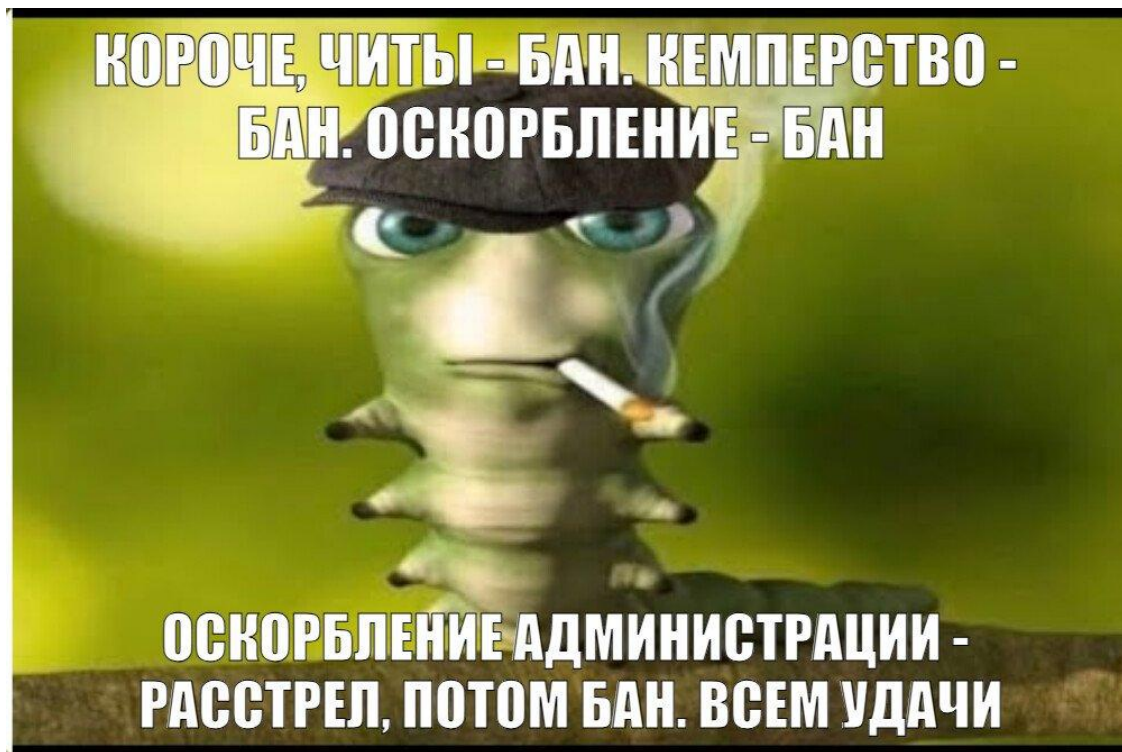


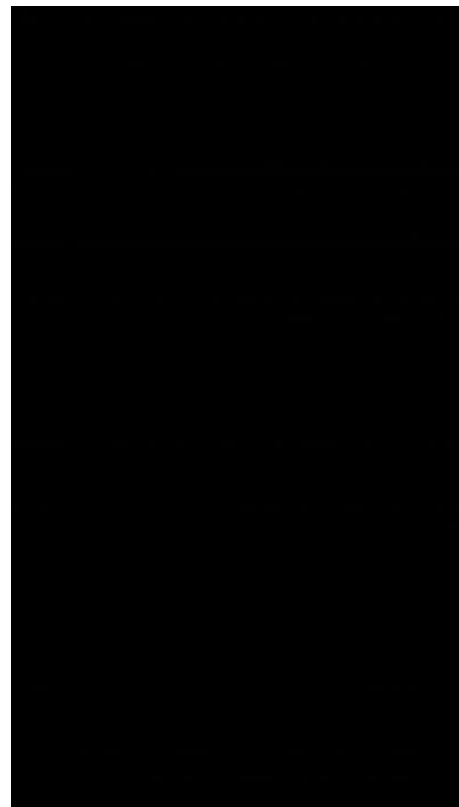
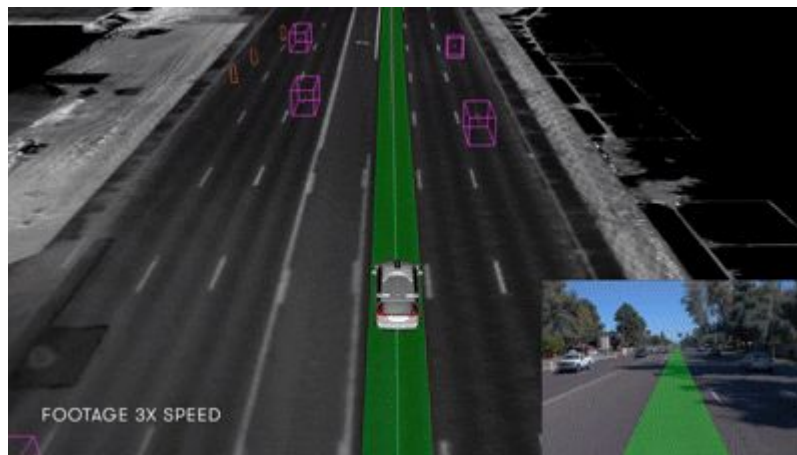
# Applications of RL for self-driving

