v > 0,2, приравниваются 0,2) и перенормируются;
- L1-норма: $f = \frac{v}{v_1 + e}$;
- корень из L1-нормы: $f = \sqrt{\frac{v}{v_1 + e}}$;

где v — ненормированный вектор, содержащий все гистограммы блока, v_k — его k-норма при $k = 1,2$ и e — бесконечно малая константа.

Опытным путем Далалом и Триггсом было установлено, что L1-норма дает менее надежный результат, чем остальные три, в то время как L2-норма, L2-hys и корень из L1-нормы приводят приблизительно к одинаковым хорошим результатам.

Конечным шагом в локализации объектов с использованием HOG является классификация полученных дескрипторов при помощи алгоритмов машинного обучения. Наиболее часто для данной задачи в связке с HOG используют метод опорных векторов.

1.4.2 Теоретическое описание метода опорных векторов

Метод опорных векторов — набор алгоритмов обучения с учителем, использующихся для решения задач классификации и регрессионного анализа [2].

Метод опорных векторов является переведенным понятием с английского support vector machine (SVM). Поэтому, когда в данной пояснительной записке упоминается аббревиатура SVM, имеется в виду метод опорных векторов.

Принцип работы метода удобно рассмотреть на примере. Пусть даны точки на плоскости, которые разделены на два класса A и B.

Между классами проводится разделяющая прямая (рисунок 2). Далее, новые точки классифицируются на основании их положения относительно разделяющей прямой.

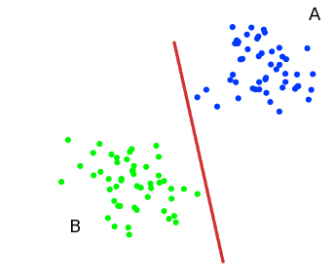

Рисунок 2 – Графическая иллюстрация SVM для двумерной задачи

Более сложные примеры, чем и является классификация HOG дескрипторов, имеют размерность большую чем два, поэтому простая прямая уже не будет разделять множества на классы. Поэтому для многомерных задач рассматривается разделение гиперплоскостями.

Гиперплоскость – пространство, размерность которого на единицу меньше исходного пространства, в котором оно находится. Например для трехмерной задачи R^3 гиперплоскостью будет обычная двухмерная плоскость.

Для разделения классов можно провести несколько различных гиперплоскостей, удовлетворяющих условию, однако, с точки зрения классификации, лучше выбрать прямую с максимальным расстоянием до каждого класса. Изобразим пример для выбора разделяющей прямой на рисунке 3.

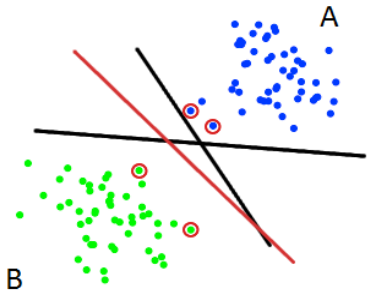

Рисунок 3 – Графическая иллюстрация поиска опорных векторов и разделяющей гиперплоскости

Вектора, располагающиеся ближе всех к разделяющей гиперплоскости, называют опорными векторами, на основе чего и было дано название методу. Опорные вектора на рисунке 3 обведены.

Разберем математическую модель метода более детально. Пусть дана следующая обучающая выборка $x_1, y_1 , \dots , x_m, y_m , x_i \in R^n, y_i \in -1, 1$, где каждый x_i – вещественный n-мерный вектор.

На основе обучающей выборки SVM строит классифицирующую функцию: $F\ x\ = sign\ \ w, x\ + b$, где w, x – скалярное произведение w и x, w – перпендикуляр к разделяющей гиперплоскости, b – вспомогательный параметр. Соответственно, если значение функции F(x) = 1, то объект попадает в один класс, а если F(x) = -1, то в другой. Гиперплоскость задается на основе F(x) в виде: $w, x\ + b = 0$ для некоторых w и b (рисунок 4) .

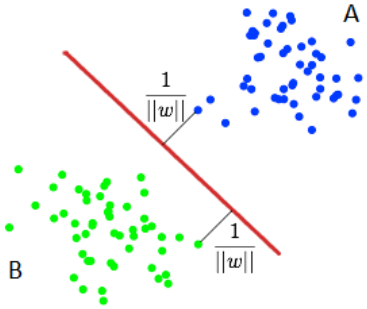

Рисунок 4 – Графическая иллюстрация поиска разделяющей гиперплоскости на основании математического обоснования

На следующем шаге подбираются такие значения w и b, при которых расстояние до каждого класса максимально. В общем виде данную задачу оптимизации можно записать следующим образом:

$$\|w\|^2 \to min,$$
$$y_i \left(\langle w, x_i \rangle + b\right) \geq 1, \qquad i = 1, \ldots, m.$$

Представленная выше задача оптимизации является типичной задачей квадратичного программирования и решается с помощью множителей Лагранжа.

На практике случаи, когда данные легко можно разделить гиперплоскостью встречаются довольно редко. Данная проблема называется линейной неразделимостью. При возникновении линейной неразделимости: все элементы обучающей выборки вкладываются в пространство более высокой размерности с помощью специального отображения, так, чтобы выборка в полученном пространстве была линейно разделима.

1.5. Свёрточные нейронные сети

1.5.1 Общие понятия нейронных сетей

Искусственная нейронная сеть (ИНС) - это парадигма обработки информации, вдохновленная тем, как биологические нервные системы, такие как мозг, обрабатывают информацию. Ключевым элементом этой парадигмы является новая структура системы обработки информации. Он состоит из большого количества сильно взаимосвязанных процессорных элементов (нейронов), работающих в унисон для решения конкретных проблем.

ИНС, как и люди, учатся на собственном примере. Для каждого конкретного приложения нейронные сети конфигурируются по-своему, так как распознавание образов или классификация данных, требуют различных структур и данных для обучения. Обучение в биологических системах включает в себя корректировку синаптических связей, существующих между нейронами. Это верно и для ИНС.

Простейшим элементом искусственной нейронной сети является нейрон.

Искусственный нейрон - это программный модуль или устройство с множеством входов и одним выходом. Нейрон имеет два режима работы: режим обучения и режим использования. В режиме обучения нейрон может быть обучен для определенных входных паттернов. В режиме использования, когда обученный входной шаблон обнаружен на входе, его связанный выход становится текущим выходом. Пример простого нейрона представлен на рисунке 5.

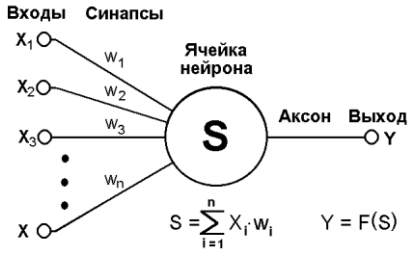

Рисунок 5 – Простой искусственный нейрон

На вход нейрона подаются данные либо в целом, либо в вещественном виде. Часто, при проектировании нейронной сети прибегают к нормализации данных. Синапсы (или веса) являются сутью работы нейронных сетей. На первой итерации входы (x1…xn) умножаются на соответствующие им веса (w1..wn). Задача нейрона – просуммировать произведение входов на веса, т о есть: $\sum_{i=1}^{n} x_i w_i$. После суммирования данные направляются на преобразователь, который использует функцию активации.

1.5.3 Функция активации и основы обучения нейронной сети

Функция активации (активационная функция, функция возбуждения) – функция, вычисляющая выходной сигнал искусственного нейрона. Функции активации используются для нормализации выходных данных.

Наиболее простой функцией активации является ступенчатая функция. Пример ступенчатой функции показан на рисунке 6. Однако, на практике

наиболее применимы её различные модификации, например, сигмоидальная функция, представленная на рисунке 7.

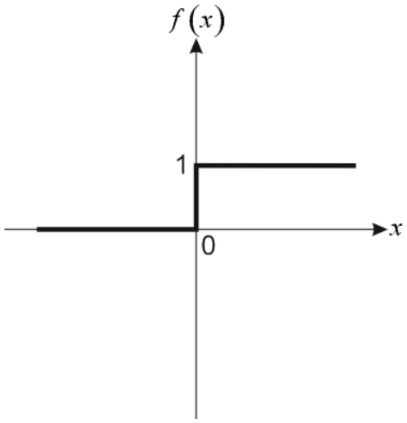

Рисунок 6 – Ступенчатая функция активации

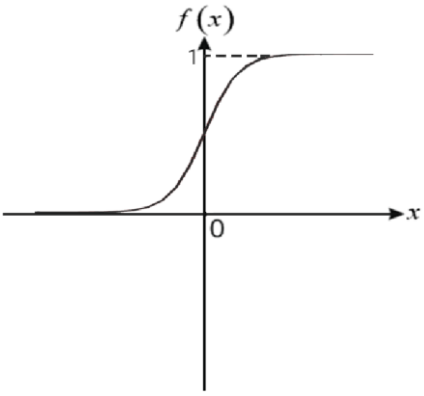

Рисунок 7 – Сигмоидальная функция активации

Обучение нейронной сети - это процесс, в котором параметры нейронной сети настраиваются посредством моделирования среды, в которую эта сеть встроена. Тип обучения определяется способом подстройки параметров. Базовым для многих остальных алгоритмом обучения нейронной сети является алгоритм дельта-правило.

Дельта-правило — метод обучения персептрона по принципу градиентного спуска по поверхности ошибки. Метод дельта-правила используется в других методах обучения нейронных сетей. Для дельта-формулы нам нужно знать сетевую ошибку. В большинстве случаев ошибка - это разница между правильным и неправильным ответом. Как правило, обучение прекращается, когда ошибка становится равной нулю или меньше указанного порогового значения.

Дельта-правило определяется следующими формулами:

$$error = t - y,$$

где t — правильный ответ, y — ответ сети.

