МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра технологий программирования

Черепенников Роман Михайлович

ИССЛЕДОВАНИЕ АЛГОРИТМОВ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Курсовой проект
студента 3 курса 8 группы

«Допустить к защите»
Руководитель работы
Карпович Наталья Александровна
Старший преподаватель
Кафедра технологий программирования
_____
«_____» _____ 2021 г

Минск 2021

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Способность видеть, то есть воспринимать информацию об окружающем мире с помощью органов зрения – одно из важнейших свойств человека. Посмотрев на картинку, мы, почти не задумываясь, можем сказать, что на ней изображено. Человек различает отдельные предметы: дом, дерево или гору. Мы понимаем какой из предметов находится ближе к нам, а какой – дальше. Мы осознаем, что крыша дома – красная, а листья на дереве – зеленые. Наконец, мы можем с уверенностью заявить, что наша картина – пейзаж, а не натюрморт или портрет. Все эти выводы мы делаем за считанные секунды.

Компьютер справляется со многими задачами гораздо лучше, чем человек. Например, он гораздо быстрее считает. Однако такое, казалось бы, несложное задание, как найти на картинке дом или гору, может поставить машину в тупик. Почему так происходит?

Человек учится распознавать – то есть находить и отличать друг от друга объекты на протяжении всей своей жизни. Он видел дома, деревья и горы бессчетное количество раз, как в действительности, так и на картинах, фотографиях и в кино. Он помнит, как выглядят те или иные предметы в разных ракурсах и при разном освещении.

Компьютеры же создавались для работы с числами. Необходимость наделить их зрением возникла относительно недавно. Распознавание номерных знаков автомобилей, чтение штрихкодов на товарах в супермаркете, анализ записей с камер видеонаблюдения, поиск лиц на фото, создание беспилотных автомобилей, умеющих находить и избегать препятствия – всё это задачи, которые требуют от компьютера способности «видеть» и интерпретировать увиденное.

Область компьютерных наук, занимающаяся формированием выводов на основе анализа изображений и видео, полученных с помощь датчиков, называется компьютерным зрением. Существует множество алгоритмов, с помощью которых можно частично решить задачи компьютерного зрения. В последнее время наиболее популярны и эффективны алгоритмы, построенные с использованием искусственных нейронных сетей.

В данной работе будут рассмотрены вопросы представления изображений в памяти компьютера и алгоритмы компьютерного зрения, основанные на нейронных сетях. В практической части работы будет решена задача классификации изображений дорожных знаков.

ГЛАВА 1. ПРЕДСТАВЛЕНИЕ 2D ИЗОБРАЖЕНИЙ В ПАМЯТИ КОМПЬЮТЕРА

Перед непосредственной обработкой изображений с целью классификации объектов на нем или для решения других задач компьютерного зрения необходимо некоторым образом сохранить их в памяти компьютера, при этом не потеряв информацию, например, цвета объектов, которую несет в себе изображение. В этой главе будут рассмотрены способы представления изображений в памяти, а именно, растровая, векторная и фрактальная графика. А также способы представления цвета (цветовые модели и пространства).

1.1 Векторная графика

Векторная графика – способ представления изображений, основанный на математическом описании элементарных геометрических объектов, которые обычно называют примитивами, таких как: точки, линии, сплайны, кривые Безье, круги и окружности, многоугольники.

В памяти данный тип графики будет представлен как некоторое подобие базы данных содержащая информацию о примитивах. Так, например, для описания прямой достаточно знать две точки, через которые она проходит, для круга и окружности – центр и радиус. То есть запись будет выглядеть примерно так:

Прямая(x1,y1,x2,y2)
Окружность(x, y, r)

Благодаря такому подходу размеры графических файлов, в которых хранятся векторные изображения в несколько раз меньше, чем аналогичные растровые изображения, что безусловно является достоинством векторной графики. Так же преимуществом является легкая масштабируемость изображений, так как масштабирование будет требовать пересчета примитивов по формулам, то качество изображения не пострадает.
К недостаткам можно отнести то, что с помощью векторной графики не позволяет получить изображения фотографического качества. Дело в том, что фотография – «мозаика» с очень сложным распределением цветов и яркостей пикселей, что делает представление такого изображения с помощью примитивов очень сложной задачей.

Векторная графика используется для иллюстраций, иконок, логотипов и технических чертежей, но сложна для воспроизведения фотореалистичных изображений.

1.2 Фрактальная графика

Фрактал (лат. fractus – дробленый) – сложная геометрическая фигура, обладающая свойством самоподобия, то есть составленная из нескольких частей, каждая из которых подобна всей фигуре в целом. Фрактальные изображения формируются на основе использования рекурсивных математических формул. Отсюда и вытекает преимущество фрактальных изображений, для хранения их в памяти необходимо даже меньше памяти, чем для хранения векторных изображений, причем требуемое количество памяти не будет зависеть от степени детализации. Не любой объект обладает свойством самоподобия, что ограничивает применение фрактальных изображений. Так же к недостаткам можно отнести то, что вычисление формул для построения изображений может требовать значительных вычислительных ресурсов. В основном фрактальная графика используется для изображения различных природных объектов, например, облаков, снежинок, поверхностей воды.

1.3 Растровая графика

Растровый метод – изображение представляется в виде матрицы окрашенных точек, называемых пикселями. Это наиболее простой способ представления, ведь таким образом видит наш глаз.

Пиксель (от англ. **pic**ture **el**ement) – атомарный элемент растрового изображения. Каждый пиксель характеризуется тремя величинами (x, y, u), где $(x, y) \in \mathbb{Z}^2$ характеризует местоположение, а величина u является значением пикселя, которая некоторым способом характеризует цвет, более подробно это будет рассмотрено далее.

