

# Как решать задачи

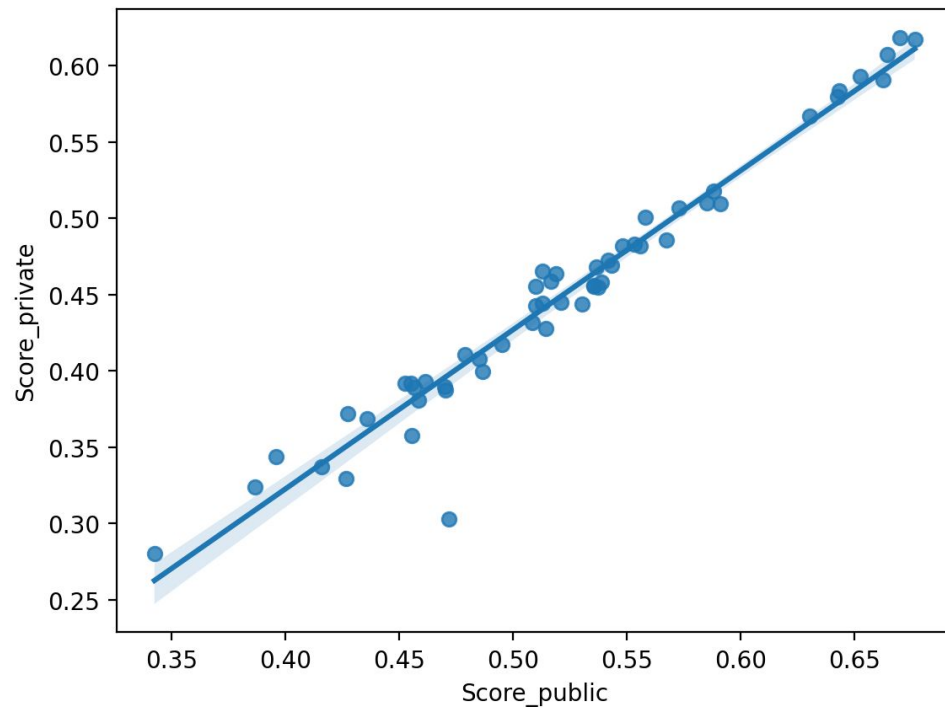
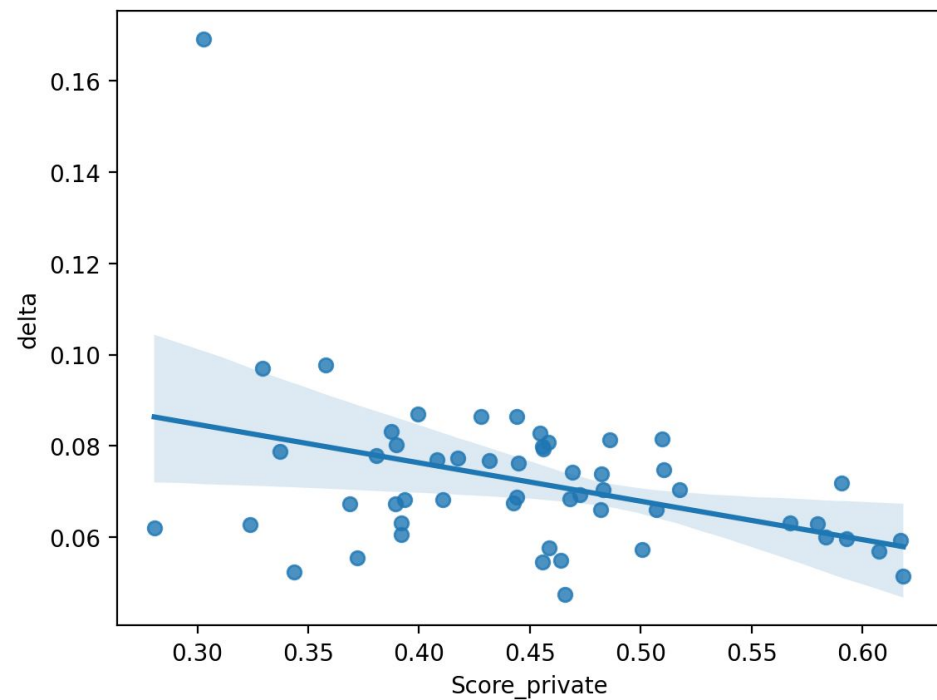
Без боли и страданий

# План

1. Результаты соревнования
2. Доклады (5 минут презентация + 3-5 вопросы)
3. Best practices

# Лидерборд HW2

	Median	Max
Public	51.55	67.65
Private	44.97	61.85



# Пайплайн решения ML-задачи

1. Постановка задачи

2. Данные

3. Модель

4. Обучение

5. Оптимизация

6. Деплой и поддержка

- Чаще всего в виде абстрактной проблемы
- Перевод с продуктового языка на метрический: как померить качество получившегося решения

