

**Министерство образования и науки Российской Федерации  
Московский физико-технический институт (государственный университет)**

**Физтех-школа прикладной математики и информатики  
Кафедра банковских информационных технологий**

**Выпускная квалификационная работа бакалавра**

# **Разработка системы автоматизированного проектирования PointCloud2CAD**

**Автор:**

Студент Б05-103 группы  
Родригес Яновицкий Антонио Александрович

**Научный руководитель:**  
Пасечник Игорь Игоревич



Москва 2025

## **Аннотация**

Разработка системы автоматизированного проектирования  
PointCloud2CAD

CAD (Computer-Aided Design) широко применяется в инженерном проектировании. Однако современная «end-to-end» реализация промышленных проектов обходится дорого и требует значительных временных затрат. Например, создание 3D-модели автомобиля может занять до трёх месяцев и стоить около полумиллиона. Кроме того, на российском рынке ощущается нехватка квалифицированных специалистов в сфере 3D-инжиниринга. Все эти факторы указывают на высокую актуальность задачи генерации CAD-объектов, вызывающей интерес у многих компаний. На сегодняшний день эта задача не решена и остаётся сложной из-за нехватки больших наборов данных и отсутствия унифицированного формата представления.

В работе представлен новый синтетический датасет, позволяющий улучшить метрики state-of-the-art-решения CADReCODE (без учёта RL-дообучения). Я сравнил два кодовых представления CAD-моделей на специально собранном валидационном наборе, оценивая способность системы воспроизводить инженерные эскизы, и выбрал более эффективное. Также в статье описана модернизация генератора синтетических данных CADRecode путём интеграции скетчей, созданными людьми.

## **Abstract**

Разработка системы автоматизированного проектирования PointCloud2CAD

## Содержание

<b>1 Введение</b>	<b>4</b>
<b>2 Постановка задачи</b>	<b>7</b>
<b>3 Обзор существующих решений</b>	<b>8</b>
3.1 Эволюция представлений CAD-геометрии . . . . .	8
3.2 Разновидности входных данных . . . . .	8
3.3 Датасеты . . . . .	8
3.4 Синтетические данные и их ценность . . . . .	9
3.5 Метрики и валидационные сеты . . . . .	9
3.6 SOTA: CADRecode и <i>cadrille</i> . . . . .	9
<b>4 Исследование и построение решения задачи</b>	<b>10</b>
4.1 План решения и разбитие на подзадачи . . . . .	10
4.2 CADAxt — универсальный формат хранения CAD . . . . .	10
4.3 CADQuery и CADScript. . . . .	12
4.4 Подход CAD-Recode. . . . .	15
4.5 Процесс обучения. . . . .	15
4.6 Метрики и эксперимент. . . . .	16
4.7 Результаты и анализ. . . . .	19
4.8 Валидационный сет SketchGraph . . . . .	20
4.9 Тестирование экспериментов на SketchGraph . . . . .	24
4.10 Синтетический датасет на SketchGraph. . . . .	26
4.11 Будущая работа . . . . .	28
<b>5 Заключение</b>	<b>29</b>

## 1 Введение

В данной работе рассматривается задача генерации CAD-объектов из облаков точек. Хотя входными данными для модели могут служить как текстовое описание инженерной задачи, так и изображения или наброски, наибольший интерес представляет именно *point cloud*, поскольку в промышленных отраслях всё чаще применяется 3D-сканирование и обратная разработка (reverse engineering), позволяющие воссоздавать CAD-модели из облаков точек.

Важно уточнить, что CAD — это не просто 3D-модель: она задаётся последовательностью команд (*constructive sequence*), которые, в свою очередь, реализуются геометрическими движками. Набор команд в разных движках может отличаться, однако базовый список обычно включает:

1. Sketch — команда, позволяющая создавать 2D-эскизы, задавая контуры и элементы, пригодные для дальнейшего 3D-моделирования
2. Extrude — выдавливает (экструдирует) выбранный эскиз, превращая 2D-контуры в трёхмерную деталь заданной высоты
3. Fillet — стягивает или скругляет выбранные ребра для создания плавных переходов между поверхностями
4. Chamfer — срезает выбранные ребра под заданным углом и на заданное расстояние, формируя фаску
5. Mirror — копирует выбранную геометрию зеркально относительно заданной плоскости или плоскости симметрии
6. CircularPattern — создает массив (паттерн) из выбранных элементов, равномерно расположенных по окружности
7. Shell — превращает твердое тело в полую оболочку, задавая толщину стенок и удаляя выбранную грань
8. Sweep — выдавливает (протягивает) профиль вдоль заданного пути (траектории), создавая более сложные объёмные формы
9. Revolve — вращает 2D-профиль вокруг оси, формируя осесимметричное твердое тело

Поставленная задача имеет множество ограничений:

1. **Отсутствие унифицированного формата представления CAD-объектов.** Существует множество форматов представления CAD: зарубежные движки (Fusion360, Autodesk, Onshape) и российские (например, КОМПАС-3D) имеют собственные реализации и кодовые представления, которые зачастую не совместимы друг с другом.
2. **Нехватка данных и бенчмарков.** Несмотря на то, что инженерные модели повсеместно встречаются в различных отраслях, компании не публикуют в открытий доступ соответствующие датасеты, опасаясь утечки технологий и финансовых потерь. На сегодняшний день крупнейший датасет ABC содержит около 1 млн реальных образцов, чего недостаточно для обучения полноценной обобщающей модели. Кроме того, в нём невозможно извлечь *constructive sequence*, на

которых базируется всё обучение. По сути, наибольшим (но всё ещё ограниченным) датасетом реальных данных является DeepCAD размером в 270 тыс. образцов, включающий лишь операции *sketch-extrude*. Также отсутствуют бенчмарки, позволяющие проверить умения системы по отдельным сложным инженерным навыкам, например, по построению сложного скетча.

3. **Отсутствие конструктивных метрик.** Наиболее распространёнными метриками качества считаются *chamfer distance* (расстояние между облаками точек) и *intersection over union*, но они не отражают способность модели к более «инженерному» обобщению. Например, сложную модель можно с достаточной точностью аппроксимировать набором простых примитивов, что удовлетворит вышеуказанные метрики, однако это не продемонстрирует реальную способность модели использовать предусмотренные операции CAD.
4. **Слабые бейзлайны.** В настоящее время все модели для генерации CAD обучены лишь на датасетах, поддерживающих ограниченный набор операций (в основном *sketch-extrude*), и на промышленных данных показывают непригодные для реального применения результаты. По сути, не существует ни универсальной архитектуры, ни метода обучения, которые бы существенно превосходили существующие решения и позволяли сравнивать их напрямую, поскольку многие из существующих пайплайнов специфичны и мало сопоставимы друг с другом.

Перечисленные ограничения наглядно демонстрируют, что задача генерации CAD-объектов по облаку точек пока не решена и вряд ли будет закрыта в ближайшем будущем. Требуется разработка новых датасетов, бенчмарков и правил, подкреплённых высокинформативными метриками. В настоящей работе будут затронуты все четыре проблемы

Приведём список используемых в работе понятий с их определениями:

**CAD** Компьютерное проектирование (Computer-Aided Design), создание 2D/3D-моделей.

**Селектор** Инструмент или функция, позволяющая выбирать (выделять) объекты, элементы или параметры в модели или интерфейсе программы.

**Квантизация** Процесс уменьшения точности числовых данных (например, весов нейронной сети) для оптимизации вычислений.

**B-rep (Boundary Representation)** Границное представление, описание 3D-объекта через грани, рёбра и вершины.

**Constructive sequence (CS)** Конструктивная последовательность, история операций при моделировании в CAD.

**CSG (Constructive Solid Geometry)** Конструктивная блочная геометрия, построение сложных тел через булевые операции.

**Mesh** Полигональная сетка, представление 3D-модели в виде вершин, рёбер и граней.

**Point cloud** Облако точек, массив 3D-координат, описывающий форму объекта.

**Onshape** Облачная CAD-система для совместной работы.

**Synthetic data** Синтетические данные, искусственно сгенерированные для обучения ИИ.

**Reinforcement learning** Обучение с подкреплением, метод ИИ, основанный на наградах и штрафах.

**Finetuning** Дообучение модели под конкретную задачу с настройкой параметров.

**Pretrain** Предобучение, начальная тренировка модели перед дообучением.

**Sample** Образец данных (например, одно изображение в датасете).

**STEP (Standard for the Exchange of Product Data)** Стандартный формат обмена CAD-данными (ISO 10303).

**STL (Standard Tessellation Language)** Стандартный формат представления полигональных сеток.

**JSON (JavaScript Object Notation)** Текстовый формат обмена данными, основанный на структурах «ключ–значение».

**Python** Высокоуровневый язык программирования общего назначения.

**CadQuery** Python-библиотека для параметрического 3D-моделирования на базе OpenCASCADE.

**OpenCASCADE** Открытая CAD-библиотека ядра для 3D-моделирования и обработки геометрии.

**PythonOCC** Python-обёртка для OpenCASCADE, позволяющая использовать его функционал в Python.

## 2 Постановка задачи

Главной задачей является повысить метрики SOTA решения CADRecode. Этого я собираюсь добиться путем изменения train данных, а именно формата CAD представления и самого распределения 3D объектов для обучения. Именно поэтому я ставлю перед собой следующий задачи:

1. Определить какой формат представления CAD объекта лучше
2. На предпочтительном формате внедрить изменения в train данные, которые повысят метрики эксперимента

Таким образом задачи поставлены так, чтобы понять каким образом исходные данные влияют на работу модели и в итоге улучшить ее работу. В данном случае первое - это различные форматы представления CAD, второе - различные синтетические датасеты.

### 3 Обзор существующих решений

#### 3.1 Эволюция представлений CAD-геометрии

До появления глубоких моделей обратный инжиниринг CAD-объектов основывался на *детерминированных алгоритмах* преобразования сеток в CSG-деревья или B-rep. Классические работы InverseCSG [1] и GEOUNED [2] показали, что жёсткие эвристики плохо масштабируются на детали со сложной историей построения. Переломный момент наступил в 2018–2021 гг., когда начали появляться первые *нейронные* парсеры CSG (CSGNet [3]) и первые попытки учить трансформеры прямо на историях построения скетчей (SketchGraph [4]). Параллельно в сообществе шло обсуждение: какое представление (CSG, CS, B-rep) является наилучшим носителем семантики для обучения. Сегодня можно наблюдать следующую тенденцию:

- **CSG** остаётся привлекательным из-за компактности дерева, однако плохо описывает свободные формы и трудно конвертируется обратно в B-rep;
- **CS** даёт точное соответствие средам CAD, хорошо компилируется и позволяет производить редактирование. Именно это представление доминирует в современных SOTA-моделях (DeepCAD, CAD-SIGNet, CADRecode, *cadrille*);
- **B-rep** используется, когда цель — непосредственная генерация замкнутого контура (BrepGen [5], SolidGen [6]). Однако отсутствие явной истории построения затрудняет интерпретацию.

Таким образом, фокус сообщества смещается от «геометрии ради геометрии» (B-rep) к «программируемой геометрии» (CS/CSG), что напрямую связано с переходом к LLM-подходам.

#### 3.2 Разновидности входных данных

Входными данными для генерации CAD объекта могут выступать Text/Image/PointCloud. Давайте рассмотрим дендекции каждого направления:

1. **Image2CAD.** Ранние работы строились на CNN+GRU-декодере, сегодня лидируют VLM-гибриды: OpenECAD [7], Img2CAD [8], CAD-MLLM [9];
2. **PointCloud2CAD.** Линия эволюции начинается с DeepCAD, проходит через CAD-SIGNet [10], TransCAD [11] и выходит на CADRecode [12] и *cadrille* [13], где облако точек стало лишь одной из модальностей входа для LLM-декодера;
3. **Text2CAD.** Первые пробы пера (Query2CAD [14], Text2CAD [15]) показали, что GPT-подобные модели способны описывать простые детали, но требуют жёсткого *self-refine*. Современные решения предпочитают мультимодальный conditioning (*cadrille*).