Основными характеристиками растрового изображения являются:

1. Размер изображения в пикселях, оно может выражаться в виде количества пикселей по ширине и по высоте (800 x 600 px), либо же в виде общего количества пикселей.
2. Количество используемых цветов и глубина цвета
   Под глубиной цвета понимается количество бит, используемое для хранения и представления цвета при кодировании, либо одного пикселя растровой графики (выражается единицей бит на пиксель bpp), либо для каждого цвета составляющего один пиксель (определяется как бит на компонент, бит на канал, бит на цвет).
3. Цветовое пространство
4. Разрешение изображения
   Разрешение – это степень детализации изображения, число пикселей, отводимых на единицу площади или единицу длины. Поэтому имеет

смысл говорить о разрешении только применительно к какому-либо устройству ввода или вывода изображения.

Достоинством такого способа является возможность получения фотореалистичного изображения высокого качества в различном цветовом диапазоне. Еще одним преимуществом является простота работа с устройствами вывода, такими как принтеры и мониторы. Высокая точность и широкий цветовой диапазон требуют увеличения объема файла для хранения изображения и оперативной памяти для его обработки, что можно отнести к его недостаткам. Вторым существенным недостатком является потеря качества изображения при его масштабировании.

Основные сферы применения растровой графики: обработка фотографий, реставрационные работы, создание графических объектов с большой цветовой гаммой.

1.4 Представление цвета в компьютере

Воспринимаемый человеком не имеет объективного определения. Разные люди воспринимают цвета по-разному, причем освещение так же может играть важную роль. Примером разного восприятия цвета у людей и влияния освещения может служить известная фотография платья, вокруг цвета которого в интернете были бурные споры.

Человек может различить лишь несколько десятков оттенков серого на экране, но при этом различает от нескольких сотен до нескольких десятков тысяч цветов.

В этом разделе будут рассмотрены вопросы, связанные с понятием цвета, и цветовые модели, используемые при работе с компьютерной графикой.

1.4.1 Цвет и свет

Понятие цвета и света тесно связаны между собой, и являются основополагающими в компьютерной графике. Свет может рассматриваться либо как электромагнитная волна, либо как поток фотонов, такое свойство называется корпускулярно-волновым дуализмом. Мы будем рассматривать его с точки зрения электромагнитной волны. Одной из характеристик электромагнитной волны является ее длина λ. Видимым светом называют электромагнитные волны, воспринимаемые человеческим глазом. В качестве коротковолновой (нижней) границы видимого

света принимают участок с длиной волны 380-400 нм, а в качестве длинноволновой (верхней) границы – 760-780 нм. В жизни мы редко встречаемся со светом какой-то определенной длины волны. Источник света называется ахроматическим, если наблюдаемый свет содержит все длины волн в приблизительно равных количествах. Белыми выглядят объекты, ахроматически отражающие более 80% света, а черными – менее 3%. Промежуточные значения соответствуют различным источникам серого. Если воспринимаемый цвет содержит длины волн в неравных произвольных количествах, то он хроматический. Под монохроматическим светом понимают свет, имеющий одну длину волны.

«Средний человек» воспринимает цвета видимого спектра следующим образом:

- *красный* (приблизительно от 625 до 780 нм) и *оранжевый* (приблизительно от 590 до 625 нм) соответствуют длинноволновой части видимого спектра (за которым следует инфракрасная область)
- *желтый* (приблизительно от 565 до 590 нм), *зеленый* (приблизительно от 500 до 565 нм) и *голубой* (приблизительно от 485 до 500 нм) занимают среднюю часть видимого спектра
- *синий* (приблизительно от 440 до 485 нм) соответствует короткой длине волны
- *фиолетовый* (приблизительно от 380 до 440 нм) соответствует самой коротковолновой части видимого спектра, за которой следует ультрафиолетовая область [(1), 49]

При описании цвета люди используют следующие субъективные характеристики:

- Цветовой тон ассоциируется в человеческом сознании с обусловленностью окраски предмета определенным типом пигмента, краски, красителя. Тон определяется характером распределения излучения в спектре видимого света. Именно тон определяет название цвета.
- Светлота (яркость) это различимость участков, сильнее или слабее отражающих свет. Уровень светлоты окрашенных объектов определяется при сравнении их с ахроматическими объектами и при выявлении степени их приближения к белому цвету, отражающему максимум света.
- Насыщенность характеризует степень, уровень, силу выражения цветового тона. [(2), 26]

1.4.2 Цветовые модели и пространства

Под *цветовой моделью* понимают математическую модель описания представления цветов в виде кортежей чисел, называемых цветовыми компонентами или цветовыми координатами. Все возможные значения цветов, задаваемые цветовой моделью, определяют *цветовое пространство*. Таким образом, цветовая модель задает соответствие между воспринимаемым человеком цветами, хранимыми в памяти, и цветами, которые формируются на устройствах вывода.
Цветовые модели можно разделить на три типа:

- Аппаратно-зависимые (модели данной группы описывают цвет применительно к конкретному цветовоспроизводящему устройству), к ним относятся, например, модели RGB, CMYK.
- Аппаратно-независимые (они дают однозначную информацию о цвете), к ним относятся модели XYZ и Lab.
- В основе психологических моделей лежат особенности психологического восприятия человека, примером могут служить модели HSB, HSV, HSL.

