МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет компьютерных наук
Кафедра цифровых технологий

Применение сверточных нейронных сетей к задаче классификации трехмерных моделей

ВКР Бакалаврская работа
02.03.01 Математика и компьютерные науки
Распределенные системы и искусственный интеллект

Допущено к защите в ГЭК __ . _____ . 2017

Зав. кафедрой _____ *С.Д. Кургалин, д. ф.-м. н., профессор*
Обучающийся _____ *А.А. Вакулин, 4 курс, д/о*
Руководители _____ *А.А. Крыловецкий, к. ф.-м. н., доцент*
_____ *И.С. Черников, к. ф.-м. н., ассистент*

Воронеж 2017

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ"
Факультет компьютерных наук
Кафедра цифровых технологий

УТВЕРЖДАЮ
заведующий кафедрой

_____________ ___________________________
подпись *расшифровка подписи*

ЗАДАНИЕ
НА ВЫПОЛНЕНИЕ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

ОБУЧАЮЩЕГОСЯ ____
фамилия, имя, отчество

1. Тема работы ______________________________, утверждена решением ученого совета _______________ факультета от ___.___.20___
2. Направление подготовки ______________________________________
шифр, наименование
3. Срок сдачи студентом законченной работы ___.___.20___
4. Календарный план:

№	Задание	Срок выполнения
1		
2		
3		
4		
5		
6		

Обучающийся _____________ ___________________________
подпись *расшифровка подписи*

Руководитель _____________ ___________________________
подпись *расшифровка подписи*

_ _

Выпускная квалификационная работа представлена на кафедру ___.___.20___

Рецензент ____
должность, ученая степень, ученое звание

Выпускная квалификационная работа на тему ______________________________

____

допущена к защите в ГЭК ___.___.20___

Заведующий кафедрой ___________________________ ___.___.20___
подпись, расшифровка подписи

Реферат

Бакалаврская работа 46 с., 17 рис., 3 прил.

КОМПЬЮТЕРНОЕ ЗРЕНИЕ, КЛАССИФИКАЦИЯ ТРЕХМЕРНЫХ ОБЪЕКТОВ, ДЕСКРИПТОР ФОРМЫ ПОВЕРХНОСТИ, ИНТЕГРАЛЬНОЕ СПИНОВОЕ ИЗОБРАЖЕНИЕ, СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ.

Объектом исследования являются возможные представления(дескрипторы) трехмерных объектов и технологии сверточных нейронных сетей.

Цель работы - разработка математических методов и алгоритмов классификации трехмерных объектов с помощью сверточных нейронных сетей

В процессе выполнения работы проводились эксперименты по вычислению дескрипторов трехмерных объектов и использованию сверточных нейронных сетей для классификации полученных дескрипторов.

В результате исследования была создана программная реализация системы распознавания трехмерных объектов с использованием сверточных нейронных сетей.

Степень внедрения ———–

Результаты работы системы показали способность сверточных нейронных сетей классифицировать дескрипторы трехмерных объектов в виде интегральных спиновых изображений.

Содержание

Введение

Трёхмерное компьютерное зрение является одной из быстро развивающихся подобластей компьютерного зрения. Это связано, прежде всего, с тем, что лишь совсем недавно появились вычислительные мощности, способные справиться с обработкой крупных объёмов трёхмерных данных. Кроме того, в настоящее время непрерывно увеличивается количество доступных трёхмерных моделей в сети Интернет, а также в специализированных базах данных.Работы по созданию программно-аппаратных комплексов для получения пространственных данных объектов реального мира и воссоздания (реконструкции) точных трёхмерных моделей успешно ведутся лабораторией Stanford Computer Graphics Laboratory (Стэнфорд, США), а также компанией DAVID Vision Systems GmbH (Кобленц, Германия). Модели поиска трёхмерных моделей в крупных базах данных разрабатываются в лабораториях Center for Geometry, Imaging and Virtual Environments (Утрехт, Нидерланды), Laboratoire d'Informatique Fondamentale de Lille (Вильнёв-д'Аск, Франция), Microsoft Research Cambridge (Кембридж, Великобритания), Princeton Shape Retrieval and Analysis Group(Принстон, США).

Здесь возникает потребность классификации моделей для дальнейшего их распознавания или поиска, а именно разработка соответствующей системы обработки баз данных трехмерных моделей.Разработкой методов и алгоритмов реконструкции и поиска трёхмерных моделей занимаются учёные всего мира: P. Min, M. Kazhdan, S. Rusinkiewicz (Принстон, США), R.C. Veltkamp (Утрехт, Нидерланды), Chi-Chou Kao (Тайнань, Тайвань), группа под управлением Dr.-Ing. Simon Winkelbach (Брауншвейг, Германия).

Центральной проблемой, возникающей при классификации или поиске в базах данных, содержащих трехмерные объекты, является разработка метода сравнения трёхмерных моделей. В данной работе представлена реализация подобного метода, состоящая из двух основных этапов: построение дескрипторов трехмерных моделей и сравнение их между собой.

На первом этапе информацию о форме поверхности трёхмерных полигональных моделей следует представить в достаточно сжатом и ёмком виде. Это продиктовано тем, что модели имеют разный масштаб, по-разному ори-

ентированы и локализованы в пространстве, что делает невозможным непосредственное сравнение трехмерных моделей. Удачным способом такого представления модели является ее глобальный дескриптор в виде интегрального спинового изображения[1]. Классификация таких дескрипторов трехмерных моделей требует сравнения интегральных спиновых изображений, представляющих собой массив пикселей, а не структур, описывающих трехмерные объекты.

Однако при более детальном рассмотрении поставленной задачи, в дальнейшем станет понятно, что интегральное спиновое изображение способно хранить информацию о форме поверхности только если целевая полигональная модель имеет определенное разрешение(среднюю длину ребра между точками). Алгоритм контроля разрешения трехмерных моделей также будет рассмотрен в данной работе.

После построения дескриптора трехмерной модели возникает второй этап представленного метода. Сравнение двух интегральных спиновых изображений довольно сложная задача, т.к. выделить какой-либо отличительный признак для определенного класса объектов из данных массивов на первый взгляд практически невозможно. Достаточно эффективным инструментом определения более сложных признаков из сырых массивов пикселов в настоящий момент является отдельный тип нейронных сетей, называемых свёрточными.В настоящий момент над задачей распознавания трехмерных объектов с помощью нейронных сетей работают ученые из Стэнфорда, Принстона и Мюнхенского технического университета. Создаются обширные открытые библиотеки доступных аннотированных 3D-данных такие как ScanNet или The Princeton Shape Benchmark, которые, в свою очередь, будут использоваться для обучения нейронных сетей.

В работе будут подробнее рассмотрены оба этих этапа и приведены результаты применения данного метода для небольшой базы данных трехмерных моделей.

1 Глобальные дескрипторы поверхности

1.1 Дескрипторы формы поверхности

Понятие дескрипторов формы поверхности можно определить, как отображение пространства трёхмерных моделей в некоторое многомерное конечное векторное пространство.

Формирование отображения, сохраняющего наибольшее количество информации о поверхности модели и при этом формирующего вектор соответствующих параметров наименьшей размерности, является основной целью разработки дескрипторов поверхности. Так же дескриптор формы поверхности должен быть инвариантен относительно таких преобразований, как трансляция, вращение и масштабирование. Таким образом он будет одинаково описывать модель вне зависимости от её масштаба, положения и ориентации в пространстве. Соответствие последним требованиям зачастую может быть достигнуто предварительной обработкой данных трёхмерной модели.

Дескрипторы формы поверхности возможно разделить на три основных класса рис.1.1: характеристики формы, графы и другие. Данные категории достаточно взаимосвязаны и не являются изолированными. Таким образом, дескрипторы, основанные на распределениях, содержат глобальные характеристики формы поверхности, а дескрипторы-гистограммы могут рассматриваться как дескрипторы, основанные на методе пространственных карт. Так же, например, графы могут восприниматься как глобальные характеристики формы поверхности.

С точки зрения сравнения трёхмерных моделей их характеристиками служат геометрические и топологические свойства их поверхности. Таким образом, измерение и сравнении этих характеристик определяет сходство или различие рассматриваемых моделей.

Для методов, базирующихся на характеристиках формы поверхности, можно выделить четыре подгруппы, которые различаются типами характеристик формы: глобальные характеристики, локальные характеристики, распределения и пространственные карты. Исключая локальные характеристики, во всех остальных подгруппах данного класса форма поверхности трёхмерных моде-

Рис. 1.1: Классификация дескрипторов формы поверхности

лей описывается единственным дескриптором, который представляет собой n-мерный вектор значений. Заметим, что размерность этого вектора фиксирована для всех моделей. В методе, который основан на локальных характеристиках, дескрипторы вычисляются для всех точек поверхности модели. Таким образом каждый дескриптор содержит информацию о форме небольшого участка поверхности модели.

К другим дескрипторам отнесем расширенное Гауссово изображение (Extended Gaussian Image (EGI)) [2], сложное расширенное Гауссово изображение (complex EGI) [3], трёхмерные моменты Зернике [4] и др.

1.2 Спиновые изображения

Основная идея спиновых изображений - это сопоставление опорной точке (точке поверхности, для которой вычисляется локальный дескриптор) цилиндрической системы координат без учёта полярного угла. Опорная точка в этом случае принимается за начало системы координат, а ось z расположена вдоль вектора нормали к поверхности в опорной точке. Теперь поставим в соответствие координатам ρ и z цилиндрической системы координат две относительные координаты спинового изображения α и β, как показано на рис.1.2. Так как полярный угол φ цилиндрической системы координат в данном случае не учитывается, спиновое изображение получается инвариантно относительно преобразования вращения.

