Глубинное обучение
Лекция 5: организация DL-экспериментов

Лектор: Максим Рябинин

ФКН ВШЭ, 2020

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

Проблемы при обучении нейросетей

- «Neural net training is a leaky abstraction» — Andrej Karpathy [1]
- Знания архитектур, оптимизаторов порой недостаточно для получения хорошей модели
- No free lunch: универсально наилучшего решения не бывает
- Важна точка начала экспериментов и инкрементальные улучшения

[1] http://karpathy.github.io/2019/04/25/recipe/

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

Порядок действий

- Чтобы проще 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
- DL-код — всё ещё код: unit-тесты полезно писать

1. Переобучение на одном батче
2. Переобучение (или хотя бы сходимость) на обучающей выборке
3. Адекватное качество на валидации

Типичные ошибки: модели

- Использование ad-hoc архитектур, когда это не требуется
- Использование нестабильных/сложных функций потерь вместо кросс-энтропии в классификации
  Здесь же: softmax->NLLLoss, sigmoid вместо softmax, активации перед softmax
- Плохая инициализация: нули/константы вместо Glorot/He/равномерной
  Rule of thumb: для классификации стартовый loss $\approx \log(K)$ (число классов)

Типичные ошибки: данные

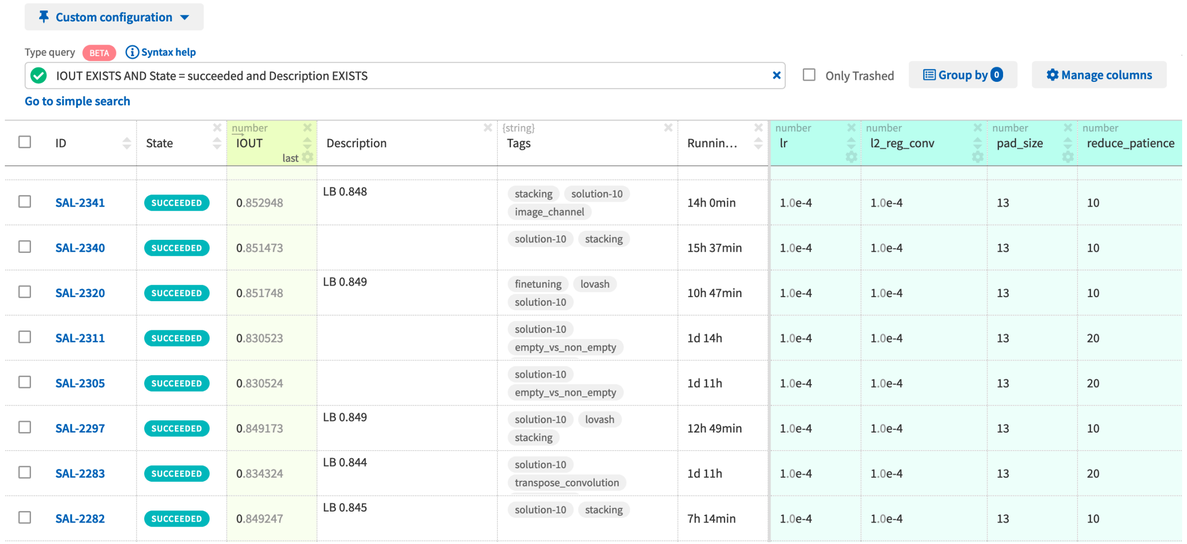
- Отсутствие аугментаций
- Использование некорректных аугментаций
  Важно: визуализируйте данные непосредственно на входе в сеть!
- Если используете предобученные модели, препроцессинг должен быть максимально похожим
- Разные аугментации при обучении и валидации (исключение — random crop)
- Считывать весь датасет сразу: используйте Dataset/DataLoader

Типичные ошибки: обучение

- Не забывайте делать zero_grad :)
- Переключайте model.train()/model.eval() в нужных фазах обучения
  anecdote: иногда стоит обновлять статистики batchnorm на тестсете
- loss.item() перед сохранением значения между итерациями
- Если сохраняете чекпойнты для дообучения, сохраняйте также параметры оптимизатора (optim.state_dict()) и расписания

Организация экспериментов

- Важно вносить **только одно** изменение за раз
- На ранних стадиях необязательно учить до сходимости!
- Ведите лог всех экспериментов
- Есть ряд готовых инструментов [1,2,3]