Возможна еще одна классификация цветовых моделей, основанную на принципе действия:

- Аддитивные (RGB), основанные на сложении цветов
- Субтрактивные (CMYK), основу которых составляет операция вычитания цветов, называемая субтрактивным синтезом

1.4.2.1 Цветовая модель Grayscale

Самое простое и понятное цветовое пространство – Grayscale. Оно используется для отображения черно-белого изображения. Цвет в данной модели описывается всего одним параметром. Параметр принимает значение от 0 до 255, где 0 соответствует черному цвету, 255 – белому, а промежуточные значения соответствуют различным оттенкам серого.

Рисунок 1.1 Цветовая модель Grayscale

1.4.2.2 Цветовая модель RGB

В основе одной из наиболее распространенных цветовых моделей, называемой RGB моделью, лежит воспроизведение любого цвета путем сложения трех основных цветов: красного (Red), зеленого (Green) и синего (Blue). В компьютерах для представления каждой из координат представляются в виде одного октета, значения которого обозначаются для удобства целыми числами от 0 до 255 включительно, где 0 — минимальная, а 255 — максимальная интенсивность. Для большинства приложений значения координат R, G и B можно считать принадлежащими отрезку [0, 1], что позволяет представлять пространство RGB в виде куба $1 \times 1 \times 1$, представленного на рисунке ниже.

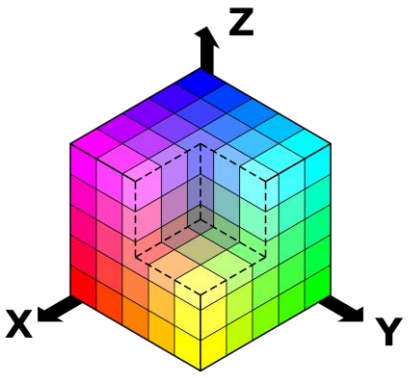

Рисунок 1.2 Цветовой куб RGB

Система RGB имеет неполный цветовой охват - некоторые насыщенные цвета не могут быть представлены смесью указанных трех компонент. В первую очередь, это цвета от зеленого до синего, включая все оттенки голубого. Речь здесь идет о насыщенных цветах, поскольку, например, ненасыщенные голубые цвета смешиванием компонентов RGB получить можно. несмотря на неполный охват, система RGB широко используется в настоящее время - в первую очередь, в цветных телевизорах и дисплеях компьютеров. отсутствие некоторых оттенков цвета не слишком заметно.

1.4.2.3 Цветовые модели CMY и CMYK

Модель CMY использует также три основных цвета: Cyan (голубой), Magenta (пурпурный, или малиновый) и Yellow (желтый). Эти цвета описывают отраженный от белой бумаги свет трех основных цветов RGB модели. Модель CMY является субтрактивной (основанной на вычитании) цветовой моделью. В CMY-модели описываются цвета на белом носителе, т. е. краситель, нанесенный на белую бумагу, вычитает часть спектра из падающего белого света. Например,

на поверхность бумаги нанесли голубой (Cyan) краситель. Теперь красный свет, падающий на бумагу, полностью поглощается. Таким образом, голубой носитель вычитает красный свет из падающего белого. Такая модель наиболее точно описывает цвета при выводе изображения на печать, т. е. в полиграфии. Поскольку для воспроизведения черного цвета требуется нанесение трех красителей, а расходные материалы дороги, использование CMY модели является не эффективным. Дополнительный фактор, не добавляющий привлекательности CMY-модели, – это появление нежелательных визуальных эффектов, возникающих за счет того, что при выводе точки три базовые цвета могут ложиться с небольшими отклонениями. Поэтому к базовым трем цветам CMY-модели добавляют черный (blacK) и получают новую цветовую модель CMYK. Для перехода из модели CMY в модель CMYK используют следующее соотношение:

$$K = \min(C, M, Y);$$
$$C = C - K$$
$$M = M - K$$
$$Y = Y - K$$

Заметим, что цветовой охват модели CMY и CMYK значительно уже чем модели RGB, разница охватов приведена на рисунке.

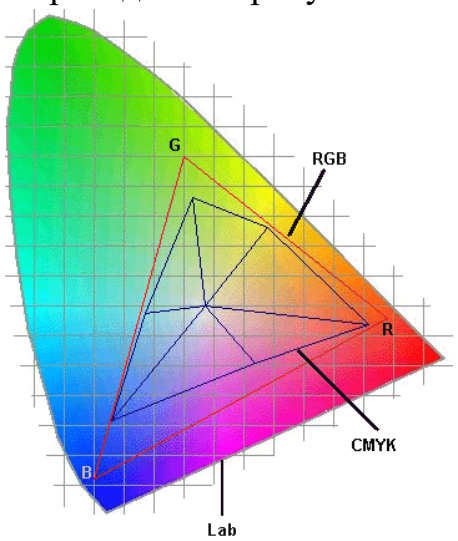

Рисунок 1.3 Сравнение охвата моделей RGB и CMYK

1.4.2.4 Цветовая модель HSV

Модели рассмотренные выше ориентированы на работу с аппаратурой и могут быть неудобны для некоторых людей. В интерфейсах выбора цвета часто используется модель цвета HSV (Hue – оттенок, Saturation –насыщенность, Value

– значение), иногда эту модель еще называют HSB (Hue, Saturation Brightness), она основана на интуитивных концепциях, а не на наборе основных цветов.