by. Деревягин Александр

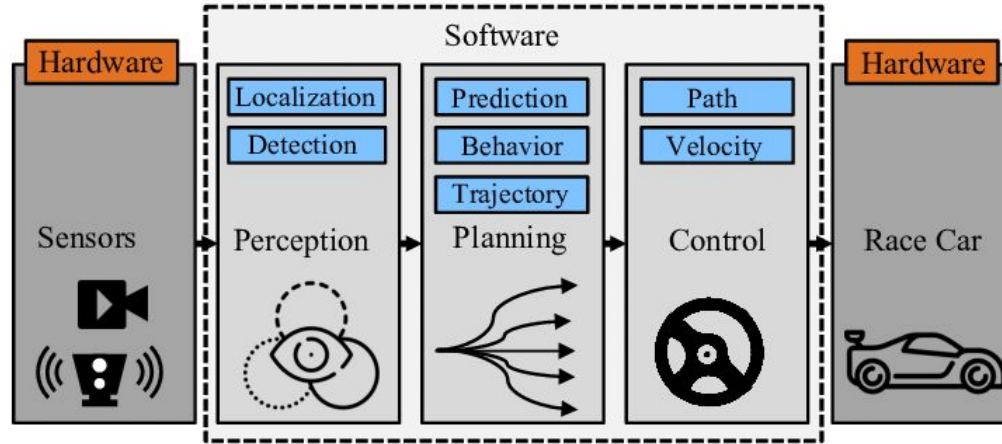
# Правила на вебинар



# Удивительный мир autonomous driving



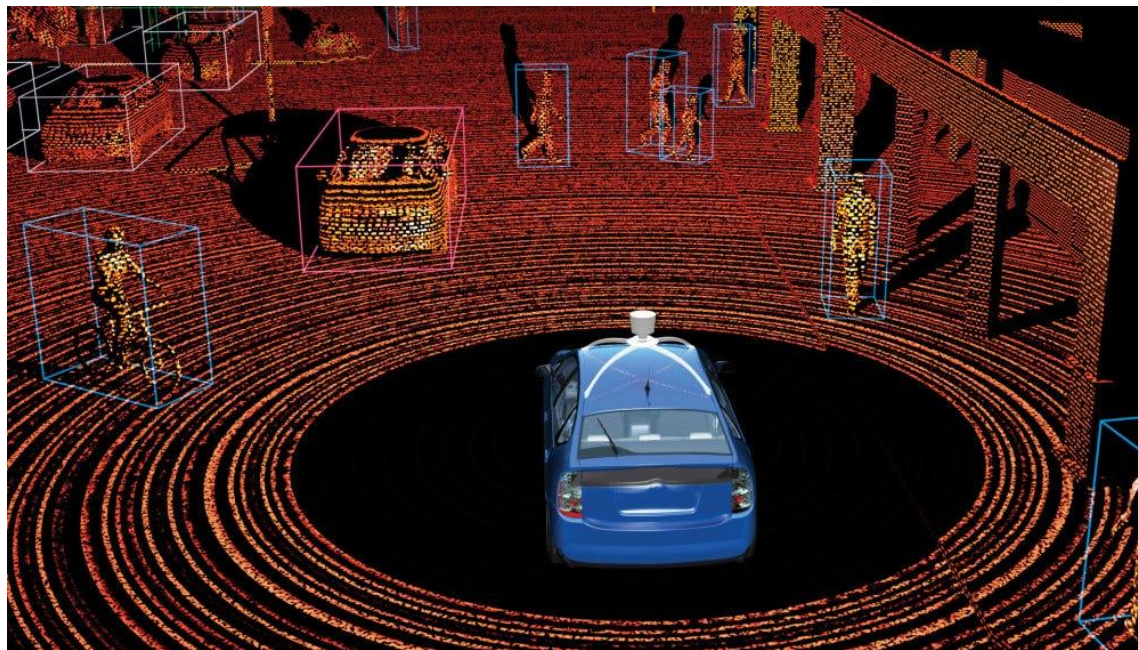
# Постановка задачи (Модульный подход)



Три модуля – perception, planning и control. Perception отвечает за формирование общего состояния автомобиля, детекцию объектов (например, динамических объектов). Control переводит предсказанные траектории в действия руля.

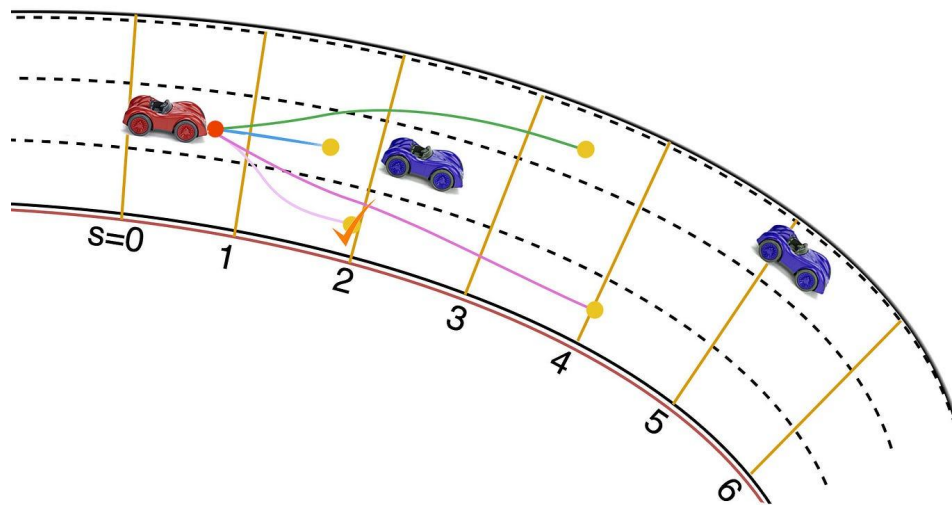
# Постановка задачи (модуль planning)

- На вход – энкодинг сцены, матрица  $R^{N \times D}$
- В каком виде мы бы хотели предсказывать траекторию?