#### 3.3 Датасеты

Ниже приведены ключевые публичные коллекции (табл. 1).

Выделяются два общих недостатка:

1. распределение операций далеко от равномерного (см. ContrastCAD [16]); большинство моделей — «игрушечные» примитивы без технологической семантики;
2. крупнейший набор — ABC — не содержит описания конструктивной последовательности, а остальные ограничиваются лишь операциями *sketch-extrude*.

Таблица 1: Крупнейшие открытые CAD-датасеты

Датасет	Количество моделей
ABC	1 000 000
DiffCAD	318 229
Fusion360	231 000
OpenECAD	200 000
DeepCAD	178 238

### 3.4 Синтетические данные и их ценность

Первый массовый синтетический набор ABC оказался достаточным, чтобы обучить классические CNN, но недостаточным для LLM. Проблема усилилась, когда задачи стали мультимодальными: невозможно одновременно получить согласованные point-cloud, рендеры и CAD-код из реального скана в требуемом объёме.

На этом фоне появляются генераторы *процедурной геометрии*. CAPRI-Net [17] и D<sup>2</sup>CSG [18] создают компактные CSG-деревья, однако не обеспечивают полного контроля над конструктивной последовательностью. CADRecode предлагает иной путь: **миллион моделей** с полной историей построения в виде исполнимого CadQuery-кода, что делает датасет *реплицируемым, расширяемым* и идеально подходящим для RL-обучения LLM. Именно этот набор лёг в основу SOTA-моделей CADRecode и *cadrille*.

### 3.5 Метрики и валидационные сети

Оценка качества CAD-реконструкции традиционно сводилась к Chamfer или IoU по мешу. С появлением компилируемого кода стало возможным проверять:

- *Topological accuracy* — совпадение графа «эскиз–операция»;
- *Compile rate* — успешность интерпретации кода в ядре OpenCascade;
- *Manufacturability* — пригодность к 3D-печати или обработке.

Тем не менее ни DeepCAD, ни Fusion360 не предоставляют аннотаций для подобных проверок. Современные работы начали публиковать собственные валидационные подмножества (CAD-SIGNet, CAD-MLLM), но их объём пока невелик. В литературе всё чаще звучит призыв к **единобразному бенчмарку**, учитывающему не только геометрию, но и потенциал редактирования кода.

### 3.6 SOTA: CADRecode и *cadrille*

CADRecode [12] показал, что LLM, обученная только на исходниках CadQuery, пре-восходит специализированные сети DeepCAD и MultiCAD по задаче PointCloud2CAD, однако модель оставалась одномодальной.

*cadrille* [13] делает следующий шаг: объединяет изображение, облако точек и текст в едином VLM-пространстве и дообучает декодер с помощью онлайн-RL (GRPO). Результат — новый SOTA на DeepCAD, Fusion360 и CC3D, подтверждающий гипотезу о том, что *богатая синтетика + RL на компилируемом коде дают максимальный прирост качества*.

Дальнейшие исследования, вероятно, будут сосредоточены на улучшении генераторов, унификации бенчмарков и расширении форматов кодового описания CAD.

## 4 Исследование и построение решения задачи

### 4.1 План решения и разбитие на подзадачи

Напомню, что для достижения цели я поставил перед собой следующие задачи:

1. Определить, какой формат представления CAD объекта лучше.
2. На предпочтительном формате внедрить изменения в train-данные, которые повышают метрики эксперимента.

**Зачем тестировать различные форматы.** На данный момент не существует исследований, которые сравнивали результаты генерации на различных кодовых представлениях *constructive-sequence*. Однако, хотя логика построения геометрии в каждом формате должна быть одинаковой, синтаксис и структура кода могут приводить к различиям в работе. Кроме того, в зависимости от способа представления меняется длина кода. Введение дополнительных переменных способно дать определённые преимущества при реализации 3D-операций, требующих селекторов, но может привести к ухудшению некоторых метрик и потере способности к реализации других операций.

Поиск «золотой середины», при которой кодовое представление обладает достаточной функциональностью и при этом остаётся коротким и синтаксически понятным, — важная задача в генерации CAD-объектов.

### 4.2 CADAxt — универсальный формат хранения CAD

Для начала отметим, что при тестировании различных представлений для CAD-моделей хочется реализовать промежуточный этап хранения данных. Это позволит впоследствии добавлять новые форматы представления при необходимости. Подобная идея также возникает из необходимости иметь реальные данные, получаемые путём парсинга публичных сайтов.

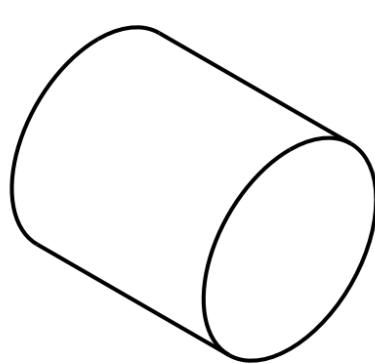
Таким образом, данный формат должен быть гибким для добавления дополнительных параметров и операций, в то время как поддержание их перевода в нужные форматы остаётся на этапе формирования датасета.

Подобную идею уже реализовали авторы статьи *DeepCAD*. В ней они распарсили данные с сайта Onshape в определённый формат JSON, структура которого была заимствована из датасета Fusion360. Их формат поддерживает только две операции: *Sketch* и *Extrude*.

Для внедрения других типов операций была внедрена ссылочная система на топологию, созданную при применении операции. Она необходима и является универсальным селектором для операций, требующих выбора топологии, например: *Fillet* (сглаживание ребра), *Chamfer* (резка ребра).

Формат, описывающий *constructive sequence*, получил название **CADAxt** и оформлен в виде библиотеки, доступной по ссылке: <https://github.com/axtonio/cadaxt>.

Однако в рамках настоящих экспериментов будут использоваться только операции *Sketch* и *Extrude*. Для простоты изложения детали реализации ссылочной системы будут опущены здесь. Рассмотрим на простом примере строение подобного JSON:



```
{
  "sequence": [
    {
      "type": "Sketch",
      "entity": "sketch1_id"
    },
    {
      "type": "ExtrudeFeature",
      "entity": "extrude1_id"
    }
  ],
  "entities": {
    "sketch1_id": {
      "profiles": {
        "profile1_id": {
          "loops": [
            {
              "is_outer": true,
              "profile_curves": [
                {
                  "curve": "curve1_id",
                  "center_point": {
                    "x": 0.0,
                    "y": -50.0,
                    "z": 0.0
                  },
                  "radius": 45.0,
                  "normal": {
                    "x": 0.0,
                    "y": 1.0,
                    "z": 0.0
                  },
                  "type": "Circle3D"
                }
              ]
            }
          ]
        }
      }
    },
    ...
  },
  "extrude1_id": {
    "reference_profiles": [
      {
        "parent": "sketch1_id",
        "id": "profile1_id"
      }
    ],
    "extent_one": {
      "distance": {
        "value": 100.0
      },
      ...
    }
  }
}
```

Рис. 1: Представление цилиндра в формате CADAxt.

Под ключом `sequence` идёт описание последовательности команд, где `type` — тип операции, а `entity` — уникальный `id` операции. Далее, под ключом `entities` хранится параметрическое описание каждой команды по соответствующему `id`. Некоторые параметры здесь специально опущены, так как они не потребуются в дальнейшем тексте.

Для понимания строения CAD-объекта обсудим параметры `Sketch`. Каждый скетч состоит из `loops` (замкнутых кривых) и `profiles`. Кривые (`curve`) бывают трёх типов: `Arc` (дуга), `Circle` (окружность) и `Line` (линия). Безусловно, в реальном CAD-пакете есть и другие типы кривых (например, В-сплайны), но в рамках данной работы мы ограничимся указанными тремя. Набор `Loop`, по которым можно однозначно определить ориентацию, формирует `Profile`. Валидный профиль всегда состоит из внешнего `Loop` и внутренних непересекающихся, причём внутренние задают отверстия во внешнем.

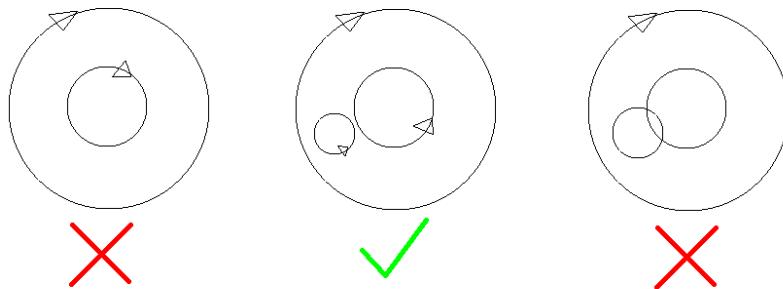


Рис. 2: Пример валидных и невалидных скетчей.

Координаты, описывающие скетч, заданы в трёхмерном пространстве относительно стандартной системы отсчёта. Однако, по сути, скетч — это 2D-операция, задаваемая на плоскости. При реализации `profile` может быть вычислена плоскость, относительно которой рисуется скетч, что даёт возможность отказаться от третьей координаты и задавать скетч параметрами плоскости и координатами в этой плоскости. В формате CADAxт для операции `Sketch` хранится параметр `transform`, указывающий плоскость, относительно которой рисуется скетч.

Теперь рассмотрим параметры экструдирования:

1. `profiles` — список профилей, к которым применяется экструд. Он однозначно задаётся путём указания `id` скетча и `id` нужного профиля.
2. `extent_one` — величина, на которую выдавливается скетч по направлению его нормали. Направление может меняться при наличии дополнительных параметров, специально опущенных здесь.

Таким образом, описанный формат даёт полноценное описание *constructive sequence* относительно операций `Sketch` и `Extrude`. Не рекомендуется использовать его напрямую для генерации, так как размер подобного файла может достигать нескольких тысяч символов. Однако он легко транслируется в любые иные форматы, что является существенным преимуществом при тестировании различных представлений.

#### 4.3 CADQuery и CADScript.

После создания унифицированного формата необходимо решить три основные задачи:

1. Определить, какие форматы представления CAD будут сравниваться.

2. Определить, на каких данных будет происходить обучение.
3. Выбрать архитектуру модели, подходящую для решения поставленной задачи.

Отвечая на первый вопрос, на ум сразу приходят форматы **CADQuery** и **PythonOCC**. **PythonOCC** — это низкоуровневый способ создания CAD-объектов, основанный на технологиях OpenCascade. Формат **CADQuery** представляет собой надстройку над **PythonOCC** и обеспечивает более удобный способ записи. По сути, **CADQuery** является набором макросов над **PythonOCC**, благодаря чему код одной и той же топологии получается более компактным именно в **CADQuery**. Тем не менее сравнивать эти два формата на равных основаниях некорректно, поэтому необходимо подобрать ещё один формат, который, с одной стороны, будет иметь более лаконичное представление, а с другой — позволит сопоставить его с **CADQuery**.

В литературе был предложен формат **OpenECAD** [7], в котором чётко описаны правила оформления и представления скетчей. Основные моменты, выделяемые в **OpenECAD**, следующие:

1. Моделирование в CAD обычно сводится к последовательности операций, приводящих к созданию твердотельных форм. В **OpenECAD** эти операции оформляются в виде структурированных команд.
2. Скетчи в **OpenECAD** задаются на плоскости и определяются с помощью примитивов: линий, дуг и окружностей, а также их параметров.
3. Для получения трёхмерных объектов в **OpenECAD** используется исключительно операция «скетч-экструзия», когда 2D-скетч преобразуется в 3D-форму за счёт протяжки.
4. В **OpenECAD** применяются читаемые обозначения операций, что облегчает понимание и использование кода.

**CADQuery** — это большая библиотека с богатым набором функций и селекторов. Однако у неё есть один недостаток: нет встроенного селектора, позволяющего выбирать рёбра, образованные конкретной операцией (например, экструзией). Такой селектор важен при генерации последовательности действий: если инженер в процессе проектирования видит текущее состояние 3D-модели и выбирает конкретное ребро, плоскость или точку, то генератору необходимо отслеживать, какие элементы были созданы на предыдущих шагах.

