Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод распознавания летательных аппаратов с аэрофотоснимков с использованием нейронный сетей»

Студент ИУ7-81Б _____ Мицевич М. Д.
(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель ВКР _____ Тассов К. Л.
(Подпись, дата) (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 45 с., 8 рис., 1 табл., 26 источн., 1 прил.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

—

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

—

ВВЕДЕНИЕ

Одной из основных задач, возникающих при обработке изображений, является классификация различных объектов на снимке. В роли объектов может быть различная техника: вертолеты, самолеты, машины, корабли.

Обработка в реальном времени полезна для задач контроля движения судов, поиска объектов на местности в случае аварии и крушения, предотвращение таких ситуаций, картографии.

Классификация объектов с аэрофотоснимков полезна во время непрерывно ведущегося наблюдения с воздуха. В этом случае можно вести наблюдение с воздуха и в автоматическом режиме выдавать информацию о происходящем на земле.

Целью данной работы является разработка метода распознавания летательный аппаратов с аэрофотоснимков.

Для достижения поставленной цели требуется выполнить следующие задачи:

- определить термины, связанные с предметной областью;
- провести анализ методов классификации летальной техники на аэрофотоснимках;
- определить критерии сравнения методов;
- провести их сравнительный анализ;
- разработать метод классификации летальной техники на аэрофотоснимках;
- спроектировать программное обеспечение для реализации разработанного метода;
- реализовать спроектированный метод;
- провести исследование точности распознавания модели на тестовой выборке при различных подходах к обучению.

1 Аналитический раздел

1.1 Детерминированный подход

Детерминированный подход основан на сравнении признаков распознаваемого объекта с признаками эталона и предполагает, что в любой точке пространства признаков могут появляться реализации только одного класса объектов. Алгоритмы данного подхода должны иметь возможности к автосмещению, автомасштабированию и автоповороту.

Одним из таких является алгоритм распознавания изображений на основе градиентного совмещения объекта с эталоном, предложенный в сборнике статей [1].

Все алгоритмы детерминированного подхода можно разбить на 4 основных шага:

1. выделение контуров объектов;
2. выбор наиболее информативных узловых точек эталона и объекта, а также установление связи между ними;
3. совмещение точек, на этом шаге должна осуществляться инвариантность к смещению, повороту и масштабированию;
4. принятие решения о принадлежности объекта к тому или иному классу.

Основным преимуществом такого подхода является то, что для классификации не требуется никаких данных, кроме эталонных изображений.

Главным недостатком является сложность получения устойчивого контура на зашумленных изображениях.

1.2 Экспертный подход

Методы экспертного подхода основаны на разделении всего множества входных объектов по некоторым заранее заданными критериям. Одним из таких методов экспертного подхода является дерево решений.

Дерево решений представляет собой иерархическую древовидную структуру в узлах, которой содержатся условия. Условия генерируются при создании дерева, то есть в процессе обработки обучающего множества данных.

Листьями дерева являются конкретные классы по которым производится классификация.

Процесс построения дерева решений предполагает рекурентное последовательное разбиение обучающего множества на подмножества с применением решающий правил в узлах дерева. Процесс разбиения продолжается до тех пор, пока во всех листах не будет содержаться конкретных класс, по которому производится классификация. Узел становится листом либо когда он содержит объект одного класса, либо когда достигнуто какое-либо из условий остановки, к примеру, достигнута максимальная глубина дерева.

При построении дерева используются жадные алгоритмы, которые выбирают оптимальные решения для конкретного шага алгоритма, а не всей системы целиком. То есть при выборе условия разбиения множества на два подмножества алгоритм будет выбирать лучшее только для этого шага и не сможет в дальнейшем вернуться и изменить его, даже если это будет оптимальнее для всей системы целиком.

На каждом этапе выбирается одно условие из всех возможных по средством максимизации функции прироста информации [2]. Для ее вычисления необходимо сначала задать функцию ошибки. К примеру, можно использовать среднюю квадратичную ошибку 1.1

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(ytrue - ypred_{\text{i}})^2, \tag{1.1}$$

где n – число элементов в подмножестве, $ytrue$ – значение атрибута в узле дерева, $ypred_{\text{i}}$ – значение атрибута в i элементе подмножества.

Тогда функция прироста информации может быть выражена зависимостью 1.2

$$IG = MSE_{\text{root}} - (\frac{n_{\text{left}}}{n}MSE_{\text{left}} + \frac{n_{\text{right}}}{n}MSE_{\text{rigth}}), \tag{1.2}$$

где MSE_{root} – значение ошибки в корневом узле, MSE_{left} – значение ошибки в левом подмножестве, MSE_{right} – в правом подмножестве.

Таким образом, на каждом шаге построения дерева из всех возможных условий выбирается то, для которого значение функции прироста информации будет наибольшим. В узлах, в которых значение функции ошибки равно 0 или остался всего один элемент, процесс построения дерева останавливается и они считаются листами.

1.3 Нейрокомпьютерный подход

Методы данного подхода основаны на создании и последующим обучении некоторой математической модели. Алгоритмы данного подхода можно разделить на две категории:

1. обучение с учителем;
2. обучение без учителя.

При обучении с учителем каждый прецедент представляет собой пару «объект, ответ». Требуется найти функциональную зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ.

В случае обучения без учителя ответы не задаются и требуется найти зависимость между объектами.

К задачам, которые решаются с помощью алгоритмов обучения с учителем можно отнести следующие:

- классификация – результатом является значение из конечного множества классов;
- регрессия – результатом является действительное число или числовой вектор.

К задачам обучения без учителя могут быть отнесены следующие:

- задача кластеризации, целью которой является группировка объектов в кластеры на основе данных об их попарном сходстве;
- задача фильтрации выбросов, целью которой является выделение в обучающей выборке нетипичных элементов.

К алгоритмам обучения с учителем относятся нейрнонные сети. Подробное описание нейронных сетей приведено в разделе главе 1.4.

1.4 Нейронные сети

Модель нейронной сети основана на биологическом нейроне. У нейрона есть ядро, которое называется телом. В теле накапливается электрический

заряд. С телом соединены отростки. Отростки, по которым сигнал поступает в тело, называются дендритами. Отросток, по которому сигнал передается другим нейронам, называется аксоном. Место, где аксон соединяется с дендритами, называется синапсом. Синапс отвечает за количество заряда, которое перейдет от аксона к дендриту. Синапс может изменяться со временем. Именно с настройкой синапса и связана тренировка биологической нейронной сети.

1.4.1 Математическая модель МакКаллока-Питтса

В математической модели МакКаллока-Питтса, тело нейрона, где накапливается заряд, заменяется на сумматор. Дендриты являются входами сумматора, а выходом – аксоном. Биологический нейрон накапливает заряд до тех пор, пока этот заряд не достигнет какого-то значения, и только после этого этот заряд уходит по аксону к другим нейронам. В математической модели к сигналу после выхода из сумматора применяется функция активации и только после этого сигнал попадает на дендрит следующего нейрона. Синапсы в математической модели заменяются на веса входов нейрона. Математическая модель нейрона выражается зависимостью 1.3

$$y = f\left(\sum_{i=1}^{n}(w_{\mathrm{i}}x_{\mathrm{i}}) + b\right), \tag{1.3}$$

где y – сигнал на выходе из нейрона, f – функция активации, w_{i} – вес i входа, x_{i} – сигнал этого входа, b – некоторое значение смещения, которое задается отдельно для каждого нейрона. Обучение нейронной сети происходит за счет настройки синаптических весов w_{i} и смещения b.

1.4.2 Функции активации

Существует много различных функций активации (фактически любая функция может быть функцией активации). Наиболее популярными считаются логистическую функцию, гиперболический тангенс, ReLU [3]. Важной особенностью функций активации является их дифференцируемость (хотя для некоторых функций это выполняется не всегда), поскольку при обратном распространении ошибки необходимо вычислять градиенты, использующие производную функции активации.