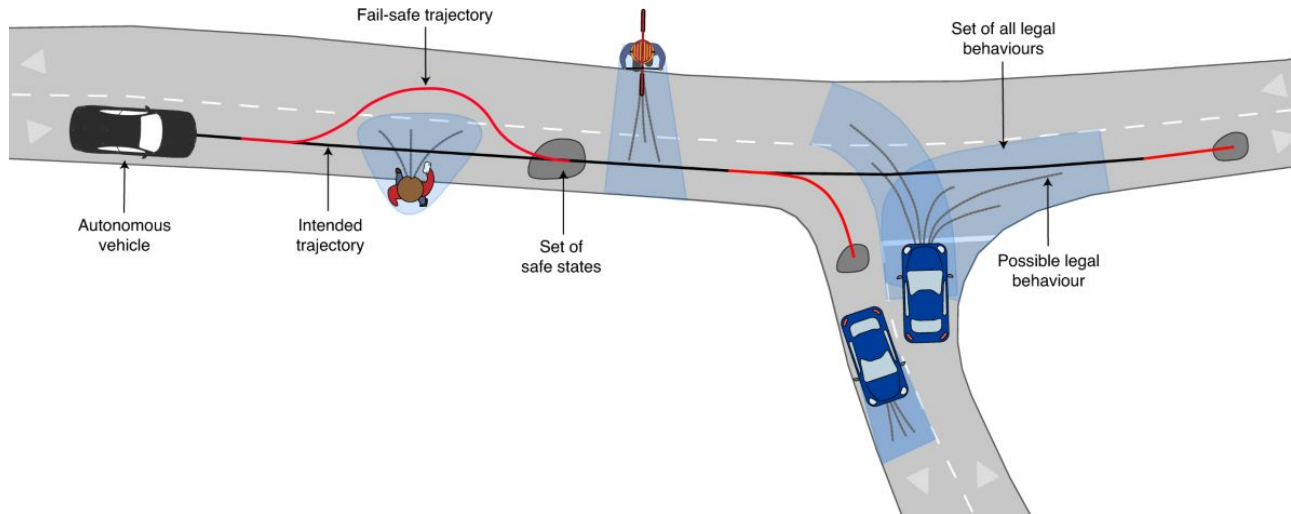
# Постановка задачи (модуль planning)

- На вход – энкодинг сцены, матрица  $R^{N \times D}$
- В каком виде мы бы хотели предсказывать траекторию?
- На выходе –  $k$  траекторий  $R^{K \times T}$  – каждую предсказываем на  $T$  шагов вперед
- Как выбирать подходящую?



# Постановка задачи (модуль planning)

- На вход – энкодинг сцены, матрица  $R^{N \times D}$
- В каком виде мы бы хотели предсказывать траекторию?
- На выходе –  $k$  траекторий  $R^{K \times T}$  – каждую предсказываем на  $T$  шагов вперед
- Как выбирать подходящую?
- Исходя из значений метрик – мы можем посчитать комфорт, безопасность (об этом далее)



# Как оцениваем?

В self-driving рассчитывается целый ряд метрик

- Безопасность (Time-To-Collision)
- Комфорт
- Соблюдение скоростного режима
- Консистентность
- Метрики Imitation Learning (FDE и ADE)

И многое другое. Мы рассмотрим только часть из них



# Как оцениваем?

Безопасность: Time-To-Collision – оцениваем минимальное столкновение авто с другими динамическими агентами

$$TTC = \min_{i=0}^n \text{time.to.collision}(\text{pos}_i, v_i, \text{posego}, v_{ego})$$

Как вариант мы можем считать бинарно будет ли столкновение через threshold секунд

$$TTCbin = I\left\{\min_{i=0}^n \text{time.to.collision}(\text{pos}_i, v_i, \text{posego}, v_{ego}) \leq \text{threshold}\right\}$$

# Как оцениваем?

Комфортность:

Считаем Jerk и YawRate

$$Jerk = \frac{1}{n} \sum_{i=0}^n \frac{\Delta a_i}{\Delta t_i}$$

$$YawRate = \frac{1}{n} \sum_{i=0}^n \frac{\Delta \phi_i}{\Delta t_i}$$

Комфорт считаем также бинарно как нахождение jerk и yawrate в определенных интервалах

$$Comfort = \frac{1}{n} \sum_{i=0}^n I \left\{ (l_{jerk} \leq \left| \frac{\Delta a_i}{\Delta t_i} \right| \leq r_{jerk}) \wedge (l_{yaw} \leq \left| \frac{\Delta \phi_i}{\Delta t_i} \right| \leq r_{yaw}) \right\}$$

# Как оцениваем?

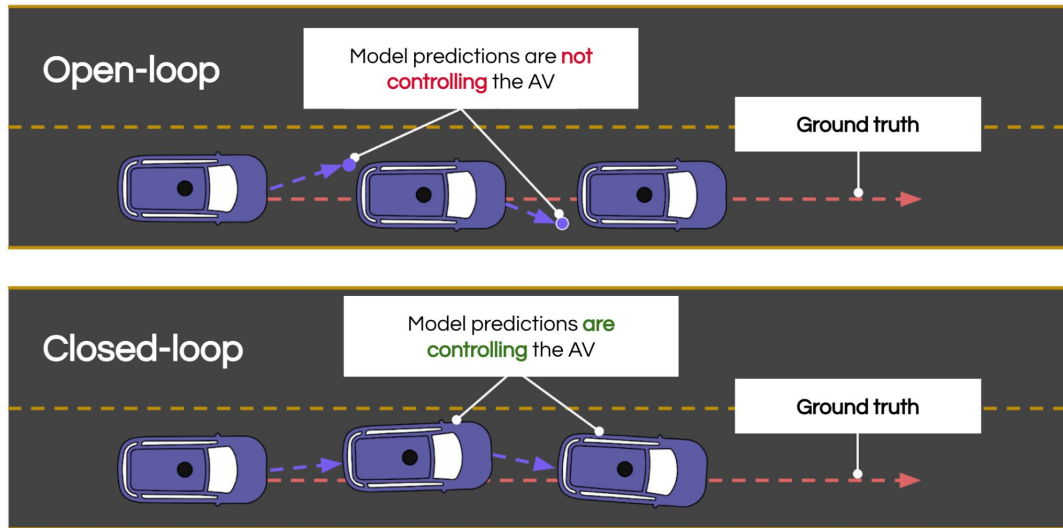
Что значит остальное (словами)

