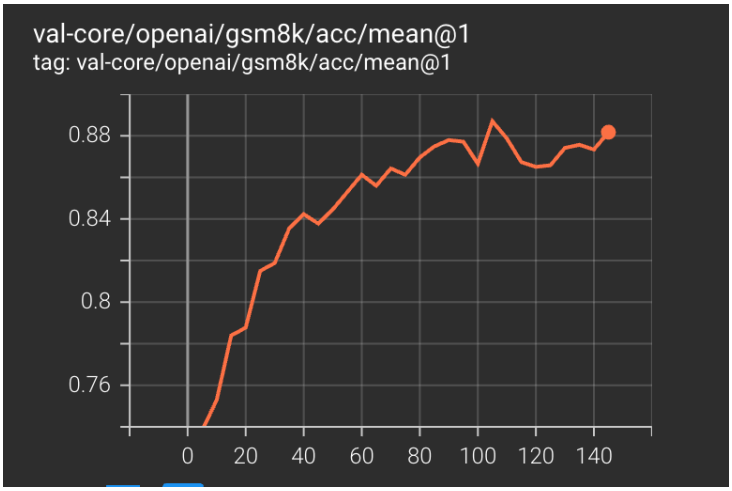
Введение

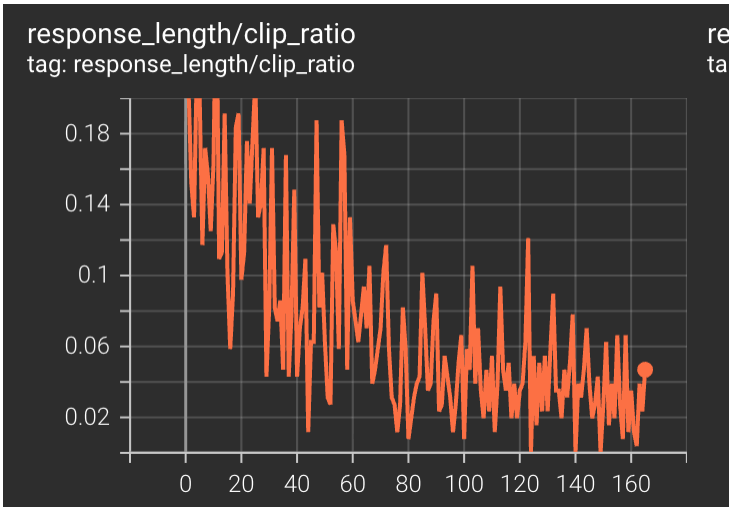
Очень давно хотел позапускать эксперименты в новой версии VERL, но никак не получалось. В добавок к этому, когда-то был плохой опыт работы с unsloth и я не хотел использовать jupyter notebook. Также хотелось использовать более сильную модель Qwen3-1.7B с thinking. Учитывая всё это я решил сделать ДЗ используя VERL. Надеюсь это не сильно усложнит проверку и не приведет к 0 баллов за ДЗ...

