УДК 004.93'12

СРАВНЕНИЕ ПОДХОДОВ К СОЗДАНИЮ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ОБНАРУЖЕНИЯ ОБЪЕКТОВ НА ЛИДАРНЫХ ДАННЫХ В СИСТЕМАХ АВТОНОМНОГО ВОЖДЕНИЯ

Белокрылов К.В., студент группы ИИм-241, II курс
Научный руководитель: Сыркин И.С., к.т.н., доцент
Кузбасский государственный технический университет имени Т.Ф. Горбачёва, г. Кемерово

1. Введение

Разработка систем автономного вождения является одной из самых сложных задач современности. Ключевым компонентом таких систем выступает подсистема восприятия, которая должна обеспечивать безопасное передвижение в динамичной городской среде, интерпретируя окружающий мир в режиме реального времени. В центре этого процесса находятся лидары – сенсоры, формирующие точные 3D-карты окружения в виде облаков точек.

Однако обработка лидарных данных сопряжена с серьёзными техническими вызовами, такими как разреженность массивов данных и необходимость работы с минимальной задержкой. Для решения этих задач были созданы специализированные подходы и технологии. Данная статья посвящена сравнению современных технологических средств, включая базовые библиотеки общего назначения и высокоуровневые фреймворки, а также анализу критериев их выбора для задач обнаружения объектов в системах беспилотного транспорта.

2. Предметная область

Разработка систем автономного вождения – одна из самых сложных междисциплинарных задач современности. Для безопасного передвижения в динамической городской среде беспилотный автомобиль должен обладать надежной системой восприятия, которая позволяет “видеть” и интерпретировать окружающий мир в режиме реального времени.

Современные системы беспилотности практически полностью полагаются на нейронные сети. Они решают критические задачи, с которыми не справляются классические алгоритмы машинного обучения:

- Определение координат и размеров транспортных средств, пешеходов и препятствий;
- Предсказание будущей траектории движения объектов;
- Классификация каждой точки пространства или пикселя изображения (дорога, препятствие, машина, человек).

Для получения данных об окружении используются разные типы сенсоров, каждый из которых имеет свои сильные и слабые стороны. Камеры дают богатую информацию о цвете и текстуре (распознавание знаков, сигналов светофора), но сильно зависят от освещения и не дают точной информации о расстоянии. Радары отлично определяют скорость и работают в любых погодных условиях, но имеют низкое разрешение и шумные данные. Лидары же создают точную 3D-карту окружения с большой точностью, работая независимо от освещения. Они излучают лазерные импульсы и измеряют время их возвращения, формируя облако точек. Это делает их основной технологией для определения геометрии объектов в пространстве [1].

В отличие от обычных 2D-изображений, данные лидара представляют собой разреженные и неупорядоченные наборы точек в 3D-пространстве. Это создает специфические вызовы для нейронных сетей: одно облако может содержать более 100 000 точек; большая часть пространства пуста, и стандартные сверточные нейросети здесь неэффективны; обработка должна происходить с минимальной задержкой, чтобы автомобиль мог вовремя среагировать на сигнал. Именно для решения этих проблем и были созданы специализированные подходы и библиотеки, которые рассматриваются в данной работе.

3. Описание технологий

Программная реализация алгоритмов беспилотного управления требует баланса между гибкостью для исследований и эффективностью для работы в реальном времени. В отличие от классического 2D-зрения, где доминируют стандартные свёрточные нейросети [2, 3], работа с лидарными данными подразумевает обработку неструктурированных и разреженных массивов. Современные решения включают:

1. Библиотеки общего назначения, обеспечивающие базовые операции и автоматическое дифференцирование;
2. Высокоуровневые фреймворки, объединяющие реализации SOTA-моделей и инструменты для работы с популярными датасетами (KITTI [4], Waymo [5], nuScenes [6]).

3.1. TensorFlow и PyTorch

TensorFlow – библиотека с открытым исходным кодом для машинного обучения, разработанная компанией Google. Основной API для работы реализован для языка Python. Она поддерживает масштабируемое распределённое обучение и удобна для промышленного развёртывания. В качестве API использует Keras. Использование статичных вычислительных графов позволяет значительно оптимизировать работу [7, 8].