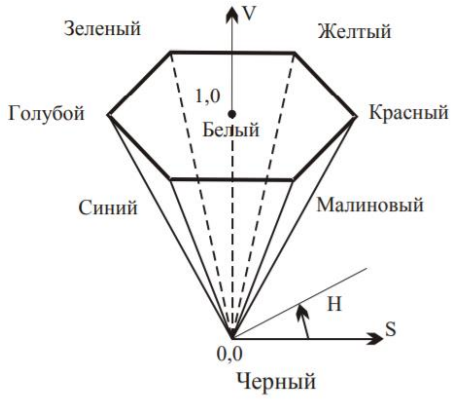

Рисунок 1.4 Цветовое пространство модели HSV

В цветовом пространстве модели HSV используется цилиндрическая система координат, а множество допустимых цветов представляет собой шестигранный конус, поставленный на вершину. Основание конуса представляет яркие цвета и соответствует $V = 1$. Тон H измеряется углом, отсчитываемым вокруг вертикальной оси OV. При этом красному цвету соответствует угол 0°, зелёному – угол 120° и т. д. Цвета, взаимно дополняющие друг друга до белого, находятся напротив один другого, т. е. их тона отличаются на 180°. Величина S изменяется от 0 на оси OV до 1 на гранях конуса. Конус имеет единичную высоту (V = 1) и основание, расположенное в начале координат. В основании конуса величины H и S смысла не имеют.

ГЛАВА 2. ОБЗОР И АНАЛИЗ АЛГОРИТМОВ КОМПЬЮТЕРНОГО ЗРЕНИЯ

Компьютерное зрение – теория и технология создания устройств или программ, которые могут производить обнаружение, отслеживание и классификацию объектов по получаемому изображению с фото- и видеокамер.

Целью компьютерного зрения является формирование выводов на основе анализа изображений и видео, полученных с помощь датчиков.

В последние годы основные успехи в задачах компьютерного зрения были достигнуты, благодаря алгоритмам, основанным на использовании нейронных сетей. В данной главе будут рассмотрены нейронные сети прямого распространения, в качестве общего примера работы нейронных сетей, и свёрточные нейронные сети, используемые для задач компьютерного зрения.

2.1 Нейронные сети прямого распространения

Нейронные сети прямого распространения, которые также называют *многослойными перцептронами*– самые типичные примеры моделей глубокого обучения. Цель сети прямого распространения – аппроксимировать некоторую неизвестную функцию f^* . Например, в случае задачи классификации, $y = f^*(x)$ отображает входные данные x в категорию y. Сеть прямого распространения определяет отображение $y = f(x, w)$ и путем обучения находит значения параметров w, дающих наилучшую аппроксимацию f^*.

Слова прямое распространение означают, что распространение информации начинается с x, проходит через промежуточные вычисления, необходимые для определения f , и заканчивается выходом y. Не существует обратных связей, по которым выходы модели подаются ей на вход. Обобщенные нейронные сети, включающие такие обратные связи, называются *рекуррентными* и в данной работе не рассматриваются.

В ходе обучения нейронной сети мы стремимся приблизить $f(x)$ к $f^*(x)$. Обучающие данные – это зашумленные приближенные примеры $f^*(x)$. Каждый пример x сопровождается меткой $y \approx f^*(x)$. Обучающие примеры напрямую указывают, что в выходном слое должно соответствовать, тому что поступило на входной слой. Поведение остальных слоев напрямую обучающими данными не определяется. Алгоритм обучения должен решить, как использовать эти слои для порождения желаемого выхода, но обучающие данные ничего не говорят о том, что должен делать каждый слой. [(3), 150]

Атомарным элементом вычислений в нейронных сетях является *нейрон*. Структурно он состоит из следующих частей:

- Несколько входов, принимающих входящие сигналы x_i
- Вычислительный центр, агрегирующий входные сигналы с использованием w_i весов, которые являются изменяющимися параметрами, именно они и настраиваются на этапе обучения
- Несколько выходов, распространяющих полученный сигнал y

В качестве функции-агрегатора обычно берут линейную:

$$y = w_0 + \sum_{i=1}^{n} w_i x_i$$

Источником входных сигналов могут быть как изучаемые данные, так и выходы других нейронов. Заметим, что у одного нейрона имеется $n + 1$ весов, где n – число входных сигналов. Далее нейроны объединяются в слои, один слой состоит из нескольких несоединенных между собой нейронов, однако, нейроны одного слоя разделяют входные сигналы и приемников их выходных сигналов (рисунок 2.1).

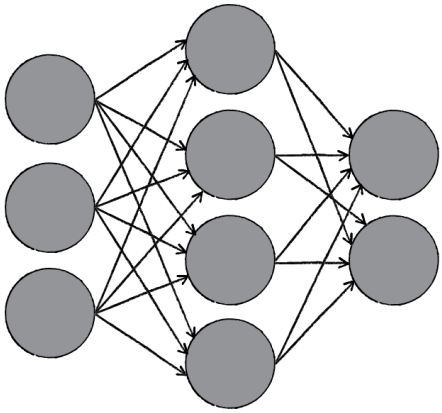

Рисунок 2.1 Схема простейшей нейросети

Отдельно в нейронных сетях выделяют *входной* (первый) слой и выходной (последний) слой нейросети. Входной слой используется исключительно как распространитель сигналов к следующему слою и не производит никаких вычислительных операций. А размерность выходного слоя определяет размерность выходной функции. Стоит отметить, что таким образом в слое с m нейронами и входными данными размерности n всего $m(n + 1)$ весов.

Соединяя несколько слоев, описанных выше, с одинаковым или различным числом нейронов получаем нейронную сеть. Она, как результат композиции функций, тоже функция принимающая на вход вектор, размера входного слоя, а результатом этой функции так же является вектор, только уже размерности выходного слоя.

