Санкт-Петербургский государственный университет

ГЛАДЫШЕВ Тимофей Юрьевич

Выпускная квалификационная работа

Алгоритмы машинного обучения в задаче реконструкции трехмерных объектов

Уровень образования: бакалавриат

Направление 01.03.02 «Прикладная математика и инфроматика»

Основная образовательная программа «Прикладная математика, фундаментальная информатика и программирование»

Профиль «Математическое и программное обеспечение вычислительных машин»

Научный руководитель:
доцент,
Кафедра теории систем управления электрофизической аппаратурой,
Козынченко Владимир Александрович

Рецензент:
Доктор физико-математических наук,
профессор кафедры теории управления,
Котина Елена Дмитриевна

Санкт-Петербург
2021

Содержание.

Введение.

Задача реконструкции объемных объектов уходит корнями в 60-е годы прошлого века, когда только начинали исследоваться возможности компьютеров по работе с трехмерной графикой. А когда в конце 80-х — начале 90-х годов индустрии кино и видеоигр стали движущим механизмом в развитии компьютерной графики, проблема реконструкции трехмерных объектов получила новый толчок.

Если необходимо представить с помощью графики некий объект, существующий в реальном мире, то обычно этим занимаются определенные специалисты — художники по 3D (3D Artist) — которые строят трехмерную модель полагаясь на фотографии или иные изображения объекта. Однако работа эта довольно сложная и трудоемкая. Очевидно возникает идея автоматизации данного процесса или хотя бы его части. Отсюда и появляется задача реконструкции.

Классические методы решения этой задачи строятся в основном на геометрических принципах. Имея множественные двумерные представления объекта (фотографии), описывающие его с разных ракурсов и зная при этом точную информацию о конкретных ракурсах, а также настройках камеры, таких как например фокусное расстояние (определяющее по большей части угол обзора), мы можем, строить проекции наших изображений на трехмерное пространство, как бы проекция за проекцией высекая нужную нам форму. Один из таких подходов сформулирован в [1].

Человек, однако, может представить форму объекта, увидев его всего раз, с одного ракурса. Мы способны на это благодаря большому опыту взаимодействия с разнообразными вещами. Когда мы видим какой-то предмет, то сразу ассоциируем его с другими подобными предметами, которые нам приходилось видеть в течение жизни, и таким образом можем

представить форму объекта, а также предположить, как этот объект будет выглядеть с другого ракурса.

Когда речь идет о решении подобных задач, которые человек выполняет с легкостью, несмотря на то, что строго алгоритмически задать их решения будет крайне проблематично, на помощь нам приходят методы машинного обучения.

На данный момент существует множество хорошо изученных и показывающих впечатляющие результаты, методов решения разнообразных задач обработки обычных, двумерных изображений, основанных на глубоком обучении (задача классификации, сегментации и т. п.) Соответственно, используя глубинные нейронные сети, мы можем легко обрабатывать входные изображения объемных объектов. Однако, выходной результат существующих алгоритмов обычно имеет вид числовых векторов, либо таких же двумерных изображений. И потому перед нами стоит задача построения трехмерных представлений, удобных для восприятия нейронной сетью, а также модификации существующих методов для получения возможности работы с этими представлениями.

Любая задача машинного обучения начинается с набора данных. В нашем случае, началом эры нейросетевой реконструкции можно считать 2015 год, дату публикации ShapeNet [2] — первого набора данных для работы с объемными объектами.

Первой серьезной работой в этой области можно считать 3D-R2N2 [3] — архитектура рекуррентной сверточной сети, представленная в 2016. На тот момент архитектура показывала state-of-the-art результат и до сих пор современные подходы сравниваются с ней по эффективности.

Также были и другие, относительно успешные на момент создания, подходы решения задачи. В их числе GAN архитектуры [4] — например 3D-VAE-GAN [5].

Другим популярным решением является разработанный в 2017 году принцип дифференцируемой согласованности лучей (differentiable ray consistency, DRC) [6]. Авторы этой работы предложили новый способ оценки согласованности двумерного и трехмерного представления объектов, что дало прирост эффективности.

Долгое время лучшим решением задачи реконструкции 3D объектов из изображений считался подход, использующий промежуточные, псевдотрехмерные наброски, в частности, предложенная в 2017 году архитектура MarrNet [7], усовершенствованная в 2018 году [8].

Однако на данный момент существуют и более эффективные подходы к решению задачи, например [9], в этой работе будет использоваться именно архитектура MarrNet так как при относительной эффективности работы, она не требует таких же трудозатрат, а также настолько серьезных вычислительных ресурсов, как более поздние решения.

Цели и задачи ВКР.

Целью данной выпускной квалификационной работой является построение прикладного решения задачи реконструкции трехмерных представлений объектов из одинарных изображений (то есть для объекта дано всего одно изображение с одного ракурса) с использованием технологий глубокого обучения. Задачу можно разбить на следующие подзадачи:

1. Выбор набора данных.
2. Предобработка данных из набора.
3. Разработка и реализация архитектуры необходимой нейросетевой модели.
4. Обучение нейросетевой модели.
5. Оценка и анализ работы решения.

