Обзор обучающих наборов

Обучающий набор — это коллекция данных, которая используется для обучения моделей машинного обучения. В него входят примеры, на которых модель «учится» распознавать паттерны, делать предсказания или классифицировать информацию. Например, для обучения модели распознавания изображений в набор могут входить фотографии с обозначенными объектами, а для обучения модели по языку — тексты с правильными ответами или метками.

ShapeNet

ShapeNet
ShapeNet — это крупномасштабная база данных 3D-моделей, созданная для использования в области компьютерного зрения, графики, робототехники и машинного обучения. Она служит аналогом ImageNet, но для трёхмерных объектов. ShapeNet активно используется в научных исследованиях, особенно в задачах распознавания, сегментации, генерации и реконструкции 3D-форм.

Основные особенности ShapeNet:

- **Содержит более 3 миллионов 3D-моделей**, из которых около 220 тысяч моделей имеют аннотированные метки и классификацию (в ShapeNetCore).
- **Классификация по категориям**, таким как "стул", "самолёт", "автомобиль", "кровать" и т.д.
- **Аннотации и метаданные**: модели снабжены текстовой информацией, категориями и иерархиями объектов.
- **Множество форматов хранения**: 3D-модели представлены в форматах `.obj`, `.off` и других.
- **Связь с WordNet**: объекты организованы в иерархию на основе WordNet (аналогично ImageNet), что позволяет использовать семантические отношения.

Варианты ShapeNet:

1. **ShapeNetCore** — основной поднабор с тщательно аннотированными объектами из ~55 категорий.
2. **ShapeNetSem** — версия с расширенными семантическими аннотациями (например, для задачи семантической сегментации).
3. **ShapeNetPart** — поднабор, где объекты размечены по частям (например, у стула размечены ножки, сиденье, спинка).

4. **ShapeNetV2** — расширенная версия, улучшенная по качеству и количеству моделей.

Применения:

- Обучение нейросетей на 3D-данных
- Генерация новых 3D-объектов (например, с помощью GAN'ов)
- Сегментация и классификация 3D-форм
- 3D-восстановление по изображению или точкам
- Разработка алгоритмов для дополненной и виртуальной реальности

Структура каталогов **ShapeNet** (например, для набора **ShapeNetCore.v1** или **ShapeNetCore.v2**) организована по принципу:

```
ShapeNetCore.v1/
├── synset_id_1/
│   ├── model_id_1/
│   │   ├── model.obj
│   │   ├── model.mtl
│   │   ├── textures/
│   ├── model_id_2/
│   │   ├── model.obj
│   │   └── ...
├── synset_id_2/
│   └── ...
```

Вот более подробное описание:

Верхний уровень (`ShapeNetCore.v1/` или `ShapeNetCore.v2/`)

- Каждая папка на этом уровне — это **synset** (набор синонимов из WordNet).
- Названия папок — это **WordNet synset ID**. Например:
  - `02691156` — category *airplane*
  - `04256520` — *sofa*
  - `02958343` — *car*

Полный список соответствий ID → категориям можно найти в accompanying JSON-файле (например, `synset_metadata.json`).

Уровень категорий (например, `02691156/`)

- В каждой папке `synset_id` находятся подпапки, каждая из которых — отдельная **3D-модель**.
- Название подпапки — это **уникальный ID модели**.

Пример:

```
ShapeNetCore.v1/
└── 02691156/  ← это "airplane"
    └── bfae6b9d40f63c4f46db6c3a1768b6c3/
        ├── model.obj       ← сама 3D-модель
        ├── model.mtl       ← файл материалов (если есть)
        ├── textures/       ← папка с текстурами (опционально)
```

Внутри каждой модели:

- `model.obj` — основной файл 3D-сетки (в формате Wavefront OBJ).
- `model.mtl` — файл с информацией о материалах (опционален).
- `textures/` — папка с изображениями текстур (если применяются).
- Могут также быть `.json`, `.png`, `.off`, или `.binvox` файлы — в зависимости от поднабора.