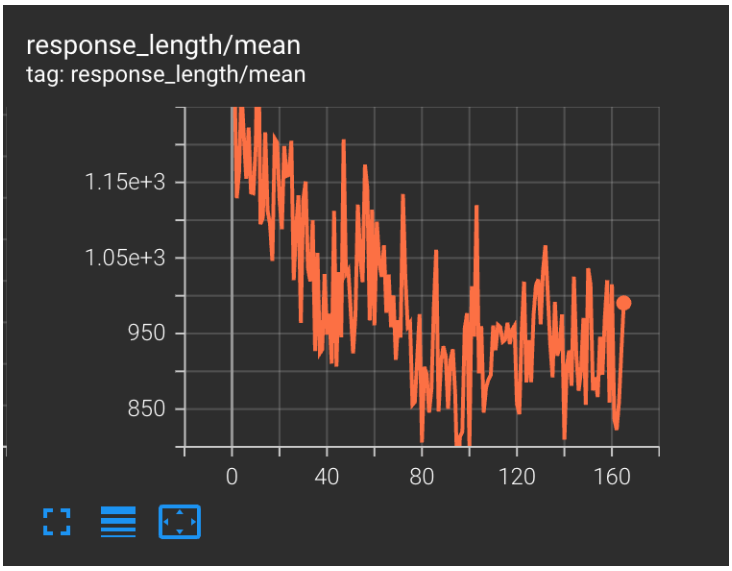
Не связано с главной задачей

Сначала я решил запустить хотя бы какое-то обучение на qwen3-1.7B (далее буду обозначать qwen3-1.7B как модель / llm) на VERL. Взял gsm8k и после продолжительного дебага получилось запустить. max_response_len = 2048



Из интересного, графики снизу показывают, что модель на самом деле во многом не училась решать матешу, а училась коротко размышлять, чтобы влезать в max_response_length. Такое я встречал раньше, при обучение готовой qwen3 на очень стандартных задачах.





(чтобы это не звучало так голословно, можно запустить обучение с max_response_len = 16к и посмотреть, что будет. Но я не успеваю)

Запустить его можно так: training/scripts/run_grpo_qwen3_1p7b_gsm8k_think_4gpu.sh (только там надо изменить CUDA_VISIBLE_DEVICES, gpu_memory_utilization, n_gpus_per_node под железо)

Реализация env

Потратил какое-то время в попытке придумать что-то интересное, но несильно преуспел в этом. Решил реализовать что-то достаточно простое и сконцентрировать на процессе обучения и изменения тактики решения задач.

Сразу скажу: идею и архитектуру решения сделал полностью я. Но для реализации какой-то части кода я использовал ChatGPT.

Я решил сделать такой environment: модель получает задачу в виде небольшой программы на регистровой машине и должна выдать только финальное значение R0. В задаче есть фиксированный набор инструкций (SET, ADD, XOR, MOD, SWP, JNZ, NOP, HALT), регистры считаются по модулю, и есть лимит шагов, чтобы не зациклиться.

Сложность я сделал по уровням (1–10): с ростом уровня увеличиваются число регистров, длина программы и сложность циклов. Для каждой сгенерированной задачи есть точный ground truth из симулятора, поэтому reward простой и прозрачный: 1, если ответ совпал, иначе 0. Формат ответа тоже зафиксировал (...</think> и потом итог R0=...), чтобы можно было отдельно контролировать рассуждение и финальный ответ.

Почему выбрал это: легко генерировать много разных задач, легко валидировать, а главное была теория что при маленькой максимальной длине рассуждений модель начнет как-то эффективно выполнять эти программы: не просто идти шаг за шагом, а пропускать какие-то шаги, считать какие-то шаги внутри слоёв чтобы снизить количество токенов на ответ.

Сначала я решил проверить насколько хорошо модель может решать такие задачи. Для этого я реализовал environment тут: https://github.com/sekopylov/grpo-algorithmic-reasoning/blob/main/hw_2/code_and_results/pipeline.py