Обзор литературы.

Для построения решения рассматриваемой задачи с использованием глубокого обучения (deep learning) необходимо доскональное теоретическое изучение тем и разделов, предполагаемых к использованию. Для этих целей предлагается следующая литература:

1. Ян Эрик Солем, "*Программирование компьютерного зрения на языке Python*" — рассказывает про обработку изображений средствами Python. Важной составляющей поставленной задачи является предобработка изображений.
2. Сантану Паттанаяк, "*Глубокое обучение и TensorFlow для профессионалов. Математический подход к построению систем искусственного интеллекта на Python.*" — в этой книге подробно объясняются разнообразные архитектуры глубинных нейронных сетей, в том числе используемые в данной работе, а также процесс их обучения.
3. Орельен Жерон, "*Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow*" — здесь поясняются принципы алгоритмов машинного обучения, книга также охватывает искусственные нейронные сети.
4. A. X. Chang, T. Funkhouser, L. Guibas et al., "Shapenet: An information-rich 3D model repository" [2] — одна из первых статей посвященных использованию нейронных сетей в задаче реконструкции объемных объектов, в статье представлен набор данных.
5. J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum. *"MarrNet: 3D Shape Reconstruction via 2.5D Sketches"* [7] — первая статья, представляющая двухэтапную архитектуру сети, идеи из которой используются в данной работе

6. X. Z. Xingyuan Sun, Jiajun Wu and Z. Zhang, *"Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling"* [8] — в статье представлен усовершенствованный набор данных для задачи реконструкции объемных объектов, а также некоторые улучшения архитектуры MarrNet, использованные в работе.

Глава 1. Нейронные сети в задачах компьютерного зрения.

В первой главе рассмотрена теоретическая составляющая работы, главным образом нейронные сети, их архитектуры, параметры и процесс обучения.

§1. Машинное обучение.

Машинным обучением называют класс методов решения задач, не предполагающих составление подробных инструкций (алгоритмов) для достижения поставленной цели, а вместо этого предполагающих использование математического аппарата (теория вероятностей, математический анализ, численные методы и т. д.) для поиска закономерностей среди имеющихся решений подобных задач.

В качестве примера типичного метода машинного обучения можно привести линейную регрессию. Имеется линейная функция с n параметрами, где n — размерность множества, на котором определена функция. Необходимо аппроксимировать[1] некое “обучающее” множество путем подбора наилучших параметров.

Данный метод относится к т. н. *обучению с учителем.* У нас есть обучающее множество, состоящее из входных данных алгоритма, а также соответствующих им выходных данных, меток. Помимо обучения с учителем, также разделяют *обучение без учителя,* в цели которого входит самостоятельный поиск алгоритмом закономерностей среди входных данных, а также *обучение с подкреплением,* где алгоритму заранее неизвестны искомые результаты, однако в процессе обучения размечаются

[1] Аппроксимация — задача поиска функции с наименьшим средним отклонением от заданного множества значений.

более или менее успешные попытки. Последний класс методов чаще всего используется в задачах управления.

В рамках этой работы будет рассматриваться так называемое *глубокое обучение*. В отношении искусственных нейронных сетей под глубоким обучением подразумевается обучение сетей с достаточно большим количеством слоев. Задачи, решение которых будет реализовываться, относятся к классу обучения с учителем.

§2. Архитектура и принципы обучения многослойного персептрона.

Самой первой архитектурой искусственной нейронной сети является *многослойный персептрон*. Данную архитектуру можно представить как ориентированный граф, имеющий несколько входных и несколько выходных путей — размерности сети соответственно. Узлы графа называются *нейронами*, а связи — *синапсами*. Нейроны соединены синапсами в слои, каждый нейрон соединен с каждым нейроном следующего и каждым нейроном предыдущего слоя. Нейрон принимает на вход взвешенную сумму чисел, полученных на выходе из нейронов предыдущего слоя (первый слой нейронов принимает взвешенную сумму элементов входного вектора сети), затем принятое значение преобразуется с помощью так называемой активационной функции. Например сигмоида, имеющая формулу $S(x) = \frac{1}{1+e^{-x}}$ используется чтобы ограничить значение промежутком от 0 до 1. Веса для суммирования элементов вектора ассоциируются с синапсами и все вместе составляют так называемый вектор весов. Можно представить модель многослойного персептрона (и любой другой архитектуры) как сложную вектор-функцию

с большим количеством суперпозиций умноженных на веса — параметры. И задачей обучения нейронной сети с учителем будет аппроксимация обучающего множества при помощи такой функции.