Соответственно новый весовой коэффициент рассчитывается следующим образом:

$$w\ i+1\ = w + error * x_i.$$

Более сложные нейронные сети состоят из множества слоев и связей между нейронами, а также используют модифицированные алгоритмы обучения, основанные на идеях метода дельта-правила, например, метод обратного распространения ошибки.

Большую сложность вызывает и выбор правильной структуры нейронной сети для конкретной задачи. В современных разработках при упоминании задачи распознавания и классификации объектов все чаще стали использовать сверточные нейронные сети.

1.5.4 Архитектура свёрточной нейронной сети

Сверточные нейронные сети (CNN) - это глубокие искусственные нейронные сети, которые используются, главным образом, для классификации изображений. CNN также могут кластеризовать изображения по сходству и выполнять распознавание объектов в сценах (локализовать). CNN могут идентифицировать лица, уличные знаки, опухоли и многие другие аспекты визуальных данных, в зависимости от обучающей выборки, которая была задана.

Выделяют следующие преимущества сверточной нейронной сети:

- меньшее количество параметров (в сравнении с полносвязанной сетью);
- устойчивость к перемене расположения и поворотам распознаваемого объекта;

Однако CNN обладает и рядом недостатков:

- долгое время обучения

- необходимость большой обучающей выборки

Однако, не смотря на недостатки, на сегодняшний день архитектура сверточной нейронной сети является наиболее пригодной для решения задачи классификации изображений.

Сверточная нейронная сеть в общем виде состоит из следующих слоев и их чередования:

- слои свертки;
- пулинговые слои (слои подвыборки);
- полносвязные слои.

Сверточные нейронные сети имеют различное количество слоев и часто используют различные функции активации. Пример возможной структуры сверточной нейронной сети представлен на рисунке 8.

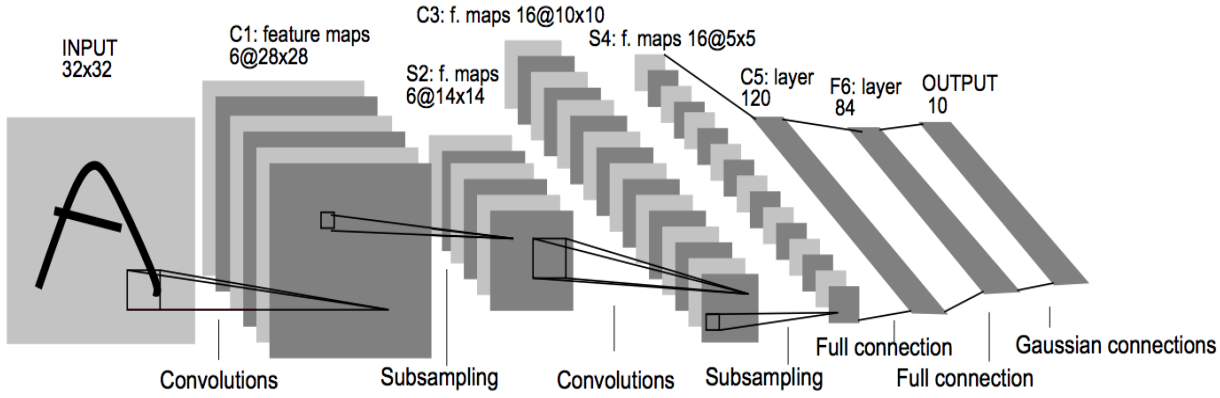

Рисунок 8 – Пример структуры сверточной нейронной сети (LeNet-5)

На сверточном слое к исходному изображению применяется операция свертки. Использование алгоритма свертки позволяет находить объект на изображении не зависимо от его положения или углового смещения. Исходное изображение разбивается на несколько пересекающихся связанных фрагментов.

Каждый полученный фрагмент по очереди подается на вход нейронной сети, которая их обрабатывает с использованием одинаковых весов, формирующих ядро свертки.

На каждом фрагменте вычисляется значение интенсивности каждого пикселя, которое в последствии умножается на соответствующий элемент ядра свертки, суммируется и подается на вход функции активации нейрона сверточного слоя.

Результатом работы алгоритма свертки является уменьшенное изображение, которое содержит наиболее «выдающиеся» участки родителя. Пример работы алгоритма свертки представлен на рисунке 9.

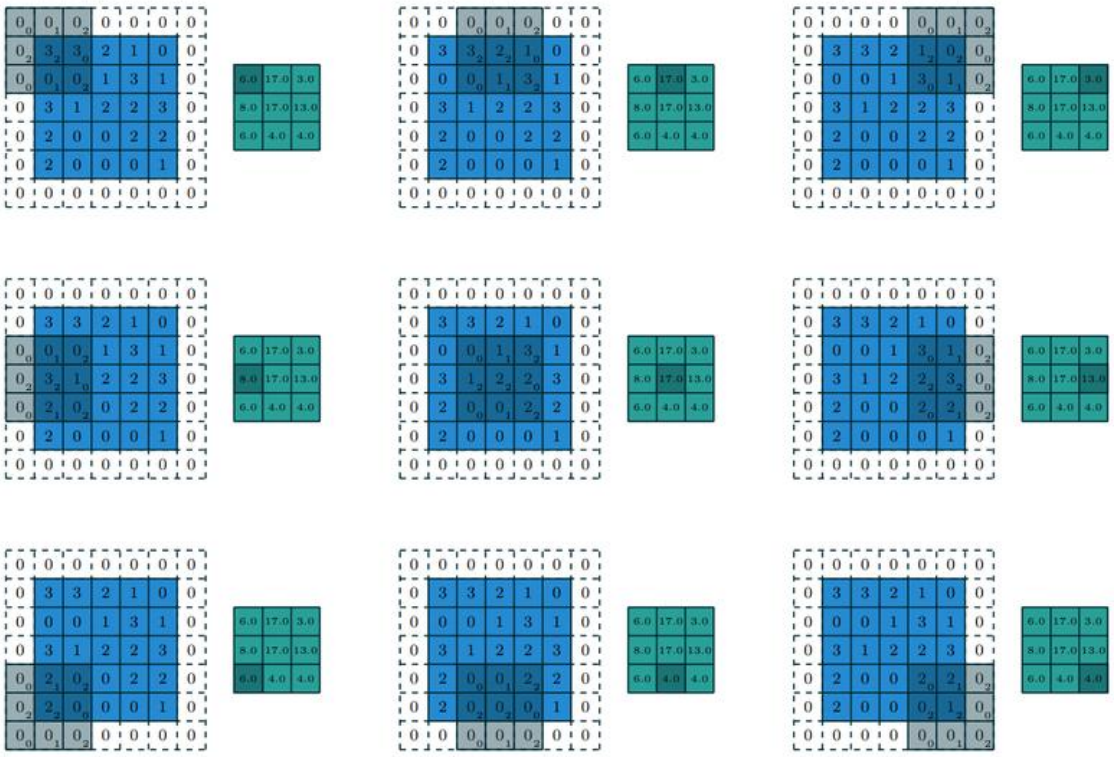

Рисунок 9 – Пример свертки изображения

Следующим слоем сети является пулинговый (субдискретизирующий) слой или слой подвыборки. На данном слое производится уменьшение размерности сформированных карт признаков. Данное действие называется операцией субдискретизации (пулинга). В некоторых редких случаях вместо операции субдискретизации используют среднее значение.

Слои подвыборки уменьшают размер данных, ускоряют дальнейшие вычисления и делают сеть более инвариантной к масштабу входного изображения. Пример работы слоя подвыборки представлен на рисунке 10.

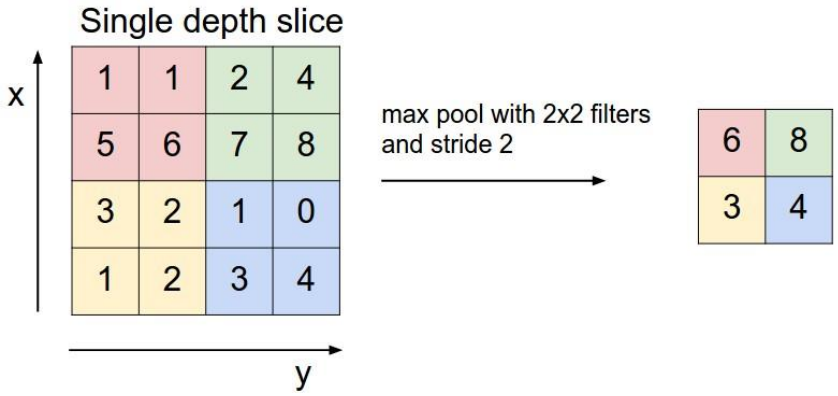

Рисунок 10 – Работа операции подвыборки

После нескольких этапов свертки и пулинга исходное изображение преобразуется в более абстрактную карту признаков.

Как правило на каждом следующем слое увеличивается количество карт признаков и уменьшается размерность изображения. Данные со всех карт признаков объединяются и подаются на полносвязную нейронную сеть (может состоять из одного или более полносвязных слоев).

Полносвязный слой – обыкновенный персептрон, на вход которому подаются уже подготовленные карты признаков. Полносвязный слой уже не обладает пространственной структурой исходного изображения и обладает меньшей размерностью.

Наиболее популярные способы обучения CNN — метод обратного распространения ошибки и его модификации.

1.6 Обоснование выбора методов

Как было отмечено в разделе 1.2 данной главы, задачу распознавания можно разбить на два основных этапа – локализация и классификация. Соответственно, необходимо реализовывать и рассматривать методы, наиболее актуальные для конкретного этапа.

Хоть задачу классификации также можно решить с помощью HOG, данный способ не будет оптимальным, так как придется обучать дескриптор для каждого конкретного класса, что в конечном итоге приведет к уменьшению производительности разработанного решения.

Сравнительный анализ HOG с другими методами локализации, такими как вейвлеты Хаара (использующиеся в методе Виолы-Джонса) и контексты формы представлен в оригинальной работе [19], где Далал и Триггс проводили тестирование своего алгоритма на двух наборах данных с изображениями людей, а именно MIT [5] и IRNA [7].

