МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики
Кафедра прикладной математики

Допустить к защите
Заведующий кафедрой
д-р физ.-мат. наук, профессор
_____ М.Х. Уртенов
16.06. _____ 2022 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(БАКАЛАВРСКАЯ РАБОТА)

ИСПОЛЬЗОВАНИЕ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РЕШЕНИЯ ЗАДАЧИ РАСПОЗНАВАНИЯ РЕГИСТРАЦИОННЫХ АВТОМОБИЛЬНЫХ НОМЕРОВ

Работу выполнила _____ Д.С. Овчарик

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Математическое и информационное обеспечение экономической деятельности

Научный руководитель
канд. техн. наук, доц. _____ Е.Ю. Пелипенко

Нормоконтролер
преподаватель _____ Е.С. Троценко

Краснодар
2022

РЕФЕРАТ

Выпускная квалификационная работа 42 с., 3 ч., 11 рис., 19 источников.

СВЁРТОЧНАЯ НЕЙРОННАЯ СЕТЬ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, РАСПОЗНАВАНИЕ ГРАФИЧЕСКИХ ОБРАЗОВ, СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЯ, РЕГИСТРАЦИОННЫЙ АВТОМОБИЛЬНЫЙ НОМЕР, ГОСУДАРСТВЕННЫЙ НОМЕРНОЙ ЗНАК

Объектом исследования в данной работе выступают методы распознавания графических образов.

Цель выпускной квалификационной работы – изучение технологий компьютерного зрения и написание программы, осуществляющей сегментацию и распознавание символов регистрационного автомобильного номера на изображении.

В результате работы были проанализированы существующие методы распознавания графических образов, разработан алгоритм сегментации и распознавания букв и цифр государственного номерного знака транспортного средства на изображении с использованием свёрточной нейронной сети.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время происходит бурное развитие информационных технологий. С их помощью автоматизируются многие процессы в самых различных отраслях – в промышленности, бизнесе, медицине, образовании, транспорте и бытовой сфере. В области обеспечения безопасности и контроля дорожного движения тоже есть проблемы, решение которых возможно благодаря применению компьютерных технологий.

В современном мире наличие автомобиля является потребностью чуть ли не каждого человека. Число автомобилей на дорогах стремительно растет.

Аналитическое агентство «АВТОСТАТ» периодически проводит анализ данных по автомобильному рынку и автобизнесу. Так, в 2016 году оно зафиксировало на территории Российской Федерации 40,9 миллионов легковых автомобилей. А недавное исследование показало, что сейчас в стране около 45,5 миллионов легковых автомобилей [1].

Транспортная инфраструктура развивается быстрыми темпами, и необходимость контролировать движение транспорта возрастает.

Распознавание регистрационных номерных знаков – это одна из задач идентификации транспортных средств (ТС). Системы распознавания регистрационных номеров по изображению и видео используются во многих сферах, связанных с транспортом, например, на автобазах, предприятиях, стоянках ТС, автозаправочных станциях, станциях технического обслуживания, возле терминалов платных дорог, при автоматической фото- и видеофиксации нарушений на трассах и т. д.

Использование систем идентификации транспорта значительно увеличивает скорость работы многих предприятий, позволяет контролировать движение ТС, повышает уровень безопасности дорожного движения, предотвращает преступность.

На настоящий момент существует несколько алгоритмов распознавания автомобильных номеров, разработаны аппаратные устройства

для использования этих алгоритмов, но из-за высокой стоимости таких продуктов и сложности работы пока не произошло их массового внедрения [2]. Поэтому актуальны задачи разработки новых алгоритмов распознавания графических образов и оптимизации существующих решений.

1 Технологии компьютерного зрения

1.1 Теоретические аспекты

Основы компьютерного зрения – которое также называют техническим зрением и машинным зрением – были заложены еще в середине двадцатого века. Тем не менее, эта область науки и техники считается достаточно новой и активно развивающейся в настоящее время.

Компьютерное зрение (Computer Vision) тесто связано с машинным обучением, компьютерной графикой и обработкой графической информации. Его главной задачей является идентификация объектов на изображениях и на видеопотоке, то есть распознавание образов.

Системы компьютерного зрения представляют собой совокупность технологий и алгоритмов, с помощью которых компьютер обрабатывает изображения и видеофайлы, идентифицирует объекты на изображениях, классифицирует и анализирует их. Такие системы получают изображения и видеофайлы посредством фотокамер, видеокамер, сканеров, рендеринга 2D- и 3D-моделей и т. д.

Общую схему решения задачи машинного зрения можно увидеть на рисунке 1.

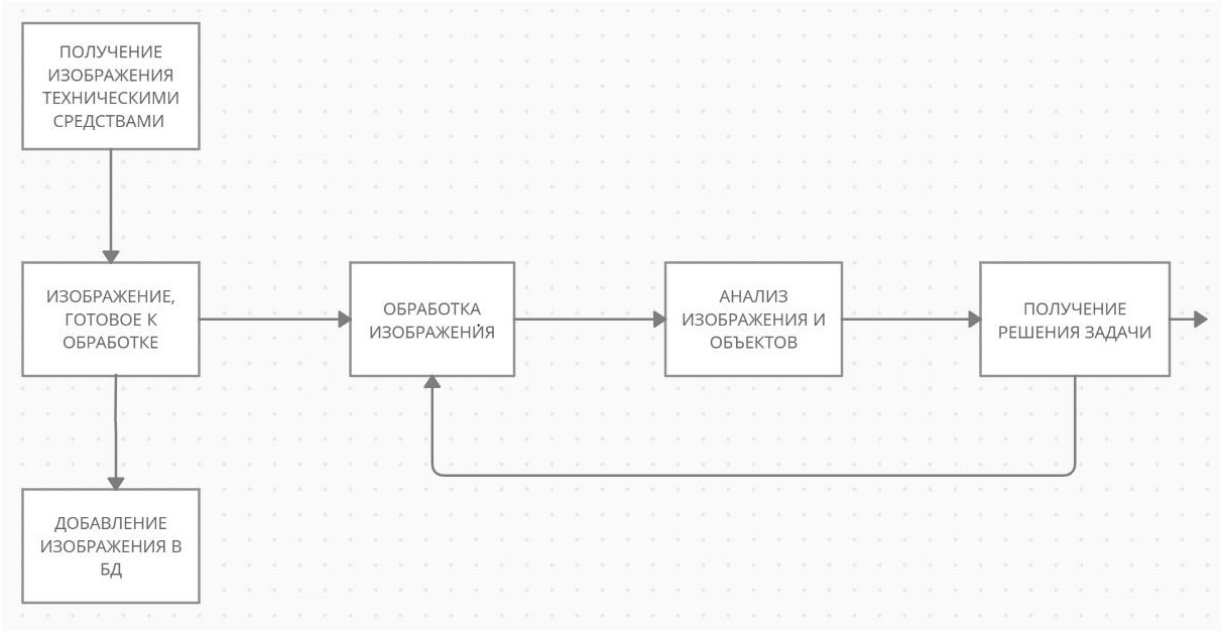

Рисунок 1 – Схема решения задачи машинного зрения

В зависимости от сферы деятельности в блоке схемы «Обработка изображения» будут присутствовать различные алгоритмы. Выбор методов и технологий обработки и анализа графической информации зависит от размера и качества изображений, ракурсов съемки, количества объектов для обнаружения, наличия или отсутствия базы данных и ее объема [3]. В простейшем случае решение задачи может иметь двузначный ответ (например, «да, нет»), а в более сложном – вывод распознанного объекта или множества объектов, классификация объектов, их поиск в базе данных и другое.

Одним из наиболее распространенных классов алгоритмов для разработки систем компьютерного зрения являются искусственные нейронные сети (ИНС), которые рассмотрены в главе 2.