Итак, обучением нейронной сети называют процесс подбора параметров сети с целью наилучшего приближения обучающего множества. Представим этот процесс более формально. Обозначим за $N(x)$ — вектор-функцию — нашу нейронную сеть. $X = \{x_1, x_2, ..., x_n\}$ — входные данные из обучающего множества. $Y = \{y_1, y_2, ..., y_n\}$ — метки. $D(A, B)$ — функция расстояния между двумя множествами (например среднеквадратическое отклонение). Тогда определим *функцию потерь* как $L = D(\{N(x_i), i = 1,..., n\}, Y)$. Именно эту функцию, имеющую разный вид в зависимости от задачи, нам нужно минимизировать.

В основе методов обучения нейронных сетей стоит принцип спуска по градиенту. Градиент нейронной сети вычисляется при помощи метода обратного распространения ошибки. Теоретическое пояснения работы метода обратного распространения ошибки описано у Паттанаяка (см. Обзор литературы). Для изменения весов используются различные методы оптимизации функции ошибки, например Adam [12], SGD (Стохастический градиентный спуск) и т. п.

§3. Архитектура автокодировщика.

Простейшая модель автокодировщика представляет собой сеть прямого распространения (персептрон), состоящую из кодировщика и декодировщика. Кодировщик переводит входной вектор в промежуточный вектор меньшей размерности, затем декодировщик, имеющий симметричную с кодировщиком архитектуру, переводит промежуточный вектор в конечный результат. Изначально автокодировщики

использовались для “каскадного” предобучения глубоких сетей без учителя. При обучении автокодировщику на вход и выход подаются одни и те же значения. Благодаря уменьшенной размерности промежуточного вектора, также называемого вектором скрытых признаков, сеть вынуждена искать закономерности между данными, а не просто выдавать тривиальный результат. Позднее данный класс архитектур стали использовать во множестве задач, где вход и выход имеют одинаковый формат (например задача сегментации изображений, задача машинного перевода текста).

§4. Сверточные сети.

В 80-е годы прошлого века были придуманы первые сверточные нейронные сети. Свое название они получили от математической операции свертки. Сверточные слои могут иметь как одномерную, так и двух- и более мерную структуры. В случае двумерной сверточной сети, входным сигналом будет являться матрица, которая на каждом слое поэлементно перемножается на матрицы меньших размерностей — ядра — и таким образом сеть выделяет некоторые признаки.

Архитектуры сверточных нейронных сетей обычно представляют собой чередование сверточных и пулинговых слоев. Под пулинговыми слоями понимаются слои, понижающие размерность матрицы не с помощью операции свертки, а с помощью более простых методов, таких как поиск максимального или среднего значения фрагмента матрицы. Архитектура сверточной сети прямого распространения LeNet — одной из классических работ этой области — приведена на рис. 1. В LeNet помимо сверточной части также присутствует блок полносвязных слоев, выполняющий роль классификатора либо регрессора в зависимости от задачи.

Для обучения сверточных нейронных сетей применяются те же оптимизаторы, что и для обучения полносвязных сетей (типа многослойный персептрон).

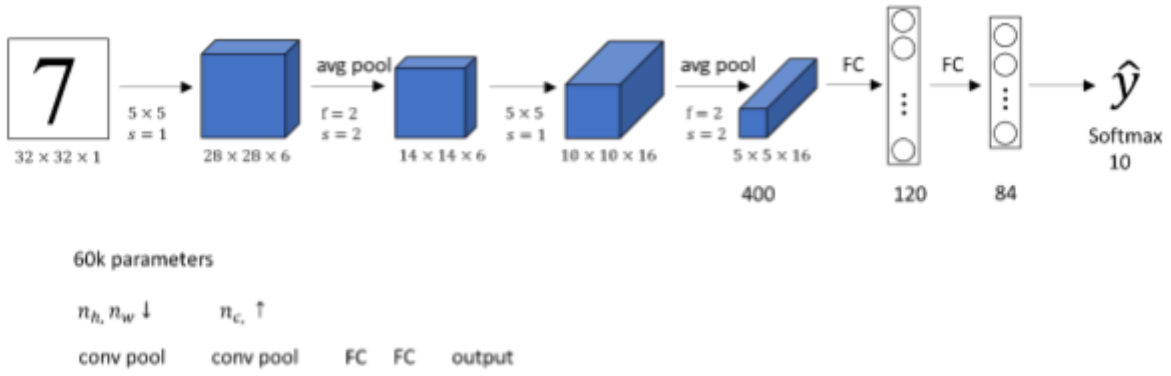

Рис. 1. Архитектура LeNet-5. (источник: datahacker.rs)

§5. U-net.

Архитектура U-net сети по сути представляет собой автокодировщик на базе сверточных слоев. Такой подход хорош для задач, в которых входом и выходом являются изображения схожей природы (одинаковый формат, разрешение и т. п.)

Классическим примером такой задачи является задача сегментации изображения. Имеется изображение, содержащее некий объект, а также множественные предметы окружения. Необходимо выделить “маску” (силуэт) объекта, т. е. сгенерировать изображение, в котором пиксели, позиция которых совпадает с позицией пикселей оригинального изображения, в которых содержится нужный нам объект, имеют значение насыщенности 1 (белый цвет), а все остальные — значение насыщенности 0 (черный цвет). Таким образом можно извлечь фрагмент путем простого поэлементного умножения.