Сравнительный анализ методов локализации представлен в таблице 1, а её графическое представление на рисунке 11.

Таблица 1 – сравнительный анализ методов локализации изображений

Дескриптор	**Набор данных**	**Доля пропущенных изображений**	**Доля ошибок первого рода**
HOG	MIT	≈0	10^{-4}
HOG	INRIA	0.1	10^{-4}
Обобщенные вейвлеты Хаара	MIT	0.01	10^{-4}
Обобщенные вейвлеты Хаара	INRIA	0.3	10^{-4}
PCA-SIFT, контексты формы	MIT	0.1	10^{-4}
PCA-SIFT, контексты формы	INRIA	0.5	10^{-4}

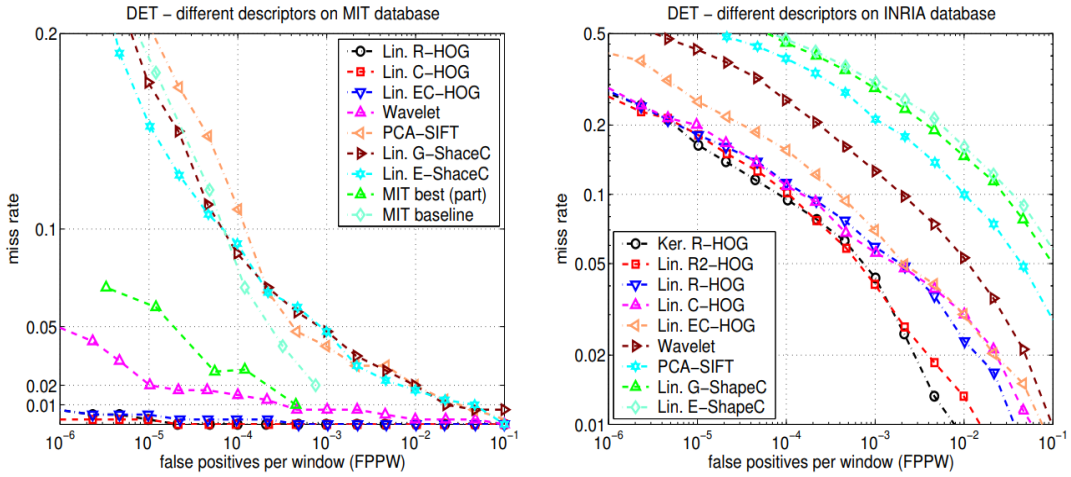

Рисунок 11 – График сравнительного анализа методов локализации

Как видно из таблицы 1 и рисунка 11, HOG классификаторы показали сравнительно лучшие результаты, даже в задаче локализации более сложной формы (относительно формы дорожного знака), как человек, поэтому метод основанный на гистограммах направленных градиентов является оптимальным для данной задачи.

Еще одним преимуществом HOG&SVM дескрипторов является популярность данного подхода и наличие инструментов и реализаций, которые упрощают разработку, например, реализация HOG&SVM дескрипторов в библиотеке Dlib.

Для решения задачи классификации сегодня наиболее актуальны методы, основанные на машинном обучении, так как они показывают наилучшие результаты. Однако, основной проблемой данных методов является отсутствие точных сравнительных цифр, так как для конкретных задач они могут отличаться. Выделяют следующие преимущества CNN, на основе которых было принято решение, использовать именно данную архитектуру для задачи классификации знаков:

- меньшее количество настраиваемых весов (по сравнению с полносвязной нейронной сетью)

- возможность реализации параллельных вычислений (на отдельных слоях), а, следовательно, возможность реализации обучения на графических процессорах;
- большая устойчивость к деформациям и сдвигам распознаваемого изображения;
- обучение классическим методом обратного распространения ошибки.

Описанные преимущества CNN делают данный метод одним из лучших для задачи распознавания и классификации изображений, поэтому был сделан выбор в пользу такого подхода.

Еще одним не мало важным преимуществом CNN является популярность данного способа классификации изображений. Инструменты для проектирования CNN присутствуют во многих популярных открытых библиотеках машинного обучения, например, TensorFlow, что позволяет сосредоточиться на проектировании архитектуры сети абстрагировавшись от проблем реализации и оптимизации разрабатываемого решения.

Выводы к главе 1

На основании, приведенного в первой главе данной выпускной квалификационной работы теоретического материала, можно сделать следующие выводы:

1. На данный момент не существует универсальных методов, позволяющих идеально решить задачу локализации и классификации объектов на изображении;

2. Были проанализированы наиболее популярные и актуальные методы для локализации и классификации объектов на изображении;

3. Для решения задач, поставленных в рамках данной ВКР, было принято решение использовать HOG&SVM дескрипторы в модуле локализации, а локализованное изображение передавать на CNN, реализующую модуль классификации.

Глава 2 АЛГОРИТМ ЛОКАЛИЗАЦИИ И КЛАССИФИКАЦИИ ОБЪЕКТОВ НА ИЗОБРАЖЕНИИ

2.1 Локализация дорожных знаков

Первым этапом в задаче распознавания дорожных знаков является локализация их расположения на изображении. Для данной задачи были рассмотрены наиболее известные методы и сделан выбор в пользу HOG&SVM классификаторов (классификаторы, основанные на гистограммах направленных градиентов с использованием метода опорных векторов), так как такой подход позволит получить большую точность, в сравнении с методом Виолы-Джонса, и сравнительно меньшие затраты ресурсов и времени, относительно нейросетевых подходов.

Для создания HOG&SVM классификаторов можно выделить два основных этапа:

1. формирование базы обучения;
2. обучение классификаторов.

Рассмотрим реализацию каждого из этих этапов подробно.

2.1.1 Формирование и подготовка базы дорожных знаков

Формирование базы данных - это очень большая проблема всех практико-ориентированных задач. К сожалению, в свободном доступе не было найдено базы российских дорожных знаков, которую бы можно было использовать для обучения классификатора. Было решено использовать базу немецких дорожных знаков GTSRB (German Traffic Sign Recognition Benchmark) [6], так как немецкие дорожные знаки в большинстве идентичны русским.

База данных GTSRB состоит из 43 различных классов дорожных знаков, представленных в виде набора реальных изображений. В базе данных содержится 39367 изображений для обучения и 12630 изображений для тестирования, а также описательные метафайлы с данными о находящихся на изображениях знаках.

Примеры изображений дорожных знаков, имеющиеся в базе GTSRB представлены на рисунке 12.

Рисунок 12 – Выборочные примеры из исходной базы данных

Так как изображения в базе имеют разное разрешение, яркость и контрастность, было принято решение нормализовать их яркость, контрастность и масштабировать до одинакового размера (с сохранением пропорций), также, каждый знак центрировался на изображении. Скрипт для предподготовки изображений представлен в Приложении А.

Так как различные классы изображений имеют различную форму знака, то было принято решение разбить их на подклассы. Два очевидных подкласса – круглые и треугольные, однако, остается два знака, которые не соответствуют этим двум подклассам: «уступи дорогу», «главная дорога». Поэтому было принято решение создать отдельные подклассы для данных знаков.

Из исходной базы данных была произведена выборка изображений, в среднем по 200 изображений из каждого класса, чтобы сократить время обучения классификатора. Пример предподготовки изображений показан на рисунке 13.

Рисунок 13 – Результат предподготовки изображения

После проверки и подготовки данных можно переходить к этапу обучения классификаторов. Программные реализации модулей, описываемых в данном разделе представлены в Приложении А («preprocessing.py», «hog_prepare.py»).

2.1.2 Обучение HOG&SVM дескрипторов

Создание детектора объектов HOG& SVM с нуля - довольно сложный процесс. К счастью, сегодня существуют различные библиотеки для работы с компьютерным зрением и машинным обучением, значительно упрощающие разработку. В данной работе для обучения и реализации классификаторов HOG&SVM используются библиотеки OpenCV и Dlib, написанные на C++, но также имеющие подвязки на Python и других языках программирования.

Обучение классификатора с помощью библиотеки Dlib можно разделить на три основных этапа:

- создание дескриптора HOG с определенными аннотациями, необходимыми для обучения (количество пикселей в блоке, положение рамки и др.);
- извлечение объектов HOG с использованием дескриптора из каждой аннотированной области;
- создание и обучение линейной модели SVM на извлеченных HOG объектах.

Для обучения и тестирования HOG&SVM классификаторов были разработаны необходимые модули и классы. Листинг описываемых в данном разделе классов и модулей представлен в Приложении А.

Для обучения был разработан класс Detector («detector.py»), в котором были реализованы методы-обвязки, необходимые для обучения и тестирования классификаторов. Также был разработан модуль для ручного аннотирования объектов, но так как набор данных уже был проаннотирован, то в нем нет необходимости для данной задачи, однако, он может быть использован для обучение дополнительным знакам.

Для более удобного обучения был разработан модуль тренировки классификаторов («train.py»), с помощью которого были обучены соответствующие классификаторы. Визуализированное представление обученных HOG классификаторов представлено на рисунке 14.

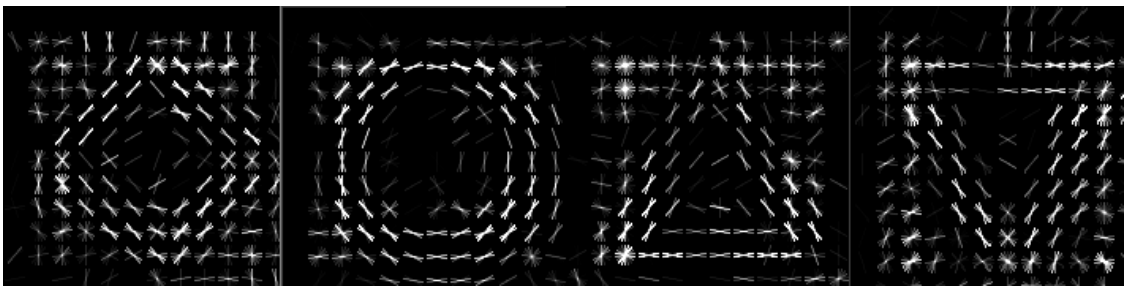

Рисунок 14 – Визуализированное представление обученных HOG дескрипторов

Для тестирования было разработано два модуля – «test.py» и «video_test.py» для единичного тестирования на изображении и тестирования в видеопотоке (с веб-камеры) соответственно. Результаты локализации и детектирования изображения представлены на рисунках 15-16.