- FDE и ADE – (Final Displacement Error и Absolute Displacement Error) – оценка того, насколько финальная точка в траектории эксперта отличается от основной, и насколько в среднем отличаются в среднем (L2) траектория эксперта и предсказанная траектория
- Консистентность – оценка насколько соотносятся между собой скорости, повороты в траектории.
- Соблюдение скоростного режима – бинарная метрика насколько мы соблюдаем скоростной режим

# Как оцениваем?

Два этапа оценивания

- open-loop – мы оцениваем траекторию **исключительно** из траекторий экспертов в датасете: даем сцену в модель + экспертную траекторию, оцениваем насколько мы хорошо имитируем траекторию
- close-loop – мы симулируем траекторию с нуля в симуляторе, в такой постановке модель сама по себе используется для управления автомобилем.



# Open-loop vs Close-loop

## Плюсы open-loop

- быстрая оценка, позволяет отметить сразу плохие решения, понять что плохо работает (проверить что имитация работает плохо)
- показывает насколько хорошо мы справляемся с имитацией траекторий водителей, что также важно

## Плюсы close-loop

- Позволяет оценить модель “с нуля”, на различных ситуациях, без “подсказок” от экспертной траектории водителя
- Лучше охватывает экстренные ситуации (аварии, нарушения пдд)

# Open-loop vs Close-loop

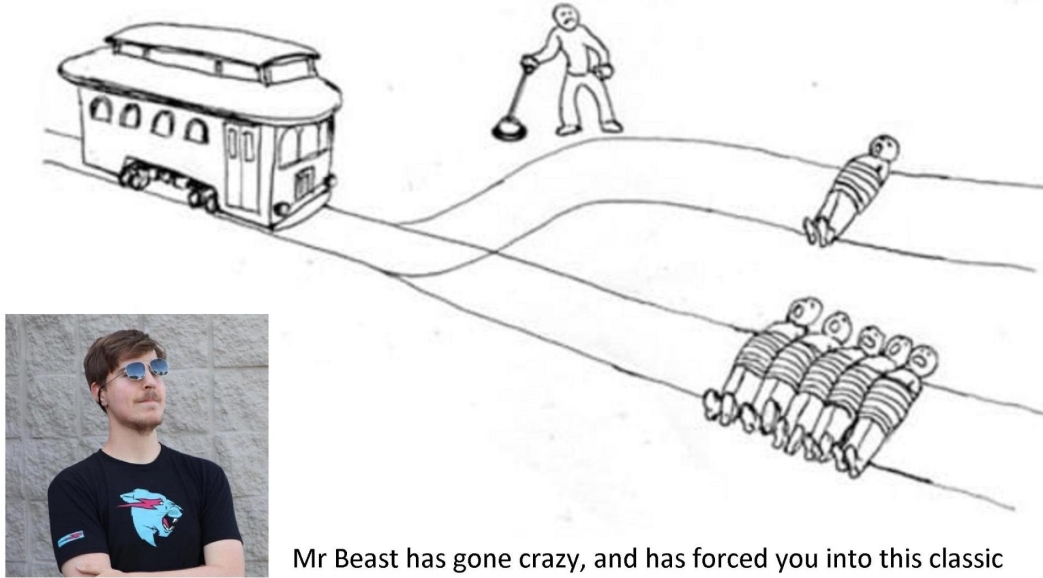
## Минусы open-loop

- Не охватывает ситуации аварий, нарушений пдд
- Получаем смещенную оценку для ряда метрик (комфорт, безопасность)

## Минусы close-loop

- Долгая симуляция, особенно если рассчитываем большой набор сценариев
- Совокупность метрик ничего не говорит, нужно оценивать по отдельности

# Проблема вагонетки



Mr Beast has gone crazy, and has forced you into this classic trolley problem. He will give you \$200,000 for every person who dies in this trolley problem. Do you pull the lever?

## В реальных сценариях

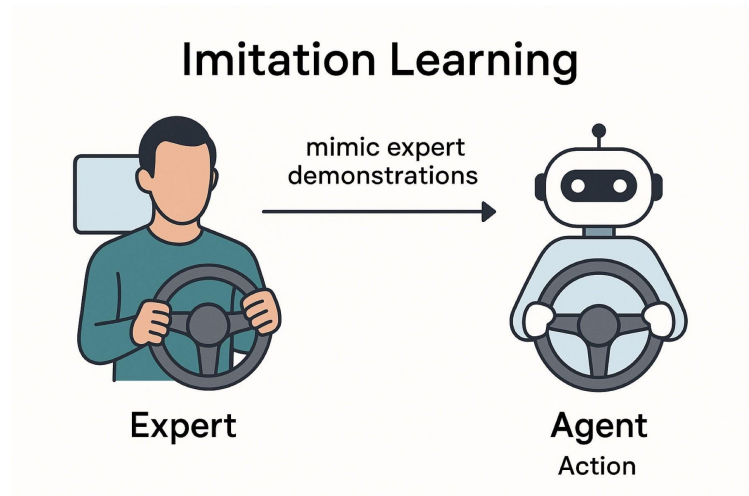
- Есть сценарии, где нужно пожертвовать комфортом ради безопасности
- Есть сценарии, где нужно пожертвовать консистентностью траектории ради безопасности и комфорта
- Соответственно, каждый сценарий в close-loop индивидуален и нужно смотреть их по отдельности

Если одна метрика проседает, это не значит что планирование идет плохо.



