Анализ видео дорожного трафика

Емельянова Ольга

14 мая 2020 г.

1. Использование приложения

1.1. Сборка

Чтобы собрать приложение, понадобятся библиотеки PyTorch для C++ и OpenCV, а также CMake. В файле CMakeLists.txt необходимо будет поменять пути в двух местах: `CMAKE_PREFIX_PATH` для PyTorch и `OpenCV_DIR` для OpenCV на соответствующие пути в вашей системе. Лучше использовать Qt Creator, так как я не проверяла, как именно сбилдить просто с командной строки.

1.2. Демо-видео

Поскольку я пока не могу сбилдить под Windows, я выложила несколько демо-видео на GitHub.

1. `adam-100e.mp4` – Результат, который я считаю лучшим. Модель и приложение для этого варианта будут описаны далее.
2. `other-sample.mp4` – Та же модель, другое видео. О видео будет далее (в части 2).
3. `adam-img20k.mp4` – Здесь модель обучалась на 20000 изображениях в течении 64 эпох. Но результат получился визуально хуже: почему-то модель очень часто объединяет несколько машин в одну группу.
4. `tinier.mp4` – Я попыталась убрать часть сверточных слоев. Однако результат мне кажется хуже (автобус в начале не распознается, например), а скорость не увеличилась: с основным вариантом примерно 220-230 мс на кадр, с этим 190-200. Визуально примерно одинаково.

2. Нейронная сеть

2.1. Этапы построения модели

За основу я взяла YOLOv3. В репозитории есть куча конфигов, из которых я взяла Tiny YOLOv3, но я оставила только первые 16 слоев (до первого слоя YOLO),

примерно как в YOLOv1. Еще у меня был вариант, где я убрала 13 и 14 слои, но даже так результат был хуже (см. демо-видео), поэтому я решила больше с этим не экспериментировать.

Сразу замечу, что на Google Colab оказывается есть лимиты, как удивительно, поэтому я вынуждена была быть экономной с вариантами сетей, которые я проверяла.

- **Размер входа**: 416*416
- **Количество каналов**: см. конфиг, часть filters, а также схему ниже
- **Количество классов**: 5 (машина, грузовик, автобус, мотоцикл, человек)

Схема:

Номер слоя	Тип	Размер ядра	Кол-во фильтров	Примечания
0	свертка	3	16	
1	maxpool	2		
2	свертка	3	32	
3	maxpool	2		
4	свертка	3	64	
5	maxpool	2		
6	свертка	3	128	
7	maxpool	2		
8	свертка	3	256	
9	maxpool	2		
10	свертка	3	512	
11	maxpool	2		
12	свертка	3	1024	
13	свертка	1	256	полносвязный
14	свертка	3	512	
15	свертка	1	30	полносвязный
16	YOLO			извлекает результаты

Вывод предпоследнего слоя имеет (для одного изображения) форму 13*13*30. Могу описать подробнее, если надо, но суть в том, что каждый вектор длины 30 состоит из 3 предсказаний рамок, каждая из которых в свою очередь состоит из координат рамки, оценки вероятности содержания в ней объекта и вектора оценок принадлежания этого объекта к каждому из классов. При этом считается, что изображение как бы разбито на клетки, каждая из которых предсказывает рамки относительно своих границ. Последний же слой все это извлекает в более понятный вид: вектор рамок с координатами относительно изображения.

Функция потерь состоит из нескольких частей. Во-первых, это среднеквадратичное отклонение координат рамки. Во-вторых, это кросс-энтропия между предсказаниями модели и правильной классификацией. Но тут есть нюанс. Опять же, могу написать подробнее, но если кратко: в зависимости от того, какая клетка по IoU лучше всего соответствует какой рамке, она назначается ”ответственной” за нее. Эта клетка теряет, если плохо предсказывает эту рамку. Это первая часть

классификационной функции потерь. Остальные рамки опять же по IoU делятся на два типа: которые вообще не соответствуют и которые соответствуют, но есть клетка с лучшим IoU. Это вторая часть (надо правильно предсказать, где нет объектов). Общая функция является взвешенной суммой этих двух частей. Это было только для классификации еть объект/нет объекта. Наконец, есть еще кросс-энтропия собственно классов, к которым должен принадлежать объект. Суммируя все описанные слагаемые, получаем общую функцию потерь.

Кстати, это все можно посмотреть в коде в файлах `models.py` в методе `forward` класса `YOLOLayer` и `utils.py` в методе `build_targets`. Я постаралась написать комментарии с моим пониманием неочевидных моментов.