В **OpenECAD** формат описания CAD-объектов более избыточен, однако он удобен для реализации подобных селекторов. Из-за этой гибкости код в **OpenECAD** может получаться длиннее, чем в **CADQuery**.

На основе подхода из **OpenECAD** был разработан формат **CADScript**. Его главное отличие заключается в том, что он поддерживает дополнительные операции, не рассматриваемые в данной работе, а также предусматривает квантизацию.

Примеры кода для двух представлений показаны на рисунке 3.

**Использование CADRecode и связка с CADAxt.** Для решения второго и третьего вопросов в работе было выбрано решение **CADRecode** [12]. Его автор в своей статье разработал генератор синтетических данных, которые показали лучшие результаты на бенчмарках DeepCAD и Fusion360. Архитектура модели в **CADRecode** достаточно проста и быстро обучается, не требуя больших вычислительных ресурсов.

В рамках настоящей работы было принято решение встроить в генератор **CADRecode** конвейер перевода в универсальный формат **CADAxt**, из которого затем реализован

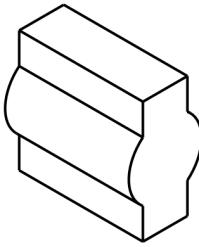
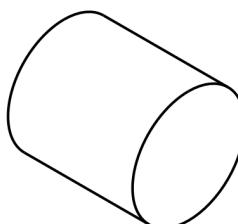
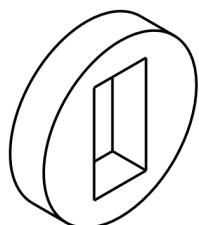
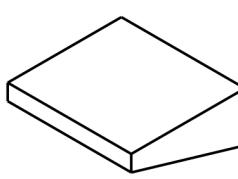
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <pre>CadQuery</pre> <pre><code>import cadquery as cq w0 = cq.Workplane("ZX", origin=(0, 50, 0)) r = (     w0.sketch()     .segment((49, -18), (-24, -18))     .arc((0, -30), (24, -18))     .segment((49, -18))     .segment((24, 18))     .arc((0, 30), (-24, 18))     .segment((49, 18))     .close()     .assemble()     .finalize()     .extrude(100) )</code></pre> </div> <div style="width: 45%;"> <pre>CadScript</pre> <pre><code>SketchPlane0 = add_sketchplane(     origin=[0., 50., 0.], normal=[0., 1., 0.], x_axis=[0., 0., 1.]) Loop0_0 = [] Curves0_0 = [] Line0_0_0 = add_line(start=[-49., -18.], end=[-24., -18.]) Arc0_0_1 = add_arc(start=[-24., -18.], end=[24., -18.], radius=31., big_angle=False) Line0_0_2 = add_line(start=[24., -18.], end=[49., -18.]) Line0_0_3 = add_line(start=[49., -18.], end=[24., 18.]) Line0_0_4 = add_line(start=[24., 18.], end=[-24., 18.], radius=31., big_angle=False) Line0_0_5 = add_line(start=[-24., 18.], end=[-49., 18.]) Line0_0_6 = add_line(start=[-49., 18.], end=[-49., -18.]) Loop0_0_7 = add_loop(Curves0_0) Profile0 = add_profile(Loop0_0, sketch_plane=SketchPlane0) Extrude0 = add_extrude(profile=Profile0,     operation=1, type=0, extent_one=-100., extent_two=0.)</code></pre> </div> </div>
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <pre>CadQuery</pre> <pre><code>w0=cq.Workplane('XY',origin=(0,0,50)) r=w0.workplane(offset=-100/2).box(42,16,100)</code></pre> </div> <div style="width: 45%;"> <pre>CadScript</pre> <pre><code>SketchPlane0 = add_sketchplane(     origin=[0., 0., 50.], normal=[0., 0., 1.], x_axis=[1., 0., 0.]) Loop0_0 = [] Curves0_0 = [] Line0_0_0 = add_line(start=[-21., -8.], end=[21., -8.]) Line0_0_1 = add_line(start=[21., -8.], end=[21., 8.]) Line0_0_2 = add_line(start=[21., 8.], end=[-21., 8.]) Line0_0_3 = add_line(start=[-21., 8.], end=[-21., -8.]) Loop0_0 = add_loop(Curves0_0) Profile0 = add_profile(Loop0_0, sketch_plane=SketchPlane0) Extrude0 = add_extrude(profile=Profile0,     operation=1, type=0, extent_one=-100., extent_two=0.)</code></pre> </div> </div>
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <pre>CadQuery</pre> <pre><code>import cadquery as cq w0=cq.Workplane('ZX',origin=(0,-50,0)) r=w0.workplane(offset=100/2).cylinder(100,45)</code></pre> </div> <div style="width: 45%;"> <pre>CadScript</pre> <pre><code>SketchPlane0 = add_sketchplane(     origin=[0., -50., 0.], normal=[0., 0., 1.], x_axis=[0., 0., 1.]) Loop0_0 = [] Curves0_0 = [] Circle0_0_0 = add_circle(center=[0., 0.], radius=45.) Loop0_0 = add_loop(Curves0_0) Profile0 = add_profile(Loop0_0, sketch_plane=SketchPlane0) Extrude0 = add_extrude(profile=Profile0,     operation=1, type=0, extent_one=100., extent_two=0.)</code></pre> </div> </div>
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <pre>CadQuery</pre> <pre><code>import cadquery as cq w0 = cq.Workplane("ZX", origin=(0, 11, 0)) r = w0.sketch().circle(50).rect(64, 34, mode="s").finalize().extrude(-22)</code></pre> </div> <div style="width: 45%;"> <pre>CadScript</pre> <pre><code>SketchPlane0 = add_sketchplane(     origin=[0., 11., 0.], normal=[0., 1., 0.], x_axis=[0., 0., 1.]) Loop0_0 = [] Curves0_0 = [] Circle0_0_0 = add_circle(center=[0., 0.], radius=50.) Loop0_0 = add_loop(Curves0_0) Curve0_1_0 = [] Line0_1_0_0 = add_line(start=[-32., -17.], end=[-32., -17.]) Line0_1_0_1 = add_line(start=[-32., -17.], end=[32., -17.]) Line0_1_0_2 = add_line(start=[32., -17.], end=[-32., -17.]) Line0_1_0_3 = add_line(start=[-32., 17.], end=[-32., -17.]) Loop0_1_0 = add_loop(Curves0_0) Profile0 = add_profile(Loop0_0, sketch_plane=SketchPlane0) Extrude0 = add_extrude(profile=Profile0,     operation=1, type=0, extent_one=-22., extent_two=0.)</code></pre> </div> </div>
	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <pre>CadQuery</pre> <pre><code>import cadquery as cq w0 = cq.Workplane("ZX", origin=(0, 50, 0)) r = (     w0.sketch()     .segment((-20, -46), (20, -46))     .segment((20, 46))     .segment((7, 46))     .close()     .assemble()     .finalize()     .extrude(-100) )</code></pre> </div> <div style="width: 45%;"> <pre>CadScript</pre> <pre><code>SketchPlane0 = add_sketchplane(     origin=[0., 50., 0.], normal=[0., 1., 0.], x_axis=[0., 0., 1.]) Loop0_0 = [] Curves0_0 = [] Line0_0_0 = add_line(start=[-20., -46.], end=[20., -46.]) Line0_0_1 = add_line(start=[20., -46.], end=[20., 46.]) Line0_0_2 = add_line(start=[20., 46.], end=[7., 46.]) Line0_0_3 = add_line(start=[7., 46.], end=[-20., -46.]) Loop0_0 = add_loop(Curves0_0) Profile0 = add_profile(Loop0_0, sketch_plane=SketchPlane0) Extrude0 = add_extrude(profile=Profile0,     operation=1, type=0, extent_one=-100., extent_two=0.)</code></pre> </div> </div>

Рис. 3: Примеры представлений CADQuery и CADScript.

трансфер в CADQuery и CADScript. После этого модель обучалась на одних и тех же облаках точек, но с различными кодовыми представлениями.

Генератор одного миллиона синтетических данных работал в течение 7,5 часов на 8 H100 используя 96 CPU. То есть скорость генерации одного семпла составляет 0.4/sec на одном ядре

#### 4.4 Подход CAD-Recode.

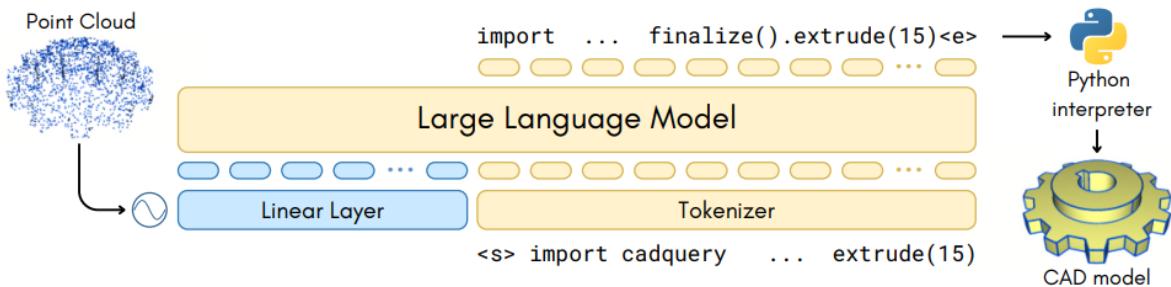


Рис. 4: Архитектура CAD-Recode

CAD-Recode опирается на предварительно обученных специалистов и их опыт работы с Python-кодом, дополняя их возможностями по обработке облаков точек, а также знаниями, специфичными для CAD-кода на Python. Как показано на рис. 4 в оригинальной статье, архитектура CAD-Recode состоит из двух частей: (1) проектора облака точек, преобразующего 3D-облако точек в обучаемые токены, и (2) предварительно обученного авторегрессионного CAD-декодера на базе LLM.

**Модуль проекции облака точек.** В CAD-Recode представлен лёгкий модуль проекции  $\Psi_P$ , который напрямую отображает плотное облако точек  $P \in \mathbb{R}^{n \times d_p}$  (где  $d_p = 3$  соответствует размерности координат) в последовательность из  $n_p$  токенов  $Q_p = [q_p^1, \dots, q_p^{n_p}] \in \mathbb{R}^{n_p \times d_q}$ , где  $d_q$  — размерность встраивания. Модуль проекции, обусловленный совместно с модулем CAD-декодера, состоит из трёх простых компонентов: (1) алгоритма выбора самых удалённых точек для уменьшения входного облака точек до  $n_p$ ; (2) позиционного кодирования координат методом Фурье; (3) линейного слоя, проецирующего закодированные координаты в  $Q_p$ .

**LLM в качестве CAD-декодера.** В качестве CAD-декодера кода, обозначаемого  $\Psi_{LLM}$ , выступает предварительно обученный LLM-специалист, приспособленный к задаче генерации кода CAD. В качестве базы берётся модель Qwen2-7b. На вход этому декодеру подаются токены  $Q_p$  от проектора облака точек, дополненные кодовыми токенами  $Q_t \in \mathbb{R}^{n_t \times d_q}$ , полученными после токенизации входного кода. Полная входная последовательность обозначается  $[Q_p; Q_t] \in \mathbb{R}^{(n_p+n_t) \times d_q}$ . Модель предсказывает следующую часть кодовой последовательности CAD, сопоставляя каждый предсказанный токен с символом из словаря  $\Sigma$  (буквенно-цифровые символы и операторы). Таким образом, CAD-Recode перенастраивает возможности моделирования LLM для задачи преобразования облаков точек в исполняемый CAD-код.