# Подход 1 (Imitation Learning)

- Трансферим из робототехники, собираем данные с набором водителей на разных сценариях, учимся копировать следующее действие
- Легко “засетапить”: экспертных водителей много, мест где можно водить машину много, большую и богатую выборку можно собрать эффективно
- Простой способ на который можно опираться для обучения своего беспилотного автомобиля



# Подход 1 (Imitation Learning)

Для политики  $\pi$ :

$$\pi_{\theta}(s_t) \sim a_t$$

Учим модель на L2-лосс копировать экспертное действие:

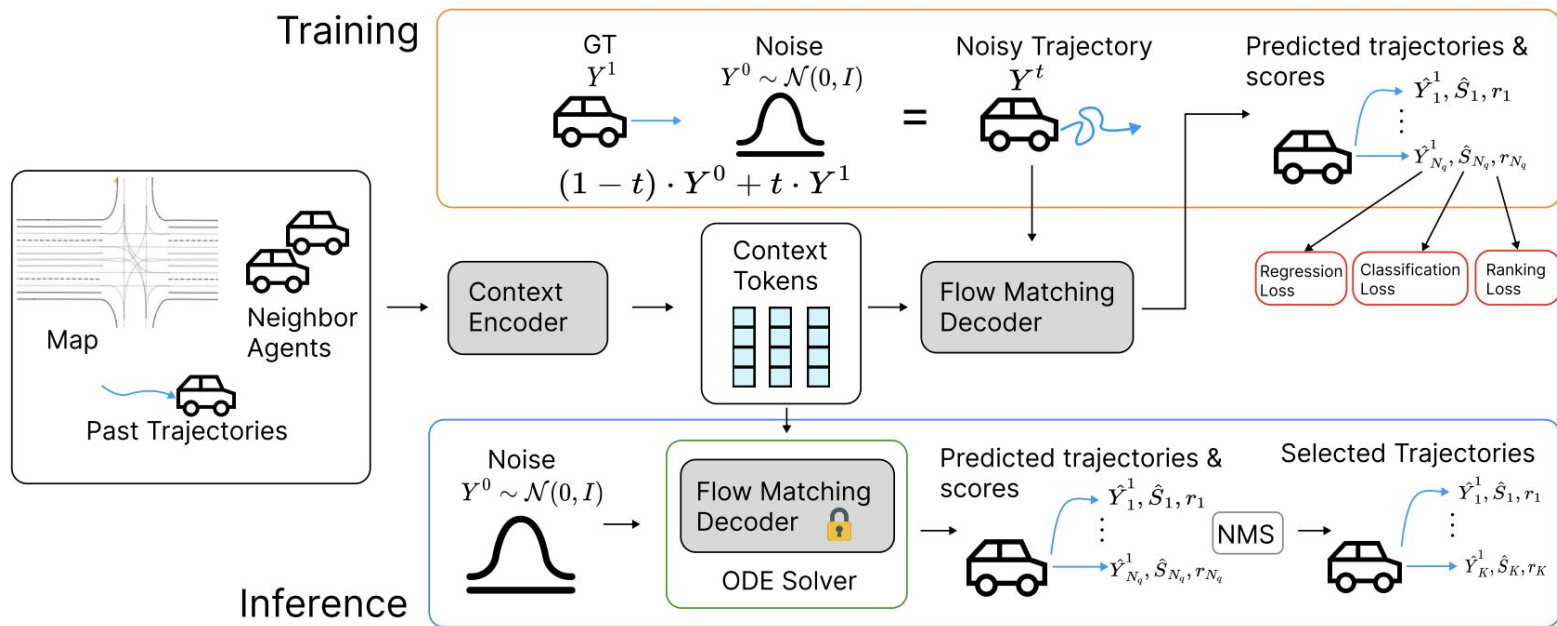
$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \|a_i - \pi_{\theta}(s_i)\|_2^2$$

# Подождите, но... траекторий же несколько?

Да, и это можно делать несколькими способами

- Простой – сделать  $K$  обучаемых запросов (последовательностей) и делать imitation learning как seq-to-seq задачу – так например сделано в PLUTO, на который мы сегодня будем ссылаться
- Сложный, но при этом охватывающий большее число траекторий – выучивать распределение траекторий с использованием Flow Matching или диффузионных моделей.

# Генерация траектории с использованием flow matching



# Проблемы Imitation Learning

- Водители водят идеально – проблемных сценариев в выборке (аварии, нарушение ПДД) минимальное количество
- Планировщик может скатиться в “прямое копирование” траектории эксперта – будет плохо себя вести если немного от нее отклонится