ID	State	number IOUT last	Description	{string} Tags	Runnin…	number lr	number l2_reg_conv	number pad_size	number reduce_patience
SAL-2341	SUCCEEDED	0.852948	LB 0.848	stacking solution-10 image_channel	14h 0min	1.0e-4	1.0e-4	13	10
SAL-2340	SUCCEEDED	0.851473		solution-10 stacking	15h 37min	1.0e-4	1.0e-4	13	10
SAL-2320	SUCCEEDED	0.851748	LB 0.849	finetuning lovash solution-10	10h 47min	1.0e-4	1.0e-4	13	10
SAL-2311	SUCCEEDED	0.830523		solution-10 empty_vs_non_empty	1d 14h	1.0e-4	1.0e-4	13	20
SAL-2305	SUCCEEDED	0.830524		solution-10 empty_vs_non_empty	1d 11h	1.0e-4	1.0e-4	13	20
SAL-2297	SUCCEEDED	0.849173	LB 0.849	solution-10 lovash stacking	12h 49min	1.0e-4	1.0e-4	13	10
SAL-2283	SUCCEEDED	0.834324	LB 0.844	solution-10 transpose_convolution	1d 11h	1.0e-4	1.0e-4	13	20
SAL-2282	SUCCEEDED	0.849247	LB 0.845	solution-10 stacking	7h 14min	1.0e-4	1.0e-4	13	10

[1] https://www.wandb.com/
[2] https://www.comet.ml/
[3] https://neptune.ai/

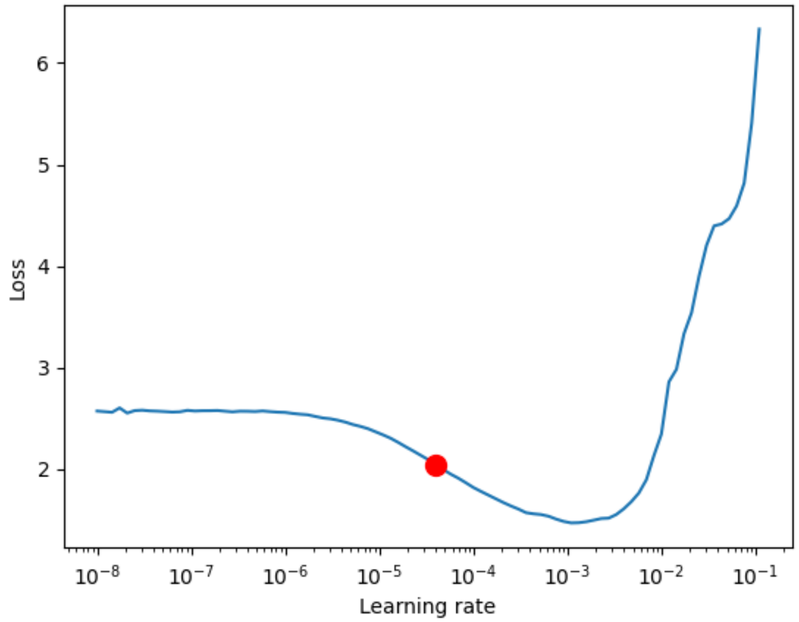
Как улучшать качество?

- Работа с данными (количество, качество, предобработка) зачастую приносит гораздо больше эффекта
- Функция потерь должна быть максимально близка к метрике
- Архитектуры влияют существенно, но учитывайте свои ресурсы
- Размер батча важен (ряд моделей иначе просто не учится)

  hint: используйте gradient accumulation!
- Оптимизация гиперпараметров в самую последнюю очередь!
  (random search > hyperparameter tuning > grid search)

LR range test aka LRFinder

- Эвристика для подбора learning rate [1,2,3]
- Увеличиваем learning rate после каждого батча, остановка после взрыва значений loss-функции
- Реализовано в off-the-shelf DL-фреймворках [4,5,6], но несложно реализовать самому [1]
- Аналогично можно искать оптимальное значение для momentum



[1] https://sgugger.github.io/how-do-you-find-a-good-learning-rate.html

[2] https://arxiv.org/abs/1506.01186

[3] https://arxiv.org/abs/1803.09820

[4] catalyst.dl.callbacks.scheduler.LRFinder

[5] fastai.Learner.lr_find

[6] pytorch_lightning.tuner.lr_finder.lr_find

Выводы

- Пользуйтесь проверенными техниками и опытом других людей
- Начните с небольших экспериментов; масштабируйтесь только когда всё отлажено
- Тестируйте одно изменение за раз, чтобы понимать влияние каждого фактора по отдельности
- Отслеживайте все доступные метрики, сохраняйте результаты всех экспериментов