Для решения этой задачи необходимо обучить кодировщик нашего U-net находить в изображении детали искомого объекта и кодировать необходимую информацию в вектор скрытых признаков. Затем, нужно, чтобы декодировщик, используя нужную информацию об объекте, строил силуэт. Как уже было сказано ранее, автокодировщики должны иметь симметричную структуру. Что же используется для создания эффекта, обратного свертке?

Для реализации принципа автокодировщика в U-net архитектуре используется понятие так называемого слоя транспонированной свертки. Их принцип заключается в распределении значений матрицы на матрицу большей размерности (обычно с помощью дублирования либо усреднения значений) и последующим проведением операции свертки, по итогу выходной результат имеет большую размерность. Математическая формулировка процесса транспонированной свертки описана у Паттанаяка (см. Обзор литературы)

§6. ResNet.

Архитектура ResNet [11] (residual network) использует принцип резидуальных связей и предназначена для улучшения качества работы и скорости обучения сверточных нейронных сетей за счет решения проблемы так называемого угасающего градиента (vanishing gradient problem). Проблема угасающего градиента возникает во время обучения глубоких нейронных сетей и связана с крайне малыми значениями градиента для некоторых весов, что препятствует их изменению и как следствие замедляет или даже останавливает процесс обучения. Дело в том, что для подсчета частных производных относительно некоторых весов (принцип метода обратного распространения ошибки) используется дифференцирование сложной функции. А так как многие функции

активации возвращают значения по модулю меньше единицы, то перемножая подобные значения множества раз, на выходе получаются крайне малые числа.

Идея резидуальных связей возникла в связи с открытием пирамидальных нейронов головного мозга. Суть архитектуры заключается в том, что некоторые связи соединяют нейроны, находящиеся не в соседних слоях, а пропуская некоторые слои (чаще всего один или два слоя). Благодаря этому удается снизить влияние проблемы угасающего градиента и повысить эффективность обучения сети. Схема резидуальных связей представлена на рисунке ниже:

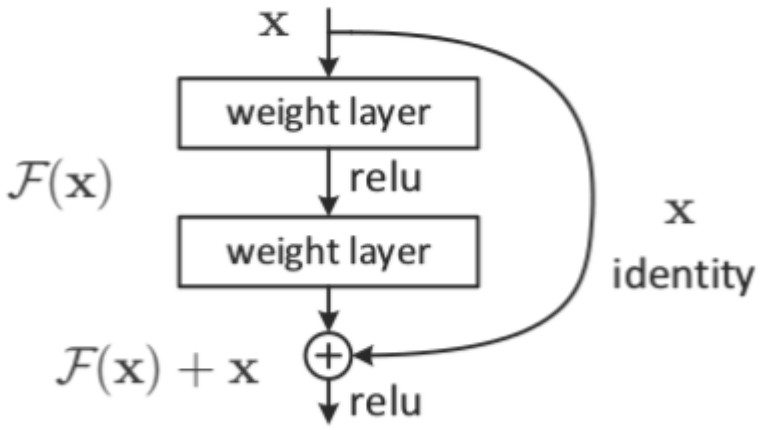

Рис. 2. Схема резидуальной связи ([11]).

Глава 2. Задача реконструкции трехмерных объектов из одинарных изображений.

Во второй главе будут рассмотрены формулировки постановки задачи, а также одного из методов ее решения.

§1. Математическая постановка задачи.

Сформулируем математическую постановку задачи. Обозначим за $I = \{I_k, k = 1,..., n\}$— множество изображений объектов. Сформулируем задачу реконструкции как поиск предсказательной функции $f(I)$, возвращающей упорядоченное множество объектов $\widehat{X}$, являющийся приближением множества неизвестных точных объектов X. Другими словами, задача — минимизация $L(I) = d_\Theta(f(I), X)$, где Θ — набор параметров f, а d — расстояние.

В нашем случае в качестве элементов множества I будут выступать RGB изображения в разрешении 256×256 пикселей, которые также можно представить в виде тензоров размерности $3 \times 256 \times 256$. А в качестве восстанавливаемых объектов множества X — сетки вокселей (кубов одинакового размера, объемный аналог пикселя) размерностью $128 \times 128 \times 128$.

§2. Набор данных.

Ранее уже приводились некоторые наборы данных для решения рассматриваемой задачи, например ShapeNet. Однако он, как и многие другие датасеты, имеет ряд недостатков. На основе работы [10] был

проведен анализ и выбран наиболее подходящий в нашем случае набор данных Pix3D.

Pix3D — набор данных, сформированный учеными Массачусетского технологического института. По словам его создателей, набор отличается от предшественников тем, что содержит несравнимо большее число данных (более 10 тысяч изображений против 759 в другом популярном наборе от IKEA), а также включает в себя как реальные, так и синтетические данные.