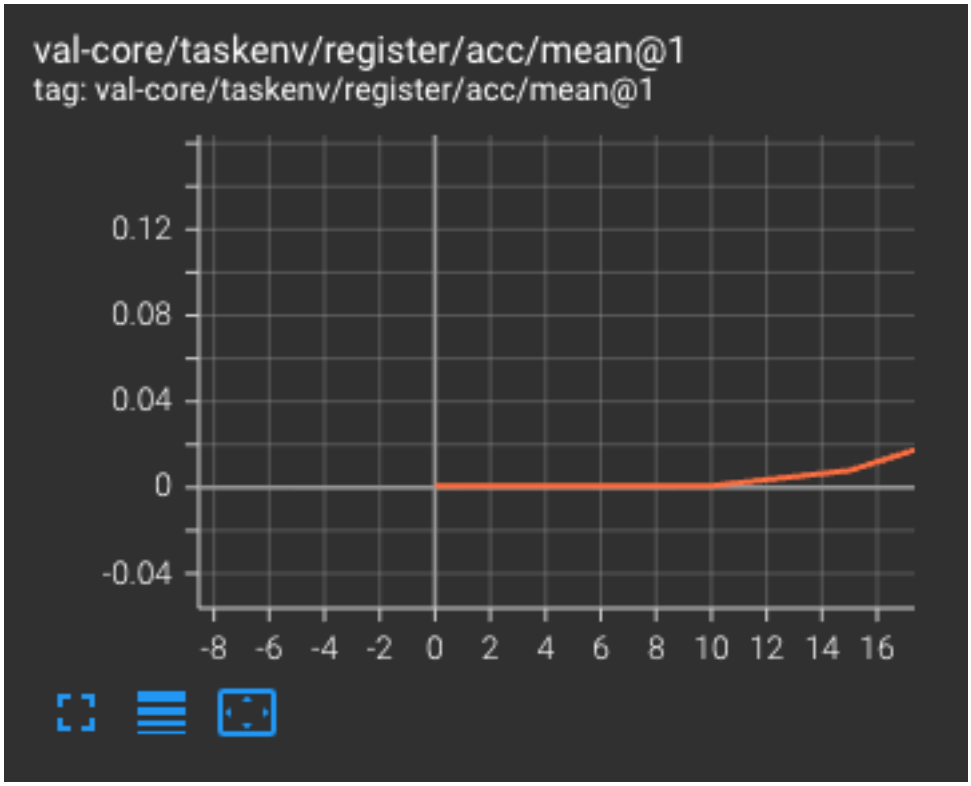
Нагенерил данных, проинференсил и посмотрел результаты (https://github.com/sekopylov/grpo-algorithmic-reasoning/blob/main/hw_2/code_and_results/results/summary.json и https://github.com/sekopylov/grpo-algorithmic-reasoning/blob/main/hw_2/code_and_results/show.py)

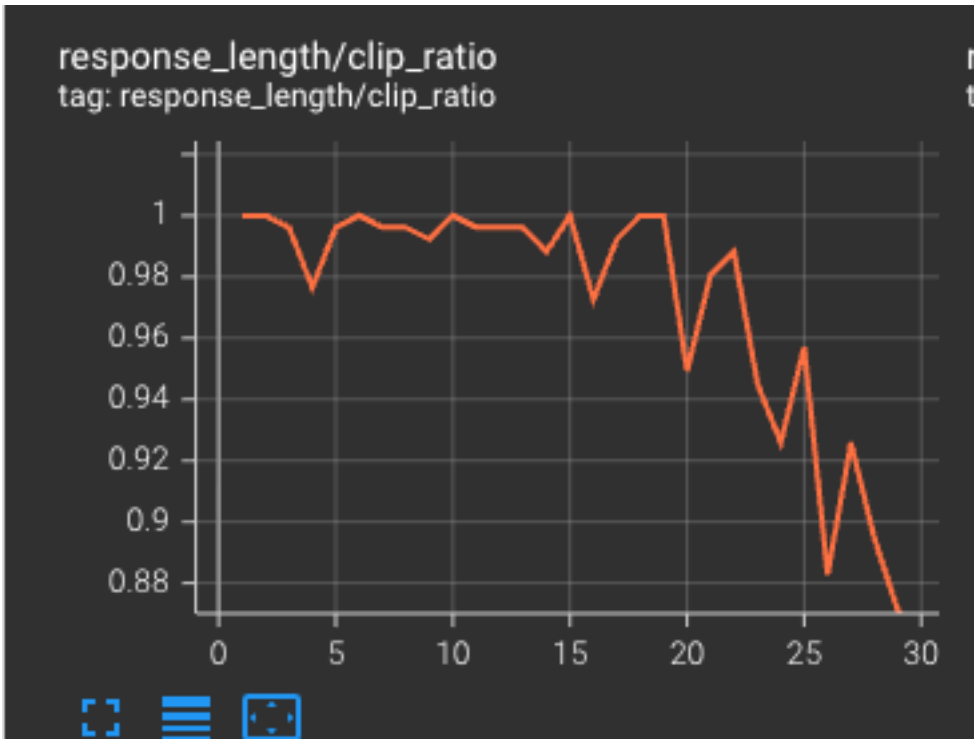
Задачи оказались слишком сложные, но какие-то задачи модель решала, значит сигнал для обучения был. Времени переделовать не оставалось.