У такой конфигурации нейронной сети есть существенный недостаток. Так как результирующая функция есть композиция линейных функций, то и она сама является линейной функцией. И получается, что объединение нескольких таких

слоев не дает никаких результатов, так как один слой мог бы выполнять точно такое же преобразование, как и построенная нами нейронная сеть.

Решением данной проблемы являются функции активации. Суть работы функций активации заключается в следующем: добавить нелинейных преобразований, за счет чего мы сможем аппроксимировать в том числе нелинейные функции. Так же функция активации позволяет ограничить диапазон выходных значений нейрона. После добавления функции активации выход нейрона будет выглядеть следующим образом:

$$y = \phi(w_0 + \sum_{i=1}^{n} w_i x_i)$$

Где $\phi(x)$ – некоторая функция активации. Вообще говоря, в качестве функции активации может выступать любая нелинейная функция. Но на практике, в большинстве случаев, используют один из следующих вариантов:

- ReLU (англ. Rectified linear unit)

$$\phi(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

- Сигмоидная функция

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

- Гиперболический тангенс

$$\phi(x) = th(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- Арктангенс

$$\phi(x) = arctg(x)$$

На рисунке 2.2 приведены графики этих функций в окрестности нуля. Отметим, что эти функции не имеют никаких настраиваемых параметров, а также всюду дифференцируемы, за исключением ReLU (ее производную обычно просто доопределяют в точке 0).

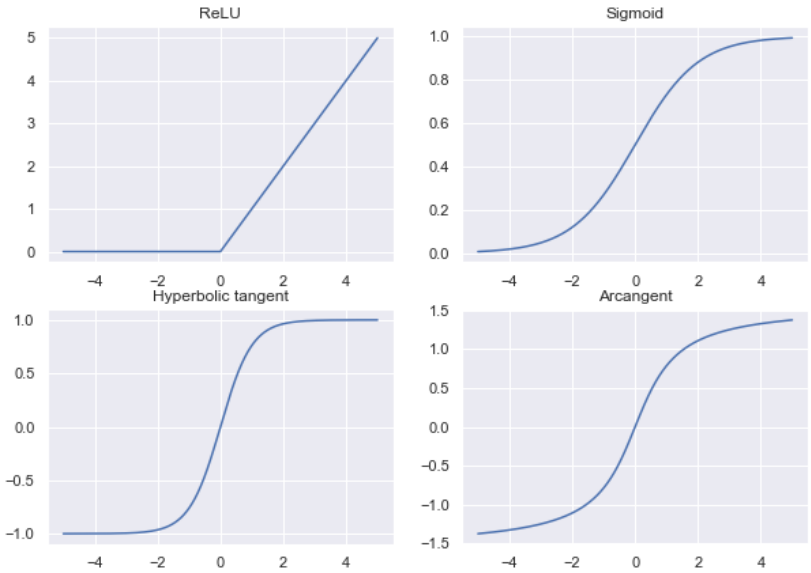

Рисунок 2.2 Наиболее используемые функции активации

2.2 Свёрточные нейронные сети и их применение в задачах компьютерного зрения

Нейронные сети прямого распространения являются *полносвязными*. Такие сети имеют один заметный недостаток: при большом размере входных данных либо получится большое количество параметров, либо сеть будет недостаточно хорошо приближать целевую функцию, если в ней существуют сложные зависимости. Такая проблема автоматически исключила этот тип сетей из использования в анализе изображений, так как там входные данные всегда имеют солидный размер. Для решения этой проблемы существует другой тип нейронных сетей, называемых *сверточными*.

Сверточные нейронные сети (англ. Convolutional Neural Networks, CNN) были предложены Яном Лекуном в 1988 году. Название архитектура получила из-за наличия сверточных слоев, которые работают по следующему принципу:

- Входные данные представляются в виде матрицы
- Методом скользящего окна матрица преобразуется в новую матрицу меньшего размера, используя так называемые *ядра свертки*
- Уменьшенная выходная матрица проходит через функцию активации и подается дальше.

Кроме сверточных слоев в таких нейронных сетях используются следующие типы слоев:

- *Слой пуллинга*

Этот слой принимает на вход матрицу или вектор и по некоторому принципу удаляет определенную долю значений. Слой пуллинга, как и сверточный слой, направлен на уменьшение размерности входных данных, что позволяет сократить дальнейшее число вычислений. Помимо этого, данный слой позволяет сети быть более устойчивой по отношению к небольшим сдвигам объектов на изображении. Сама операции пуллинга может быть совершенно разной, можно выбирать максимальное значение (max pooling), минимальное значение (min pooling) или, например, вычислять среднее значение (average pooling). [(4)]

- *Слой пакетной нормализации (batch-normalization)*
  Суть данного слоя заключается в том, чтобы передать следующему слою данные имеющие нулевую математическое ожидание и единичную дисперсию. Данный слой позволяет повысить производительность и стабилизировать работу искусственных нейронных сетей за счет борьбы с внутренним ковариантным сдвигом (явлением, в результате которого происходит изменение среднего значения и дисперсии входных данных во внутренних слоях во время обучения) [(5)]
- Dropout слой
  Основным назначением данного слоя, является борьба с *переобучением* нейронной сети. Главная идея заключается в обучении нескольких нейронных сетей вместо одной, а затем усреднить результаты. Причем сеть для обучения получается исключением из исходной сети некоторых нейронов в сети с вероятностью p. Исключенные нейроны не вносят свой вклад в процесс обучения, поэтому исключение хотя бы одного нейрона равносильно обучению новой нейронной сети. По словам авторов, [(6)] “В стандартной нейронной сети производная, полученная каждым параметром, сообщает ему, как он должен измениться, чтобы, учитывая деятельность остальных блоков, минимизировать функцию конечных потерь. Поэтому блоки могут меняться, исправляя при этом ошибки других блоков. Это может привести к чрезмерной совместной адаптации (co-adaptation), что, в свою очередь, приводит к переобучению, поскольку эти совместные адаптации невозможно обобщить на данные, не участвовавшие в обучении. Мы выдвигаем гипотезу, что Dropout предотвращает совместную адаптацию для каждого скрытого блока, делая присутствие других скрытых блоков ненадежным. Поэтому скрытый блок не может полагаться на другие блоки в исправлении собственных ошибок.”

