Санкт–Петербургский государственный университет

ДАНИЛОВ Глеб Сергеевич

Выпускная квалификационная работа

Реконструкция трехмерной модели объекта по одиночному снимку

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2019 «Прикладная математика, фундаментальная информатика и программирование»

Научный руководитель:
Кандидат технических наук,
доцент кафедры КММС
Гришкин В. М.

Рецензент:
Кандидат физико-математических наук,
доцент кафедры МУД
Шиманчук Д. В.

Санкт-Петербург
2023 г.

Содержание

Введение

В последние годы заметен возрастающий интерес к приобретению модных изделий через Интернет. Динамика онлайн-торговли в натуральном выражении продемонстрировала рост на 104% в 2021 году, на 65% в 2022 году, а для 2023 года прогнозируется увеличение рынка на 33-34% [1]. Тем не менее, несмотря на удобство интернет-шопинга, потребители зачастую испытывают опасения относительно соответствия выбранного на фотографиях товара модного предмета их размерам и стилю. В связи с этим, разработка технологии быстрой и доступной трехмерной реконструкции предметов гардероба может существенно улучшить опыт покупок, трансформировать подход к выбору одежды, а также привлечь новую клиентуру для розничных продавцов.

К тому же, данная технология обладает потенциалом расширения своего применения не исключительно в сфере электронной коммерции, но и в других областях, таких как виртуальная и дополненная реальность, а также компьютерные игры. Трехмерная реконструкция обыденного гардероба способствует оптимизации процесса наполнения открытых игровых миров реалистичными персонажами второго плана, и, в дополнение к этому, предоставляет пользователям возможность индивидуализации их визуального облика.

Ориентированность разрабатываемой технологии на скорость и дешевизну реконструкции является еще одним существенным аспектом, который делает ее привлекательной для широкого круга потенциальных пользователей и секторов применения.

Выбор предметов гардероба в качестве целевых объектов для трехмерной реконструкции в данном исследовании обусловлен несколькими причинами.

- Во-первых, эти объекты представляют собой интересный и сложный пример для реконструкции по одиночному снимку, поскольку они обычно имеют сложные формы и структуры, многовариантные фактуры и материалы.

- Во-вторых стоит отметить, что если модель успешно справляется с такой сложной задачей, как трехмерная реконструкция предметов гар-

дероба, то ее адаптация и дообучение для работы с более простыми объектами не составит большого труда.

Постановка задачи

Целью данной работы является разработка технологии для быстрой и экономически эффективной трехмерной реконструкции элементов повседневного гардероба, с акцентом на работу с ограниченными наборами данных. Задача обладает высокой сложностью, поскольку объекты гардероба часто обладают уникальной и детализированной структурой, что делает реконструкцию особенно требовательной к объемам данных для обучения моделей. Однако, в ходе исследования будет сформулирован подход, позволяющий эффективно обучать модель даже при ограниченном объеме данных.

Обзор литературы

Геометрические методы: Один из классических подходов к трехмерной реконструкции основан на геометрических методах, таких как стереосопоставление и триангуляция [2]. Геометрические методы используют информацию о камере и множественные точки зрения для определения глубины объектов и восстановления трехмерных моделей. Основными алгоритмами, используемыми в геометрических методах, являются Structure from Motion (SfM) [3] и Multi-view Stereo (MVS) [4]. SfM восстанавливает трехмерную структуру сцены и положения камеры, используя информацию о движении камеры между кадрами, в то время как MVS объединяет информацию из нескольких изображений для получения точных трехмерных моделей.

Методы оптимизации: Оптимизационные методы предлагают решение задачи трехмерной реконструкции путем поиска оптимального соответствия между двумерными изображениями и трехмерными моделями [5]. Эти методы включают в себя различные подходы, такие как эволюционные алгоритмы, градиентный спуск и другие алгоритмы оптимизации [6]. Оптимизационные методы могут быть применены как в комбинации с геометрическими методами, так и с методами обучения для достижения лучших результатов в трехмерной реконструкции. Примером такого метода является PatchMatch [7], который оптимизирует соответствие патчей между изображениями для восстановления глубины объектов.

Методы на основе плоскостей: Эти методы используют гипотезу о том, что объекты в сцене обычно состоят из плоских поверхностей или поверхностей с небольшими отклонениями от плоскостей [8]. Такие подходы могут быть особенно полезны для реконструкции архитектурных объектов и других структур с преимущественно плоскими поверхностями. Примером такого метода является Manhattan-World Assumption [9], который использует предположение о наличии ортогональных плоскостей в городских сценах для упрощения процесса трехмерной реконструкции.

Методы на основе нескольких модальностей: Иногда использование только двумерных изображений недостаточно для достижения хороших результатов в трехмерной реконструкции. В таких случаях, можно использовать

данные из разных источников и модальностей, таких как видео, лидар, радар или тепловые изображения. Использование нескольких модальностей может помочь улучшить точность реконструкции и обработать сложные сцены с низким контрастом или слабо различимыми объектами. Примером такого подхода является работа [10], в которой авторы предлагают метод трехмерной реконструкции на основе данных с лидара и стереокамеры для улучшения точности моделирования городских сцен.

Методы на основе обучения: С развитием глубокого обучения появились методы, основанные на использовании нейронных сетей для трехмерной реконструкции. Эти методы могут быть подразделены на две основные категории:

- Методы, использующие сверточные нейронные сети (CNN) для реконструкции на основе одного или нескольких изображений [11]. Примерами таких методов являются 3D-R2N2 [12], который восстанавливает трехмерные модели на основе набора двумерных изображений с помощью рекуррентного блока, Pix2Vox [13], который для реконструкции использует предобученную сверточную сеть, или Unet, который в своем подходе использует последовательное применение энкодера и декодера к входному изображению, в следствии чего итоговая архитектура напоминает латинску букву "U".

- Методы, применяющие генеративно-состязательные сети (GAN) для создания реалистичных трехмерных моделей. Примером такого метода является 3D-GAN [14], который использует генеративно-состязательную архитектуру для создания реалистичных трехмерных объектов.

Глава 1. Задача трехмерной реконструкции по одиночному снимку

1.1 Общая постановка задачи

В отличие от классической постановки задачи 3D-реконструкции, в данном случае доступен лишь один вид объекта, что затрудняет восстановление полной 3D-структуры. Основная идея подхода заключается в использовании глубоких сверточных сетей для извлечения признаков из 2D-изображения и последующего восстановления 3D-модели на основе этих признаков.

Общая постановка задачи включает следующие основные компоненты:

1. Формулировка задачи: Входными данными является 2D-изображение $\mathbf{I}$ объекта, а цель состоит в том, чтобы восстановить 3D-модель объекта $\mathbf{V}$, представленную в виде структуры (например, облака точек, трехмерной сетки или воксельной решетки). Функция потерь $\mathbf{L}$, которая оценивает разницу между предсказанной и истинной 3D-моделями, минимизируется в процессе обучения.

2. Архитектура нейронной сети: Для извлечения признаков из 2D-изображения и восстановления 3D-модели необходимо задать архитектуру нейронной сети.

3. Поиск или создание датасета: Важным этапом работы является поиск или создание подходящего датасета, содержащего пары 2D-изображений и соответствующих им 3D-моделей объектов. Этот датасет будет использоваться для обучения и оценки производительности разрабатываемой нейронной сети. В зависимости от типа объектов, для которых проводится реконструкция, может потребоваться создание собственного датасета, сгенерированного с использованием 3D-моделирования или полученного на основе реальных сцен.

4. Функция потерь: Функция потерь $\mathbf{L}$ определяет, насколько хорошо предсказанная 3D-модель $\mathbf{V}'$ соответствует истинной 3D-модели $\mathbf{V}$. Функция потерь минимизируется в процессе обучения CNN для улучшения качества предсказания 3D-модели.

5. Обучение: В процессе обучения нейронная сеть настраивает свои параметры с помощью градиентного спуска или других оптимизационных алгоритмов с целью минимизации функции потерь L. Обычно для обучения используется набор данных, состоящий из пар 2D-изображений и соответствующих им истинных 3D-моделей объектов. В процессе обучения сеть учится извлекать признаки из 2D-изображений и создавать на их основе 3D-модели объектов.

6. Подбор гиперпараметров и настройка оптимизатора: В процессе обучения нейронной сети необходимо определить оптимальные значения гиперпараметров, таких как скорость обучения, количество слоев, размеры ядер свертки, функции активации и др. Подбор гиперпараметров может быть выполнен с использованием различных методов, таких как поиск по сетке, случайный поиск или более продвинутые методы, такие как оптимизация на основе байесовских методов или генетических алгоритмов.