Рисунок 15 – Результат локализации дорожного знака на изображении

Рисунок 16 – Результат локализации дорожного знака в видеопотоке

Как видно из рисунков 15-16 модуль локализации успешно реализован и протестирован. Численное тестирование эффективности и выводы о работе разработанного решения представлены в разделе 3.

2.2 Классификация дорожных знаков

Вторым этапом в задаче распознавания дорожных знаков является классификация локализованных объектов. Для данной задачи были рассмотрены наиболее известные методы и сделан выбор в пользу сверточной нейронной сети, так как такой подход позволит получить большую точность, в сравнении с другими подходами, основанными на машинном обучении или цветовых особенностей изображений, и сравнительно меньшие затраты ресурсов и времени, относительно обычного многослойного персептрона.

Для создания классифицирующей модели с использованием сверточной нейронной сети, можно выделить три основных этапа:

1. формирование и подготовка обучающей базы данных;
2. проектирование нейронной сети;
3. обучение нейронной сети.

Рассмотрим реализацию каждого из этих этапов подробно.

2.2.1 Подготовка обучающей базы данных

В обучающем наборе GTSRB изображения не обязательно имеют фиксированные размеры, и знак не обязательно центрируется на каждом изображении. Каждое изображение содержит около 10% границы вокруг фактического дорожного знака. Как было описано в предыдущих разделах изображения в обучающем наборе очень сильно отличаются друг от друга в плане освещенности, контрастности и положении знака, поэтому их необходимо привести к некоторому общему виду. Для решения данной проблемы был использован разработанный ранее модуль «preprocessing.py», решающий данную проблему. Все изображения были масштабированы до

единого размера: 48 на 48 пикселей. Результат работы модуля предобработки представлен на рисунке 17.

Рисунок 17 – Результат работы модуля предподготовки изображений

После нормализации входных данных и приведении их к единому размеру, можно переходить к проектированию и обучению нейронной сети.

2.2.2 Проектирование и обучение нейронной сети

Для решения задачи классификации дорожных знаков было принято решение воспользоваться библиотечным решением Keras, которая позволяет проектировать нейронные сети различных структур без необходимости их реализации с нуля, что значительно ускоряет разработку. Однако, выбор и проектирование правильной структуры сети – важный вопрос, который в значительной мере влияет на конечный результат.

Keras — открытая библиотека для работы с нейросетевыми алгоритмами, написанная на языке программирования Python. Она является надстройкой над другими открытыми библиотеками и фреймворками: Deeplearning4j, TensorFlow и Theano.

В рамках решения описанной ранее задачи была разработана и использована следующая модель сверточной нейронной сети: шесть свёрточных слоев, три слоя подвыборки, два полносвязных слоя, а также дополнительные нормализующие слои. Для более подробного представления о структуре разработанной модели приведено часть кода, отвечающая за формирование данной модели. Часть кода представлена на рисунке 18. Полный листинг модуля обучения представлен в Приложении А.

```
def cnn_model():
    """ Функция, определяющая модель разработанной сети """
    model = Sequential()

    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=(3, IMG_SIZE, IMG_SIZE),
                     activation='relu'))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), padding='same',
                     activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(128, (3, 3), padding='same',
                     activation='relu'))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(NUM_CLASSES, activation='softmax'))
    return model
```

Рисунок 18 – Часть листинга программы, проектирования модели сети

Sequential – контейнер Keras для линейного набора слоев. Conv2D – двумерный слой свертки. MaxPooling2D – слои подвыборки. Dropout слои добавлены, чтобы предотвратить перенасыщение. Dense - полносвязные слои. Чтобы прикрепить полносвязный слой к набору сверточных и слоев подвыборки используется Flatten слой, который выравнивает выходные значения сверточного слоя. Спроектированная структура CNN представлена графически на рисунке 19.

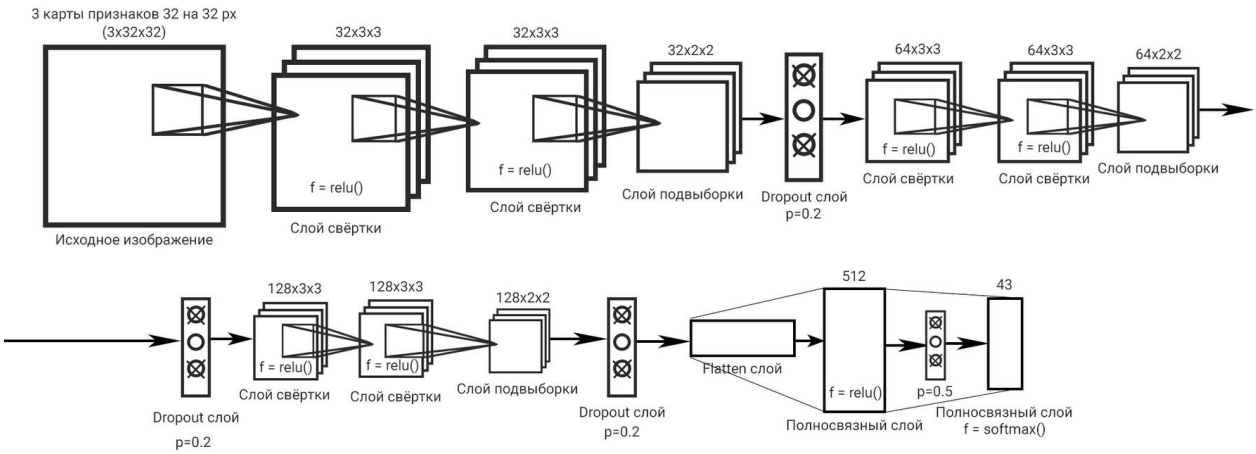

Рисунок 19 – Структура используемой CNN

На всех слоях, кроме последнего, было принято решение использовать Relu функцию активации, а на последнем Softmax, для вывода вероятности принадлежности объекта к конкретному классу.

Математически использованные функции активации описываются следующим образом:

Relu: $f\ x\ = \max(0, x)$

Softmax: $f(z)_i = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}$

Графическое представление данных функций активации представлено на рисунках 20 и 21.

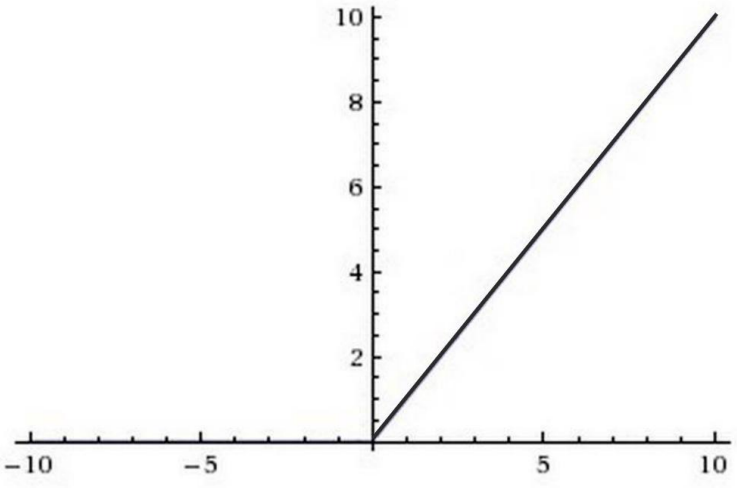

Рисунок 20 – График Relu-функции активации

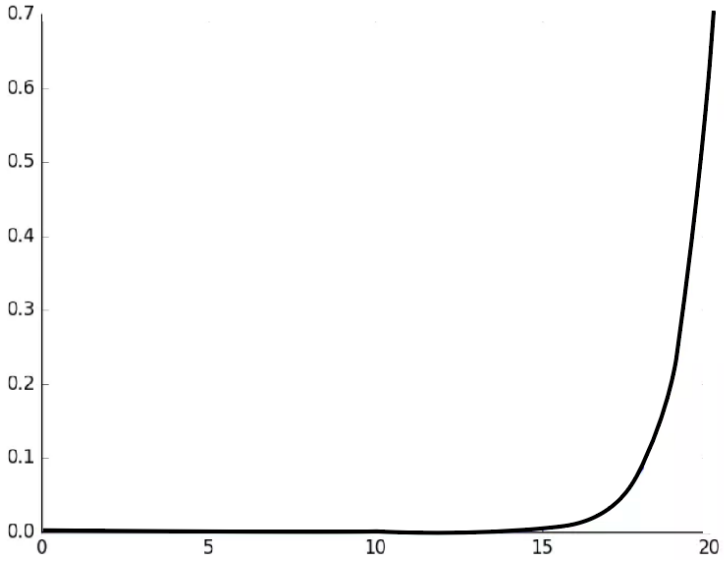

Рисунок 21 – График Softmax-функции активации

Также в модуль обучения нейронной сети были добавлены: планировщик скорости обучения, который помогает модели лучше учиться, и проверка контрольной точки (сохранение модели с наилучшей точностью).