Логистическая функция преобразовывает поступающие в неё значения

в вещественный диапазон [0, 1]. Это означает, что при x>0 выходное значение будет примерно равно единице, а при x<0 будет близким к нулю. Данная функция часто используется в задачах классификации [3]. Логистическая функция определяется зависимостью 1.4.

$$y = \frac{1}{1 + e^{-x}}. \tag{1.4}$$

Гиперболический тангенс схож с логистической функцией, но в отличии от нее может принимать отрицательные значения. Гиперболический тангенс определяется зависимостью 1.5.

$$y = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{1.5}$$

Функция ReLU возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число. Функция ReLU определяется зависимостью 1.6.

$$\mathrm{ReLU}(x) = \begin{cases} x, & \text{если } x > 0 \\ 0, & \text{иначе} \end{cases} \tag{1.6}$$

ReLU решает проблему обнуления градиента (ситуация, при которой во время обучения градиенты по всем весам становятся близкими или равными нулю) для положительных чисел, также она вычисляется гораздо проще, чем сигмоидальные функции (логистическая функция, гиперболический тангенс) [3].

1.4.3 Составляющие нейронной сети

При обучении нейронной сети используются две подвыборки обучающего множества. Вся обучающая выборка состоит из какого-то количества объектов, для которых известны признаки, на которые должна обучиться нейронная сеть. Первая подвыборка называется тренировочной и используется для итеративного обучения нейронной сети. Вторая называется тестовой и используется для оценки того, насколько хорошо обучена нейронная сеть.

Нейронную сеть определяют следующие параметры:

- архитектура нейронной сети – отвечает за то, как нейроны связаны

между собой;

- функция потерь – определяет насколько точно работает модель [4];
- метод оптимизации – определяет способ уменьшения функции потерь на каждой итерации обучения.

Нейроны делятся на три типа: входной, скрытый и выходной. В том случае, когда нейросеть состоит из большого количества нейронов, вводят термин слоя. Соответственно, есть входной слой, который получает информацию, некоторое количество скрытых, которые ее обрабатывают и выходной слой, который выводит результат [5]. Количество скрытых слоев и число нейронов в каждом из них задают архитектуру нейронной сети.

1.4.4 Методы оптимизации

Самый используемый метод оптимизации – градиентный спуск [6]. Градиентный спуск основан на пошаговом приближении функции к локальному минимуму. На каждой итерации алгоритма новые значения получаются по формуле 1.7

$$w_1 = w_0 - \alpha \Delta f(w_0), \tag{1.7}$$

где w_1 – вектор новых значений, которые подбираются алгоритмом, w_0 – значения параметров на текущем шаге, $\Delta f(w_0)$ – вектор градиентов функции потерь по каждому из параметров на текущем шаге, α – скорость обучения.

На каждой итерации градиентного спуска требуется считать градиент функции потерь, которая зависит от функций активации каждого из нейронов сети. В связи с этим к фукнциям потерь и активации применяются требования по дифференцируемости.

В связи с тем, что градиентный спуск находит только локальный минимум, не всегда полученный результат будет оптимальным. Результат работы алгоритма зависит от изначальных настроек параметров нейронной сети.

Выделяют три основных типа градиентного спуска [6]:

- мини-пакетный градиентный спуск – в этом случае обучающий набор данных разбивается на небольшие партии, которые используются для расчета ошибки модели и обновления коэффициентов модели;

- стохастический градиентный спуск – в этом случае градиент оптимизируемой функции считается на каждом шаге не как сумма градиентов от каждого элемента выборки, а как градиент от одного, случайно выбранного элемента;

- пакетный градиентный спуск – это разновидность алгоритма градиентного спуска, который вычисляет ошибку для каждого примера в наборе обучающих данных, но обновляет модель только после того, как все обучающие примеры были оценены.

1.4.5 Функции потерь

Согласно исследованиям [7] для задачи классификации изображений самой эффективной функцией потерь являются категориальная перекрестная энтропия, которая определяется выражением 1.8

$$CM_{\text{i}} = -\sum_{i=1}^{N} t_{\text{i}} \log p_{\text{i}}, \tag{1.8}$$

где N – число классов классификации, t_{i} – 0 или 1 в зависимости от того принадлежит ли изображение на входе нейронной сети классу, за который отвечает i нейрон выходного слоя, p_{i} – результат на выходе из нейрона.

В задачах классификации используют категориальную перекрестную энтропию в качестве функции потерь. В таких случаях на выходном слое нейронной сети создается столько нейронов, сколько возможных классов может иметь объект на входе. В качестве функции активации для каждого из таких нейронов используют софт макс. Софт макс определяется выражением 1.9

$$SM_{\text{i}} = \frac{e^{y_{\text{i}}}}{\sum_{i=1}^{N} e^{y_{\text{j}}}}, \tag{1.9}$$

где y_{i} – результат на выходе из нейрона, к которому применяется функция активации, N – число нейронов в выходном слое, y_{j} – результат на выходе из j нейрона выходного слоя.

Знаменатель в выражении 1.9 отвечает за нормировку. Таким образом, каждый из нейронов выходного слоя показывает вероятность принадлежности объекта на входе нейронной сети к некоторому классу, а сумма всех этих

вероятностей будет равна 1.

1.5 Виды нейронных сетей

1.5.1 Перцептрон

Перцептрон – математическая модель восприятия информации головным мозгом. Перцептрон состоит из трёх типов элементов, а именно: поступающие от сенсоров сигналы передаются ассоциативным элементам, а затем реагирующим элементам [8].

Каждый из типов элементов относится к определенному слою в архитектуре нейронной сети. Так все сенсоры располагаются на входном слое, ассоциативные элементы находятся на одном или нескольких скрытых слоях, реагирующие элементы занимают выходной слой. Общий вид перцептрона с тремя слоями приведен на рисунке 1.1.

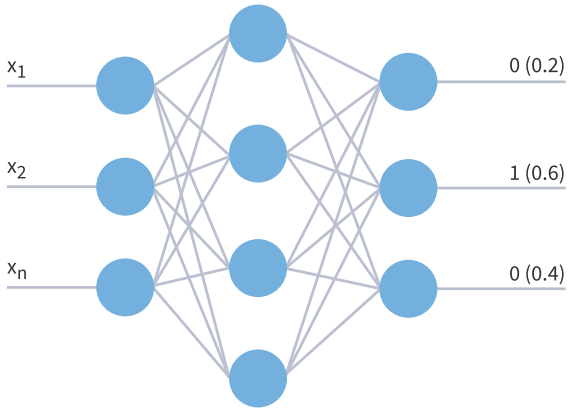

Рисунок 1.1 – Вид перцептрона с тремя слоями

На этом рисунке x_1, x_2 и x_3 обозначают входы перцептрона, которые поступают на входной слой, следующие 4 нейрона образуют слоя, их выходы поступают на 3 нейрона выходного слоя.

Увеличение числа скрытых слоев или числа нейронов на этом слое не всегда приводит к улучшению точности работы нейронной сети, поэтому данные параметры, как правило, подбираются экспериментальным путем. Число нейронов на выходном слое соответствует числу классов, по которым проводится классификация. На входном слое перцептрона число нейронов равно числу пикселей на изображении, которые подаются на вход нейронной

сети.

Для нейронов скрытого слоя применяется функция активации Relu, а для нейронов выходного слоя – софт макс.

1.5.2 Сверточные нейронные сети

Свёрточная нейронная сеть — нейронная сеть, в которой присутствует слой свёртки [9]. Свертка из себя представляет некоторую маску, которая называется ядром. Маска накладывается на пиксели исходного изображения с некоторым шагом, далее значения в маске перемножаются со значениями, которые эта маска покрыла, и результаты перемножений суммируются. Полученная сумма добавляется в результирующую матрицу сверточного слоя. Для сохранения размеров исходного изображения к нему добавляются столбцы и ряды из нулей перед началом свертки.

В случае когда на вход сверточному слою поступает трехканальное изображение, ядро свертки будет не двумерным, а трехмерным. Оно будет состоять из трех матриц – по одной для каждого канала. После применения свертки поочередно к каждому из каналов результаты суммируются и записываются в результирующую матрицу.

На одном сверточном слое к входной матрице может применяться не одна свертка, а сразу несколько. В таком случае каждая свертка считается по отдельности и записывается в свою результирующую матрицу. Результатом работы такого слоя будет несколько каналов с матрицами. Число каналов равно числу фильтров.

Таким образом, сверточный слой определяется следующими величинами:

- $padding$ – число нулевых строк и столбцов, которые добавляются к исходному изображению;
- $stridex$ – шаг свертки по столбцам;
- $stridey$ – шаг свертки по строкам;
- N – число каналов на входе;
- M – число каналов на выходе.