# Contrastive Imitation Learning (PLUTO)

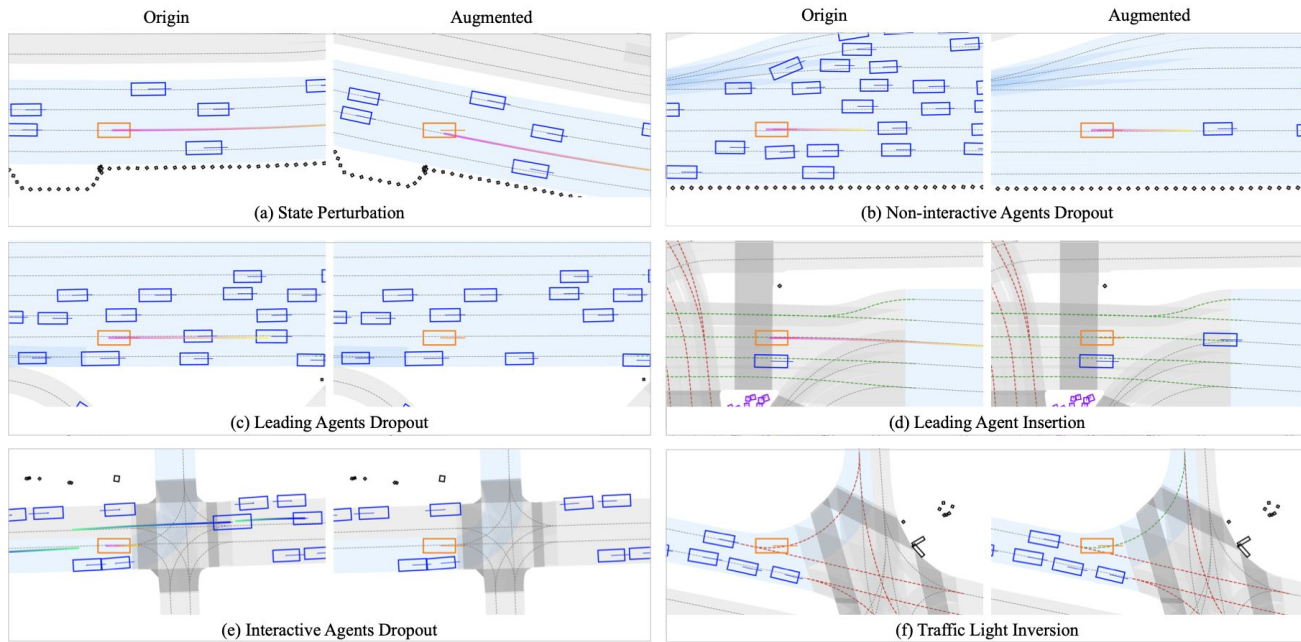


Fig. 4. Exemplary scenarios of the proposed data augmentations. In each group, the figure on the left denotes the origin scenario and the right one shows the augmented scene. AV is marked in the orange, other vehicle agents are in blue, and dash-colored lines on the lanes denote the traffic light status. (a)-(b) belongs to the positive augmentations  $\mathcal{T}^+$  and (c)-(f) are negative augmentations  $\mathcal{T}^-$ .

# Contrastive Imitation Learning (Contrastive loss)

In practice, we randomly sample a minibatch of  $N_{bs}$  samples. Each sample undergoes positive and negative augmentation, executed by augmentors randomly chosen from the sets  $\mathcal{T}^+$  and  $\mathcal{T}^-$ , respectively. This augmentation triples the total number of samples to  $3N_{bs}$ . All samples are processed by the same encoder and projection head. Let  $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{v} / \|\mathbf{u}\| \|\mathbf{v}\|$  denote the dot product between the  $l_2$  normalized  $\mathbf{u}$  and  $\mathbf{v}$ , the softmax-based triple contrastive loss [50] is defined as:

$$\mathcal{L}_c = -\log \frac{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\sigma)}{\exp(\text{sim}(\mathbf{z}, \mathbf{z}^+)/\sigma) + \exp(\text{sim}(\mathbf{z}, \mathbf{z}^-)/\sigma)}, \quad (13)$$

where  $\sigma$  denotes the temperature parameter. The contrastive loss is computed across all triplets in the mini-batch. Besides this, we provide supervision to both the original and positively augmented samples using the unmodified ground truth trajectory. Note that negatively augmented samples are only used to calculate the contrastive loss as their origin ground truth may be invalid after augmentation. The overall training loss comprises four components: imitation loss, prediction loss, auxiliary loss, and contrastive loss, represented as:

$$\mathcal{L} = w_1 \mathcal{L}_i + w_2 \mathcal{L}_p + w_3 \mathcal{L}_{aux} + w_4 \mathcal{L}_c. \quad (14)$$

## Подход 2 (Offline RL)

- Ревард – комбинация метрик описанных выше

### 3.5 Reward Function Engineering

To enable reinforcement learning, we designed a dense, multi-objective reward function  $R(s_t)$  to provide a learning signal at every timestep. The function is a weighted combination of components that define our desired driving behavior. The final score is normalized to a consistent range using a hyperbolic tangent function to ensure stable training. The total reward is calculated as:

$$R(s_t) = \tanh \left( \frac{1}{C} (w_r r_{\text{route}} + w_s p_{\text{safety}} + w_c p_{\text{comfort}}) \right) \quad (1)$$

where  $C$  is a scaling factor and  $w$  are the component weights. The individual reward and penalty terms are summarized in Table 3. This squashing normalization is crucial for bounding the final reward signal to the range  $[-1, 1]$ , preventing un-normalized penalty values from creating extreme gradients and destabilizing the critic's learning process.



## Подход 2 (Offline RL)

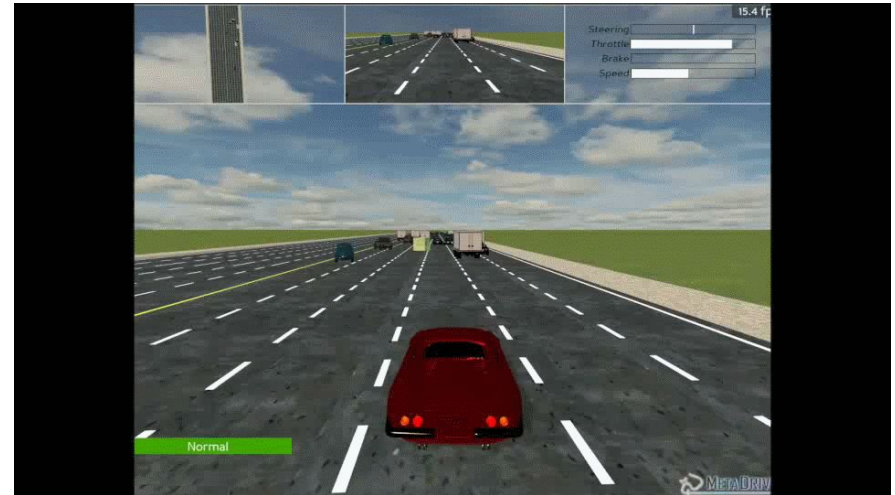
- Ревард – комбинация метрик описанных выше
- Алгоритм – обычно берут алгоритмы, которые могут обобщаться на действия из OOD, например Batch-Constrained Q-learning и Conservative Q-learning.