Глава 2. Методы решения

2.1 Критерии выбора архитектуры нейронной сети

В процессе выбора подходящей архитектуры нейронной сети для решения задачи трехмерной реконструкции объектов по одиночному снимку, ряд критериев был определен как основополагающий для принятия окончательного решения. В числе этих ключевых критериев были:

- Интерпретируемость работы нейронной сети: Важным критерием является возможность четкого понимания и объяснения процессов, происходящих внутри нейронной сети. Это облегчает анализ и оптимизацию модели, а также позволяет прогнозировать ее поведение на новых данных.

- Гибкость архитектуры нейронной сети: Необходимо иметь возможность адаптировать архитектуру нейронной сети под конкретную задачу, и, в случае необходимости, масштабировать ее для достижения компромисса между ресурсами и точностью.

- Скорость работы и низкие требования к вычислительной мощности при обучении сети: Важным аспектом при выборе архитектуры нейронной сети является ее производительность в сочетании с относительно небольшими требованиями к вычислительным ресурсам. Это позволяет обучать модель быстрее и с меньшими затратами, особенно в условиях ограниченной доступности мощных аппаратных ресурсов, таких как графические процессоры (GPU) или тензорные процессоры (TPU). Выбранная архитектура должна оптимально использовать доступные ресурсы, обеспечивая эффективное обучение и высокую скорость инференции, что является крайне важным для широкого круга пользователей.

Архитектура Unet [15] удовлетворяет основным критериям выбора, включая простоту интерпретируемости работы нейронной сети, возможность гибкой настройки архитектуры под решение конкретной задачи, а также ско-

рость работы и низкие требования к вычислительной мощности при обучении сети.

2.2 Математическое определение Unet

UNET является популярной архитектурой сверточной нейронной сети (CNN), предложенной Olaf Ronneberger и его коллегами для работы с изображениями на основе пикселей. Архитектура состоит из двух частей: энкодера (или сжимающего пути) и декодера (или расширяющего пути).

- Энкодер: Энкодер принимает на входе 2D-изображение объекта и преобразует его в скрытое представление (латентный вектор) при помощи сверточных слоев. В случае Unet, энкодер состоит из последовательности сверточных слоев с функциями активации ReLU и слоев макспулинга. Математически, для каждого сверточного слоя можно записать:

$$\mathbf{C}_l = f(\mathbf{I} * \mathbf{K}_l + \mathbf{b}_l),$$

где $\mathbf{I}$ — входные данные, $\mathbf{K}_l$ — ядро свертки для l-го слоя, $\mathbf{b}_l$ — смещение для l-го слоя, f — функция активации.

- Декодер: Декодер преобразует скрытые представления, полученные от рекуррентного блока, в 3D-модель объекта. В рассматриваемой модели декодер состоит из последовательности сверточных слоев с функциями активации ReLU и слоев апсемплинга (транспонированные свертки). Математически, для каждого сверточного слоя можно записать:

$$\mathbf{D}_l = f(\mathbf{H} * \mathbf{K}_l + \mathbf{b}_l)$$

где $\mathbf{H}$ — входные данные, $\mathbf{K}_l$ — ядро свертки для l-го слоя, $\mathbf{b}_l$ — смещение для l-го слоя, f — функция активации.

В результате, обобщенный подход семейства архитектур Unet использует энкодер для извлечения информации из 2D-изображений и декодер для восстановления 3D-модели объекта.

2.3 Энкодер

Далее произведем детальный анализ компонентов энкодера в контексте архитектуры Unet:

- Сверточный слой (CNN слой): В сверточном слое выполняется операция свертки между входными данными и ядром (фильтром) свертки. При этом каждый элемент выходной карты признаков получается путем вычисления скалярного произведения между ядром свертки и соответствующим участком входных данных. Математически свертка определяется следующим образом:

$$S(R, C) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} I(R + i, C + j) \cdot K(i, j)$$

Здесь:

  - $S(R, C)$ - результат свертки в позиции (R, C).
  - $I(R + i, C + j)$ - пиксели из исходного изображения.
  - $K(i, j)$ - соответствующие веса в ядре свертки.
  - m и n - размеры ядра свертки.

Размеры выходной карты признаков после применения сверточного слоя определяются с использованием следующих формул:

$$W_{out} = \frac{W_{in} - F_w + 2P_w}{S_w} + 1$$

$$H_{out} = \frac{H_{in} - F_h + 2P_h}{S_h} + 1$$

где:

  - W_{in} и H_{in} - ширина и высота входной карты признаков соответственно

- W_{out} и H_{out} - ширина и высота выходной карты признаков соответственно
- F_w и F_h - ширина и высота ядра свертки (фильтра) соответственно
- P_w и P_h - ширина и высота паддинга (дополнения нулями по краям входной карты признаков) соответственно
- S_w и S_h - шаг свертки по ширине и высоте соответственно Отметим, что эти формулы предполагают, что входная и выходная карта признаков имеют одинаковое количество каналов, что соответствует количеству фильтров в сверточном слое.

- Max-pooling слой: Слой максимального пулинга (max-pooling) применяется для уменьшения размерности входных данных, сохраняя при этом наиболее важные признаки. Это достигается путем выбора максимального значения в каждом участке входной карты признаков. Математически операция max-pooling определяется как:

$$P(R, C) = \max_{i=0}^{m-1} \max_{j=0}^{n-1} I(R + i, C + j)$$

Здесь:

- $P(R, C)$ - результат операции Max Pooling в позиции (R, C).
- $I(R + i, C + j)$ - пиксели из исходного изображения.
- m и n - размеры ядра Max Pooling.

- Leaky ReLU активация: Функция активации Leaky Rectified Linear Unit (Leaky ReLU) [16] является вариацией стандартной функции активации ReLU и используется для введения нелинейности в нейронные сети. Leaky ReLU предотвращает проблему "умерших"нейронов, которые могут возникнуть при использовании стандартной функции активации ReLU, сохраняя небольшой градиент при отрицательных значениях. Математически функция Leaky ReLU определяется следующим образом:

$$\text{LeakyReLU}(x) = \begin{cases} x, & \text{если } x \geq 0 \\ \alpha x, & \text{если } x < 0 \end{cases}$$

где x - входное значение, а α - небольшой положительный коэффициент утечки.

2.4 Декодер

Теперь рассмотрим декодер в контексте архитектуры Unet и обобщим его для трехмерного случая:

- Трехмерная свертка (3D Convolution) является обобщением классической двумерной свертки для обработки трехмерных данных, таких как 3D-изображения или видео. Трехмерная свертка выполняет операцию свертки между трехмерным входным тензором и трехмерным ядром (фильтром) свертки. Математически трехмерная свертка определяется следующим образом:

$$S(R, C, D) = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \sum_{k=0}^{p-1} I(R+i, C+j, D+k) \cdot K(i, j, k)$$

Здесь:

  - $S(R, C, D)$ - результат свертки в позиции (R, C, D).
  - $I(R+i, C+j, D+k)$ - воксели из исходных данных.
  - $K(i, j, k)$ - соответствующие веса в ядре свертки.
  - m, n и p - размеры ядра свертки.

Выходные размерности трехмерной свертки могут быть вычислены по следующим формулам:

$$W_{out} = \frac{W_{in} - F_w + 2P_w}{S_w} + 1$$

$$H_{out} = \frac{H_{in} - F_h + 2P_h}{S_h} + 1$$
$$D_{out} = \frac{D_{in} - F_d + 2P_d}{S_d} + 1,$$

где W_{in}, H_{in}, и D_{in} - ширина, высота и глубина входного тензора соответственно; F_w, F_h, и F_d - размеры ядра свертки по ширине, высоте и глубине соответственно; P_w, P_h, и P_d - отступы (padding) по ширине, высоте и глубине соответственно; S_w, S_h, и S_d - шаги (strides) свертки по ширине, высоте и глубине соответственно.

- UpSampling3D является операцией, которая увеличивает размерности трехмерного тензора по ширине, высоте и глубине с использованием метода интерполяции, такого как ближайший сосед (nearest neighbor) или трилинейная интерполяция (trilinear interpolation). Если размерности входного тензора (W_{in}, H_{in}, D_{in}), а множители увеличения размерности по ширине, высоте и глубине равны r_w, r_h и r_d, соответственно, то размерности выходного тензора после операции UpSampling3D равны $(W_{out}, H_{out}, D_{out})$, где:

$$W_{out} = r_w \cdot W_{in}$$
$$H_{out} = r_h \cdot H_{in}$$
$$D_{out} = r_d \cdot D_{in}$$

  Операция UpSampling3D часто используется в декодирующей части сетей, для восстановления исходных размеров объекта после его сжатия на предыдущих этапах обработки. Это позволяет нейронной сети генерировать высокоразрешенные трехмерные модели объектов на основе извлеченных признаков.