Набор состоит из 10 069 фотографий преимущественно предметов мебели. Каждым нескольким фотографиям ставится в соответствие одна трехмерная модель. Всего моделей 395, т. е. около 25 фотографий на штуку, что позволяет алгоритму обучаться как на фотографиях с разных ракурсов, так и на предметах, имеющих одинаковую форму, но разный внешний вид (поверхности разного цвета, текстуры и прочее).

Трехмерные модели представлены в наборе в двух видах: матрицы воксельной сетки, а также в классическом для компьютерной графики формате многогранников. Для решения задачи понадобилось использование псевдотрехмерных набросков. Для их получения многогранники были обработаны в бесплатном инструменте для трехмерного моделирования — Blender3D.

§3. MarrNet.

Для решения задачи используется нейросетевая модель, основанная на архитектуре MarrNet [7]. Основная идея данного решения заключается в двухступенчатости. Авторы этого подхода ссылаются на работу нейробиолога Дэвида Марра (отсюда и название архитектуры) 1982 года, который утверждал, что в своем восприятии форм человек полагается на

так называемые *первоначальные наброски* (primal sketches), на основе которых затем образуются объемные представления.

Предлагается решать задачу в два этапа. На первом этапе из разноцветных фотографий получаются так называемые карты нормалей и карты глубин. Карта нормалей — это изображение, в котором цвет пикселя используется как трехмерный вектор для задания угла наклона нормали к поверхности. Карта глубин — черно-белое изображение, на котором яркостью пикселя задается расстояние от поверхности до наблюдателя. Примеры карт нормалей и глубин представлены на рисунке ниже:

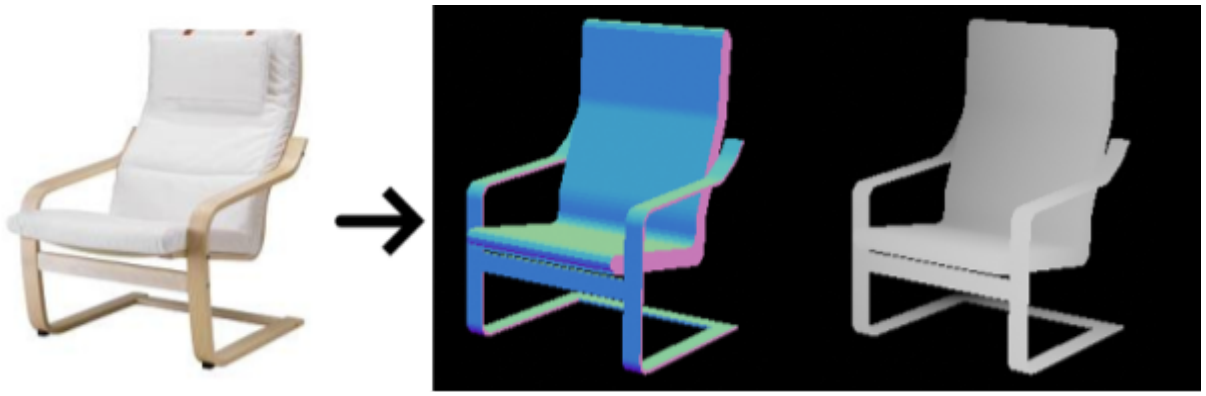

Рис. 3. Пример псевдо-трехмерных набросков.

Карты нормалей и глубин вместе с силуэтом объекта будем называть *псевдо-трехмерными набросками* (2.5D sketches, скетчи).

Архитектура первого модуля представляет из себя Unet-подобную модель с сетью ResNet-18 [11] (Таблица 1) в качестве кодировщика, переводящую изображение в 512-мерный вектор скрытых признаков. Затем декодировщик — зеркально отраженный ResNet — строит необходимые псевдо-трехмерные наброски. Будем реализовывать три экземпляра данной архитектуры для предсказания каждого из трех набросков (карта нормалей, карта глубин, силуэт объекта) по отдельности.

Таблица 1. Архитектура сети ResNet-18.

Тип слоя	Параметры
conv2d	размерность: 3 в 64, ядро:7×7, сдвиг:2, активация: relu
batchnorm2d	—
maxpool2d	ядро: 3×3, сдвиг: 2
conv2d	размерность: 64 в 64, ядро:3×3, сдвиг:1, активация: relu
batchnorm2d	—
conv2d	размерность: 64 в 64, ядро:3×3, сдвиг:1, активация: нет
batchnorm2d	—
conv2d	размерность: 64 в 64, ядро:3×3, сдвиг:1, активация: relu
batchnorm2d	—
conv2d	размерность: 64 в 64, ядро:3×3, сдвиг:1, активация: нет
batchnorm2d	—
conv2d	размерность: 64 в 128, ядро:3×3, сдвиг:2, активация: relu
batchnorm2d	—
conv2d	размерность: 128 в 128, ядро:3×3, сдвиг:1, активация: нет
batchnorm2d	—
conv2d	размерность: 64 в 128, ядро:1×1, сдвиг:2, активация: нет

