УДК 004.93’1

НЕЙРОСЕТЕВАЯ СИСТЕМА РАСПОЗНАВАНИЯ ТРЕХМЕРНЫХ ОБЪЕКТОВ

Ляшов М.В., Шемякина М.А., Бабаев А.М.

Институт сферы обслуживания и предпринимательства (филиал) ДГТУ, Шахты, e-mail: shemyakina.marina97@gmail.com

Данная статья посвящена разработанному нейросетевому алгоритму распознавания трехмерных объектов. Было принято решение рассматривать задачу распознавания трехмерных объектов в контексте автономного управления автомобилем. Для этого были проанализированы наиболее популярные алгоритмы в области детектирования трехмерных объектов. Это позволило сделать вывод о том, что для обучения точного алгоритма необходимо использовать облака точек, вычисляемые с помощью Lidar-сканеров. Кроме того, требуется сохранять как можно больше пространственной информации на этапе представления облака точек в виде упорядоченной структуры. В то же время для сохранения уровня производительности, достаточного для работы в реальном времени, следует избегать использования затратных по времени операций при извлечении признаков из облака точек, а при построении собственного алгоритма необходимо сделать выбор в пользу одноступенчатых алгоритмов обнаружения объектов. В результате был получен алгоритм, который способен обрабатывать более 17 кадров в секунду, что является достаточной скоростью для работы в режиме реального времени. Помимо этого, он имеет высокую точность, сопоставимую с качеством распознавания других алгоритмов обнаружения трехмерных объектов.

Ключевые слова: распознавание образов, распознавание трехмерных объектов, компьютерное зрение, свёрточные нейронные сети, Tensotflow

NEURAL NETWORK SYSTEM FOR RECOGNITION 3D-OBJECTS

Lyashov M.V., Shemyakina M.A., Babaev A.M.

Institute of Service and Business (branch) Don State Technical University, Shakhty, e-mail: shemyakina.marina97@gmail.com

This article is devoted to the developed neural network algorithm for recognition 3d-objects. It was decided to consider the problem of recognizing 3d-objects in the context of autonomous driving. Popular algorithms for detecting 3d-objects were analyzed. This allows us to conclude that point clouds calculated using Lidar scanners should be used to train the exact algorithm. In addition, it is required to store as much spatial information as possible during the stage of representing the point cloud in the form of an ordered structure. At the same time, to maintain a level of performance sufficient for real-time operation, one should avoid using expensive operations when extracting features from a point cloud, and when building a custom algorithm, one should make a choice in favor of one-stage object detection algorithms. As a result, an algorithm was obtained that is capable of processing more than 17 frames per second, which is a speed sufficient for working in real time. In addition, it has high accuracy comparable to the recognition quality of other 3D-object detection algorithms.

Keywords: object recognition, recognition 3D-objects, computer vision, convolutional neural network, Tensorflow

С развитием методов глубокого обучения исследователи получили очень эффективный способ распознавания образов – свёрточные нейронные сети [1]. Наиболее выдающиеся результаты были достигнуты в области распознавания двумерных объектов [2]. Однако с повышением уровня доступности трехмерных сканирующих устройств всё больше исследователей привлекает проблема распознавания объектов на основе трехмерных данных.

Использование трехмерных данных предоставляет дополнительную пространственную информацию, что позволяет решить ряд проблем распознавания двумерных изображений. Одной из популярных задач, где важна информация о форме объектов и их расположении в окружающей среде, является автономное управление автомобилем. При распознавании RGB-изображений высок риск распознавания автомобилей, например, на наружной рекламе. Это может приводить к возникновению ДТП. Однако при использовании облаков точек вероятность подобных ошибок снижается. Поэтому решение задачи распознавания трехмерных объектов в контексте автономного управления автомобилем является актуальным.

Цель исследования заключалась в разработке алгоритма, способного эффективно решать задачу распознавания трехмерных объектов в контексте автономного вождения автомобиля. Учитывая специфику управления автомобилем, разрабатываемая система должна иметь высокую производительность для работы в режиме реального времени, а также достигать высокой точности распознавания объектов для обеспечения безопасности.