Помимо сверточных слоев в сверточной нейронной сети присутствуют слой субдискретизации и полносвязный слой. В слое субдискретизации также

присутствует свертка, которая с некоторым шагом проходится по входной матрице только вместо перемножения элементов и последующего суммирования выполняется какая-либо другая операция, к примеру, выбор наибольшего элемента. С помощью слоя субдискретизации достигается устойчивость к небольшим сдвигам входного изображения, а также уменьшается размерность последующих слоёв. Полносвязный слой – обычный скрытый слой многослойного перцептрона, соединённый со всеми нейронами предыдущего слоя [9].

Общий вид сверточной нейронной сети приведен на рисунке 1.2.

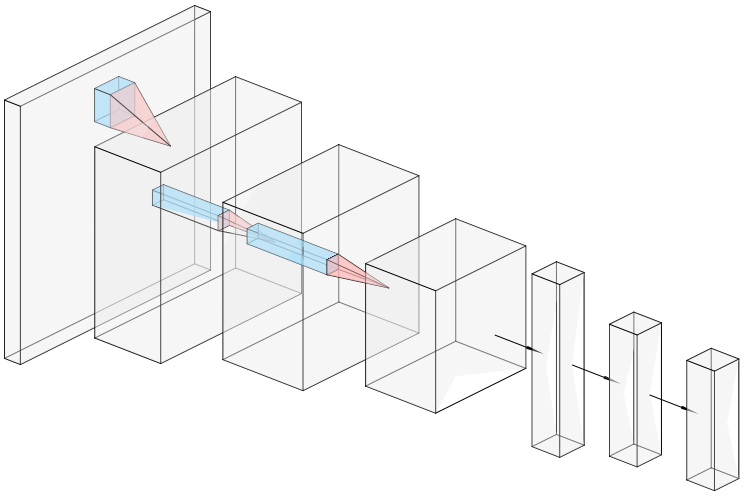

Рисунок 1.2 – Сверточная нейронная сеть с 3 сверточными слоями

На вход этой сверточной сети поступает изображение некоторой размерности, далее к нему применяется свертка с несколькими фильтрами, за счет чего происходит уменьшение размерности и увеличение числа каналов. Данная операция повторяется три раза. Далее все матрицы из всех каналов переводятся в линейный вектор и поступают вход полносвязного слоя. Дальше идет один скрытых слой и выходной аналогично тем, что были в перцептроне.

1.5.3 Капсульные нейронные сети

В капсульных нейронных сетях присутствует капсульный слой. Капсула строится на основе искусственного нейрона, но вместо скалярной расширяет его до векторной формы, что позволяет сохранять больше информации об объекте. На выходе мы получаем вектор, способный сохранять состояние объекта, например его позу [10]. Длина вектора определяет вероятность обнаружения

объекта, а его положение отвечает за состояние объекта.

Функция активации в этом случае имеет вид 1.10

$$v_{\mathrm{j}} = \frac{\|s_{\mathrm{j}}\|^2}{1 + \|s_{\mathrm{j}}\|^2} \frac{s_{\mathrm{j}}}{\|s_{\mathrm{j}}\|}, \tag{1.10}$$

где v_{j} – выходной вектор капсулы j, s_{j} – входные данные. Правая часть этого уравнения делает входной вектор единичным, а левая выполняет масштабирование таким образом, чтобы чем длиннее был входной вектор, тем ближе длина выходного была к единице, и чем меньше длина входного, тем ближе длина выходного к нулю.

Для всех капсул, кроме первого слоя, вход s_{j} является взвешанной суммой по всем векторам предсказаний u_{ji} из капсул в нижележащем слое, которые получаются путем умножения выходного u_{i} капсулы из нижележащего слоя на весовую матрицу W_{ij}. Таким образом вход в капсулу s_{j} определяется выражением 1.11

$$s_{\mathrm{j}} = \sum_i c_{\mathrm{ij}} u_{\mathrm{ji}}, \tag{1.11}$$

где c_{ij} – коэффициент связи между капсулами i и j, который определяется выражением 1.12

$$c_{\mathrm{ij}} = \frac{e^{b_{\mathrm{ij}}}}{\sum_k b_{\mathrm{ik}}}, \tag{1.12}$$

где b_{ij} – вероятность того, что капсула i связана с капсулой j. Эти вероятности итеративно обновляются путем измерения соответствия между текущим выходом v_{j} капсулы j и предсказанием u_{ji} капсулы i. Соответствие считается как скалярное произведение v_{j} на u_{ji}.

Одной из проблем сверточных нейронных сетей является их неспособность сохранять пространственную информацию о входных данных. Например, при обработке изображений обычная нейронная сеть может утерять информацию о положении объектов на изображении. Капсульные нейронные сети решают эту проблему, сохраняя информацию о пространственной структуре входных данных.

1.6 Проблема переобучения нейронной сети

Проблема переобучения в нейронных сетях заключается в том, что модель слишком хорошо запоминает данные из обучающей выборки, не обобщая свои знания на новые, ранее не встречавшиеся данные. Это происходит из-за того, что модель адаптируется к обучающим примерам, вместо того, чтобы учиться классифицировать новые данные [11]. Признаком переобучения модели является существенно большое значение ошибки распознавания на тренировочной выборке, нежели на тестовой. Зачастую переобучение появляется из-за использования слишком сложных моделей, либо наборов данных, в которых вхождения похожи друг на друга [11].

Недообучение - это противоположная проблема переобучения нейронных сетей. Оно характеризуется тем, что алгоритм обучения не достигает удовлетворительной точности на обучающем множестве. Это может быть связано с тем, что выбрана слишком простая модель или недостаточно обучающих примеров. В результате модель не сможет классифицировать данные в более сложных случаях. [11].

Примеры недообученной, переобученной и оптимально обученной нейронной сети приведены на рисунке 1.3.

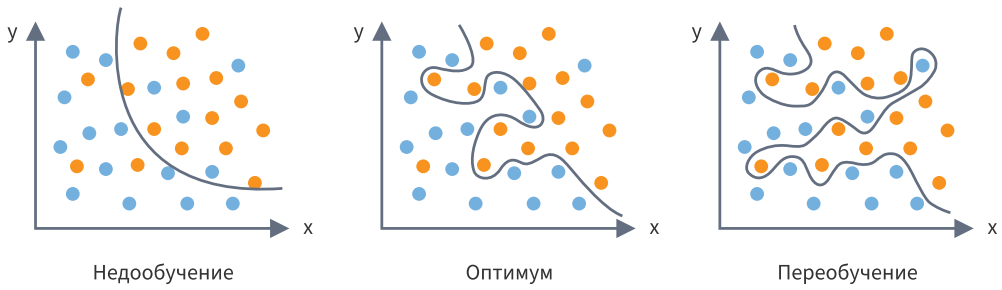

Рисунок 1.3 – Пример переобучения, недообучения и оптимального обучения

В этом примере нейронная сеть используется для разделения входного множества объектов на два класса. В первом случае нейронная сеть является недообученной, так как вероятность ошибки равна 0.22 и может быть еще уменьшена за счет использование более сложной формы зависимости.

На примере по середине нейронная сеть строит общую зависимость для данного набора данных, не подгоняя значения под аномальные элементы для

рыжего и голубого класса слева и справа соответственно.

Пример справа показывает переобученную нейронную сеть, которая строит зависимость, подгоняя параметры под аномальные параметры обучающей выборки.

Для борьбы с переобучением можно использовать следующие способы:

- аугментация обучающей выборки;
- метод раннего останова;
- регуляризация;
- батч нормализация.

1.6.1 Аугментация

Первый способ борьбы с переобучением – аугментация обучающей выборки. Аугментацией называется этап обучения нейронных сетей, состоящий в модификации обучающих изображений (поворот, масштабирование, зеркальное отражение и т. д.) по определенному правилу с целью расширить обучающую выборку и повысить ее разнообразие [12].

Существует три основных вида аугментации:

- геометрическая аугментация – изменение геометрических параметров изображения, таких как поворот, масштабирование, сдвиг и отражение;
- цветовая аугментация – изменение цветовых параметров изображения, таких как яркость, контрастность и насыщенность;
- добавление шума.