2.2. Transfer Learning

Я брала веса из уже упоминавшегося репозитория для первых 15 слоев (чтобы получить веса, надо сначала сбилдить содержимое, а потом использовать одну из программ). Смысл в том, чтобы та часть, которая извлекает фичи, уже что-то "знала" о том, что важно для распознавания. Результат получается лучше.

2.3. Датасет

Картинки я брала из OpenImages, так как еще по опыту прошлого семестра я поняла, что там больше всего изображений, которые хоть как-то соответствуют тому, на чем в итоге нужно распознавать. Кроме того, я сделала небольшой тест и выяснила, что обучать модель только на "правильных" изображениях, то есть фото дорог с машинами и т.д. – не очень хорошая идея, так как надо фильтровать изображения вручную.

- **Количество изображений**: 20000. Но на самом деле, мой финальный вариант обучался только на 5000. Потому что это был один из первых вариантов.
- **Разбиение**: 90% на обучение, 10% на валидацию.
- Для **тестирования** я взяла два видео с YouTube. Оказывается, довольно тяжело найти подходящие видео, поэтому там в основном машины. Но на одном есть автобус в начале! Я хотела снять что-то сама, но сейчас я в основном сижу дома, а в тех 2-3 поездках на машине я не нашла удачных ситуаций.

Более конкретно формат датасета описан в README моего репозитория, так что можете посмотреть еще там.

Вот мой notebook на Колабе для формирования датасета. Только там некоторые клетки дублируют функции, только с немного разным результатом. Причина этому в том, что (оказывается) гугл-диск выдает таймаут на доступ, если слишком много файлов в папке, поэтому мне пришлось по-быстрому менять так, чтобы раскладывать картинки по разным папкам.

2.4. Метрики

Я решила не использовать метрики, а оценивать визуально по следующим причинам:

- Метрика mAP, которую достигли для моей модели авторы, около 30 из 100. И это при большем времени обучения и лучших ресурсах. После того, как я обрезала модель, у меня точно не было шансов достичь и такого результата.
- По опыту как прошлого, так и этого семестра, больше mAP не значит лучший результат визуально. Особенно для маленьких отклонений.
- mAP, похоже, единственная метрика, по которой оценивают модели в Object Detection. По крайней мере, она используется во всех статьях, которые я видела, и во всех датасетах с челленджами.

2.5. Логи

Вот пример использованного мной лога. В основном для того, чтобы после постоянно случавшихся прерываний из-за лимитов Колаба знать, откуда начинать.К сожалению, я все время использовала одно и то же имя для файла, так что логи не сохранились.

3. Приложение

1. Кадр поступает на обработку, увеличиваем количество поступивших кадров.
2. В зависимости от остатка от деления на количество потоков, кадр забирает себе тот или иной поток.
3. Если это первый кадр из набора из 5 кадров, то обрабатываем. Иначе берем сохраненные результаты.
4. Получаем кадр с нарисованными на нем рамками и классами обратно.
5. Если это кадр ”в последовательности” (то есть его номер поступления на один больше номера предыдущего кадра), то:
   (a) Отображаем кадр.
   (b) Увеличиваем счетчик отображенных кадров.
   (c) Пока приоритетная очередь (по номеру кадра) не пуста и номера кадров идут последовательно, извлекаем из нее кадры и отображаем их.
6. Иначе кладем кадр в приоритетную очередь.

Каким образом обрабатывается каждый кадр:

1. Убираем альфа-канал и меняем размер на 416*416.

2. Прогоняем через нейросеть.
3. Делаем NMS (Non-Maximum Suppression), чтобы избавиться от лишних пересекающихся рамок.
4. Извлекаем класс с наибольшей оценкой. Извлекаем координаты и переводим их в пиксели на оригинальном кадре.
5. Рисуем рамку, название класса и оценку принадлежности к нему.
6. Передаем обратно (далее см. выше, пункт 4).

Также есть отчет, который я писала для универа. Там факты представлены таким образом, чтобы лишний раз никого не путать (потому что в прошлый раз я получила вопрос ”Где UML-диаграммы вашего приложения?”), но можно посмотреть следующее:

- 4.1.3 – краткое описание приложения с диаграммами.
- 4.3.2 и 4.3.3 – более многословно о вычислительном конвейере.
- 4.3.1 – и возможно про GUI.

Если чего-то не хватает, пишите. Если это вопрос о том, почему я не сделала так-то и так-то, то причины, скорее всего, следующие:

1. Я пробовала и было хуже.
2. Мне не хватило времени (все-таки в этом семестре на курсовую я потратила 130+ часов уже).
3. Я об этом не подумала (надеюсь нет).

Спасибо за внимание.