# Итоги

- RL поможет дифференцировать сценарии, улучшить ситуацию на проблемных сценариях
- Не решает проблему полностью, тк плохих сценариев катастрофически мало, может только помочь поалайнить политику на определенные метрики

# Подход 3 (RL в симуляторе)

Симуляторы (CARLA, MetaDrive)



# Проблемы обучения RL в симуляторе

- Медленно
- Есть проблема переноса sim2real

# Решение проблем

- Для sim2real – решается фотореалистичной симуляцией, созданием достаточно точных симуляторов, в которых можно учить модели
- Скорость – пока самая большая проблема подхода

# Реальный пример обучения в симуляторе (Practical ML Conf)

13689.96

567.90

8715.32

1715.32

567.90

## RL Open Loop (LLM like)

- One scene encoding
- Multiple rollouts
- Реворд по исходным метрикам

Agents

Ego

Map

Model

Multiple Ego Rollouts

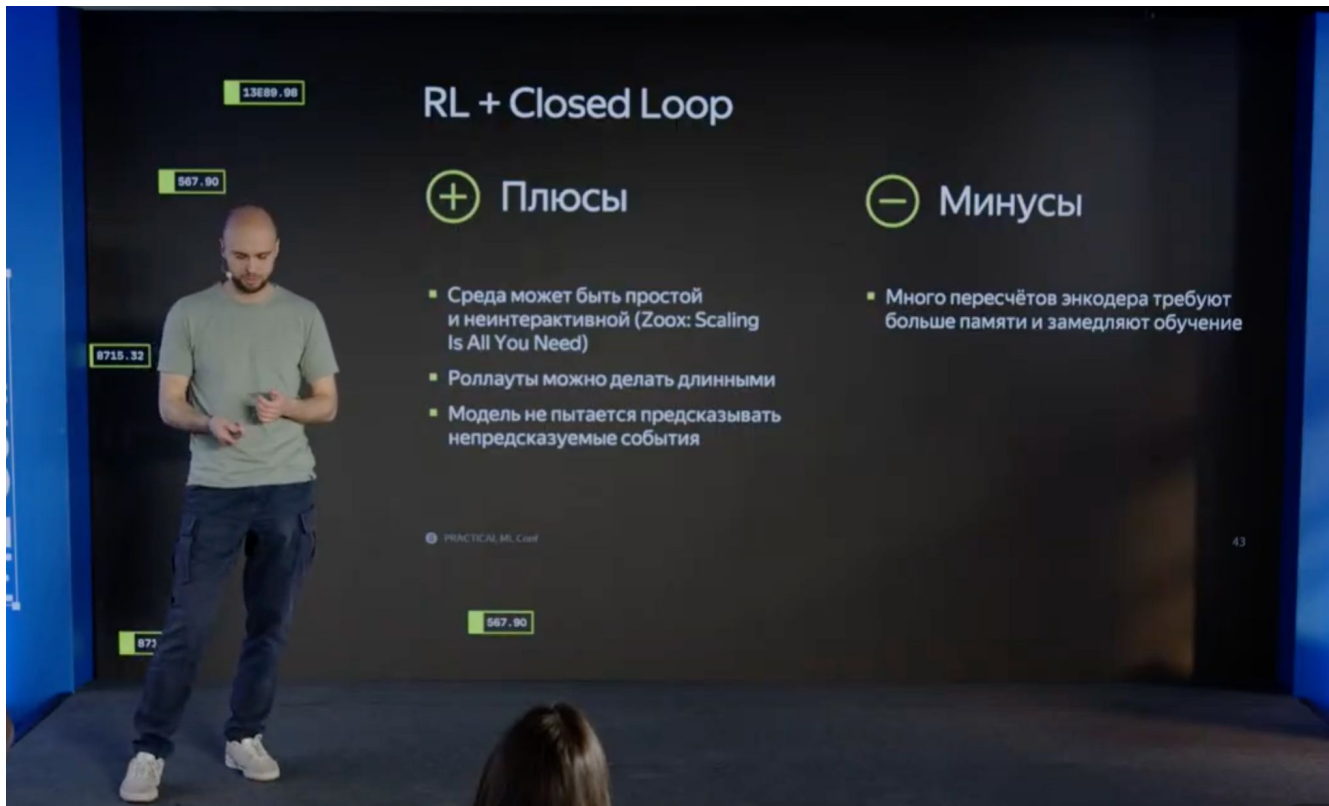
Rewards

RL

33

```
graph TD; Agents[Agents] --> Model[Model]; Ego[Ego] --> Model; Map[Map] --> Model; Model --> Rollouts[Multiple Ego Rollouts]; Rollouts --> Rewards[Rewards]; Rewards --> RL[RL];
```

# Реальный пример обучения в симуляторе (Practical ML Conf)



13689.98

567.90

8715.32

871

## RL + Closed Loop

### + Плюсы

- Среда может быть простой и неинтерактивной (Zoox: Scaling Is All You Need)
- Роллауты можно делать длинными
- Модель не пытается предсказывать непредсказуемые события

