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. В этом задании и всех следующих заданиях курса для организации логики обучения рекомендуется использовать библиотеку Lightning.

Описание задания

Дообучение (fine-tuning) – 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 последний слой обученной сети заменяется на полносвязаный слой с softmax-активацией и количеством выходов, равным количеству классов. Благодаря тому, что обученные на большой базе слои запоминают универсальные, не привязанные к конкретным изображениям или классам, признаки, нейросеть быстрее сходится, а также в меньшей степени подвержена переобучению.

Базовая часть

Чтобы побыстрее обучить сеть, возьмите самую лёгкую предобученную модель `MobileNetV2` из статьи "MobileNetV2: Inverted Residuals and Linear Bottlenecks".

Возьмите из данный архитектуры блок `features`, добавьте после него `Global Average Pooling` слой и один полносвязный слой. В получившейся архитектуре сделайте обучаемыми последние 3-6 слоёв/блоков.

Обучайте хотя бы 3 эпохи. Не забудьте также сделать поменьше learning rate, чтобы не испортить веса предобученной модели. Подобный базовый подход должен обеспечить точность не меньше 40%.

Продвинутая часть

Дальше вы можете поэкспериментировать и добиться лучшей точности.

1. Чтобы убедиться, что ваша модель не переобучается, разделите датасет на обучающую и валидационную выборку.

   Учитывайте, что в данной задаче выделяется много различных видов птиц. Соответственно, обычное случайное разбиение на обучающую и валидационную выборки может привести к дисбалансу классов.

2. Вы можете обучать сеть большее количество эпох, использовать разные темпы обучения и оптимизаторы.

3. Вы можете поэкспериментировать с количеством замороженных/размороженных слоёв в предобученной модели. Также, количество размороженных слоев можно менять по ходу обучения.

4. Вы можете добавить другие дополнительные слои после предобученного блока в вашей сети.

5. Вы можете взять более тяжёлую модель, а также обучать сеть на картинках большего разрешения. Не забывайте об ограничениях по памяти и времени в тестовой системе.

   Для уменьшения количества потребляемой памяти можно при тестировании обрабатывать по одному изображению за раз.

6. Использование более тяжелых моделей при маленьком размере датасета может приводить к переобучению. Чтобы обойти эту проблему, вам могут помочь методы регуляризации обучения. Например: аугментация данных, `Batch Normalization`, `Dropout`, `Weight Decay`, `Label Smoothing` и другие.

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, ключи которого – имена файлов, а значения – числа, означающие метку класса.

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

Ваше решение проверяется 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 в тестовой системе ваше решение не должно подключаться к интернету, не должно писать какие-либо файлы на диск, не должно использовать графические ускорители. Нарушение этих требований приведет к ошибке выполнения. Также рекомендуется не использовать механизмы параллельной загрузки и предобработки данных в функции `classify`.

В первом тесте алгоритм тестируется на публичной выборке, во втором тесте – на скрытой выборке. Для тестирования загружается обученная модель `birds_model.pt`. Запуск функции на публичной обучающей выборке с флагом `fast_train=False` должен позволять воспроизвести сданную в `pt`-файле модель с небольшой погрешностью, связанной со случайностью в процессе обучения.

Решения без функции обучения или с функцией обучения не воспроизводящей целевую точность приравниваются к плагиату.

Результаты второго теста и итоговый балл скрыты до окончания срока сдачи задания. Итоговый балл считается по последней посылке с ненулевой точностью. Точность *acc* на скрытой выборке конвертируется в итоговый балл:

$acc \geqslant 0.88$ – 10 баллов

$acc \geqslant 0.86$ – 9 баллов

$acc \geqslant 0.83$ – 8 баллов

$acc \geqslant 0.80$ – 7 баллов

$acc \geqslant 0.75$ – 6 баллов

$acc \geqslant 0.70$ – 5 баллов

$acc \geqslant 0.65$ – 4 балла

$acc \geqslant 0.60$ – 3 балла

$acc \geqslant 0.50$ – 2 балла

$acc \geqslant 0.40$ – 1 балл