Современные алгоритмы распознавания трехмерных объектов. Наиболее по-

пулярными алгоритмами распознавания трехмерных объектов на основе данных, получаемых от Lidar-сканеров, являются:

1. VoxelNet [3] – это двухступенчатый алгоритм, который использует дополнительную сеть для поиска областей интереса в трехмерном пространстве. Одной из проблем, возникающих при распознавании трехмерных объектов, является разреженность облаков точек. Для ее решения в VoxelNet облако точек делят на воксели равных размеров. VoxelNet обеспечивает высокую точность, но проигрывает другим методам в плане производительности. В большей степени это происходит из-за вычислительно дорогой операции трехмерной свертки. Кроме того, из-за разреженности облака точек возникает большое количество пустых вокселей. Как следствие, при обработке сцены выполняется огромное число лишних вычислений и алгоритм позволяет обрабатывать не более двух кадров в секунду.

2. Complex-YOLO [4]. В отличие от VoxelNet, это одноэтапный алгоритм, благодаря чему достигается более высокая производительность, что позволяет обрабатывать до 50 кадров в секунду. Данный метод заимствует архитектуру детектора YOLO и адаптирует ее для случая трехмерных данных. В Complex-YOLO для решения проблемы разреженных данных точки трехмерного облака проецируются на двумерную сетку. Облако точек также разбивается на ячейки, где высота каждой из них равна высоте всего облака. Значения ячеек сетки основаны на свойствах точек, которые в них содержатся. В Complex-YOLO такими значениями являются: максимальная интенсивность, высота и нормализованная плотность точек. Эти три величины кодируются как три канала изображения, создавая входную карту признаков, которая может обрабатываться операциями двумерной свертки как в алгоритме YOLO. Полученная двумерная сетка является проекцией облака точек, которую называют видом сверху (birds-eye-view, далее – BEV). BEV-проекция позволяет существенно понизить количество анализируемых данных, но в то же время теряет информацию о высоте и интенсивности каждой конкретной точки, что может отрицательно сказываться на точности алгоритма при решении некоторых задач.

3. AVOD. Существует также класс архитектур, которые распознают трехмерные объекты, используя, кроме данных Lidar-сканера, также изображение с обычной камеры. Одним из наиболее популярных методов данной категории является алгоритм Aggregated View Object Detection (далее AVOD) [5]. Изображение представляется в виде RGB-матрицы, которая обрабатывается в два шага. Сначала изображения билинейно изменяются до одинакового размера. Затем для каждого канала RGB среднее значение конкретного канала по всем изображениям вычитается из каждого изображения.

Для того чтобы использовать данные алгоритмы в системах, работающих в режиме реального времени, необходимо:

– для получения лучшей производительности использовать одноэтапные детекторы;

– достичь высокой точности детекции, сохраняя как можно больше пространственной информации, как в алгоритме VoxelNet.

Данные особенности реализованы в предлагаемом алгоритме.

Предлагаемое решение. С целью сохранения большего количества пространственной информации для представления облака точек в виде упорядоченной структуры было решено воспользоваться BEV-проекцией. Предварительно из облака точек удаляются точки, координаты которых не соответствуют условиям, указанным в формуле

$$x \in [0\text{ м}, 70\text{ м}],$$
$$y \in [-40\text{ м}, 40\text{ м}], \quad (1)$$
$$z \in [-2\text{ м}, 1{,}25\text{ м}].$$

Это позволяет не учитывать точки, находящиеся на большом расстоянии от Lidar-сканера. За пределами указанной области точность сканера снижается, а облако точек становится разреженным. Ограничения для координаты z были выбраны с учетом того, что Lidar установлен на высоте 1,73 м. Таким образом, облако точек после обработки охватывает высоту, равную 3 м над уровнем автомобильной дороги, которая является достаточной для детектирования автомобилей.