Глава 3. Набор данных

3.1 Критерии выбора обучающего набора данных

При выборе датасета для обучения модели, решающей задачу трехмерной реконструкции объектов гардероба по одиночному снимку, были определены основные критерии, на основе которых принималось окончательное решение. Среди этих ключевых критериев следует отметить:

- Открытый исходный код (open-source): Датасет должен быть доступен для свободного использования и распространения, что позволяет использовать его в личных исследованиях.

- Наличие реальных данных, а не сгенерированных синтетически: Датасет должен содержать трехмерные представления реальных объектов гардероба, поскольку это позволяет обучать модель на данных, более близких к реальным условиям использования, и улучшать качество реконструкции.

- Разнообразие: Для эффективного обучения модели и достижения высоких результатов, датасет должен включать в себя достаточное число разнообразных и качественных образцов трехмерных моделей объектов гардероба, представленных множеством различных классов одежды и их положений в пространстве.

Deep Fashion3D [17] удовлетворяет большинству указанных критериям, за исключением наличия двумерных обучающих изображений в открытом доступе. Однако, это ограничение может быть преодолено с помощью подхода генерации двумерных обучающих снимков по трехмерным объектам, который будет рассмотрен ниже.

3.2 Обзор состава датасета

В датасете содержится 597 трехмерных объектов одежды, представленных в формате облака точек [2] и разбитых на 9 [1] различных классов. Классы одежды включают в себя:

1. верхнюю одежду с длинными рукавами (long sleeve upper)
2. верхнюю одежду с короткими рукавами (short sleeve upper)
3. верхнюю одежду без рукавов (no sleeve upper)
4. платья с длинными рукавами (long sleeve dress)
5. платья с короткими рукавами (short sleeve dress)
6. платья без рукавов (no sleeve dress)
7. длинные брюки (long pants)
8. короткие брюки (short pants)
9. платья (dress)

Для некоторых трехмерных объектов в датасете присутствует дополнительная информация о различных позах в пространстве [3]. Это обогащает данные, предоставляя разнообразные примеры положений объектов гардероба, и способствует успешному обучению модели на разнообразных сценариях и ситуациях, связанных с трехмерной реконструкцией.

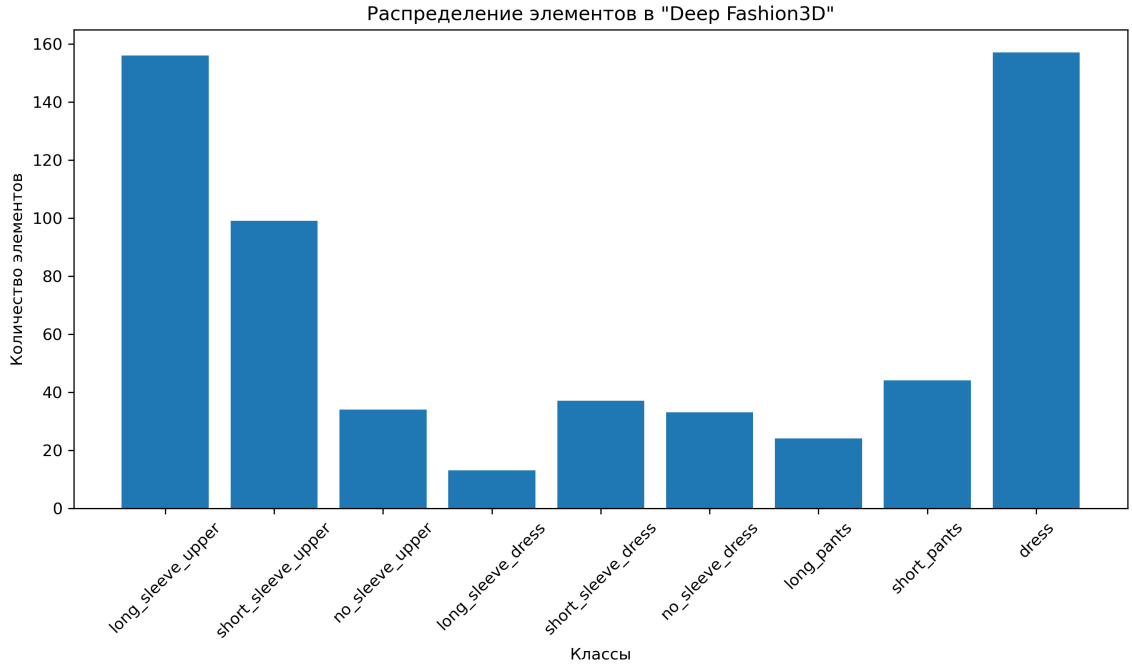

Рис. 1: Распределение элементов по классам

Рис. 2: Примеры объектов датасета представленных в виде облака точек

Рис. 3: Примеры разных положений в пространстве одного и того же объекта

3.3 Ограничения датасета

В процессе работы с датасетом были замечены некоторые ограничения, такие как отсутствие двумерных изображений, на основе которых были построены трехмерные объекты, в открытом доступе и малое количество обучающих данных.

Для решения проблемы, связанной с отсутствием двумерных изображений, было принято решение восстанавливать двумерные изображения на основе имеющихся облаков точек трехмерных объектов. Это позволило создать соответствующий набор данных из двумерных изображений, который затем использовался для обучения модели трехмерной реконструкции.

Для решения же проблемы малого объема обучающих данных будет проведена работа по детальной настройке гиперпараметров нейронной сети, для увеличения обобщающих способностей модели и предотвращения ее переобучения.

Процесс преобразования облака точек в соответствующее изображение состоял из нескольких шагов, одним из которых является Poisson Surface Reconstruction (PSR) [18]. Этот метод преобразования облака точек, которое не содержит информацию о сложных геометрических зависимостях, в объект формата OBJ, представляющий собой трехмерную модель, с информацией о тенях, текстуре и т.д.

Poisson Surface Reconstruction основан на решении уравнения Пуассона и использует следующую формулу:

$$\Delta f = \text{div}(v)$$

где Δf - это Лапласиан изображения f, которое мы хотим найти, $\text{div}(v)$ - это дивергенция векторного поля v, который представляет градиенты исходного изображения.

Алгоритм PSR выполняется в несколько этапов:

1. Оценка нормалей: на этом этапе вычисляются нормали к поверхности для каждой точки облака точек, используя методы, такие как Principal

Component Analysis (PCA) или другие алгоритмы оценки нормалей.

2. Решение уравнения Пуассона: после получения нормалей к поверхности решается уравнение Пуассона для вычисления скалярного поля $u(\mathbf{p})$. Это может быть выполнено с использованием различных методов, таких как многосеточный алгоритм или другие итерационные методы.

3. Извлечение поверхности: после того, как скалярное поле было вычислено, из него извлекается трехмерная поверхность, представляющая объект. Это может быть сделано с помощью алгоритмов, таких как Marching Cubes, Dual Contouring или других методов поверхностной реконструкции.

Для решения проблемы преобразования файла формата PLY в файл формата OBJ [4] была написана программная реализация с использованием описанного выше алгоритма.

Рис. 4: Примеры объектов датасета представленных в виде файла формата OBJ

Следующим шагом в процессе преобразования трехмерных моделей в двумерные изображения является рендеринг сцены. Для этого необходимо

настроить параметры сцены рендеринга, такие как свет, тени и положение камеры. Рассмотрим детально каждый из этих параметров:

- Свет: Важным аспектом рендеринга является настройка освещения сцены. Правильное освещение позволяет выделить детали объекта и создает реалистичное изображение. Обычно используются различные типы источников света, такие как направленный свет (directional light), точечный свет (point light) и рассеянный свет (ambient light). Комбинация этих источников света обеспечивает разнообразное и реалистичное освещение сцены.

- Тени: Тени играют важную роль в создании реалистичного изображения и добавляют глубину сцене. Для получения теней обычно используются техники, такие как shadow mapping или ray tracing. При использовании shadow mapping генерируется глубинная карта (depth map) относительно источника света, а затем она используется для определения видимости точек на объекте. Ray tracing позволяет вычислять тени, прослеживая лучи света от источника света до камеры через сцену.

- Положение камеры: Для получения вида объекта с произвольной точки зрения необходимо определить положение и ориентацию камеры. Положение камеры задается координатами (x, y, z) в пространстве, а ориентация определяется тремя углами Эйлера: углом крена (ϕ), углом тангажа (θ) и углом рыскания (ψ). Матрица вращения, представляющая ориентацию камеры, может быть получена из углов Эйлера следующим образом:

$$R = R_z(\psi)R_y(\theta)R_x(\phi)$$

где R_x, R_y и R_z - матрицы вращения вокруг осей x, y и z соответственно.