### - Минусы

- Много пересчётов энкодера требуют больше памяти и замедляют обучение

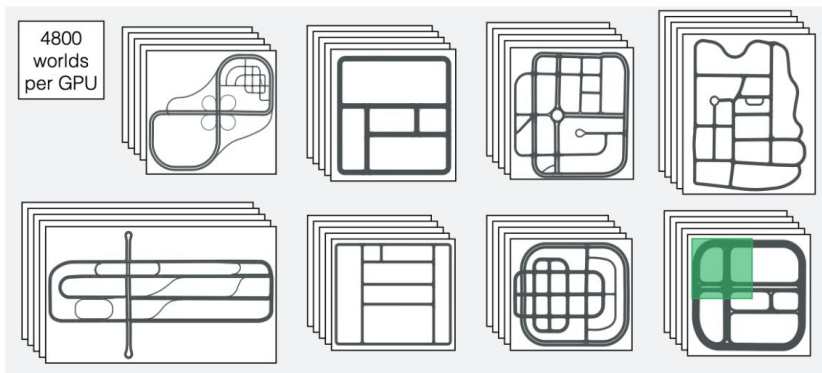
567.90

Practical ML Conf

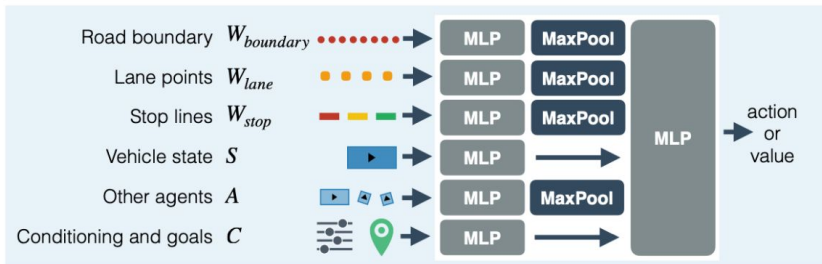
43

# Advanced подход (Self-Play)

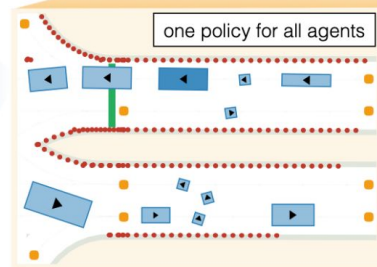
**a** Batched Gigaflow simulation



**b** Gigaflow world



**d** Gigaflow policy



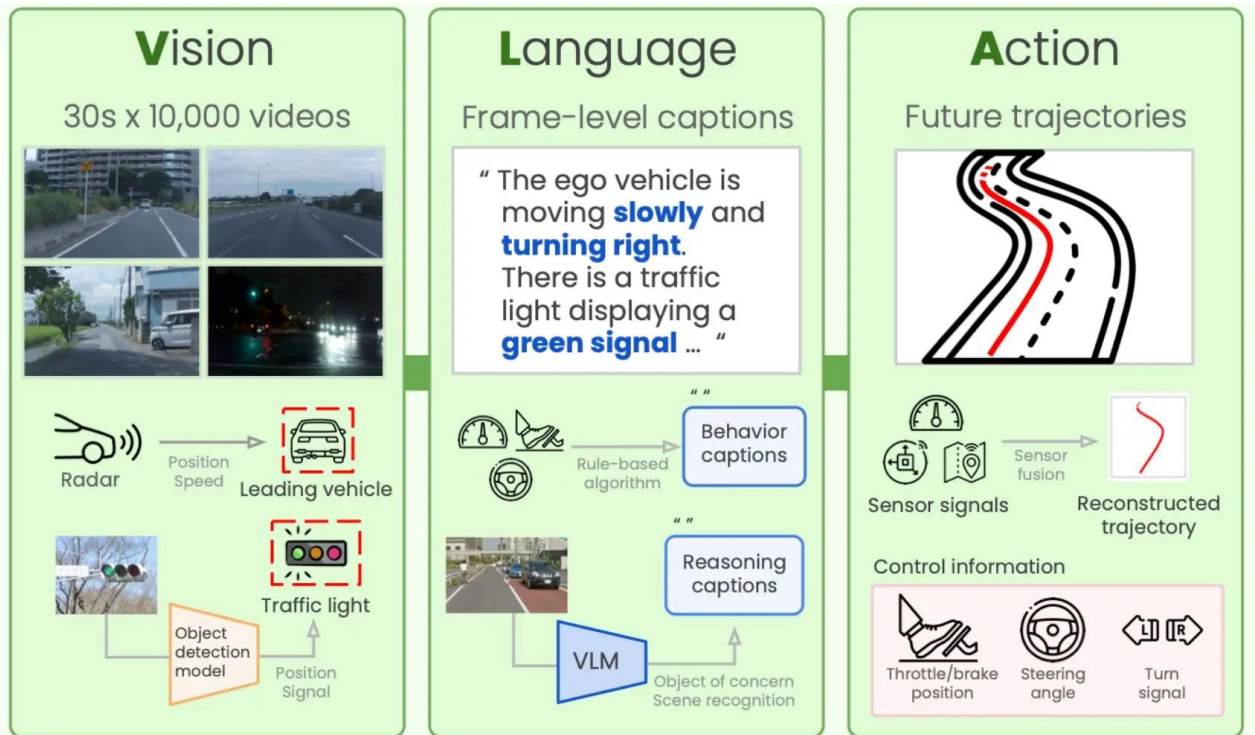
**c** Gigaflow agent



# Advanced подход (Self-Play)

- Одна политика на всех агентов
- Модель сама себе генерирует роллауты
- Обучаем генералист-политику для всех агентов с использованием PPO

# End-to-End модель (VLA)



# Зачем End-to-End?

- Текущие алгоритмы планирования явно зависят от perception модулей и обработки входных данных для модели
- End-to-end подход монолитно соединяет perception и planning, делая модель независимой от изначальных представлений (нужны только входные данные).
- На задачах робототехники показана генерализуемость VLA на задачи робототехники, которую можно было бы перенести на наши сценарии

Вопросы?