Матрица, которую в дальнейшем планируется использовать для подачи в детектор, имеет 800 строк и 704 столбцов и агрегирует данные из BEV-проекций разных масштабов. Первая проекция также имеет размерность 800×704. Для её построения облако точек разбивается на вертикальные ячейки с шагом дискретизации 0,1 м. Каждая полученная ячейка дополнительно разбивается с шагом 0,65 м на 5 ячеек. Каждая основная ячейка в дальнейшем кодируется с помощью семимерного вектора. Первые пять чисел – максимальная высота точек, h_i, $i = 1,2,\ldots,5$, в каждой дополнительной ячейке. Шестое и седьмое – максимальная интенсивность точек I и нормализованная плотность точек D в основной ячейке, которая вычисляется по формуле

$$D = \min\left(1, \frac{\log(N+1)}{\log(64)}\right). \quad (2)$$

Далее выполняется построение BEV-проекции, размер которой 400×352, то есть в два раза меньше первой. Для этого исходное облако точек необходимо дискретизировать с шагом 0,2. Однако в целях получения более вычислительно эффективного алгоритма вторая BEV-проекция формируется путем применения операций:

– max pooling с размером ядра 2×2 и шагом (2, 2) к первым 6 значениям вектора, кодирующего ячейки первой проекции;

– average pooling с размером ядра 2×2 и шагом (2, 2) к последнему каналу, соответствующему нормализованной плотности точек.

Аналогичным образом осуществляется формирование третьей проекции размером 200×176 на основе предыдущей проекции. С уменьшением размера матрицы увеличивается шаг дискретизации – для третьей проекции он равен 0,4 м. Далее выполняется объединение полученных проекций. Для этого вектору значений каждой ячейки из начальной BEV-проекции ставится в соответствие вектор значений ячейки из уменьшенных BEV-проекций. В результате данной операции будет получена матрица 800×704×2. Благодаря учету максимальных высот точек из дополнительных ячеек было сохранено больше пространственной информации, чем в архитектуре Complex-YOLO. А объединение BEV-проекций разных масштабов позволило добавить к каждой ячейке исходной проекции информацию о точках в ее окрестности.

Полученный в ходе обработки облака точек тензор размера 800×700×21 подается на вход сверточной нейронной сети. Она была разработана на основе архитектуры DarkNet-19, которая использовалась в детекторе YOLO. В табл. 1 представлена схема предлагаемой нейронной сети. В отличие от DarkNet-19 разработанная архитектура имеет 16 сверточных слоев и 3 слоя max pooling, а также меньшее количество фильтров, что позволяет получить вычислительно эффективный алгоритм в условиях входных данных большой размерности. Кроме того, для улучшения качества обучения алгоритма в слоях свертки была использована функция активации Leaky ReLU, а также добавлены слои пакетной нормализации, которые, несмотря на добавление дополнительных вычислений, позволяют ускорить сходимость моделей глубокого обучения. Матрица, получаемая на выходе сверточной нейронной сети, имеет размер 100×88. Таким образом, полученная матрица карт признаков соответствует облаку точек, дискретизированному с шагом 0,8 м. Число фильтров в последнем слое равно $B*(8 + C)$, где B – это количество якорей (anchors) для каждого из C классов.