Каждая из матриц вращения R_x, R_y и R_z представляет собой вращение вокруг одной из осей координат: x, y и z соответственно. Они выглядят следующим образом:

– Вращение вокруг оси x на угол ϕ (угол крена):

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

– Вращение вокруг оси y на угол θ (угол тангажа):

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

– Вращение вокруг оси z на угол ψ (угол рыскания):

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Результатом последовательного умножения является матрица R размером 3×3, которая описывает ориентацию камеры в трехмерном пространстве.

С учетом описанных выше техник и алгоритмов была разработана программная реализация процесса рендеринга двумерного изображения из трехмерной сцены. Эта реализация обеспечивает возможность преобразования облака точек в соответствующие двумерные изображения с использованием Poisson Surface Reconstruction для создания геометрической модели объектов и настройки сцены рендеринга, включая управление светом, тенями и положением камеры с использованием матриц вращения на основе эйлеровых углов.

Поскольку в данном проекте нет необходимости восстанавливать цвет трехмерного объекта, было принято решение оптимизировать процесс обучения сети, исключив информацию о цвете из полученных двумерных изображений. Это позволит сократить время обучения и упростить обработку данных, сосредоточив внимание на геометрии и структуре объектов, а не на их цвете.

Глава 4. Построение архитектуры нейронной сети

4.1 Оптимизатор

В процессе обучения модели в качестве оптимизатора был применен алгоритм Adam. Оптимизатор Adam (Adaptive Moment Estimation) [20] представляет собой широко используемый алгоритм оптимизации при обучении нейронных сетей. Алгоритм является адаптивным методом градиентного спуска, объединяющим концепции двух других алгоритмов: RMSProp и Momentum.

Оптимизатор Adam выполняет свою работу путем оценки первого момента (среднего значения) и второго момента (несмещенной дисперсии) градиентов функции потерь. Он динамически адаптирует скорость обучения для каждого параметра на основе полученных оценок.

Алгоритм Adam состоит из следующих шагов:

1. Инициализация параметров:

   - $t \leftarrow 0$ (счетчик итераций)
   - $m_0 \leftarrow 0$ (оценка первого момента)
   - $v_0 \leftarrow 0$ (оценка второго момента)
   - α (скорость обучения, learning rate)
   - $\beta_1, \beta_2 \in [0, 1)$ (гиперпараметры для оценки моментов)
   - ϵ (малое число для предотвращения деления на ноль)

2. На каждой итерации t:

   - Вычисление градиента функции потерь $\nabla_\theta L(\theta_t)$.
   - Обновление оценки первого момента:

   $$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta L(\theta_t)$$

   - Обновление оценки второго момента:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta L(\theta_t))^2$$

- Корректировка смещения для первого и второго моментов:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

- Обновление параметров:

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

4.2 Инициализатор

В данной работе были рассмотрены следующие методы инициализации весовых коэффициентов нейронных сетей: He normal, LeCun normal и Glorot normal.

- He normal: Инициализация весов нейронной сети с использованием нормального распределения с нулевым средним и дисперсией $\sigma^2 = \frac{2}{n_{in}}$, где n_{in} - количество входных нейронов для слоя. Эта инициализация обеспечивает лучшую инициализацию весовых коэффициентов, особенно для глубоких сетей с активационными функциями Relu или Leaky ReLU. Формула для инициализации весов:

$$W \sim N(0, \sigma^2), \sigma^2 = \frac{2}{n_{in}}$$

- LeCun normal: Инициализация весов нейронной сети с использованием нормального распределения с нулевым средним и дисперсией $\sigma^2 = \frac{1}{n_{in}}$, где n_{in} - количество входных нейронов для слоя. LeCun normal инициализация особенно эффективна при использовании активационных функций симметричными относительно 0, таких как гиперболический тангенс. Формула для инициализации весов:

$$W \sim N(0, \sigma^2), \sigma^2 = \frac{1}{n_{in}}$$

- Glorot normal: Инициализация весов нейронной сети с использованием нормального распределения с нулевым средним и дисперсией $\sigma^2 = \frac{2}{n_{in}+n_{out}}$, где n_{in} и n_{out} - количество входных и выходных нейронов для слоя, соответственно. Glorot normal инициализация подходит для сетей с активационными функциями, симметричными относительно 0, таких как гиперболический тангенс или сигмоид. Формула для инициализации весов:

$$W \sim N(0, \sigma^2), \sigma^2 = \frac{2}{n_{in} + n_{out}}$$

Выбор подходящей инициализации весов является важным шагом при обучении нейронных сетей, так как он напрямую влияет на процесс обучения и сходимость алгоритма. Неправильная инициализация весов может привести к затуханию или взрыву градиентов, что замедляет обучение и может привести к неудовлетворительным результатам.

4.3 Метрики качества

В рамках настоящего исследования для оценки эффективности обученной модели были использованы две ключевые метрики: Binary Accuracy (бинарная точность) и Intersection over Union (IoU).

- Бинарная точность (Binary Accuracy): Эта метрика используется для оценки качества моделей бинарной классификации. Исходя из названия, эта метрика оценивает точность предсказаний модели в контексте двух классов (например, объект присутствует или отсутствует). В контексте этой работы, бинарная точность оценивает, насколько точно модель может классифицировать каждый пиксель на изображении как принадлежащий к объекту или фону.

Математически бинарная точность определяется следующим образом:

$$\text{Binary Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}},$$

где Number of Correct Predictions - это количество верно классифицированных пикселей (как объект или фон), а Total Number of Predictions - это общее количество пикселей на изображении.

- Intersection over Union (IoU): Эта метрика широко используется для оценки качества моделей в задачах сегментации и обнаружения объектов. IoU измеряет степень перекрытия между истинной областью объекта и предсказанной областью. Значение IoU равное 1 указывает на идеальное совпадение, в то время как значение 0 указывает на отсутствие перекрытия.

  Математически IoU определяется следующим образом:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}},$$

  где Area of Overlap - это область перекрытия между истинной и предсказанной областями, а Area of Union - это область объединения истинной и предсказанной областей.

Использование этих двух метрик позволяет получить более полное представление о производительности модели. Бинарная точность показывает, насколько точно модель может классифицировать каждый пиксель, в то время как IoU дает оценку точности локализации объекта на изображении.

4.4 Подбор гиперпараметров

В ходе проведения данного исследования был выполнен ряд эмпирических экспериментов, направленных на определение оптимального числа слоев в энкодере и декодере с целью обеспечения соответствующего баланса

между скоростью работы нейронной сети и ее способностью к обобщению.

В результате анализа экспериментальных данных было принято решение использовать нейронную сеть, состоящую из 6 блоков Conv2D-слоев в энкодере, после каждого из которых следует слой MaxPooling. В качестве функции активации была выбрана Leaky ReLU, которая позволяет решить проблему обнуления весов в глубоких нейронных сетях.

В декодере также было применено 6 блоков, состоящих из последовательного использования операции Conv3D на входных данных. Перед первыми четырьмя блоками применяется операция UpSampling3D, которая увеличивает пространственное разрешение передаваемых данных вдвое. Это позволяет получить на выходе декодера тензор размерностью 32 x 32 x 32 вокселей. В последнем слое была применена сигмоидная функция активации, которая предсказывает вероятность наличия объекта в соответствующем вокселе.

Для решения проблемы затухающих градиентов в глубокой нейронной сети была применена методика shortcut, также известная как связи пропускания или "skip connections". Эта методика представляет собой механизм, который позволяет пропускать один или несколько слоев в нейронной сети, создавая таким образом прямое соединение между входными и выходными данными слоев. В результате градиенты могут более эффективно распространяться обратно через сеть в процессе обратного распространения ошибки. Данная методика была применена в каждом блоке нейронной сети, соединяя выход предыдущего блока с выходом текущего блока.

На графике [5] видно, что добавление "skip connections"положительно повлияло на обобщающие способности модели.