CNN представляют составляются из последовательного соединения слоев пуллинга, пакетной нормализации, dropout и сверточных слоев с разным количеством ядер свертки. Но на практике самой распространённой архитектурой нейронной сети является соединение сверточной и полносвязной сетей. В результате, полученная сеть работает следующим образом.

Сверточная часть выступает в роли генератора признаков, выделяя из входных данных признаки меньшего размера, чем сами входные данные

Полносвязная часть выступает в роли классификатора по выделенным признакам.

Перейдем к рассмотрению популярных архитектур сверточных нейронных сетей, используемых в задачах компьютерного зрения в прошлом и современности.

2.2.1 LeNet-5

Была разработана в 1994 году и является одной из первых сверточных нейронных сетей. Архитектура данной нейросети представлена на рисунке 2.3

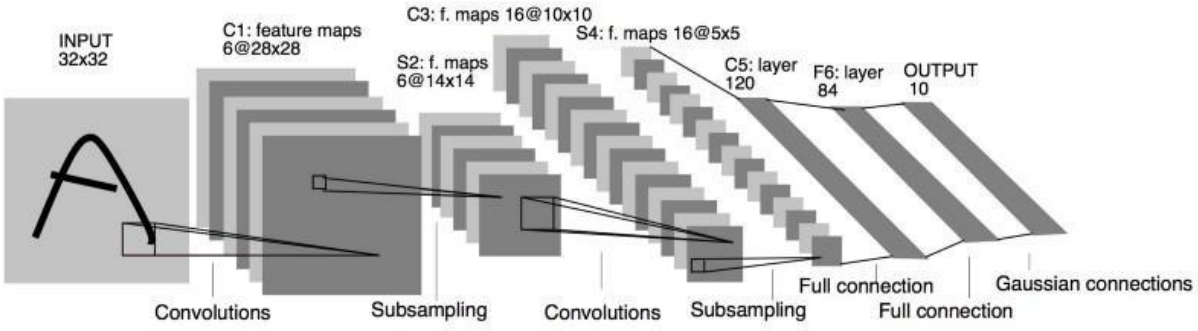

Рисунок 2.3 Архитектура сети LeNet-5

Архитектура LeNet-5 стала фундаментальной для глубокого обучения, особенно с точки зрения распределения свойств изображения по всей картинке. Свёртки с обучаемыми параметрами позволяли с помощью нескольких параметров эффективно извлекать одинаковые свойства из разных мест. В те годы ещё не было видеокарт, способных ускорить процесс обучения, и даже центральные процессоры были медленными. Поэтому ключевым преимуществом архитектуры оказалась возможность сохранять параметры и результаты вычислений, в отличие от использования каждого пикселя в качестве отдельных входных данных для большой многослойной нейросети. В LeNet5 в первом слое пиксели не используются, потому что изображения сильно коррелированы пространственно, так что использование отдельных пикселей в качестве входных свойств не позволит воспользоваться преимуществами этих корреляций.

К особенностям данной архитектуры можно отнести следующее:

- Использует свёртку для извлечения пространственных свойств.
- Пуллинг с использованием усреднения
- Нелинейность в виде гиперболического тангенса или сигмоида
- Финальный классификатор в виде полносвязной нейросети

Сейчас эта архитектура имеет только историческую значимость, так как впоследствии легла в основу многих архитектур.

2.2.2 AlexNet

В 2012-м Алексей Крижевский опубликовал AlexNet, углублённую и расширенную версию LeNet, которая с большим отрывом победила в сложном соревновании ImageNet.

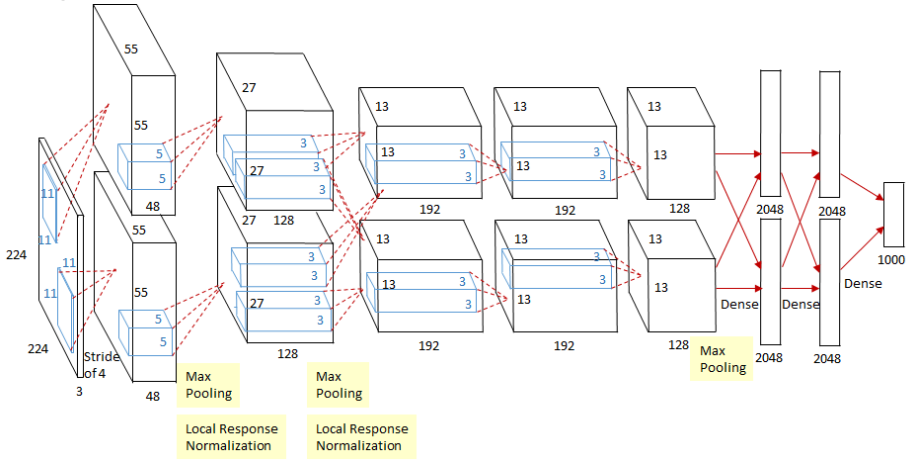

Рисунок 2.4 Архитектура сети AlexNet

В AlexNet результаты вычислений LeNet масштабированы в гораздо более крупную нейросеть, которая способна изучить намного более сложные объекты и их иерархии. Особенности этого решения:

- Впервые в качестве функции активации была применена ReLU
- Использование методики отбрасывания (Dropout) для выборочного игнорирования отдельных нейронов в ходе обучения, что позволяет избежать переобучения модели.
- Перекрытие max pooling, что позволяет избежать эффектов усреднения average pooling.

