# Разработка метода управления роботизированными системами на основе генеративного подхода обучения с подкреплением

Огородников Даниил Николаевич

**Аннотация**: Разработка систем управления роботами традиционно является сложным и ресурсоёмким процессом. Предлагаемый генеративный подход на основе обучения с подкреплением позволяет существенно упростить и ускорить эту задачу.

Разработанная модель самостоятельно формирует сжатое латентное представление окружающей среды (пространственное и временное) без учителя. На основе полученных признаков обучается крайне компактная политика управления способная решать целевую задачу. Ключевое преимущество - возможность полного обучения агента внутри собственной сгенерированной моделью мира с последующим переносом (zero-shot sim-to-real) полученной политики в реальную среду.

**Проблема, рассматриваемая в работе:**

Разработка систем управления для роботов – это сложный и многогранный процесс, требующий значительных затрат времени и усилий. Для достижения успеха в данной области необходимо учитывать множество факторов и принимать во внимание различные аспекты.

Во-первых, разработка системы управления для роботов требует глубокого понимания конкретной задачи, которую робот должен решать. Это включает в себя анализ требований, исследование уже существующих решений и определение оптимальной стратегии действий.

Во-вторых, подготовка и реализация системы управления включает в себя программирование, создание алгоритмов и моделей, а также интеграцию всех компонентов робота. Это требует хорошего знания языков программирования, методов оптимизации и математического моделирования.

## Введение

Человеческий мозг формирует внутреннюю модель мира, сжимая пространственную и временную информацию в абстрактные представления и используя предсказания будущего для управления восприятием и действиями. Предлагаемая архитектура воспроизводит этот принцип: вариационный автоэнкодер (VAE) создаёт сжатое латентное представление текущего наблюдения, рекуррентная сеть MDN-RNN прогнозирует будущее в латентном пространстве, а простой линейный контроллер принимает решения только на основе этих представлений.

## 1.Теоретическая часть

### 1.1 Основные ограничения классического обучения с подкреплением

В большинстве методов RL без модели используются небольшие нейронные сети из-за проблемы credit assignment при редком вознаграждении в конце эпизода. Это ограничивает выразительность агента. Методы с моделью мира обычно всё равно требуют обучения в реальной среде.

### 1.2 Предлагаемая архитектура агента

Агент состоит из трёх независимо обучаемых компонентов:

• VAE — сжатие каждого кадра в низкоразмерный вектор $z_t$.

• MDN-RNN — предсказание распределения $p(z_{t+1} \mid z_t, h_t, a_t)$ в виде смеси гауссиан (модель мира).

• Контроллер — однослойная линейная сеть, отображающая $[z_t, h_t] \rightarrow a_t$.

VAE и MDN-RNN обучаются без учителя и без доступа к сигналу вознаграждения. Только контроллер оптимизируется с подкреплением (CMA-ES) и имеет доступ к reward.

### 1.3 Обучение полностью в сгенерированной среде

Контроллер обучается исключительно внутри rollout-ов, сгенерированных собственной моделью мира (MDN-RNN + декодер VAE). Для предотвращения эксплуатации артефактов модели используется повышенная температура сэмплирования, создающая более шумные и неопределённые траектории.

## 2 Практическая часть

### 2.1 Подготовка моделей

Для обучения модели VAE использовался набор данных ego4d состоящий из 3870 часов видеоматериала, где люди выполняют ручную работу, и был преобразован в $10^7$ изображений размером 512x540 пикселей, и датасет DROID с роботами выполняющие различные действия. Суммарный объём данных составил 1.5 терабайта, но после предобработки и сжатия изображений до 128x128 пикселей объём составил 96 гигабайт.

Полученный набор данных используется для обучения VAE, чтобы узнать скрытое пространство каждого наблюдаемого кадра. Как уже ранее упоминалось, задача VAE кодировать каждый кадр в низкоразмерный скрытый вектор $z$ минимизируя разницу между заданным кадром и реконструированной версией кадра, полученной декодером из $z$. Следующая демонстрация показывает результаты VAE после обучения:

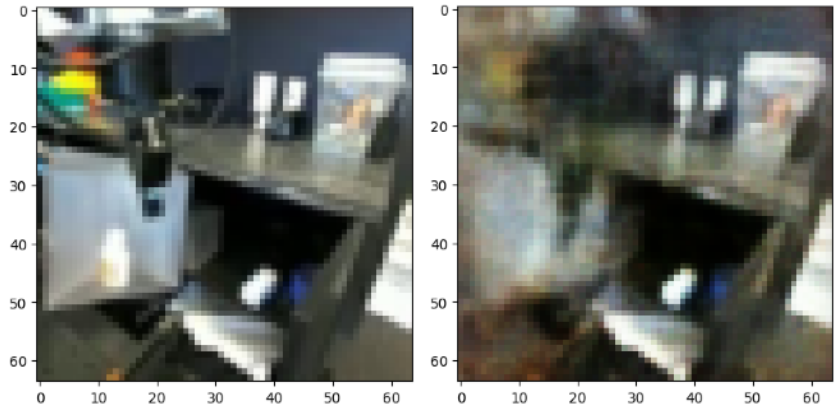

Рисунок 3. Обработка изображение нейросетью и последующие восстановление

Теперь имеет смысл использовать обученную модель VAE для предварительной обработки каждого кадра в определенный момент времени.$t$ в $z_t$ для обучения нашей модели. Используя эти предварительно обработанные данные, а также записанные случайные действия $a_t$, модель MDN-RNN теперь может быть обучена моделировать значения вероятностей (формула 3) как смесь гауссианов[1]

