Optimization

Page • 1 backlink • uni

Всегда ищем баланс между throughput и latency (ранее упоминали)

На данный момент чем больше модель, тем дольше инференс и тем более быстрое железо и алгоритмы приходится изобретать (уменьшить модель во вред качеству – не вариант)

Сжатие модели (очень актуальная стезя ресерча)

- квантизация (BF16, INT8, INT4, ...) – больше размер батча или просто ускорение инференса, но в качестве точно теряем
    - post-training quantization
    - quantization-aware training
- knowledge distillation
    - плюсы: несложно выстроить пайплайн обучения модели-ученика, для ученика можно использовать другую архитектуру
    - минусы: качество ученика сильно зависит от архитектуры учителя и того, насколько хорошо он обучен
- pruning – отрезаем части модели, убиваем самые незначимые параметры (либо в ноль, либо в среднее значение)
- факторизация – матрицу $N \times N$ превращаем в $(N \times F) \cdot (F \times N)$, например, в сверточных сетях

На примере BERT: DistillBERT, Dynamic Share Input, Quantization в сумме ускорили инференс в 30 раз (пример, скорее, исторический, поскольку сейчас эта модель неактуальна)

Для генеративных сетей

- speculation
- dynamic batching (по тэгам continuous batching, inflight batching)

Инженерные оптимизации – фреймворков много (даже несмотря на то, что PyTorch лидирует), устройств тоже: каждый нужно оптимизировать под каждое

Подход такой: модель из фреймворка конвертируется в промежуточное представление (например, вычислительный граф) и уже из него – в

машинный код

- (?)
- operator fusion
- graph optimization

Реализация инженерных оптимизаций ad hoc для устройства сложно, поэтому есть исследования в сторону поиска обобщенного подхода

- AutoScheduler – cuDNN autotune, TVM, TensorRT

Одно из современных направлений: оптимизации для применения в браузерах (есть JavaScript и WebAssembly)