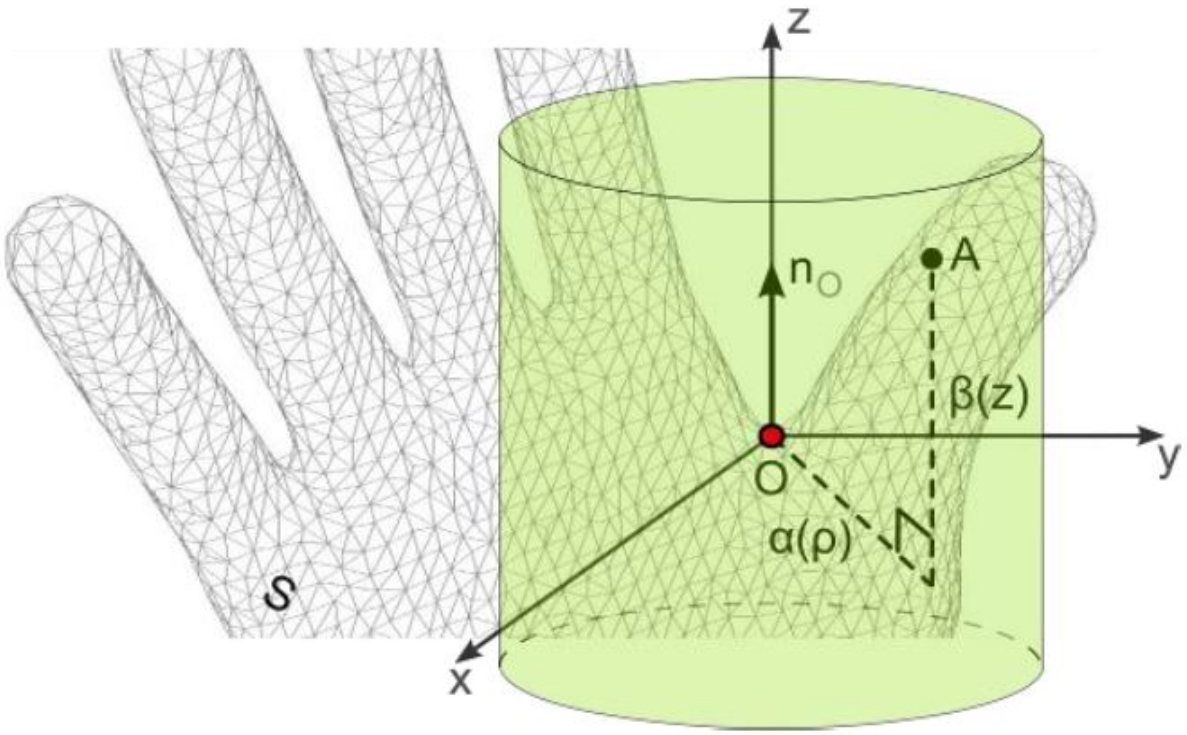

Рис. 1.2: Относительные координаты спинового изображения α и β

Отсюда можно сделать вывод, что точки с одинаковыми значениями α и β лежат на окружности радиуса α и на расстоянии β от точки O вдоль её нормали. На практике точек с абсолютно одинаковыми значениями α и β, как правило, не существует. Поэтому значения α и β разбивают на классы (кор-

зины) и подсчитывают количество точек поверхности, попавших в каждый класс.

Таким образом, для вычисления спинового изображения используются точки поверхности S, ограниченные цилиндром с центром в точке O, высотой $\beta_{max} = W_\beta b_\beta$ и радиусом $\alpha_{max} = W_\alpha b_\alpha$. В случае спиновых изображений корзины имеют форму трёхмерных колец как показано на рис.1.3.

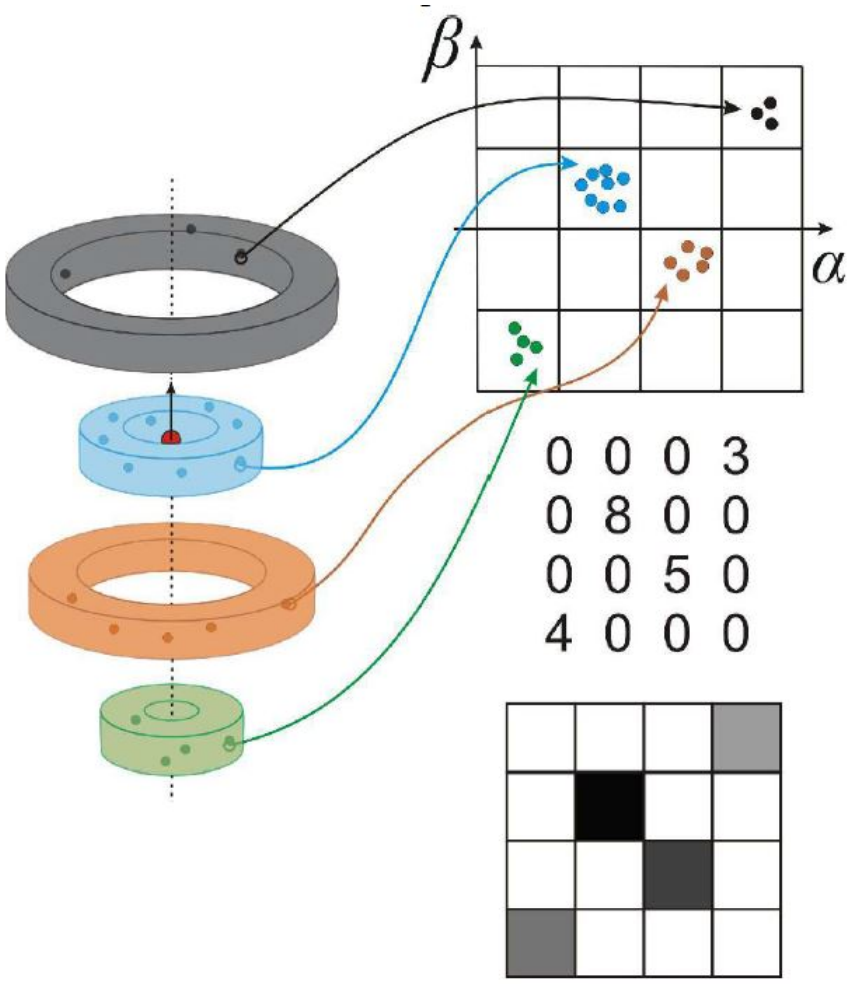

Рис. 1.3: Корзины спинового изображения

Зависимость между значениями относительных координат спинового изображения α, β и целочисленными координатами (индексами) i, j корзин $M_O[i,j]$ приведена ниже:

$$i = \left[\frac{\frac{W_\beta}{2} - \beta}{b_\beta}\right], \qquad j = \left[\frac{\alpha}{b_\alpha}\right] \tag{1.1}$$

1.3 Интегральные спиновые изображения в качестве глобальных дескрипторов поверхности

Понятие глобального дескриптора поверхности можно определить, как вектор параметров, описывающий геометрию поверхности трёхмерной модели в компактной и информативной форме. Данное представление двух трехмерных моделей удобно использовать для их сравнения.

В данной работе, для сравнения формы моделей предлагается использовать интегральные спиновые изображения в качестве глобальных дескрипторов поверхности. Конкретнее, интегральным спиновым изображением, в нашем случае, будет являться, рассмотренный ранее, локальный дескриптор поверхности - спиновое изображение, вычисленный для всей модели сразу. Таким образом, понятие интегрального спинового изображения является расширением понятия обычных спиновых изображения, пригодным для описания формы всей поверхности модели. Выбор интегральных спиновых изображений в качестве глобальных дескрипторов поверхности обусловлен их хорошими описательными характеристиками, достаточной простотой вычисления, а также устойчивостью к наличию шумов.

Очевидно, что для вычисления обычного спинового изображения необходимо знание координат опорной точки и вектора нормали к поверхности модели в этой точке. При этом опорной точкой принимается одна из точек модели. При построении же глобального дескриптора интегрального спинового изображения выбор опорной точки и вектора нормали представляет собой более сложную задачу. Это обусловлено, например, тем, что глобальные дескрипторы поверхности, вычисленные для идентичных моделей, должны быть совершенно одинаковыми даже если они по-разному ориентированы в пространстве.

В качестве опорной точки предлагается выбрать, так называемый, центр модели. Определим его как среднее арифметическое координат точек модели. Также для нахождения опорного вектора, необходимо, чтобы центр модели был расположен в начале системы координат. Для этого нужно преобразовать модель таким образом, чтобы среднее арифметическое точек модели стало равным нулю. Данное преобразование достигается смещением всех точек мо-

дели в соответствии с вектором трансляции t, который находится по формуле:

$$t = \frac{\sum_{i=1}^{n} p_i}{n}, \tag{1.2}$$

где $p_i \in \mathbb{R}^3$ - координаты точек трехмерной модели, n - количество точек трехмерной модели.

Для выбора вектора нормали предлагается использовать метод уменьшения размерности системы – метод главных компонент PCA описанный в работе [5].

Точки, формирующие дескриптор, ограничены некоторой окрестностью опорной точки спинового изображения. В случае интегральных спиновых изображений, все точки трёхмерной модели должны быть учтены при построении глобального дескриптора. Естественно, что параметры генерации (размер корзин и ширина) интегрального спинового изображения должны оставаться постоянными для всех моделей. Проблема попадания всех точек трёхмерной модели в окрестность опорной точки решается путём оптимального масштабирования трёхмерных моделей и дальнейшей нормализации (масштабировании) интегральных спиновых изображений. Ещё одна проблема, приводящая к ошибкам сравнения трёхмерных моделей, связана с неравномерным распределением точек модели по поверхности (то есть в одной части поверхности модели точки могут быть расположены тесно, а в другой они могут быть сильно разрежены) рис.1.4. Другими словами, разрешение трёхмерной модели (средняя длина рёбер, соединяющих точки модели) может значительно различаться на разных участках поверхности модели. Проблема разрешения трехмерной модели играет очень важную роль в процессе построения дескрипторов и их сравнения. Подробнее эта тема обсуждается в следующем разделе.

Интегральное спиновое изображение вычисляется аналогично схеме для обычного спинового изображения, за исключением того, что используемые специфические опорная точка и вектор, а также окрестность опорной точки выбираются таким образом, чтобы абсолютно все точки модели попали в неё. За счёт вычисления относительных координат α и β для всех точек мо-

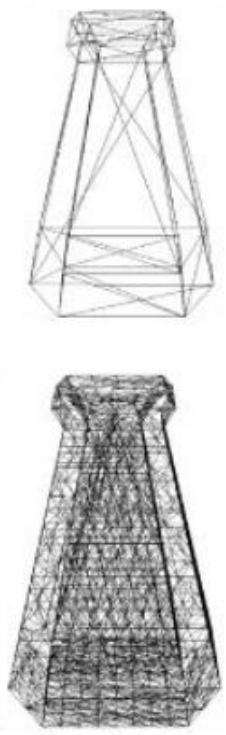

Рис. 1.4: Верхнее изображение – трёхмерная модель с неравномерным распределением точек; нижнее изображение – та же модель с унифицированным разрешением

дели интегральное спиновое изображение описывает форму всей поверхности модели. Преимуществами использования идеи спиновых изображений при построении глобальных дескрипторов поверхности являются их хорошие описательные характеристики, относительно небольшие вычислительные затраты на их построение и обработку, а также наличие эффективного и простого метода их сравнения.

2 Алгоритм контроля разрешения трехмерных объектов

2.1 Определения

Как уже было сказано ранее, неравномерное распределение точек модели по поверхности значительно влияет на вид интегрального спинового изображения и, как следствие, приводит к ошибкам сравнения дескрипторов. Более того, модели, состоящие из малого количества точек, не позволяют построить спинового изображения, которое бы явно выделяло его признаки. На рис.2.1 представлена одна и та же модель, но с разными средними длинами ребер, а также глобальные дескрипторы для каждого случая. Совершенно очевидно, что никаких отличительных признаков для модели с малым количеством точек и длинными ребрами выявить невозможно.