Обучающий набор был разбит на партии по 32 экземпляра. Обучение производилось в течении 30 эпох. Скриншот части лога процесса обучения представлен на рисунке 22.

```
Train on 31367 samples, validate on 7842 samples
Epoch 1/30

   32/31367 [..............................] - ETA: 45:11 - loss: 3.7998 - acc: 0.0000e+00
   64/31367 [..............................] - ETA: 25:52 - loss: 3.7818 - acc: 0.0000e+00
   96/31367 [..............................] - ETA: 19:06 - loss: 3.7745 - acc: 0.0000e+00
  128/31367 [..............................] - ETA: 15:42 - loss: 3.7707 - acc: 0.0078
  160/31367 [..............................] - ETA: 13:46 - loss: 3.7689 - acc: 0.0125
  192/31367 [..............................] - ETA: 12:26 - loss: 3.7679 - acc: 0.0156
  224/31367 [..............................] - ETA: 11:28 - loss: 3.7654 - acc: 0.0134
  256/31367 [..............................] - ETA: 10:42 - loss: 3.7638 - acc: 0.0117
  288/31367 [..............................] - ETA: 10:08 - loss: 3.7618 - acc: 0.0104
  320/31367 [..............................] - ETA: 9:42 - loss: 3.7610 - acc: 0.0125
  352/31367 [..............................] - ETA: 9:20 - loss: 3.7598 - acc: 0.0142
  384/31367 [..............................] - ETA: 9:03 - loss: 3.7560 - acc: 0.0182
  416/31367 [..............................] - ETA: 8:50 - loss: 3.7531 - acc: 0.0240
  448/31367 [..............................] - ETA: 8:36 - loss: 3.7507 - acc: 0.0268
  480/31367 [..............................] - ETA: 8:23 - loss: 3.7491 - acc: 0.0292
  512/31367 [..............................] - ETA: 8:16 - loss: 3.7462 - acc: 0.0293
  544/31367 [..............................] - ETA: 8:07 - loss: 3.7455 - acc: 0.0276
  576/31367 [..............................] - ETA: 7:57 - loss: 3.7413 - acc: 0.0312
  608/31367 [..............................] - ETA: 7:50 - loss: 3.7396 - acc: 0.0296
  640/31367 [..............................] - ETA: 7:41 - loss: 3.7379 - acc: 0.0297
  672/31367 [..............................] - ETA: 7:34 - loss: 3.7351 - acc: 0.0283
  ...
31367/31367 [==============================] - 419s 13ms/step - loss: 1.6071 - acc: 0.5506 - val_loss: 0.1903 - val_acc: 0.9438
Epoch 2/30

   32/31367 [..............................] - ETA: 6:38 - loss: 0.3295 - acc: 0.9062
   64/31367 [..............................] - ETA: 6:47 - loss: 0.4158 - acc: 0.8750
   96/31367 [..............................] - ETA: 6:46 - loss: 0.4127 - acc: 0.8542
  128/31367 [..............................] - ETA: 6:44 - loss: 0.4572 - acc: 0.8438
  160/31367 [..............................] - ETA: 6:41 - loss: 0.4806 - acc: 0.8500
  192/31367 [..............................] - ETA: 6:41 - loss: 0.4666 - acc: 0.8438
  224/31367 [..............................] - ETA: 6:40 - loss: 0.4263 - acc: 0.8527
  256/31367 [..............................] - ETA: 6:39 - loss: 0.4130 - acc: 0.8594
  288/31367 [..............................] - ETA: 6:39 - loss: 0.4141 - acc: 0.8646
  320/31367 [..............................] - ETA: 6:40 - loss: 0.4033 - acc: 0.8656
  352/31367 [..............................] - ETA: 6:41 - loss: 0.3917 - acc: 0.8665
  384/31367 [..............................] - ETA: 6:40 - loss: 0.3772 - acc: 0.8698
  416/31367 [..............................] - ETA: 6:39 - loss: 0.3540 - acc: 0.8798
  448/31367 [..............................] - ETA: 6:39 - loss: 0.3531 - acc: 0.8817
  480/31367 [..............................] - ETA: 6:39 - loss: 0.3632 - acc: 0.8812
  512/31367 [..............................] - ETA: 6:39 - loss: 0.3557 - acc: 0.8848
  544/31367 [..............................] - ETA: 6:39 - loss: 0.3527 - acc: 0.8824
  ...
```

Рисунок 22 – Скриншот лога процесса обучения модели

Обучение представленной и описанной выше модели заняло около 8-10 часов на процессоре Intel Core I7-6700HQ. Стоит отметить, что если бы обучение проводилось на графическом процессоре, то оно заняло бы на

много меньше времени, благодаря присущей нейронным сетям параллелизуемости. В дальнейшем, при необходимости обучить более большую и сложную модель, лучше использовать такой подход.

Так как разработка и обучение велись на языке программирования Python, то также необходимо установить соответствующий интерпретатор, поддерживающий выполнение на графическом процессоре.

После обучения модель была сохранена в формате двоичных данных HDF5 и в дальнейшем может использоваться без необходимости обучать её снова.

2.3 Разработка и проектирование алгоритма комплексного распознавания дорожных знаков

После разработки отдельных модулей, можно переходить к проектированию и разработке комплексного алгоритма локализации и классификации объектов на изображении.

Для реализации алгоритма комплексного распознавания дорожных знаков использовался язык программирования Python и сторонние библиотеки с открытым исходным кодом: OpenCV, Dlib, NumPy, Keras.

В общем виде предлагаемый алгоритм можно описать следующим образом:

1. исходное изображение подается на вход HOG&SVM дескрипторов, с помощью которых локализуются потенциальные искомые объекты;
2. потенциальные локализованные области исходного изображения подаются на вход модуля предподготовки изображений;
3. изображения обрабатываются (выравнивается яркость, контрастность, приводятся к единому размеру) и подаются на вход CNN;
4. обученная модель возвращает идентификатор для каждой потенциальной области;

5. локализованные и классифицированные знаки обводятся рамкой на исходном изображении и подписываются в соответствии с идентификатором, который выдала CNN;

Суть работы предлагаемого алгоритма показана в виде блок-схемы на рисунке 22.

Целью данной ВКР является разработка универсального алгоритма локализации и классификации объектов на изображении, поэтому блок-схема, представленная на рисунке 22, построена без привязки к конкретным моделям и дескрипторам, обученным ранее. Разработанный алгоритм достаточно универсален и может применяться и для других практико-ориентированных задач.

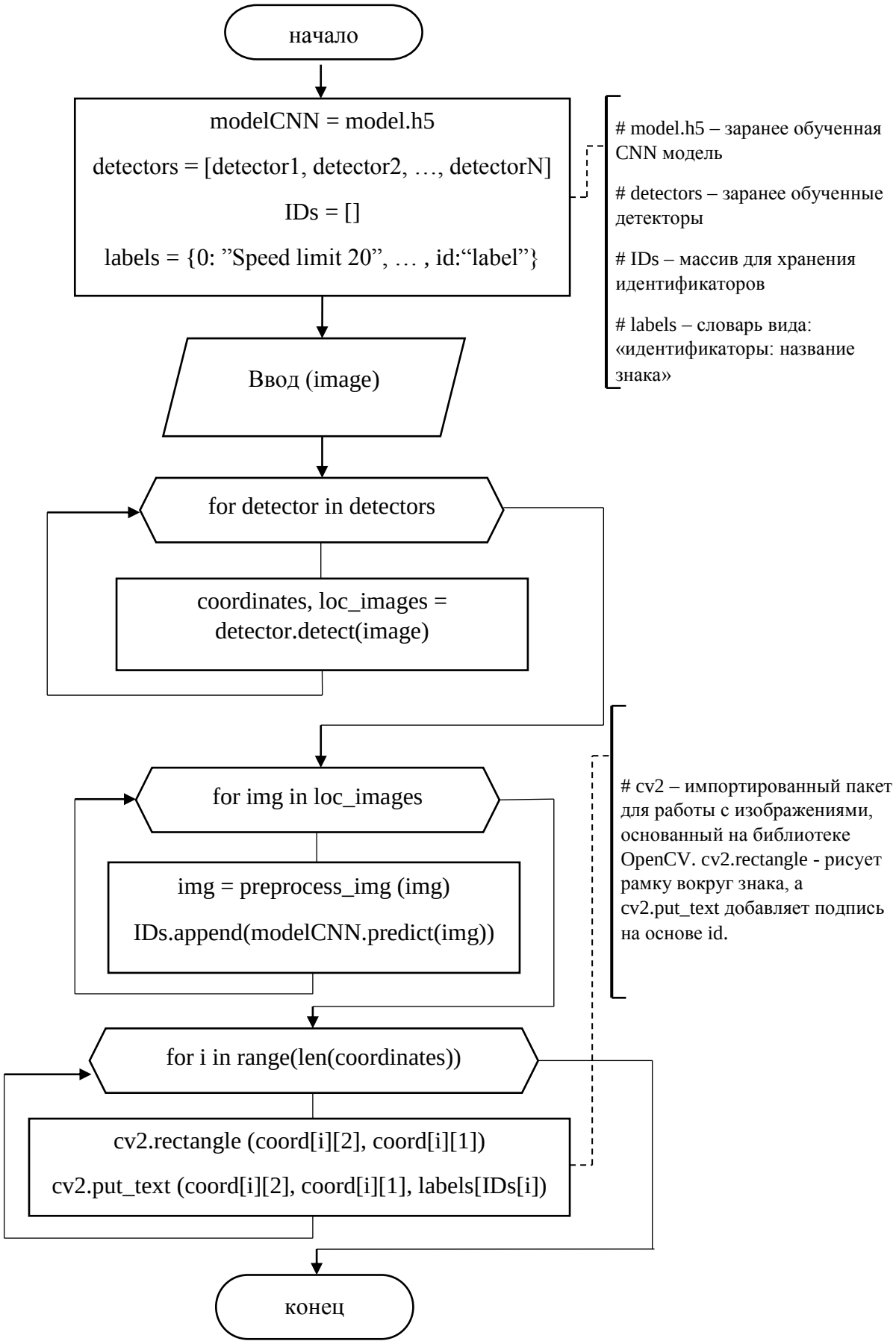

Рисунок 23 – Блок-схема алгоритма локализации и классификации объектов на изображении

Алгоритм, представленный на рисунке 22 был разработан и применен для задачи распознавания дорожных знаков. Пример успешной работы разработанного решения представлен на рисунке 23.