#### 4.5 Процесс обучения.

Стратегия обучения состоит из одного этапа. Модель работает с токенами запроса размерности  $d_q = 1536$  и обрабатывает изначальные облака точек в момент, когда их количество уменьшено до  $n_p = 256$ . К координатам входных точек добавляется гауссовский шум со средним ноль и стандартным отклонением 0.01 с вероятностью 0.5. Модель обучаются на процедурно сгенерированных CAD-кодах, где доступен набор функций CAD и методов проектирования, включённых в алгоритм генерации. Цель обучения — минимизация ошибки *Negative Log Likelihood* (NLL). При этом последовательность CAD-кода ограничивается специальными маркерами `<s>` и `<e>`. Модуль проекции облака точек изучает геометрические объекты с нуля, а предварительно обученный декодер донастраивается под задачу генерации CAD-кода.

## 4.6 Метрики и эксперимент.

Для оценки качества генерируемого CAD-кода в последовательностях создания эскизов использовались три основных метрики:

1. *Chamfer Distance* (CD).
2. *Intersection over Union* (IoU).
3. *Invalid Rate* (IR).

Результаты представлены как в виде среднего значения по выборке. Значение CD рассчитывалось с использованием 8192 точек и умножалось на  $10^3$ . Показатель IoU вычислялся на основе объединённых сеток CAD-моделей и выражался в процентах. IR указывает процент сгенерированных последовательностей, которые оказались некорректными для построения модели CAD.

Ниже приведены используемые формулы для перечисленных метрик.

**Chamfer Distance (CD).** Пусть  $X$  и  $Y$  — два множества точек, взятых равномерной выборкой с поверхностей двух 3D-моделей. Тогда метрика Chamfer Distance определяется как:

$$\text{CD}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|^2 + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} \|y - x\|^2.$$

В коде это реализуется путём выборки фиксированного числа точек  $n$  (в примере  $n = 8192$ ) с последующим вычислением среднеквадратичных расстояний от каждой точки одной модели до ближайшей точки другой модели.

**Intersection over Union (IoU).** Пусть  $A$  и  $B$  — объёмы (или сетки) двух 3D-моделей в пространстве. Тогда метрика IoU определяется как отношение объёма пересечения к объёму объединения:

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

В коде это достигается путём разбиения сеток на отдельные части, вычисления их пересечения и суммирования объёмов.

**Invalid Rate (IR).** Пусть имеется общее число сгенерированных последовательностей  $N$ , из которых  $M$  оказываются некорректными для построения 3D-модели. Тогда метрика IR вычисляется следующим образом:

$$\text{IR} = \frac{M}{N}.$$

Она показывает долю сгенерированных решений, которые не удалось корректно преобразовать в итоговую CAD-модель.

Обучение проводилось на различном количестве карт H100 и при разных размерах батча, поскольку длина кода в разных форматах отличается.

Эксперименты проводились на трёх тестовых наборах по 1000 моделей: DeepCAD, Fusion360, CC3D и CAD-Recode. Облака точек для DeepCAD и Fusion360 получены путём выборки точек на сетках, а набор CC3D содержит реальные 3D-сканы с шумом, сглаженными краями и отсутствующими деталями. Ниже на рисунке 19 10 представлены примеры семплов из каждого датасета

Таблица 2: Условия обучения

Format	H100	Batch size	Run time	Max code length	Epoch
CADQuery	4	9	3d	800	10
CADScript	6	6	3d17	2517	10

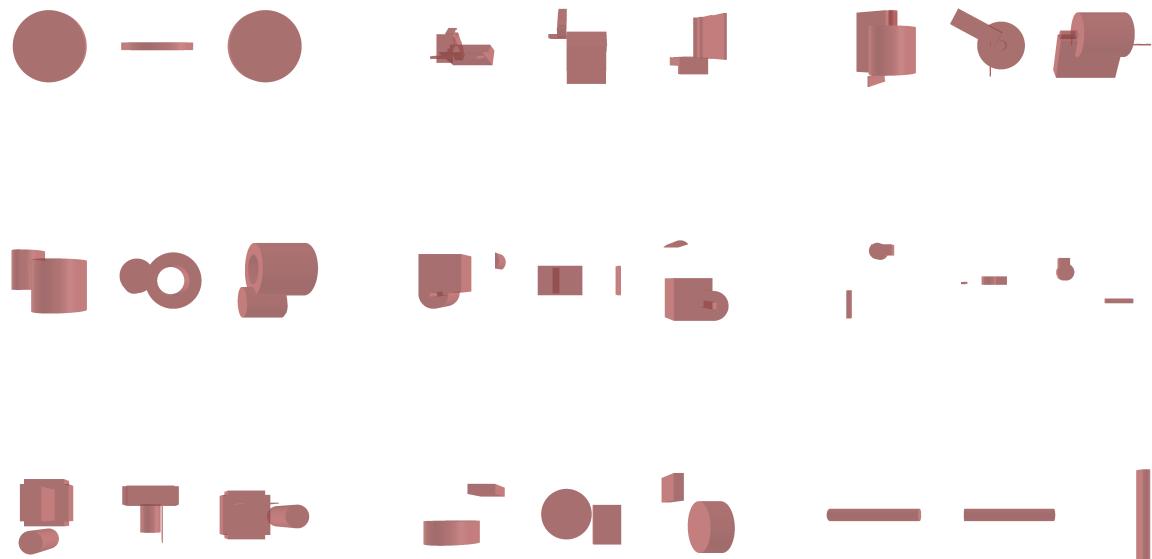


Рис. 5: CAD-Recode

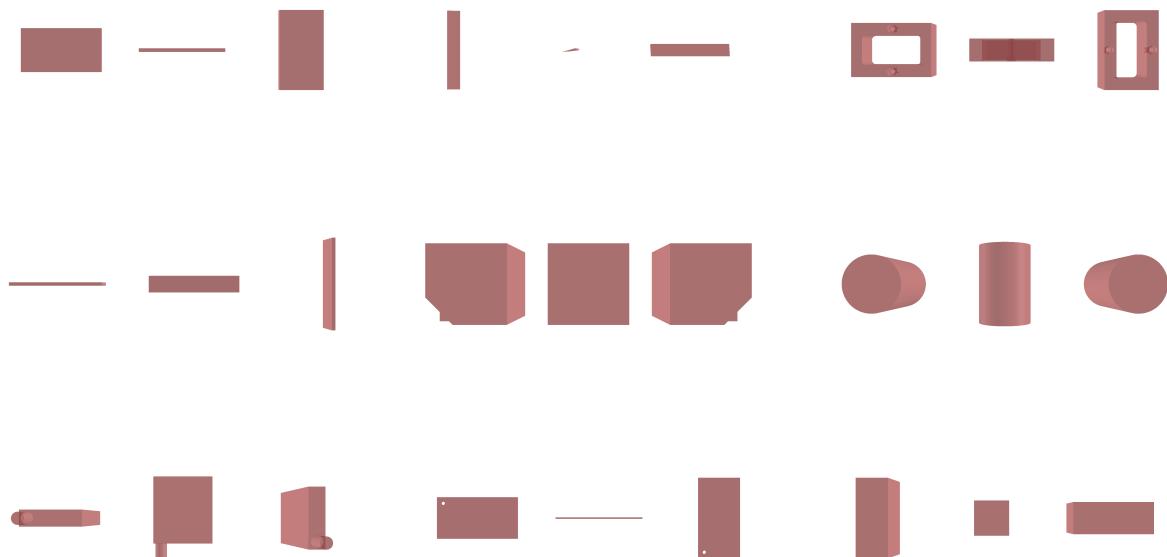


Рис. 6: DeepCAD

Рис. 7: Примеры данных из валидационных наборов CAD-Recode и DeepCAD

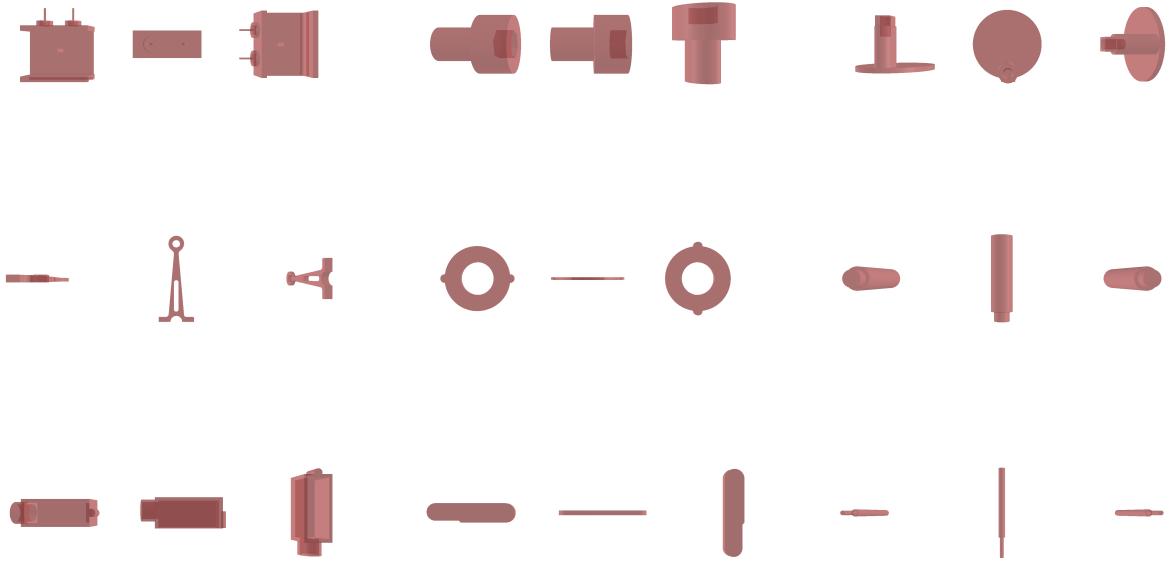


Рис. 8: Fusion360

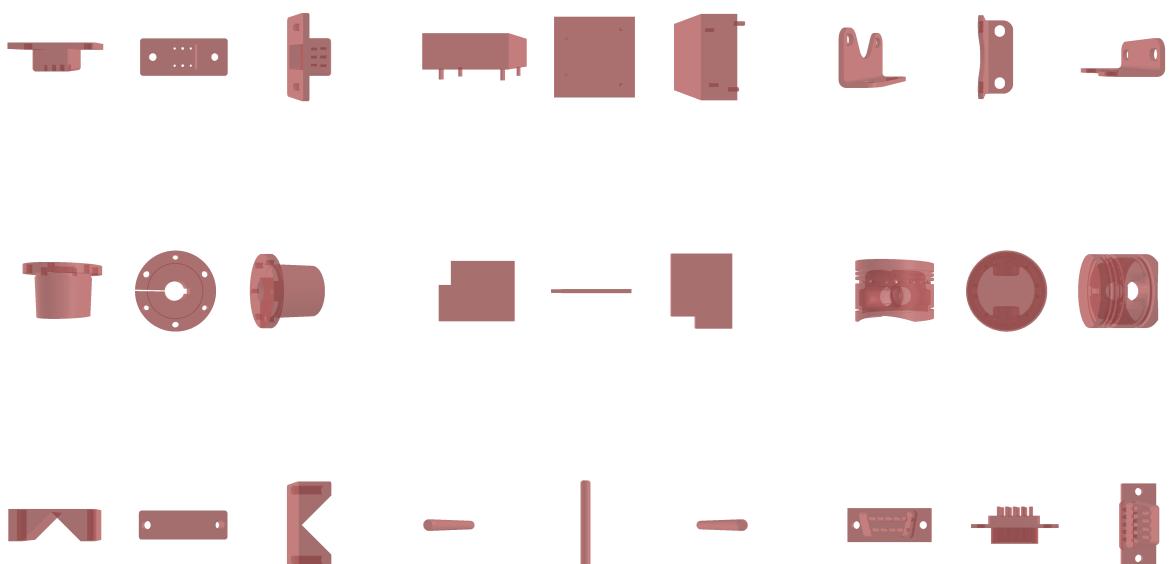


Рис. 9: CC3D

Рис. 10: Примеры данных из валидационных наборов Fusion360 и CC3D

## 4.7 Результаты и анализ.

Я проведу анализ каждой эпохи на вышеописанных датасетах, это позволит нам понять тренд обучения и проанализировать закономерности.