Для работы системы машинного обучения, основанной на технологии нейронных сетей, требуется достаточно мощное оборудование. Организация работы больших нейронных сетей обходится дорого не только с точки зрения бюджета, но и с точки зрения вычислительных ресурсов. Однако в настоящее время активно разрабатываются технологии, которые способны функционировать и на небольшом оборудовании. Например, в 2017 году

компания Movidius выпустила устройство под названием Neural Compute Stick [4]. Это компактное устройство, имеющее размер USB-флэш-накопителя, позволяет ускорять работу и настройку нейронных сетей. Решение Neural Compute Stick повышает производительность работы ИНС в таких задачах, как классификация графических изображений, распознавание объектов. Сейчас оно активно используется в умных домах и в системах фото- и видеонаблюдения.

1.2 Существующие системы распознавания

Решение задачи обнаружения объектов на изображении и видеофайле используется во многих областях деятельности, таких как контроль дорожного движения, охранные системы, промышленность, распознавание текстовых документов, кинематограф, робототехника, медицинские исследования, анализ космических снимков и многих других. Задачи компьютерного зрения являются актуальными, поскольку перечисленные сферы в настоящее время важны и активно развиваются.

В кинематографе, анимации, строительстве и дизайне интерьера компьютерное зрение применяется для генерации 3D-моделей объектов.

В сельском хозяйстве тоже есть место технологиям машинного зрения. Система Taranis в объединении с беспилотным летательным аппаратом способна изучать поля и определять наличие или отсутствие вредителей сельскохозяйственных культур [5].

В промышленности технологии Computer Vision нужны для обеспечения безопасности рабочих. На многих предприятиях проводят идентификацию личности. На опасных производствах системы контролируют ношение рабочими средств индивидуальной защиты – спецодежды, касок, перчаток и др.

При разработке ИИ-решений в области медицины большую роль имеют технологии машинного зрения. Не так давно компания Philips

разработала нейросетевый алгоритм, который позволяет улучшать качество рентгеновских изображений и изображений компьютерной томографии. Он убирает шумы, искажения, дисторсию, что позволяет обследуемым намного меньше времени находиться в томографе и получать меньшую лучевую нагрузку [6].

Сейчас, в связи со сложившейся эпидемиологической ситуацией, во многих учреждениях имеются системы компьютерного зрения, которые контролируют соблюдение людьми масочного режима: они анализируют видеофайлы и работают в режиме реального времени, выявляют нарушения, связанные с отсутствием масок или других средств защиты органов дыхания. Например, в КубГУ при входе в здание университета камеры проверяют наличие маски на лице входящего человека.

Системы распознавания присутствуют даже в каждом современном смартфоне. В смартфонах Pixel и Pixel XL от компании Google есть приложения, позволяющие пользователям обрабатывать изображения во время съемки. Алгоритмы приложения распознают лицо человека, рассчитывают световые потоки и контраст для создания идеальной по цвету и качеству фотографии. Также, основываясь на базе компьютерного зрения, они могут создать объемную модель пространства.

В области контроля дорожного движения тоже используются технологии Computer Vision. В России разработано несколько аппаратно-программных комплексов для распознавания автомобильных государственных номерных знаков (ГНЗ). Самыми распространенными являются «Дигнум Авто», «Автомаршал» и «Neurocore» [7].

«Дигнум Авто» – это система, которая предназначена для считывания регистрационных автомобильных номеров движущихся транспортных средств и проверки этих номеров по базе данных. Система в основном используется предприятиями для контроля въезжающих и выезжающих машин на охраняемую территорию (например, автопарки, стоянки, платные дороги и т. д.).

«Дигнум Авто» включает в себя следующие функциональные возможности:

– распознавание номеров транспортных средств на изображении, видео и в режиме реальной съемки,

– создание базы данных номеров ТС,

– фиксация изображений автомобилей с нераспознанными номерами,

– автоматическая проверка ГНЗ по базе.

«Автомаршал» – это программное обеспечение для видеоконтроля транспорта. Это ПО предназначено для автоматизации работы автобаз, стоянок, заправочных станций, а также для учета транспорта, движущегося по автомагистралям.

Система анализирует видеозаписи с камер и распознает ГНЗ. Она записывает сведения обо всех распознанных ТС в базу данных: дата и время проезда, изображение транспортного средства, его номер и др. Кроме того, система может управлять внешними устройствами, подключенными к ней, – например, шлагбаумами и воротами.

«Neurocore» – это решение по распознаванию автомобильных номеров, созданное на основе нейронных сетей. Программа распознает не только номера, но и вид транспортного средства – легковой, грузовой, гоночный и т. д. Программный комплекс используется для упрощения процесса пропуска транспорта на охраняемую территорию, он также может управлять внешними устройствами. Благодаря использованию технологий нейронных сетей эта программа может распознавать номера ночью, при положении транспортного средства под сложным углом, в ситуациях с низким качеством изображения.

1.3 Проблемы систем распознавания

К основным проблемам систем распознавания объектов на изображениях и видеофайлах относят следующие:

– низкое качество изображения (причинами могут быть сломанная камера, неверные установки съемочной аппаратуры, большое расстояние от камеры до объекта, неправильная обработка фото и др.),

– плохое освещение, наличие теней,

– размытость изображения (эта проблема чаще всего присутствует в системах, которые производят видеофиксацию движущихся объектов),

– наличие предмета, который полностью или частично закрывает собой исследуемый объект.

Помимо перечисленных выше проблем, свойственных всем типам систем распознавания, можно выделить дополнительный ряд проблем, которые встречаются в системах распознавания ГНЗ:

– отличия между номерными знаками разных стран. Эту проблему можно решить расширением базы данных путем добавления в нее примеров номерных пластин других стран,

– наличие прицепа. В этом случае необходимо распознать две номерные пластины – основного транспортного средства и его прицепа,

– наличие буксира (аналогично проблеме выше),

– смена водителем автомобиля полосы движения в момент считывания камерой его ГНЗ. Эта проблема свойственна системам, работающим в режиме реального времени, – например, фото- и видеокамерам, фиксирующим нарушения правил дорожного движения на дорогах,

– закрытие некоторых символов номерного знака грязью, багажом, буксирными крюками и другими предметами.

Чем с большим количеством перечисленных выше проблем справляется система, тем более качественной она является.

2 Методы распознавания графических образов

2.1 Основные методы распознавания образов

Образом, как правило, называют воспроизведение некоторого объекта, или группу элементов, объединенных между собой по какому-либо признаку и обладающих характерными свойствами. Исследуемый объект относят к тому или иному образу по установленным заранее правилам.

Каждый объект может быть рассмотрен как многомерный вектор, координаты которого отражают значения его характеристик.

Сходства и различия между объектами можно определить с помощью понятия расстояния между векторами – метрики. Векторы представляют собой наборы характеристик объектов. Чем меньше расстояние между векторами, тем более похожими являются соответствующие им объекты. Расстояние вычисляется по заранее заданному способу, который зависит от структуры объектов.

В теории распознавания образов выделяют три основные группы методов распознавания:

- методы перебора,
- методы с использованием характеристик объектов,
- методы, использующие нейронные сети.

В методах перебора исследуемый объект сравнивается с образами, хранящимися в базе данных.

Методы с использованием характеристик объектов работают сложнее. При их применении осуществляют анализ образа. Например, при обнаружении объектов на изображении решают задачу определения области, которая занята объектом, задачу выделения краев, выполняют сегментацию изображения, определяют геометрические размеры объекта и другие его характеристики.

Методы, использующие искусственные нейронные сети, считаются одними из самых сложных. Для каждой отдельной задачи необходимо подбирать специальную структуру нейронной сети. Структура зависит от особенностей задачи и от вида требуемого решения. Однако достоинство таких методов заключается в их высокой производительности и эффективности.

2.2 Распознавание по признакам

Распознавание образов по набору их признаков относится к группе методов с использованием характеристик объектов.

Объекты описываются набором признаков (характеристик). Иначе говоря, объект представляется вектором *x*, принадлежащим *n*-мерному пространству, и компоненты этого вектора являются характеристиками объекта.

При распознавании образов на изображениях векторы могут включать в себя:

– геометрические размеры объекта: периметр, площадь, высота, ширина, длина,

– структуру: контур и его размеры, положение краев,

– другие характеристики [3].

Компоненты векторов имеют числовые значения.