К сожалению, разработанное решение не дает 100% гарантию успешного результата, поэтому возможно получить не то поведение, которое ожидается. Пример ошибочной работы разработанного решения представлен на рисунке 24.

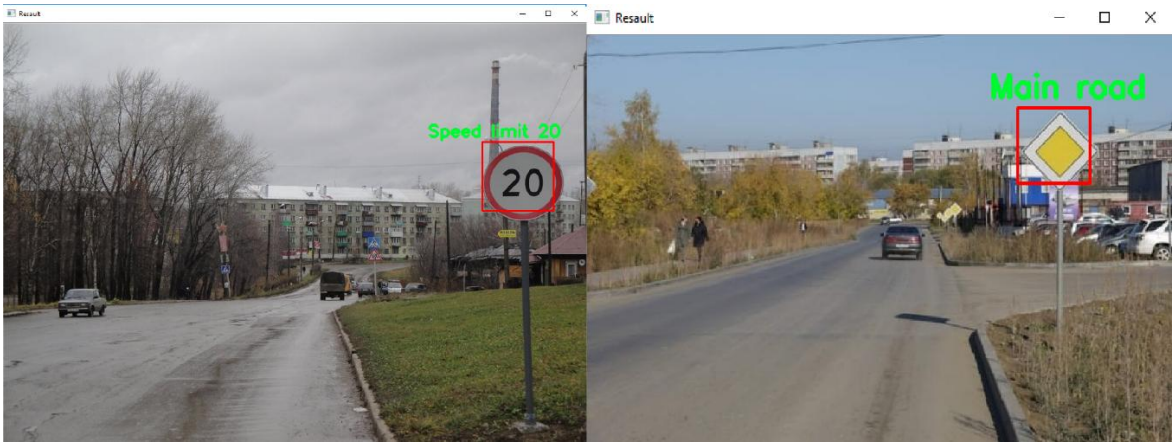

Рисунок 24 – Пример успешной работы разработанного решения

Рисунок 25 – Пример неудачной работы разработанного решения

Стоит отметить, что разработанное решение работает достаточно точно на всех этапах, но необходимо провести численное тестирование разработанного решения.

2.4 Тестирование разработанного программного решения

Для определения того, на сколько точен предложенный алгоритм и его программная реализация, для рассматриваемой в рамках данной ВКР задачи распознавания дорожных знаков, необходимо провести численное тестирования основных его модулей, а именно модуля локализации и модуля классификации.

2.4.1 Тестирование и оценка точности модуля локализации

Модуль локализации играет важную роль в работе всего алгоритма. Особенно важно локализовать именно дорожный знак, а не иной схожий объект, поэтому на данном этапе особенно требуется высокая точностью работы.

В результате тестирования разработанного модуля было выявлено, что модуль локализации вполне успешно справляется и со сложными ситуациями, даже когда дорожный знак находится под углом относительно снимающей камеры. Пример такой ситуации представлен на рисунке 26.

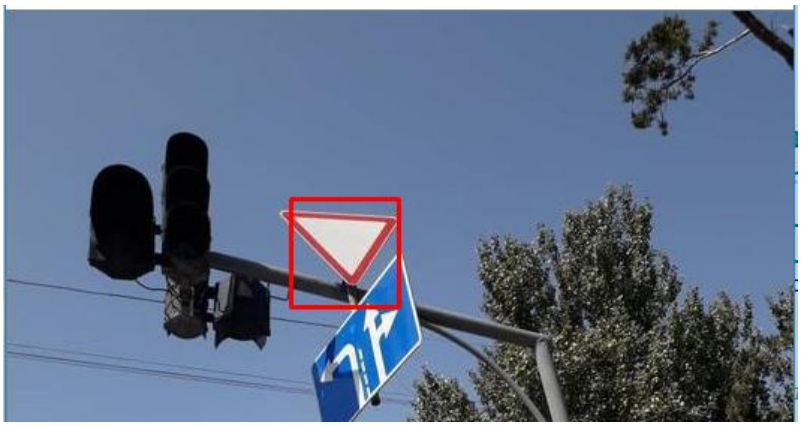

Рисунок 26 – Пример успешной локализации знака под углом

Однако, существуют ситуации не посильные для локализации, например, когда часть знака перекрыта деревом или иным объектом, пример такой ситуации представлен на рисунке 27.

Рисунок 27 – Пример неудачной локализации знака

В конечном результате, нам не принципиально, какой именно детектор распознал дорожный знак, поэтому логично будет тестировать их все вместе, а не по отдельности. Тестирование модуля локализации, будет проводиться на тестовом наборе данных из GTSRB [6].

Следует учитывать, что ошибкой локализации будет считаться, как несрабатывание детектора, когда это необходимо, так и его ложные срабатывания, поэтому следует проверять количество среагировавших и не среагировавших детекторов на каждом тестовом изображении.

Точность обученных детекторов будет вычисляется по следующей формуле: $acc = \frac{\text{количество корректных предсказаний}}{\text{количество объектов тестирования}}$, где под корректными предсказаниями подразумеваются как отсутствие ложных срабатываний, так и наличие предсказания, при наличии знака на изображении.

Для тестирования работы HOG&SVM детекторов был разработан соответствующий модуль «test_hog.py», результат работы которого показан на рисунке 26. Полный листинг модуля тестирования CNN представлен в Приложении А.

```
...
testForHOG\12622.png
testForHOG\12623.png
testForHOG\12624.png
testForHOG\12625.png
testForHOG\12626.png
testForHOG\12627.png
testForHOG\12628.png
testForHOG\12629.png
9807
Точность локализации знаков = 77.64845605700714 %
```

Рисунок 28 – Результат тестирования модуля локализации дорожных знаков

Из рисунка 28 видно, что модуль локализации успешно обнаружил 9807 знаков из 12630 изображений. Хоть точность 77.65 % не идеальна, данный способ локализации вполне пригоден в практическом применении, так как чаще всего стоит задача локализовать дорожный знак в видеопотоке, где большое количество непрерывных кадров.

Точность работы модуля локализации может быть улучшена с использованием Region-based convolutional neural network (R-CNN). Однако, в таком случае придется пожертвовать производительностью и возможностью детектировать знаки в реальном времени, так как время обнаружения знака может занять 40-60 секунд. Сегодня набирают популярность различные модификации R-CNN: Fast R-CNN и Faster R-CNN, которые позволяют значительно ускорить распознавание, с некоторой потерей точности.

Подводя итог тестирования модуля локализации, можно сделать вывод, что HOG&SVM дескрипторы для данной задачи – оптимальный компромисс между точностью и быстродействием. Возможно, в будущем, в связи с развитием и ростом популярности сферы машинного обучения, появятся еще более точные и быстродейственные подходы.

2.4.2 Тестирование классифицирующей свёрточной нейронной сети

После локализации потенциальный знак подается на вход заранее обученной CNN. Точность классификации очень важны для корректной работы всей системы. Если локализованный объект будет распознан неправильно, то разработанная система не только не приносит пользы, но и может наоборот создать проблемы, дезинформируя пользователя.

Для тестирования работы нейронной сети необходимо подготовить большую выборку разнообразных изображений и желательно с пометками о наличии на них объекта, относящегося к тому или иному классу. Данная задача достаточно трудоемкая, ведь для корректных результатов необходимо несколько тысяч таких изображений. К счастью в наборе данных GTSRB [6] присутствует набор тестовых промаркированных данных, состоящий из 12630 изображений для тестирования, который и будет использоваться при тестировании разработанной модели.

Точность работы классифицирующей модели будет определяться по аналогичной формуле, представленной в разделе 3.1. Для тестирования классифицирующей нейронной сети был разработан соответствующий модуль «test_cnn.py», результат работы которого показан на рисунке 27. Полный листинг модуля тестирования CNN представлен в Приложении А.

```
Модель загружена...
Точность классификации набора = 97.02296120348377 %
```

Рисунок 29 – Результат тестирования CNN

Как видно из рисунка 29, были получены очень неплохие результаты. Распознавание с точностью 97% может гарантировать хорошую работу модуля классификации, так как данный результат, близок к средней точности распознавания человеком 98.84% [11], который установили J. Stallkampa, M. Schlipsinga, J. Salmena и C. Igelb в своей научной работе.

Выводы к главе 2

На основании, разработанного в данной главе алгоритма и его программной реализации можно сделать следующие выводы:

1. Представленный в разделе 2.3 алгоритм достаточно универсален и может быть применен к любой подобной задачи;
2. Разработанное решение выдает как верные, так и неверные результаты;

В разделе 2.4 было проведено численное тестирование разработанного решения результатами которого стали следующие показатели:

1. Модуль локализации детектирует дорожный знак с вероятностью 77,65 %;
2. Модуль классификации верно классифицирует дорожный знак с вероятностью 97%.

Полученные результаты вполне удовлетворительны и применимы на практике, однако, возможно, в ближайшем будущем появятся и зарекомендуют себя ещё более точное нейросетевые подходы для локализации объектов.

ЗАКЛЮЧЕНИЕ

В рамках данной выпускной квалификационной работы были выполнены все цели и задачи. Была рассмотрена задача локализации и классификации объектов на изображении на примере актуальной проблемы современного общества и компьютерного зрения – распознавание дорожных знаков.

В процессе выполнения бакалаврской работы решены следующие задачи:

1. Исследованы наиболее успешно зарекомендовавшие себя методы локализации и классификации изображений и сделан обоснованный выбор наиболее удачных решений.

2. На основании сделанного выбора разработан универсальный подход и алгоритм для решения задачи локализации и классификации объектов на изображении, который в последствии может быть применен для других практико-ориентированных задач локализации и распознавания образов.