2.2.3 VGG

Следующим шагом в развитии сверточных нейронных сетей стала архитектура VGG, а точнее два варианта VGG16 и VGG19.

Основным отличием от предыдущих архитектур стал отказ от сверток с ядрами размеров 5×5 и 11×11 в пользу сверток с размером ядра 3×3. Интуиция в использовании больших сверток простая – мы хотим получать больше информации от соседних пикселей, но гораздо лучше использовать маленькие фильтры чаще. И вот почему:

- Каждый последующий слой свертки обладает информацией о предыдущем. И чем сильнее мы углубляемся, тем больше информации последний сверточный слой содержит о первом
- Мы слабее уменьшаем размерность нашего изображения, следовательно, можем применять больше сверток
- Больше сверток – больше активаций, а, следовательно, и больше нелинейности
- Также использование сверток меньших размеров повышает производительность

Рисунок 2.5 Архитектуры VGG16 и VGG19

2.2.5 ResNet

В декабре 2015-го произошла еще одна революция – опубликовали ResNet.

Основным новшеством данной архитектуры стали так называемые *residual* блоки, предназначенные для борьбы с проблемой затухающего градиента.

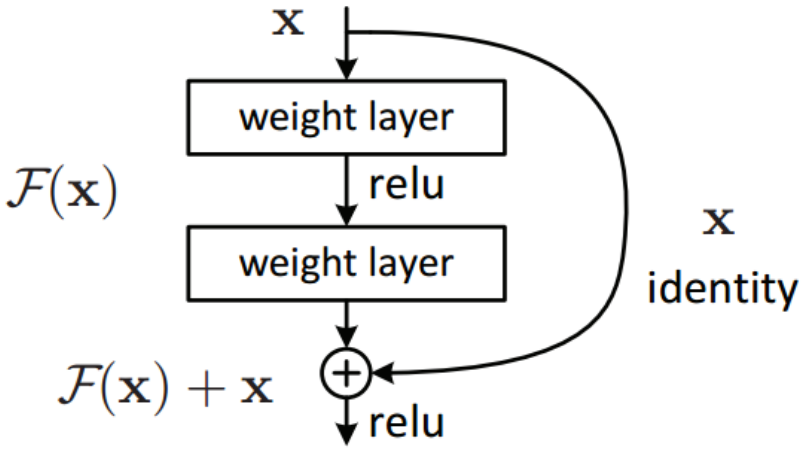

Рисунок 2.6 Residual блок

Residual блок представляет собой несколько сверточных слоев с активациями, которые преобразуют входной сигнал x в сигнал $F(x)$. А *shortcut* соединение это тождественное преобразование $x \to x$. Так же в результате такой конструкции residual блок учит, как входной сигнал x отличается от $F(x)$. Поэтому если на некотором слое сеть уже достаточно хорошо аппроксимировала исходную функцию, порождающую данные, то на дальнейших слоях оптимизатор может в этих блоках делать веса близкими к нулю, и сигнал почти без изменений проходить по shortcut соединению. В некотором смысле, это означает что такая сеть сама может определять свою глубину.

Заметим также, что ResNet стала первой архитектурой которая использовала сотни и даже тысячи слоев.

ГЛАВА 3. РЕШЕНИЕ ЗАДАЧИ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ ДОРОЖНЫХ ЗНАКОВ

3.1 Анализ программных систем для обучения глубоких нейронных сетей

В последнее время появилось большое количество, программных систем глубокого обучения. К их числу относятся программные библиотеки, расширения языков программирования, и даже самостоятельные языки. Все эти системы позволяют использовать готовые алгоритмы создания и обучения нейронных сетей.

Правильный выбор инструмента – важная задача, позволяющая достичь необходимого результата с наименьшими затратами сил и за наименьшее время.

3.1.1 TensorFlow

TensorFlow — это комплексная платформа для машинного обучения с открытым исходным кодом. Она была разработана командой Google Brain как продолжение закрытой системы машинного обучения DistBelief, однако в ноябре 2015 года компания передумала и открыла фреймворк для свободного доступа.

Как и большинство фреймворков глубокого обучения, TensorFlow имеет API на Python поверх механизма C и C ++, что ускоряет его работу.

TensorFlow имеет гибкую экосистему инструментов, библиотек и ресурсов сообщества. Это позволяет исследователям использовать самые современные МО-технологии, а разработчикам — создавать и развёртывать приложения на базе машинного обучения.

Стоит отметить, что фреймворк постоянно развивается за счёт открытого исходного кода и огромного сообщества энтузиастов. Также за счёт его популярности есть множество уже решённых задач, что существенно упрощает жизнь новоиспечённым разработчикам.

Однако фреймворк не лишён недостатков. Компания Google известна своей любовью к созданию собственных стандартов, что коснулось и фреймворка.

Плюсы:

- Отличный фреймворк для создания нейронных сетей, которые будут работать в продакшене.
- Берёт на себя оптимизацию ресурсов для вычислений.
- Огромное сообщество.

- За счёт популярности выше вероятность, что проблему, подобную вашей, уже решили.

Минусы:

- Сложен в использовании и освоении.
- Недружелюбный.
- Необходимо постоянно контролировать используемую видеопамять.
- Имеет свои стандарты.
- Плохая документация.