Классификатор с помощью такого вектора относит объект *x* к тому или иному классу в соответствии с разбиением *n*-мерного пространства. При этом он использует метрику.

Чаще всего применяют евклидову метрику, вычисляемую по формуле (1):

$$d_E(x_i, x_j) = \sqrt{\sum_k (x_{ik} - x_{jk})^2} \quad (1)$$

где

$x_i,\ x_j$ – векторы;

k – индекс компоненты вектора.

Еще часто используют «манхэттенское расстояние» (расстояние городских кварталов) – метрику, введенную Германом Минковским. Она равна сумме модулей разностей компонент векторов (формула (2)):

$$d_M(x_i, x_j) = \sum_k |x_{ik} - x_{jk}| \quad (2)$$

Также вводят в рассмотрение расстояние Хэмминга, равное числу позиций, в которых векторы различаются. Если векторы *n*-мерные, то очевидно, что расстояние Хэмминга не будет превышать *n* (формула (3)):

$$d_H(x_i, x_j) \leq n \quad (3)$$

Векторы называются соседними, если расстояние Хэмминга между ними равно единице.

2.3 Структурные методы распознавания

В методе распознавания образов по набору их признаков, описанном выше, используются только числовые значения. Для объектов со сложной структурой этого может быть недостаточно. Тогда необходимо к имеющимся характеристикам объекта добавить дополнительные. Такие методы и называются структурными, так как структура объекта не может быть описана

только с помощью числовых параметров. Объект, как правило, разбивают на более простые элементы и определяют отношения между ними.

Сложные образы состоят из более простых подобразов. Описание отношений между подобразами можно представить в виде иерархии, например, в графовой структуре. Вершинами графа являются части и элементы образа, а ребрами – отношения между ними. Отношения могут описываться логическими или математическими функциями.

Можно сказать, что элементы, отношения и ориентированный граф в совокупности образуют один из типов семантических сетей [8] – функциональную сеть. Формат функциональных сетей предполагает использование целостных образов, которые связаны между собой. Связи представлены в виде графа. В графах допускается множественность связей (ребер) между парами вершин.

На рисунке 2 представлен фрагмент семантической сети, соответствующей предметной области «Транспортные средства».

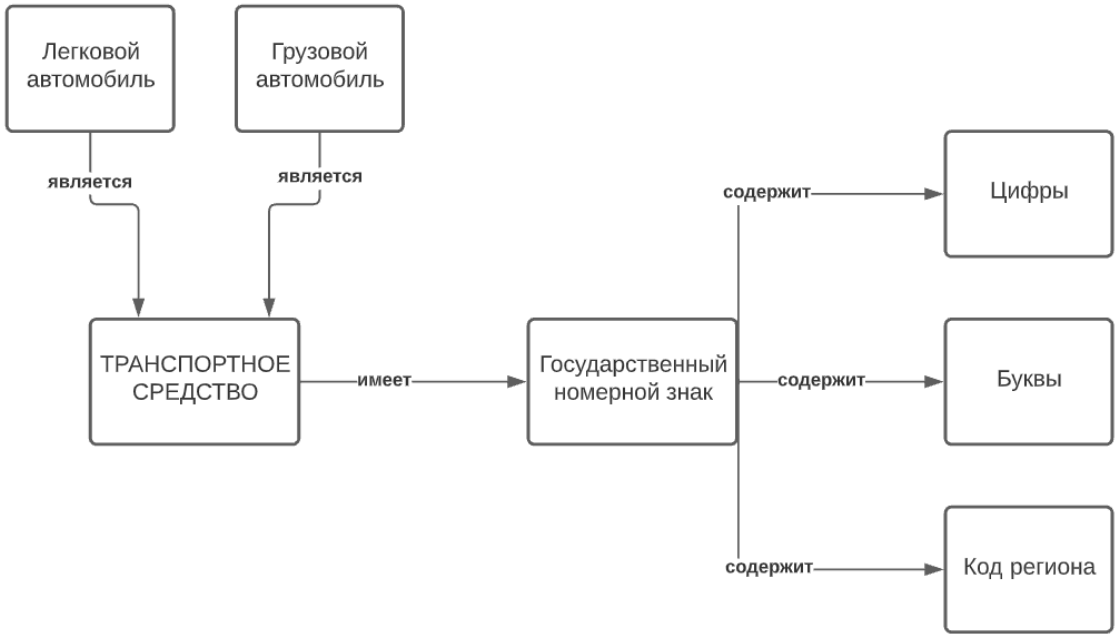

Рисунок 2 – Фрагмент семантической сети

Структурный метод распознавания дает возможность описывать большое количество сложных объектов, используя относительно небольшую базу с несложными образами.

Распознавание проводится в три этапа.

На первом этапе осуществляется подготовка объекта к распознаванию – проверка его наличия на изображении, определение границ и краев.

Далее, на втором этапе, объект представляют в виде структуры: проводится сегментация, объект делится на части – элементы, между ними определяются отношения. Элементы и отношения берутся из базы правил, отражающей семантику предметной области.

На третьем этапе выполняется анализ полученной структуры. На этом шаге принимается решение, является ли представление объекта верным, и в зависимости от результата решения объект либо исключается из рассмотрения, либо определяется в класс образов.

2.4 Статистические методы

Методы, основанные на теореме Байеса, классифицируют объекты, учитывая значение вероятности принадлежности этих объектов к различным классам. В качестве гипотез выступают гипотезы принадлежности объекта к определенному классу.

Теорема Байеса представима в виде формулы (4):

$$P(H_i|D) = \frac{P(D|H_i)P(H_i)}{P(D)} \quad (4)$$

В формуле (4) вероятность $P(H_i|D)$ называется апостериорной. Такие вероятности вычисляются для каждого класса. Апостериорная вероятность является условной и зависит от данных, полученных из опыта.

Вероятность $P(D|H_i)$ называется правдоподобностью получения уже известных данных в случае, если рассматриваемая гипотеза верна.

$P(H_i)$ называется априорным убеждением и выражает вероятность гипотезы до введения в рассмотрение опытных данных.

$P(D)$ – это вероятность данных, которые наблюдаются вне зависимости от гипотезы [9]. Иногда последнюю вероятность не используют, так как вычислять ее затруднительно. Вместо этого сравнивают максимальные величины числителя правой части формулы (4).

Статистические методы применимы в случае, когда все классы обладают собственными значениями признаков принадлежащим им объектов. Если же классы пересекаются в некоторых значениях признаков, то возможны ошибки распознавания.

2.5 Искусственные нейронные сети

2.5.1 Концептуальная модель

Одним из наиболее часто используемых классов алгоритмов для решения задач машинного зрения являются нейронные сети.

Искусственная нейронная сеть (ИНС, НС) – это параллельно распределенная структура обработки информации, которая состоит из нейронов, связанных между собой. Модель нейронной сети в программировании является машинной интерпретацией головного мозга. Центральная нервная система человека представлена спинным и головным мозгом, функциональность которых осуществляется при помощи огромного количества нейронов, которые связаны между собой синаптическими связями и могут передавать информацию посредством импульсов.

Персептрон – это математическая модель восприятия информации человеческим мозгом, предложенная Ф. Розенблаттом в середине прошлого

века [14]. Перцептрон стал одной из первых моделей нейронных сетей, которую впоследствии стали активно развивать и улучшать.

НС обладают способностью к самообучению. Иначе говоря, они могут выдавать результат на основе полученного опыта, обобщая имеющиеся прецеденты на новые случаи.

ИНС состоят из входного, скрытых и выходного слоев (рисунок 3). Каждый слой содержит определенное количество нейронов.