Метаданные:

В корне датасета могут быть вспомогательные файлы:

- `taxonomy.json` — древовидная структура категорий с названиями и synset ID.
- `synset_metadata.json` — соответствие synset ID и человеческих названий категорий.
- `model_metadata.json` — описание каждой модели (например, аннотации, ссылки на источники).

Пример структуры для одной категории "airplane":

```
ShapeNetCore.v1/
└── 02691156/  ← "airplane"
    ├── 1a04e3eab45ca15dd86060f189eb133/
    │   ├── model.obj
```

```
│   ├── model.mtl
│   └── textures/
├── 1a6ad7a24bb89733f412783097373bdc/
│   ├── model.obj
│   └── ...
└── ...
```

ModelNet

ModelNet

ModelNet — это широко используемый набор данных для обучения и оценки моделей 3D-объектов. Структура датасета обычно включает следующие компоненты:

1. **Каталоги с классами**: Каждому классу объектов соответствует отдельная папка, например, "chair", "table", "sofa" и т.д.
2. **Файлы моделей**: Внутри каждого класса находятся файлы с 3D-моделью объектов в форматах, таких как:
   - `.off` — формата Object File Format (чаще всего используется),
   - `.ply`,
   - `.obj`.
3. **Метаданные и списки**:
   - Текстовые файлы, такие как `train.txt`, `test.txt`, содержащие список имен файлов для обучения и тестирования.
   - Иногда есть файлы с аннотациями или метками (например, `class labels`).
4. **Описание структуры**:
   - Обычно структура папок выглядит так:

```
ModelNet40/
  ├── chair/
  │    ├── 0.off
  │    ├── 1.off
  │    └── ...
  ├── table/
  │    ├── 0.off
  │    ├── 1.off
  │    └── ...
  ├── train.txt
  ├── test.txt
```

5. **Формат данных**:
   - Основные файлы модели — `.off` файлы — это текстовые файлы, описывающие вершины и грани модели.

- Можно также встретить предварительно обработанные данные в виде точечных облаков (`.ply` , `.xyz`) или векторных представлений.

Scannet

Scannet

ScanNet — это крупномасштабный аннотированный датасет трехмерных реконструкций помещений.

Если говорить просто, **ScanNet — это обширная "библиотека" 3D-сканов реальных комнат и помещений (офисы, гостиные, кухни, ванные и т.д.), где каждый объект в этих сканах подписан.** Компьютер "знает", где находится стена, пол, стол, стул, кровать, унитаз и так далее.

Этот датасет является одним из самых важных и популярных ресурсов в области компьютерного зрения и робототехники.

Ключевые характеристики ScanNet:

- **Объем:** Содержит более **1500** сканов помещений.
- **Аннотации:** Включает **2.5 миллиона** видов с RGB-изображениями, глубиной и данными с инерциальных датчиков (IMU).
- **3D-семантическая сегментация:** Каждая точка в 3D-модели помечена одним из **20** классов (стена, пол, мебель, окно, дверь, книги, холодильник и т.д.).
- **Разнообразие сцен:** Охватывает различные типы жилых и коммерческих помещений.

Как создавался ScanNet?

Процесс сбора данных был очень умным и эффективным:

1. **Сканирование:** Участники использовали планшет (например, iPad) со специальным приложением, которое с помощью встроенного датчика глубины (например, Structure Sensor) сканировало комнату. Нужно было просто медленно обходить помещение.
2. **Автоматическая реконструкция:** Видеопоток и данные о глубине автоматически объединялись для создания единой 3D-модели (воксельной сетки) помещения.
3. **Краудсорсинговая разметка:** Для аннотирования объектов использовалась краудсорсинговая платформа. Людям показывали 3D-сцену и просили обводить контуры объектов на 2D-кадрах, а затем выбирать их класс из списка. Эти 2D-разметки автоматически проецировались обратно на 3D-модель.

Для чего используется ScanNet?