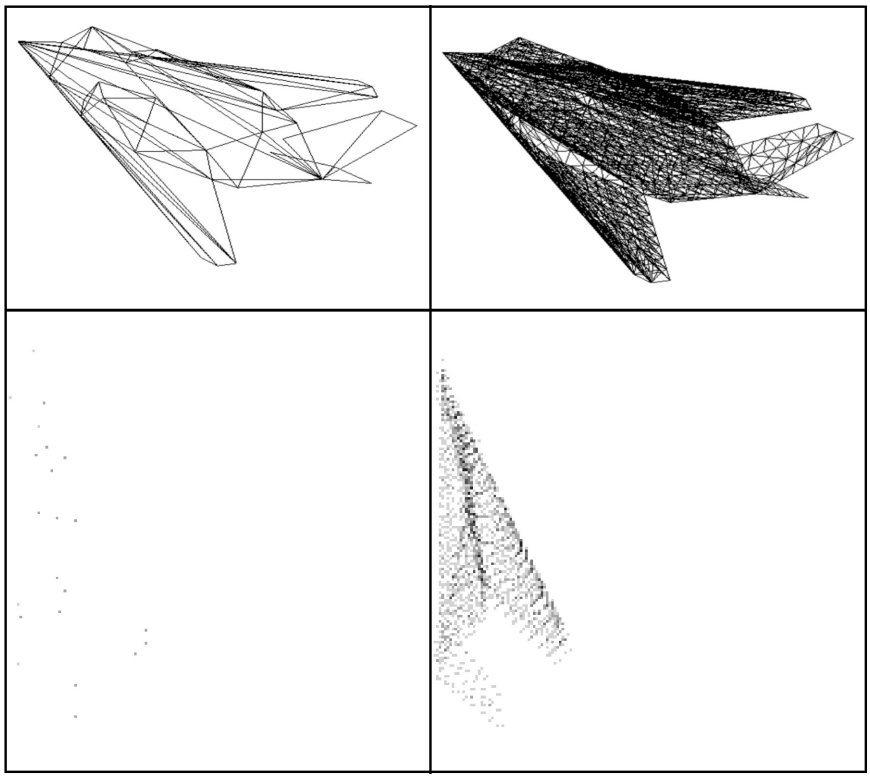

Рис. 2.1: Спиновые изображения одной и той же модели с разной средней длинной ребер

Решение данной проблемы было представлено в работе [6]. Далее пред-

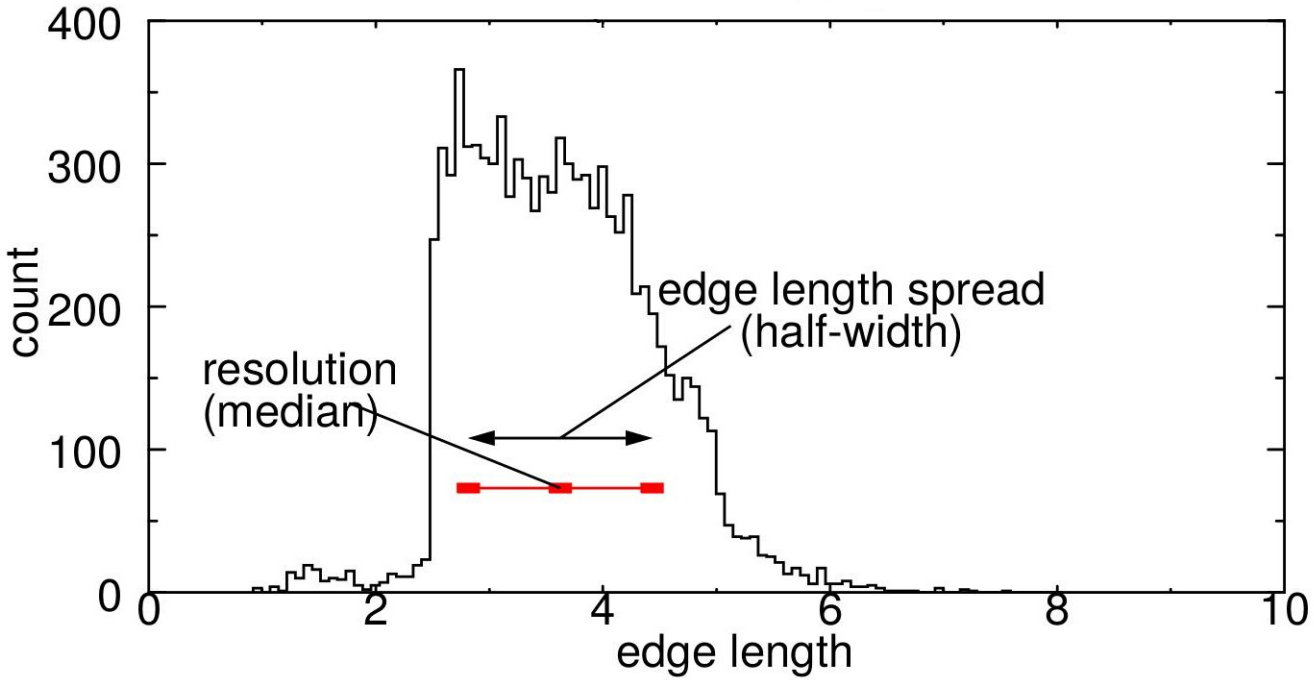

Рис. 2.2: Гистограмма отражающая количество ребер определенной длины в трехмерном объекте

лагается в деталях рассмотреть описание и реализацию алгоритма контроля разрешения трехмерных полигональных моделей.

Для произвольных трехмерных объектов невозможно иметь одинаковое расстояние между всеми вершинами и при этом адекватно описать их форму. Таким образом, длины ребер могут быть измерены с использованием локальных и глобальных статистических данных. Наглядно эти данные удобно рассматривать с помощью гистограммы. Пример такой гистограммы представлен на рис.2.2.

На данной гистограмме по горизонтальной оси представлены длины ребер, а по вертикальной количество ребер определенной длины в объекте. Также по гистограмме можно оценить разрешение объекта и его длину распространения ребра. Определим разрешение объекта как медиану длин всех его ребер, а длину распространения ребра как верхнюю квартиль длин ребер минус нижнюю квартиль длин ребер (то есть, половину ширины). Учитывая эти определения, цель алгоритма - привести исходное разрешение объекта к желаемому путем минимизации длины распространения ребра в гистограмме.

Назовем этот процесс нормализацией длины.

Дополнительным ограничением на нормализацию длины ребра является то, что первоначальная форма объекта должна быть сохранена. Предполагается, что первоначальная форма объекта определяется сетью точек и ребер поданной на вход алгоритма. Для получения нужного разрешения, к ребрам модели применяются две операции:

1. edge-split - убрать длинное ребро;

2. edge-collapse - убрать короткое ребро.

Реализация методов, соответствующих данным операциям приведена в Приложении А.

Во время edge-split ребро разделяют в его средней точке на два ребра. Эта операция не меняет форму объекта. Во время edge-collapse ребро превращается в точку, по этому локально форма поверхности изменяется. В рассматриваемом алгоритме положение результирующей точки после edge-collapse выбирают так, чтобы сохранить форму объекта, но некоторые изменения формы неизбежны, когда ребра приходится удалять. Тем не менее, изменения формы поверхности модели могут быть сведены к минимуму за счет разумного порядка обработки ребер. Более конкретно, ребра упорядочены для работы путем измерения изменения формы объекта, которое является результатом применения операции к каждому ребру. Используя этот порядок, сначала обрабатываются ребра, которые меньше всего изменяют поверхность, тем самым сводя к минимуму изменение формы объекта на каждой итерации. Этот подход позволяет получить объекты, в которых форма поверхности менее отличается от первоначальной при нормализации длины, чем алгоритм, который выбирает ребра случайным образом.

Предотвратить слишком сильное изменение поверхности модели возможно, путем размещения ограничения на максимально допустимое изменение в формы объекта.

Если рассматриваемое ребро укладывается в установленные границы его длины, оно не будет уничтожено. Ограничение на максимальное изменение

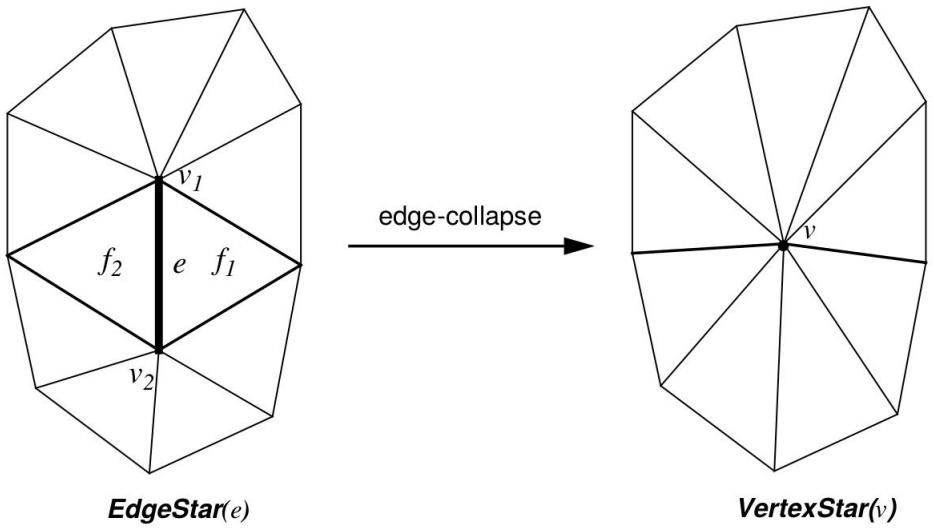

Рис. 2.3: Влияние edge-collapse на его ближайших соседей: ребро e свернуто в вершину v, при этом удалены ребро e, грани f_1 и f_2, и вершины v_1 и v_2

формы и нахождение длины ребра в некотором допустимом интервале исключает удаление всех ребер. Это является критерием остановки алгоритма. Так как алгоритм направлен на нормализацию длины ребер, сохраняя при этом форму объекта, очевидно, что большинство, но не все, из длин ребер будут находится в установленном допустимом интервале.

Для корректного описания алгоритма в деталях стоит определить понятия ближайших соседей ребер и вершин поверхности объекта. Названия данных сущностей, приведенные в данной работе, соответствуют названиям методов их поиска в реализованном алгоритме.

Пусть EdgeStar(e) соответствует ближайшим соседям ребра e, когда к нему применяется операция edge-collapse. EdgeStar(e) содержит все грани, в которых есть как минимум одна из вершин ребра e вместе с ребрами и вершинами, которые составляют данные грани. Определим VertexStar(v) как ближайших соседей вершины v которые созданы после применения к ребру операции edge-collapse. VertexStar(v) содержит все грани, содержащие вершину v вместе с составляющими ее ребрами и вершинами. Иллюстрации EdgeStar(e) и VertexStar(v) приведены на рис.2.3.