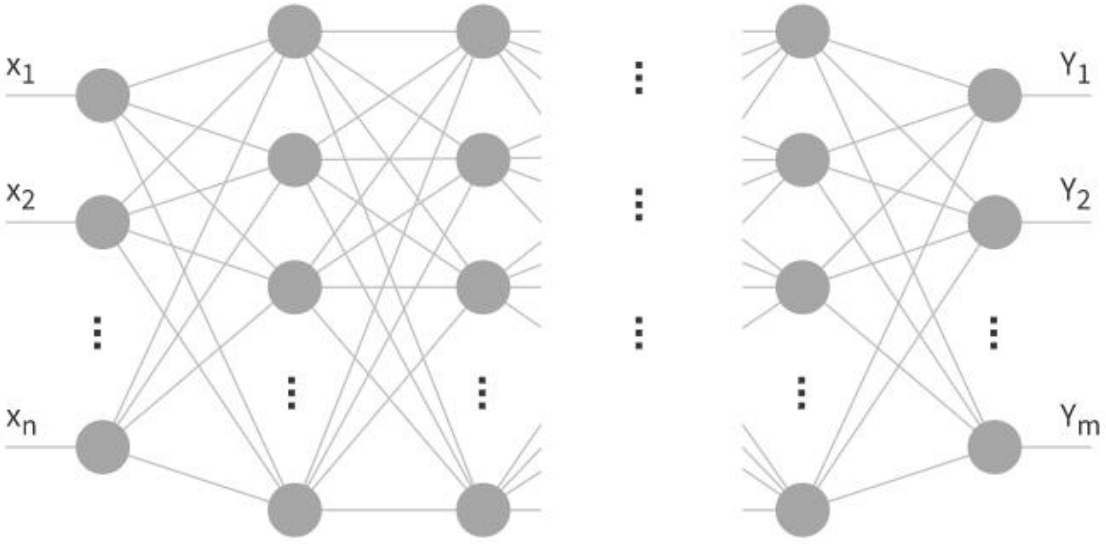

Рисунок 3 – Слои нейронной сети

Процесс обучения нейронной сети является итеративным, его шаги называют эпохами.

Для обучения НС чаще всего используют методы обучения с учителем, например, метод обратного распространения ошибки и его модификации. Нередко используют и обучение без учителя. Тип обучения главным образом зависит от решаемой задачи.

Обучение с учителем происходит при наличии полного набора размеченных данных. В обучающем наборе каждому примеру соответствует решение, которое сеть должна получить. Разность между правильным

решением и полученным представляет собой ошибку, которую необходимо устранять с помощью новой настройки параметров.

Обучение без учителя проводится без контроля разработчика над процессом. На вход подается набор данных, и нейронная сеть пытается самостоятельно найти взаимосвязи. При обучении без учителя у разработчика отсутствуют правильные решения, которые сеть должна получить.

Одним из достоинств НС является решение задач в условиях неопределенности. Способность к самообучению позволяет сетям искать решение задач с неизвестными закономерностями и зависимостями между входными и выходными данными.

Сверточная искусственная нейронная сеть (СИНС, СНС) – это архитектура искусственных нейронных сетей, которая входит в состав технологий глубокого обучения. Такие сети нацелены на распознавание образов. Их модель в программировании основана на особенностях работы клеток зрительной коры головного мозга.

Для решения задач компьютерного зрения широко используется технология сверточных нейронных сетей. Преимуществом такого подхода является то, что такие сети являются одним из лучших алгоритмов по распознаванию и классификации изображений, устойчивы к повороту и сдвигам изображений, имеют гораздо меньше настраиваемых весов по сравнению с обычной нейронной сетью. Еще одним достоинством является возможность реализации параллельно работающих алгоритмов [3], что позволяет повысить производительность системы и увеличить скорость вычислений.

Сверточные сети работают по тому же принципу, что и обычные нейронные сети, только помимо умножения матриц в них присутствует операция свертки. Свертка – это линейная операция, применяющаяся к двум функциям вещественного аргумента и возвращающая третью, которая

характеризует сходство одной функции с отраженной и сдвинутой копией другой функции [14].

Еще одно отличие сверточных сетей – это разреженная связность [10].

В обычном многослойном персептроне каждый нейрон следующего слоя связан со всеми нейронами предыдущего слоя, и все связи имеют свои веса в матрице весов. А в СНС используется небольшая матрица весов, которая на каждом шаге движется по обрабатываемому слою.

Матрицу весов называют ядром свертки, в сети ядер несколько. Ядра кодируют наличие каких-либо признаков на изображении – горизонтальных и вертикальных линий, линий под углом, дуг, сложных фигур (эллипсов, треугольников, квадратов и других). Следующий слой, формирующийся в результате свертки одной из матриц весов, будет отражать существование определенного признака в обработанном слое. Таким образом формируется карта признаков, представляющая собой массив матриц.

Каждая карта содержит в себе синаптическое ядро или фильтр. Фильтр показывает наличие определенного признака. Проход каждым набором весов составляет свой экземпляр карты признаков, делая нейронную сеть многоканальной (то есть много независимых карт признаков на одном слое) [10].

При выполнении операции свертки окно с размерностью ядра проходит с заданным шагом всю область изображения, на каждом шаге прохода умножает содержимое окна на ядро, результат суммирует и записывает в матрицу результата – очередную карту признаков. Размеры карт признаков одного сверточного слоя одинаковы.

Операция подвыборки или пулинг (англ. Pooling) необходима для уменьшения масштаба карт признаков. Чаще всего используется операция MaxPooling – отбор наибольших значений: из нескольких соседних нейронов карты выбирают максимальный и принимают его за один нейрон новой карты признаков меньшей размерности. Благодаря этому можно значительно

снизить объем используемой памяти и ускорить процесс дальнейших вычислений.

Описанную выше процедуру в графическом представлении можно увидеть на рисунке 4.

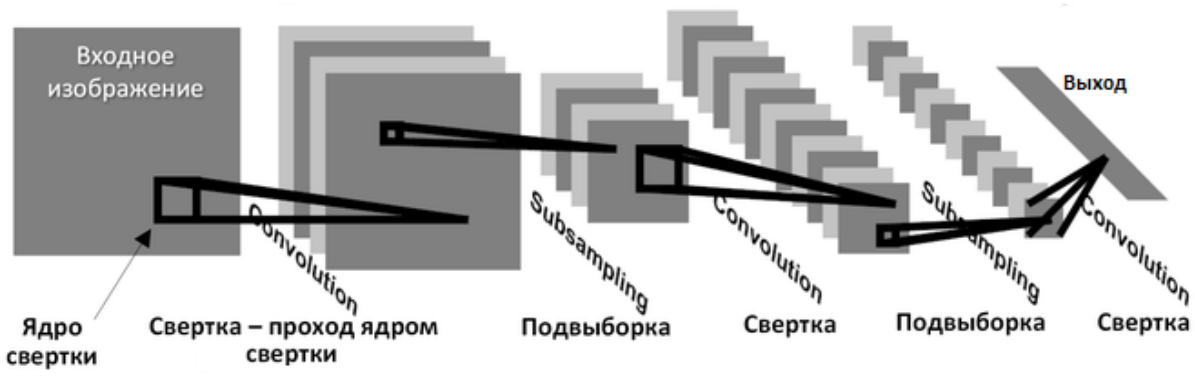

Рисунок 4 – Схема работы сверточной сети

Сигнал проходит через некоторое количество слоев (на рисунке: один входной слой, представляющий собой непосредственно само входное изображение, четыре скрытых слоя, один выходной слой), где чередуются свертка и подвыборка (пулинг). При этом составляются карты признаков. Их количество увеличивается с каждым слоем, но одновременно уменьшается разрешение карт [11].

На выходе получается большой набор каналов, содержащих небольшое количество данных, которые отражают признаки, выявленные на входном изображении. Затем, как правило, эти данные передаются на обычную полносвязную нейронную сеть, которая формирует конечный выходной сигнал. В итоге исходное изображение проходит через множественную фильтрацию.

Ядра свертки формируются с помощью обучения нейронной сети методом обратного распространения ошибки. Процессы свертки, выполняемые по каждой карте признаков, происходят параллельно, что является существенным преимуществом с точки зрения времени работы сети.

Количество карт определяется требованиями к ответу решаемой задачи. Если будет большое число карт, то улучшится качество распознавания, но в то же время повысится сложность вычислений. Чаще всего берется соотношение 1:2 – на одну карту предыдущего слоя приходится две карты нового слоя.

Недостатком СНС является наличие большого количества варьируемых параметров – количество слоев, количество и размерность ядер свертки, шаг сдвига ядра при обработке слоя, параметры подвыборок, а также параметры выходной полносвязной сети.