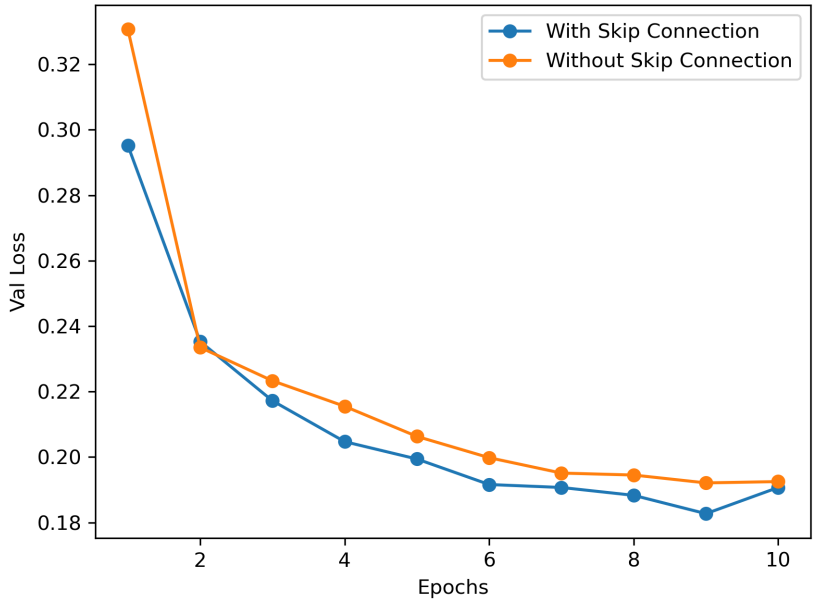

Рис. 5: Влияние "skip connection"на значение функции потерь

Для определения остальных гиперпараметров сети, таких как α, β_1, β_2, ϵ использовался случайный поиск по сетке. Алгоритм случайного поиска по сетке (Random Grid Search) [19] - это метод оптимизации гиперпараметров, который используется для выбора наилучшего набора гиперпараметров для модели машинного обучения. В отличие от традиционного поиска по сетке, случайный поиск по сетке не проверяет все возможные комбинации гиперпараметров, а выбирает случайный набор значений гиперпараметров из заданных диапазонов. К тому же, данный подход к подбору гиперпараметров, часто показывает сопоставимые результаты по сравнению с более сложными аналогами, такими как Байесовский поиск или Гинетические алгоритмы, но при этом затрачивает значительно меньше временных ресурсов [19]:

- Learning rate (скорость обучения, α): Скорость обучения определяет размер шага, с которым оптимизатор обновляет параметры модели. Математически это выражается следующим образом:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_\theta L(\theta_t),$$

где θ_t — вектор параметров модели на t-ом шаге обучения, $\nabla_\theta L(\theta_t)$ —

градиент функции потерь L по параметрам модели θ_t, а α — скорость обучения.

- Beta1 (β_1) и Beta2 (β_2): В используемом для оптимизации обучения алгоритме Adam применяются две экспоненциально взвешенные скользящие средние: одна для оценки первого момента градиента (среднее значение), а другая для оценки второго момента градиента (несмещенная дисперсия). β_1 и β_2 являются гиперпараметрами, которые определяют забывание предыдущих значений этих моментов. Они обновляются следующим образом:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2,$$

где m_t и v_t — оценки первого и второго моментов на t-ом шаге соответственно, а g_t — градиент функции потерь на текущем шаге.

- Epsilon (ϵ): Эпсилон является малой константой, добавляемой к оценке второго момента градиента для предотвращения деления на ноль при обновлении весов. В алгоритме Adam параметры обновляются следующим образом:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$
$$\theta_{t+1} = \theta_t - \frac{\alpha \cdot \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon},$$

где $\hat{m}_t$ и $\hat{v}_t$ — смещенные оценки первого и второго моментов на t-ом шаге соответственно. Таким образом, ϵ служит для обеспечения численной стабильности при обновлении параметров модели. Обычно ϵ выбирается очень малым, например, 10^{-8}.

В ходе исследования были проведены эксперименты с различными ком-

бинациями гиперпараметров, чтобы определить их влияние на значение функции потерь. Результаты представлены на графике [6].

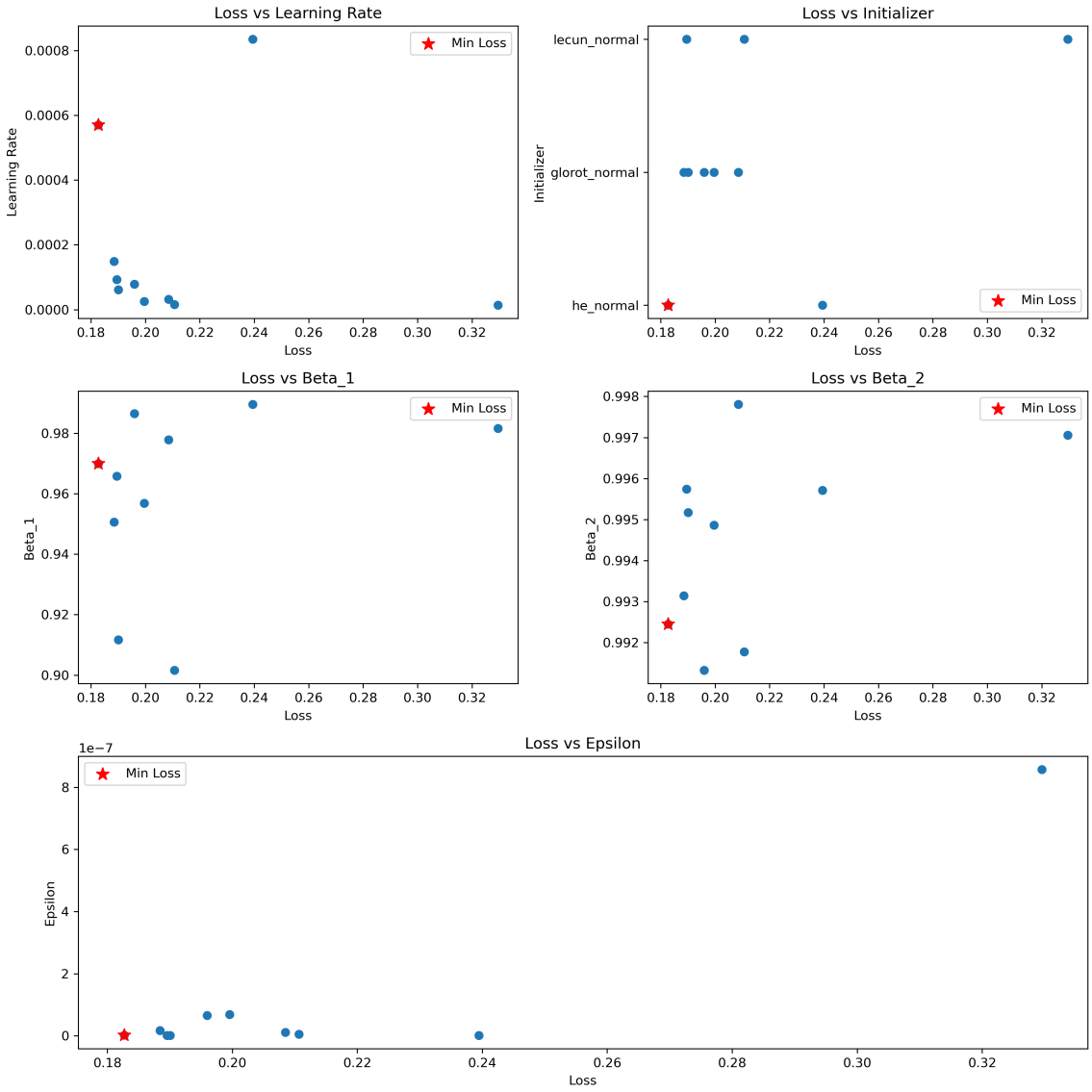

Рис. 6: Зависимость значения $Loss$ функции от гиперпараметров

Анализируя полученные результаты, можно сделать следующие наблюдения:

- Инициализатор $HeNormal$ дает наилучший результат функции потерь (0.1827) среди всех испытанных комбинаций гиперпараметров. Это говорит о том, что данная стратегия инициализации весов оказалась наиболее подходящей для данной модели.

- Наблюдается обратная корреляция между значениями функции потерь

и learning rate, что указывает на то, что с увеличением значения learning rate модель быстрее достигает оптимального значения функции потерь.

- Значения параметров β_1, β_2 и ϵ варьируются в пределах исследованных диапазонов, однако нет явной зависимости между этими параметрами и значениями функции потерь. Тем не менее, стоит отметить, что оптимальные значения функции потерь достигаются при $\beta_1 \approx 0.97$ и $\beta_2 \approx 0.99$, что может указывать на то, что в данной модели наиболее предпочтительными являются небольшие значения скорости накопления моментов.

- В случае параметра ϵ наблюдается слабая обратная корреляция между его значением и значением функции потерь. Это может указывать на то, что меньшие значения epsilon могут привести к более точным результатам при оптимизации модели.