Пусть EdgeDiamond(e) соответствует ближайшим соседям ребра e перед применением к нему операции edge-split. EdgeDiamond(e) содержит грани,

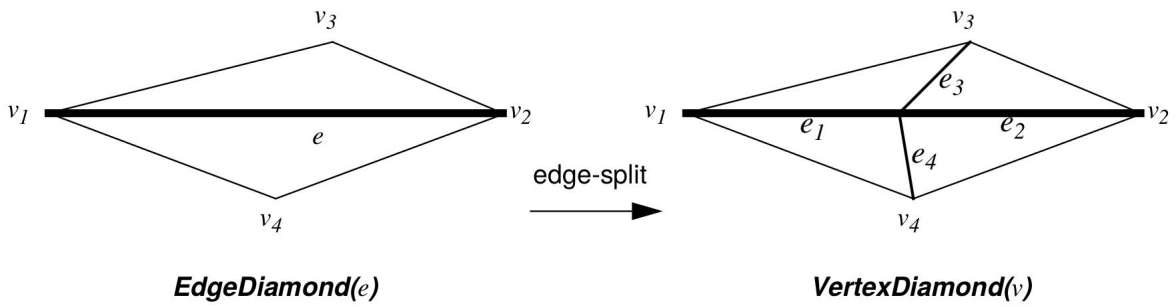

Рис. 2.4: Влияние edge-split на его ближайших соседей: ребро e разбито вершиной v, добавлены три новых ребра две новые грани; ближайшие соседи e перед edge-split - EdgeDiamond(e), а ближайшие соседи вершины v после edge-split - VertexDiamond(v)

примыкающие к ребру . Ребра и вершины, составляющие эти грани также включены в EdgeDiamond(e). Также определим VertexDiamond(v) для вершины v как ее ближайших соседей после применения к ней операции edge-split. VertexDiamond(v) содержит четыре грани которые составлены из вершины v вместе с ребрами и вершинами которые включены в данные грани. Иллюстрации EdgeDiamond(e) и VertexDiamond(v) приведены на рис.2.4.

2.2 Обзор алгоритма

Входом в алгоритм служит желаемое разрешение L_0 и допустимое отклонение длины ребер L_D от L_0 в нормализованном объекте. Тогда верхняя и нижняя границы длин ребер:

$$L_{MIN} = L_0 - \frac{L_D}{2} \qquad L_{MAX} = L_0 + \frac{L_D}{2} \tag{2.1}$$

Также в алгоритм на вход передается максимально возможное изменение формы поверхности C_{MAX}.

В деталях, алгоритм выглядит следующим образом: сначала создается динамическая очередь из ребер объекта. Положение ребра в очереди определено его длиной и мерой изменения формы для данного ребра. Таким образом ребра оказываются упорядочены в соответствии с их длиной по убыванию, а ребра с одинаковой длиной расположены таким образом, чтобы сначала обра-

батывались те из них, для которых мера изменения формы поверхности будет меньшей. За тем первое ребро в очереди выталкивается из нее и обрабатывается. Если длина ребра больше L_{MAX}, ребро разбивается в его средней точке. Разбиение меняет ближайших соседей ребра добавлением новых ребер, граней и вершины. Если длина ребро меньше чем L_{MIN}, оно сворачивается в точку, изменяя своих ближайших соседей путем уничтожения ребра и двух граней вместе с их ребрами. Когда ребро сворачивается, его мера изменения формы добавляется в переменные контролирующие изменение формы поверхности для новых ближайших соседей вершины. После проведения данных операций, ближайшие соседи обработанных ребер для старого случая удаляются из очереди. За тем новые ребра добавляются в очередь если они соответствуют следующим критериям:

1. длина ребер не входит в требуемый интервал для длины ребер L_{MAX} и L_{MIN};

2. накопление изменения формы (accumulated shape change) для этих ребер не больше чем C_{MAX};

Ребра выталкиваются из очереди и обрабатываются пока очередь не станет пустой.

2.3 Мера изменения формы

Мера изменения формы определяется как разница между старой и новой поверхностью до и после обработки ребра. Как уже упоминалось ранее, перед началом работы алгоритма пользователь определяет максимально возможное изменение формы поверхности C_{MAX}. Поскольку обработка ребра влияет только на его ближайших соседей, разница между старой и новой формой объекта на одной итерации может быть измерена путем сравнения только окрестности (ближайших соседей) обрабатываемого ребра до и после проведения над ним необходимых операций.

Логично предположить, что форма объекта должна определяться его гранями, поэтому точная мера изменения между старой и новой моделью должна

учитывать расстояние между гранями. Определим расстояние между поверхностями M_1 и M_2 как максимальное расстояние между любой точкой на M_1 и ее ближайшей связанной точкой на M_2. Поскольку модели представляют собой объединение подмножеств линейных элементов (точки, линии и плоскости), максимальное отличие M_1 и M_2 заключено между вершиной на M_1 и гранью из M_2. Отсюда следует что разница между M_1 и M_2 определяется как максимум Евклидова расстояния между вершиной v_i из M_1 и ближайшей к ней точкой, $v_{closest}$, на грани f_i из M_2.

$$d(M_1, M_2) = \max_{v_i \in M_1}(\min_{f_i \in M_2} ||v_i - v_{closest}(v_i, f_i)||) \tag{2.2}$$

Ближайшая точка на треугольнике к точке в пространстве определена как проекция этой точки на плоскость данного треугольника. Если спроецированная точка лежит внутри треугольника, тогда - это и есть искомая точка. В противном случае точка перпендикулярно проецируется на прямые, на которых лежат стороны треугольника. Если проекция точки расположена между двумя вершинами треугольника, тогда - это искомая точка. В противном случае, искомой точкой является одна из вершин треугольника.

Расстояние $d(M_1, M_2)$ не симметрично, поэтому определим показатель различия между двумя поверхностями $D(M_1, M_2)$ как максимум между $d(M_1, M_2)$ и $d(M_2, M_1)$:

$$D(M_1, M_2) = max(d(M_1, M_2), d(M_2, M_1)) \tag{2.3}$$

В представленном алгоритме нормализации длины ребер будем использовать $D(M_1, M_2)$ как меру изменения формы. Очевидно, что мера изменения формы будет равна нулю, когда поверхности совпадают, даже если грани, ребра и вершины не являются точно равными. Использование максимального различия между поверхностями как меры изменения формы дает алгоритму возможность обрабатывать ребра вдоль поверхностных неоднородностей модели, таких как гребни или углы, пока различие между объектами остается малым во время обработки ребер.

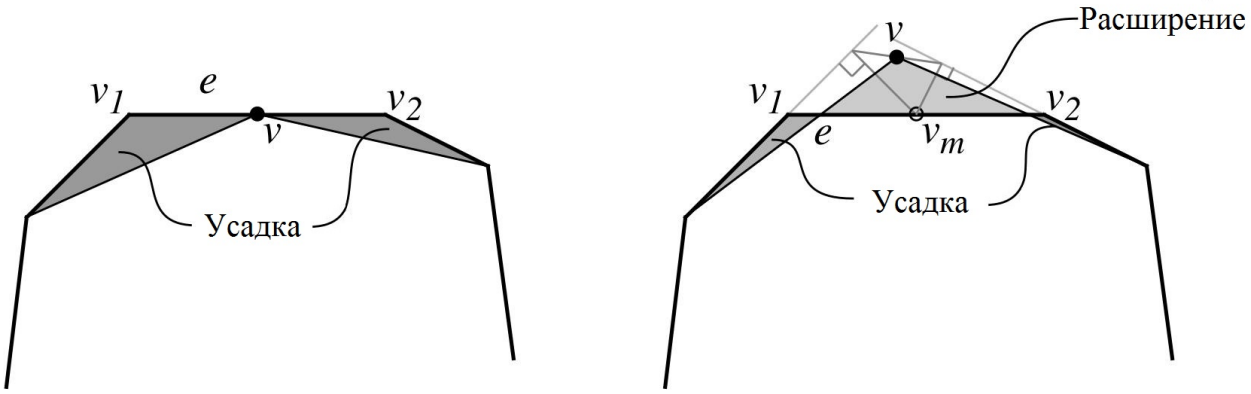

Рис. 2.5: Расположение результирующей точки после операции edge-collapse

2.4 Базовые операции алгоритма

В данном подразделе подробно рассматриваются операции edge-collapse и edge-star, и некоторые особенности их реализации.

Первой рассмотрим операцию edge-collapse. Как показано на рис.2.3 цель данной процедуры - сжать ребро в точку посредством этого удаляя ребро и две смежные с ним грани. Операция edge-collapse может быть представлена через локальные операции, которые удаляют задействованные ребра и вершины с поверхности модели в наборе данных, а за тем добавляют новые ребра в очередь.

Важной переменной является позиция новой вершины, которая является результатом edsge-collapse. Позиции остальных вершин в EdgeStar(e) остаются неизменными в течении edge-collapse. Простым решением является поместить результирующую точку в центр сворачиваемого ребра. Однако, как показано на рис.2.5 (слева), размещение новой вершины на поверхности сети объекта может вызвать чрезмерное изменение формы (усадку или расширение) в районах с высокой кривизной поверхности.

Таким образом в большинстве случаев стоит поместить вершину, которая получена после edge-collapse вне прямой, на которой лежит удаляемое ребро. В частности, за позицию новой вершины v следует принять среднее проекций середины сворачиваемого ребра v_m на N плоскостей которым принадлежат грани из EdgeStar(e).

$$v = v_m - \frac{1}{N}\sum_{i=1}^{N}(n_i v_m + d_i)n_i \qquad v_m = \frac{v_1 + v_2}{2} \tag{2.4}$$

Грани из EdgeStar(e) в этом случае определяются нормалями плоскостей на которых лежат эти грани n_i и их смещением d_i относительно центра сворачиваемого ребра v_m. На рис.2.5 (справа) представлен пример расположения новой вершины после применения операции edge-collapse на основе проецирования середины сворачиваемого ребра на соседние плоскости. Таким образом усадка сети объекта предотвращается путем распределения изменения формы поверхности выше и ниже уничтоженного ребра.

Мера изменения формы для ребра, которое мы собираемся свернуть в точку может быть вычислена, используя EdgeStar(e) и VertexStar(v). После edge-collapse вершины вдоль границы VertexStar(v) те же что и вершины на границе EdgeStar(e). Таким образом, в соответствии с рис.2.3, мера изменения формы может быть вычислена как максимум расстояния между v и ее ближайшей точкой на гранях EdgeStar(e), v_1 и ее ближайшей точкой на гранях VertexStar(v), или v_2 и ее ближайшей точкой на гранях VertexStar(v).