ScanNet стал эталонным бенчмарком для обучения и оценки алгоритмов в самых разных задачах:

1. **Семантическая сегментация 3D-сцен:** Самая прямая задача — научить нейронную сеть понимать, что именно находится в 3D-пространстве. Это основа для "зрения" роботов и умных домов.
2. **Обнаружение объектов в 3D (3D Object Detection):** Найти и выделить ограничивающей рамкой конкретные объекты (например, "найди все стулья в комнате") в трехмерном пространстве.
3. **Задача "Instance Segmentation":** Различать отдельные экземпляры объектов одного класса (например, "это стул №1, а это стул №2").
4. **Задача "Scene Completion":** Предсказание полной 3D-геометрии сцены по неполному скану.
5. **Сопоставление сцен (Scene Retrieval):** Найти в базе данных 3D-сцену, похожую на заданную.
6. **Обучение роботов (Robotics):** Роботы, обученные на ScanNet, лучше ориентируются в реальных помещениях, понимают, какие объекты можно перемещать, и как с ними взаимодействовать.
7. **Дополненная реальность (AR):** Для точного позиционирования виртуальных объектов в реальном мире, чтобы они реалистично взаимодействовали с окружением.

Особенность ScanNet

- **Масштаб и качество:** До его появления не было столь крупного и хорошо аннотированного датасета реальных 3D-сцен.
- **Открытость:** Он публично доступен для академических исследований, что стимулировало огромный прогресс в области.
- **Реалистичность:** Данные собраны в реальных, зачастую "захламленных" условиях, а не в идеализированных лабораторных, что делает модели, обученные на нем, более robust (устойчивыми).

ScanNet — это фундаментальный ресурс, который сыграл ключевую роль в развитии способности искусственного интеллекта понимать и взаимодействовать с трехмерным миром, в котором мы живем.

Отличное уточняющий вопрос! Важно понимать, что **в оригинальном ScanNet нет лидарных данных** в традиционном понимании. Однако есть производные датасеты и способы работы с данными, которые эмулируют лидар.

Оригинальный ScanNet: данные глубины вместо лидара

Вместо лидара ScanNet использует **датчики глубины** (например, Structure Sensor), которые создают плотные карты глубины:

```
scannet/
├── scans/
│   ├── sceneXXXX_XX/
│   │   ├── depth/              # Плотные карты глубины (1280x960)
│   │   │   ├── 0.png           # 16-битные PNG, глубина в мм
│   │   │   ├── 1.png
│   │   │   └── ...
```

Производные датасеты с лидарными данными

1. ScanNet++ (более современная версия)

Включает настоящие лидарные сканы:

```
scannet_pp/
├── scenes/
│   ├── scene_XXXX/
│   │   ├── lidar/
│   │   │   ├── lidar_scan.ply          # Полное лидарное облако точек
│   │   │   ├── lidar_scan_colored.ply  # Лидар с цветом от камер
│   │   │   └── lidar_poses.txt         # Позы лидара
│   │   ├── rgb/                        # RGB изображения
│   │   ├── depth/                      # Данные глубины
│   │   └── calibration/                # Калибровка между сенсорами
```

2. Предобработанные облака точек (эмулирующие лидар)

Многие исследователи создают из глубинных данных разреженные облака точек, похожие на лидар:

```
scannet_processed/
├── train/
│   ├── scene0707_00.ply              # Плотное облако точек
│   ├── scene0707_00_lidar.ply        # Разреженное (лидар-подобное)
│   └── scene0707_00_0.05m.pcd        # Downsampled облако
├── val/
└── test/
```

Semantic3D

Semantic3D

Semantic3D — это крупномасштабный датасет для точечных облаков с семантической разметкой, но в отличие от ScanNet, он focuses на **outdoor-сценах** (наружные сцены).

