Глубокое обучение и вообще

Ульянкин Филипп

Посиделка 9: Организация DL-экспериментов

Проблемы при обучении нейросетей

- «Neural net training is a leaky abstraction» — Andrej Karpathy
- Знания архитектур, оптимизаторов порой недостаточно для получения хорошей модели
- Универсально наилучшего решения не бывает
- Важна точка начала экспериментов и инкрементальные улучшения

Andrej Karpathy: http://karpathy.github.io/2019/04/25/recipe/
Максим Рябинин: https://github.com/aosokin/dl_cshse_ami/blob/master/2021-fall/lectures/DL21-fall-lecture5-bestpractices.pdf

Перед началом

- Используйте проверенные временем стандарты
- Вместо своих моделей — архитектуры из популярных публикаций (ResNet в зрении, ELMo/Transformer в текстах) и репозиториев [1,2,3]
- Adam со стандартным LR без расписания обойти нелегко
- Сложные функции потерь/аугментации лучше отложить
- Первые запуски на небольших датасетах, подвыборке или синтетике

[1] https://github.com/pytorch/vision
[2] https://github.com/huggingface/transformers/
[3] https://github.com/pytorch/fairseq

Как искать ошибки

- Чтобы легче находить ошибки, снизьте число факторов влияния
- Баги могут быть как в определении и обучении модели, так и в проверке качества (даже в загрузке данных)
- В меньшем масштабе можно быстрее итерироваться и находить проблемы
- Пробуйте прогнать код на одном батче и оценить, насколько адекватные результаты вы получили: есть ли сходимость, есть ли переобучение на валидации, адекватно ли меняются метрики
- Визуализируйте всё, что можете: метрики, примеры работы модели
- DL-код — всё ещё код: полезно писать unit-тесты

Типичные ошибки: модели

- Использование ad-hoc архитектур, когда не надо
- Использование нестабильных/сложных функций потерь вместо кросс-энтропии в классификации
- Использование устаревших функций активации в глубоких сетях (сигмоида, тангенс)
- Плохая инициализация: нули/константы вместо Glorot/He

Типичные ошибки: данные

- Отсутствие аугментации/использование некорректной аугментации, разные аугментации при обучении и валидации (исключение — random crop)
- Если используете предобученные модели, препроцессинг данных должен быть максимально похожим
- Считывать все данные сразу, используйте Dataset/DataLoader

Типичные ошибки: обучение

- Не забывайте делать zero_grad :)
- Переключайте model.train()/model.eval() в нужных фазах обучения
- Делайте чекпойнты, при них сохраняйте также параметры оптимизатора optim.state_dict() и расписания
- Функция потерь должна быть максимально близка к метрике, которую вы оптимизируете

Организация экспериментов

- Тестируйте за один раз только одно изменение, чтобы понимать влияние каждого фактора по отдельности

- На ранних стадиях не обязательно учить до сходимости и использовать всю выборку целиком

- Ведите лог всех экспериментов

- Примеры инструментов для лога экспериентов:
  https://www.wandb.com/
  https://www.comet.ml/
  https://neptune.ai/
  https://dvc.org/
  https://mlflow.org/

Как улучшать качество?

- Функция потерь должна быть максимально близка к метрике
- Начните с небольших экспериментов и масштабируйтесь, когда всё отлажено
- Работа с данными (количество, качество, предобработка) зачастую приносит гораздо больше эффекта, чем перебор архитектур и оптимизаторов
- Архитектуры влияют существенно, но учитывайте свои ресурсы
- Занимайтесь оптимизацией гиперпараметров в самую последнюю очередь
- Размер батча важен (ряд моделей иначе просто не учится)

Выводы

- Пользуйтесь проверенными техниками и опытом других людей
- Начните с небольших экспериментов
- Когда всё протестировано, можно масштабироваться
- Отслеживайте все доступные метрики
- Тестируйте одно изменение за раз
- Сохраняйте код/конфигурацию всех экспериментов и их результаты