Аугментация первого типа обычно улучшает качество работы сверточных нейронных сетей, так как такие сети не инвариантны к масштабу, и изменение масштаба изображения значительно повышает разнообразие данных, позволяя сети обучаться на более разнообразных наборах данных [12]. В статье [13] описывается повышение точности распознавания нейронной сети на 10 процентов за счет использования аугментации масштаба.

Аугментации второго типа предполагают случайное изменение компонент R, G, B цвета пикселей изображения. Это один из самых эффективных

методов аугментации данных, потому что нейросети без этой аугментации имеют тенденцию к заучиванию правил вида «сумма цветов пикселей в области». Также такая аугментация может улучшить способность распознавания сети при различных условиях освещенности [12].

Аугментация добавлением шума на изображение повышает устойчивость модели к шуму на реальных изображениях.

1.6.2 Метод раннего останова

Метод раннего останова после каждой эпохи обучения проверяет точность модели на обучающей и тестовой выборках. Обучение нейронной сети начинается при случайных значениях весов, и с каждой эпохой обучения точность на обучающей выборке будет повышаться, а на тестовой точность сначала будет расти, потом в момент, когда сеть будет достаточно обучена, зафиксируется на некотором значении, а потом начнет падать из-за переобучения модели.

Суть метода раннего останова заключается в отслеживании точности модели на тестовой выборке и остановке обучения в момент, когда она начинает расти.

К преимуществам данного метода можно отнести:

- сокращение времени обучения, так как нейронная сеть не будет обучаться, когда в этом уже нет необходимости;
- отсутствие дополнительных затрат на дополнение обучающей выборки.

1.6.3 Регуляризация

Метод регуляризации заключается в ограничение значений весовых коэффициентов нейронной сети, что делает их распределение более равномерным. Это достигается за счет добавления некоторого штрафа за увеличение весов нейронной сети в функцию потерь.

Существует три основных вида регуляризации [14]:

- L1 регуляризация, которая также называется Лассо регуляризацией, она добавляет штраф от суммы абсолютных значений весов модели;
- L2 регуляризация, которая также называется регуляризацией Тихонова, она добавляет штраф от суммы квадратов весов модели;

- дропаут, который случайным образом удаляет связи между нейронами.

В пером виде регуляризации новая функция потерь описывается выражением 1.13

$$L_{\text{new}} = L_{\text{old}} + \lambda \sum_{i=1}^{N} |w_{\text{i}}|, \tag{1.13}$$

где L_{new} – новое значение функции потерь, полученное после регуляризации, L_{old} – значение функции потерь до проведения регуляризации, λ – коэффициент штрафования весов, N – число весов в модели, w_{i} – значение i-го веса модели.

При коэффициенте λ равном нулю никакой регуляризации не будет и модель переобучится. При повышении коэффициента штрафования модель будет приближаться к оптимальной, то есть значение ошибки на тестовой выборке будет падать, но чем больше значение этого коэффициента, тем ближе значения всех весов будут к нулю, тем дальше модель отклоняется от локального минимума функции потерь до регуляризации и тем больше растет ошибка модели не тренировочной выборке. Таким образом, значение коэффициента λ должно подбираться экспериментально во время обучения. Чем меньше ошибка на тренировочной выборке, тем лучше подобран коэффициент штрафования.

В регуляризации Тихонова штраф считается не по сумме абсолютных значений, а по сумме квадратов и выражается зависимостью 1.14

$$L_{\text{new}} = L_{\text{old}} + \lambda \sum_{i=1}^{N} w_{\text{i}}^2, \tag{1.14}$$

где все параметры аналогичны параметрам в выражении 1.13.

Главное различие между двумя методам заключается в том, что регуляризация Лассо уменьшает коэффициент менее важной характеристики до нуля, полностью удаляя ее из рассмотрения, а регуляризация Тихонова уменьшает веса, но не делает их равными нулю [14].

Еще одним видом регуляризации является дропаут. Суть метода заключается в том, что на каждой итерации обучения нейронной сети все связи между нейронами удаляются с некоторой вероятностью p. Иными словами это означает, что на каждой итерации обучения модели значение каждого веса w_{i} нейронной сети может быть на одну итерацию приравнено к нулю с

некоторой вероятностью p.

Таким образом, регуляризация борется с проблемой переобучения нейронной сети и повышает ее обобщающую способность [14]. К недостатками регуляризации можно отнести то, что:

- добавление штрафа или удаление некоторых связей может привести к ухудшению точности модели на обучающих данных;
- нельзя заранее оптимальным образом определить коэффициент штрафования весов λ и вероятность удаления связи между нейронами p.

1.6.4 Нормализация

Обычно при обучении нейронной сети шаг градиентного спуска делается не по одному конкретному примеру, а сразу по некоторому набору обучающих примеров. Такой подход имеет следующие преимущества [14]:

- усреднение градиента по нескольким примерам представляет собой апроксимацию градиента по всему тренировочному множеству, и чем больше примеров используется в одном мини-батче, тем точнее это приближение, использование всего обучающего множества невозможно в силу ограничений вычислительных ресурсов;
- в глубоких нейронный сетях к каждому примеру в отдельности требуется применить большое число последовательных операций, в случае использования некоторого набора обучающей примеров, можно выполнять эти последовательные операции в параллельном режиме для каждого примера в отдельности.

При таком подходе к обучению и использовании глубоких нейронных сетей возникает проблема, связанная с тем, что изменение распределения активаций выходов первых слоев на очередном шаге градиентного спуска приводит к сдвигу распределения данных во всех последующих слоях, что затрудняет их обучение и может ухудшить результаты. Для борьбы с этой проблемой используется батч-нормализация, которая позволяет нормализовать выходы каждого слоя в процессе обучения. Это делает распределение данных более стабильным и уменьшает влияние сдвига распределения на

последующие слои. Такая проблема получила название внутреннего сдвига переменных.

В исследованиях, приведенных в статье [15], говорится, что процесс обучения сходится быстрее, когда входы нейронной сети нормализованы, то есть их математическое ожидание приведено к нулю, а матрица ковариаций – к единичной. Если применять нормализацию к входам каждого слоя, то удастся избежать проблемы внутреннего сдвига переменных.

Для выполнения нормализации требуется предварительно рассчитать математическое ожидание и дисперсию элементов батча, которые определяются выражениями 1.15 и 1.16 соответственно.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_{\text{i}}, \tag{1.15}$$

где μ – математическое ожидание элементов бача, N – размер бача, x_{i} – i-ый элемент бача.

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{\text{i}} - \mu)^2, \tag{1.16}$$

где σ – дисперсия элементов бача, N – размер бача, x_{i} – i-ый элемент бача, mu – значение математического ожидания, посчитанное по формуле 1.15.

Тогда нормализацию входов можно проводить, используя выражение 1.17

$$\hat{x_{\text{i}}} = \frac{x_{\text{i}} - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{1.17}$$

где $\hat{x_{\text{i}}}$ – нормализованное значение i-го входа, x_{i} – ненормализованное значение i-го входа, μ и σ – математическое ожидание и дисперсия, посчитанные по формулам 1.15 и 1.16 соответственно, ϵ – некоторая константа, которая нужна для предотвращения деления на ноль.

Такая нормализация имеет существенный недостаток: в случае, если в качестве функции активации слоя используется сигмоидальная функция, например логистическая 1.4, то после нормализации нелинейность, которую давала эта функция активации, пропадет, так как большинство значений будут попадать в область, где эта функция ведет себя линейно, и функция активации фактически станет линейной [16].

Для того, чтобы компенсировать этот недостаток, слой нормализации

должен быть способен в некоторых случаях практически никак не менять входные значения. Достигается это при помощи введения двух новый коэффициентов: коэффициент масштарбирования и сдвига нормализации. Итоговое выражение для слоя нормализации определяется зависимостью 1.18

$$y_{\mathrm{i}} = \gamma_{\mathrm{i}}\hat{x_{\mathrm{i}}} + \beta_{\mathrm{i}}, \tag{1.18}$$

где y_{i} – i-ый выход слоя нормализации, $\hat{x_{\mathrm{i}}}$ – величина, полученная из выражения 1.17, γ_{i} и β_{i} – коэффициенты масштабирования и сдвига, которые настраиваются во время обучения модели.