Далее скопипастил код из pipeline.py в https://github.com/sekopylov/grpo-algorithmic-reasoning/blob/main/training/scripts/taskenv_preprocess.py, и добавил нормальную генерацию датасетов для обучения.

Потом поставил обучение с max_response_length=768 (run_grpo_qwen3_1p7b_taskenv_think_4gpu.sh).

Оказалось, что этой макс длины слишком мало, это явно следует из этих графиков:





Поэтому параллельно поставил второй эксп с max_response_length=2048 (run_grpo_qwen3_1p7b_taskenv_think_4gpu_2048.sh)

Обучение шло долго В целом идея была посмотреть насколько хорошо LLM может сжимать шаги при маленькой максимальной длины ответа. Однако стало очевидно, что при маленьком max_response_len, LLM всегда будет получать reward = 0 и не обучаться. Поэтому чтобы всё же заставить llm, я решил ввести награду за длину ответа. Для этого я сделал training/scripts/reward_taskenv_len_wrapper.py и запустил training/scripts/ run_grpo_qwen3_1p7b_taskenv_think_4gpu_2048_len.sh. (только для правильных траекторий)

Посмотреть метрики любого из обучений можно через tensorboard. Пример команды для запуска:

tensorboard –logdir training/grpo_qwen3_1p7b_taskenv_think_4gpu_2048_len_20260227_023535/
tensorboard –host 127.0.0.1 –port 12921

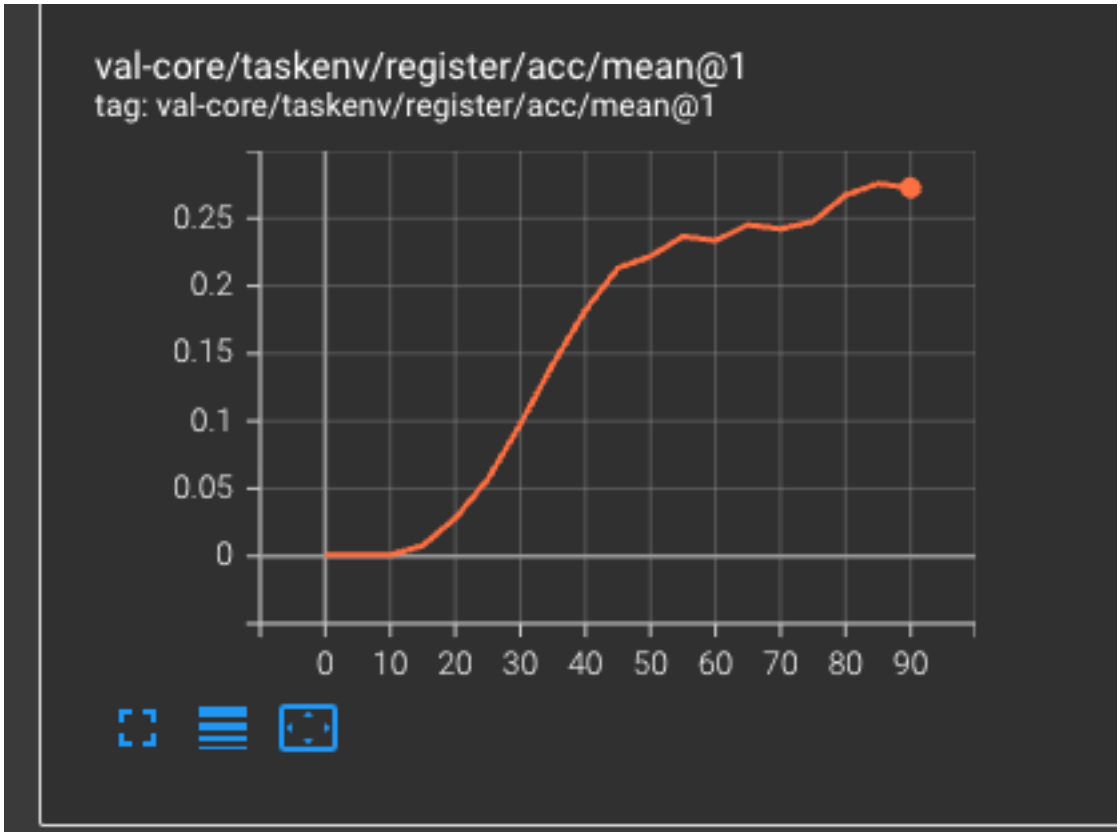
В GRPO_768_validation_trajectories.txt можно найти реальные примеры ответов модели на валидации по шагам обучения. По ним видно, что при max_response_length=768 в начале обучения модель часто уходит в длинное рассуждение и не успевает дописать финальный R0=…, из-за чего ответ обрезается и reward получается нулевой.