Semantic3D — это датасет, содержащий плотные точечные облака, полученные с помощью стационарного наземного лидара (террестриального лазерного сканера). Ключевые особенности:

- **Тип сцен**: Наружные сцены - городские площади, улицы, исторические здания, природные ландшафты
- **Масштаб**: Очень большие сцены (до **миллиардов точек** в полных сценах)
- **Разрешение**: Высокоплотные сканы (~1-5 млн точек на файл)
- **Аннотации**: 8 семантических классов для сегментации
- **Сенсор**: Стационарный лидар (не мобильный)

Семантические классы:

```
0 – unlabeled
1 – man-made terrain (искусственный рельеф)
2 – natural terrain (естественный рельеф)
3 – high vegetation (высокая растительность)
4 – low vegetation (низкая растительность)
5 – buildings (здания)
6 – hard scape (мощение, тротуары)
7 – scanning artefacts (артефакты сканирования)
8 – cars (автомобили)
```

Структура каталогов Semantic3D

Базовая структура датасета:

```
semantic3d/
├── train/
│   ├── bildstein_station1_xyz_intensity_rgb.labels
│   ├── bildstein_station1_xyz_intensity_rgb.txt
│   ├── bildstein_station3_xyz_intensity_rgb.labels
│   ├── bildstein_station3_xyz_intensity_rgb.txt
│   └── ... (16 тренировочных сцен)
├── test/
│   ├── marketplacefeldkirch_station1_intensity_rgb.txt
│   ├── marketplacefeldkirch_station4_intensity_rgb.txt
│   └── ... (8 тестовых сцен)
├── test_reduced/              # Уменьшенные тестовые сцены
│   ├── sg27_station1_intensity_rgb.txt
│   └── ...
└── reduced/                   # Уменьшенные тренировочные сцены
    ├── birdfountain_station1_xyz_intensity_rgb.txt
    └── ...
```

Форматы файлов

1. Файлы точечных облаков (`*.txt`)

Содержат собственно точки в формате:

```
x y z intensity r g b
```

Пример:

```
-12.345 6.789 0.123 0.85 128 64 32
-12.340 6.792 0.124 0.82 130 66 34
...
```

2. Файлы меток (`*.labels`)

Содержат семантические метки для каждой точки (только для тренировочных данных):

```
1
2
5
1
...
```

Расширенная структура для практического использования

На практике исследователи обычно создают более сложную структуру:

```
semantic3d_processed/
├── original/                  # Оригинальные данные
│   ├── train/
│   └── test/
├── processed/
│   ├── train/
│   │   ├── bildstein_station1.ply
│   │   ├── bildstein_station1.h5
│   │   └── ...
│   ├── test/
│   └── reduced/               # Для быстрого прототипирования
├── splits/                    # Разделы для обучения
│   ├── train_files.txt
│   ├── val_files.txt
│   └── test_files.txt
├── predictions/               # Предсказания моделей
│   ├── test_pred/
```

```
│   └── val_pred/
└── metadata/
    ├── class_names.txt
    ├── class_colors.txt
    └── statistics.json
```

Популярные форматы для глубокого обучения

1. HDF5 формат (для тренировки):

```
# Каждый .h5 файл содержит:
# points: [N, 6] - XYZ + Intensity + RGB
# labels: [N] - семантические метки
# normals: [N, 3] - нормали (опционально)
```

2. PLY формат (для визуализации):

```
ply
format ascii 1.0
element vertex 1000000
property float x
property float y
property float z
property uchar red
property uchar green
property uchar blue
property uchar class
end_header
```

Ключевые отличия от ScanNet:

Характеристика	Semantic3D	ScanNet
Тип сцен	Outdoor (улицы, площади)	Indoor (помещения)
Размер сцен	Огромные (километры)	Комнаты (метры)
Плотность	Очень высокая	Средняя
Источник	Стационарный лидар	Датчики глубины (Kinect)
Классы	8 (+1 unlabeled)	20 (+1 unannotated)
Формат	Текстовые файлы	.sens + изображения

S3DIS

S3DIS

S3DIS — это один из самых важных и часто используемых датасетов для семантической сегментации помещений.

S3DIS (Stanford Large-Scale 3D Indoor Spaces Dataset) — это масштабный датасет для 3D семантической сегментации внутренних помещений, созданный в Стэнфордском университете.