В этой работе модели VAE и MDN-RNN не имеют никаких знаний о фактических сигналах вознаграждения из окружающей среды. Их задача — просто сжать и предсказать последовательность наблюдаемых кадров изображения. Только модель контроллера имеет доступ к информации о вознаграждении из окружающей среды. Поскольку внутри линейной модели контроллера всего 867 параметров, эволюционные алгоритмы, такие как CMA-ES, хорошо подходят для этой задачи оптимизации.

Рисунок ниже сравнивает фактическое наблюдение, данное агенту, и наблюдение, зафиксированное моделью мира. Также можно рассматривать VAE для реконструкции каждого кадра с помощью$m$на каждом временном шаге для визуализации качества информации, которую агент фактически видит во время развёртывания:

[1] Хотя можно обучать VAE и MDN-RNN вместе сквозным образом, было выявлено, что обучение каждого по отдельности более практично, достигает удовлетворительных результатов и не требует исчерпывающей настройки гиперпараметров. Поскольку изображения не требуются для обучения MDN-RNN сами по себе, мы можем обучать даже на больших пакетах длинных последовательностей скрытых векторов, кодирующих все 1000 к. среды, чтобы захватить долгосрочные зависимости, на одном GPU.

## 2.2 Принцип работы

Рассмотрим принцип работы на основе среды, созданной в программе для 3D симуляций - UNIGINE sim, где задачей является “сложить тряпку в контейнер”. Первым этапом модель VAE расставляет ассоциации с объектами, выделенными на этапе обучения в латентном пространстве на основе полученных изображений с камер робота, затем происходит передача состояния латентного пространства $z$ на вход MND-RNN, где происходит расчёт дальнейшего состояния и создание матрицы вероятностей действий. Следующем шагом происходит передача матрицы модели контроллера, где просчитываются действия, и передача действий в среду и дублируется обратно в MDN-RNN для повторных расчётов.

Так же можно представить вариант обучение полностью в замкнутом пространстве, т. е. позволить генерировать среду и отклики от неё, модели обратной модели VAE из состояний заранее подготовленного латентного пространства. Ход данного метода обучения практически ничем не отличается от обучения совместно со средой но есть и ограничения например невозможность контролировать точные значения.

### 2.4.1 Использование результатов.

Отличительной чертой моей модели заключается в отсутствии необходимости наличия тренировочной среды на большей части этапа обучения как это требуют все отличные алгоритмы например: DQN (Deep Q Network), DDPG (Deep Deterministic Policy Gradients) [8] или метод Reinforce [6].

Ниже в таблице 1 приведены результаты сравнения моей разработки с похожими методами обучения с подкреплением.

Таблица 1. Сравнение прототипа с аналогами

| | Время обучения (шаги) | Требуемые ресурсы | Кол-во параметров | Точность/ потери | Восприимчивость к изменению среды |
|---|---|---|---|---|---|
| DQN | $10^6$-$10^7$ | NxCPU | ~20.000 | 0.45/0.05 | - |
| DDPG | $10^7$-$10^8$ | NxCPU | ~32.000 | 0.51/0.25 | - |
| BCO [10] | $10^5$-$10^6$ | 4xGPU | ~563.000 | 0.32/0.04 | - |
| VIP [11] | $10^8$-$10^9$ | 4xGPU | ~1.230.000 | 0.53/0.06 | + |

| | | | | | |
|---|---|---|---|---|---|
| RLV [9] | $10^5$-$10^6$ | 10xTPU | ~653.000 | 0.56/0.09 | - |
| разработанная ммодель | $10^4$ | 5xGPU | 4.341.790 | 0.66/0.07 | + |

По итогам сравнения можно выявить, что для обучения представленной в работе модели требуется на порядок меньше времени чем у других. Конечно, в данном сравнении не учитывается время на обучения VAE так как эта модель универсальна для всех сред обучения.

Анализируя эффективность работы модели, можно сделать вывод, что она демонстрирует одинаково высокие результаты на различных типах задач без необходимости проведения повторного обучения нейросети. Это означает, что модель обладает универсальностью и способностью применяться для разнообразных задач, не требуя дополнительной настройки или переобучения.

Поскольку модель способна моделировать будущее, для эксперимента её можно заставить самостоятельно придумывать гипотетические сценарии среды и окружения. Можно попросить модель создать распределение вероятностей $\mathbf{z}_{t+1}$ учитывая текущие состояния, и использовать этот образец как реальное наблюдение. Мы можем вернуть модель контроллера в эту среду самогенерации, созданную MDN-RNN.