3. Представлена программная реализация разработанного алгоритма и проведено численное тестирование её эффективности. В ходе тестирования были получены хорошие результаты на каждом модуле разработанного решения. На этапе локализации программа локализует дорожный знак с вероятностью 77,65%, а на этапе классификации удалось достичь точность в 97%, что очень близко к точности человека.

Разработанное решение может быть расширено и использоваться как система помощи водителю (автопилот), или же предложенный алгоритм может применяться для совершенно иной задачи с схожей проблематикой.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Нормативно-правовые акты

1. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем.

Научная и методическая литература

2. Вьюгин В. Математические основы теории машинного обучения и прогнозирования. — МЦМНО, 2013 г. — 390 с.

3. Изучаем Python, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011 – 1280 с.

4. Назаренко А.В. Компьютерное зрение. Современный подход. Издательство: Вильямс, 2004 г. – 928 с.

Электронные ресурсы

5. CBCL PEDESTRIAN DATABASE [Электронный ресурс]. – Режим доступа: http://cbcl.mit.edu/software-datasets/PedestrianData.html – Pedestrian Dataset Massachusetts Institute of Technology – (дата обращения 14.02.2019).

6. German traffic sign recognition benchmark [Электронный ресурс]. – Режим доступа: http://benchmark.ini.rub.de/?section=gtsrb&subsection=dataset – (GTSRB) dataset – (дата обращения 26.02.2019).

7. THOTH [Электронный ресурс]. – Режим доступа: http://lear.inrialpes.fr/data – INRIA Person Data Set – (дата обращения 15.02.2019).

8. Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Метод_Виолы_—_Джонса – Метод Виолы-Джонса – (дата обращения 25.12.2018).

9. Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Признаки_Хаара – Признаки Хаара – (дата обращения 26.12.2018).

10. Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Гистограмма_направленных_градиентов – Гистограмма направленных градиентов – (дата обращения 13.01.2019).

Литература на иностранном языке

11. J. Stallkampa, M. Schlipsinga, J. Salmena C. Igelb Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition, 2012. – 11 p.

12. Melin, P. Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization / Patricia Melin, Oscar Castillo, Janusz Kacprzyk. – Springer International Publishing, 2015. – 637 p.

13. C. Ronan, W. Jason. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. Proceedings of the 25th International Conference on Machine Learning. ICML '08. New York, NY, USA: ACM. 2008. 160–167 p.

14. Ciresan, D. C., Meier, U., Masci, J., & Schmidhuber, J. A committee of neural networks for traffic sign classification. In International Joint Conference on Neural Networks, 2011. – 4 p.

15. Habibi Aghdam. H, Jehani Heravi. E, Guide to Convolutional Neural Networks A Practical Application to Traffic-Sign Detection and Classification, 2017. – 282 p.

16. Jain, V. and Seung, S. H. Natural image denoising with convolutional networks. 2008. - 8p.

17. Liu, C.-L., Yin, F., Wang, Q.-F., & Wang, D.-H. ICDAR 2011 chinese handwriting recognition competition. In International Conference on Document Analysis and Recognition, 2011. – 1464 –1469 p.

18. Mark Summerfield. Programming in Python 3: a complete introduction to the Python language 2nd ed. 2009. – 636 p.

19. Navneet Dalal, Bill Triggs. Histograms of Oriented Gradients for Human Detection, 2005. – 8 p.

20. Zed A. Shaw. Learn python the hard way – Addison-Wesley, 2013. – 306 p.

ПРИЛОЖЕНИЕ А

Листинг кода программной реализации

preprocessing.py:

```
from skimage import io, color, exposure, transform
import numpy as np
IMAGE_SIZE = 240
# ПРЕПРОЦЕССИНГ ИЗОБРАЖЕНИЯ
# выравнивания гистограммы в цветовом пространстве HSV и изменения размера изображения до стандартного размера
def preprocess_img(img):
    hsv = color.rgb2hsv(img)
    hsv[:,:,2] = exposure.equalize_hist(hsv[:,:,2])
    img = color.hsv2rgb(hsv)

    min_side = min(img.shape[:-1])
    centre = img.shape[0]//2, img.shape[1]//2
    img = img[centre[0]-min_side//2:centre[0]+min_side//2,
              centre[1]-min_side//2:centre[1]+min_side//2,
              :]
    img = transform.resize(img, (IMAGE_SIZE, IMAGE_SIZE))
    img = np.rollaxis(img,-1)
    return img

def get_class(img_path):
    return int(img_path.split('\\')[-2])
```

hog_prepare.py:

```
import cv2
import matplotlib.pyplot as plt
import csv
```

```
import numpy as np
from scipy.ndimage import io
from preprocessing import preprocess_img
import scipy.misc

circles = [0,1,2,3,4,5,6,7,8,9,10,14,15,16,17,32,33,34,35,36,37,38,39,40,41,42]
triengles = [11,18,19,20,21,22,23,24,25,26,27,28,29,30,31]
main_road = 12
give_way = 13

def prepare_signs(rootpath, array):
    images = []
    labels = []
    annotations = []
    img_paths = []
    if not isinstance(array, list):
        prefix = rootpath + '/' + format(array, '05d') + '/'
        gt_file = open(prefix + 'GT-' + format(array, '05d') + '.csv')
        gt_reader = csv.reader(gt_file, delimiter=';')
        next(gt_reader)
        check = 200
        for row in gt_reader:
            if check > 1:
                images.append(plt.imread(prefix + row[0]))
                image_array = preprocess_img(io.imread(prefix + row[0]))
                scale = 240 / float(row[1])
                (x, y, xb, yb) = [float(row[3]) * scale, float(row[4]) * scale, float(row[5])
* scale,
                                  float(row[6]) * scale]
                annotations.append([int(x), int(y), int(xb), int(yb)])
```

```
            path = "circles/" + str(row[0][:-4]) + "_" + str(array) + ".jpg"
            scipy.misc.toimage(image_array).save(path)
            img_paths.append(path)
            labels.append(row[7])
            check -= 1
        else:
            break
    else:
        for c in array:
            prefix = rootpath + '/' + format(c, '05d') + '/'
            gt_file = open(prefix + 'GT-'+ format(c, '05d') + '.csv')
            gt_reader = csv.reader(gt_file, delimiter=';')
            next(gt_reader)
            check = 10
            for row in gt_reader:
                if check > 1:
                    images.append(plt.imread(prefix + row[0]))
                    image_array = preprocess_img(io.imread(prefix + row[0]))
                    scale = 240 / float(row[1])
                    (x, y, xb, yb) = [float(row[3])*scale, float(row[4])*scale,
float(row[5])*scale, float(row[6])*scale]
                    annotations.append([int(x), int(y), int(xb), int(yb)])
                    path = "circles/"+str(row[0][:-4])+"_"+str(c)+".jpg"
                    scipy.misc.toimage(image_array ).save(path)
                    img_paths.append(path)
                    labels.append(row[7])
                    check-=1
                else:
                    break
    return images, labels, annotations, img_paths
```

```
# КРУГЛЫЕ ЗНАКИ
images, labels, annotations, im_paths =
prepare_signs('GTSRB/Final_Training/Images/', circles)
annotations = np.array(annotations)
im_paths = np.array(im_paths, dtype="unicode")
ann = annotations[0]
image = cv2.imread(im_paths[0])
image = cv2.rectangle(image,(ann[0], ann[1]),(ann[2], ann[3]), (0,255,0), 3)
cv2.imshow("Image", image)
cv2.waitKey(0)
np.save('CircleAnnot.npy',annotations)
np.save('CircleImages.npy', im_paths)

# ТРЕУГОЛЬНЫЕ ЗНАКИ
images, labels, annotations, im_paths =
prepare_signs('GTSRB/Final_Training/Images/', triengles)
annotations = np.array(annotations)
im_paths = np.array(im_paths, dtype="unicode")
ann = annotations[5]
image = cv2.imread(im_paths[5])
image = cv2.rectangle(image,(ann[0], ann[1]),(ann[2], ann[3]), (0,255,0), 3)
cv2.imshow("Image", image)
cv2.waitKey(0)
np.save('TriangleAnnot.npy',annotations)
np.save('TriangleImages.npy', im_paths)

# ЗНАК ГЛАВНОЙ ДОРОГИ
images, labels, annotations, im_paths =
prepare_signs('GTSRB/Final_Training/Images/', main_road)
```

```
annotations = np.array(annotations)
im_paths = np.array(im_paths, dtype="unicode")
ann = annotations[1]
image = cv2.imread(im_paths[1])
image = cv2.rectangle(image,(ann[0], ann[1]),(ann[2], ann[3]), (0,255,0), 3)
cv2.imshow("Image", image)
cv2.waitKey(0)
np.save('MainRoadAnnot.npy',annotations)
np.save('MainRoadImages.npy', im_paths)
# ЗНАК УСТУПИ ДОРОГУ
images, labels, annotations, im_paths =
prepare_signs('GTSRB/Final_Training/Images/', give_way)
annotations = np.array(annotations)
im_paths = np.array(im_paths, dtype="unicode")
ann = annotations[1]
image = cv2.imread(im_paths[1])
image = cv2.rectangle(image,(ann[0], ann[1]),(ann[2], ann[3]), (0,255,0), 3)
cv2.imshow("Image", image)
cv2.waitKey(0)
np.save('GiveWayAnnot.npy',annotations)
np.save('GiveWayImages.npy', im_paths)
```