- В ходе анализа также была выявлена зависимость [7] между совокупностью значений β_1 и β_2. Наблюдается тенденция, что комбинации с меньшими значениями β_2 и β_2 приводят к улучшению результатов обучения и меньшим значениям функции потерь. Это указывает на то, что в данной модели меньшие скорости накопления моментов могут способствовать более эффективному обучению и сходимости оптимизационного алгоритма. Важно отметить, что исследование совокупного влияния гиперпараметров позволяет получить дополнительные наблюдения о взаимодействии параметров и их эффекте на процесс обучения.

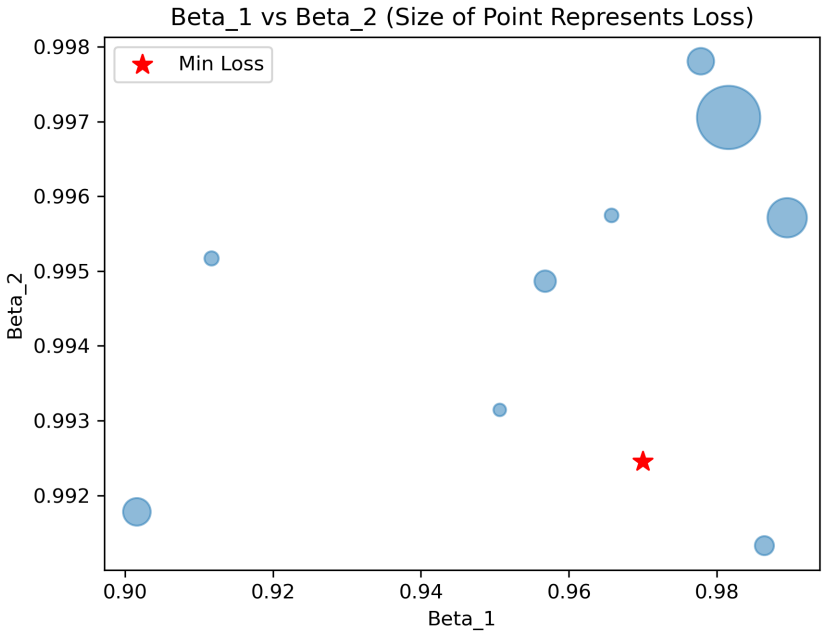

Рис. 7: Зависимость значения $Loss$ функции от β_1 и β_2

В целом, исследование показало, что выбор подходящих гиперпараметров имеет существенное влияние на процесс обучения модели и достижение оптимальных значений функции потерь. Особенно важным является выбор инициализатора весов и значения learning rate, поскольку они оказывают наибольшее влияние на результаты обучения.

4.5 Программная реализация

В рамках данного исследования выбор языка программирования был сделан в пользу Python, который широко используется в сообществе исследователей машинного и глубокого обучения для реализации алгоритмов, а также для создания пайплайнов сбора и обработки данных.

Реализация модели нейронной сети была выполнена с использованием инструментов фреймворка Keras. Детализированная архитектура разработанных энкодера [8] и декодера [9] представлена на приведенных ниже схемах.

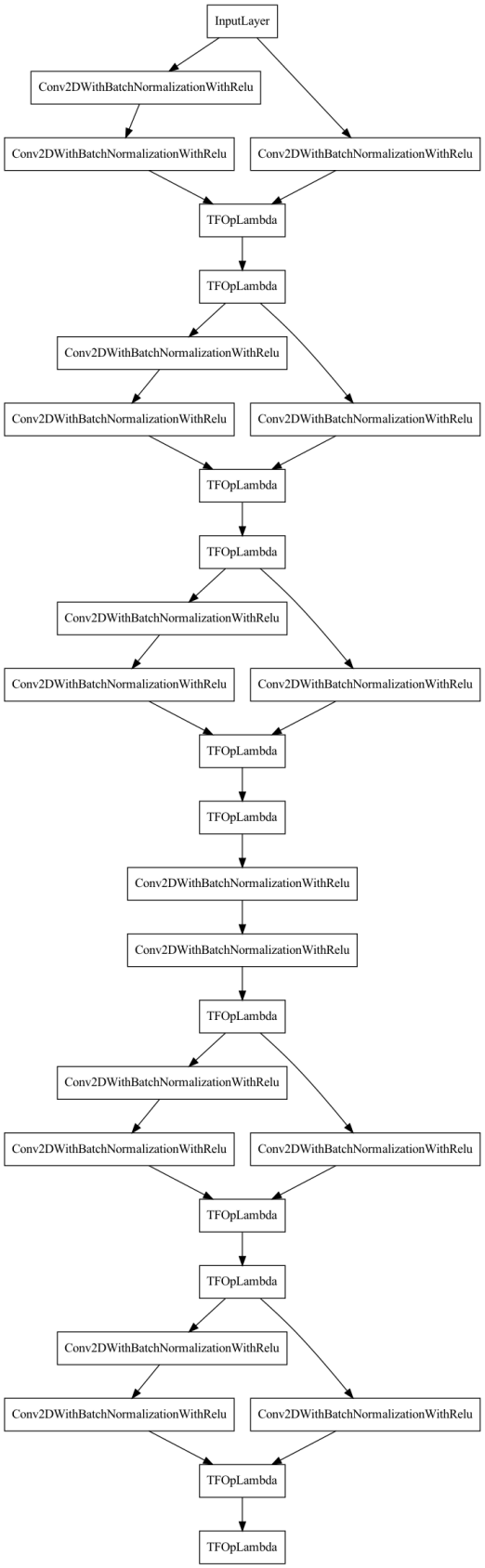

Рис. 8: Архитектура энкодера

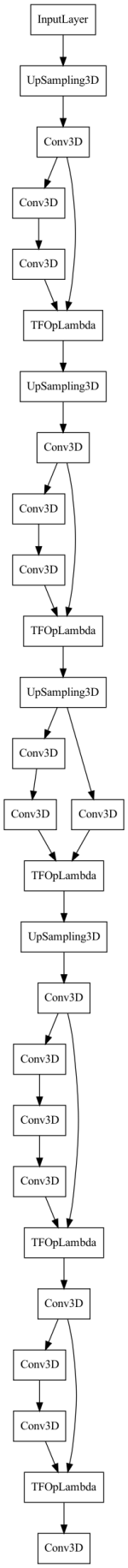

Рис. 9: Архитектура декодера

Глава 5. Обучение модели

5.1 Доступные аппаратные средства

Обучение представленной модели проводилось на встроенном графическом видеоядре, которое входит в состав процессора M1 Pro. Это графическое ядро имеет в своём распоряжении 8 ГБ видеопамяти формата GDDR-5. Важно отметить, что общий объем доступной видеопамяти составляет 16 ГБ, однако половина этого объема (8 ГБ) выделена под нужды операционной системы.

5.2 Процесс обучения

В ходе обучения разработанной модели, набор данных был стандартным образом разделен на две части - обучающие и валидационные данные - составляющие 90% и 10% от общего объема данных соответственно. Во время процесса обучения модель проходила валидацию на отложенных данных с целью предотвращения переобучения.

Для борьбы с переобучением также применялся метод уменьшения скорости обучения (learning rate) после определенного количества эпох без улучшения значения функции потерь на валидационных данных. Если значение функции потерь на валидационных данных не изменялось в течение 10 эпох, скорость обучения уменьшалась в 5 раз. В случае отсутствия улучшений на протяжении 20 эпох, модель завершала обучение и сохраняла наилучшее состояние с точки зрения значения функции потерь на валидационных данных.

Ход изменения метрик и лосс-функции на обучающем и валидационном наборе данных в процессе обучения можно увидеть на графике [10].