batchnorm2d	—
conv2d	размерность: 128 в 128, ядро:3×3, сдвиг:1, активация: relu
batchnorm2d	—
conv2d	размерность: 128 в 128, ядро:3×3, сдвиг:1, активация: нет
batchnorm2d	—
conv2d	размерность: 128 в 256, ядро:3×3, сдвиг:2, активация: relu
batchnorm2d	—
conv2d	размерность: 256 в 256, ядро:3×3, сдвиг:1, активация: нет
batchnorm2d	—
conv2d	размерность: 128 в 256, ядро:1×1, сдвиг:2, активация: нет
batchnorm2d	—
conv2d	размерность: 256 в 256, ядро:3×3, сдвиг:1, активация: relu
batchnorm2d	—
conv2d	размерность: 256 в 256, ядро:3×3, сдвиг:1, активация: нет
batchnorm2d	—
conv2d	размерность: 256 в 512, ядро:3×3, сдвиг:2, активация: relu
batchnorm2d	—
conv2d	размерность: 512 в 512, ядро:3×3, сдвиг:1,

	активация: нет
batchnorm2d	—
conv2d	размерность: 256 в 512, ядро:1×1, сдвиг:2, активация: нет
batchnorm2d	—
conv2d	размерность: 512 в 512, ядро:3×3, сдвиг:1, активация: relu
batchnorm2d	—
conv2d	размерность: 512 в 512, ядро:3×3, сдвиг:1, активация: нет
batchnorm2d	—
adaptive avg pool 2d	разрешение: 1×1
fully connected	размер входа: 512, размер выхода: зависит от задачи

На втором этапе, на основе результатов работы первого модуля, предсказываются конечные воксельные сетки. Архитектура второго модуля основана на принципе автокодировщика. В начале аналогичная сеть ResNet-18 переводит наброски в 200-мерный вектор скрытых признаков, затем декодировщик на базе слоев трехмерной транспонированной свертки строит искомые сетки вокселей размерности 128^3. Подробная схема слоев декодировщика приведена в таблице ниже:

Таблица 2. Архитектура декодировщика второго модуля.

Тип слоя	**Параметры**
deconv3d	размерность: 200 в 512, ядро:4×4×4, сдвиг:1, активация: relu

batchnorm3d	—
deconv3d	размерность: 512 в 256, ядро:4×4×4, сдвиг:2, активация: relu
batchnorm3d	—
deconv3d	размерность: 256 в 128, ядро:4×4×4, сдвиг:2, активация: relu
batchnorm3d	—
deconv3d	размерность: 128 в 64, ядро:4×4×4, сдвиг:2, активация: relu
batchnorm3d	—
deconv3d	размерность: 64 в 32, ядро:4×4×4, сдвиг:2, активация: relu
batchnorm3d	—
deconv3d	размерность: 32 в 1, ядро:4×4×4, сдвиг:2, активация: relu

§4. Оценка работы сети.

Для оценки работы алгоритма будем применять *коэффициент Жаккара.* В англоязычной литературе он также носит название intersection over union (IoU). Данная метрика является распространенной в задачах сегментации, детекции. Она оперирует множествами любой природы и соответственно легко распространяется на трехмерное евклидово пространство, с которым мы работаем в рамках результата, который выдает модель.

По английскому названию метрики легко понять ее смысл. Ищется отношение пересечения двух множеств к их объединению. Соответственно

идентичные множества будут иметь значение метрики 1, а не пересекающиеся — 0. Формула:

$$IoU(A,\ B)\ =\ \frac{A \cap B}{A \cup B}$$

Глава 3. Практическая реализация и эксперимент.

Последняя глава посвящена рассмотрению практической реализации рассмотренных технологий для решения поставленной задачи. Сначала говорится о программном обеспечении, которое использовалось в работе, а затем непосредственно о эксперименте.

§1. Программное обеспечение.

Для практической реализации алгоритма реконструкции использовалась онлайн-среда разработки Google Colab. Выбор в сторону этого инструмента был сделан исходя из предоставляемой компанией Google возможности бесплатного удаленного использования профессионального аппаратного обеспечения, а именно Tesla K80 — дорогостоящей профессиональной видеокарты, используемой в обучении глубоких нейронных сетей.

Основной технологией, используемой в этой среде является популярный в сфере машинного обучения язык программирования Python версии 3.7. Для описания самих архитектур, а также их обучения использовался пакет PyTorch, представляющий из себя Python-интерфейс библиотеки для разработки глубокого изучения Torch, написанной на языках C/C++.

Также для предобработки, а именно создания псевдо-трехмерных скетчей, была использована бесплатная программа для редактирования трехмерных моделей Blender (https://www.blender.org/).

§2. Эксперимент.