detector.py:

```
import dlib
import cv2
class ObjectDetector(object):
    def __init__(self,options=None,loadPath=None):
        self.options = options
        if self.options is None:
            self.options = dlib.simple_object_detector_training_options()
```

```
        if loadPath is not None:
            self._detector = dlib.simple_object_detector(loadPath)

    def __prepare_annot(self, annotations):
        annots = []
        for (x,y,xb,yb) in annotations:

annots.append([dlib.rectangle(left=int(x),top=int(y),right=int(xb),bottom=int(yb))]
)
        return annots

    def __prepare_img(self, img_paths):
        images = []
        for imPath in img_paths:
            image = cv2.imread(imPath)
            image = cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
            images.append(image)
        return images

    def fit(self, imagePaths, annotations, visualize=False, savePath=None):
        annotations = self.__prepare_annot(annotations)
        images = self.__prepare_img(imagePaths)
        self._detector = dlib.train_simple_object_detector(images, annotations,
self.options)

        if visualize:
            win = dlib.image_window()
            win.set_image(self._detector)
            dlib.hit_enter_to_continue()
```

```
        if savePath is not None:
            self._detector.save(savePath)
        return self

    def predict(self,image):
        boxes = self._detector(image)
        preds = []
        for box in boxes:
            (x,y,xb,yb) = [box.left(),box.top(),box.right(),box.bottom()]
            preds.append((x,y,xb,yb))
        return preds

    def detect(self,image,annotate=None):
        croped_imgs = []
        preds = self.predict(image)
        for (x, y, xb, yb) in preds:
            cv2.rectangle(image, (x, y), (xb, yb), (0, 0, 255), 2)
            croped_imgs.append(image[y:yb, x:xb])
        return preds, croped_imgs
```

train.py:

```
from detector import ObjectDetector
import numpy as np
import argparse
ap = argparse.ArgumentParser()
ap.add_argument("-a","--annotations",required=True,help="путь до папки с
аннотациями.")
ap.add_argument("-i","--images",required=True,help="путь до папки с
изображениями.")
ap.add_argument("-d","--detector",default=None,help="путь для сохранения
```

```
обученного детектора.")
args = vars(ap.parse_args())
print ("[INFO] аннотации и изображения загруженны")
annots = np.load(args["annotations"])
imagePaths = np.load(args["images"])
detector = ObjectDetector()
print("[INFO] детектор успешно обучен и сохранен")
detector.fit(imagePaths,annots,visualize=True,savePath=args["detector"])
```

test.py:

```
from detector import ObjectDetector
import cv2
import argparse
ap = argparse.ArgumentParser()
ap.add_argument("-d","--detector",required=True,help="путь до папки с
обученным детектором")
ap.add_argument("-i","--image",required=True,help="путь до тестового
изображения")
ap.add_argument("-a","--annotate",default=None,help="текст аннотации")
args = vars(ap.parse_args())

detector = ObjectDetector(loadPath=args["detector"])
imagePath = args["image"]
image = cv2.imread(imagePath)
detector.detect(image,annotate=args["annotate"])
video_test.py
import cv2
from detector import ObjectDetector
video_capture = cv2.VideoCapture(0)
detector1 = ObjectDetector(loadPath='circle_sign_detector.svm')
```

```
detector2 = ObjectDetector(loadPath='triangle_sign_detector.svm')
detector3 = ObjectDetector(loadPath='main_road_sign_detector.svm')
detector4 = ObjectDetector(loadPath='give_way_sign_detector.svm')
while True:
    ret, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    image = frame
    detector1.detect(image, annotate='Circle sign')
    detector2.detect(image, annotate='Triangle sign')
    detector3.detect(image, annotate='MainRoad sign')
    detector4.detect(image, annotate='GiveWay sign')
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

train_model.py:

```
import time
import numpy as np
from skimage import io
import glob
import os
import h5py
from keras.models import Sequential
from keras.layers.convolutional import Conv2D
from keras.optimizers import SGD
from keras.layers.pooling import MaxPooling2D
from keras import backend as K
from keras.callbacks import LearningRateScheduler, ModelCheckpoint
```

```
from keras.layers.core import Dense, Dropout, Flatten
K.set_image_data_format('channels_first')
from preprocessing import preprocess_img
IMAGEG_SIZE = 48
NUMS_CLASSES = 43
def get_class(img_path):
    return int(img_path.split('\\')[-2])
# Предварительно обрабатываем все тренировочные образы в массив и
пишем в файл X.h5
# где X.h5 хранящий пары img + lables
try:
    with h5py.File('X.h5') as hf:
        X, Y = hf['imgs'][:], hf['labels'][:]
    print("Loaded images from X.h5")
except (IOError, OSError, KeyError):
    print("Error in reading X.h5. Processing all images...")
    root_dir = 'GTSRB/Final_Training/Images/'
    imgs = []
    labels = []
    all_img_paths = glob.glob(os.path.join(root_dir, '*/*.ppm'))
    np.random.shuffle(all_img_paths)
    for img_path in all_img_paths:
        try:
            img = preprocess_img(io.imread(img_path))
            label = get_class(img_path)
            imgs.append(img)
            labels.append(label)
            if len(imgs) % 1000 == 0: print("Processed {}/{}".format(len(imgs),
len(all_img_paths)))
        except (IOError, OSError):
```

```
            print('missed', img_path)
            pass
    X = np.array(imgs, dtype='float32')
    Y = np.eye(NUMS_CLASSES, dtype='uint8')[labels]
    with h5py.File('X.h5', 'w') as hf:
        hf.create_dataset('imgs', data=X)
        hf.create_dataset('labels', data=Y)
# Определяем Keras модель нашей сети
def mode_cnnl():
    """ Функция, определяющая модель разработанной сети """
    model = Sequential()
    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=(3, IMAGE_SIZE, IMAGE_SIZE),
                     activation='relu'))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(64, (3, 3), padding='same',
                     activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Conv2D(128, (3, 3), padding='same',
                     activation='relu'))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))
    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
```

```
    model.add(Dense(NUMS_CLASSES, activation='softmax'))
    return model
model = model_cnn()
lr = 0.01
sgd = SGD(lr=lr, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
          optimizer=sgd,
          metrics=['accuracy'])
def lr_schedule(epoch):
    return lr*(0.1**int(epoch/10))
# Начинаем тренировку
batch_size = 32
epoch_nums = 30
start_time = time.time()
model.fit(X, Y,
          batch_size=batch_size,
          epochs=epoch_nums,
          validation_split=0.2,
          shuffle=True,
          callbacks=[LearningRateScheduler(lr_schedule),
                ModelCheckpoint('model.h5',save_best_only=True)]
          )
print("--- Время обучения модели: {} seconds ---".format(time.time() -
start_time))
```

test_hog.py:

```
import os
from detector import ObjectDetector
import cv2
import csv
from scipy.ndimage import io
```

```
from preprocessingHOG import preprocess_img
from scipy.misc import toimage
detector1 = ObjectDetector(loadPath='hogsvm/circle_sign_detector.svm')
detector2 = ObjectDetector(loadPath='hogsvm/triangle_sign_detector.svm')
detector3 = ObjectDetector(loadPath='hogsvm/main_road_sign_detector.svm')
detector4 = ObjectDetector(loadPath='hogsvm/give_way_sign_detector.svm')
detectors = [detector1, detector2, detector3, detector4]
def prepareSigns(rootpath):
        gtFile = open(rootpath + "GT-final_test.csv")
        gtReader = csv.reader(gtFile, delimiter=';')
        next(gtReader)
        for row in gtReader:
            image_array = preprocess_img(io.imread(rootpath + row[0]))
            path = "testForHOG/"+str(row[0][:-4])+".png"
            toimage(image_array).save(path)
def testTrafficSigns(rootpath):
    truecount = 0
    img_paths = []
    for file in os.listdir(rootpath):
        if file.endswith(".png"):
            img_paths.append(os.path.join(rootpath, file))
    for img in img_paths:
        print(img)
        k = 0
        for detector in detectors:
            coords, imgs = detector.detect(cv2.imread(img), annotate='Sign')
            if len(coords) == 1:
                k +=1
        if k == 1:
            truecount += 1
```

```
    return truecount
prepareSigns("GTSRB/Final_Test/Images/")
truecount = testTrafficSigns('testForHOG')
print(truecount)
print("Точность локализации знаков = {} %".format((truecount/12630)*100))
```

test_cnn.py:

```
from keras.engine.saving import load_model
from skimage import io
import os
import numpy as np
import pandas as pd
from preprocessing import preprocess_img
test = pd.read_csv('GTSRB/GT-final_test.csv', sep=';')
X_test = []
y_test = []
i = 0
for file_name, class_id in zip(list(test['Filename']), list(test['ClassId'])):
    img_path = os.path.join('GTSRB/Final_Test/Images/', file_name)
    X_test.append(preprocess_img(io.imread(img_path)))
    y_test.append(class_id)
X_test = np.array(X_test)
y_test = np.array(y_test)
model = load_model('model1.h5')
print('Модель загружена...')
y_pred = model.predict_classes(X_test)
acc = np.sum(y_pred==y_test)/np.size(y_pred)
print("Точность классификации набора = {} %".format(acc * 100))
# Полученная точность классификации 97,28 %
```

Общая программная реализация предложенного алгоритма:

img_detector_and_classificator.py:

```
import cv2
from PIL import Image
from keras.engine.saving import load_model
import numpy as np
from preprocessing import preprocess_img
from detector import ObjectDetector
detector1 = ObjectDetector(loadPath='hogsvm/circle_sign_detector.svm')
detector2 = ObjectDetector(loadPath='hogsvm/triangle_sign_detector.svm')
detector3 = ObjectDetector(loadPath='hogsvm/main_road_sign_detector.svm')
detector4 = ObjectDetector(loadPath='hogsvm/give_way_sign_detector.svm')
detectors = [detector1, detector2, detector3, detector4]
model = load_model('model1.h5')
image = cv2.imread('test_data/9.jpg')
for detector in detectors:
    coords, imgs = detector.detect(image, annotate='Sign')
    i=0
    for crop_img in imgs:
        x = coords[i][2]
        y = coords[i][1]
        X_test = Image.fromarray(crop_img)
        X_test = preprocess_img(X_test)
        X_test = np_image = np.expand_dims(X_test, axis=0)
        text = "ID:" + str(model.predict_classes(X_test)[0])
        cv2.putText(image, text, (x-180, y - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 1.0, (50, 250, 0), 3)
        i += 1
cv2.imshow('Resault', image)
cv2.waitKey(0)
```