Теперь рассмотрим операцию edge-split. Она используется чтобы убрать слишком длинные ребра. Как показано на рис.2.4, edge-split делит ребро вершиной и производит новые ребра, две новые грани и новую вершину. Позиция новой вершины выбирается в середине разделяемого ребра. Так как поверхность до и после edge-split не изменяется, значит мера изменения формы для этой операции всегда равна нулю и не рассматривается в процессе работы алгоритма. Операция edge-split может быть реализована через локальные изменения поверхности, которые добавляют новые ребра и вершину в очередь и за удаляют разделяемое ребро из очереди.

2.5 Накопление изменения формы

Определим накопление изменения формы как некоторое значение, которое соответствует изменению формы всей модели в процессе нормализации длины ее ребер. Как было рассмотрено ранее, каждый раз после удаления ребра, форма объекта немного меняется. Мера изменения формы в нашем случае -

это максимальное расстояние между поверхностями до и после сворачивания ребра для его ближайших соседей. Первоначально у каждого ребра нулевая мера изменения формы, а у модели нулевое накопление изменения формы. Когда ребро e свернуто, его мера изменения формы вычисляется и добавляется накопление изменения формы. Отслеживая худший случай изменения формы для каждого ребра, мы можем ограничить глобальный максимум изменения формы объекта. Другими словами, ребра, при сворачивании которых накопление изменения формы может превысить максимально возможное изменение формы поверхности C_{MAX}, следует удалить из очереди, несмотря на то, что они могут не входить в целевой интервал для длины ребер.

3 Сверточные нейронные сети

3.1 Общие представления

Компьютерные методы, имитирующие человеческое зрение, сегодня используются для решения многих задач – от определения лиц на Facebook и автопилотируемых Google-мобилей до суперсовременных алгоритмов диагностики заболеваний. Однако на поверку имитация работы человеческих органов зрения оказывается задачей не из легких. Там, где мы автоматически распознаем линии, контуры и объекты, компьютер видит всего лишь огромные числовые матрицы.

Для решения задачи определения более сложных признаков из сырых массивов пикселов предлагается использовать отдельный тип нейронных сетей, называемых сверточными.

Свёрточная нейронная сеть (convolutional neural network, CNN, LeNet) была представлена в 1998 году французским исследователем Яном Лекуном (Yann LeCun), как развитие модели неокогнитрон [7]. Архитектура данного типа нейронных сетей схожа с некоторыми особенностями зрительной коры, в которой имеются простые клетки, реагирующие на прямые линии под разными углами, и сложные клетки, реакция которых связана с активацией определённого набора простых клеток.

Таким образом, идея свёрточных нейронных сетей заключается в чередовании свёрточных слоев (англ. convolution layers) и субдискретизирующих слоев (англ. subsampling/pooling layers). Для обучения используются градиентные методы, чаще всего метод обратного распространения ошибки.

Название архитектура сети получила из-за наличия операции свёртки , суть которой в том, что каждый фрагмент изображения умножается на матрицу (ядро) свёртки поэлементно, а результат суммируется и записывается в аналогичную позицию выходного изображения.

$$(f * g)[m, n] = \sum_{k,l} f[m - k, n - l] \cdot g[k, l] \tag{3.1}$$

Здесь f - исходная матрица изображения, g - ядро (матрица) свёртки.

Теперь выделим некоторые преимущества и недостатки, данного типа нейронных сетей.

Преимущества:

1. По сравнению с полносвязной нейронной сетью — в сверточной нейронной сети требуется гораздо меньшее количество настраиваемых весов, так как одно ядро весов используется целиком для всего изображения, вместо того, чтобы делать для каждого пикселя входного изображения свои персональные весовые коэффициенты. Это подталкивает нейросеть при обучении к обобщению демонстрируемой информации, а не попиксельному запоминанию каждой показанной картинки в мириадах весовых коэффициентов, как это делает перцептрон.

2. Удобное распараллеливание вычислений, а, следовательно, возможность реализации алгоритмов работы и обучения сети на графических процессорах.

3. Относительная устойчивость к повороту и сдвигу распознаваемого изображения.

4. Обучение при помощи классического метода обратного распространения ошибки.

Недостатки:

1. Архитектура свёрточной нейронной сети по большей части пригодна только для распознавания изображений.

2. Слишком много варьируемых параметров сети. Существует несколько выверенных и прекрасно работающих конфигураций сетей, но нет правил, по которым нужно делать выбор для новой задачи.

3.2 Архитектура сверточной нейронной сети

Модель свёрточной сети рис.3.1, состоит из трёх типов слоёв: свёрточные (convolutional) слои, субдискретизирующие (subsampling/pooling) слои и слои "обычной"полносвязной нейронной сети (fully-connected layer).

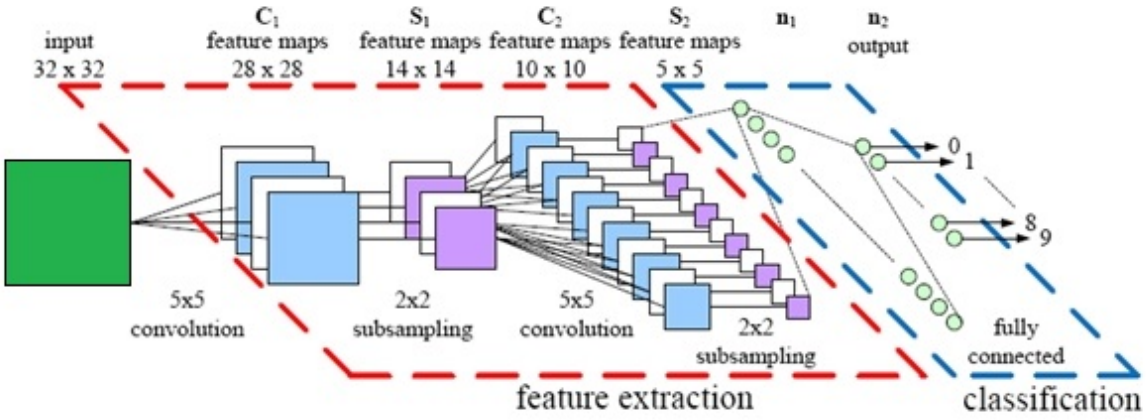

Рис. 3.1: Общая схема свёрточной нейронной сети

Первые два типа слоёв (convolutional, subsampling/pooling), чередуясь между собой, формируют входной вектор признаков для многослойного перцептрона (fully connected MLP).

Далее рассмотрим подробнее каждый из типов слоев сверточной нейронной сети.

3.2.1 Свёрточный слой

Свёрточный слой (англ. Convolutional layer) является основным строительным блоком сверточной нейронной сети. Параметры слоя состоят из набора фильтров для обучения (или ядер свертки), которые имеют небольшое рецептивное поле, но простираются на всю глубину входной матрицы. В течение прямого прохода каждый фильтр осуществляет свертку по ширине и высоте матрицы на входе, вычисляя скалярное произведение данных фильтра и входа, и формируя двумерную карту активации этого фильтра.

Размерность матрицы на выходе сверточного слоя контролируют три гиперпараметра: глубина(depth), шаг(stride) и нулевое дополнение(zero-padding).

1. Глубина контролирует количество нейронов слоя, которые соединяются с одной и той же областью изображения на входе. Другими словами, это количество ядер свертки в данном слое.

2. Шаг контроллирует перемещение ядра свертки по входной матрице. Например, если шаг равен 1, то ядро свертки каждый раз перемещается на один пиксель.

3. Иногда удобно дополнять вход нулями по краям входной матрицы. Размер этого нулевого дополнения является третьим гиперпараметром. Нулевое дополнение позволяет контролировать пространственный размер выходных матриц.

Пространственный размер выходной матрицы W_2 может исчисляться как функция от размера входной матрицы W_1, размера рецептивного поля нейронов сверточного слоя F, шага, с которым они применяются S, и величины нулевого дополнения P, применяемой на краях [8].

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \tag{3.2}$$

Если это число не является целым, то шаги установлено неправильно, и нейроны не могут быть размещены вдоль входной матрицы симметричным образом. В общем случае, установление нулевого дополнения в $P = (F-1)/2$, когда шагом является $S = 1$, обеспечивает, чтобы входной и выходной объемы имели одинаковый пространственный размер.

Следует также отметить, что глубина выходной матрицы равна количеству примененных ядер свертки(фильтров).

3.2.2 Субдискретизирующий слой

При реализации архитектуры сверточных нейронных сетей, между последовательно идущими сверточными слоями обычно помещают субдискретизирующие слои. Слои этого типа выполняют уменьшение размера входной карты признаков. Это можно делать разными способами, в данном случае рассматривается метод выбора максимального элемента (max-pooling) - вся карта признаков разделяется на ячейки $[2 \times 2]$ элемента, из которых выбираются максимальные по значению рис.3.2.

Субдискретизирующий слой формирует размер выходной матрицы W_2 как функцию от размера входной матрицы W_1, рецептивного поля F и шага S

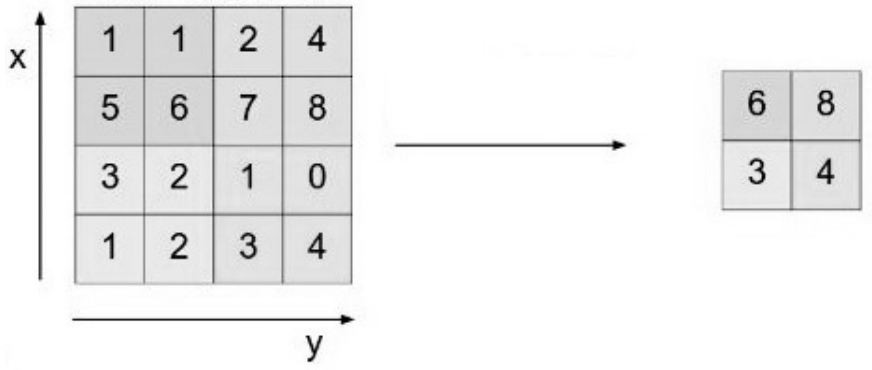

Рис. 3.2: Пример выбора максимальных элементов из матрицы 4x4 при действии на нее pooling слоем размерности 2x2

субдискретизирующего слоя[8]:

$$W_2 = \frac{W_1 - F}{S} + 1 \tag{3.3}$$

В данном случае глубина выходной матрицы соответствует глубине входной матрицы.

3.2.3 Слой полносвязной нейронной сети

Нейроны в полносвязной нейронной сети соединены со всеми картами активаций в предыдущем слое, как в обычных нейронных сетях. На выходе эта нейронная сеть выдает вектор признаков в котором каждый элемент соответствует одному из распознаваемых классов. Чем больше значение одного из элементов этого вектора при работе сети, тем больше, проверяемое изображение соответствует тому или иному классу.