Сводные таблицы с метриками оказываются очень громоздкими и малоинформационными, поэтому анализ будет произведен на графиках.

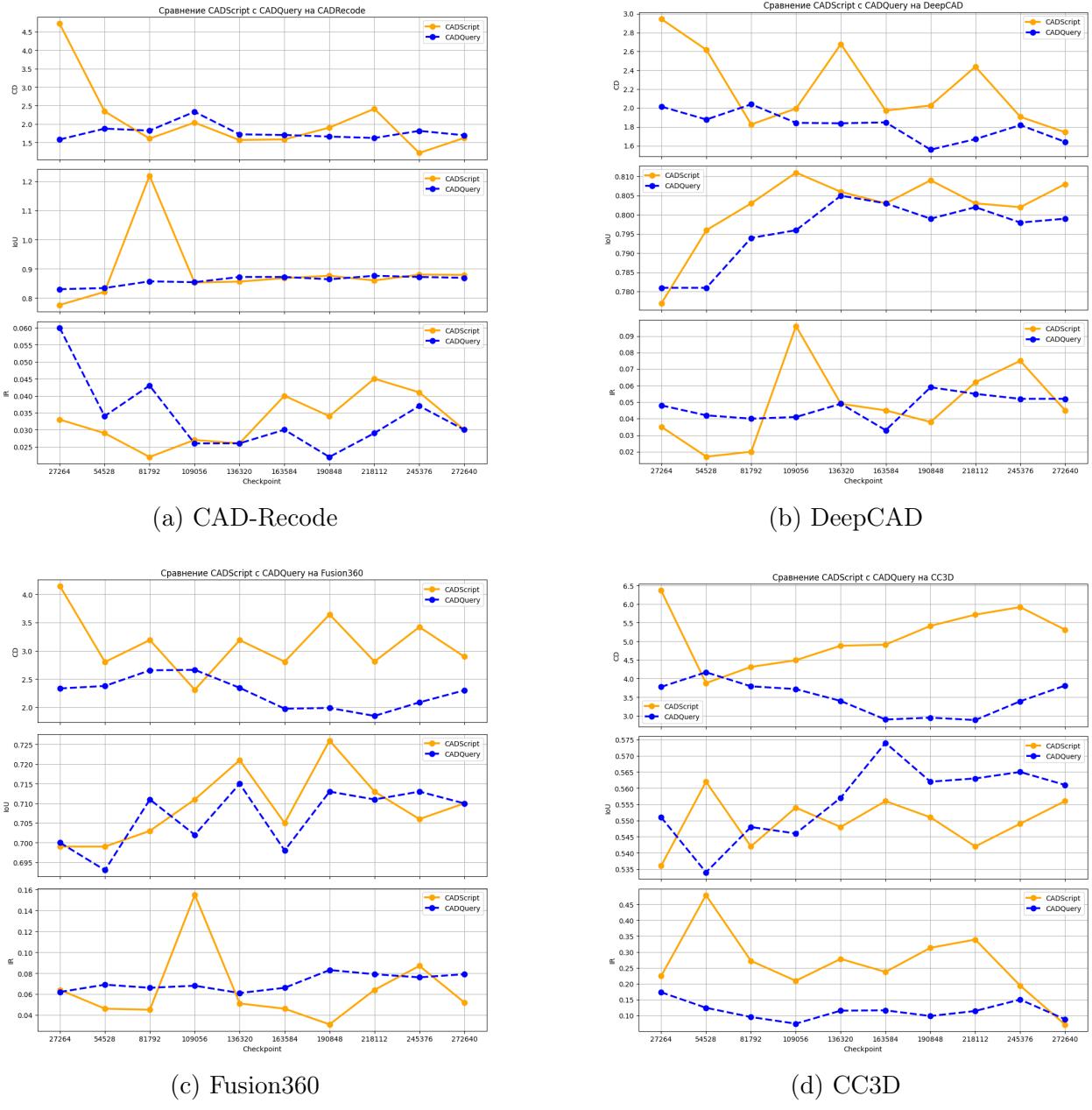


Рис. 11: Результаты экспериментов на CAD-Recode, DeepCAD, Fusion360 и CC3D. Верхний график относится к метрике Chamfer Distance (CD) и чем меньше, тем лучше. Второй график Intersection over Union (IoU) — чем больше, тем лучше. Последний относится к Invalid Rate (IR) — чем меньше, тем лучше. CD и IoU подсчитаны только на валидных генерациях.

Из графика валидации на CADRecode мы можем сделать вывод, что оба формата успешно выучивают train-данные и на синтетических данных дают по сути одинаковые результаты, разница между которыми крайне мала. На остальных графиках виден чёткий тренд: CADScript даёт всегда больше Chamfer Distance, чем CADQuery на реальных датасетах DeepCAD, Fusion360 и CC3D. По метрике Intersection over Union сделать

чёткого вывода нельзя: разница на Fusion360 и DeepCAD настолько мала, что больше похожа на дисперсию эксперимента, нежели на тренд, хотя на DeepCAD CADScript показал лучшие результаты. Однако на самом сложном датасете CC3D CADScript явно работает хуже.

По Invalid Rate также ничего конкретного нельзя утверждать, кроме как на датасете CC3D.

На самом деле именно CC3D показывает проигрыш CADScript перед CADQuery. Однако CADQuery тоже не показывает хорошие метрики на подобном датасете. Вероятно, проигрыш CADScript перед CADQuery на топологии такой высокой сложности заключается в отсутствии у CADScript базовых макросов для создания бокса или цилиндра, которые есть у CADQuery. И в то время, когда CADQuery приближает сложную топологию простыми примитивами, CADScript пытается сделать сложный loop. Также различия могут быть вызваны претройном Qwen на CADQuery, хотя вероятность этого невысока.

Однако если мы даже понимаем, что CADScript проигрывает CADQuery, мы не можем сказать конкретно, что именно он хуже делает. Так мы плавно переходим к проблеме, что отсутствуют метрики или валидационные сеты, которые бы оценивали способность модели к каким-то конкретным операциям.

#### 4.8 Валидационный сет SketchGraph

В статье Computer-Aided Design as Language [4] авторы описывают модель для генерации CAD-эскизов. Архитектура их решения не представляет интереса в рамках данного изложения, однако собранные ими данные оказываются практически полезны. Для формирования датасета исследователи использовали репозиторий документов, находящихся в открытом доступе на платформе Onshape (Onshape Developers). Результатирующий набор данных представлен в виде последовательностей токенов, описывающих геометрические примитивы и отношения между ними (constraints). Это ещё один формат представления скетчей, дополнительно к формату profile-loop.

Конструкции constraints позволяют описывать сложные эскизы значительно меньшим количеством символов. На рисунке 12 приведены примеры MirrorConstraint (зеркализование), TangentConstraint (касательная), OrthogonalConstraint (перпендикуляр) и CoincidentConstraint (совпадение). Благодаря им, даже без явного указания координат, можно реализовать столь сложную фигуру, как сердце.

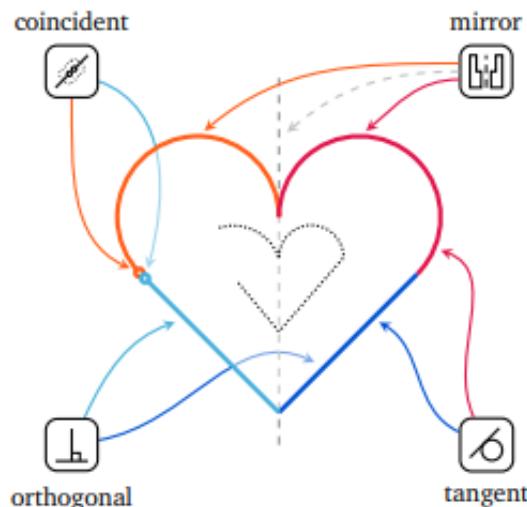


Рис. 12: Примеры constraints

Однако не все скетчи оказываются валидными для применения операции Extrude. На рисунке 13 видно, что в вариантах 13а и 13б нельзя однозначно определить, что именно является вырезом из внешнего контура.

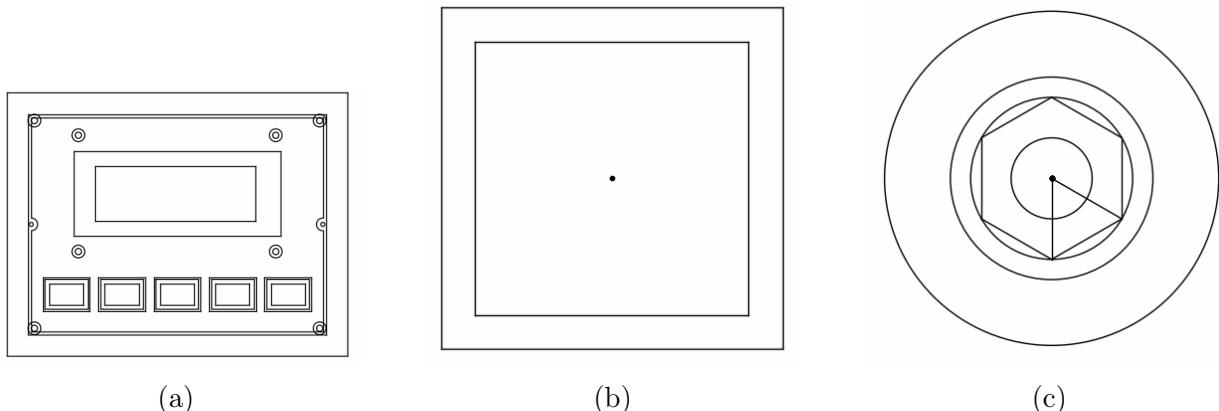


Рис. 13: Примеры скетчей из SketchGraph

Чтобы решить эту проблему, можно взять все возможные замкнутые контуры (loop) в SketchGraph и выделить их подмножества, которые образуют валидный profile. Затем остаётся лишь перевести constraints в парадигму loop-profile, и мы получим валидные скетчи, которые можно сохранить в формате CADAxt.

Благодаря интеграции CADAxt с синтетическим генератором возможно выполнить экструдирование полученных эскизов. Таким образом формируется набор 3D-моделей, которые получены при помощи рисования скетчей различной сложности с последующей операцией Extrude.

Эти 3D-модели разделяют на три класса по уровню сложности:

1. Простые: количество граней в интервале (5, 15].
2. Средние: количество граней в интервале (15, 30].
3. Сложные: количество граней в интервале (30, 50].

В итоге получается набор 3D-моделей, созданных одной операцией Extrude и разделенных на три класса. На рисунке 14 показан пример такого набора: слева направо приведено по 9 представителей каждого класса от сложных к простым.

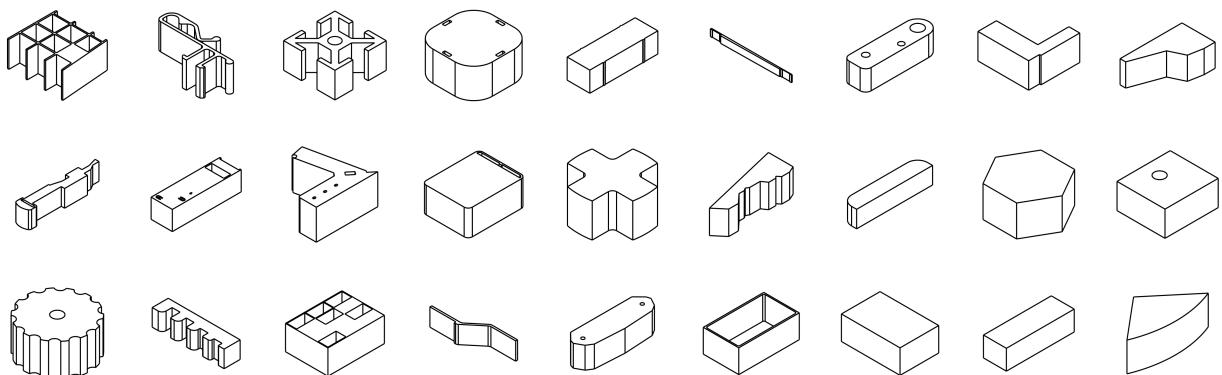


Рис. 14: Валидационный сет SketchGraph: примеры 3D-моделей, распределённых по сложности (сложные, средние, простые).