Таблица 1

Схема разработанной сверточной нейронной сети

Слой	Фильтр	Размер	Шаг	Вход	Выход
Сверточный № 1	3×3	64	1, 1	800×704×21	800×704×64
Сверточный № 2	3×3	128	1, 1	800×704×64	800×704×128
Max pooling № 1	2×2		2, 2	800×704×128	400×352×128
Сверточный № 3	3×3	128	1, 1	400×352×128	400×352×128
Сверточный № 4	1×1	64	1, 1	400×352×128	400×352×64
Сверточный № 5	3×3	128	1, 1	400×352×64	400×352×128
Max pooling № 2	2×2		2, 2	400×352×128	200×176×128
Сверточный № 6	3×3	256	1, 1	200×176×128	200×176×256
Сверточный № 7	1×1	128	1, 1	200×176×256	200×176×128
Сверточный № 8	3×3	256	1, 1	200×176×128	200×176×256
Сверточный № 9	1×1	128	1, 1	200×176×256	200×176×128
Сверточный № 10	3×3	256	1, 1	200×176×128	200×176×256
Max pooling № 3	2×2		2, 2	200×176×256	100×88×256
Сверточный № 11	3×3	512	1, 1	100×88×256	100×88×512
Сверточный № 12	1×1	256	1, 1	100×88×512	100×88×256
Сверточный № 13	3×3	512	1, 1	100×88×256	100×88×512
Сверточный № 14	1×1	256	1, 1	100×88×512	100×88×256
Сверточный № 15	3×3	512	1, 1	100×88×256	100×88×512
Сверточный № 16	1×1	B * (8 + C)	1, 1	100×88×512	100×88×B * (8 + C)

Итоговая матрица карт признаков имеет размер $100\times88\times B^*(8 + C)$, что дает $880\times B$ предсказаний. В условиях задачи многоклассовой классификации и использования нескольких якорей для каждого класса число итоговых прогнозов может достигать больших значений. Кроме того, необходимо удалять пересекающиеся ограничительные окна. Для отбора подходящих прогнозов используется алгоритм подавления, который является одним из компонентов алгоритма YOLO. В контексте задачи распознавания трехмерных объектов он включает следующие шаги:

1. Удалить прогнозы с вероятностью нахождения объекта в ячейке $s \le 0{,}6$.

2. Вычислить итоговые оценки $\Pr(class\ j)_i$ классов j, $j = 1,2,\ldots,C$, для каждого отобранного якоря i, $i = 1,2,\ldots,B$, по формуле

$$\Pr\left(class\ j\right)_i = \begin{cases} p_{ij} * s_i, p_{ij} * s_i \ge 0{,}2 \\ 0, p_{ij} * s_i < 0{,}2 \end{cases}. \quad (3)$$

3. Отсортировать полученные оценки для каждого класса и выбрать ограничительное окно с максимальной оценкой.

4. Обнулить оценки ограничительных окон, для которых пересечение с ограничительным окном, выбранным в 3 шаге, больше 0,5 по метрике IoU.

5. Повторить шаги 4 и 5 для других ограничительных окон, оценки которых не равны нулю.

6. Повторить шаги 2–5 для всех классов.

Для каждого ограничительного окна отсортировать полученные оценки классов. Если наибольший элемент не равен нулю, то ограничительное окно включается в итоговый прогноз модели, а индекс этого элемента – это класс объекта, который находится в данном ограничительном окне.

Экспериментальные исследования. Традиционно для обучения моделей детектирования трехмерных объектов в задаче автономного вождения автомобилем, а также для оценки её качества, исследователями используется набор данных KITTI [6]. Он предоставляет несколько вариантов данных: RGB-изображения, стереоизображения и облака точек. А также данные для калибровки устройств, что при работе с облаками точек позволяет наносить обнаруженные ограничительные окна объектов на RGB-изображения. Предлагаемый алгоритм был реализован с помощью фреймворка глубокого обучения TensorFlow с использованием Keras для упрощения разработки [7]. Набор данных KITTI был разделен на обучающую и валидационную выборки, которые включают 70% и 30% всех данных соответственно. В качестве аппаратной платформы использовался сервис Google Colaboratory, предоставляющий доступ к виртуальной машине, которая настроена на работу с алгоритмами машинного обучения и имеет установленную видеокарту Tesla P100 [8]. Обучение производилось в течение 100 эпох с размером пакетов равным 16.

Для оценки производительности и качества детекции объектов было выполнено сравнение разработанного алгоритма с архитектурами VoxelNet, Complex-YOLO и AVOD. В качестве критерия сравнения использовалась метрика AP для класса «Car» при каждом из трех уровней сложности. Значение порога IoU было установлено равным 0,5. Результаты тестирования приведены в табл. 2.