# Пайплайн решения ML-задачи

1. Постановка задачи

2. Данные

3. Модель

4. Обучение

5. Оптимизация

6. Деплой и поддержка

- ТЗ на разметку
- Шумная разметка / domain adaptation
- Синтетика
- Препроцессинг максимально похож на претрейн
- train и val аугментации должны быть похожи (кроме ТТА)
- Аугментации адекватны таргету

# Пайплайн решения ML-задачи

- |                       |   |
|-----------------------|---|
| 1. Постановка задачи  | • Беспольный Ad-Hoc   |
| 2. Данные             | • Baseline: сильный переобучится под шумную разметку, слабый не перформит |
| 3. Модель             | • Softmax $\leftrightarrow$ NLLLoss, сложные активации и лоссы            |
| 4. Обучение           |   |
| 5. Оптимизация        |   |
| 6. Деплой и поддержка | • Инициализация нулями (CE: $\sim \log K$ )                               |

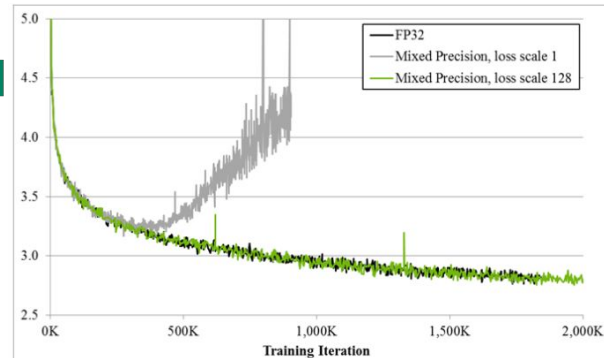
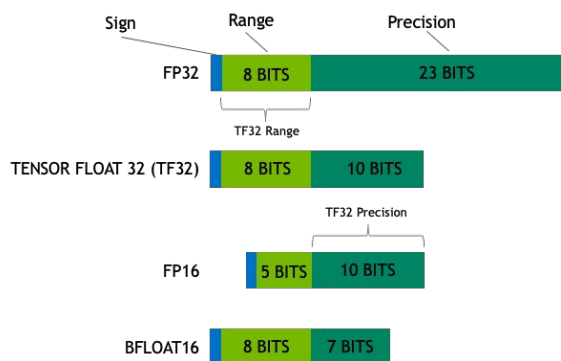
# Пайплайн решения ML-задачи

- |                       |   |
|-----------------------|---|
| 1. Постановка задачи  | ● One Batch Overfitting   |
| 2. Данные             | ● zero_grad   |
| 3. Модель             | ● model.train()/model.eval() (BatchNorm)  |
| 4. Обучение           | ● loss.item() (или detach) между итерациями   |
| 5. Оптимизация        | ● torch.save(optim.state_dict())  |
| 6. Деплой и поддержка | ● 1 изменение за раз  |
|                       | ● Не учить короткие экспы до сходимости   |
|                       | ● Адекватность прироста качества (не шум)   |
|                       | ● Лосс как прокси метрики   |
|                       | ● <a href="#">LRFinder</a> , ScheduleFree ( <a href="#">1</a> , <a href="#">2</a> ) |
|                       | ● НПО: <a href="#">свипы</a> в wandb  |
|                       | ● Всё нужно логгировать и отсматривать  |

# Пайплайн решения ML-задачи

1. Постановка задачи
2. Данные
3. Модель
4. Обучение
5. Оптимизация
6. Деплой и поддержка

- `torch.compile()/TensorRT/ONNX`
- Прунинг/Квантизация/Дистилляция (позже)
- [Mixed precision training](#)