2.5.2 Функции активации нейронов

В нейросетях функции активации (ФА) определяют выходной сигнал нейрона в зависимости от входного сигнала или набора входных сигналов.

Выделяют три основных типа функций активации:

– пороговые функции или функции единичного скачка,

– кусочно-линейные функции,

– сигмоидальные функции.

Пороговая функция в простой форме является двоичной: нейрон либо возбужден, либо нет. Такая функция представляет собой ступенчатую функцию Хэвисайда и вычисляется по формуле (5). Она подходит для бинарной классификации.

$$\varphi(x) = \begin{cases} 1, x \geq 0; \\ 0, x < 0 \end{cases} \quad (5)$$

Кусочно-линейная функция в общем случае представляется в виде формулы (6):

$$\varphi(x) = \begin{cases} 1, x \geq 0{,}5; \\ |x|, -0{,}5 < x < 0{,}5; \\ 0, x \leq -0{,}5 \end{cases} \quad (6)$$

Но чаще используют функцию ReLU, так называемую выпрямленную линейную единицу (формула (7)):

$$f(x) = \max(0, x) = \begin{cases} x, x \geq 0; \\ 0, x < 0 \end{cases} \quad (7)$$

Функция ReLU имеет несколько преимуществ: быстрое вычисление производной, легкое оптимизирование. Но также у нее есть недостаток, который называют проблемой умирающего ReLU: если нейрон стал отрицательным, то он не восстановится и, соответственно, обучаться не будет. В таком случае обычно увеличивают количество нейронов. Функция ReLU и ее модификации являются наиболее широко используемыми в сверточных нейронных сетях.

Сигмоидальная функция – это непрерывная и дифференцируемая функция, которая поддерживает баланс между линейным и нелинейным поведением [11]. На входе она принимает вещественное число от минус бесконечности до плюс бесконечности, на выходе выдает вещественное число в диапазоне от 0 до 1. Сигмоида имеет вид формулы (8). Несмотря на полезные свойства непрерывности и дифференцируемости, она имеет недостаток, заключающийся в том, что при насыщении функции со стороны нуля или единицы градиент будет близок к нулю.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (8)$$

Функция Softmax (взвешенная сигмоида) используется для множественной классификации. Такая функция нормализует N-мерный вектор произвольных значений в N-мерный вектор, в котором каждая координата принимает значение в диапазоне от 0 до 1. Как правило, Softmax применяют в последнем слое нейронной сети.

Координаты $j = \overline{1,N}$ вычисляются по формуле (9). Координата j полученного вектора отражает вероятность принадлежности объекта к классу с номером j.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{n=1}^{N} e^{z_n}} \quad (9)$$

2.5.3 LeNet

LeNet – это первая структура сверточной нейронной сети, разработанная Янном ЛеКуном в конце прошлого века [14]. Тогда, при крайне ограниченных ресурсах, она позволила построить и успешно обучить НС, распознающую и классифицирующую черно-белые очертания цифр. Это был прорыв в распознавании цифр и букв.

Сеть состоит из 7 слоев. На входе – черно-белое изображение цифры размером 32 на 32 пикселя. Сверточный слой состоит из 6 фильтров 5×5, они выделяют на изображении простые формы (ломаные, полосы и др.). После применения фильтров получается 6 проекций 28×28 (шаг свертки равен 1, формула вычисления размера выходного изображения: 32 минус 5 плюс 1 равно 28), то есть размер исходного изображения уменьшается. Далее применяется подвыборка MaxPooling 2×2. Размерность изображений снова уменьшается и становится 14×14. Далее идет второй слой свертки – 16 ядер 5×5. Слой подвыборки аналогичен предыдущему. Размерность становится 10×10.

Следующие два слоя – обычные слои: 120 нейронов с функцией активации tanh() и 84 нейрона с этой же функцией. Выходной (последний) слой имеет 10 нейронов, так как классифицирует 10 классов цифр.

На оборудовании прошлого века сеть обучалась несколько часов, обучение прошло успешно.

Считается, что LeNet положила начало сверточным НС.

2.5.4 AlexNet и VGG

LeNet и ее модификации – способные модели, их легко обучать, они имеют высокую производительность. Но эти НС нельзя применять к изображениям с высоким разрешением.

В 2012 году была опубликована архитектура СНС AlexNet, разработанная Алексом Крижевским в сотрудничестве с Ильей Суцкевером и Джеффри Хинтоном [14]. Ее схема изображена на рисунке 5.

Эта сеть позволяет работать с большими полноцветными изображениями.

AlexNet с большим отрывом выиграла конкурс по распознаванию изображений ImageNet Large Scale Visual Recognition Challenge с количеством ошибок 15,3 процентов против 26,1 процентов у другого решения, занявшего второе место.

Архитектура содержит 8 слоев. Первые 5 являются сверточными, в них в качестве функции активации используется ReLU. За счет использования этой ФА скорость работы увеличилась в 6 раз.

Архитектуру AlexNet используют до сих пор, например, она реализована в Python-библиотеках глубокого обучения TensorFlow и Keras.

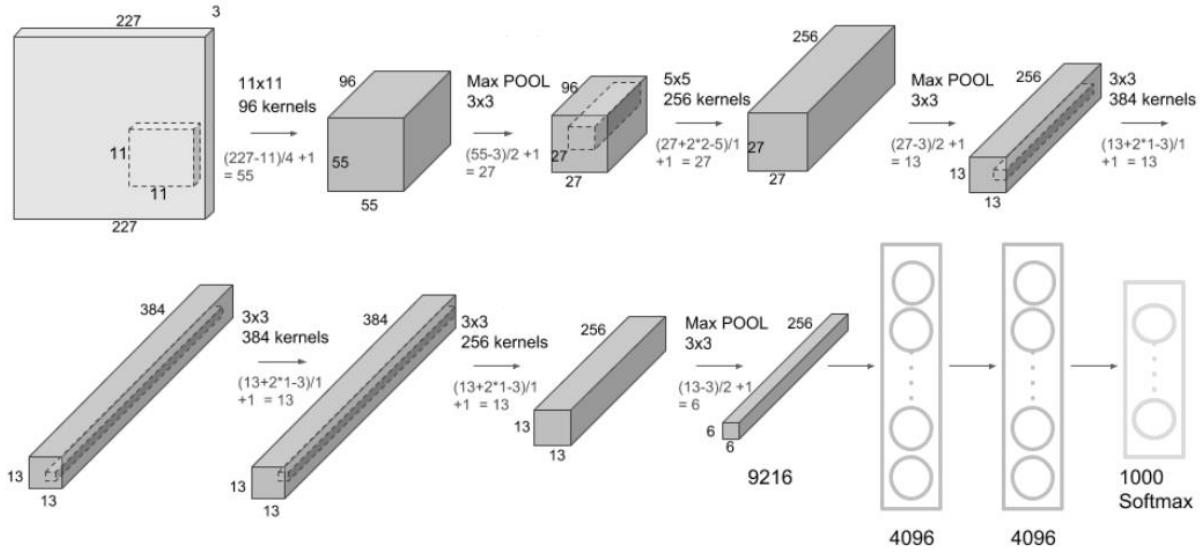

Рисунок 5 – Сеть AlexNet

Исходное изображение имеет разрешение 227×227 пикселей. Имеется 3 цветовых канала: красный, зеленый, голубой (RGB), на каждый канал накладывается фильтр размера 11×11. Размерность в результате работы слоев меняется следующим образом: 227×227, 55×55, 27×27, 13×13, 6×6.

После 5 сверточных слоев идут 2 полносвязных слоя, содержащих по 4096 нейронов. Выходной слой, использующий ФА Softmax, способен классифицировать объекты на 1000 классов.

Архитектура AlexNet послужила основой для многих современных сверточных сетей. Концепцию архитектуры стали активно развивать.

В 2014 году в качестве улучшения архитектуры AlexNet была предложена сеть VGG. Ее разработала группа компьютерного зрения Оксфордского университета (Visual Geometry Group) вместе с исследователями из Google DeepMind [14].