Сперва была проведена предобработка данных. Важным шагом этого процесса было получение псевдо-трехмерных набросков из трехмерных моделей. В наборе данных модели представлены в формате OBJ и являют собой информацию о наборе вершин, соединенных гранями. Был написан код на языке Python, использующийся через встроенный Python-интерпретатор программы Blender для автоматизации процесса рендеринга (отрисовки) объектов с необходимым для получения набросков набором шейдеров (параметров рендеринга). Как итог, на выходе были получены псевдо-трехмерные скетчи в формате PNG разрешения 256×256 для каждой из представленных фотографий набора.

Для обработки самих фотографий использовался Python-модуль PIL (Python Imaging Library). Был создан программный класс наследующий `torch.utils.Dataset`. В рамках этого класса все изображения сжимались до разрешения 256×256, при этом если изображение имело соотношение сторон не 1:1 — оставшееся место заливалось черным цветом для набросков и белым — для фотографий. Затем все изображения приводились к матрицам типа данных `torch.Tensor`. Таким образом данные были приведены к формату, позволяющему корректную и оптимальную работу с нейронной сетью.

За основу самой сети было взято описание реализации архитектуры MarrNet в работе [8]. Обучение первого и второго модулей велось отдельно, более того, нововведением с моей стороны было разделение первого модуля еще на три составляющих — отдельные сети для предсказания каждого псевдо-трехмерного скетча. Данный подход позволил существенно повысить скорость обучения при этом не ухудшая точности работы конечной модели. Все три сети имеют одинаковую архитектуру за исключением размерности выходного сигнала. Для

реализации такого разделенного обучения мною были подобраны некоторые параметры обучения, такие как шаг (learning rate) и количество эпох и в итоге удалось добиться точности не хуже, чем у авторов. Для всех трех сетей первого модуля была использована функция погрешности MSE (среднеквадратическое отклонение). Модели, предсказывающие карту нормалей и карту глубин, обучались 32 эпохи каждая. Для модели, предсказывающей силуэт объекта, хватило всего 11 эпох. Для всех трех сетей был использован оптимизатор Adam [12] с шагом обучения 2×10^{-4}

Второй модуль был реализован как две отдельные сети (кодировщик и декодировщик) которые обучались совместно. На вход кодировщику подавались ground truth скетчи. Выходной тензор сравнивался с тензорами полученными из MAT файлов, входящих в состав набора данных и представляющих собой воксельную сетку $128 \times 128 \times 128$.

Сети второго модуля обучались в течение 37 эпох. В качестве функции погрешностей была взята BCE (бинарная перекрестная энтропия). Методом оптимизации послужил SGD (стохастический градиентный спуск) с шагом обучения 0.1 и импульсом 0.9. График обучения второго модуля изображен на Рис. 4.

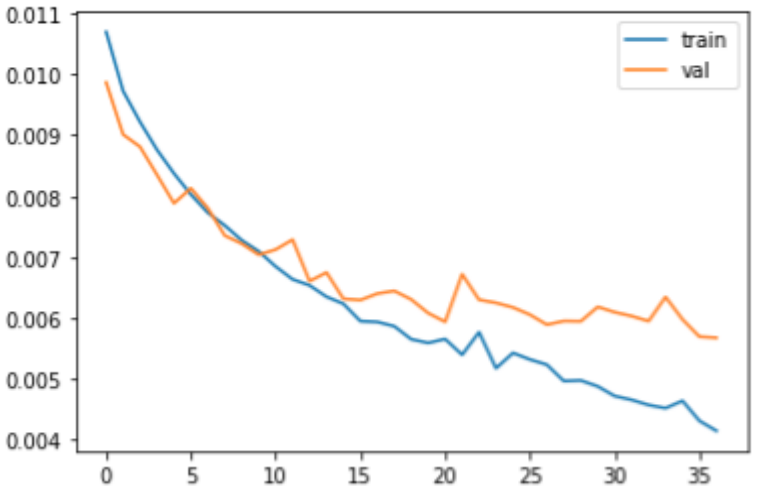

Рис. 4. График обучения второго модуля.

Оценка работы модели проводилась следующим образом: случайным образом было взято 100 изображений тестового множества (не использовавшегося во время обучения), для каждой фотографии множества был предсказан объемный объект путем последовательного использования обоих модулей. Так как выходной результат работы модели представляет собой тензор вещественных чисел, необходимо было рассчитать порог для вычисления метрики IoU. Расчет порога производился на промежутке [-5, 5] методом бинарного поиска. Было выяснено, что оптимальный порог для достижения наилучшего среднего значения IoU — -1. При использовании этого значения, метрика выдает показатель 0.52. Авторы оригинальной работы MarrNet представили показатель IoU равный 0.57. С учетом различий в подходах к решению задачи, таких как разный набор данных, модифицированная архитектура, а главное — существенный по сравнению с MarrNet дефицит вычислительных мощностей в рамках данной работы, можно считать два этих показателя эквивалентными, а следовательно работу — успешно выполненной. Результаты работы модели представлены на Рис. 5.