По результатам тестирования можно сделать ряд выводов.

1. Разработанный алгоритм имеет высокую производительность, а также качество детектирования трехмерных объектов, сопоставимое с точностью аналогов.

2. Алгоритм обладает более низкой производительностью по сравнению с Complex-YOLO из-за усложненного процесса построения BEV-проекций и более сложной архитектуры сверточной нейронной сети. Однако эти особенности позволили алгоритму превзойти Complex-YOLO в точности детекции объектов.

3. VoxelNet имеет более высокую точность. Это связано с тем, что предлагаемый алгоритм имеет упрощенный алгоритм извлечения признаков. Но это также является преимуществом, так как отсутствие вычислительно сложной нейронной сети для извлечения признаков вокселей позволяет алгоритму обрабатывать облако точек в реальном времени.

Таблица 2

Сравнение алгоритмов распознавания трехмерных объектов

	Easy, AP	Moderate, AP	Hard, AP	FPS
VoxelNet	0,8197	0,6485	0,6285	4,3
Complex-YOLO	0,6772	0,6400	0,6301	50,4
AVOD	0,7359	0,6578	0,5838	8,5
Предлагаемый алгоритм	0,7487	0,6374	0,6071	17,1

4. При тестировании также удалось достичь сопоставимого с AVOD качества работы. Во многом этого удалось достичь за счет сохранения большого количества пространственной информации при агрегировании BEV-проекций разного масштаба.

Выводы

В данной работе был представлен разработанный алгоритм распознавания трехмерных объектов. Задача распознавания трехмерных объектов решалась в контексте автономного управления автомобилем. На основе проведенного анализа был сделан вывод, что для создания точного алгоритма необходимо использовать облака точек, вычисляемые с помощью Lidar-сканеров. Кроме того, требуется сохранять как можно больше пространственной информации на этапе представления облака точек в виде упорядоченной структуры. В то же время для работы алгоритма в режиме реального времени был сделан выбор в пользу одноступенчатых алгоритмов обнаружения объектов. Для ускорения этапа извлечения признаков было решено отказаться от использования нейросетевых технологий и воспользоваться методом построения BEV-проекций. Архитектура сверточной нейронной сети для распознавания трехмерных объектов была разработана на основе YOLO – одноступенчатого детектора двумерных алгоритма. Разработанный алгоритм имеет аналогичную структуру расположения сверочных слоев, а также слоев субдискретизации и пакетной нормализации. При этом глубина сети и количество фильтров было уменьшено для достижения высокой производительности. В результате был получен алгоритм, который способен обрабатывать более 17 кадров в секунду, что является скоростью, достаточной для работы в режиме реального времени. Помимо этого, он имеет высокую точность, сопоставимую с качеством распознавания других алгоритмов обнаружения трехмерных объектов.

Список литературы

1. Goodfellow I., Bengio Y., Courville A. Deep Learning. The MIT Press. 2016. 800 p.

2. Simonyan K. Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR. 2014. P. 587–601.

3. Maturana D. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. International Conference on Intelligent Robots and Systems (IROS). Hamburg, 2015. P. 922–928.

4. Simon M. Complex-YOLO: Real-time 3D Object Detection on Point Clouds. European Conference on Computer Vision. Munich, 2018. P. 1–14.

5. Ku J. Joint 3D Proposal Generation and Object Detection from View Aggregation. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid, 2019. P. 1–8.

6. Geiger A. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. Conference on Computer Vision and Pattern Recognition (CVPR). Providence, 2012. P. 1–8.

7. TensorFlow: официальный сайт. [Электронный ресурс]. URL: https://www.tensorflow.org (дата обращения: 07.09.2020).

8. Параллельные вычисления CUDA: официальный сайт NVIDIA Corporation. [Электронный ресурс]. URL: http://www.nvidia.ru/object/cuda-parallel-computing-ru.html (дата обращения: 07.09.2020).