ВМК

Задание 1. Сжатие нейронных сетей для предсказания временных рядов

Эффективные системы машинного обучения, 2024

Начало выполнения задания: 20 октября 2024 года, 23:00.
Жесткий дедлайн: **10 ноября 2024 года, 23:59.**

Формулировка задания

Данное задание направлено на ознакомление с основными алгоритмами сжатия нейронных сетей, а также на изучение архитектур для работы с временными рядами. В задании необходимо:

1. Обучить базовые модели для предсказания временных рядов (см. первый пункт раздела эксперименты)
2. Сравнить различные алгоритмы сжатия нейронных сетей (см. соответствующие пункта раздела эксперименты)
3. Оформить закрытый GitHub репозиторий с кодом экспериментов. Репозиторий должен содержать реализации всех использованных методов, ссылки на источники, если код не был написан самостоятельно, а также код/ноутбуки для получения таблиц и графиков, использованных в отчете.
4. Написать отчёт объемом до 5 страниц о проделанной работе (формат PDF). Отчет должен содержать следующие разделы: обзор методов предсказания временных рядов, введение и постановка задачи (зачем и почему мы хотим сжимать сети), описание основных концепций сжатия, раздел с экспериментами, вывод (нужно выбрать, какой алгоритм сжатия работает лучше с точки зрения количества бит на параметр)

Список экспериментов

Эксперименты этого задания необходимо проводить на датасете `traffic-96-96`.

1. **Обучение базовых моделей.** Обучите модели для решения поставленной задачи с помощью Time Series Library. Модели: `Non-Stationary Transformer (14M)`, `Autoformer (1.2M)`, `TimesNet (2.4M)`. Доложите метрику и лосс на тестовой части датасета, а также количество бит в моделях. Инструкция, как обучать локально:
   - склонировать проект Time Series Library,
   - создать виртуальную среду Python 3.8 и установить pip-пакеты из файла `requirements.txt`,
   - обновить торч командой `pip install -upgrade torch`,
   - скачать датасет traffic и поместить в папку `./dataset` в корне проекта,
   - скрипты обучения для `Non-Stationary Transformer` взять отсюда. Например, для `Autoformer` скрипт тут.

   Мы не накладываем ограничений на способ настройки виртуальной среды. Данный способ представлен для ознакомления, повторять именно его необязательно.
2. **Дистилляция знаний.** В качестве модели учителя используйте `Non-Stationary Transformer (14M)`. В качестве учеников – `Autoformer (1.2M)`, `TimesNet (2.4M)`. Используйте **не менее 2х** алгоритмов дистилляции. Сделайте выводы. Для этого могут помочь следующие вопросы. Дает ли дистилляция прирост в метриках? Какой из алгоритмов дистилляции работает лучше? Какую модель следует использовать в качестве ученика? Почему?
3. **Спарсификация.** Реализуйте один и более алгоритм структурной и один и более алгоритм неструктурной спарсификации в парадигме post-training. Примените его для модели `Non-Stationary Transformer (14M)`. Используйте следующие уровни разреженности: 50 % и 75%.Проведите fine-tuning каждой из полученных моделей. Сделайте выводы. Для этого могут помочь следующие вопросы. Какой алгоритм спарсификации лучше? Какой уровень разреженности лучше? Помогает ли fine-tuning? Почему?

4. **Квантизация.** Реализуйте не менее одного алгоритма post training quantization (за исключением uniform квантизации) (сюда же относятся алгоритмы one shot квантизации, такие как GPTQ). Реализуйте процедуру quantization aware training (за исключением STE). Примените полученные алгоритмы к модели `Non-Stationary Transformer (14M)` для ее сжатия в 8 и 4 бита. Сделайте выводы. Для этого могут помочь следующие вопросы. Какой уровень битности лучше использовать? Помогает ли QAT? Почему?

5. **PEFT.** Реализуйте логику QLoRA/QDoRA для модели, сквантизованной в 4 бита. Найдите такое минимальное значение ранга адаптеров, что:

   - Параметров с точки зрения количества бит станет примерно столько же, сколько в модели сжатой в 8 бит.
   - Перформанс такой модели станет лучше, чем у модели сжатой в 8 бит.

   Сделайте выводы. Для этого могут помочь следующие вопросы. Имеет ли смысл использовать LoRA адаптеры для квантизованных моделей?

6. **Сравнение полученных моделей**. Предоставьте визуальное сравнение полученных моделей с точки зрения перформанса и количеств бит. Сделайте выводы, какие алгоритмы сжатия работают лучше других. Выберите лучшую модель с точки зрения соотношения перформанса и количества бит.

Требования, советы и замечания

- Важное замечание. Реально сжатия мы не используем, так как это сложно с точки зрения реализации. Поэтому мы сжимаем модельки до n-бит, но вычисляем по прежнему в fp16. Из-за этого реального ускорения вы не увидите, но можете попробовать это сделать в бонусе.
- Также мы считаем, что в случае спарсификации удаленные параметры занимают 0 бит. То есть при спарсификации на 50% количество бит у модели равно половине бит относительно исходной модели.
- Задание выполняется в командах до 4х человек.
- В отчете должен быть раздел, в котором описан вклад каждого участника команды.
- Текст вклада должен соответствовать истории коммитов на GitHub.
- В задании **запрещается** пользоваться библиотечными реализациями алгоритмов сжатия/LoRA адаптеров. Для всего остального можно пользоваться библиотеками (то есть для работы с временными рядами использовать Time-Series-Library можно)
- В задании **разрешается** пользоваться чужим кодом (за исключением пункта ниже), но нужно оставлять ссылки на источники. Если вы взяли кусок кода библиотеки и смогли его запустить без всех остальных зависимостей, то это не попадает под прошлый пункт.
- Брать код других команд, выполняющих данное задание **запрещается**.
- **Эксперименты должны быть воспроизводимы**. Поэтому рекомендуется зафиксировать все сиды, а также приложить файл для установки используемого окружения.
- Если какой-то пункт задания сделать не получилось, то об этом следует написать в отчете и сказать почему, это лучше, чем ничего не сделать.
- Генерация отчета с использованием LLM не запрещается, но лучше потом перечитать текст и попытаться скрыть следы использования БЯМ.
- Если берете готовый код, то желательно это делать из официальных источников, а не использовать случайный код из GitHub.
- В отчете должны быть ссылки на статьи, откуда вы взяли метод.
- Преподаватели оставляют за собой право в одностороннем порядке пополнить список запретов при обнаружении вопиющих случаев нечестного выполнения задания.

Бонусная часть

Бонусные исследования могут сделать так, что проверяющие закроют глаза на проблемы в основной части. Чем больше бонусных исследований *качественно* сделано, тем больше косяков проверяющие готовы простить.

1. Реализуйте логику fast-forwarding для обучения LoRA адаптеров для квантизованных моделей. Проведите сравнение с обучением с помощью обычного Adam с точки зрения скорости и качества. Сделайте выводы.

2. Найдите в литературе методы сжатия, созданные специально для моделей, работающих с временными рядами. Реализуйте их, добавьте полученные модели в сравнительный анализ.

3. Заставьте сжатые модельки правильно работать с железом. Например, пока что мы просто симулировали пониженную битность, а теперь предлагается сделать так, чтобы она стала реальностью. Проведите эксперименты со скоростью работы полученных моделей после того, как они подружились с железом. *Замечание.* В этом пункте **разрешается** использовать готовые библиотеки.

4. Любые другие исследования, согласованные с преподавателями.