Основная задача данного валидационного набора:

1. Проверить способность модели воспроизводить представленные 3D-модели за одну операцию Extrude.
2. Оценить, как модель справляется с топологиями разной сложности.

**Сравнение CADScript и CADQuery на SketchGraph.** Теперь можно проверить, насколько эффективно наши два представления (CADScript и CADQuery) способны рисовать скетчи.

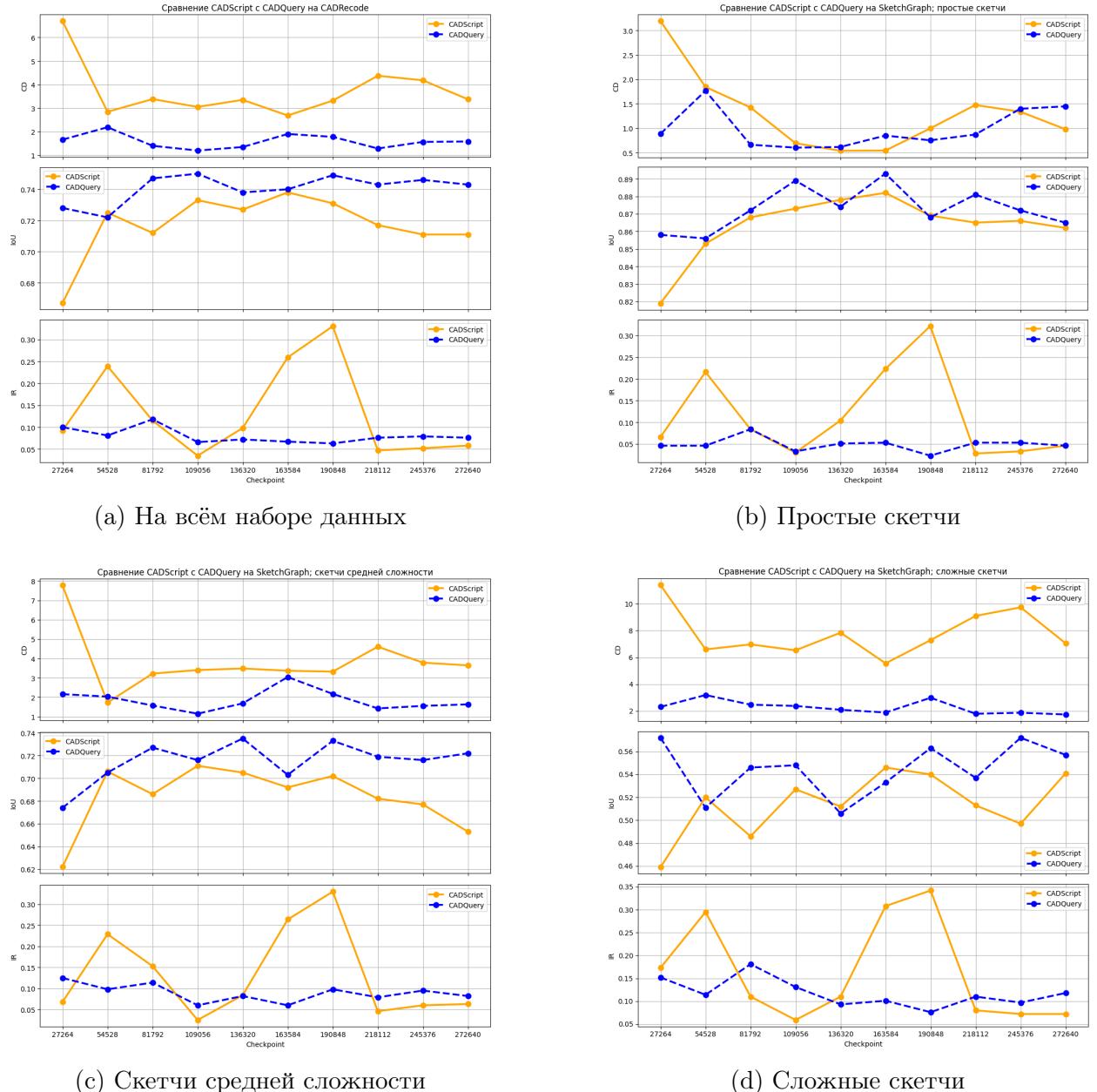


Рис. 15: Результаты экспериментов на SketchGraph и на трёх классах по уровню сложности топологии. Верхний график иллюстрирует метрику Chamfer Distance (CD), где меньшее значение лучше. Второй график — Intersection over Union (IoU), где большее значение лучше. Третий график — Invalid Rate (IR), где меньшее значение лучше. Метрики CD и IoU подсчитаны только на валидных генерациях.

Из графиков (рис. 15) хорошо видно, что на простых скетчах результаты CADScript и CADQuery сопоставимы. Однако на средних и сложных скетчах CADScript уступает

CADQuery по метрикам CD и IoU, а показатель Invalid Rate сильно колеблется. Может возникнуть мысль, что CADQuery просто приближает топологию (за счёт большого числа простых команд) и поэтому показывает более высокие результаты, тогда как CADScript действительно старается точно воспроизводить скетчи. Или же CADQuery всегда использует простейшие макросы (cylinder/box) и, благодаря этому, повышает свои показатели.

Чтобы разобраться в этом, введём две дополнительные метрики:

1. Extrude Mean (EM) — среднее количество операций extrude.
2. Base Extrude Ratio (BER) — отношение числа простейших макросов (cylinder и box) у CADQuery ко всем его командам extrude.

По этим метрикам получены следующие результаты:

Из рис. 16 видно, что CADQuery и CADScript в среднем используют сопоставимое число команд extrude, близкое к единице. При этом доля простейших макросов у CADQuery не превышает 9 %, то есть CADQuery не злоупотребляет такими командами. Следовательно, CADQuery действительно лучше справляется с задачей рисования скетчей, чем CADScript. Причиной этому может служить то, что в CADScript требуется явное задание описания Loop, в то время как в CADQuery оно происходит неявно.

Таким образом, формат CADQuery предпочтительнее формата CADScript: на реальных данных он даёт меньшие значения метрики CD и лучше справляется с задачей рисования скетчей.

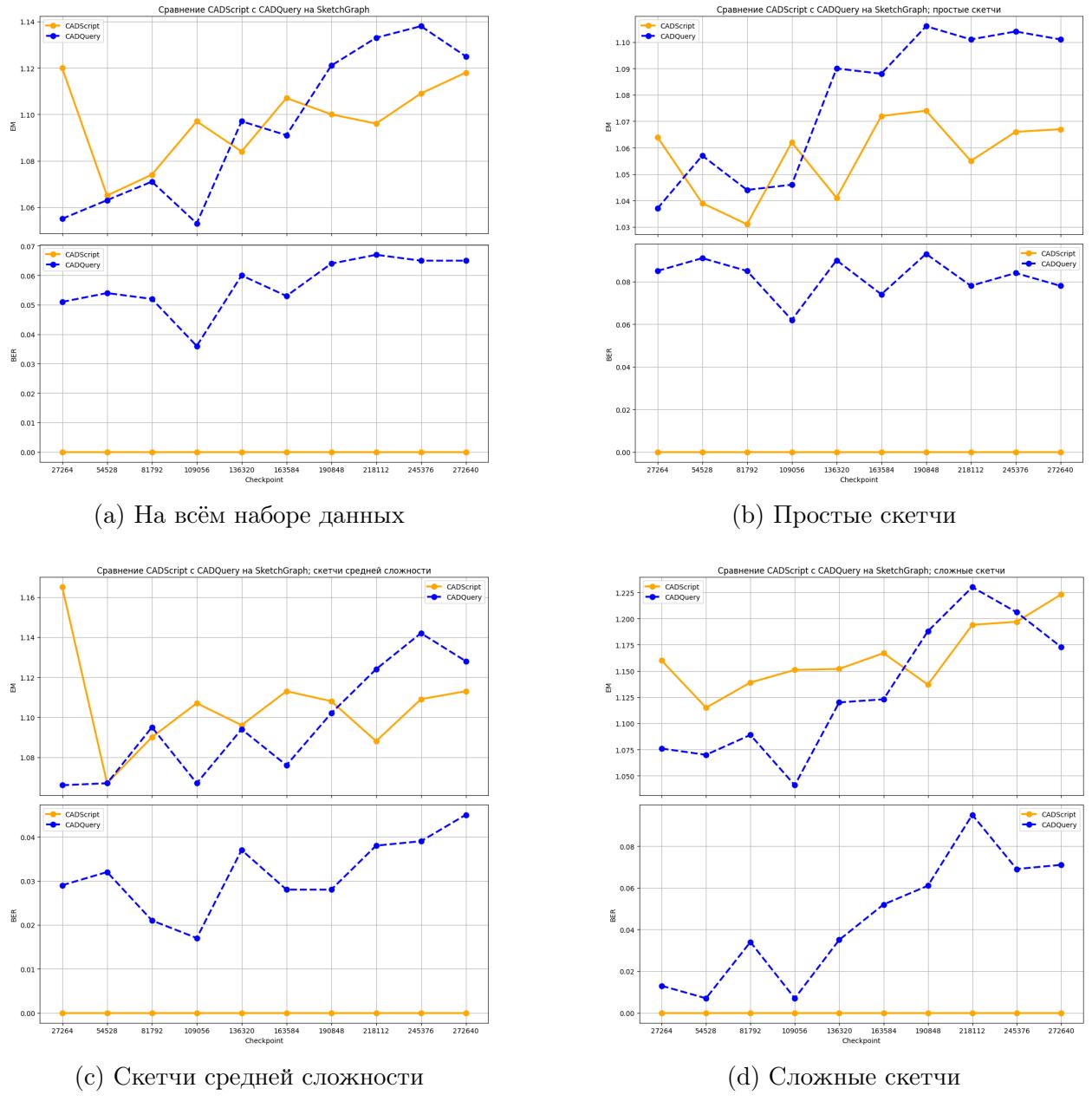


Рис. 16: Результаты по метрикам Extrude Mean (EM) и Base Extrude Ratio (BER) на SketchGraph и на трёх классах по уровню сложности топологии. EM (верхний график) показывает среднее число команд extrude (чем меньше, тем лучше). BER (нижний график) отражает долю простейших макросов (cylinder или box) у CADQuery; желательно, чтобы она не была слишком большой.

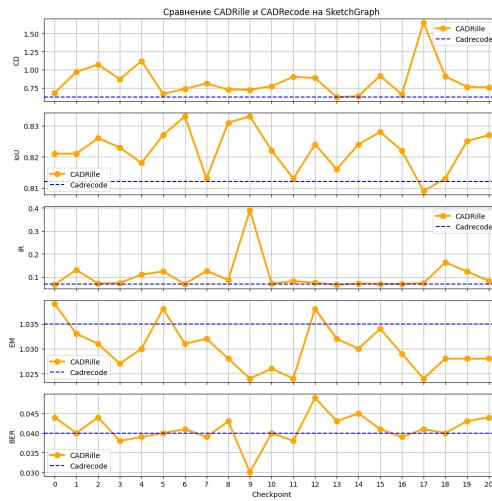
#### 4.9 Тестирование экспериментов на SketchGraph

Набор данных SketchGraph служит удобным средством для валидации и сравнения новых архитектур. Авторы работы CADRille [13] применили дообучение с подкреплением (RL finetune) к уже обученной модели, основанной на данных DeepCAD и Fusion360. Для этого они использовали метод GRPO, функция потерь которого описывается соответствующим лоссом. Их основной целью было минимизировать показатель Invalid Rate (IR), не снижая метрику Intersection over Union (IoU). Введённая ими функция вознаграждения (reward) предполагала штраф в  $-10$  за несгенерированный сэмпл и вознаграждение в размере  $+iou \times 10$  за успешно сгенерированный.