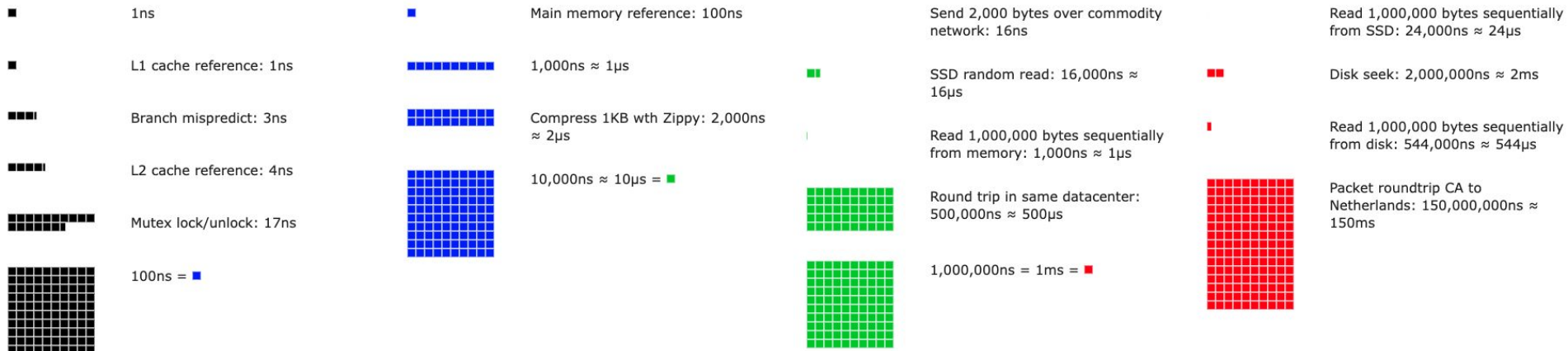
# Пайплайн решения ML-задачи

1. Постановка задачи
  - Как в SWE: unit-тесты, мониторинг, A/B
2. Данные
  - Data Drift, часто детектится отдельной моделью
3. Модель
  - Унифицированный препроцессинг
4. Обучение
  - Эффективный (асинхронный) инференс
5. Оптимизация
  - Постпроцессинг и чистка предиктов,
6. Деплой и поддержка
  - [Active Learning](#)
  - Профилирование кода

# Ботлнеки в загрузке данных

Latency Numbers Every Programmer Should Know

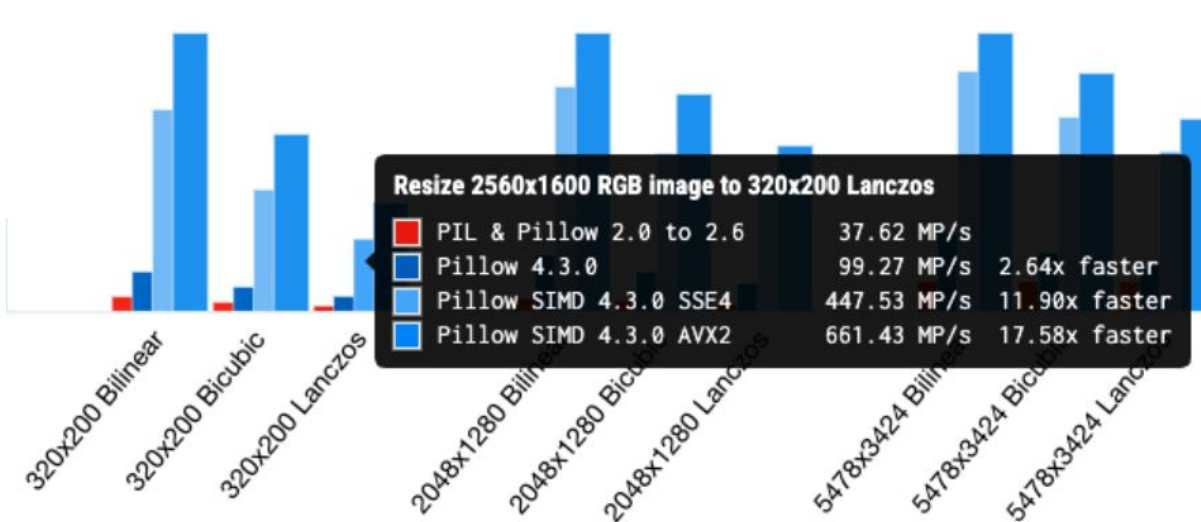
2020



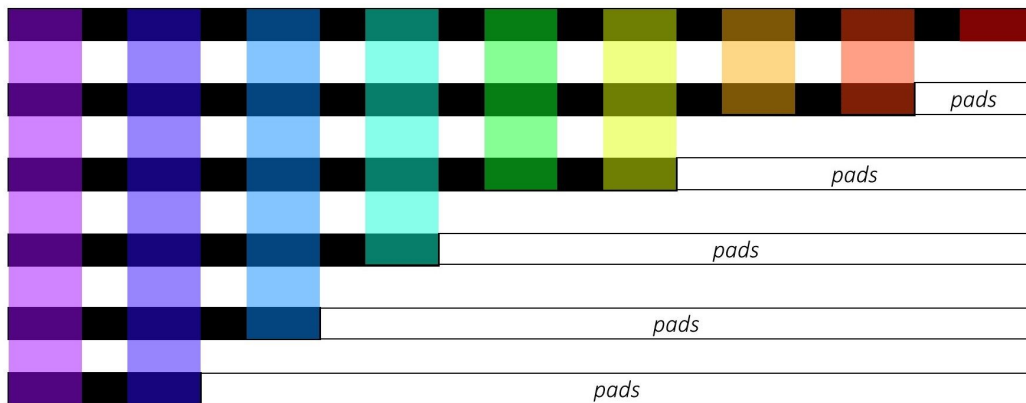
[https://colin-scott.github.io/personal\\_website/research/interactive\\_latency.html](https://colin-scott.github.io/personal_website/research/interactive_latency.html)

# Препроцессинг

- Что и как читать
- `num_workers > 1` (кэп)
- [Pillow-SIMD](#): drop-in замена PIL
- [Faster JPEG decoders](#)
- CPU bound: перенос тяжёлых аугов на GPU ([DALI](#)) или в офлайн
- Последовательности: регенерация и умный паддинг



*Padded sequences sorted by decreasing lengths*



# Профилирование

- White-box оптимизация пайплайна обучения и инференса
- [DeepSpeed FLOPs breakdown](#)
- [PyTorch Profiler](#)