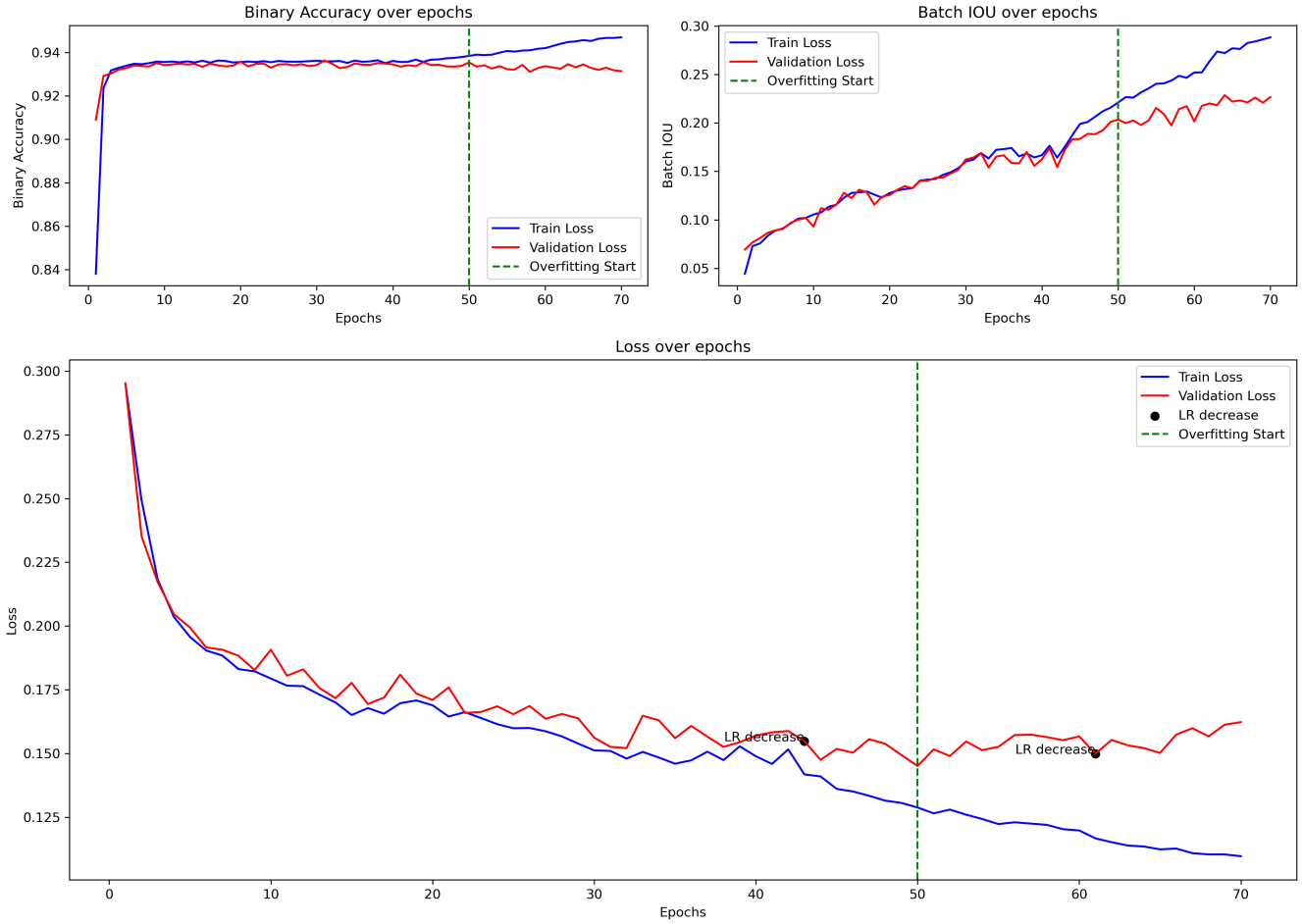

Рис. 10: Процесс обучения

Проанализировав графики, можно прийти к следующим выводам:

- Можно сделать вывод о том, что после 50-й эпохи наблюдается стагнация показателей на валидационных данных. При этом модель продолжает оптимизироваться на обучающих данных, что свидетельствует о возможном переобучении.

- Было обнаружено, что на 43-й эпохе скорость обучения уменьшилась в 5 раз. Это произошло в результате наблюдения стагнации показателей функции потерь на валидационных данных в течение предыдущих 10 эпох. Уменьшение скорости обучения в этот момент позволило модели "тонко"настроить веса, увеличив точность предсказаний на валидационных данных.

В итоге модель смогла достичь следующих показателей метрик на валидационных данных:

1. Бинарная точность (Binary Accuracy): **0.9354**
2. Intersection over Union (IoU): **0.2035**

Глава 6. Обзор результатов

6.1 Визуализация предсказаний

В ходе настоящего исследования была обучена модель, которая успешно продемонстрировала свою способность к воспроизведению общих образов и деталей входных изображений в трехмерном пространстве [11]. Однако, следует отметить, что модель показала некоторые ограничения в воспроизведении более мелких деталей, что может быть связано с обучением на относительно небольшом наборе данных, включающем менее тысячи изображений.

Предположительно, увеличение объема обучающих данных может способствовать повышению качества работы модели и улучшению детализации изображений. Это основывается на широко признанной теории, что более обширный и разнообразный набор данных может обеспечить более глубокое и точное обучение модели.

Одним из заметных преимуществ рассматриваемой модели, отличающих ее от других подходов, является эффективность в отношении скорости обучения и требований к объему видеопамяти. Несмотря на относительно низкое потребление вычислительных ресурсов, модель способна достигать удовлетворительного уровня качества, что делает ее привлекательной альтернативой в сценариях с ограниченными ресурсами.

В совокупности, преимущества модели, включая ее способность к обучению и воспроизведению изображений в трехмерном пространстве, высокую скорость обучения и относительно низкие требования к видеопамяти, делают ее мощным инструментом для задач, где требуется оптимальное сочетание качества результатов и доступных вычислительных ресурсов.

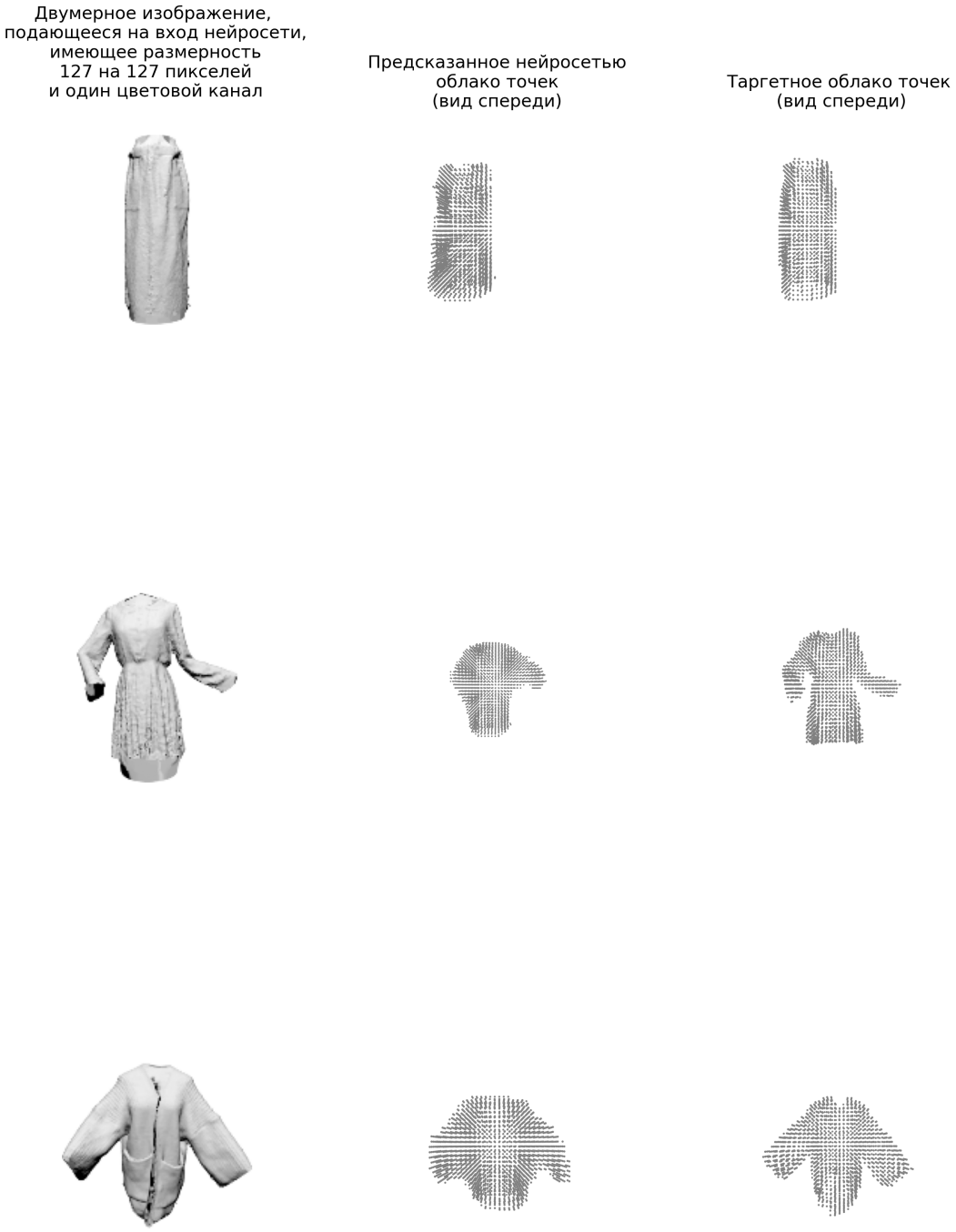

Рис. 11: Результаты работы нейросети

6.2 Сравнение результатов

Рассмотрим основные state-of-the-art подходы к решению поставленной задачи:

- 3D-R2N2 (3D Recurrent Reconstruction Neural Network) – это нейросетевая архитектура, предназначенная для восстановления трехмерных структур из набора двумерных изображений. Она состоит из двух основных компонентов:

  1. Сверточная нейросеть (CNN): Этот компонент используется для преобразования входных 2D-изображений в скрытые векторные представления.
  2. Рекуррентная нейросеть (RNN) с модулями долгой краткосрочной памяти (LSTM или GRU): Этот компонент используется для агрегации скрытых представлений от CNN и генерации трехмерной структуры.