PyTorch – библиотека для языка Python с открытым исходным кодом для машинного обучения, созданная в Facebook (является продуктом организации

Meta Platforms Inc., признанной экстремистской и запрещённой на территории Российской Федерации) на базе Torch. Использование динамических вычислительных графов позволяет изменять архитектуру нейронной сети во время обучения. Вместе с нативной поддержкой языка Python, это делает PyTorch гибким, а значит популярным для прототипирования и исследовательских задач [9, 10].

3.2. MMDetection3D и OpenPCDet

MMDetection3D – универсальный фреймворк с открытым исходным кодом для создания и обучения нейронных сетей для задач обнаружения объектов и сегментации на облаках точек, созданный лабораторией OpenMMLab и написанный с использованием библиотеки PyTorch. Фреймворк поддерживает создание архитектур моделей путём комбинирования заранее предусмотренных блоков. Поддерживает технологию объединения в одной модели данных от нескольких сенсоров, например лидара и камеры. Поддерживает как датасеты, снятые внутри, так и снаружи. Активно применяется для разработки 3D-моделей общего назначения [11].

OpenPCDet – специализированный фреймворк с открытым исходным кодом для создания и обучения нейронных сетей для обнаружения объектов на лидарных облаках точек для систем автономного движения, также созданный лабораторией OpenMMLab с использованием библиотеки PyTorch. Благодаря более узкому фокусу, OpenPCDet зачастую показывает более высокую скорость инференса. Фреймворк также поддерживает создание архитектур путём совмещения готовых компонентов. Поддерживает популярные для создания беспилотных систем датасеты. Поддерживает создание моделей, использующих в дополнение к лидару камер, а также моделей, оперирующих во временном измерении. Одновременно служит официальным релизом нескольких мощных моделей. Также активно применяется для разработки и тестирования моделей для систем беспилотного транспорта [12].

4. Сравнение

Выбор подходящих технологий для задачи обнаружения объектов на облаках точек зависит от конечной цели проекта, требований к производительности и уровня экспертизы команды разработчиков. На основе рассмотренных технологий можно выделить несколько ключевых критериев для сравнения и итоговой оценки.

Фундаментальный выбор на старте разработки заключается в том, писать ли архитектуру с нуля или использовать готовые модульные решения. Базовые библиотеки предоставляют низкоуровневый API. Это даёт большую гибкость, но требует глубокого понимания лежащих в основе механизмов и написания большого объема шаблонного кода. Библиотеки лучше применять для разработки принципиально новых, нестандартных архитектур нейросетей,

реализация которых отсутствует в готовых фреймворках, для работы со специфическими сенсорами, требующими уникальной предобработки данных, а также в сугубо учебных целях для углубления понимания механизмов работы 3D нейросетей.

Высокоуровневые фреймворки предоставляют готовые строительные блоки и автоматизируют процессы обучения и оценки. Их стоит использовать для быстрого прототипирования и проверки гипотез. Это оптимальный выбор для большинства задач, особенно для начинающих команд, так как фреймворки скрывают сложность реализации, но оставляют возможность наладки отдельных модулей.

Если приоритетом является развертывание модели на конечном устройстве, сильным преимуществом обладает TensorFlow. Благодаря развитой экосистеме он отлично оптимизируется под различные аппаратные средства, обеспечивая необходимую для автономного вождения скорость инференса. Для исследовательских задач и экспериментов, где архитектура сети постоянно меняется, более предпочтительным вариантом остаётся PyTorch благодаря своему динамическому вычислительному графу и простоте отладки.

При выборе между высокоуровневыми фреймворками решающую роль играет предметная область. OpenPCDet является ультимативным решением, если нейронная сеть – это часть системы автономного вождения, а основным источником данных выступают лидарные облака точек. Фреймворк обладает обширным зоопарком одних из самых сильных моделей для обнаружения объектов на лидарных данных (например SECOND [13], PointPillars [14], PV-RCNN [15]), и максимально оптимизирован для этих задач. MMDetection3D же следует выбирать, когда предметная область шире и связана с 3D-нейросетями в целом. Фреймворк незаменим, если архитектура требует сложного объединения данных от лидара, радара и нескольких камер, или если проект включает работу с облаками внутри помещений.