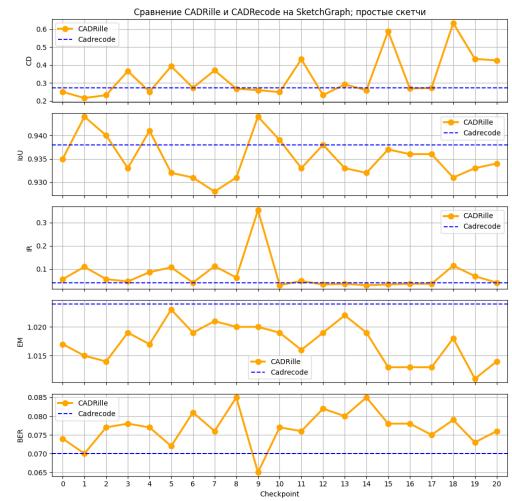
Хотя такая схема действительно улучшила результаты на датасетах DeepCAD и

Fusion360, концептуально она может привести к поведению, при котором модель старается генерировать упрощённую топологию и постепенно «доращивать» её до итога (напоминающее «покадровое» приближение). При таком подходе модель не получает штрафов за IR и постепенно учится повышать IoU. Поэтому важно проверить, что при таком дообучении модель не утрачивает способность корректно рисовать сложные скетчи.

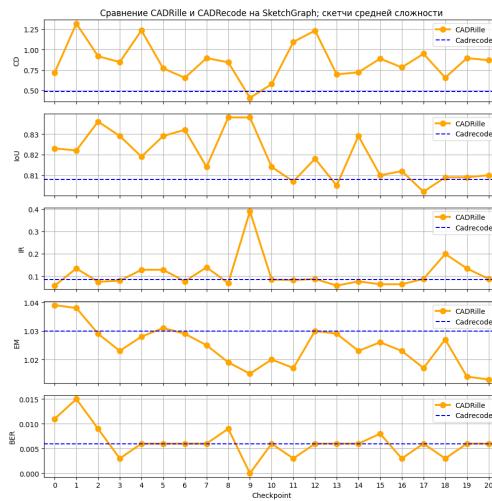
В своих экспериментах я обучал модель CADRecode по схеме RL finetune с использованием GRPO, как в работе авторов CADRillе [13]. Обучение длилось 4 дня на 8 GPU H100 на подвыборке из 10 000 семплов (20 эпох). Итоговые результаты представлены на рисунке 17.



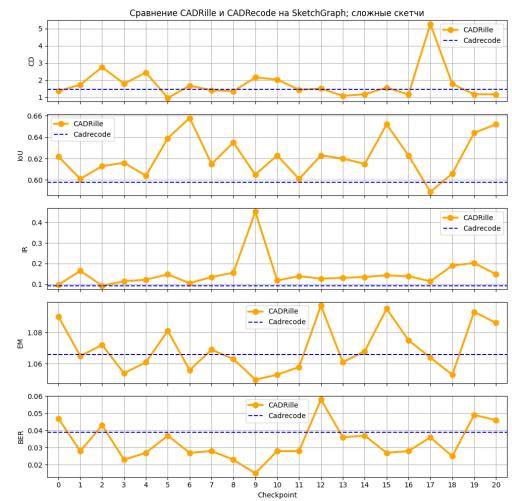
(a) На всём наборе данных



(b) Простые скетчи



(c) Скетчи средней сложности



(d) Сложные скетчи

Рис. 17: Результаты по метрикам CADRille с моделью CADRecode на SketchGraph и на трёх классах, разделённых по уровню сложности топологии. Верхняя диаграмма иллюстрирует метрику Chamfer Distance (CD), где меньшее значение лучше. Вторая диаграмма показывает Intersection over Union (IoU), где большее значение лучше. Третья диаграмма отражает показатель Invalid Rate (IR), где меньшее значение лучше. Метрики CD и IoU вычисляются только на валидных генерациях. EM (верхняя диаграмма) — это среднее число команд extrude (чем оно меньше, тем лучше). BER (нижняя диаграмма) — доля простейших макросов (cylinder или box) в среде CADQuery; желательно, чтобы это значение не было слишком высоким.

Третья диаграмма отражает показатель Invalid Rate (IR), где меньшее значение лучше. Метрики CD и IoU вычисляются только на валидных генерациях. EM (верхняя диаграмма) — это среднее число команд extrude (чем оно меньше, тем лучше). BER (нижняя диаграмма) — доля простейших макросов (cylinder или box) в среде CADQuery; желательно, чтобы это значение не было слишком высоким.

По результатам можно увидеть, что у CADRille растёт метрика IoU, а на скетчах лёгкого и среднего уровня наблюдается снижение количества команд Extrude. Таким образом, подход CADRille не ухудшает способность CADRecode корректно рисовать скетчи, при этом повышая обобщающую способность модели.

#### 4.10 Синтетический датасет на SketchGraph.

После интеграции скетчей из SketchGraph в пайплайн генератора CADRecode появилась возможность сгенерировать новый датасет и обучить на нём модель, сравнив результаты с датасетом на синтетических скетчах. Предполагается, что такой подход внесёт дополнительный индуктивный байес и улучшит метрики на всех датасетах.

Для эксперимента было сгенерировано два датасета на пайплайне генератора CADRecode [1], различающихся только типом скетчей. Обучение на каждом из них проходило на четырёх GPU H100 с размером батча 9 в течение 10 эпох. Время обучения обоих вариантов оказалось одинаковым.

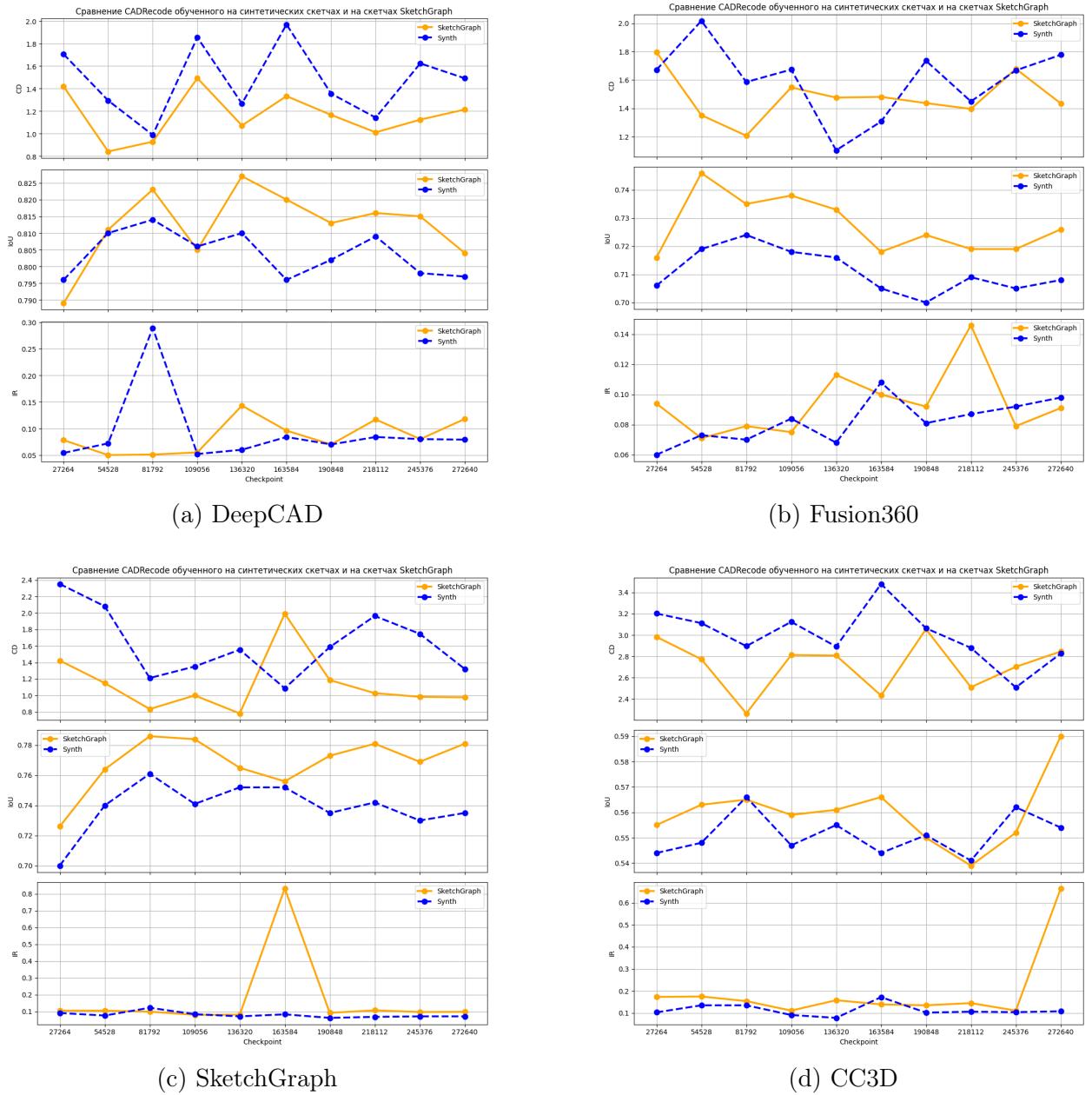


Рис. 18: Результаты экспериментов на DeepCAD, Fusion360, SketchGraph и CC3D. Верхний график показывает метрику Chamfer Distance (CD), где меньшие значения лучше. Второй график — Intersection over Union (IoU), где большие значения лучше. Последний график — Invalid Rate (IR), где меньшие значения лучше. CD и IoU подсчитаны только среди валидных генераций.

На рис. 18 заметно, что модель, обученная на скетчах из SketchGraph, чаще демонстрирует более высокий IoU и меньший CD, что говорит об успешной интеграции данного подхода. Сравним два лучших чекпоинта: модель на 81792-й итерации (скетчи из SketchGraph) и на 136320-й итерации (синтетические скетчи).

Из таблиц видно, что модель, обученная на реальных скетчах, особенно хорошо проявляет себя на датасетах DeepCAD и SketchGraph. Вероятно, более сложные скетчи

Модель	CD	IoU	IR
SketchGraph	1.207	<b>0.735</b>	0.079
Synth	<b>1.104</b>	0.716	<b>0.068</b>

Таблица 3: Результаты измерений на Fusion360.

Модель	CD	IoU	IR
SketchGraph	<b>0.928</b>	<b>0.823</b>	<b>0.051</b>
Synth	1.266	0.810	0.060

Таблица 4: Результаты измерений на DeepCAD.

Модель	CD	IoU	IR
SketchGraph	<b>2.265</b>	<b>0.565</b>	0.154
Synth	2.892	0.555	<b>0.078</b>

Таблица 5: Результаты измерений на CC3D.

Модель	CD	IoU	IR
SketchGraph	<b>0.836</b>	<b>0.786</b>	0.099
Synth	1.556	0.752	<b>0.071</b>

Таблица 6: Результаты измерений на SketchGraph.

из Fusion360 и CC3D затрудняют процесс генерации, однако на DeepCAD и SketchGraph способность к более сложному рисованию становится ключевым фактором качества.

С другой стороны, на Fusion360 и CC3D нельзя однозначно утверждать превосходство одной модели над другой, в частности из-за отличий в Invalid Rate (IR).

Рассмотрим визуальное сравнение на датасете CC3D (рис. 19).

На данных примерах модель, обученная на реальных скетчах, лучше улавливает симметрию и демонстрирует аккуратные результаты, которые выглядят более редактируемыми. В целом модель стала работать лучше, хотя систематически имеет более высокий Invalid Rate. Вероятно, при применении RL-подхода (например, CADRille [13]) этот недостаток можно компенсировать и улучшить метрики ещё заметнее.

#### 4.11 Будущая работа

В данной работе используется генератор CADRecode [12], однако я веду разработку собственного генератора, который будет ещё больше приближен к инженерным объектам за счёт новых правил построения. К сожалению, на данный момент мой генератор имеет недоработки, из-за чего обучение не проходит успешно.