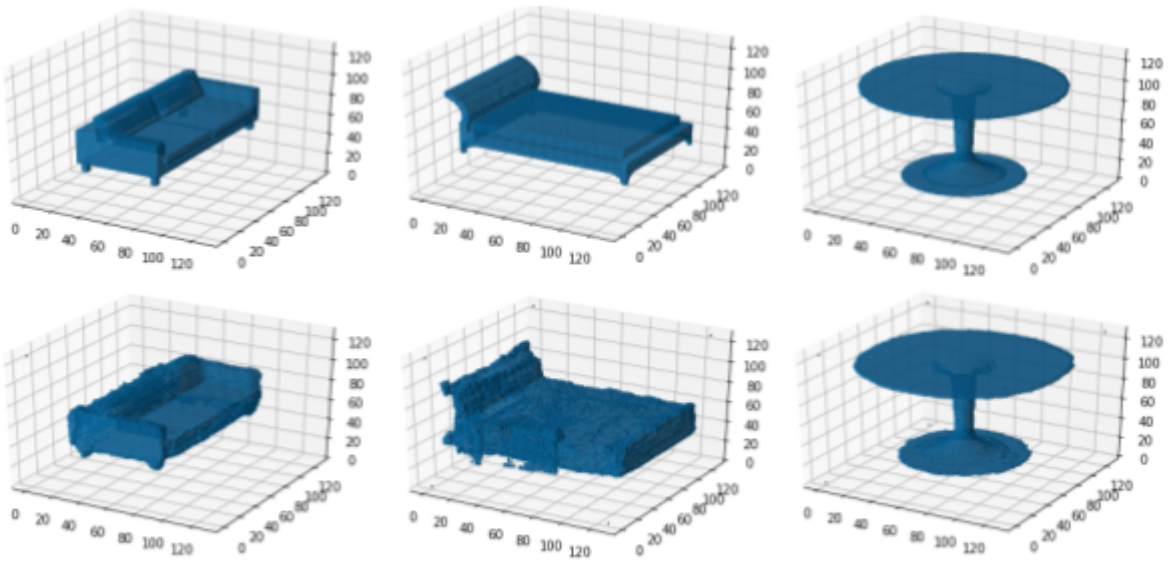

Рис. 5. Примеры работы модели. Сверху изображены ground truth объекты, снизу - предсказания сети.

Заключение.

В рамках данной выпускной квалификационной работы были выполнены:

1. Выбор набора данных для решения поставленной задачи. Был выбран набор Pix3D [8] так как он является относительно свежим и собранным специально для решения задачи, рассматриваемой в работе.
2. Предобработка данных из набора. Данные, включенные в набор, были недостаточными для реализации выбранного метода решения задачи. Набор был дополнен псевдо-трехмерными набросками. Также все данные были приведены к формату, оптимальному для восприятия нейросетевой моделью, использованной для решения задачи.
3. Разработка и реализация нейросетевой модели для решения задачи. На основе изученных работ посвященных рассматриваемой задаче, была разработана и реализована нейросетевая модель, которая впоследствии была модифицирована самостоятельно для повышения эффективности обучения и точности получаемого результата.
4. Обучение нейросетевой модели. Было проведено обучение разработанной модели с использованием обработанных данных из набора.
5. Оценка и анализ работы решения. Была оценена эффективность обучения и точность выдаваемых нейросетевой моделью результатов. Анализ этих показателей помог оптимизировать и модифицировать архитектуру сети, а также параметры обучения, что в конечном итоге позволило добиться достаточно хороших результатов.

Список литературы.

1. A. Laurentini, “The visual hull concept for silhouette-based image understanding,” IEEE TPAMI, vol. 16, no. 2, pp. 150–162, 1994.
2. A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su et al., “Shapenet: An information-rich 3D model repository” arXiv:1512.03012, 2015.
3. C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In ECCV, 2016
4. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. WardeFarley, S. Ozair, A. Courville, and Y. Bengio, “Generative Adversarial Nets,” in NIPS, 2014, pp. 2672–2680.
5. J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling. In NIPS, 2016.
6. S. Tulsiani, T. Zhou, A. A. Efros, and J. Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In CVPR, 2017
7. J. Wu, Y. Wang, T. Xue, X. Sun, W. T. Freeman, and J. B. Tenenbaum. MarrNet: 3D Shape Reconstruction via 2.5D Sketches. In NIPS, 2017.
8. X. Z. Xingyuan Sun, Jiajun Wu and Z. Zhang, “Pix3D: Dataset and Methods for Single-Image 3D Shape Modeling” in IEEE CVPR, 2018.
9. L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, A. Geiger, “Occupancy Networks: Learning 3D Reconstruction in Function Space”, CoRR, 2020.

10. Xian-Feng Han, Hamid Laga, Mohammed Bennamoun, “Image-based 3D Object Reconstruction: State-of-the-Art and Trends in the Deep Learning Era”, IEEE, 2019.

11. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, IEEE CVPR, 2016.

12. Diederik P. Kingma and Jimmy Ba, “Adam: A Method for Stochastic Optimization”, ICLR, 2015.