3.3 Обучение сверточных нейронных сетей

В данном подразделе рассматривается общий принцип обучения сверточной нейронной сети. Первоначально значения весов сверточного слоя инициализированы случайным образом. Корректировку их значений в данной работе предлагается провести с помощью метода обратного распространения ошибки[9]. Данный метод предполагает корректировать веса с помощью обучающего набора изображений у каждого из которых есть ярлык, говорящий о принадлежности данного изображения определенному классу.

Метод обратного распространения ошибки можно разделить на четыре этапа: прямое распространение, функцию потери, обратное распространение и обновление веса. Во время прямого распространения, тренировочное изображение пропускается через всю сеть и регистрируется полученный результат. Сравнение полученного и желаемого результатов - есть качество распознавания. Для получения возможности такого сравнения определим функцию потери обучающей пары. В данной работе предлагается использовать функцию среднеквадратической ошибки:

$$L(W) = (\sum_{p=1}^{N} \frac{1}{2}(D_p - O(I_p, W))^2)/N \tag{3.4}$$

,где $L(W)$ — это функция ошибки для всей обучающей выборки, p - номер обучающей пары, N - количество обучающих пар, D_p — желаемый выход сети, $O(I_p, W)$ — выход сети, зависящий от p-го входа и весовых коэффициентов W. Задача обучения так настроить веса W, чтобы они для любой обучающей пары (I_p, D_p) давали минимальную ошибку L. Для этого сначала нужно выяснить, какие веса самым непосредственным образом способствовали ошибкам сети, а затем скорректировать их так, чтобы снизить потерю.

Значение производной функции ошибки по одному из весовых коэффициентов($\frac{dL}{dW}$ определяет влияние этого веса на ошибки сети. При обратном распространении данная производная вычисляется для весов каждого слоя после чего происходит обновление весов на рассматриваемом слое. Формула по которой происходит корректировка каждого фильтрового веса приведена ниже:

$$w = w_i - \eta \frac{dL(W_i)}{dW} \tag{3.5}$$

,где w_i - начальное значение регулируемого веса, $L(W_i)$ - функция ошибки, η - скорость обучения.

В данной формуле параметр, который отвечает за скорость обучения, выбирается программистом. По своей сути он определяет шаг, с которым будет корректироваться вес. Он должен быть выбран таким образом, чтобы, с одной стороны, корректировка фильтровых весов была достаточно точной без слишком больших скачков, а с другой - позволила бы достичь минимума функции ошибок к окончанию обучения.

Процесс прямого распространения, функцию потери, обратное распространение и обновление весов обычно называют одним периодом дискретизации (или epoch — эпохой). Программа будет повторять этот процесс фиксированное количество периодов для каждого тренировочного изображения. После того, как обновление параметров завершится на последнем тренировочном образце, сеть в теории должна быть достаточно хорошо обучена и веса слоёв настроены правильно.

4 Реализация системы классификации трехмерных моделей

В данном разделе представлена реализация системы классификации трехмерных объектов и результаты ее тестирования.

4.1 Подготовка данных для обучения сверточной нейронной сети

Для реализации подобной системы было решено использовать модели, представленные в базе трехмерных моделей университета Princeton, а именно 180 объектов трех различных классов: "Самолет"(airplane aircraft), "Мебель"(furniture) и "Человек"(human). Для контроля разрешения трехмерных моделей, алгоритм, представленный в главе 2, был реализован в виде программы на языке C. При подборе оптимального разрешения моделей стоит отметить, что чрезмерное уменьшение разрешения не только заметно увеличивает время формирования базы данных, но и не ведет к каким-либо преимуществам при обучении сверточной нейронной сети. Это продиктовано тем, что после формирования глобальных дескрипторов в виде интегральных спиновых изображений, значения в "корзинах"дескриптора должны находиться в некотором интервале в зависимости от архитектуры сверточной нейронной сети.

После подбора оптимального разрешения моделей и формирования окончательной базы данных трехмерных объектов, для каждого из них были получены глобальные дескрипторы в виде интегральных спиновых изображений методом, описанным в работе [5]. В результате тестирования систем классификации было установлено, что размер спинового изображения должен быть равен размеру изображений, передаваемых на вход сверточной нейронной сети, а размер корзин должен обеспечивать тот факт, что каждая из корзин будет включена в спиновое изображение (не выйдет за его границы). Также очевидно, что слишком большой размер корзин не обеспечит отражения отличительных признаков каждого класса на дескрипторе. Примеры дескрипторов, использованных для обучения, вы можете видеть на рис.4.1.

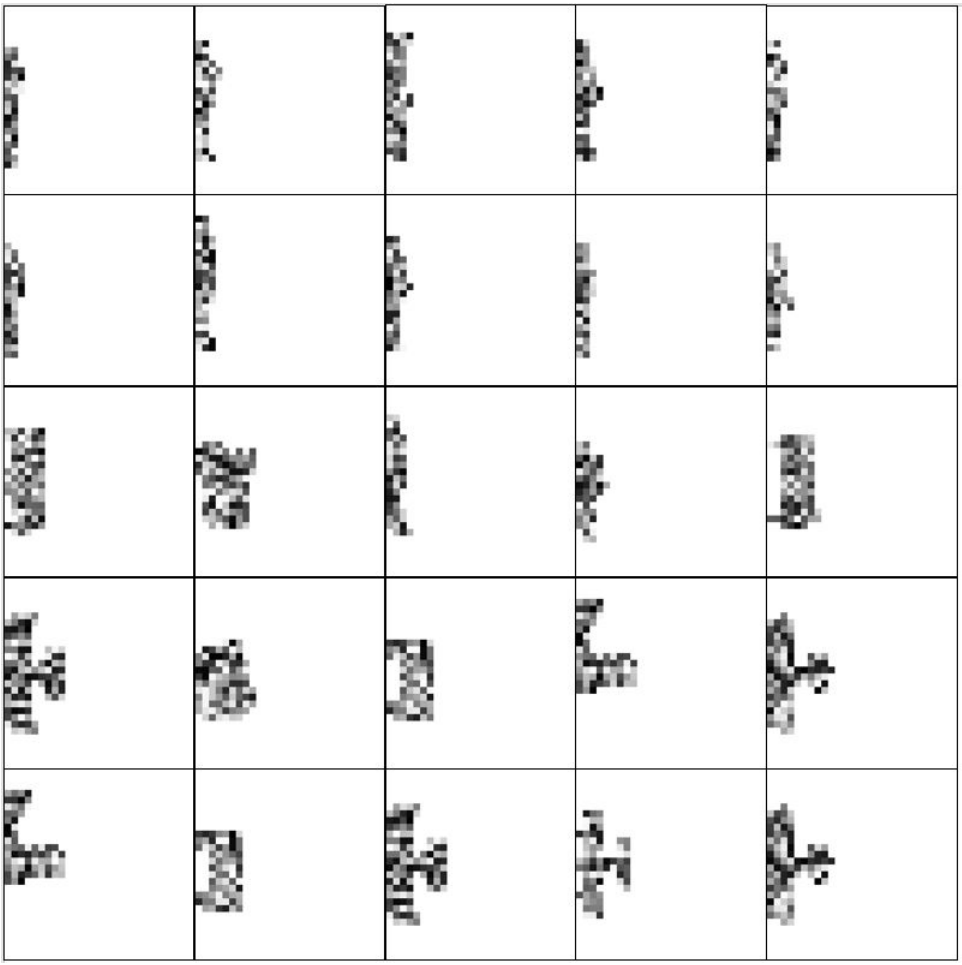

Рис. 4.1: Примеры глобальных дескрипторов, использованных для обучения сверточных нейронных сетей

После формирования набора дескрипторов для всех моделей базы данных, происходит этап обучения сверточной нейронной сети. Далее предлагается сравнить результаты обучения сверточных нейронных сетей для двух различных архитектур. В первом случае, за основу архитектуры была принята модель сети, предназначенная для распознавания цветных изображений размерности $[32 \times 32 \times 3]$ - CIFAR-10. Во втором случае сеть строилась на основе архитектуры, предназначенной для классификации рукописных символов из базы изображений MNIST. Реализация данных сверточных нейронных сетей осуществлялась при помощи пакета MatConvNet для Matlab.

4.2 Реализация архитектуры на основе сети для цветных изображений

Рассмотрим подробнее систему классификации трехмерных объектов в случае, когда за основу архитектуры сверточной нейронной сети была принята сеть, реализующая распознавание десяти классов изображений из базы CIFAR-10. На рис.4.2 представлена архитектура данной нейронной сети, адаптированная для классификации трех классов дескрипторов.

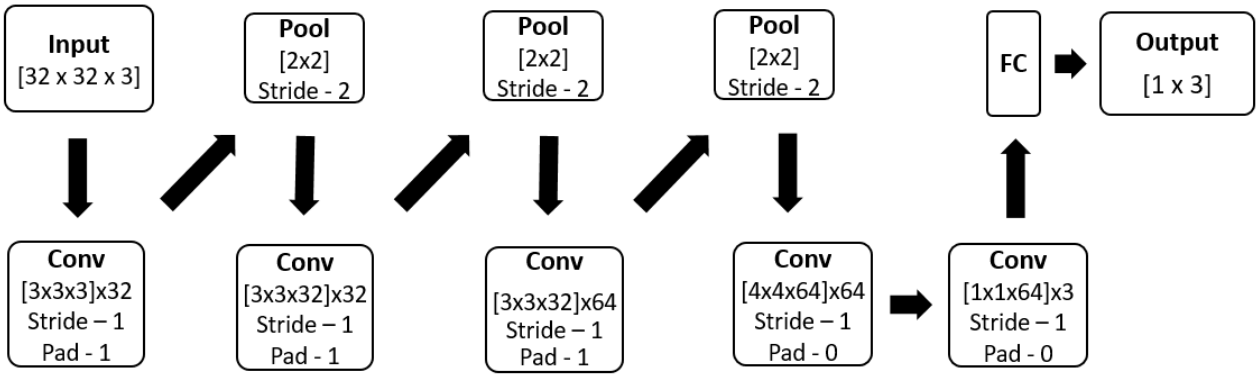

Рис. 4.2: Архитектура сверточной нейронной сети, принимающей на вход объекты размерности $[32 \times 32 \times 3]$

Прежде всего, стоит отметить, что так как данная нейронная сеть предназначена для распознавания цветных изображений, на вход ей должны передаваться объекты соответствующей размерности $[32 \times 32 \times 3]$. Глобальные дескрипторы поверхности в случае рассматриваемой системы классификации имеют вид двумерных матриц. При подготовке данных значения дескрипторов передаваемые на вход сверточной нейронной сети дублировались для всех трех измерений. Таким образом имитировались цветные изображения из базы данных CIFAR-10. В перспективе возможно использование глобальных дескрипторов поверхности в виде интегральных спиновых изображений с учетом полярного угла как описано в работе [5]. Такие спиновые изображения имеют вид трехмерных массивов данных, что делает возможным более корректное использование архитектур сверточных нейронных сетей данного типа.