В этой сети большие фильтры заменены на некоторое количество маленьких. Вместо фильтров 11×11 и 5×5 в первых двух сверточных слоях несколько раз используются фильтры размера 3×3. После каскада сверток используется та же конфигурация полносвязных слоев, что и в сети AlexNet.

Благодаря введенному новшеству удалось уменьшить количество весов в сети. VGG существенно превзошла предыдущие поколения моделей в производительности.

2.5.5 ResNet

ResNet – это сокращение от Residual Network (остаточная сеть) [14]. Такая сеть позволяет решить проблему ухудшения точности предсказания с увеличением глубины структуры сети.

Проблема исчезающего градиента состоит в том, что на каком-то этапе обучения сети он становится очень малым, что не позволяет изменяться весам. В остаточных нейросетях есть ярлыки соединения, в основе работы которых лежит отображение идентичности. Сеть позволяет на некоторых этапах обучения пропускать слои. Также у ResNet нет полносвязных слоев в конце, присутствует только один слой, предсказывающий выходные классы. Обучение длится примерно 60 000 эпох.

В результате экспериментов с ResNet было установлено, что нейросети с большой глубиной можно обучать без ухудшения точности.

3 Программная реализация алгоритма

3.1 Постановка задачи

Практической задачей работы являлась разработка алгоритма распознавания символов на государственных номерных знаках, использующего технологию сверточных нейронных сетей.

Для создания и обучения нейронной сети было необходимо определиться с инструментами разработки, выбрать оптимальную структуру сети, настроить ее параметры.

При решении задачи была использована структура СНС LeNet.

Для обучения сети был найден набор данных с различными изображениями 21 символа – 10 цифр и 11 букв. Нейронная сеть обучилась на этом наборе данных распознавать отдельные символы на ГНЗ.

Идея алгоритма заключалась в следующем. Алгоритм получает на вход изображение номерного знака транспортного средства, затем разбивает изображение на сегменты – то есть, каждый символ номерного знака становится отдельным изображением. Далее происходит работа нейронной сети, которая распознает каждый выделенный символ отдельно, после чего алгоритм собирает в одну строку распознанные символы номера транспортного средства и выдает эту строку в качестве ответа на поставленную задачу.

3.2 Средства разработки решения

Основным инструментом разработки был выбран язык программирования Python. Он является высокоуровневым, объектно-ориентированным языком, включает возможность интегрироваться с другими языками программирования. Его интерпретатор имеет открытый

исходный код. В настоящее время язык широко используется в сфере анализа данных, машинном обучении, веб-разработке и других областях.

В качестве среды разработки была выбрана интерактивная среда Jupyter Notebook, в которой можно сразу видеть результаты выполнения кода и его отдельных частей. В Jupyter код можно разделять на фрагменты и выполнять их в произвольном порядке. Еще одно преимущество – код программы может объединяться с документацией и изображениями.

Версия 3 языка Python предоставляет большое количество библиотек для специалистов Data Science. Наиболее популярными являются следующие:

– numpy. Она предоставляет реализации математических операций и алгоритмов, скомпилированных на языке программирования C. Ядром библиотеки является многомерный массив данных, с которым и работает программист [12];

– matplotlib. Эта библиотека является средством визуализации данных. С ее помощью можно формировать графики и диаграммы, двумерную и трехмерную графику, статичные и анимированные изображения [13];

– PIL (Python Imaging Library). Эта библиотека создана для работы с растровой графикой, она поддерживает многие распространенные форматы: BMP, JPEG, PNG, GIF и другие. Полезной возможностью библиотеки является преобразование изображений – масштабирование, применение фильтров, рисование и т. д.;

– keras. Она является популярным инструментом машинного обучения. Библиотека поставляется с различными наборами данных, которые можно найти в репозитории GitHub;

– opencv (Open-Source Computer Vision Library). Эта библиотека хранит алгоритмы обработки изображений.

3.3 Набор данных для обучения НС

Для решения задачи распознавания символов на автомобильных номерах был использован набор данных Train_Cells. Источником данных выступил ресурс Kaggle, принадлежащий корпорации Google.

В этом наборе цифры и буквы, вырезанные с изображений российских государственных номеров транспортных средств, разложены по отдельным папкам в формате BMP. Формат BMP – это растровый формат, который реализует сжатие изображения без потерь.

Этот набор данных выполнил роль тренировочной коллекции – коллекции объектов, для которых известны их образы. Часть обучающего набора представлена на рисунке 6:

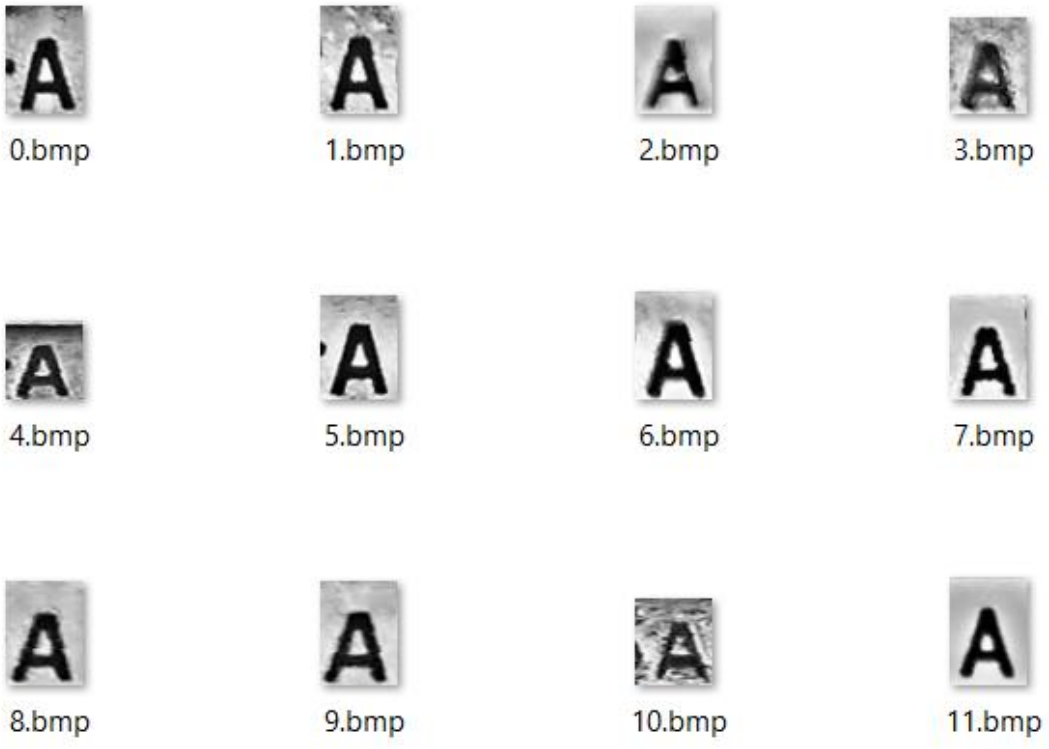

Рисунок 6 – Часть обучающего набора

Вид регистрационных номерных знаков Российской Федерации определен ГОСТ Р 50577–93. Структура ГНЗ представлена на рисунке 7.

Знаки маршрутных ТС, военных ТС, прицепов, мотоциклов и некоторых других ТС имеют формат, немного отличающийся от стандартного.

Комбинации на стандартных ГНЗ состоят из 3 букв и 3 цифр. ГОСТом разрешены 12 букв кириллицы, у которых есть аналоги в латинском алфавите: A, B, C, E, K, M, H, O, P, T, У, X. Буквы означают серию номерного знака, а цифры – его номер.

Формат имеет вид: 1 буква, 3 цифры, 2 буквы.

Рисунок 7 – Формат ГНЗ

Важно то, что в обучающем наборе данных содержались не снимки номерных пластин, а изображения отдельных символов. Нейронная сеть распознает каждый символ отдельно, а затем собирает их воедино – в распознанный автомобильный номер.

3.4 Структура алгоритма

Алгоритм распознавания государственного номерного знака включает в себя следующие шаги:

1) Обучение нейронной сети.