Также планируется внедрить изменения в архитектуру PointCloud-модуля в CADRecode, а именно протестировать на различных *PointCloudEncoder* и подобрать оптимальное количество семплируемых точек для генерации.

Кроме того, планируется поставить эксперименты с RL-finetune на синтетике со скетчами SketchGraph, а также провести эксперименты с другой reward-функцией.

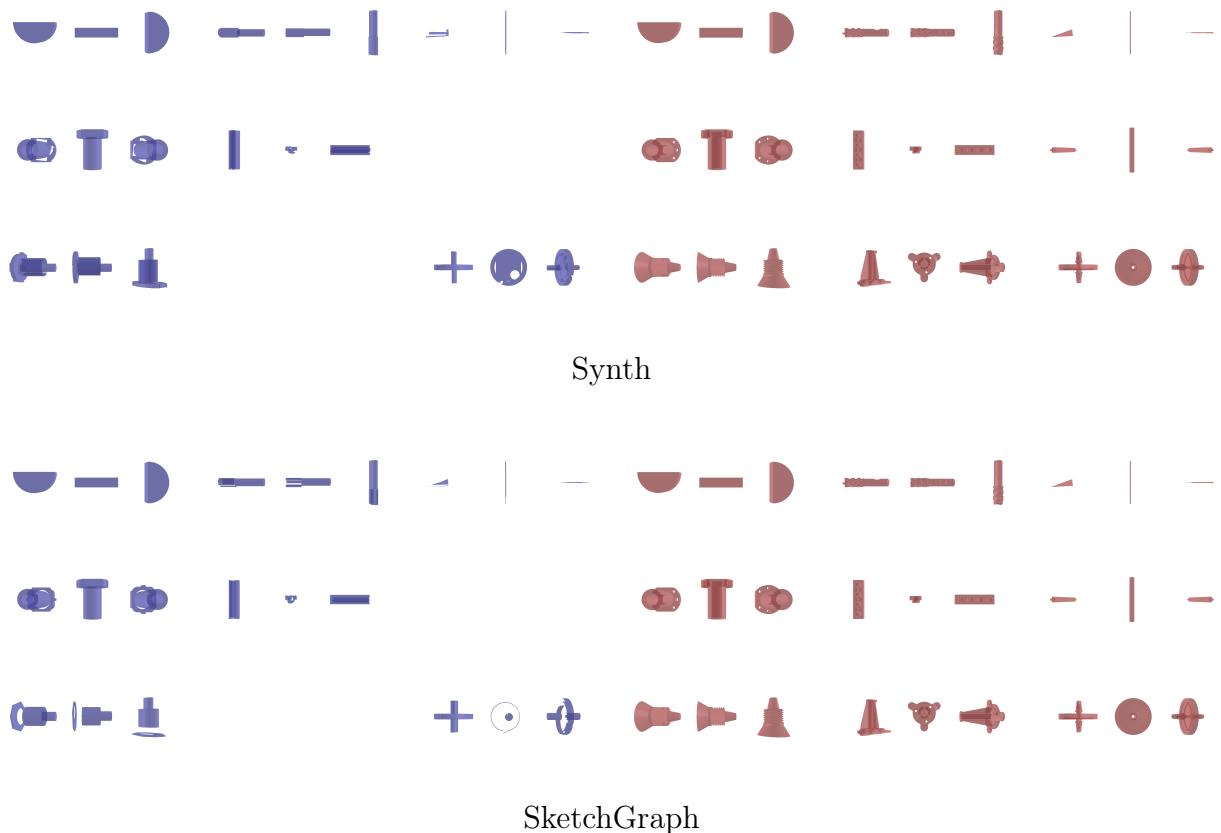


Рис. 19: Визуальное сравнение результатов на CC3D.

## 5 Заключение

В ходе данной работы был сформирован валидационный набор скетчей на основе библиотеки SketchGraph, что позволило провести сравнительный анализ двух форматов представления данных: CADScript и CADQuery. Также был создан синтетический датасет, основанный на инженерных скетчах, и проведён эксперимент, в котором результаты, полученные с новым датасетом, сравнивались с исходными синтетическими данными.

Проведённый эксперимент продемонстрировал улучшение метрик CD и IoU при некотором увеличении показателя IR. Тем не менее, визуальный анализ подтверждает возрастание качества генерируемых скетчей, что указывает на эффективность предложенного подхода и его положительное влияние на работу модели.

## Список литературы

- [1] *Inala, Jeevana Priya.* InverseCSG: Automatic Conversion of 3D Models to CSG Trees / Jeevana Priya Inala et al. // Proc. SIGGRAPH Asia. — 2018.
- [2] GEOUNED: A New Conversion Tool from CAD to Monte Carlo Geometry / J. P. Catalán, P. Sauvan, J. García et al. // Ann. Nucl. Energy. — 2024. — <https://www.sciencedirect.com/science/article/pii/S1738573324000548>.
- [3] *Sharma, Gopal.* CSGNet: Neural Shape Parser for Constructive Solid Geometry. — <https://arxiv.org/abs/1712.08290>. — 2018.
- [4] *Ganin, Yaroslav.* Computer–Aided Design as Language. — <https://arxiv.org/abs/2105.02769>. — 2021.
- [5] *Xu, Xiang.* BrepGen: A B–Rep Generative Diffusion Model with Structured Latent Geometry. — <https://arxiv.org/abs/2401.15563>. — 2024.
- [6] *Jayaraman, Pradeep Kumar.* SolidGen: An Autoregressive Model for Direct B–Rep Synthesis. — <https://arxiv.org/abs/2203.13944>. — 2022.
- [7] *Yuan, Zhe.* OpenECAD: An Efficient Visual Language Model for Editable 3D–CAD Design. — <https://arxiv.org/abs/2406.09913>. — 2024.
- [8] *Uy, Mikaela Angelina.* Img2CAD: Reverse Engineering 3D CAD Models from Images through VLM–Assisted Conditional Factorization. — <https://arxiv.org/abs/2408.01437>. — 2024.
- [9] *Xu, Jingwei.* CAD–MLLM: Unifying Multimodality–Conditioned CAD Generation with MLLM. — <https://arxiv.org/abs/2411.04954>. — 2024.
- [10] *Dupont, Elona.* CAD-SIGNet: CAD Language Inference from Point Clouds Using Layer–Wise Sketch Instance Guided Attention. — <https://arxiv.org/abs/2402.17678>. — 2024.
- [11] *Dupont, Elona.* TransCAD: A Hierarchical Transformer for CAD Sequence Inference from Point Clouds. — <https://arxiv.org/abs/2407.12702>. — 2024.
- [12] *Rukhovich, Danila.* CAD–Recode: Reverse Engineering CAD Code from Point Clouds. — <https://arxiv.org/abs/2412.14042>. — 2024.
- [13] *Kolodiazhnyi, Maksim.* cadrille: Multi–Modal CAD Reconstruction with Online Reinforcement Learning. — <https://arxiv.org/abs/2505.22914>. — 2025.
- [14] *Badagabettu, Akshay.* Query2CAD: Generating CAD Models Using Natural Language Queries. — <https://arxiv.org/abs/2406.00144>. — 2024.
- [15] *Khan, Mohammad Sadil.* Text2CAD: Generating Sequential CAD Models from Beginner–to–Expert Level Text Prompts. — <https://arxiv.org/abs/2409.17106>. — 2024.
- [16] *Jung, Minseop.* ContrastCAD: Contrastive Representation Learning for Computer–Aided Design Models. — <https://arxiv.org/abs/2404.01645>. — 2024.

- [17] *Yu, Fenggen.* CAPRI-Net: Learning Compact CAD Shapes with Adaptive Primitive Assembly / Fenggen Yu, Zhiqin Chen // Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR). — 2021.
- [18] *Yu, Fenggen.* D<sup>2</sup>CSG: Unsupervised Learning of Compact CSG Trees with Dual Complements and Dropouts / Fenggen Yu // Advances in Neural Information Processing Systems (NeurIPS). — 2023.
- [19] *Alrashedy, Kamel.* Generating CAD Code with Vision–Language Models for 3D Designs. — <https://arxiv.org/abs/2410.05340>. — 2024.
- [20] *Du, Changyu.* Text2BIM: Generating Building Models Using a Large Language Model–Based Multi–Agent Framework. — <https://arxiv.org/abs/2408.08054>. — 2024.
- [21] *Fayolle, Pierre-Alain.* A Survey of Methods for Converting Unstructured Data to CSG Models. — <https://arxiv.org/abs/2305.01220>. — 2023.
- [22] *Wu, Sifan.* CadVLM: Bridging Language and Vision in the Generation of Parametric CAD Sketches. — <https://arxiv.org/abs/2409.17457>. — 2024.
- [23] *Liu, Yujia.* Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds. — <https://arxiv.org/abs/2312.04962>. — 2023.
- [24] *Gao, Daoyi.* DiffCAD: Weakly–Supervised Probabilistic CAD Model Retrieval and Alignment from an RGB Image. — <https://arxiv.org/abs/2311.18610>. — 2023.
- [25] *Zhu, Chenming.* LLaVA-3D: A Simple Yet Effective Pathway to Empowering LMMs with 3D–Awareness. — <https://arxiv.org/abs/2409.18125>. — 2024.
- [26] *Qiu, Zeju.* Can Large Language Models Understand Symbolic Graphics Programs? — <https://arxiv.org/abs/2408.08313>. — 2024.
- [27] *Doris, Anna C.* DesignQA: A Multimodal Benchmark for Evaluating Large Language Models’ Understanding of Engineering Documentation. — <https://arxiv.org/abs/2404.07917>. — 2024.
- [28] *Yuan, Haocheng.* CADTalk: An Algorithm and Benchmark for Semantic Commenting of CAD Programs. — <https://arxiv.org/abs/2311.16703>. — 2023.
- [29] *McClelland, Ryan.* Generative Design and Digital Manufacturing: Using AI and Robots to Build Lightweight Instruments / Ryan McClelland // Proc. SPIE. — 2022.
- [30] *Mews, Maximilian.* Don’t Mesh with Me: Generating Constructive Solid Geometry Instead of Meshes by Fine–Tuning a Code–Generation LLM. — <https://arxiv.org/abs/2411.15279>. — 2024.
- [31] *Ali, Sk Aziz.* BRep Boundary and Junction Detection for CAD Reverse Engineering. — <https://arxiv.org/abs/2409.14087>. — 2024.
- [32] *Zhang, Shuming.* Brep2Seq: A Dataset and Hierarchical Deep Learning Network for Reconstruction and Generation of Computer–Aided Design Models. — Preprint. Available upon request. — 2022.
- [33] Hierarchical CADNet: Learning from B–Reps for Machining Feature Recognition / Andrew R. Colligan, Trevor T. Robinson, Declan C. Nolan et al. // *Comput. Aided Des.* — 2022. — Vol. 147. — P. 103207.

- [34] *Harb, Moataz.* A Novel Algorithm for CAD to CSG Conversion in McCAD. — Technical report. — 2014.
- [35] *Liu, Wenlong.* Symbol as Points: Panoptic Symbol Spotting via Point-Based Representation. — <https://arxiv.org/abs/2401.10556>. — 2024.
- [36] Draw Step by Step: Reconstructing CAD Construction Sequences from Point Clouds via Multimodal Diffusion / Weijian Ma, Shuaiqi Chen, Yunzhong Lou et al. // Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR). — 2024.
- [37] *Yang, Bingchen.* PS-CAD: Local Geometry Guidance via Prompting and Selection for CAD Reconstruction. — <https://arxiv.org/abs/2405.15188>. — 2024.
- [38] *Chen, Tianrun.* Img2CAD: Conditioned 3D CAD Model Generation from Single Image with Structured Visual Geometry. — <https://arxiv.org/abs/2410.03417>. — 2024.
- [39] *Khan, Akber Ali.* Deep Learning for Object Detection: Training Data Generation Using Parametric CAD. — <https://trepo.tuni.fi/bitstream/handle/10024/135807/KhanAkberAli.pdf>. — 2021. — Master's thesis.