Дообучение классификатора

Влад Шахуро, Евгений Ляпустин, Федор Швецов, Владимир Гузов

Graphics & Media Lab
Vision Group

Обзор задания

В данном задании предлагается настроить обученную нейросеть для задачи классификации видов птиц. Для реализации используйте библиотеку PyTorch.

Описание задания

Дообучение (finetuning) — широко используемый метод обучения нейронной сети на небольших наборах данных. Глубокая сеть, обучаемая с нуля на небольших базах, подвержена переобучению. Чтобы это предотвратить, сеть инициализируется весами, обученными на большом наборе данных (например, ImageNet), а затем донастраивается с помощью градиентного спуска на целевой базе. В случае классификации верхний слой обученной сети заменяется на полносвязный слой с softmax-активацией и количеством выходов, равным количеству классов. Благодаря тому, что обученные на большой базе нижние слои запоминают универсальные, не привязанные к конкретным изображениям или классам, признаки, нейросеть быстрее сходится, а также в меньшей степени подвержена переобучению.

Базовая часть

Эту часть задания можно успеть выполнить на семинаре. Чтобы побыстрее обучить сеть, возьмите самую лёгкую предобученную модель MobileNetV2. Сделайте в ней обучаемыми последние 3-6 слоёв (В PyTorch за это отвечает метод слоя `requires_grad_`). Добавьте к этой модели `Global Average Pooling` слой и один полносвязный слой с softmax-активацией. Обучайте хотя бы 3 эпохи. Такую модель можно успеть обучить на семинаре и достичь точности 0.40. Не забудьте также сделать поменьше learning rate, чтобы не испортить веса предобученной модели.

Продвинутая часть

Дома вы уже можете поэкспериментировать и добиться лучшей точности.

1. В первую очередь вы можете обучать сеть большее количество эпох.
2. Вы можете добавить ещё один полносвязный слой, добавив к нему также `Batch Normalization` слой (его лучше вставлять перед активацией), чтобы не было переобучения.
3. Вы можете поэкспериментировать с количеством слоёв в предобученной модели, которые вы будете дообучать.
4. Так как датасет небольшой, вам может помочь аугментация данных, вам пригодится библиотека `Albumentations`.
5. Вы можете взять более тяжёлую модель, а также обучать сеть на картинках большего разрешения. Но помните про ограничение по памяти. Предобученные модели в PyTorch.
6. Чтобы понять, что ваша модель не переобучается, разделите датасет на обучающую и валидационную выборку.

Интерфейс программы, данные и скрипт для тестирования

Необходимо реализовать две функции: `train_classifier`, обучающую классификатор на основе предобученной нейросети, и `classify`, классифицирующую входные изображения с обученной моделью. Функция `train_classifier` возвращает готовую модель нейросети, а `classify` — словарь размером N, ключи которого — имена файлов, а значения — числа, означающие метку класса. Здесь N — количество изображений.

Для обучения алгоритма выдается публичная выборка размеченных изображений птиц. Разрешается использовать только эти данные для обучения, а также предобученные на ImageNet веса из `torchvision.models`. Другие внешние данные и библиотеки с готовыми моделями использовать нельзя.

Программа тестируется на двух тестах. В каждом из тестов нейросеть сначала обучается с флагом `fast_train=True` в функции `train_classifier`. Функция обучения с этим флагом должна работать недолго, не больше 15 минут. Для этого поставьте 1 эпоху обучения и несколько батчей. Обученная модель для тестирования не используется, этот этап необходим только для проверки работоспособности функции обучения. При работе функция не должна писать в какие-либо файлы (например, сохранять обученную модель), т.к. в проверяющей системе папка с решением монтируется только для чтения (read-only). Затем в первом тесте алгоритм тестируется на публичной выборке, во втором тесте — на скрытой выборке. Для тестирования загружается обученная модель `birds_model.ckpt`. Запуск функции на публичной обучающей выборке с флагом `fast_train=False` должен позволять воспроизвести сданную в `ckpt`-файле модель с небольшой погрешностью, связанной со случайностью в процессе обучения. Решения без функции обучения могут быть не засчитаны.

Результаты второго теста и итоговый балл скрыты до окончания срока сдачи задания. Итоговый балл считается по последней посылке с ненулевой точностью. Для уменьшения количества потребляемой памяти можно при обучении не загружать все изображения в память сразу, а открывать при генерации батча. Точность acc на скрытой выборке конвертируется в итоговый балл:

$acc \geqslant 0.85$ — 10 баллов

$acc \geqslant 0.83$ — 9 баллов

$acc \geqslant 0.80$ — 8 баллов

$acc \geqslant 0.75$ — 7 баллов

$acc \geqslant 0.70$ — 6 баллов

$acc \geqslant 0.65$ — 5 баллов

$acc \geqslant 0.60$ — 4 балла

$acc \geqslant 0.50$ — 3 балла

$acc \geqslant 0.40$ — 2 балла

$acc > 0$ — 1 балл

Полезные ресурсы

Страница стэнфордского курса про transfer learning