На первом этапе происходит обучение сверточной нейронной сети на наборе данных Train_Cells. НС учится распознавать буквы или цифры на изображении. Этот шаг, как правило, выполняется один раз.

2) Получение снимка ГНЗ.

Алгоритм получает изображение номерного знака транспортного средства.

3) Разбиение ГНЗ на отдельные символы.

При помощи средств библиотеки OpenCV изображение номерного знака разбивается на отдельные части. Алгоритм находит контуры каждой буквы и каждой цифры. Далее, каждый символ сохраняется как отдельное изображение. Более подробно метод сегментации описан ниже.

4) Распознавание цифр и букв нейронной сетью.

На вход сверточной нейронной сети последовательно подаются изображения символов номерного знака, полученные на шаге 3. НС распознает каждый символ отдельно и возвращает результаты. Предсказания объединяются в одну строку, которая выдается в качестве ответа.

Подробное описание топологии нейронной сети представлено ниже, в пункте 3.6.

3.5 Сегментация изображений

Для сегментации снимка номерного знака ТС использовались некоторые средства библиотеки OpenCV.

Thresholding (Порог) – это метод сегментации, который применяется для создания бинарных изображений [14].

Thresholding делится на два типа: простой и адаптивный порог. В программе был реализован адаптивный порог. Для того, чтобы отделить объекты (символы) от фона, нужно задать отбор пикселей выше или ниже определенного порогового значения. Так как фон номерной пластины не всегда является чисто белым, то в качестве порога был выбран сероватый цвет.

Перед процессом сегментации изображение переводилось в черно-белое. Далее, с помощью методов библиотеки OpenCV были найдены контуры букв и цифр и переведены в растровый формат. Символы готовы к распознаванию. Изображение каждого символа было сохранено в директорию, из которой нейронная сеть позже будет получать входные данные.

Программный код процедуры сегментации изображения представлен в приложении А.

3.6 Архитектура нейронной сети

Реализованный в работе алгоритм распознавания символов основан на архитектуре нейронной сети LeNet.

Схему сети можно увидеть на рисунке 8:

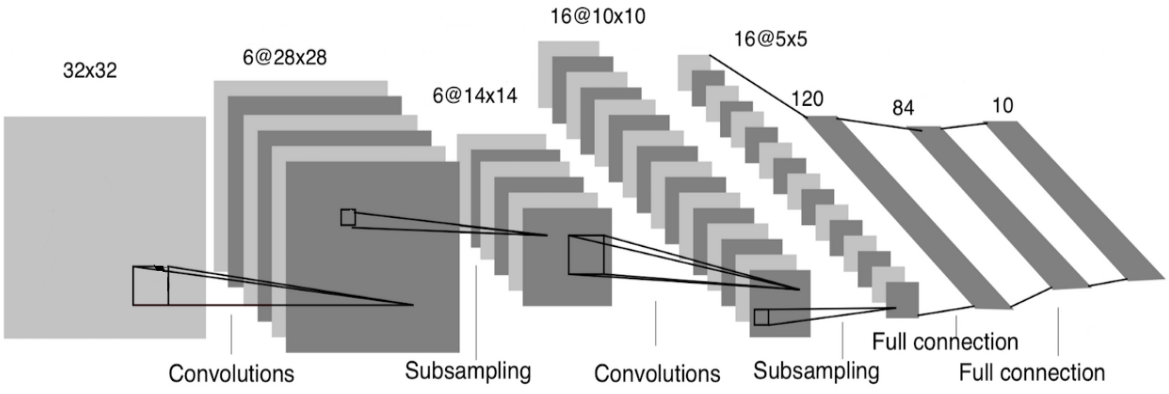

Рисунок 8 – Сеть LeNet

При помощи средств библиотеки Keras были созданы генераторы изображений, которые получали тренировочные данные из набора Train_Cells.

Для обучения нейронной сети было решено использовать не только исходные изображения из набора, но и преобразованные. Преобразования включали в себя изменение размера картинки, случайное увеличение некоторых сегментов, повышение контрастности, изменение яркости. Таким образом было увеличено количество тренировочных изображений.

Далее, с использованием объекта из библиотеки Keras была построена модель сверточной нейронной сети, согласующаяся со схемой на рисунке 8.

В сверточных слоях использовался метод инициализации весов, соответствующий нормальному распределению. На первом и втором полных (обычных) слоях сети использовалась функция активации нейронов ReLU, а на третьем, последнем слое, – функция Softmax (взвешенная сигмоида).

Ранее были рассмотрены такие проблемы функций активации, как проблема умирающего ReLU (если нейрон стал отрицательным, то он обучаться не будет) и проблема затухающего градиента (при насыщении сигмоидальной функции со стороны нуля или единицы градиент приближается к нулю). Чтобы уменьшить вероятность появления этих проблем, было решено оптимизировать нейронную сеть.

Оптимизационный алгоритм Adam (Adaptive Moment Estimation) – это расширение метода градиентного спуска. Он адаптирует скорость обучения для каждой входной переменной целевой функции, рассчитывает размер шага для каждого входного параметра (на основе градиента), реализует идею накопления движения. Для того, чтобы выяснить, как часто изменяется градиент, используется оценка дисперсии.

Алгоритм Adamax – это разновидность алгоритма Adam. В нем вместо дисперсии оценивается инерционный момент распределения градиентов. В программе был использован оптимизатор Adamax, содержащийся в библиотеке Keras.

После построения модели НС было проведено ее обучение. Сначала сеть обучалась на исходных изображениях из тренировочного набора, затем – на измененных. В обоих случаях количество эпох обучения равнялось одной сотне. После обучения нейронная сеть готова к использованию. На вход ей последовательно подавались изображения символов с номерного знака, полученные путем сегментации входного снимка. Сеть распознала каждый символ отдельно.

Экспериментальным путем было выяснено, что при добавлении изображению символа контрастности его распознавание происходило более уверенно.

Программный код модели, обучения и работы сверточной нейронной сети можно увидеть в приложении А работы.

3.7 Пример работы алгоритма

Для тестирования работы алгоритма ему на вход был подан снимок номерной пластины, представленный на рисунке 9:

Рисунок 9 – Снимок номерного знака

Алгоритм сегментации выделил следующие символы на входном изображении (рисунок 10):

Рисунок 10 – Контуры символов

На рисунке ниже представлены изображения символов (с добавленной контрастностью и измененным размером) и результат работы нейронной сети – распознанный номер транспортного средства.

Рисунок 11 – Результат работы нейросети

Как видно из средней части рисунка 11, построенная нейронная сеть правильно распознала все символы на входном изображении номерной пластины.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы были изучены основные методы распознавания объектов на изображении, рассмотрены особенности проектирования и обучения искусственных сверточных нейронных сетей, сферы их применения и часто встречающиеся проблемы. Впоследствии была реализована процедура сегментации изображения, выбрана топология и обучена сверточная нейронная сеть для распознавания символов государственного номерного знака транспортного средства на этом изображении.

При разработке программы использовался язык программирования Python и средства библиотеки Keras. Для сегментации изображения был выбран метод Thresholding и использовались инструменты библиотеки OpenCV. Для построения сверточной нейронной сети была выбрана архитектура LeNet, оптимизированная посредством алгоритма Adamax. Для обучения сети использовалась выборка, состоящая из 20 000 изображений. Результатом работы программы является регистрационный номер транспортного средства, распознанный на входном изображении.