Значения математического ожидания и дисперсии во время обучения от батча к батчу будут изменяться, но на этапе тестирования модели все изменяемые параметры должны быть зафиксированы. Для того, чтобы определить значения математического ожидания и дисперсии на этапе тестирования, эти величины накапливаются во время обучения с использованием экспоненциального скользящего среднего, которое определяется зависимостью 1.19

$$EMA_{\mathrm{t}} = \alpha * x_{\mathrm{t}} + (1 - \alpha) * EMA_{\mathrm{t}-1}, \tag{1.19}$$

где EMA_{t} – значение экспоненциального скользящего среднего в точке t, $EMA_{\mathrm{t}-1}$ – значение экспоненциального скользящего среднего в точке t минус 1, причем значение экспоненциального скользящего среднего в нуле EMA_{t} равно x_0, x_{t} – значение исходной функции, в нашем случае это математическое ожидание или дисперсии, в момент времени t, α – коэффициент характеризующий скорость уменьшения весов, принимает значение от 0 и до 1, чем меньше его значение тем больше влияние предыдущих значений на текущую величину среднего.

В случае, когда входной бачт описывается кортежем (N, C, H, W), где N – число элементов в батче, C – число каналов в каждом элементе, H и W – высота и ширина каждого изображения, нормализация считается по всем пикселям, всех изображений по каждому из каналов.

1.7 Ансамблевые методы

Ансамблевые методы классификации основаны на том, что несколько классификаторов обучаются на одном и том же наборе обучающих данных,

а затем их прогнозы объединяются для классификации элементов тестового набора данных. Математическим обоснованием этой идеи служит теорема Кондорсье о жюри присяжных [17].

Классификатор называется слабым, если его ошибка на обучающей выборке менее 50 процентов, но больше нуля. Тогда, объединив предсказания нескольких таких классификаторов, можно достичь более точности классификации на элементах тестовой выборки [17].

Выделяют 4 основных метода ансамблевой классификации [17]:

- бэггинг;
- бустинг;
- стекинг.

Идея бэггинга состоит в том, что если размер обучающей выборки не велик, то можно создать много случайных выборок из исходной путем отбора некоторых элементов, и обучить слабые классификаторы на эти подвыборки. Таким образом, каждая модель имеет свой набор обучающих примеров и старается сделать предсказания на основе своего подмножества данных. Затем результаты всех моделей комбинируются для получения итоговых предсказаний.

Бустинг – это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов [17]. Таким образом, при бустинге каждая последующая модель обучается на ошибках предыдущей и старается их компенсировать, повышая точность классификации общей модели.

Идея стекинга заключается в введении некоторого алгоритма классификации и его обучении. При стекинге, в отличие от бустинга и бэггинга, классификаторы должны быть разной природы [17]. Обучение модели при стекинге можно свести к следующим трем шагам:

- обучающая выборка разбивается на две непересекающихся подвыборки;
- первая подвыборка используется для обучения классификаторов;

- вторая для обучения алгоритма, который на вход принимает выходы со всех классификаторов.

Главным недостатком стекинга является деление обучающей выборки на две части.

1.8 Существующие архитектуры

1.8.1 LeNet

LeNet является одной из первых архитектур сверточных нейронных сетей. Ее первое описание приведено в статье [18]. Модель состоит из 5 сверточных слоев, за которыми следует 2 полносвязных слоя. Общий вид модели представлен на рисунке 1.4.

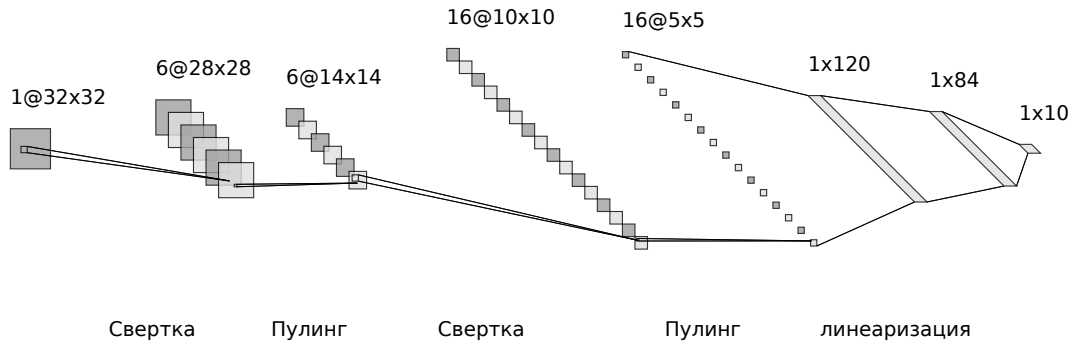

Рисунок 1.4 – Архитектура LeNet

На вход поступает одноканальное изображение размером 32 на 32 пикселей. Далее к нему применяется свертка с 6 фильтрами с ядром 5 на 5 пикселей, которая выделяет основные элементы на изображении. После свертки применяется пулинговый слой, который уменьшает размер изображения в два раза и усредняет значения пикселей, что позволяет уменьшить количество параметров.

Далее применяется свертка с 16 фильтрами и ядром 5 на 5, которая используется для выделения более сложных признаков. После к выходам этой свертки применяется пулинговый слой, аналогичный предыдущему.

После все матрицы из всех каналов пулинговго слоя переводятся в 1 вектор и поступают на вход полносвязного слоя.

Недостатком такой архитектуры является проблема затухания градиента при обучении нейронной сети [19]. Для решения этой проблемы можно использовать Max Pooling между сверточными слоями.

1.8.2 AlexNet

AlexNet по своему принципу не сильно отличается от LeNet. В AlexNet используется больше сверточных слоев, а в слоях субдисретизации используется Max Pooling. В качестве функции активации полносвязного слоя используется ReLU. Общая схема архитектуры сети приведена на рисунке 1.5.

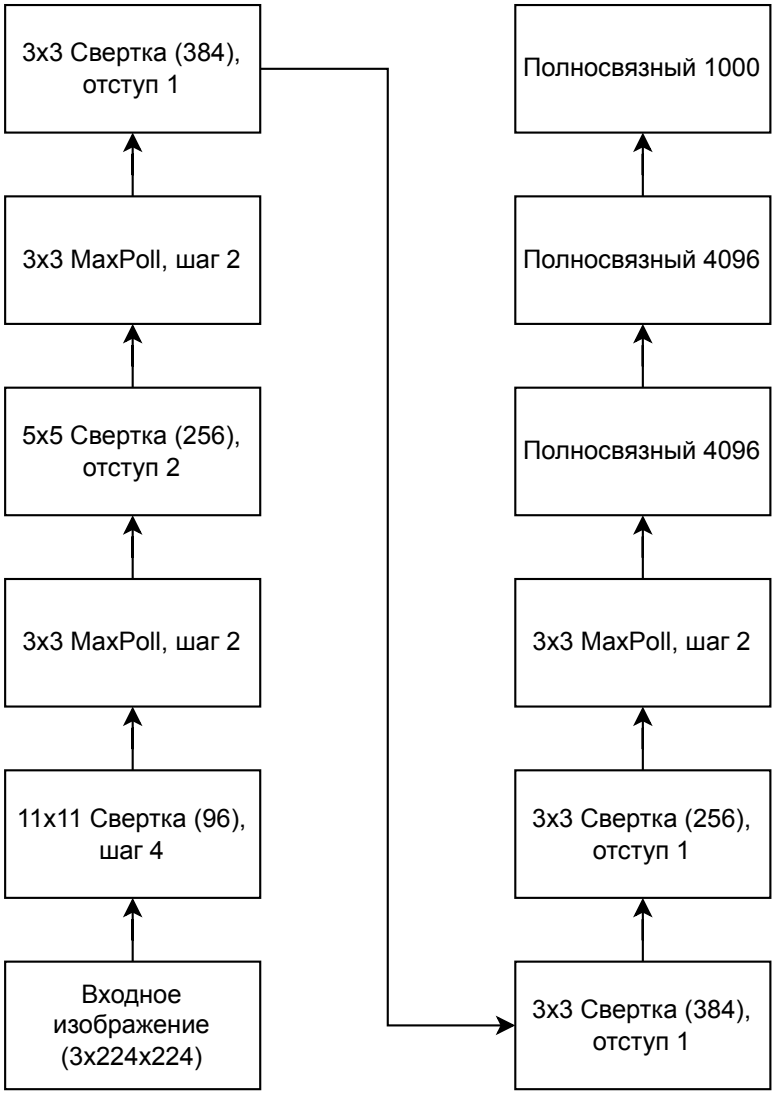

Рисунок 1.5 – Архитектура AlexNet

Преимуществом данной архитектуры является высокая точность распо-

знавания – в 2012 AlexNet показала рекордный результат в точности распознания 1000 различных объектов в соревновании ImageNet.