Ключевые характеристики:

- **Тип сцен**: Внутренние помещения (офисы, учебные аудитории, коридоры)
- **Масштаб**: 6 больших зданий, 271 комната
- **Объем**: ~215 миллионов точек
- **Аннотации**: 13 семантических классов
- **Сенсор**: Matterport Scanner (стационарный 3D сканер)
- **Особенность**: Полные 3D модели целых этажей

Семантические классы S3DIS:

```
0 - ceiling (потолок)
1 - floor (пол)
2 - wall (стена)
3 - beam (балка)
4 - column (колонна)
5 - window (окно)
6 - door (дверь)
7 - table (стол)
8 - chair (стол)
9 - sofa (диван)
10 - bookcase (книжный шкаф)
11 - board (доска)
12 - clutter (разное/мусор)
```

Структура каталогов S3DIS

Оригинальная структура:

```
s3dis/
├── Area_1/
│   ├── conferenceRoom_1/
│   │   ├── Annotation/
│   │   │   ├── beam_1.ptx
│   │   │   ├── ceiling_1.ptx
│   │   │   ├── floor_1.ptx
│   │   │   └── ...
│   │   └── conferenceRoom_1.txt
│   ├── copyRoom_1/
```

```
│   ├── office_1/
│   └── ...
├── Area_2/
├── Area_3/
├── Area_4/
├── Area_5/
└── Area_6/
```

Детальная структура комнаты:

```
Area_1/office_1/
├── Annotation/                # Аннотированные объекты
│   ├── ceiling_1.ptx         # Точки потолка
│   ├── floor_1.ptx           # Точки пола
│   ├── wall_1.ptx            # Точки стен
│   ├── door_1.ptx            # Точки дверей
│   ├── table_1.ptx           # Точки столов
│   ├── chair_1.ptx           # Точки стульев
│   └── ...
├── office_1.txt              # Полное точечное облако комнаты
└── office_1.jpg              # RGB изображение (опционально)
```

Популярное фиксированное разделение:

```
train_areas = ['Area_1', 'Area_2', 'Area_3', 'Area_4', 'Area_6']
test_area = ['Area_5']
```

Ключевые особенности S3DIS:

1. **Полные помещения**: Целые комнаты с полной геометрией
2. **Реалистичные условия**: Различные типы офисных помещений
3. **Сложная сегментация**: Много мелких объектов и "clutter"
4. **Стандартный бенчмарк**: Широко используется для сравнения методов
5. **Сбалансированные классы**: Хорошее распределение по категориям

Сравнение с другими датасетами:

Характеристика	S3DIS	ScanNet	Semantic3D
Тип сцен	Indoor помещения	Indoor помещения	Outdoor сцены
Масштаб	271 комната	1513 сканов	30 сцен
Точки	~215 млн	~2.5 млн видов	Миллиарды точек
Классы	13	20	8
Особенность	Полные комнаты	Последовательности	Высокая плотность

Kitty

KITTI

KITTI — это один из самых важных датасетов в области автономного вождения и компьютерного зрения.

KITTI (Karlsruhe Institute of Technology and Toyota Technological Institute) — это масштабный датасет для разработки и тестирования алгоритмов автономного вождения.

Ключевые характеристики:

- **Тип данных**: Уличные сцены для автономного вождения
- **Сенсоры**: Камеры, лидар, GPS/IMU
- **Объем**: 6 часов трафика, 200k+ изображений
- **Сцены**: Городские, сельские, шоссе
- **Задачи**: Детекция объектов, сегментация, одометрия, трекинг

Структура каталогов KITTI

Базовая структура:

```
kitti/
├── data_object/
│   ├── training/
│   │   ├── image_2/           # Левые RGB камеры [1242x375]
│   │   ├── label_2/           # 2D/3D аннотации объектов
│   │   ├── calib/             # Калибровочные файлы
│   │   └── velodyne/          # Данные лидара [.bin]
│   └── testing/
│       ├── image_2/
│       ├── calib/
│       └── velodyne/
├── data_object_image_2/       # Только изображения
├── data_road/                 # Детекция дороги
├── data_tracking/             # Трекинг объектов
├── data_odometry/             # Визуальная одометрия
└── raw_data/                  # Сырые данные
```