Использование сверточных нейронных сетей позволяет строить качественные алгоритмы распознавания объектов на фотографиях и видеофайлах. Как правило, структуру СНС подбирают в соответствии с особенностями задачи и с требованиями к ее решению. Ориентация на узкоспециализированную задачу позволяет получать точные и корректные решения. Также сверточные сети отличаются высокой производительностью и эффективностью. В настоящее время сфер их применения становится все больше и больше.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Аналитическое агентство «АВТОСТАТ» [Электронный ресурс]. – Режим доступа: http://www.autostat.ru/ (дата обращения: 10.05.2022).
2. Сиднев А. А. Проектирование и программная реализация информационной системы компьютерного зрения по распознаванию номеров автотранспортных средств / А. А. Сиднев // Евразийский Научный Журнал. – 2017. – № 11. – С. 1–5.
3. Селянкин В. В. Компьютерное зрение. Анализ и обработка изображений / В. В. Селянкин. – М.: Лань, 2019. – 152 с.
4. GreenTechReviews [Электронный ресурс]. – Режим доступа: http://greentechreviews.ru/ (дата обращения: 10.05.2022).
5. Полымя Агро [Электронный ресурс]. – Режим доступа: http://polymya-agro.by/news/ (дата обращения: 05.05.2022).
6. Новые медицинские технологии [Электронный ресурс]. – Режим доступа: http://evercare.ru/news/ (дата обращения: 05.05.2022).
7. ИНТЕМС [Электронный ресурс]. – Режим доступа: http://securityrussia.com/ (дата обращения: 05.05.2022).
8. Костенко К. И. Вложения формализмов семантических сетей / К. И. Костенко // Экологический вестник научных центров Черноморского экономического сотрудничества. – 2013. – Т. 10. – № 2. – С. 58–66.
9. Курт У. Байесовская статистика / У. Курт. – М.: Питер, 2021. – 304 с.
10. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение / Я. Гудфеллоу, И. Бенджио, А. Курвилль. – М.: ДМК Пресс, 2018. – 652 с.
11. Хайкин С. Нейронные сети: полный курс / С. Хайкин. – М.: Издательский дом Вильямс, 2006. – 1104 с.
12. Хилл К. Научное программирование на Python / К. Хилл. – М.: ДМК-Пресс, 2021. – 646 с.

13. Абдрахманов М. И. Devpractice Team. Python. Визуализация данных / М. И. Абдрахманов. – М.: Devpractice, 2021. – 412 с.

14. MachineLearning [Электронный ресурс]. – Режим доступа: http://www.machinelearning.ru/ (дата обращения: 15.04.2022).

15. Документация библиотеки NumPy [Электронный ресурс]. – Режим доступа: http://numpy.org/ (дата обращения: 05.05.2022).

16. Документация библиотеки Matplotlib [Электронный ресурс]. – Режим доступа: http://matplotlib.org/ (дата обращения: 05.05.2022).

17. Документация библиотеки Python Imaging Library [Электронный ресурс]. – Режим доступа: http://pypi.org/project/Pillow/ (дата обращения: 05.05.2022).

18. Документация библиотеки Keras [Электронный ресурс]. – Режим доступа: http://keras.io/ (дата обращения: 05.05.2022).

19. Документация библиотеки OpenCV [Электронный ресурс]. – Режим доступа: http://opencv.org/ (дата обращения: 05.05.2022).

ПРИЛОЖЕНИЕ А

Исходный код

```
import keras as K
from PIL import ImageEnhance
import matplotlib.pyplot as plt
import cv2

TrainingGenerator =
K.preprocessing.image.ImageDataGenerator().flow_from_directory(
   'Train_Cells', target_size=(48, 48), batch_size=32, shuffle=True)
ValidationGenerator =
K.preprocessing.image.ImageDataGenerator().flow_from_directory(
   'Train_Cells', target_size=(48, 48), batch_size=32, shuffle=True)
AugGenerator = K.preprocessing.image.ImageDataGenerator(
   width_shift_range=0.1,
   height_shift_range=0.1,
   zoom_range=0.05,
   brightness_range=(0.5,2.0)).flow_from_directory(
   'Train_Cells', target_size=(48, 48), batch_size=32, shuffle=True)
Model = K.Sequential()
Model.add(K.layers.Conv2D(filters=64, kernel_size=(3,3),
                  activation="relu",input_shape=(48, 48, 3),
                  kernel_initializer="he_normal"))
Model.add(K.layers.Conv2D(filters=64,kernel_size=(3,3), activation="relu",
                  kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.AveragePooling2D())
Model.add(K.layers.Conv2D(filters=128, kernel_size=(3,3),
                  activation="relu", kernel_initializer="he_normal"))
```

```
Model.add(K.layers.Conv2D(filters=128, kernel_size=(3,3),
                 activation="relu", kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.AveragePooling2D())
Model.add(K.layers.Flatten())
Model.add(K.layers.Dense(units=120, activation="relu",
                 kernel_regularizer=K.regularizers.l1_l2(l1=1e-4, l2=1e-5),
                 kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.Dropout(0.2))
Model.add(K.layers.Dense(units=84, activation="relu",
                 kernel_regularizer=K.regularizers.l1_l2(l1=1e-4, l2=1e-5),
                 kernel_initializer="he_normal"))
Model.add(K.layers.BatchNormalization())
Model.add(K.layers.Dropout(0.2))
Model.add(K.layers.Dense(units=21, activation="softmax"))
Model.compile(optimizer=K.optimizers.Adamax(learning_rate=1e-2),
   loss=K.losses.categorical_crossentropy)
Model.fit_generator(
   TrainingGenerator,
   steps_per_epoch=480,
   epochs=100,
   validation_data=ValidationGenerator,
   validation_steps=96)
Model.fit_generator(
   AugGenerator,
   steps_per_epoch=480,
   epochs=100,
   validation_data=ValidationGenerator,
   validation_steps=96)
```

```
def Image_Segmentation(Image_Name: str, out_size=48):
    IMAGE = cv2.imread(Image_Name)
    IMAGE_Gray_Colour = cv2.cvtColor(IMAGE, cv2.COLOR_BGR2GRAY)
    _, Threshold = cv2.threshold(IMAGE_Gray_Colour, 125, 255,
cv2.THRESH_BINARY)
    IMG_Erode = cv2.erode(Threshold, np.ones((3, 3), np.uint8), iterations=1)
    Contours, Hierarchy = cv2.findContours(IMG_Erode, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
    IMAGE_Copy = IMAGE.copy()
    Character_Images = []
    for i, contour in enumerate(Contours):
        (x, y, w, h) = cv2.boundingRect(contour)
        if Hierarchy[0][i][3] == 0:
            cv2.rectangle(IMAGE_Copy, (x, y), (x + w, y + h), (70, 0, 0), 1)
            cropped_character = IMAGE_Gray_Colour[y:y + h, x:x + w]
            size_max = max(w, h)
            square_character = 255 * np.ones(shape=[size_max, size_max],
dtype=np.uint8)
            if w > h:
                y_pos = size_max//2 - h//2
                square_character[y_pos:y_pos + h, 0:w] = cropped_character
            elif w < h:
                x_pos = size_max//2 - w//2
                square_character[0:h, x_pos:x_pos + w] = cropped_character
            else:
                square_character = cropped_character
            Character_Images.append((x, w, cv2.resize(square_character, (100, 150),
interpolation=cv2.INTER_AREA)))
    Character_Images.sort(key=lambda x: x[0], reverse=False)
    for j in range(len(Character_Images) - 1):
```

```
        One_Character = Image.fromarray(Character_Images[i][2])
        One_Character.save(f"Symbols_for_NNW\symbol_{i}.bmp")
    return Character_Images

Image_Segmentation("test_number.png")
CarNumberSymbols = "Symbols_for_NNW"
PredictionCarNumber = ""
AllLetters = "ABCEMKMPTYX"
FIGURE = plt.figure(figsize=(16,8))
SubPlots = FIGURE.subplots(1, 6)
for s in SubPlots:
    s.axis("off")
i = 0
for file in os.listdir(CarNumberSymbols):
    image = Image.open(os.path.join(CarNumberSymbols, file))
    image = image.resize((48, 48), Image.BICUBIC)
    ImageWithFilter = ImageEnhance.Contrast(image).enhance(100).convert("L")
    SubPlots[i].imshow(ImageWithFilter, cmap="gray")
    NewImage = np.asarray(np.dstack((ImageWithFilter, ImageWithFilter,
ImageWithFilter)), dtype=np.uint8)
    p = np.argmax(Model.predict(np.array([np.array(NewImage)])))
    if p < 10:
        PredictionCarNumber += str(p)
    else:
        PredictionCarNumber += AllLetters[p-10]
    i += 1
plt.show()
print("Car number:")
print(*PredictionCarNumber)
```