Главным недостатком является большое число параметров, использующихся при обучении, – около 60 миллионов [19]. Такое количество параметров требует значительно большие вычислительные мощности и память в сравнении с LeNet. Также потребуется больше элементов в обучающей выборке, чтобы корректно настроить эти параметры.

1.8.3 GoogLeNet

В GoogLeNet было введено понятие Inception блока, формальное представление которого приведено на рисунке 1.6.

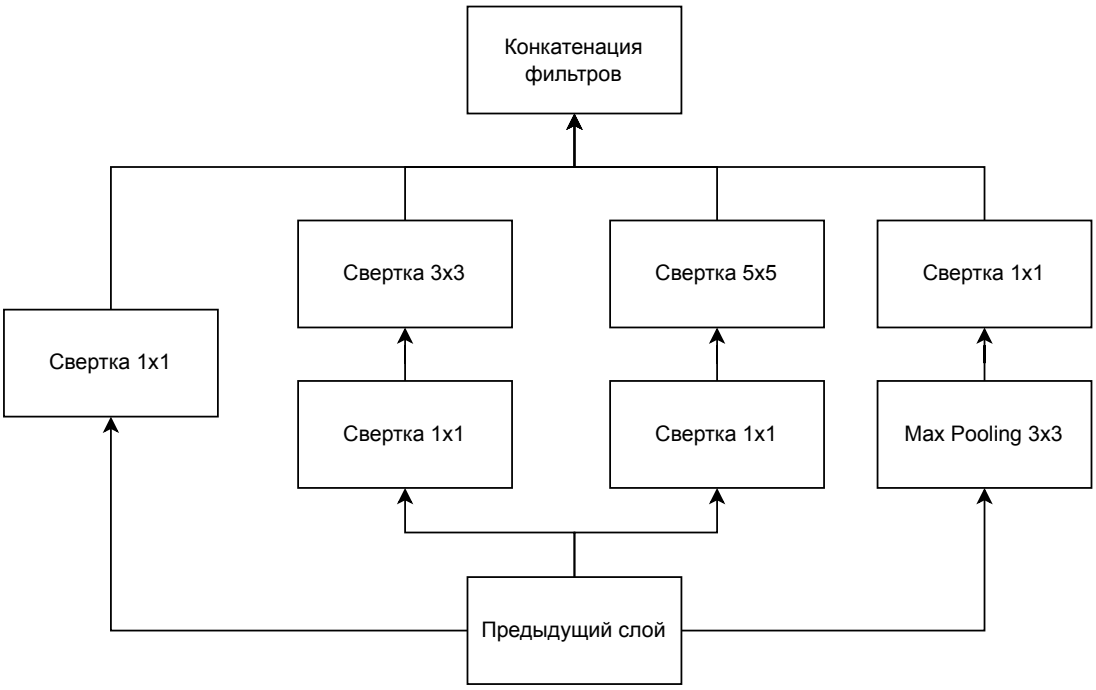

Рисунок 1.6 – Формальное представление Inception блока

В таком блоке выходы с предыдущего слоя параллельно обрабатываются сверткой 1 на 1, блоком из сверток 1 на 1 и 3 на 3, блоком из сверток 1 на 1 и 5 на 5, блоком из Max Pooling размером 3 на 3 и свертки 1 на 1. Далее выходы со всех этих блоков объединяются и передаются дальше.

Фильтр 1 на 1 используется для уменьшения количества параметров в сверточных слоях и ускорения вычислений. Он позволяет сократить количество каналов входного изображения до более низкого значения, что уменьшает количество вычислений при применении более крупных фильтров. Кроме того,

фильтры 1 на 1 могут использоваться для комбинирования признаков разных каналов, что улучшает качество классификации [20].

Такой подход имеет два основных преимущества [20]:

- возможность увеличения количества блоков на каждом этапе без неконтролируемого роста вычислительной сложности;
- визуальная информация обрабатывается в различных масштабах, а затем агрегируется, чтобы на следующем этапе можно было абстрагировать признаки из разных масштабов одновременно.

Использование таких блоков в архитектуре GoogLeNet позволяет сократить число параметров примерно в 12 раз по сравнению с AlexNet, при этом точность классификации сети не падает [20].

Для борьбы с затуханием градиента в GoogLeNet используется следующий прием: помимо классификатора в конце нейронной сети добавляется еще один после третьего Inception блока и еще один после 6. Во время обучения функция потерь считается не только по значениям из последнего классификатора, но и по 2 добавленным, домжноженным на некоторый коэффициент. Итоговое выражение для подсчета функции потерь определяется следующей зависимостью 1.20

$$TL = l + k * (l_1 + l_2), \tag{1.20}$$

где TL – общее значение функции потерь, которое нужное уменьшать во время обучения, l – значение функции потерь, посчитанное по выходам из последнего классификатора, k – некоторый коэффициент значимости, на который домножаются значения функции потерь, посчитанные по выходам двух других классификаторов, в GoogLeNet этот коэффициент равен 0.3, l_1 и l_2 – это значения функции потерь посчитанные по выходам классификаторов, добавленный в начале и в середине соответственно.

1.8.4 CapsNet

CapsNet является капсульной нейронной сетью. В статье [21] описывается работа этой сети, которую можно разделить на 4 основных шага

- слой свертки;
- слой primary caps;

- направления по соглашению;
- слой digit caps.

Общий вид архитектуры нейронной сети CapsNet представлен на рисунке 1.7.

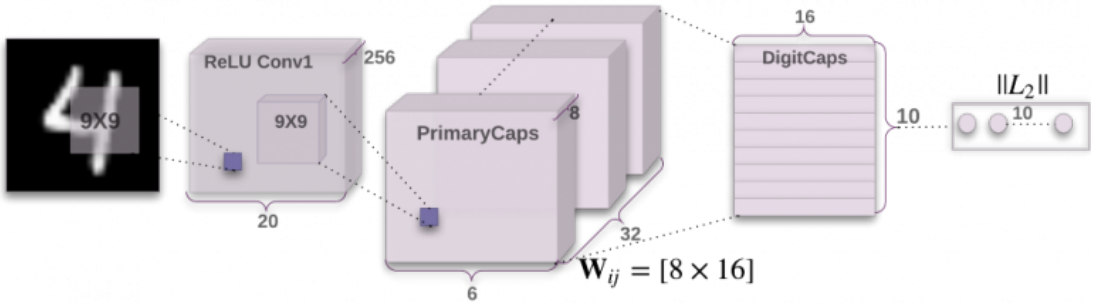

Рисунок 1.7 – Архитектура CapsNet

На вход нейронной сети поступает изображение размером 28 на 28 пикселей. В данном примере изображение имеет только 1 канал, но в реальности может быть и 3 канала для цветного изображения, и 4, если используется дополнительный альфа канал, отвечающий за прозрачность. Число каналов на входе влияет только на размерность ядра свертки на первом шаге.

На первом шаге к входному изображению применяется обычный сверточный слой с ядром 9 на 9 и 256 фильтрами. В качестве функции активации слоя используется ReLU. На выходе из слоя получается 256 каналов размером 20 на 20 пикселей. За счет этого слоя из изображения выделяются базовые признаки, такие как грани объектов.

Начало работы слоя primary caps похоже на обычный сверточный слой, только ядро свертки применяется не к каждому каналу по отдельности, а ко всем сразу, то есть размерность ядра свертки на этом слое 9x9x256. Шаг ядра свертки на этом слое равен двум. Число фильтров также равно 256, поэтому на выходе из сверточного этапа слоя primary caps получается 256 каналов размером 6 на 6 пикселей. На этом этапе выделяются более сложные формы из уже найденных граней.

Далее выходы с предыдущего этапа разбиваются на 32 части по 8 элементов в каждой. Каждый из таких кусков имеет 8 каналов размером 6 на 6 пикселей и называется капсульным слоем. Каждый капсульный слой состоит из 36 капсул, состоящих из 8 элементов. Далее к каждой капсуле применяется

функция активации, описывающаяся зависимостью (1.10), которая изменяет длину входного вектора и делает ее значение в диапазоне от 0 до 1, при этом угол поворота вектора не изменяется.