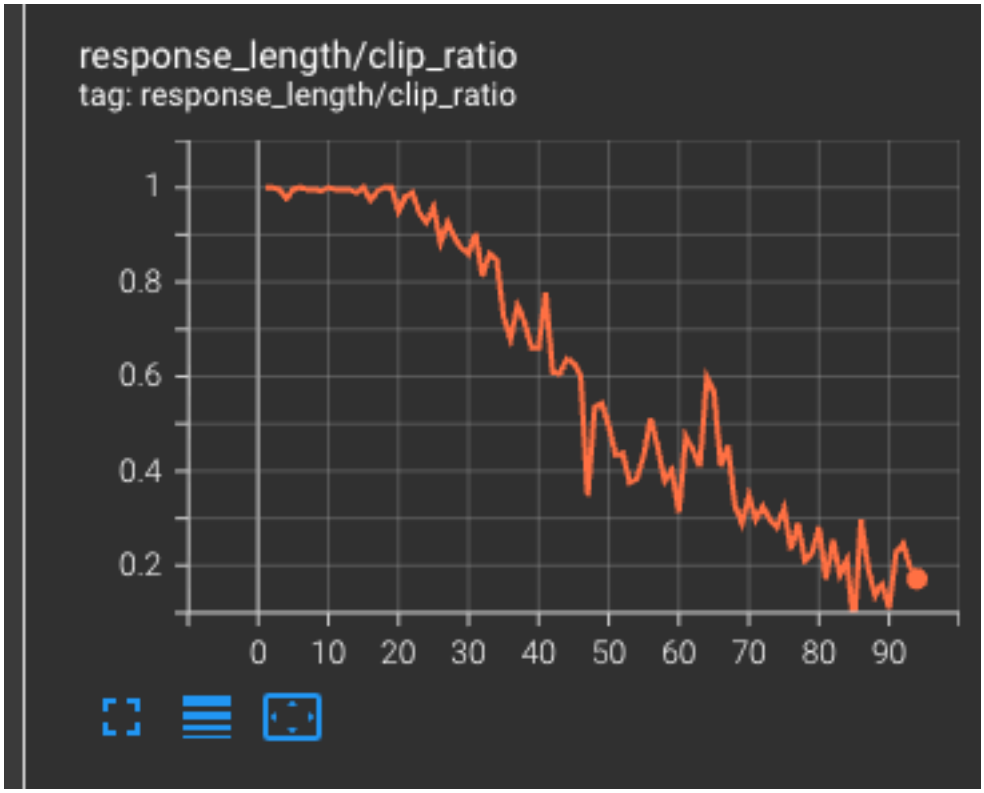
Но дальше по обучению ситуация улучшается: модель постепенно приходит к более единому и короткому формату ответа, который требует заметно меньше токенов, и за счет этого чаще успевает выдать финальный корректный вывод.