5. Заключение

В данной статье был проведён сравнительный обзор ключевых технологий, применяемых для обнаружения объектов на лидарных облаках точек в задачах автономного вождения.

Выбор базовой библиотеки определяется приоритетами разработки: PyTorch остаётся наиболее предпочтительным инструментом для исследовательской деятельности и прототипирования благодаря гибкости динамических графов, в то время как TensorFlow демонстрирует преимущества при промышленном развертывании и оптимизации моделей под специфическое аппаратное обеспечение.

Специализированные фреймворки значительно упрощают процесс разработки, предоставляя доступ к реализациям современных SOTA-моделей и инструментам для работы с популярными датасетами.

Логика выбора высокоуровневых инструментов напрямую зависит от сложности и направления системы восприятия. OpenPCDet является оптимальным решением для систем, сфокусированных на высокопроизводительной обработке лидарных данных для систем беспилотного транспорта, в то время как MMDetection3D незаменим при создании архитектур, объединяющих данные от лидаров, камер и радаров.

Таким образом, для создания надёжной системы восприятия беспилотного автомобиля необходимо соблюдать баланс между использованием готовых модульных решений для ускорения разработки и низкоуровневой оптимизацией для достижения минимальной задержки при обработке данных в реальном времени. Правильное комбинирование рассмотренных технологий позволяет эффективно решать специфические вызовы, обеспечивая безопасность движения в сложной городской среде.

Список литературы

1. Alaba S. Y., Ball J. E. A Survey on Deep-Learning-Based LiDAR 3D Object Detection for Autonomous Driving // Sensors. 2022. Т. 22. № 24. С. 9577.
2. Redmon J. и др. You Only Look Once: Unified, Real-Time Object Detection [Электронный ресурс]. URL: https://arxiv.org/abs/1506.02640.
3. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation [Электронный ресурс]. URL: https://arxiv.org/abs/1505.04597.
4. Geiger A., Lenz P., Urtasun R. Are we ready for autonomous driving? The KITTI vision benchmark suite [Электронный ресурс]. URL: https://ieeexplore.ieee.org/abstract/document/6248074/.
5. Sun P. и др. Scalability in Perception for Autonomous Driving: Waymo Open Dataset // arXiv:1912.04838 [cs, stat]. 2020.
6. Caesar H. и др. nuScenes: A multimodal dataset for autonomous driving // arxiv.org. 2019.
7. Abadi M. и др. TensorFlow: A system for large-scale machine learning // arXiv:1605.08695 [cs]. 2016.
8. TensorFlow. TensorFlow [Электронный ресурс]. URL: https://www.tensorflow.org/.
9. Paszke A. и др. PyTorch: An Imperative Style, High-Performance Deep Learning Library [Электронный ресурс]. URL: https://arxiv.org/abs/1912.01703.
10. PyTorch. PyTorch [Электронный ресурс]. URL: https://pytorch.org/.

11. MMDetection3D Contributors. OpenMMLab’s Next-generation Platform for General 3D Object Detection [Электронный ресурс]. URL: https://github.com/open-mmlab/mmdetection3d.
12. open-mmlab. GitHub - open-mmlab/OpenPCDet: OpenPCDet Toolbox for LiDAR-based 3D Object Detection. [Электронный ресурс]. URL: https://github.com/open-mmlab/OpenPCDet.
13. Yan Y., Mao Y., Li B. SECOND: Sparsely Embedded Convolutional Detection // Sensors. 2018. Т. 18. № 10. С. 3337.
14. Lang A. H. и др. PointPillars: Fast Encoders for Object Detection from Point Clouds // arXiv:1812.05784 [cs, stat]. 2019.
15. Shi S. и др. PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection // arXiv:1912.13192 [cs, eess]. 2021.