На шаге направления по соглашению определяется, какие капсулы несут полезную нагрузку и должны быть переданы на следующий слой, а какие должны быть отброшены. Каждая капсула, основываясь на самой себе, пытается предсказать активацию следующего слоя, далее из всех предсказаний капсул выбирается самое частое, и капсулы, которые сделали это предсказание, передаются дальше.

К примеру, нейронная сеть должна определять 1 из 10 различных классов. Предсказание капсулы для каждого из классов создается путем перемножения соответствующего вектора на матрицу весов для каждого класса. Общее число предсказаний равно количеству капсул, умноженному на число классов. В архитектуре CaspNet это значение равно 11520.

Далее определяется, какие из предсказаний лучше всего соотносятся друг с другом и именно они будут переданы на следующий слой. Так как каждое из предсказаний является вектором, определить лучше всего соотносящиеся можно при помощи следующего алгоритма: сначала считается средний вектор по всем предсказанием при этом каждое предсказание имеет одинаковый вес, дальше считается среднее по всем векторам, но важность каждого предсказания тем меньше, чем больше его расстояние от среднего. Эта операция повторяется еще несколько раз, и наиболее согласованными предсказаниями являются те, которые ближе всего находятся к полученному среднему вектору. Предсказания с наибольшей согласованностью отправляются в следующий слой.

На вход слоя digit caps поступает по одному предсказанию на каждый класс, который должна определять нейронная сеть. Длинна вектора каждого из предсказаний определяет вероятность того, что на вход было подано изображение с соответствующим классом.

Способ тренировки такой сети отличается от обычной сверточной. В сверточных сетях обучение нацелено лишь только на правильное определение класса объекта. В капсульных сетях помимо определения класса сеть обучается на правильную реконструкцию входного изображения по выходному вектору.

1.9 Сравнение решений

В исследовании [22] приводится сравнение алгоритмов для решения задачи классификации на наборе изображений рукописных цифр MNIST [23] и наборе изображений техники 10 разных типов CIFAR-10 [24] на основе сверточной нейронной сети и капсульной нейронной сети.

Алгоритмы сравниваются по следующим критериям:

- время обучения нейронной сети;
- число параметров модели;
- точность классификации модели на тестовой выборке.

Сверточная нейронная сеть, участвующая в исследовании, имеет следующие слои:

- слой свертки с 128 фильтрами, ядром размера 5 на 5 пикселей и шагом 2;
- слой max pooling с ядром размера 3 на 3 пикселя и шагом 2;
- слой свертки с 64 фильтрами, ядром размера 5 на 5 пикселей и шагом 1;
- слой max pooling с ядром размера 3 на 3 пикселя и шагом 2;
- полносвязный слой из 1024 нейронов;
- полносвязный слой из 10 нейронов.

В качестве функции активации для всех слоев, кроме последнего, используется ReLU (1.6), на последнем слое используется софт макс (1.9).

Капсульная нейронная сеть , участвующая в исследовании, имеет следующие слои:

- слоя свертки с 256 фильтрами, ядром размера 9 на 9 пикселей и шагом 1;
- слой primary caps, состоящий из 32 капсульных слоев из капсул размерности 8;

- слой digit caps, полученный путем перемножения капсул на матрицы предсказаний размерности 8 на 16 элементов.

Обучение, описанных выше нейронный сетей, происходило на 50 эпохах.

Исходя из результатов исследования [22], время обучения на наборе данных MNIST для капсульной нейронной сети в 30 раз больше, чем в сверточной. Точность классификации для обоих видов сетей составила 99 процентов. В капсульной нейронной сети используется примерно в три раза больше обучаемых параметров, что приводит к большим затратам оперативной памяти.

На наборе данных CIFAR-10 были получены следующие результаты: время обучения капсульной нейронной сети в 10 раз больше, чем у сверточной, число параметров в капсульной приблизительно в три раза больше, чем в сверточной, точность распознавания у капсульной 53 процента, а у сверточной – 51.

Таким образом, капсульная нейронная сеть хоть и имеет немного большую точность на многоклассовой классификации, время, которое требуется на ее обучение, и число параметров, которое нужно обучить, в разы превосходит аналогичные значения для сверточной сети. Это означает, что для работы капсульной нейронной сети требуются большие вычислительные ресурсы и больший размер обучающей выборки, чем для сверточной нейронной сети.

Согласно исследованиям [25] сверточные нейронные сети имеют большую точность распознавания по сравнению с перцептроном, а также большую устойчивость к шумам и скорость обучения за счет возможности распараллеливания алгоритма.

Так как ядро свёртки для каждой карты признаков одно, это позволяет нейронной сети научиться выделять признаки вне зависимости от их расположения во входном изображении, что не возможно в перцептронах.

Недостатком сверточных нейронных сетей является то, что отбрасывается потенциально полезная информация, теряются пространственные связи между объектами или их частями. Помимо этого, проблема заключается в неспособности сети определять положение объекта в пространстве, а также реагировать на его изменения (такие как поворот или смещение) [10].

Капсульные нейронные сети лучше реагируют на мелкие отличия по сравнению со сверточными сетями, так как свертка является загрублением, и снижают ошибку распознавания в другом ракурсе [26].

Критерии сравнения описанных выше подходов к построению нейронных сетей и результаты сравнения представлены в таблице 1.1.

Таблица 1.1 – Сравнение видов нейронных сетей

	Перцептрон	**Сверточные сети**	**Капсульные сети**
Число обучаемых параметров	больше всего	меньше всего	среднее
Точность распознавания	меньше всего	средняя	больше всего
Устойчивость к шумам	не устойчив	устойчивы	не устойчивы
Возможность к распараллеливанию	есть	есть	есть
Возможность реконструкции изображения	нет	нет	есть
Устойчивость к сдвигу	нет	есть	есть

1.10 Формализованная постановка задачи

Цель работы – разработать метод распознавания летательный аппаратов с аэрофотоснимков.

Для достижения поставленной цели требуется выполнить следующие задачи:

- провести анализ существующих программных подходов классификации летательных аппаратов с аэрофотоснимков (проведено выше);
- разработать метод классификации летальной техники на аэрофотоснимках;
- реализовать спроектированный метод;
- провести исследование точности распознавания модели на тестовой выборке при различных подходах к обучению.

На вход методу подается изображение с летательной техникой одного из

20 классов. Результатом работы метода является целое число от одного до 20, отвечающее за класс объекта, который был подан на входном изображении.

На входное изображение накладываются следующие ограничения:

- изображение сделано в дневное время суток;
- на изображении находится только один летательный аппарат, относящийся к одному из 20 классов, которые распознает модель;
- фотоснимки сделаны на высоте 600-1000 метров.

На рисунке 1.8 приведена IDEF-0 диаграмма уровня A0 метода распознавания летательных аппаратов с аэрофотоснимков с использованием нейронных сетей.

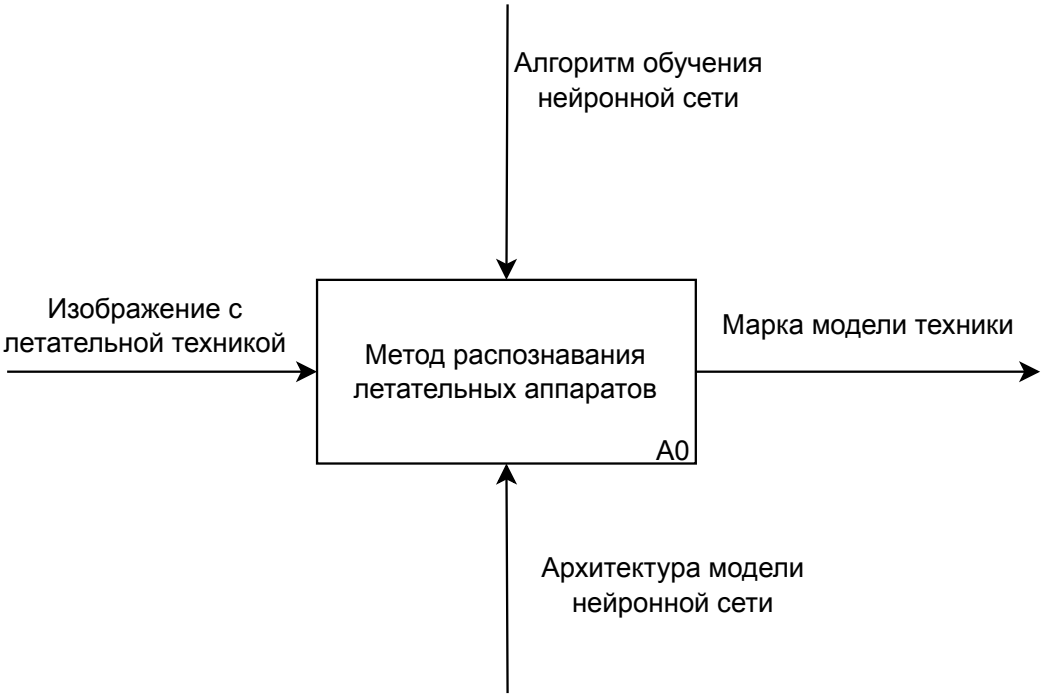

Рисунок 1.8 – IDEF-0 диаграмма метод распознавания летательных аппаратов с аэрофотоснимков