Но модель всегда расписывает все шаги последовательно. Интересно, изменится ли это при обучение с увеличенной наградой за короткие ответы. Начнет ли модель сжимать несколько шагов в один? Я сделал LENGTH_ALPHA=0.03, это может показаться слишком маленьким значением, но нормализация в формуле GRPO сделает своё дело, особенно если все роллоуты правильные.

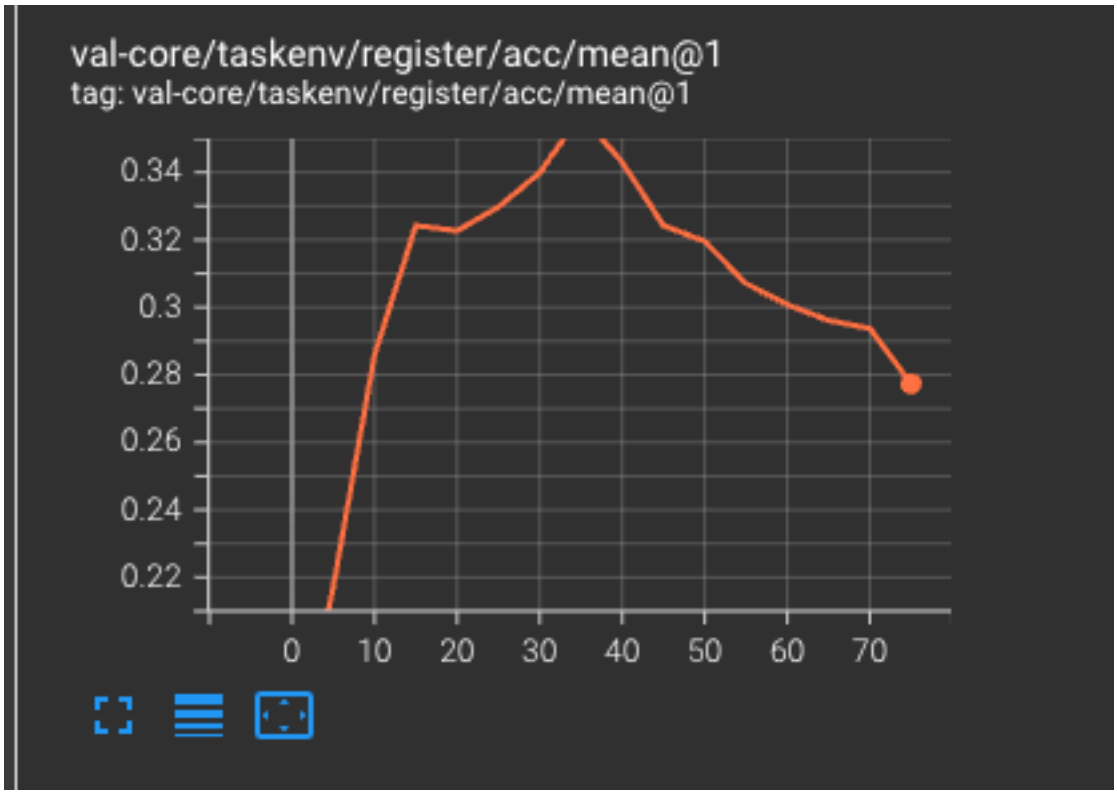
Результаты

- Обучение без награды за длины и небольшим max_response_len в итоге не стало сжимать несколько шагов в один и вышло почти на плато по validation acc. При этом clip_ratio не стало 0, так что возможно есть смысл его ещё подержать





- Обучение с наградой за короткие ответы



Обучение идёт, но ответы на валидации видно не меняются, шаги не сжимаются. При этом acc стала падать. Можно сделать такой вывод: научить модельку считать несколько шагов за раз внутри своих слоёв - сложная задача, особенно если также модель не должна просесть на валидации. Введение небольшой награды за длину может приносить шум в обучение и крайне негативно на него влиять.