## Постановка задачи

# Описание исходной задачи работы

Исходной задачей работы является реализация искусственной нейронной сети с использованием компьютерного зрения для классификации и детекции объектов на видео в режиме реального времени (Real-Time Object Detection). В качестве объектов детекции выбраны дорожные светофоры. Архитектура сети представляет собой одностадийную нейросеть SSD300 (Single Shot Multibox Detector). Данная архитектура отлично подходит для детекции объектов в режиме реального времени за счет высокой производительности (более 25 кадров в секунду).

# Актуальность задачи

С развитием технологий и возрастанием интереса к автономным транспортным средствам, особенно актуально становится использование искусственного интеллекта для анализа дорожной обстановки. В настоящее время ведется активная разработка программного обеспечения для систем автономного управления транспортом, включающего в себя компьютерное зрение и нейросети.

Одной из основных задач в процессе анализа видеопотока, получаемого от различных датчиков и камер и отражающего информацию об окружающей обстановке, является обнаружение различных дорожных объектов, в том числе светофоров. Надежная и быстрая детекция подобных объектов в беспилотных автомобилях обеспечивает выполнение безопасных маневров и позволяет снизить вероятность дорожно-транспортных происшествий.

Детекторы также находят применение в области систем управления трафиком, оптимизации потока транспорта, систем контроля скорости и предупреждения столкновений.

В целом точное и эффективное обнаружение объектов на дороге в режиме реально времени имеет значительную актуальность в контексте безопасности и оптимизации дорожного движения, а также в развитии автономных транспортных систем.

# Метрики качества

Метрика качества – Mean Average Precision (mAP).

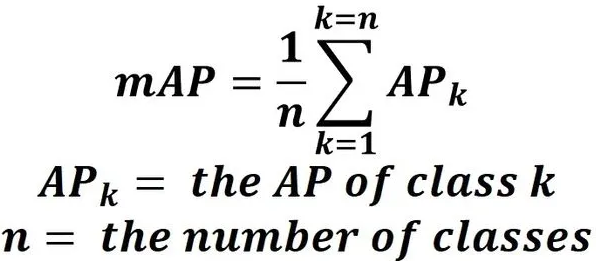
Mean Average Precision (средняя площадь под кривой Precision-Recall) – стандартная метрика для оценки качества детекторов объектов. Широко используется в задачах детекции, включая детекцию дорожных объектов. При оценке детектора объектов с помощью метрики mAP выполняются следующие шаги:

Вычисление Precision и Recall: для каждого класса объектов вычисляются значения точности (Precision) и полноты (Recall) на разных порогах бинарной классификации (то есть, определение, считать ли объект обнаруженным или нет). Precision — это доля правильно предсказанных положительных объектов от всех предсказанных положительных объектов, а Recall — это доля правильно предсказанных положительных объектов от всех реальных положительных объектов.

Построение кривой Precision-Recall: для каждого класса строится кривая, в которой по оси X откладывается Recall, а по оси   
Y - Precision. Эта кривая показывает, как меняется точность при разных уровнях полноты.

Вычисление Average Precision (AP): Average Precision вычисляется путем вычисления площади под кривой. Это значение представляет собой среднюю точность для данного класса.

Вычисление mAP: для получения mAP вычисляются средние значения AP по всем классам объектов. Для множества классов, mAP будет средним значением AP для каждого класса.



## Анализ данных

# Исходные данные:

Датасет – «S2TLD\_720x1280 Traffic Light Detection XML Format»  
[*https://www.kaggle.com/datasets/sovitrath/s2tld-720x1280-traffic-light-detection-xml-format*](https://www.kaggle.com/datasets/sovitrath/s2tld-720x1280-traffic-light-detection-xml-format)  
Количество изображений в обучающей выборке – 3785 шт.  
Количество изображений в тестовой выборке – 779 шт.  
Количество классов – 4.  
Классы – «red», «yellow», «green», «off».  
Формат аннотаций – XML.  
Формат представления боксов объектов – «[x\_min, y\_min, x\_max, y\_max]»

Особенности: особенностью датасета является несбалансированное распределение классов. Дисбаланс наблюдается как по суммарному количеству объектов, так и по среднему количеству объектов на одно изображение. Далее будет предложен метод калибровки модели для устранения дисбаланса.

|  |  |
| --- | --- |
|  |  |

Код для анализа данных: смотреть в ноутбуке **analysis\_and\_inference.ipynb**.

# Анализ аналогичных решений

Детекция дорожных объектов с использованием сверточных нейронных сетей (CNN) — это активно развивающаяся область компьютерного зрения и автономной навигации. Существует множество архитектур и подходов, используемых для решения этой задачи.