Форматы файлов

1. Файлы калибровки (`calib/*.txt`)

```
P0: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02
0.000000000000e+00 0.000000000000e+00 7.070493000000e+02
```

```
1.805066000000e+02 0.000000000000e+00 0.000000000000e+00
0.000000000000e+00 1.000000000000e+00 0.000000000000e+00
P1: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02
-3.797842000000e+02 0.000000000000e+00 7.070493000000e+02
1.805066000000e+02 0.000000000000e+00 0.000000000000e+00
0.000000000000e+00 1.000000000000e+00 0.000000000000e+00
P2: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02
4.575831000000e+01 0.000000000000e+00 7.070493000000e+02
1.805066000000e+02 -3.454157000000e-01 0.000000000000e+00
0.000000000000e+00 1.000000000000e+00 4.981016000000e-03
P3: 7.070493000000e+02 0.000000000000e+00 6.040814000000e+02
-3.341081000000e+02 0.000000000000e+00 7.070493000000e+02
1.805066000000e+02 2.330660000000e+00 0.000000000000e+00
0.000000000000e+00 1.000000000000e+00 3.201153000000e-03
R0_rect: 9.999128000000e-01 1.009263000000e-02 -8.511932000000e-03
-1.012729000000e-02 9.999406000000e-01 -4.037671000000e-03
8.470675000000e-03 4.123522000000e-03 9.999553000000e-01
Tr_velo_to_cam: 6.927964000000e-03 -9.999722000000e-01 -2.757829000000e-03
-2.457729000000e-02 -1.162982000000e-03 2.749836000000e-03
-9.999955000000e-01 -6.127237000000e-02 9.999753000000e-01
6.931141000000e-03 -1.143899000000e-03 -3.321029000000e-01
Tr_imu_to_velo: 9.999976000000e-01 7.553071000000e-04 -2.035826000000e-03
-8.086759000000e-01 -7.854027000000e-04 9.998898000000e-01
-1.482298000000e-02 3.195559000000e-01 2.024406000000e-03 1.482454000000e-
02 9.998881000000e-01 -7.997231000000e-01
```

2. Файлы аннотаций (`label_2/*.txt`)

```
# Формат: type truncation occlusion alpha bbox dimensions location
rotation_y score
Car 0.00 0 -1.57 387.63 181.54 423.81 203.12 1.67 1.87 3.69 -5.64 1.74
12.34 0.00
Pedestrian 0.00 0 -2.06 583.38 181.72 601.98 205.21 1.67 0.51 0.84 3.29
1.78 7.22 -2.35
Cyclist 0.00 0 -1.85 676.60 163.95 688.98 193.93 1.73 0.60 1.67 4.54 1.68
14.89 -1.77
```

Поля аннотаций:

- `type` : Car, Van, Truck, Pedestrian, Person_sitting, Cyclist, Tram, Misc, DontCare
- `truncation` : Степень обрезания объекта [0-1]
- `occlusion` : 0=полностью видимый, 1=частично, 2=в значительной степени, 3=неизвестно
- `alpha` : Угол наблюдения объекта
- `bbox` : 2D bounding box [x1, y1, x2, y2]
- `dimensions` : 3D размеры [height, width, length]

- `location` : 3D координаты [x, y, z]
- `rotation_y` : Угол поворота вокруг Y-оси

3. Лидарные данные (`velodyne/*.bin`)

Бинарные файлы содержат точки в формате:

```
# Каждая точка: [x, y, z, reflectance]
# x, y, z в метрах, reflectance – интенсивность отражения
points = np.fromfile('000000.bin', dtype=np.float32).reshape(-1, 4)
```

Классы объектов

```
CLASSES = {
    'Car': 0,
    'Van': 1,
    'Truck': 2,
    'Pedestrian': 3,
    'Person_sitting': 4,
    'Cyclist': 5,
    'Tram': 6,
    'Misc': 7,
    'DontCare': 8
}
```