На рис.4.2 можно видеть, что данная архитектура использует пять сверточных слоев и три слоя объединения. В первых двух сверточных слоях(Conv) использовано 32 ядра свертки, на втором и третьем - 64 и на последнем три. Основные гиперпараметры - такие как шаг(Stride) и отступ(Pad) выбирались исключительно методом подбора в зависимости от результатов обучения. На последнем слое сверточной нейронной сети расположена полносвязная нейронная сеть(FC) результатом работы которой является вектор признаков в котором каждый элемент соответствует одному из распознаваемых классов. В Приложении Б представлен программный код реализации данной архитектуры.

Теперь рассмотрим результат обучения данной нейронной сети. На рис.4.3 представлен график изменения значений функции ошибки для данной реализации системы. По оси абсцисс данного графика отражены номера эпох обучения, а по оси ординат - значение среднеквадратичной ошибки для каждой эпохи.

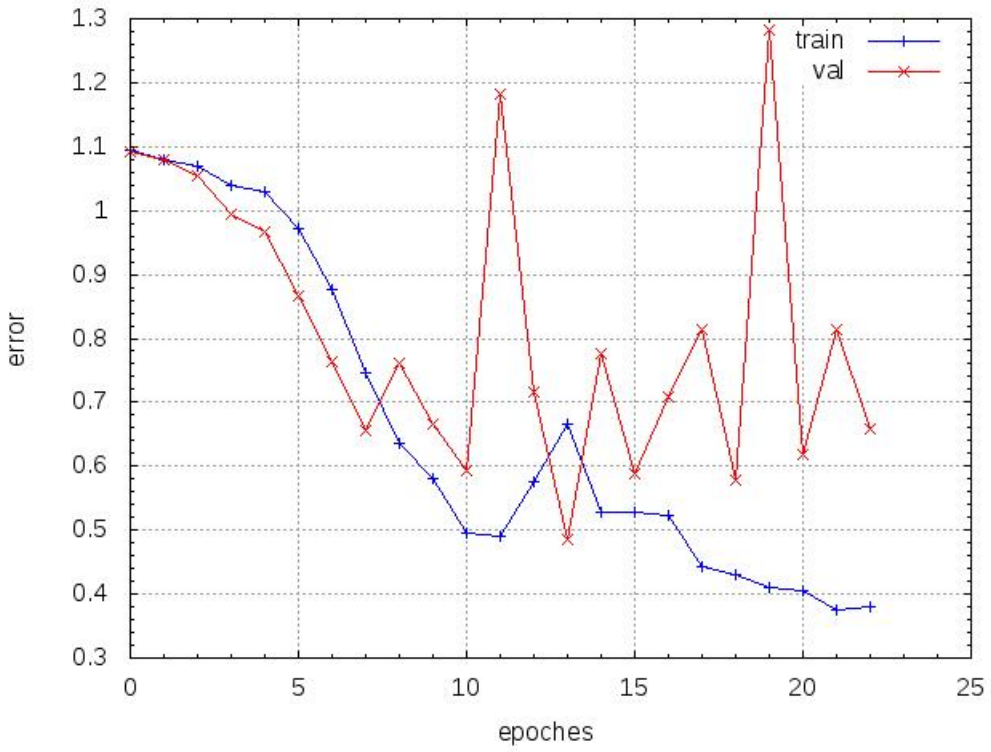

Рис. 4.3: График изменения значений функции ошибки сверточной нейронной сети, принимающей на вход объекты размерности $[32 \times 32 \times 3]$

Во время обучения сети, обучающая выборка делится на тренировочную(train) и проверочную(validation). После обновления фильтровых коэффициентов для тренировочной выборки, сеть тестируется проверочной выборкой чтобы оценить реальный результат обучения. На графике видно, что значение ошибки для проверочной выборки после некоторого момента перестает снижаться. Это говорит о том, что после этого сеть не продолжает реальный процесс обучения, а лишь затачивается под распознавание тренировочных изображений.

4.3 Реализация архитектуры на основе сети для рукописных символов

В подразделе 4.2 видно, что представленная архитектура не показала хороших результатов, однако сам факт снижения ошибки говорит о том, что сеть способна выделять признаки у глобальных дескрипторов поверхности.

После некоторого количества экспериментов было решено упростить архитектуру сети и искусственно расширить обучающий набор данных в четыре раза путем поворота изображений на 180 градусов и их инвертирования.

За основу архитектуры сверточной нейронной сети была принята модель сети, предназначенная для распознавания рукописных символов из базы изображений MNIST. Размер изображений, передаваемых на вход сети, составляет 28×28 пикселей. На рис.4.4 вы можете видеть рассматриваемую архитектуру, а в Приложении В представлен реализующий ее программный код.

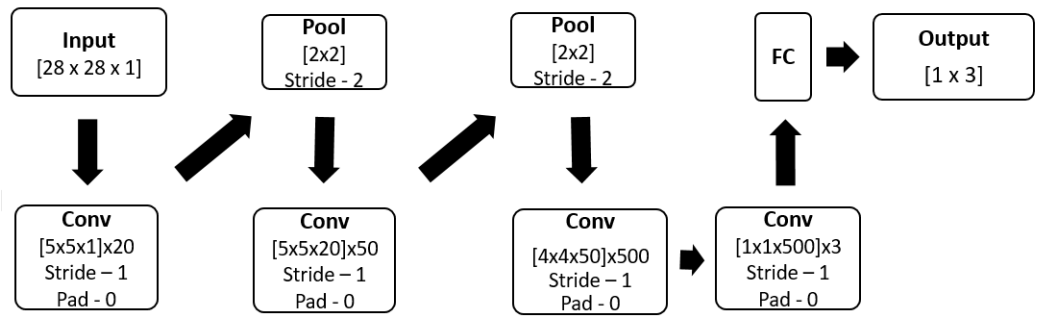

Рис. 4.4: Архитектура сверточной нейронной сети, принимающей на вход объекты размерности $[28 \times 28]$

При рассмотрении данной архитектуры можно видеть, что данная архитектура использует только четыре сверточных и два слоя объединения. Также значительно увеличено количество ядер свертки на некоторых слоях.

Также стоит отметить, что при подготовке данных из-за уменьшения размера спиновых изображений, передаваемых на вход сверточной нейронной сети, размеры "корзин"спиновых изображений были увеличены по сравнению с реализацией системы классификации, представленной в подразделе 4.2. В остальном процесс подготовки данных существенно не изменялся.

На рис.4.5 представлен график изменения значений функции ошибки для данной реализации системы.

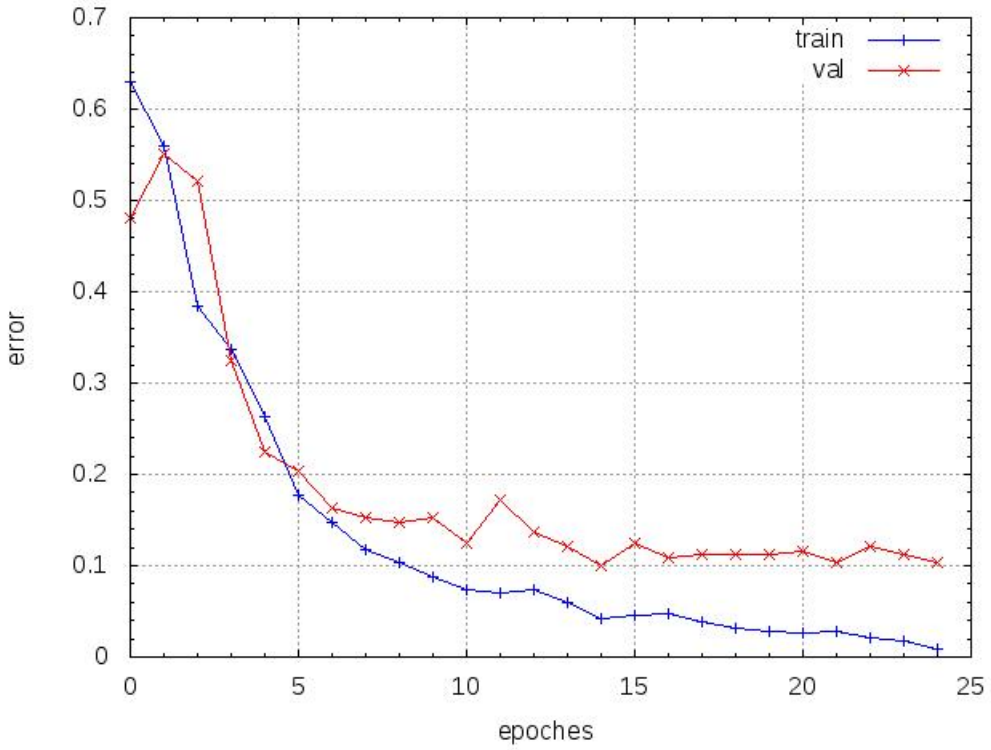

Рис. 4.5: График изменения значений функции ошибки сверточной нейронной сети, принимающей на вход объекты размерности $[28 \times 28]$

Уже на этапе обучения сети видно, что значение функции ошибки для данной реализации системы снижается для каждой эпохи анализа как для тренировочной, так и для проверочной выборок. Это свидетельствует о том, что сеть выделяет признаки из глобальных дескрипторов, а не просто «запоминает» тренировочные изображения.

4.4 Анализ результатов работы системы

Оценим результаты проведенных тестов для систем классификации трехмерных объектов с архитектурами сверточных нейронных сетей, представленных в подразделах 4.2 и 4.3. На рис.4.6 представлена диаграмма, отражающая результат распознавания для 20 случайных моделей трех классов, не входящих в обучающую выборку. В части диаграммы "Реализация системы №1"представлены результаты классификации моделей с использованием архитектуры сверточной нейронной сети, представленной в подразделе 4.2. Соответственно результат распознавания для архитектуры, представленной в подразделе 4.3 представлен в части диаграммы "Реализация системы №2".

