Разработка и применение визуально-языковым моделей для робототехники

Казачков Даниил
Email: kazachkovdaniil.rb@gmail.com
Telegram: @Dan_i_il
GitHub: RoboticVLA
W&B logs: classification, regression

Аннотация

Данная работа посвящена имплементации модели на основе трансформерной архитектуры для предсказания действий робота. Для сбора датасета был выбран чекпоинт `RT-1-X`, логгирование велось в `wandb`, в качестве трансформерной модели использована `nano-GPT`.

Введение

Первые попытки обучить робота заключались в сборе датасета и написаннии инструкций [1], [2], узко специализированных на конкретную задачу. Такой подход не позволял обобщать решение на другие области. Вместе с тем LLM показывали впечатляющие возможности в решении различных задач [3], [4], [5] что привело к идее использовать их знания о мире для решения более сложных задач. Одной из прорывных работ, использующих трансформерную архитектуру стала `RT-1` [6], натренированная на реальных траекториях и способная обощать на команды, которые она не видела. Ее ограничения:

- имитационный метод обучения
- обобщение лишь на те инструкции, которые являются комбинацией увиденных
- относительно примитивный набор задач

В следующем поколении - `RT-2` [7] - использовали VLM и претренировку на вебданных, что увеличило обощающую способность, а наличие рассуждений позволило решать более комплексные задачи. Однако ограничения всё же остались:

- сложность в приобретении новых навыков
- большая модель приводит к большей задержке

Неспособность масштабировать модель на совершенно новые задачи попробовали устранить авторы модели `Octo` [8]. Они обучили модель на еще большем числе траекторий, улучшили планирование действий, адаптировали решение под разные среды, ввели возможность выполнять невстречающиеся ранее задачи.

Решение

Чтобы поближе познакомиться с существующими методами, я попробовал адаптировать трансформерную модель nanoGPT для генерации действий робота.

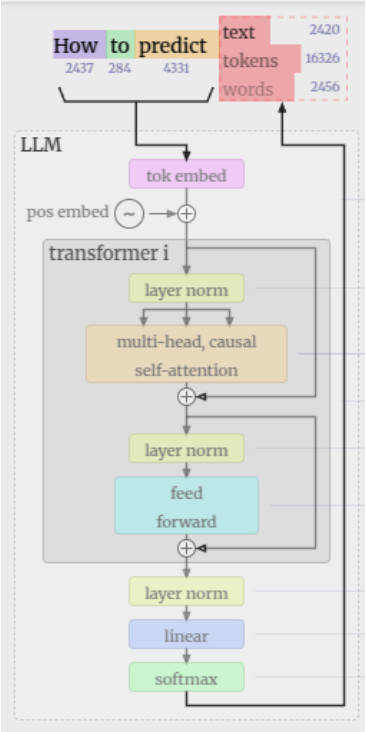

Рис. 1: Архитектура nanoGPT.

В статье $\texttt{RT}-1$ [6] авторы дискретизировали действия в 256 бинов и трансформерная модель решала задачу классификации: предсказывала индекс бина (от 0 до 255) для каждой компоненты действия. В качестве функции потерь использовали логистическую регрессию. Но платой за это является неустранимая ошибка, равная $\frac{1}{256} = 0.0039$.

Так как я тренирую модель `nanoGPT` с 0, то я решил проверить качество и на задаче классификации, и на задаче регрессии. Для последнего я внес несколько существенных изменений:

- изменил входной слой, чтобы он принимал 8-мерные векторы вместо токенов.
- изменил выходной слой, чтобы он выдавал 8-мерные векторы вместо распределения вероятностей по токенам.
- вместо кросс-энтропии использовал MSE для регрессии.

Сбор данных

Ноутбук с кодом можно найти по ссылке. Использовалось окружение Simpler, претренированная модель `RT-1-X` [9]. Собрано 134 траектории, имеющих структуру 1.

Листинг 1: Структура шага в траектории

```
{
  'timestep': 0,
  'action': {
    'world_vector': array([ 0.1291585 ,  0.10567522, -0.02739727]),
    'rot_axangle': array([ 0.29202676, -0.00307393,  0.18136406]),
    'gripper': array([0.]),
    'terminate_episode': array([0, 1, 0], dtype=int32)
  }
}
```

Далее вектор `terminate_episode` заменяется на единичный элемент со значением продолжения или остановки траектории.

Тренировка

Тренировка проводилась на видеокарте NVIDIA GeForce RTX 2080 Ti. В репозитории проекта для каждой модели написан соответствующий `.py` файл с определением и скриптом тренировки.

`nanoGPT` - модель легкая, поэтому я использовал 1024 бина для дискретизации.

Результаты

Ошибку на модели регрессии и классификации обозначим соответственно $L_{\text{регр}} = 0.259$ и $L_{\text{клас}} = 0.267$. Их значение также можно найти в colab ноутбуке.

Заключение

В задаче регрессии можно использовать специализированные трансформерные архитектуры, такие как `TrajectoryTransformer` или `DecisionTransformer`, которые обладают своими преимуществами и недостатками. Однако для первичного знакомства простая модель подходит лучше всего. Не слишком хорошие показатели качества можно объяснить малостью датасета, из-за которого в том числе модель классификации быстро переучивалась.

Список литературы

[1] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Conference on robot learning, pages 651–673. PMLR, 2018.

[2] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. arXiv preprint arXiv:2104.08212, 2021.

[3] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[4] Hugo Touvron et al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[6] Anthony Brohan et al. Rt-1: Robotics transformer for real-world control at scale, 2023.

[7] Anthony Brohan et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023.

[8] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In Proceedings of Robotics: Science and Systems, Delft, Netherlands, 2024.

[9] Embodiment Collaboration et al. Open x-embodiment: Robotic learning datasets and rt-x models, 2024.