1.11 Вывод

Методы детерминированного подхода плохо подходят для задачи классификации изображений с аэрофотоснимков в силу того, что не устойчивы к шуму.

Дерево решений не применимо для поставленной задачи, так как на этапе построения дерева не известны критерии, по которым можно классифицировать входную информацию.

Среди нейрокомпьютерных алгоритмов выделяют алгоритмы обучения с учителем и без. Задача классификации объектов на изображении относится к алгоритмам обучения с учителем и может быть решена с использованием нейронных сетей.

Среди основный видов нейронных сетей: перцептрон, сверточные сети и капсульные сети. Большую точность классификации показывают сверточные и капсульные сети. Сверточные нейронные сети обучаются быстрее и требуют меньше обучаемых данных, чем капсульные.

Таким образом, для решения задачи распознавания летательных аппаратов с аэрофотоснимков, лучше всего подходят сверточные нейронные сети за счет большей точности распознавания в сравнении с перцептроном, а также за счет устойчивости к шуму и большей скорости обучении по сравнению с капсульными сетями. При обучении сверточной нейронной сети нужно использовать регуляризацию и нормализацию для борьбы с проблемой переобучения, а также выполнять аугментацию обучающей выборки для повышения обобщающей способности модели. В случае слабых классификаторов, нужно использовать методы ансамблевой обработки: бустинг, бэггинг и стэкинг.

2 Конструкторский раздел

3 Технологический раздел

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Самойлин Е.* АЛГОРИТМ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ НА ОСНОВЕ ГРАДИЕНТНОГО СОВМЕЩЕНИЯ ОБЪЕКТА С ЭТАЛОНОМ. — Сборник трудов XXV Международной научно-технической конференции, 2019. — с. 150.
2. *Мифтахова А. А.* ПРИМЕНЕНИЕ МЕТОДА ДЕРЕВА РЕШЕНИЙ В ЗАДАЧАХ КЛАССИФИКАЦИИ И ПРОГНОЗИРОВАНИЯ. — Поволжский государственный университет телекоммуникаций и информатики, 2016. — с. 7.
3. *Бабушкина Н. Е.* ВЫБОР ФУНКЦИИ АКТИВАЦИИ НЕЙРОННОЙ СЕТИ В ЗАВИСИМОСТИ ОТ УСЛОВИЙ ЗАДАЧИ. — Донской государственный технический университет, 2022. — с. 4.
4. *Антонов Г. В.* ПРОСТАЯ НЕЙРОННАЯ СЕТЬ И ЕЕ ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ. — ФГБОУ ВО Великолукская государственная сельскохозяйственная академия, 2021. — с. 11.
5. *Левченко К. М.* Нейронные сети. — Белорусский государственный университет информатики и радиоэлектроники, 2022. — с. 5.
6. *Барвинский Д. А.* Применение метода градиентного спуска в решении задач оптимизации. — Тенденции развития науки и образования, 2021.
7. *Апарнев А. Н.* Анализ функций потерь при обучении сверточных нейронных сетей с оптимизатором Adam для классификации изображений. — ВЕСТНИК МОСКОВСКОГО ЭНЕРГЕТИЧЕСКОГО ИНСТИТУТА. ВЕСТНИК МЭИ, 2020.
8. *Митина О.* ПЕРЦЕПТРОН В ЗАДАЧАХ БИНАРНОЙ КЛАССИФИКАЦИИ. — Национальная ассоциация ученых, 2021. — с. 6.
9. *Сикорский О. С.* Обзор свёрточных нейронных сетей для задачи классификации изображений. — Новые информационные технологии в автоматизированных системах, 2017. — с. 8.
10. *Алексеев И. П.* ПЕРСПЕКТИВЫ ПРИМЕНЕНИЯ КАПСУЛЬНЫХ НЕЙРОННЫХ СЕТЕЙ В РАСПОЗНАВАНИИ ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ. — ТИНЧУРИНСКИЕ ЧТЕНИЯ - 2021 «ЭНЕРГЕТИКА И ЦИФРОВАЯ ТРАНСФОРМАЦИЯ», 2021. — с. 4.

11. *Воронецкий Ю. О.*, *Жданов Н. А.* Методы борьбы с переобучением искусственных нейронных сетей // Научный аспект. — 2019. — т. 13, № 2. — с. 1639—1647.

12. *В. П. А.* Формирование обучающей выборки в задачах машинного обучения. Обзор // Информационно-управляющие системы. — 2021. — 4 (113). — с. 61—70.

13. *Z. C.* Cascade R-CNN: High Quality Object Detection and Instance Segmentation // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2021. — т. 43, № 5. — с. 1483—1498.

14. *О.А. П.* МЕТОДЫ И ПРОБЛЕМЫ ПЕРЕОБУЧЕНИЯ МНОГОСЛОЙНОЙ НЕЙРОННОЙ СЕТИ // ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В СТРОИТЕЛЬНЫХ, СОЦИАЛЬНЫХ И ЭКОНОМИЧЕСКИХ СИСТЕМАХ. — 2020. — с. 101—103.

15. *Lecun Y.* Efficient Backprop // Neural Networks: Tricks of the Trade. — 2019. — с. 99—48.

16. *Николенко С.* Глубокое обучение. Погружение в мир нейронных сетей. — Издательство Питер, 2018. — с. 477.

17. *С. К. Ю.* Ансамблевый метод машинного обучения, основанный на рекомендации классификаторов // Интеллектуальные системы. Теория и приложения. — 2015. — т. 19, № 4. — с. 37—55.

18. *Y. LeCun L. Bottou Y. B.*, *Haffner P.* Gradient-Based Learning Applied to Document Recognition. — Proceedings of the IEEE, 1998. — с. 46.

19. Different types of CNN Architectures. — Дата обращения: 28.04.2023. Режим доступа: `https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/`.

20. *C. Szegedy W. Liu Y. J.* Going Deeper With Convolutions. — Proceedings of the IEEE Conference on Computer Vision, Pattern Recognition (CVPR), 2015. — с. 12.

21. Understanding Capsule Networks. — Дата обращения: 28.04.2023. Режим доступа: `https://www.freecodecamp.org/news/understanding-capsule-networks-ais-allurin-new-architecture-bdb228173ddc`.

22. *J. X.* CapsNet, CNN, FCN: Comparative Performance Evaluation for Image Classification // International Journal of Machine Learning and Computing. — 2019. — т. 9, № 6. — с. 9.

23. MNIST dataset. — Дата обращения: 28.04.2023. Режим доступа: `https://www.tensorflow.org/datasets/catalog/mnist`.

24. CIFAR-10 dataset. — Дата обращения: 28.04.2023. Режим доступа: `https://www.tensorflow.org/datasets/catalog/cifar10`.

25. *Паршин С. Е.* Исследование параметров алгоритмов распознавания лиц. — Сборник научных трудов Новосибирского государственного технического университета, 2019. — с. 6.

26. *Береснев Д. В.* КЛАССИФИКАЦИЯ ГАЛАКТИК С ПОМОЩЬЮ КАПСУЛЬНЫХ НЕЙРОННЫХ СЕТЕЙ. — БГУИР, 2019. — с. 4.

ПРИЛОЖЕНИЕ А