  В конце процесса RNN применяет 3D-деконволюционные слои для восстановления конечной трехмерной структуры из скрытых представлений. Функция потерь, используемая для обучения, сравнивает итоговую 3D-модель с истинной 3D-моделью, что позволяет обучать модель на реальных данных.

- Pix2Vox представляет собой метод, основанный на концепции объединения кодировщика и декодировщика с использованием предварительно обученной сети VGG16[21] в качестве базовой структуры. Два варианта этого подхода, Pix2Vox-F и Pix2Vox-A, имеют одно ключевое отличие: Pix2Vox-A инкорпорирует дополнительный модуль, именуемый Refiner. Этот модуль включает в себя последовательное применение операции 3D-конволюции к выходу декодера, что позволяет дополнительно уточнить полученные трехмерные структуры. Это отличие делает Pix2Vox-A более мощным вариантом для задач, требующих повышенной точности восстановления трехмерных объектов.

Ниже представлены результаты сравнения метрик реконструкции разными подходами по синтетическим изображениям:

Модель	IoU	Количество параметров (в миллионах)
Исследуемая модель	0.204	8
3D-R2N2	0.136	50
Pix2Vox-F	0.271	10
Pix2Vox-A	0.288	100

Вывод по представленному сравнению можно сформулировать следующим образом:

Рассматривая метрику Intersection over Union (IoU), наблюдается, что исследуемая модель превосходит подход 3D-R2N2, достигая значения 0.204 против 0.136 соответственно. Однако, она уступает моделям Pix2Vox-F и Pix2Vox-A, у которых значения IoU составляют 0.271 и 0.288 соответственно.

С другой стороны, стоит обратить внимание на количество параметров, которое является индикатором сложности модели и, следовательно, требований к вычислительным ресурсам. Исследуемая модель имеет всего 8 миллионов параметров, что значительно меньше по сравнению с другими моделями. Модель 3D-R2N2 имеет 50 миллионов параметров, в то время как Pix2Vox-F и Pix2Vox-A обладают 10 и 100 миллионами параметров соответственно.

Таким образом, несмотря на то, что исследуемая модель не демонстрирует наивысший показатель по метрике IoU, она обеспечивает разумный баланс между качеством реконструкции и сложностью модели. Это делает ее привлекательной альтернативой для задач, где необходимо учитывать ограничения вычислительных ресурсов.

Дополнительно стоит упомянуть, что модели 3D-R2N2 и Pix2Vox были обучены на наборе данных ShapeNet [22], который содержит более 50,000 реальных объектов, охватывающих 55 различных категорий. Это значительно больше по сравнению с количеством обучающих образцов, используемых в

нашем исследовании.

Таким образом, несмотря на то, что исследуемая модель обеспечивает меньшую точность по сравнению с моделями 3D-R2N2 и Pix2Vox, следует учитывать, что они обучались на более обширном и разнообразном наборе данных. Это отмечает потенциал исследуемой модели для улучшения своих результатов при обучении на большем наборе данных.

Глава 7. Заключение

В ходе проделанных исследований была сформулирована и реализована архитектура нейронной сети, на основе анализа существующих эталонных методов в области трехмерной реконструкции. Разработанная модель способна успешно реконструировать объекты гардероба, обеспечивая при этом приемлемое качество исходя из поставленных задач.

К тому же, в рамках сложившихся ограничений набора данных, а в частности отсутствия в открытом доступе обучающих двумерных изображений, был разработан подход восстановления двумерных обучающих изображений по их трехмерным таргетам, что позволило минимизировать негативное влияние замены реальных данных синтетическими на разработанную модель.

Отдельно стоит отметить, что разработанный подход обладает низкими требованиями к вычислительным ресурсам, что делает его доступным для использования на широком спектре аппаратных платформ. Это открывает возможности для гибкого масштабирования и расширения целевой аудитории пользователей разработанной модели.

Несмотря на геометрическую сложность объектов гардероба, модель продемонстрировала способность к достаточно качественному их предсказанию. Это предоставляет сильный фундамент для дальнейшего обобщения разработанного подхода на другие трехмерные классы объектов.

В сравнении с аналогами, представленный подход характеризуется значительно меньшим количеством параметров, что, тем не менее, не приводит к существенной потере в качестве. Таким образом, результаты работы подтверждают эффективность и перспективность выбранной стратегии в решении задач трехмерной реконструкции.

Весь написанный код можно посмотреть в репозитории на GitHub.

Список литературы

[1] https://www.vedomosti.ru/business/articles/2023/03/23/967746-onlain-prodazhi-v-rossii-po-itogam-2022-goda-uvelichilis. 2023.

[2] Seitz S. M., Curless B., Diebel J., Scharstein D., Szeliski R. «A comparison and evaluation of multi-view stereo reconstruction algorithms». IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2006.

[3] Snavely N., Seitz S. M., Szeliski R. «Photo tourism: Exploring photo collections in 3D». In ACM Transactions on Graphics (TOG). 2006.

[4] Szeliski R. «Computer vision: Algorithms and applications. Springer Science & Business Media». 2010.

[5] Furukawa Y., Ponce J. «Accurate, dense, and robust multi-view stereopsis». IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010.

[6] Newcombe R. A., Lovegrove S. J., Davison A. J. «DTAM: Dense tracking and mapping in real-time». In 2011 International Conference on Computer Vision. 2011.

[7] Connelly B., Eli S., Adam F., Dan B. G. «PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing». ACM Trans. Graph. 2009

[8] Hoiem D., Efros A. A., Hebert M. «Automatic photo pop-up». ACM SIGGRAPH. 2005

[9] James C., Alan Y. «Manhatten world». In Neural Computation. 2003.

[10] Geiger A., Lenz P., Urtasun R. «Are we ready for autonomous driving? The KITTI vision benchmark suite». IEEE Conference on Computer Vision and Pattern Recognition. 2012.

[11] Keiron O'Shea, Ryan N. «An Introduction to Convolutional Neural Networks». ArXiv e-prints. 2015.

[12] Christopher B. C., Danfei X., Jun Y. G., Kevin C., Silvio S. «3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction». ArXiv e-prints. 2016.

[13] Haozhe X, Hongxun Y, Xiaoshuai S, Shangchen Z, Shengping Z. «Pix2Vox: Context-aware 3D Reconstruction from Single and Multi-view Images». ArXiv e-prints. 2019.

[14] Jiajun W., Chengkai Z., Tianfan X., William T. F., Joshua B. T. «Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling». NeurIPS. 2016

[15] O. Ronneberger, P. Fischer, T. Brox «U-Net: Convolutional Networks for Biomedical Image Segmentation». ArXiv e-prints. 2015.

[16] Bing X., Naiyan W., Tianqi C., Mu L. «Empirical Evaluation of Rectified Activations in Convolutional Network». ArXiv e-prints. 2015.

[17] Heming Z., Yu C., Hang J., Weikai C., Dong D., Zhangye W., Shuguang C., Xiaoguang H. «Deep Fashion3D: A Dataset and Benchmark for 3D Garment Reconstruction from Single Images». ArXiv e-prints. 2020.

[18] M. Kazhdan, M. Bolitho, H. Hoppe «Poisson Surface Reconstruction». Eurographics Symposium on Geometry Processing. 2006.

[19] P. Liashchynskyi, P. Liashchynskyi «Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS». ArXiv e-prints. 2019.

[20] D. P. Kingma, J. Ba «Adam: A Method for Stochastic Optimization». ArXiv e-prints. 2014.

[21] K. Simonyan, A. Zisserman «Very Deep Convolutional Networks for Large-Scale Image Recognition». ArXiv e-prints. 2014.

[22] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, F. Yu «ShapeNet: An Information-Rich 3D Model Repository». ArXiv e-prints. 2015.