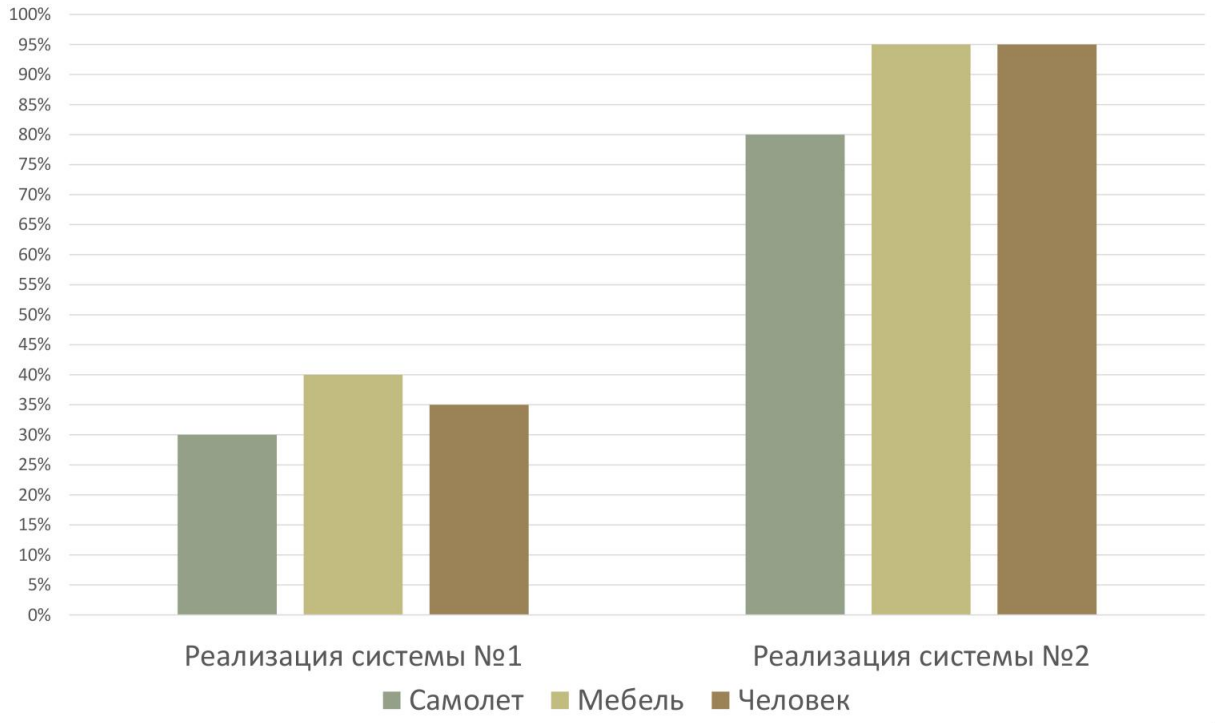

Рис. 4.6: Диаграмма результатов работы системы классификации трехмерных объектов

Из данной диаграммы видно, что первая реализация системы не показала хороших результатов распознавания. В лучшем случае сеть смогла правильно классифицировать не более 40% передаваемых ей объектов. Вторая же реализация системы показала наилучшие результаты из всех проведенных тестов. Качество распознавания в этом случае достигает 95%.

Заключение

В ходе работы была разработана система классификации трехмерных объектов. В процессе разработки данной системы были исследованы возможные представления (дескрипторы) трехмерных объектов, а также технологии сверточных нейронных сетей.

В ходе создания системы классификации трехмерных объектов были исследованы и реализованы алгоритмы подготовки моделей к построению глобальных дескрипторов формы поверхности в виде интегральных спиновых изображений, а также разработана и реализована сверточная нейронная сеть, классифицирующая объекты на три класса.

Анализируя результаты экспериментов можно сделать вывод, что предложенная система вполне способна выделять отличительные признаки трехмерных объектов не зависимо от их локальных различий в форме, масштаба и поворота в пространстве.

Основной проблемой при разработке подобной системы является большое количество варьируемых параметров на каждом ее этапе, а также малочисленность баз трехмерных объектов. В заключении стоит отметить, что полученный результат применения предложенной системы классификации при небольшом количестве трехмерных объектов дает предпосылки к тому, что при увеличении базы данных трехмерных моделей и вычислительной мощности, используемой для работы системы, возможно добиться корректных результатов распознавания данной системой для большого количества классов объектов.

Список литературы

[1] Автоматическое совмещение поверхностей в системах компьютерного зрения / А.А. Крыловецкий, И.С. Черников, С.Д. Кургалин // Математическое моделирование .— 2013 .— Т. 25, No 3. - С. 33-46.

[2] Horn B. K. P. Extended Gaussian images // Proc. of the IEEE, 72, 1984, pp. 1671-1686

[3] Kang S. B. The complex EGI: A new representation for 3D pose determination / S. B. Kang, K. Ikeuchi // IEEE Trans. Pattern Anal. and Mach. Intell., 15(7), 1993, pp. 707-721

[4] Novotni M. 3D zernike descriptors for content based shape retrieval / M. Novotni, R. Klein // In Proc. of the 8th ACM symposium on Solid modeling and applications (SM '03), New York, NY, USA, ACM Press, 2003, pp. 216-225

[5] Черников, И.С. Методы и алгоритмы реконструкции, поиска и визуализации трехмерных моделей: дис. канд. ф.-м. наук: 05.13.17:защищена 01.07.2014: утв. 05.11.14 / Черников Игорь Сергеевич. - Воронеж, 2013. - 145 с.

[6] Johnson A.E. Spin-Images: A Representation for 3-D Surface Matching, Ph. D. Thesis // Carnegie Mellon University, 1997, 288 p.

[7] Fukushima Kunihiko Neocognitron: A Self - organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. // Biological Cybernetics 1980 36 (4)

[8] CS231n: Convolutional Neural Networks for Visual Recognition: [Электронный ресурс].- 2017. - URL: http://cs231n.github.io (дата обращения 23.11.2016).

[9] Michael Neilsen. Neural Networks and Deep Learning / Michael Neilsen [Электронный ресурс].- 2017. - URL: http://neuralnetworksanddeeplearning.com (дата обращения 23.04.2017).

Приложение А

В данном приложении приведена реализация методов, которые отвечают за разделение ребер(edge-split) или их уничтожение(edge-collapse), в алгоритме контроля разрешения трехмерных объектов:

```
1  int edge_split(int ind)
2  {
3      int i;
4      Vert mid = calc_ed_midp(EdgeMass[ind]);
5      Diamond D;
6
7      if (!(EdgeMass[ind].sw))
8      {
9          printf("Trying to split delited edge!\n");
10         return 1;
11     }
12
13     D = edge_diamond(ind);
14     if (!D.val || ed_num ≥ DEF_EDGE_MASS_SIZE - 4)
15     {
16         return 1;
17     }
18     i = 0;
19     while(i < 4 && D.vert[i].x != -1)
20     {
21         add_edge(D.vert[i], mid, ed_num + i);
22         i++;
23     }
24     ed_num += i;
25     EdgeMass[ed_num].sw = 0;
26
27     EdgeMass[ind].sw = 0;
28
29     return 0;
30 }
31
32 int edge_collapse(int ind)
33 {
34     int i;
35     Star S;
36     Vert mid = calc_ed_midp(EdgeMass[ind]);
37
38     if (!(EdgeMass[ind].sw))
```

```
39      {
40          printf("Trying to collapse delited edge!\n");
41          return 1;
42      }
43
44      if(ed_num ≥ DEF_EDGE_MASS_SIZE - DEF_SV_MASS_SIZE)
45      {
46          return 1;
47      }
48
49      EdgeMass[ind].sw = 0;
50
51      S  = edge_star(ind);
52      if(S.length == -1)
53      {
54          return 1;
55      }
56      if(S.length == 0)
57      {
58          EdgeMass[ind].sw = 1;
59          return 0;
60      }
61
62      for(i = 0; i < S.length; ++i)
63      {
64          add_edge(mid, S.vert[i], ed_num);
65          ed_num++;
66      }
67      EdgeMass[ed_num].sw = 0;
68
69      return 0;
70  }
```

Приложение Б

В данном приложении приведена реализация архитектуры сверточной нейронной сети для классификации дескрипторов трехмерных объектов которой на вход передаются массивы размерности $[32 \times 32 \times 3]$.

```
function net = cnn_init()

lr = [.1 2] ;
net.layers = {} ;

net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.01*randn(3,3,3,32, 'single'), ...
                              zeros(1, 32, 'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 1) ;

net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ;

net.layers{end+1} = struct('type', 'relu') ;

net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(3,3,32,32, 'single'), ...
                              zeros(1,32,'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 1) ;

net.layers{end+1} = struct('type', 'relu') ;

net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ;

net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(3,3,32,64, 'single'), ...
                              zeros(1,64,'single')}}, ...
```

```
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 1) ;

net.layers{end+1} = struct('type', 'relu') ;

net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', [0 1 0 1]) ;

net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(4,4,64,64, 'single'), ...
                              zeros(1,64,'single')}}, ...
                           'learningRate', lr, ...
                           'stride', 1, ...
                           'pad', 0) ;

net.layers{end+1} = struct('type', 'relu') ;

net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{0.05*randn(1,1,64,3, 'single'), ...
                              zeros(1,3,'single')}}, ...
                           'learningRate', .1*lr, ...
                           'stride', 1, ...
                           'pad', 0) ;

net.layers{end+1} = struct('type', 'softmaxloss') ;

net.meta.inputSize = [32 32 3] ;
net.meta.trainOpts.learningRate = [0.05*ones(1,30) 0.005*ones(1,10)];
net.meta.trainOpts.weightDecay = 0.001;
net.meta.trainOpts.batchSize = 18;
net.meta.trainOpts.numEpochs = numel(net.meta.trainOpts.learningRate) ;

net = vl_simplenn_tidy(net) ;

end
```

Приложение В

В данном приложении приведена реализация архитектуры сверточной нейронной сети для классификации дескрипторов трехмерных объектов которой на вход передаются массивы размерности $[28 \times 28]$.

```
function net = init_cnn(varargin)

opts.networkType = 'simplenn' ;
opts = vl_argparse(opts, varargin) ;

rng('default');
rng(0) ;

f=1/100 ;
net.layers = {} ;
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(5,5,1,20, 'single'), ...
                              zeros(1, 20, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(5,5,20,50, ...
                              'single'),zeros(1,50,'single')}}, ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(4,4,50,500, 'single'), ...
                              zeros(1,500,'single')}}, ...
                           'stride', 1, ...
                           'pad', 0) ;
net.layers{end+1} = struct('type', 'relu') ;
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(1,1,500,3, 'single'), ...
```

```
                                    zeros(1,3,'single')}}, ...
36                                'stride', 1, ...
37                                'pad', 0) ;
38 net.layers{end+1} = struct('type', 'softmaxloss') ;
39
40 % Meta parameters
41 net.meta.inputSize = [28 28 1] ;
42 net.meta.trainOpts.learningRate = 0.001 ;
43 net.meta.trainOpts.numEpochs = 25 ;
44 net.meta.trainOpts.batchSize = 20 ;
45
46 % Fill in defaul values
47 net = vl_simplenn_tidy(net) ;
48
49 % Switch to DagNN if requested
50 switch lower(opts.networkType)
51   case 'simplenn'
52     % done
53   case 'dagnn'
54     net = dagnn.DagNN.fromSimpleNN(net, 'canonicalNames', true) ;
55     net.addLayer('top1err', dagnn.Loss('loss', 'classerror'), ...
56       {'prediction', 'label'}, 'error') ;
57     net.addLayer('top5err', dagnn.Loss('loss', 'topkerror', ...
58       'opts', {'topk', 5}), {'prediction', 'label'}, 'top5err') ;
59   otherwise
60     assert(false) ;
61 end
```