* Faster R-CNN (Region Convolutional Neural Network): один из пионерских методов, сочетающий сверточные слои для извлечения признаков и региональную свёртку для детекции объектов. Модель генерирует предложения для областей, которые могут содержать объекты, а затем применяет свёрточные операции для определения объектов и их координат. Faster R-CNN с ResNet-50 backbone достигает примерно 30-35% mAP на COCO датасете [1].
* YOLO (You Only Look Once): делит изображение на сетку и предсказывает ограничивающие рамки и классы объектов для каждой ячейки сетки. Он известен своей высокой скоростью работы и способностью детектировать объекты в реальном времени. В зависимости от версии (например, YOLOv3), достигается примерно 28-33% mAP на COCO датасете [2].
* RetinaNet: этот метод объединяет преимущества Faster R-CNN и SSD, используя фокусировку на слабых сторонах предыдущих методов. Он использует Feature Pyramid Network (FPN) для более эффективного извлечения признаков разных масштабов и вводит новый тип якорей для борьбы с проблемой дисбаланса классов. Максимальный mAP достигает 44-45% на валидационном датасете [3].
* EfficientDet: этот метод стремится найти баланс между скоростью и точностью детекции. Он использует компаунд-масштабирование для увеличения эффективности использования вычислительных ресурсов при обучении на разных масштабах.
* Cascade R-CNN: этот метод предлагает каскадный подход к детекции, где последовательно применяются модели разной сложности для уточнения детекций. Это позволяет добиться высокой точности за счёт меньшего числа ложных срабатываний.
* CenterNet: вместо предсказания ограничивающих рамок, этот метод предсказывает центры объектов и расстояния до краёв объектов. Он также способен детектировать объекты различных форм, таких как ключевые точки или лица.
* Detectron2: фреймворк для разработки и обучения детекторов объектов, разработанный Facebook AI Research. Он предоставляет реализации многих современных методов детекции и обладает гибкой настройкой и расширяемостью [4].

## Методика реализации

*Этап 1 (create\_data\_lists.py)*

IN: папка S2TLD\_720x1280

На первом этапе необходимо получить из исходного датасета список пар в формате «изображение – объекты на изображении». Для этого выполняется парсинг XML файла и производится запись полученных боксов и классов в JSON файлы. Параллельно записывается второй JSON файл с путями до изображений. Данная процедура применяется к обучающему и валидационному датасетам. Также формируется файл JSON, содержащий label\_map — словарь меток-индексов, с помощью которого метки кодируются в двух предыдущих JSON. Этот словарь также доступен в utils.py.

OUT: TRAIN\_objects.json, TRAIN\_images.json, TEST\_objects.json, TEST\_images.json, label\_map.json

*Этап 2.1 (формирование train\_dataset)*

IN: TRAIN\_objects.json, TRAIN\_images.json, TEST\_objects.json, TEST\_images.json, label\_map.json

Далее создается объект класса PyTorch Dataset. Для получения изображения и объектов реализован метод «\_\_getitem\_\_», где происходят необходимые трансформации. Используем базу VGG-16, предварительно обученную в ImageNet, из модуля torchvision PyTorch. В документации описана предварительная обработка, необходимая для использования этой модели — значения пикселей должны находиться в диапазоне [0,1], а затем мы должны нормализовать изображение по среднему и стандартному отклонению RGB каналов в изображениях ImageNet [6].

Этап 2.2 *(формирование train\_dataset, аугментации)*

Аугментации.

Базовые аугментации библиотеки “torchvision.transforms.functional” заменены аугментациями библиотеки “albumentations”. Причина – автоматическое изменение координат bounding box-ов при геометрических трансформациях изображения.



1. Минимальное количество цветовых аугментаций, т.к. цвет – основной признак объекта.
2. Имитация детекции в движении с помощью “motion blur”. Важно установить значение аргумента allow\_shifted = False, так как в ином случае возникают смещения bounding box-ов по отношению к детектируемым объектам на изображении.



1. Имитация пыли/загрязнений при съемке с использованием PixelDropout.



1. Имитация снега с использованием RandomSnow.

|  |  |
| --- | --- |
|  |  |

Инициализация весов.

Важно придать начальное значение весам. Крайне не рекомендуется инициализировать начальные веса нулями. Можно получить одинаковые фильтры и одинаковые градиенты по этим фильтрам. Получатся тождественные фильтры с одинаковыми преобразованиями. Поэтому инициализируем веса c помощью «torch.nn.init.xavier\_uniform\_» [5].

Pipel

CLAHE

|  |  |
| --- | --- |
|  |  |

*Откуда взять источники:*

[*https://cyberleninka.ru/article/n/obnaruzhenie-transportnyh-sredstv-na-izobrazheniyah-zagorodnyh-shosse-na-osnove-metoda-single-shot-multibox-detector/viewer*](https://cyberleninka.ru/article/n/obnaruzhenie-transportnyh-sredstv-na-izobrazheniyah-zagorodnyh-shosse-na-osnove-metoda-single-shot-multibox-detector/viewer)

# Список источников:

1. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." NeurIPS 2015.
2. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection." CVPR 2016.
3. Traffic Light Detection Using RetinaNet and PyTorch. <https://debuggercafe.com/traffic-light-detection-using-retinanet/>
4. FAIR (Facebook AI Research). <https://github.com/facebookresearch/detectron2>
5. Understanding the difficulty of training deep feedforward neural networks - Glorot, X. & Bengio, Y. (2010)
6. <https://pytorch.org/vision/stable/models/generated/torchvision.models.vgg16.html>