3.1.2 Keras

Keras — открытая среда глубокого обучения, написанная на Python. Она была разработана инженером из Google Франсуа Шолле и представлена в марте 2015 года.

Фреймворк нацелен на оперативную работу с нейросетями и является компактным, модульным и расширяемым. Подходит для небольших проектов, так как создать что-то масштабное на нём сложно, и он явно будет проигрывать в производительности нейросетей тому же TensorFlow.

Keras работает поверх TensorFlow, CNTK и Theano и предоставляет интуитивно понятный API.

Фреймворк содержит многочисленные реализации широко применяемых строительных блоков нейронных сетей, таких как слои, целевые и передаточные функции, оптимизаторы, а также множество инструментов для упрощения работы с изображениями и текстом.

Плюсы:

- Удобен в использовании.
- Лёгок в освоении.
- Быстроразвивающийся фреймворк.
- Хорошая документация.

Минусы:

- Не подходит для больших проектов.

3.1.3 PyTorch

PyTorch — это среда машинного обучения на языке Python с открытым исходным кодом, обеспечивающая тензорные вычисления с GPU-ускорением.

Она была разработана компанией Facebook и представлена в октябре 2016 года, а открыта для сторонних разработчиков — в январе 2017 года. Фреймворк подходит для быстрого прототипирования в исследованиях, а также для любителей и небольших проектов.

Фреймворк предлагает динамические графы вычислений, которые позволяют обрабатывать ввод и вывод переменной длины, что полезно, например, при работе с рекуррентными нейронными сетями. Если коротко, то за счёт этого инженеры и исследователи могут менять поведение сети «налету».

В отличие от TensorFlow, PyTorch менее гибок в поддержке различных платформ. Также в нём нет родных инструментов для визуализации данных, но есть сторонний аналог, называемый tensorboardX.

Также при развёртке сетей на GPU PyTorch самостоятельно займёт только необходимую видеопамять.

Плюсы:

- Имеет множество модульных элементов, которые легко комбинировать.
- Легко писать собственные типы слоев и работать на GPU.
- Имеет широкий выбор предварительно обученных моделей.

Минусы:

- Вам придётся самостоятельно писать тренировочный код.
- Плохая документация

Для решения задачи мы выберем Keras, так как он прост в освоении и удобен. А при необходимости использовать предобученные модели можно использовать TensorFlow, поскольку он обладает большой коллекцией таковых.

3.2 Постановка задачи

В качестве задачи для экспериментального исследования была выбрана задача классификации изображения.

Классификация изображений: задача, в которой имеется множество объектов некоторым образом разделенных на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

На вход подается изображение с одним объектом. В качестве выхода должны получить метку класса (например, одно или несколько целых чисел, сопоставленных меткам классов).

3.3 Метод решения

Для решения данной задачи будем использовать сверточную нейронную сеть следующей архитектуры:

- 2 сверточных слоя с 32 ядрами размера 5×5 и функцией активации ReLU
- MaxPool слой с размером ядра 2×2
- Dropout слой с вероятностью исключения 0.25
- 2 сверточных слоя с 32 ядрами размера 3×3 и функцией активации ReLU
- MaxPool слой с размерности 2×2
- Dropout слой с вероятностью исключения 0.25
- Flatten слой, который предназначен для перевода полученных результатов в 1 измерение
- Полносвязный слой с 500 выходами и функцией активации ReLU
- Dropout слой с вероятностью исключения 0.5
- Полносвязный слой с 43 выходами и функцией активации Softmax

Написание модели и ее обучение проводилось на языке Python с использованием библиотеки Keras.

В качестве входных данных использовать набор данных GTSRB[(7)]. Набор данных содержит более 50000 изображений различных дорожных знаков и классифицируется на 43 различных класса. Он весьма разнообразен: некоторые классы содержат много изображений, а некоторые – всего несколько изображений.

3.4 Анализ результатов

Качество модели будем оценивать с помощью метрики accuracy, которая определяется как доля верных ответов.

В качестве ответа наша сеть возвращает вероятность принадлежности к каждому из классов. Чтобы получить единственную метку класса просто выберем класс, вероятность принадлежности к которому наибольшая.

Оценивать качество будем на отложенных данных, которые никак не использовались при обучении. Такой способ оценки, позволяет оценить модель достаточно объективно.

После проведения 25 эпох обучения была достигнута точность классификации в 96.24%. Дальнейшее обучение существенным образом не повлияло на результаты. Для улучшения точности необходимо изменять структуру сети, например, добавляя в нее дополнительные сверточные слои. Благодаря таким изменением модель смогла бы выучить более сложные представления, но и обучение модели потребовало бы больше вычислительных ресурсов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Р. Клетте. Компьютерное зрение. Теория и алгоритмы. – М.: ДМК Пресс,2019. – 506с.: ил.
2. А.Ю. Дёмин Основы компьютерной графики: учебное пособие / Томский политехнический университет. – Томск: Изд-во Томского политехнического университета, 2011. – 191 с.
3. Я. Гудфеллоу Глубокое обучение / пер. с англ. А. А. Слинкина – 2-е изд., испр. – М.: ДМК Пресс, 2018 – 652 с.: цв. ил.
4. [Электронный ресурс]. – Режим доступа: https://programforyou.ru/poleznoe/convolutional-network-from-scratch-part-two-pooling-layer
5. [Электронный ресурс]. – Режим доступа: https://livepcwiki.ru/wiki/Batch_normalization
6. [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1207.0580.pdf
7. [Электорнный ресурс]. – Режим доступа: https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign