Санкт-Петербургский государственный университет

**Тыщук Кирилл Ильич**

**Выпускная квалификационная работа**

# Обнаружение скрытых шаблонов в обучении с подкреплением

Уровень образования: бакалавриат
Направление 01.03.01 «Математика»
Основная образовательная программа СВ.5000.2017 «Математика»

Научный руководитель:
Доцент, К.ф-м.н.,
СПбГУ,
Сергей Игоревич Николенко

Рецензент:
Доцент, К.ф-м.н.,
НИУ ВШЭ – Санкт-Петербург,
Сироткин Александр Владимирович

Санкт-Петербург
2021

# Аннотация

Одной из интересных открытых проблем из области обучения с подкреплением является обработка демонстраций эксперта, выполняющего определённую задачу, с целью более эффективного обучения новых агентов выполнять схожие задачи. Один из возможных подходов – выделение из экспертных данных крупных абстракцией или шаблонов, общих между всеми задачами. Это подобно тому, как мы, понаблюдав за кем-то, занятым игрой, можем в общих чертах понять, что в ней происходит, и, когда нам доведётся играть самим, быстро разобраться в происходящем. В данной работе был разработан метод на основе ранжирующей модели DSSM, а также среда и инструменты для изучения его достоинств и недостатков. По итогам анализа, базовая модель была доработана. Полученные модели показали хорошее качество на синтетических данных. Код и данные экспериментов доступны на GitHub.



# Abstract

One of the intriguing open challenges in the field of Reinforcement Learning is the processing of expert demonstrations data with the purpose of more efficient training of new agents to perform similar tasks. One of the possible approaches is the extraction of macro-abstractions or patterns that are common to all of the demonstrated tasks. It is similar to a situation, in which we observe someone playing a game, get a rough concept of what is going on there and, when we get to play, learn very fast. In this work a method based on a ranking DSSM model is implemented along with the environment and instruments for studying its advantages and limitations. Taking into account the results of the analysis, the baseline model is refined. The resulting models show decent quality on a synthetic dataset. Code and experiments data are available on GitHub.

# Содержание

# 1 Введение

## 1.1 Иерархическое обучение с подкреплением

*Обучение с подкреплением* – это раздел машинного обучения, задачей которого является создание агентов, оптимальным образом решающих поставленную задачу в заданной среде. Как правило, такая система описывается с помощью марковского процесса принятия решений – кортежа $(S, A, P, R)$:

- $S$ – множество состояний среды. На каждом шаге агент наблюдает одно из них.
- $A$ – множество действий, доступных агенту. На каждом шаге ему необходимо выбрать одно из них согласно его стратегии (политике).
- $P_a(s, s')$ – вероятность перехода в состояние $s' \in S$ при совершении действия $a \in A$ в состоянии $s \in S$. Не зависит от состояний и действий из прошлого.
- $R_a(s, s')$ – награда (подкрепление), выдаваемая агенту при переходе из состояния $s$ в состояние $s'$ с помощью действия $a$.

При обучении требуется найти стратегию агента, максимизирующую суммарную полученную награду. Подробнее эта постановка описана, например, в книге Sutton and Barto, Reinforcement Learning: An Introduction[1].

В общем случае это трудная задача, поскольку среда может быть устроена сложным (не известным агенту) образом, а награда может выдаваться крайне редко. Например, только при успешном выполнении всего поставленного задания целиком.

Идея *иерархического обучения с подкреплением* заключается в разбиении этой задачи на несколько уровней иерархии. Как правило, их всего два. Верхний уровень может отвечать за большие (крупные) действия, а нижний – за элементарные. К примеру, верхний уровень может решать, куда именно робот должен попасть в итоге, а нижний – как именно ему передвигать конечности, чтобы двигаться в этом направлении. Крупные действия могут позволять более эффективно исследовать сложную среду, а также предоставлять нижнему уровню внутреннюю награду за следование плану, которая может ускорить обучение агента.

## 1.2 Имитационное обучение

Для некоторых задач не представляется возможным напрямую построить функцию награды, которая бы стимулировала агента к нужному поведению, но в распоряжении имеется набор траекторий эксперта, демонстрирующего желаемые действия. Такой постановкой занимается задача *имитационного обучения* – тренировки агента, как можно более точно подражающего эксперту.

Каждая траектория имеет вид последовательности из пар состояние-действие, наблюдаемых и совершаемых экспертом: $[(s_1, a_1), (s_2, a_2), ..., (s_n, a_n)]$.

## 1.3 Обучение по наблюдениям

Обобщением имитационного обучения является *имитационное обучение по наблюдениям*, в котором нам доступны лишь наблюдения эксперта, но не его действия. Траектории имеют вид $[s_1, s_2, ..., s_n]$.

Такая постановка соответствует, например, ситуации, в которой нам доступна запись экрана человека, проходящего компьютерную игру, но не доступны его нажатия клавиш.

## 1.4 Постановка задачи

Данная работа направлена на разработку метода, позволяющего извлекать из набора наблюдений в экспертных траекториях крупные действия, которые в дальнейшем возможно использовать для обучения новых агентов. То есть необходимо только по экспертным данным, без доступа к самой среде, выучить некоторые полезные концепты об устройстве среды и поведении эксперта.

Такая постановка шире, чем обучение имитации, поскольку извлечённую информацию можно использовать для решении задачи агента, не обязательно совпадающей с задачей эксперта. К примеру, если удастся выделить большие действия в траекториях эксперта, то новый агент мог бы использовать их для более эффективного исследования среды.

Работа ведётся на языке программирования Python, нейросетевые модели и их обучение реализуются с помощью библиотеки PyTorch.

# 2 Обзор литературы

Один из способов формализовать большие действия – это *опции* [2][3][4], специальные подполитики агента. Большие действия соответствуют выбору одной из выученных опций и следованию ей до некоторых пор. Несмотря на то, что этот метод не является новым, к настоящему моменту неясно, как эффективно обучать абстрактные опции в сложных средах.

В двух других методах – FuN [5] и HIRO [6] – сама модель агента состоит из двух уровней. Верхний уровень предсказывает, в какое состояние агенту нужно стремиться попасть, а нижний предсказывает элементарные действия, которые необходимо совершать, чтобы двигаться в этом направлении. Таким образом возникает потребность по текущему состоянию среды $s$ предсказывать некоторое целевое состояние $s'$, к которому необходимо двигаться. Для этого необходимо использовать сложные генеративные (порождающие) модели.

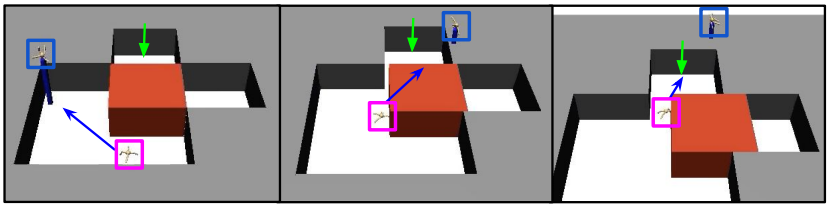

*Рис. 1: Пример целей (подняты и выделены синим), которые ставит верхний уровень модели HIRO для робота (выделен розовым)*

Метод имитационного обучения GAIL [7] добивается подражания эксперту следующим образом: в состязательной манере дискриминативная модель учится отличать экспертные траектории от траекторий агента, а агент обучается таким образом, чтобы его траектории были похожи на экспертные по мнению дискриминативной модели. Пока агент не обучен, дискриминативная модель учится отличать действия эксперта от случайных действий. Данный метод также может быть доработан [8][9] для задачи обучения по наблюдениям. В таком случае необходимо будет отличать переходы между состояниями из траекторий эксперта от переходов, соответствующих случайным действиям. Для обучения модели GAIL необходимо взаимодействие со средой. Наша работа, напротив, направлена на создание метода, получающего информацию только из траекторий эксперта, без взаимодействия со средой. Целью является выполнение некоторого предподсчёта, результаты которого можно было бы позже использовать для более эффективного обучения агентов. Непосредственное применение этих результатов лежит за рамками нашей работы.

Также из области имитационного обучения по наблюдениям к данной работе релевантен алгоритм ILPO [10], в котором по текущему состоянию предсказывается действие – одно из дискретного набора – которое должен сейчас совершить агент, а также эффект, который это действие произведёт на состояние среды. Дискретный набор действий обучается по ходу работы алгоритма.

Этот подход опирается на то, что каждый переход между двумя соседними состояниями объясняется одним из элементарных действий, и никак не использует концепцию крупных действий.

Кроме того, в данной области выделяется статья Playing hard exploration games by watching YouTube [11], где в качестве траекторий эксперта используются разнообразные видео. Основное внимание в данной работе уделяется созданию представлений, которые бы подходили для видео одной и той же игры, но разных параметров, таких как разрешение и цветовая гамма. Данный подход не опирается на концепцию больших действий, и в качестве ключевых точек рассматривает каждый $k$-ый кадр видео, от чего хотелось бы отойти.

# 3 Разработанная модель

## 3.1 Среда для тестирования

Для тестирования разрабатываемой модели была реализована среда, согласованная с интерфейсом библиотеки OpenAI Gym [12]. Среда представляет из себя клетчатое поле, внутри которого расположены пронумерованные кнопки (зелёного цвета) и сам агент (красного цвета). Размер поля был взят 5 на 5, количество кнопок взято равным 3.

*Рис. 2: Пример начального состояния среды. Кнопки с меньшим номером более яркие.*

Агенту доступны 5 действий: движение вверх, вниз, влево, вправо, а также нажатие кнопки. Агенту необходимо посетить и нажать кнопки в порядке, соответствующем их нумерации. Нажатие кнопки происходит только если агент находится ровно на нужной кнопке. Нажатые кнопки становятся синими, цвета кнопок складываются с цветом агента.

Мотивация для создания такой среды представлена далее.

Крупными действиями в данной постановке могут служить пути до следующей нужной кнопки. Хотя в клетчатом мире существует много разных оптимальных траекторий, проходящих кнопки в нужном порядке, они все разбиваются на части, в каждой из которых происходит движение от одной кнопки до следующей по номеру. Кроме того, меняя расположение кнопок, мы по сути получаем целый набор из различных сред и различных задач, которые решает эксперт. Мы будем следить за тем, чтобы разработанная модель понимала общую структуру этих задач, и выделяла некоторую информацию о переходах между кнопками. Разумеется, кроме обучения моделей на одном наборе сред, мы будем оценивать их качество на выделенном *валидационном* наборе сред, которые модель не видела при обучении.

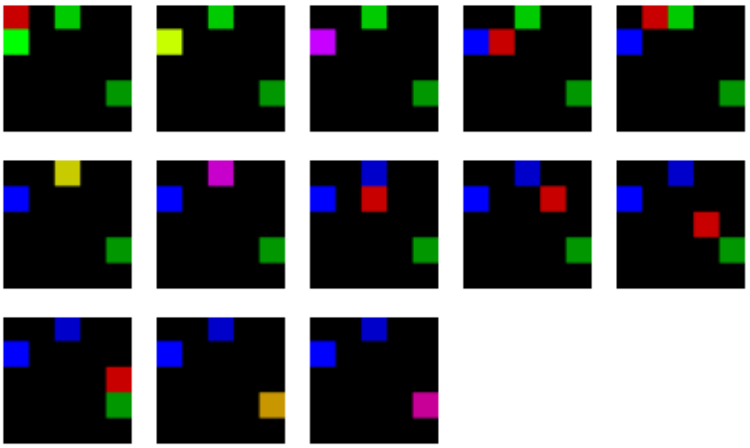

*Рис. 3: Пример экспертной траектории, порядок слева направо, сверху вниз. Агент на пустой клетке изображён красным цветом, на не нажатой кнопке – жёлтым цветом, на нажатой – фиолетовым. Не нажатые кнопки без агента изображены зелёным цветом, нажатые – синим. Цвет тем ярче, чем меньше номер кнопки.*

Для обучения и оценки моделей генерируются наборы данных, состоящих из пар $(s, s')$ состояний из экспертных траекторий. Для этого создаётся несколько (например, 100 или 1000) сред с различным положением кнопок, и в каждой из них берётся траектория агента, который на каждом шаге с вероятностью $1-\varepsilon$ совершает оптимальное действие, а с вероятностью $\varepsilon$ – случайное. Из полученной траектории $[s_1, s_2, ..., s_n]$ в набор добавляются пары $(s_i, s_j)$ для всех $i < j$.

Эпсилон брался равным 0.05. Кроме того, оптимальных путей из одной клетки в другую обычно несколько, и среди них выбирался случайный. Случайность служит для большего разнообразия экспертных траекторий.

## 3.2 Архитектура ранжирующей модели

Основное отличие предложенного в этой работе метода извлечения крупных действий от соответствующих пунктов таких алгоритмов, как FuN[5] и HIRO[6], заключается в обучении ранжирующей модели вместо генеративной. Предсказывать состояние $s'$, в которое следует совершить переход из текущего $s$, трудно, поскольку для этого нужно породить объект из сложного семейства представлений состояний $S$. Вместо этого предлагается обучать модель, выдающую число: $f(s, s')$. Оно должно показывать, насколько правильным является переход из $s$ в $s'$.

Ясно, что модель не сможет точно предсказывать, в каких именно состояниях окажется эксперт после данного $s$, поскольку оптимальных путей несколько. Однако модель могла бы выучить, что эксперт точно будет стоять на следующей кнопке, или что он будет двигаться в её направлении. Это соответствует крупным и абстрактным понятиям и представлениям об устройстве среды и экспертных задачах, которые мы как раз и хотим научиться извлекать.

Посмотрим на постановку с другой стороны: будем по паре из состояния и *перехода между состояниями* $(s, s' - s)$ предсказывать, насколько хорош донный переход из данного состояния. Здесь $s$ и $s'$ – некоторые векторные представления (эмбеддинги) состояний. Они могут быть получены напрямую из среды, либо сначала пройти через несколько слоёв нейронной сети.

Построим модель архитектуры DSSM [13]. Наша функция будет иметь вид

$$f(s, s' - s) = \exp[\alpha \langle \phi_1(s), \phi_2(s' - s) \rangle]$$

Здесь $\phi_1, \phi_2$ – две "башни" из слоёв нейронных сетей с одинаковой архитектурой, но отдельными, различными параметрами. Выход их последнего слоя нормализуется, $\alpha$ – обучаемый температурный параметр. Таким образом, модель обучает векторные представления (эмбеддинги) для состояний и переходов и выдаёт экспоненту от их скалярного произведения, домноженного на температурный параметр. Ясно, что для того, чтобы модель успешно работала, полученные с помощью $\phi_1$ и $\phi_2$ векторные представления должны извлекать из состояний и переходов некоторую информацию, на которой будет основано предсказание.

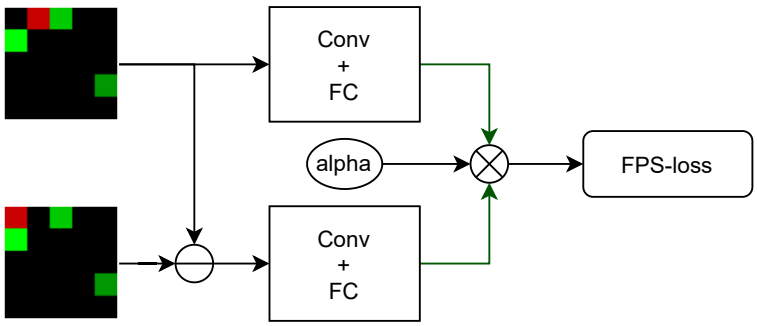


*Рис. 4: Диаграмма модели. Зелёные стрелки означают нормализацию. "Conv" и "FC" – свёрточные и полносвязные слои, "alpha" – обучаемый температурный параметр.*

Заметим, что многие существующие методы, такие как [5][6][11], привязаны к фиксированному горизонту времени. Обычно рассматриваются отрезки траекторий длины в $k$ шагов. Предложенный подход не обладает этим ограничением и направлен на обработку пар $(s, s' - s)$ вне зависимости от разности по времени между $s$ и $s'$.

**Детали реализации**

Состояние среды кодируется как многоканальное изображение, где один канал отвечает за положение агента, а остальные $2k$ – за нажатые и не нажатые

кнопки, где $k$ – количество кнопок. В каждом канале все значения нулевые, кроме, возможно, одного, указывающего на положение соответствующего объекта на клетчатом поле. Заметим, что при вычитании двух таких кодов их общие элементы сокращаются. Башни модели состоят из двух свёрточных слоёв с 16 и 32 каналами и полносвязного слоя. По умолчанию, размерность выходного вектора равна 64. Между слоями используется функция активации ReLU. В качестве оптимизатора взят Adam с параметрами по умолчанию.

### 3.3 Функция потерь FPS-loss

Для обучения ранжирующей модели необходимо демонстрировать ей позитивные и негативные пары $(s, \tilde{s}' - \tilde{s})$, и изменять параметры так, чтобы на позитивных выходное значение было выше, чем на негативных. Для этого составляется батч (набор) из пар $(s_i, s'_i - s_i)$, где $s_i, s'_i$ – состояния из экспертных траекторий, второе позже первого. Всего берётся $n_{traj}$ траекторий и по $n_{pair}$ пар из каждой. Для каждого $s_i$ позитивным примером является $s'_i - s_i$, а негативными – все $s'_j - s_j$ по $j \neq i$. Последние соответствуют случайным представлениям переходов, скорее всего не подходящих для $s_i$. Логично предположить, что модели будет легче понять, что $(s, \tilde{s}' - \tilde{s})$ – негативный пример, когда $s$ и $\tilde{s}$ взяты из разных сред. Поэтому (при $n_{pair} > 1$) батч составляется так, чтобы из одной и той же среды присутствовало несколько пар.

Для подсчёта функции потерь "FPS-loss" конструируется матрица $A_{ij}$ с элементами $F(s_i, s'_j - s_j) = \alpha\langle\phi_1(s_i), \phi_2(s'_j - s_j)\rangle$ (выход модели до экспоненты) для всех $1 \leq i, j \leq N$, где $N = n_{traj}n_{pair}$ – общий размер батча. Для этого необходимо вычислить эмбеддинги для состояний и переходов, а затем – их попарные скалярные произведения, что реализуется через простое матричное умножение и не требует больших вычислительных ресурсов.

Необходимо, чтобы в $i$-ой строке $i$-ый элемент был больше, чем остальные. Поэтому к каждой строке, как к логитам, применяется функция потерь перекрёстной энтропии для метки $i$. Например, для первой строки она равна

$$-\log\frac{\exp[F(s_1, s'_1 - s_1)]}{\sum\limits_j \exp[F(s_1, s'_j - s_j)]} = -\log\frac{f(s_1, s'_1 - s_1)}{\sum\limits_j f(s_1, s'_j - s_j)}$$

Эта функция тем меньше, чем больше $A_{11}$ по сравнению с остальными $A_{1j}$. Подобная функция потерь используется, например, в статье [14].

Покажем, где, в простом случае, достигается оптимальное значение $f$. Пусть есть две зависимые случайные величины $X$ и $Z$ с непрерывным совместным распределением, и мы применяем подобную функцию потерь для выборки из $N$ пар $(x_i, z_i)$. Изучим вклад одной строки в общую функцию потерь. Пусть эта строка соответствует элементу $x$. Среди элементов $z_1, ..., z_N$ ровно один является позитивным для $x$ и получен из распределения $\mathbb{P}(z|x)$, а остальные получены из маргинального распределения $\mathbb{P}(z)$. Тогда функция, оптимизирующая перекрёстную энтропию, должна выдавать для пары $(x, z_i)$

значение, пропорциональное вероятности того, что $z_i$ – положительный пример. Эта вероятность равна

$$\frac{p(z_i|x)\prod\limits_{j\neq i}p(z_j)}{\sum\limits_k p(z_k|x)\prod\limits_{r\neq k}p(z_r)} = \frac{\frac{p(z_i|x)}{p(z_i)}}{\sum\limits_k \frac{p(z_k|x)}{p(z_k)}}$$

Значит, оптимальным выходом модели на паре $(x, z)$ будет выход, пропорциональный $\frac{p(z|x)}{p(z)} = exp[PMI(x, z)]$, где $PMI(x, z) = \log\frac{p(x,z)}{p(x)p(z)} = \log\frac{p(z|x)}{p(z)}$ – точечная взаимная информация. При фиксированном $x$ такая функция отдаёт предпочтение тем $z$, которые много вероятнее появляются в паре с данным $x$, чем в паре со случайным. Это соответствует нашим целям: мы хотим искать переходы $s' - s$, которые эксперт часто использует из состояния $s$.

Данные рассуждения аналогичны мотивации для функции потерь в методе InfoNCE [15]. Продолжая аналогию с данным методом, можно показать, что из предложенной функции потерь возможно получить нижнюю оценку на взаимную информацию $X$ и $Z$.

Стоит отметить, что если $n_{traj} > 1$ и $n_{pair} > 1$, то функция потерь имеет схожую, но чуть более сложную структуру, поскольку столбцы разбиты на группы, соответствующие экспертным траекториям, и не равноправны.

# 4 Изучение модели, эксперименты

## 4.1 Результаты базовой модели

При обучении модели после каждой эпохи (полного прохода по обучающему набору данных) фиксируется среднее значение функции потерь и точности предсказания. Точность предсказания определяется как доля строк, в которых позитивный пример получил больший выход модели, чем все негативные. Кроме того, после прохода по обучающему набору данных совершается проход по валидационному набору (без обновления параметров модели), и на нём вычисляются те же характеристики. На рисунках 5 и 6 представлены графики обучения базовой модели в течение 240 эпох.

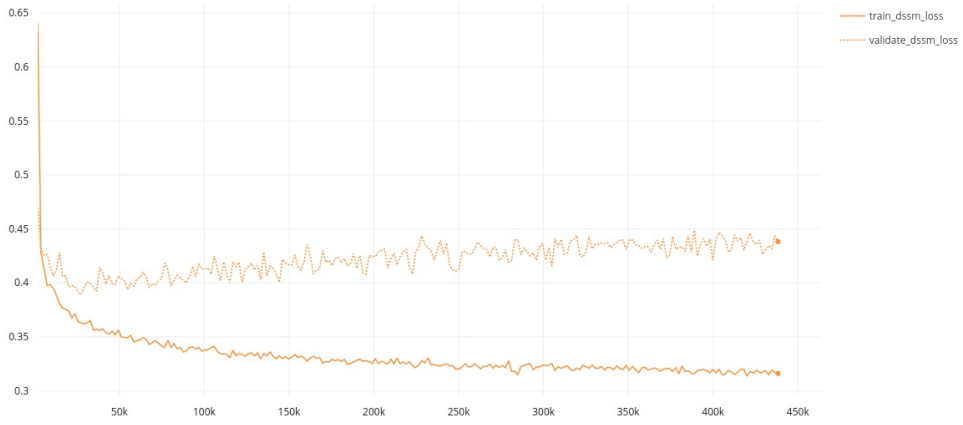


*Рис. 5: График функции потерь на тренировочных (сплошная линия) и валидационных данных (пунктирная линия). По оси абсцисс отмечены шаги обучения, то есть количество батчей, обработанных моделью. В одной эпохе их примерно 1800.*

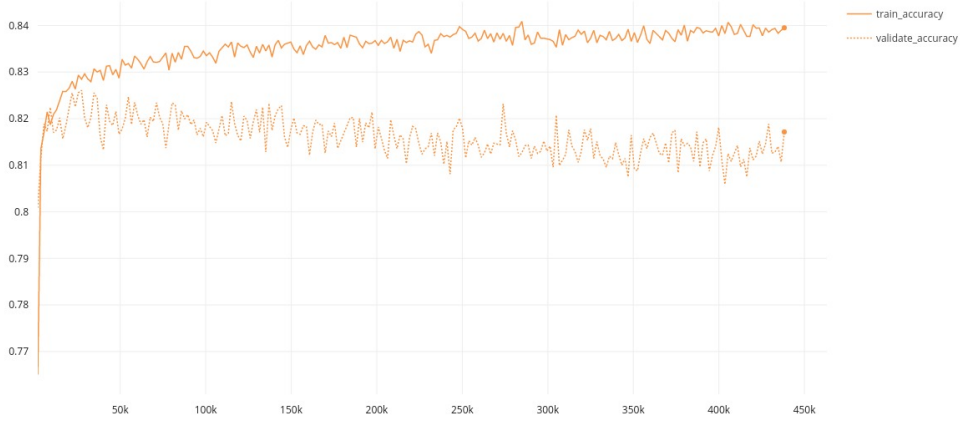


*Рис. 6: График точности предсказания на тренировочных (сплошная линия) и валидационных данных (пунктирная линия)*

Модель обучается стабильно: функция потерь на обучающей выборке падает, точность предсказания растёт. Из-за чрезмерно большого количества эпох происходит переобучение: качество на валидационных данных со временем перестаёт расти. По этой причине для остальных экспериментов количество эпох установлено равным 60 или 90. Для дальнейшего использования сохраняется модель с максимальной точностью предсказания на валидационных данных.

## 4.2 Инструменты визуализации

Поскольку наша модель должна правильным образом подбирать переходы $s' - s$ для состояний $s$, представляет интерес структура векторных представлений для переходов, получаемых с помощью обученной модели. Для изучения данной структуры переходы из обучающего набора данных пропускаются через обученную модель, а затем полученные векторы проходят кластеризацию методом k-means. Из полученных кластеров берутся представители с разными расстояниями до центроидов, и из соответствующих им пар состояний $(s, s')$ составляется визуализация данного кластера. Если модель извлекает из переходов $s' - s$ некоторую информацию о крупных действиях, то, возможно, все представления будет сгруппированы в пространстве в соответствии с этими крупными действиями. На изучение этого и направлено исследование кластеров.

В приведённом на рисунке 7 примере можно заметить, что положение агента в состоянии $s$ всегда одно и то же. Кроме этого, общая структура переходов не ясна.

Для того, чтобы в более удобном виде собирать статистику о полученных кластерах, был разработан дополнительный инструмент визуализации на основе тепловых карт (heatmap). Во-первых, для состояний $s$ и $s'$ вычисляется фаза:

- Фаза 0: агент двигается к первой кнопке
- Фаза 1: агент пришёл к первой кнопке и стоит на ней
- Фаза 2: агент нажал первую кнопку и стоит на ней
- Фаза 3: агент двигается ко второй кнопке
- ...
- Фаза 8: агент нажал последнюю кнопку и стоит на ней

Таким образом, переходу соответствует пара из фазы $s$ и фазы $s'$. Эта пара, как клетка, отмечается в матрице 9 на 9 (строки соответствуют фазе $s$, столбцы – фазе $s'$). Во-вторых, фиксируется положение агента в начальном и

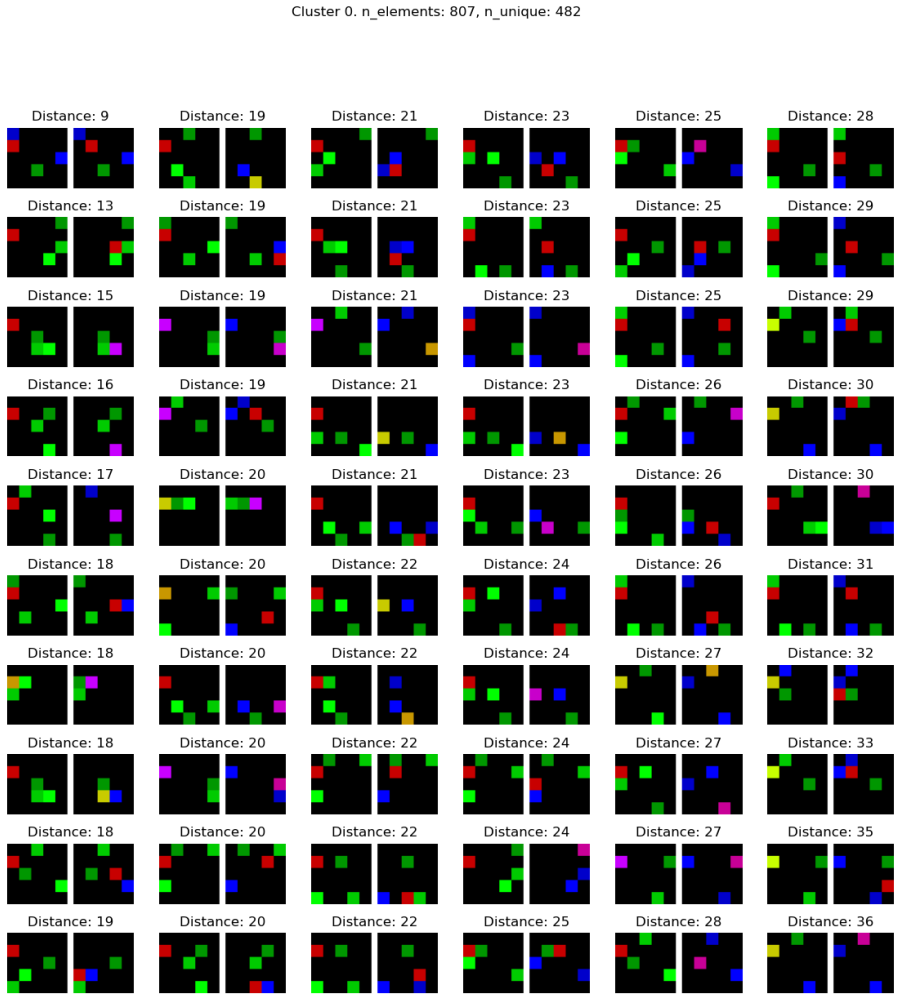


*Рис. 7: Визуализация кластера с номером 0 для базовой модели. Всего кластеров выделено 10. Большие расстояния объясняются тем, что нормализация происходит с помощью обучаемого преобразования LayerNorm, которое оставляет свободу для нормы выходного вектора.*

конечном состоянии, и они, как клетки, отмечаются в двух матрицах такого же размера, как поле среды, у нас это 5 на 5.

Клетки трёх полученных матриц отображаются тем ярче, чем больше раз они были отмечены для данного кластера.

Пример приведён на рисунке 8. Как видно, положение агента в $s$ фиксировано. Поскольку оно близко к изначальному положению агента, фаза $s$ почти всегда нулевая. При этом явной разумной структуры в положении и фазе $s'$ не прослеживается. Ситуация выглядит так в семи из десяти кластеров.

Вероятно, такое поведение обусловлено тем, что для того, чтобы отличать позитивные пары $(s_i, s'_i - s_i)$ от негативных $(s_i, s'_j - s_j)$, модель может по паре $(s, \tilde{s}' - \tilde{s})$ проверять, что $s = \tilde{s}$. Поскольку при вычитании представлений состояний информация про расположение кнопок может теряться (особенно если вычитаются чистые состояния), такую проверку проще проводить, смотря на положения агента в $s$ и $\tilde{s}$ и проверяя, что они совпадают. Такое поведение модели нежелательно, поскольку применяться обученная модель будет, ско-

рее всего, на парах $(s, s'_k - s)$ с фиксированным $s$ и разными $s'_k$, и при таком сценарии в любом случае $s = \tilde{s}$.

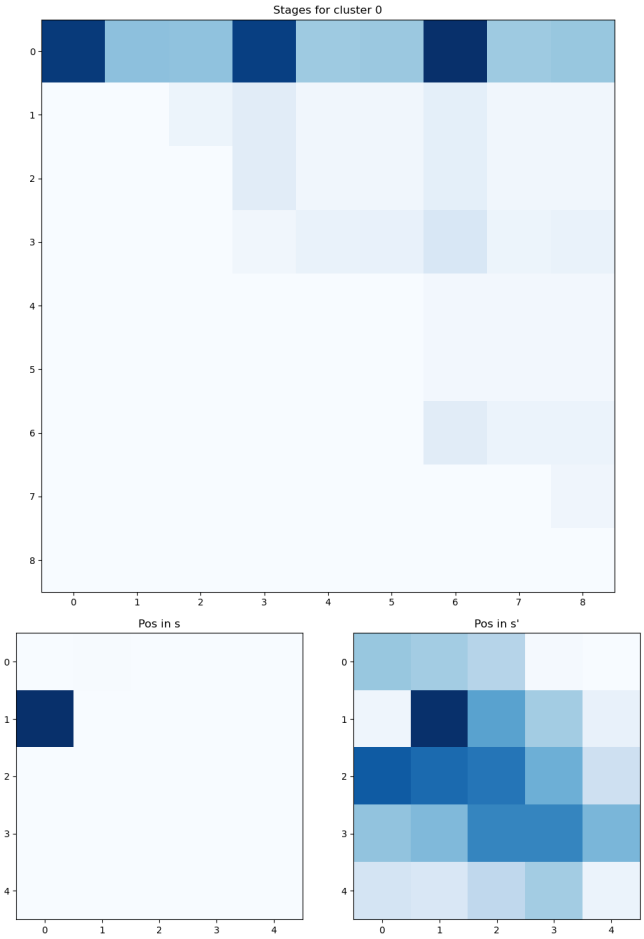


*Рис. 8: Визуализация тепловыми картами для кластера с номером 0. Верхняя карта верхнетреугольная, поскольку номер фазы растёт, кроме случаев, когда эксперт уходит с кнопки перед её нажатием в результате случайного действия.*

## 4.3 Добавление квантизации

Было решено модифицировать модель, добавив в конце второй башни операцию квантизации. Модель хранит набор из $n_z$ векторов нормы 1, координаты которых являются обучаемыми параметрами. Выход "башни" $\phi_2(s' - s)$ перестаёт быть произвольным нормированным вектором, и вместо этого заменяется на ближайший из хранимых векторов. Таким образом, представление любого перехода $s' - s$ будет иметь значение из дискретного набора из $n_z$ значений.

Мотиваций для такого решения две. Во-первых, такое "бутылочное горлышко" позволят ограничить информацию, извлекаемую из $s' - s$. Модели будет сложнее получать из перехода положение агента в начальном состоянии. Во-вторых, такое решение автоматически разбивает векторные представления переходов на кластеры, сосредоточенные вокруг дискретного набора выученных векторов.

Кроме этого, в новой модели немного по-другому обрабатываются состояния на входе: каждый цвет клетки из входного состояния переводится в вектор (эмбеддинг), после чего, опционально, общее представление состояния проходит через свёрточный слой, общий для обеих половин модели. Это сделано для того, чтобы представление состояний было более сложным, и чтобы при вычитании двух таких представлений терялось меньше информации.

На рисунке 9 представлена схема модели с квантизацией.

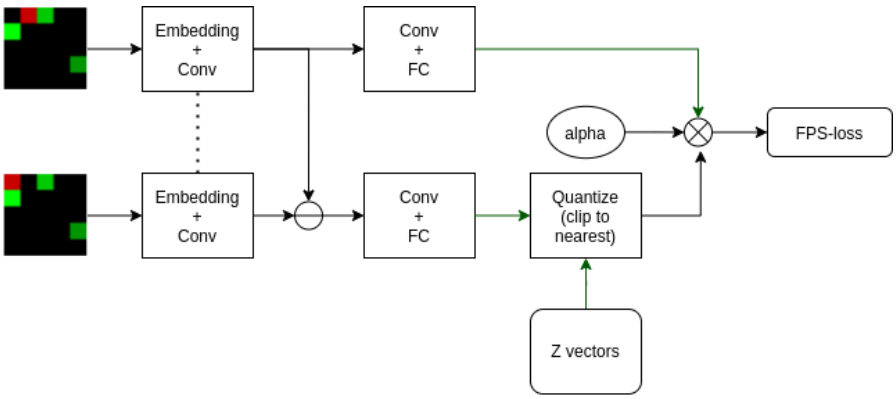


*Рис. 9: Схема модели с квантизацией. Пунктирная линия указывает на то, что параметры слоёв в данных блоках общие. "Embedding" – это перевод пикселя в векторное представление, "Conv" – свёрточный слой, "FC" – полносвязный. "Z vectors" – обучаемый набор из $n_z$ векторов. За "Quantize" обозначена операция квантизации. "alpha" – обучаемый температурный параметр.*

Трудность в обучении подобной модели заключается в том, что операция замены вектора на ближайший из дискретного набора не является гладкой. Она кусочно постоянна, и через неё не проходят градиенты. Для обхода этой проблемы используется эвристика сквозной оценки[16], по аналогии со статьёй Neural Discrete Representation Learning[17]. После подсчёта функции потерь от векторов квантизации, градиенты переносятся с них на выходы башни $\phi_2$, которые были заменены на эти векторы квантизации. В силу тонкостей реализации подобной операции на библиотеке автоматического дифференцирования, в результате данного переноса векторы квантизации остаются без градиентов. Для того, чтобы они обучались, в функцию потерь добавляется штраф за расстояние между выходами $\phi_2$ и ближайшими к ним векторами квантизации. Назовём эту добавку функцией потерь квантизации. За деталями можно обратиться к [17].

Дополнительно реализована опция, позволяющая через ещё один подсчёт функции потерь получить градиенты на векторах квантизации, и также использовать их для обучения этих векторов.

## 4.4 Результаты квантизованной модели

На рисунках 10 и 11 представлены графики обучения квантизованной модели с $n_z = 10$. Стоит отметить, что поскольку оптимизация происходит не напрямую, а используя эвристику, обучение становится сильно менее стабильным. Общая функция потерь падает не так плавно. Если посмотреть отдельно

на две её составляющие – FPS-loss и функцию потерь квантизации, то видно, что первая сначала быстро падает, затем выходит на плато и падает очень медленно, а вторая после нескольких эпох начинает падать, но затем медленно растёт. Это отражено на рисунках 12 и 13.

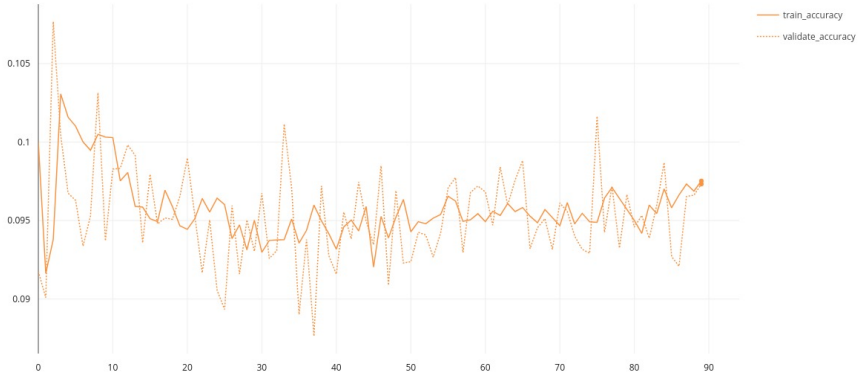


*Рис. 10: График точности предсказания на тренировочных (сплошная линия) и валидационных данных (пунктирная линия) для квантизованной модели с $n_z = 10$, обучаемой в течение 90 эпох*

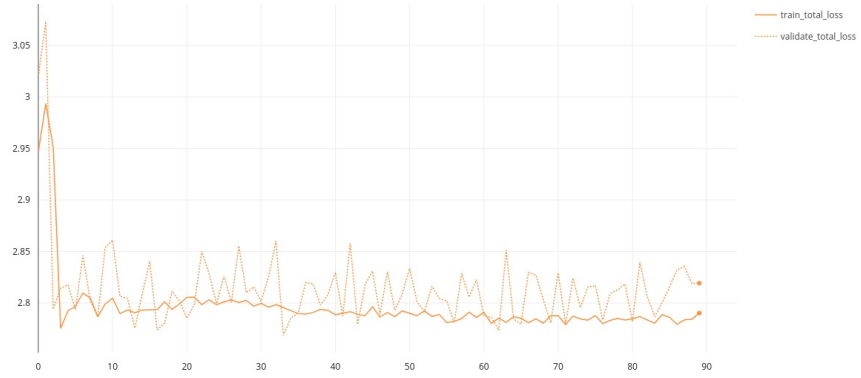


*Рис. 11: График общей функции потерь на тренировочных (сплошная линия) и валидационных данных (пунктирная линия) для квантизованной модели с $n_z = 10$, обучаемой в течение 90 эпох*

Точность предсказания не столь хорошо отражает качество модели, поскольку предсказание в $i$-ой строке считается верным, когда максимальный выход модели стоит на $i$-ом месте:

$$i = \underset{j}{argmax} A_{ij} = \underset{j}{argmax}\ \alpha\langle\phi_1(s_i), z(\phi_2(s'_j - s_j)\rangle$$

Где $z(\phi(s'_j - s_j))$ – вектор квантизации, ближайший к $\phi_2(s'_j - s_j)$. Поскольку он принимает всего $n_z$ различных значений, то среди $A_{ij}$ при фиксированном $i$ и $j = 1, 2, ..., N$ много повторяющихся элементов. А $argmin$ вычисляется в PyTorch как первое из минимальных значений. Поэтому достаточно редко будет выполняться $argmin = i$.

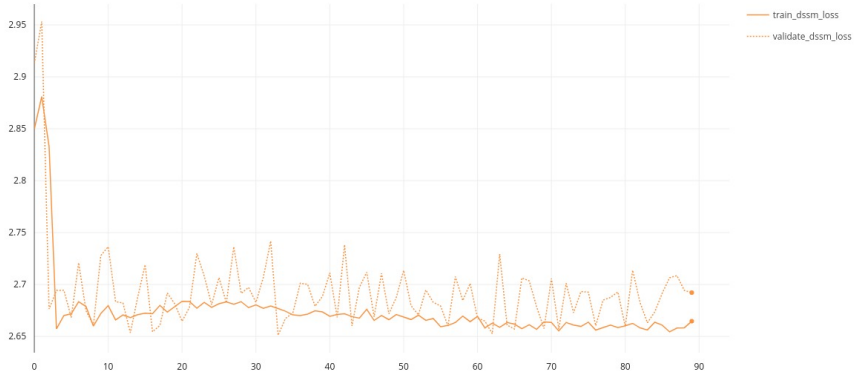


*Рис. 12: График FPS-loss на тренировочных (сплошная линия) и валидационных данных (пунктирная линия) для квантизованной модели с $n_z = 10$, обучаемой в течение 90 эпох*

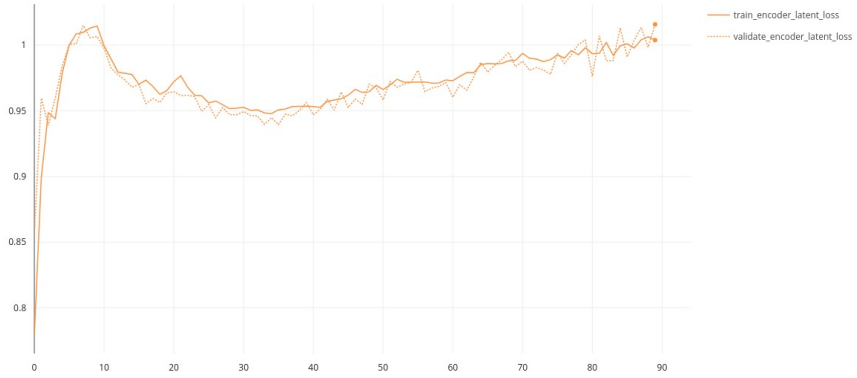


*Рис. 13: График функции потерь квантизации на тренировочных (сплошная линия) и валидационных данных (пунктирная линия) для квантизованной модели с $n_z = 10$, обучаемой в течение 90 эпох*

Функция потерь квантизации показывает среднее расстояние от выхода $\phi_2$ до ближайшего вектора квантизации и имеет значение около 1. С первого взгляда кажется, что это значение слишком велико, но, на самом деле, часть сферы, находящаяся на расстоянии не более единицы от одной из точек этой сферы – это сегмент высоты $\frac{1}{2}$. Если размерность объемлющего пространства равна $n$, то доля $n-1$-мерного объёма всей сферы, которую составляет объём данного сегмента, равна

$$\frac{1}{2} I_{0.75}\left(\frac{n-1}{2}, \frac{1}{2}\right)$$

Где $I_x(a, b)$ – регуляризованная неполная бета-функция. В нашем случае $n = 64$, и доля составляет всего около $3.5 \cdot 10^{-6}$. Она же равна вероятности, с которой случайная точка сферы попадёт к конкретному вектору квантизации на расстояние, не большее единицы. С этой точки зрения, выходы $\phi_2$ попадают близко к векторам квантизации.

На рисунке 14 представлены результаты визуализации для той же модели

с $n_z = 10$. Визуализация выполнена аналогично базовой модели, но переходы группируется в кластеры в зависимости от ближайшего к ним вектора квантизации (эмбеддинга).

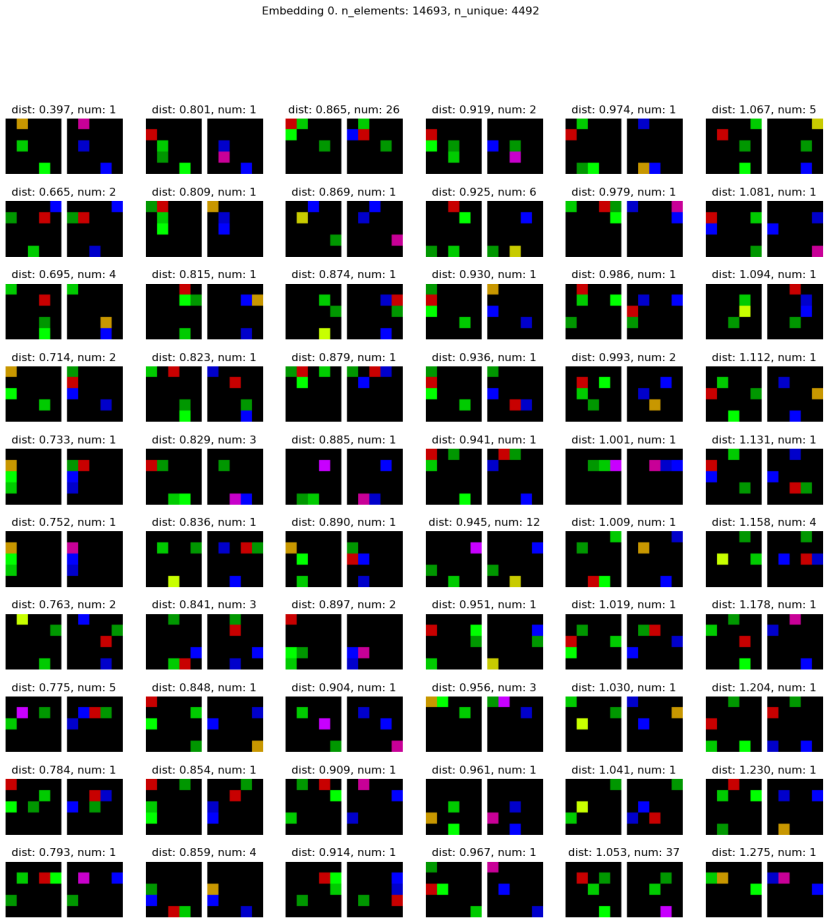


*Рис. 14: Визуализация группы переходов с векторными представлениями, ближайшими к эмбеддингу с номером 0.*

Как видно, положение агента уже не фиксировано. Кроме того, заметно, что количество нажатых кнопок не случайно: как правило, в начальном состоянии не нажато ни одной, а в конечном нажаты две или три. Это подтверждается тепловой картой на рисунке 15. Также для сравнения на рисунке 16 представлена тепловая карта, соответствующая всем переходам из обучающего набора. Поскольку из каждой экспертной траектории были взяты все пары состояний $(s, s')$, то чем позже в траектории находится $s'$, тем больше раз оно будет участвовать в парах. Поэтому в фазах конечных состояний перевес идёт в сторону более поздних. А в фазах начальных состояний, аналогично, в сторону более ранних. Кроме этого, в фазах, соответствующих пути между двумя кнопками (фазы 0, 3, 6), агент проводит больше времени, поэтому они встречаются чаще. Видно, что тепловая карта для нулевого эмбеддинга отличается от общей.

Это означает, что модель теперь учит информацию, связанную не только с положением агента, но и с количеством нажатых кнопок, к чему мы и стреми-

лись. Среди десяти эмбеддингов только у одного при визуализации положение агента оказалось фиксированным. Оно соответствует начальному положению в угловой клетке, которое встречается в каждой траектории.

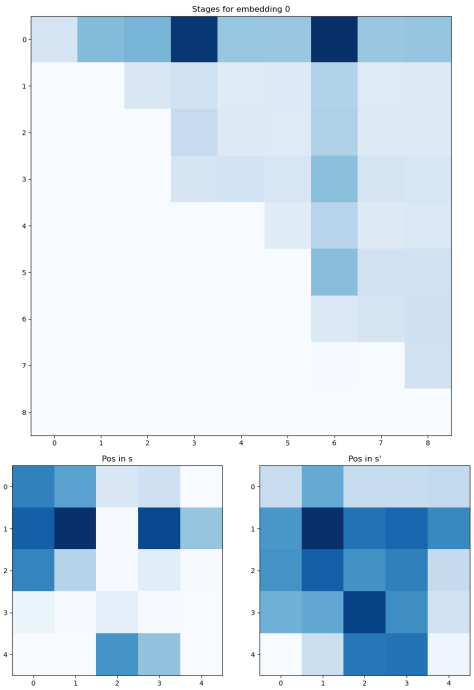


*Рис. 15: Визуализация тепловыми картами для эмбеддинга с номером 0.*

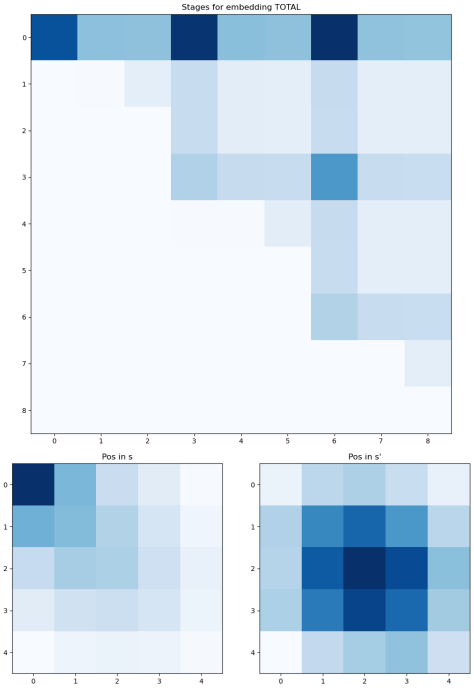


*Рис. 16: Визуализация тепловыми картами для всех переходов а обучающей выборке. Представлено для сравнения.*

## 4.5 Оценка на “синтетических” данных

Как было сказано ранее, ранжирующая модель обучается оценивать позитивные примеры выше негативных. В качестве негативных примеров необходимо использовать некоторые “неправильные” пары состояние-переход. Однако в обучающих данных присутствуют только правильные действия – действия эксперта. Для конкретного состояния $s$ из экспертной траектории, все переходы из этого же состояния, ведущие в будущее и доступные нам, являются экспертными. По этой причине в качестве неправильных переходов нам приходилось брать переходы $\tilde{s}' - \tilde{s}$ из других состояний $\tilde{s}$. И это порождало проблемы с моделью, проверяющей $s = \tilde{s}$. В идеальной ситуации, для каждого экспертного перехода $s \to s'$ мы бы хотели также располагать набором неправильных переходов $s \to s'_{neg}$, (полученных, например, случайным блужданием из $s$), чтобы демонстрировать их модели в качестве негативных примеров. К сожалению, в реальности собрать такой набор данных не представляется возможным. Даже если бы мы имели доступ к среде, в которой находился бы эксперт, было бы неоправданно трудно искать способ попасть в каждое экспертное состояние $s$, чтобы начитать из него случайное блуждание. Агента, способного попадать в любое заданное состояние среды, скорее всего, уже не нужно ничему обучать.

Однако сейчас, в целях изучения качества разработанных моделей, нет никаких препятствий к созданию описанного набора данных. Для этой цели был собран тестовый “синтетический” набор данных следующим образом:

1. Создать среду со случайным расположением кнопок
2. Построить в ней экспертную траекторию
3. Взять два случайных состояния $s$ и $s'$ из данной траектории (второе позже первого)
4. Несколько раз блуждать, выполняя случайные действия, начиная из $s$, до тех пор, пока мы не придём в состояние $s'_{neg}$, такое что $dist(s, s'_{neg}) = dist(s, s') + \delta$, где $\delta$ – случайное число из $\{-1, 0, 1\}$, $dist$ – обычная $L_1$-метрика в клетчатом поле
5. Сформировать набор из позитивного примера $(s, s' - s)$ и негативных примеров $(s, s'_{neg} - s)$

Случайное блуждание заканчивается в точке, отстоящей от начального положение примерно на столько же, как в экспертном переходе. Это сделано для того, чтобы позитивные примеры нельзя было отличать от негативных по расстоянию между начальным и конечным состоянием. Количество негативных примеров взято равным 4.

Логично предположить, что в экспертных переходах будут часто нажиматься кнопки, а в случайных – редко. Статистика по количеству нажатых

кнопок за переход представлена в таблицах 1 и 2. Оказывается, что долгое случайное блуждание с неплохой вероятностью нажимает на кнопки. Разница с экспертными переходами всё ещё есть, однако эксперимент показывает, что алгоритм “считать позитивным тот переход, в течение которого нажато больше всего кнопок” даёт точность предсказания лишь 62%.

| | 0 нажатий | 1 нажатие | 2 нажатия | 3 нажатия |
|---|---|---|---|---|
| Позитивные примеры | 27.4% | 42.9% | 24.9% | 5.8% |
| Негативные примеры | 88.7% | 10.1% | 0.8% | 0.4% |

*Таблица 1: Распределение количеств нажатых кнопок в позитивных (экспертных) и негативных (случайных) переходах*

| | 0 нажатий | 1 нажатие | 2 нажатия | 0 нажатий |
|---|---|---|---|---|
| 0 нажатий эксперта | 97.9% | 1.8% | 0.2% | 0.1% |
| 1 нажатие эксперта | 87.1% | 12.1% | 0.5% | 0.3% |
| 2 нажатия эксперта | 82.9% | 14.8% | 1.6% | 0.8% |
| 3 нажатий эксперта | 80.9% | 15.2% | 2.0% | 1.8% |

*Таблица 2: Распределение количества нажатий в случайных переходах при условии заданного числа нажатий в экспертных переходах*

Для того, чтобы иметь верхнюю границу на точность предсказания, было получено, что базовая модель, обучающаяся на большом наборе синтетических данных, показывает на тестовом наборе максимальную точность 87%. Нетрудно понять, что к идеальной точности не имеет смысла стремиться, поскольку случайные переходы могут иногда быть оптимальными.

С помощью полученного тестового набора проводились измерения точности предсказания его позитивных примеров против негативных. Эта характеристика показывает, насколько хорошо модель выучила, как действует эксперт. Если она высока, то можно говорить, что модель выучила некоторую полезную информацию и подходит для имитационного обучения.

На рисунке 17 представлены результаты трёх моделей. Точность предсказаний получилась в районе 70% – 85%, что хорошо говорит об обученных моделях. Несмотря на описанные недостатки базовой модели, ей качество на синтетических данных лучше, чем у квантизованных.

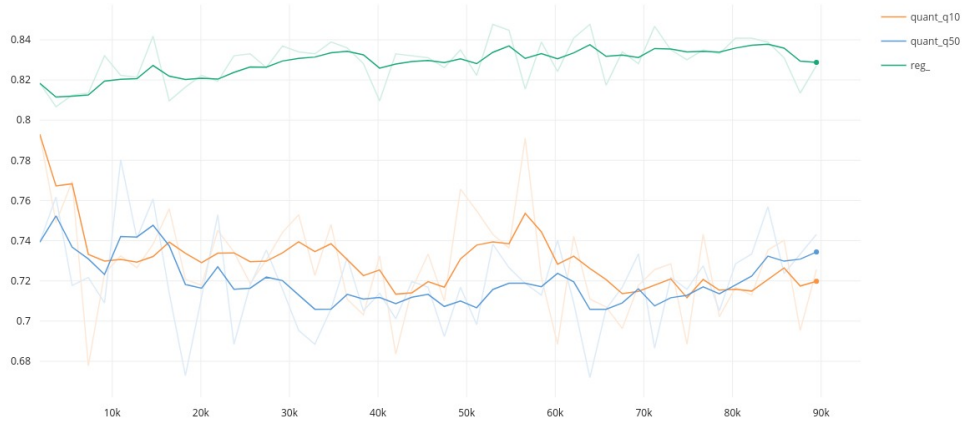


*Рис. 17: График точности предсказания на синтетических данных для базовой модели (зелёный), квантизованной с $n_z = 10$ (оранжевый) и квантизованной с $n_z = 50$ (голубой). Графики сглажены, исходные линии показаны бледно.*

## 4.6 Дополнительные эксперименты

В данном разделе представлены результаты экспериментов, исследующих зависимость точности предсказаний на валидационном или синтетическом наборе данных от различных параметров модели и процесса её обучения.

**Размер векторного представления базовой модели**

В базовой и квантизованной модели башни $\phi_1$ и $\phi_2$ переводят состояние и переход в векторные представления размерности 64. На рисунках 18 и 19 представлен перебор других размерностей: 32, 16, 8 и 128. Существенного влияния на качество этот параметр не оказывает, поэтому в остальных экспериментов использовалася параметр по умолчанию – 64.

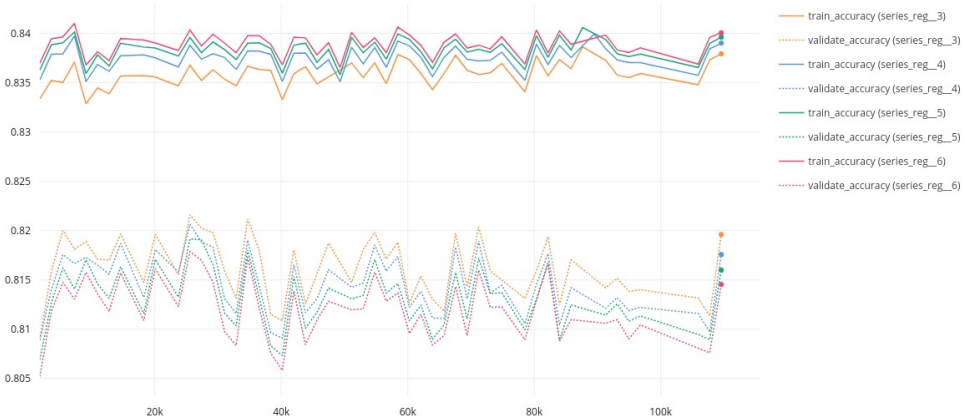


*Рис. 18: График точности предсказания на обучающей (сплошные линии) и валидационной (пунктирные линии) выборке для разных размерностей векторных представлений: 32 (оранжевый), 16 (голубой), 8 (зелёный) и 128 (малиновый). Модели ведут себя практически одинаково. По оси абсцисс отмечены шаги обучения – количество батчей, обработанных моделью. В одной эпохе их около 1800.*

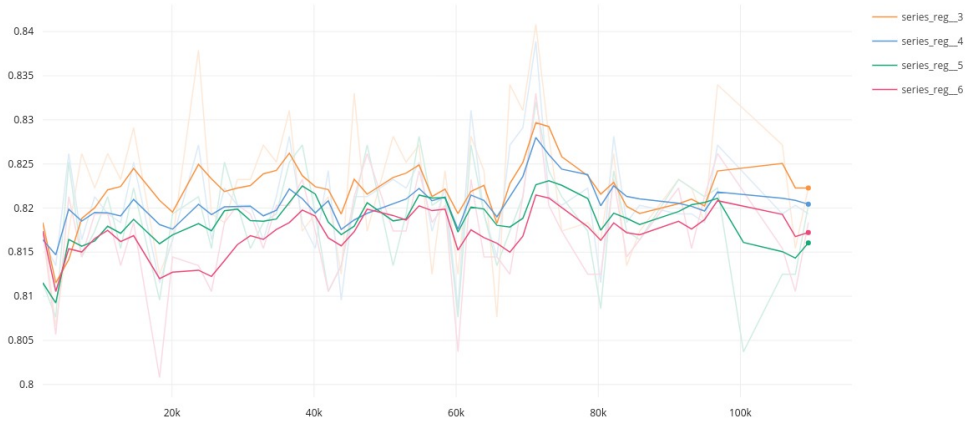


*Рис. 19: График точности предсказания на синтетических данных для разных размерностей векторных представлений: 32 (оранжевый), 16 (голубой), 8 (зелёный) и 128 (малиновый). Модели ведут себя схожим образом. Графики сглажены, исходные линии показаны бледно.*

### Различные компановки батча

Следующий эксперимент перебирает пары значений $(n_{traj}, n_{pair})$ – количества траекторий и пар из каждой траектории, используемых для создания одного батча (порции) данных. Помимо используемой по умолчанию пары $(16, 4)$, испробованы пары $(8, 8), (4, 16), (2, 32), (1, 32)$. Более 32-ух пар из одной траектории брать проблематично, потому что во многих траекториях 64-ёх пар без повторений уже не набирается. Для базовой модели лучшим оказывается пара $(16, 4)$, а уменьшение $n_{traj}$ и увеличение $n_{pair}$ только уменьшают качество на синтетических данных. Удивительно, но для модели с квантизацией, напротив, пара $(1, 32)$ показывает заметно большее качество, чем все остальные, приближаясь к качеству базовой модели. Результаты отражены на рисунках 20 и 21.

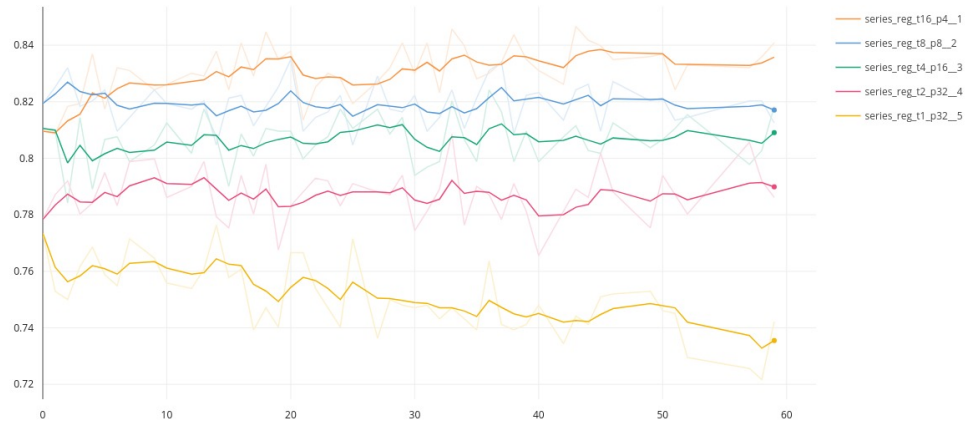


*Рис. 20: Сглаженный график точности предсказания на синтетических данных для базовой модели и разных пар $(n_{traj}, n_{pair})$: $(16, 4)$ (оранжевый), $(8, 8)$ (голубой), $(4, 16)$ (зелёный), $(2, 32)$ (малиновый), $(1, 32)$ (жёлтый).*

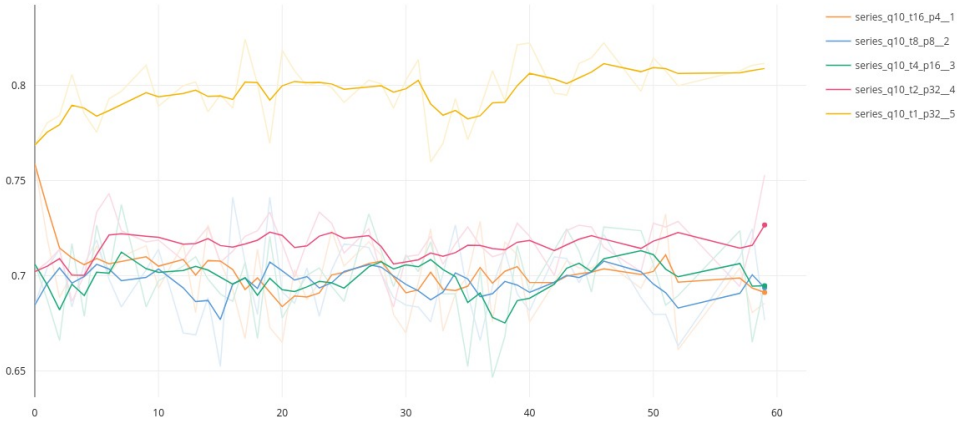


*Рис. 21: Сглаженный график точности предсказания на синтетических данных для модели с* $n_z = 10$ *и разных пар* $(n_{traj}, n_{pair})$*:* $(16, 4)$ *(оранжевый),* $(8, 8)$ *(голубой),* $(4, 16)$ *(зелёный),* $(2, 32)$ *(малиновый),* $(1, 32)$ *(жёлтый).*

### Более сложное представление состояний в базовой модели

Реализация квантизованной модели позволяет отключить операцию квантизации, по сути превращая модель в базовую. Однако состояния в таком случае сперва проходят дополнительный блок из создания векторного представления и свёрточного слоя. Данный эксперимент показывает разницу такой модели с чистой базовой. Как видно на рисунках 22 и 23, на тренировочной и валидационной выборке усложнённая модель ведёт себя лучше, но на синтетических данных – хуже. Вероятно, это связано с тем, что усложнённая модель более гибкая и лучше выучивает информацию о положении агента, хуже обобщаясь на информацию о кнопках. Таким образом, усложнение базовой модели не оправдано.

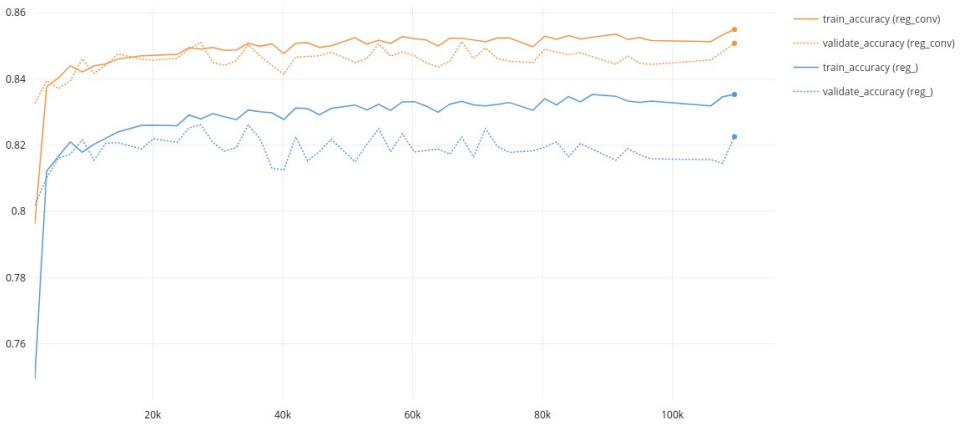


*Рис. 22: График точности предсказания на обучающей (сплошные линии) и валидационной (пунктирные линии) выборке для базовой модели (голубой цвет) и усложнённой модели (оранжевый цвет).*

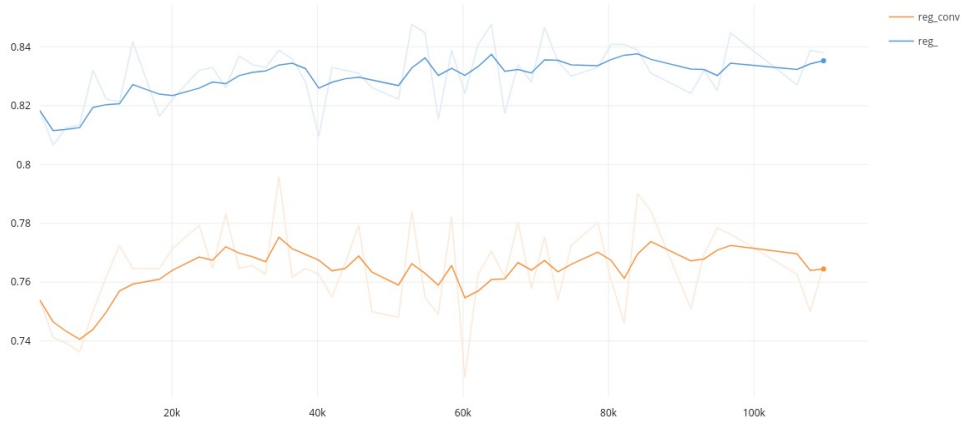


*Рис. 23: Сглаженный график точности предсказания на синтетических данных для базовой модели (голубой цвет) и усложнённой модели (оранжевый цвет).*

**Параметр $n_z$ квантизованной модели**

Представляет интерес влияние количества $n_z$ векторов квантизации на поведение модели. Изучены значения $3, 7, 10, 25, 50, 100$. Графики на рисунках 24 и 25 показывают, что качество постепенно растёт с ростом $n_z$. Однако стоит учитывать, что для больших $n_z$ модель становится сложнее для интерпретации и использования. Преимущество моделей с малым $n_z$ состоит в том, что они разбивают всё множество переходов на небольшое число групп.

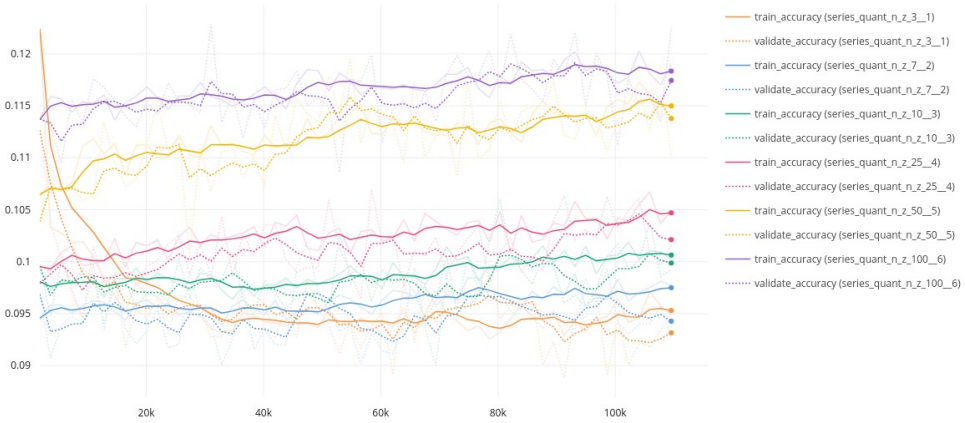


*Рис. 24: Сглаженный график точности предсказания на обучающей (сплошные линии) и валидационной (пунктирные линии) выборке для квантизованной модели с $n_z$, равным 3 (оранжевый), 7 (голубой), 10 (зелёный), 25 (малиновый), 50 (жёлтый), 100 (фиолетовый).*

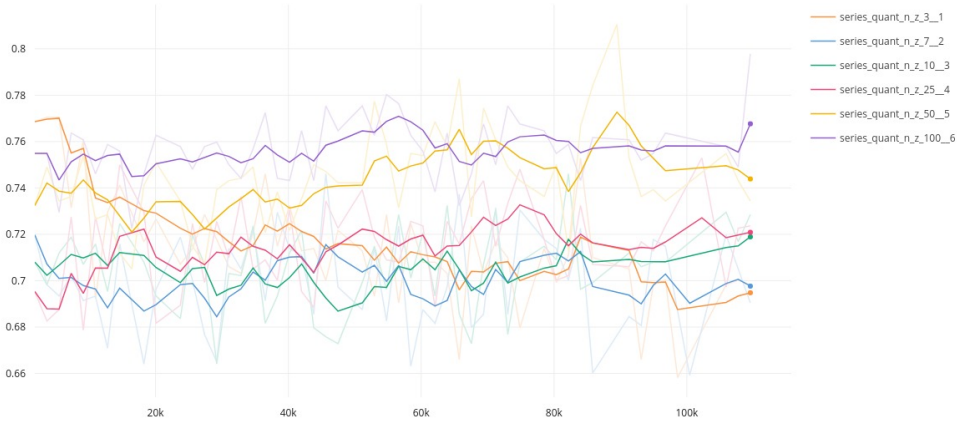


*Рис. 25: Сглаженный график точности предсказания на синтетических данных для квантизованной модели с $n_z$, равным 3 (оранжевый), 7 (голубой), 10 (зелёный), 25 (малиновый), 50 (жёлтый), 100 (фиолетовый).*

### Размер обучающей выборки

Все описанные выше модели обучались на наборе, полученном из 1000 различных сред. В нём около 117000 пар состояние-переход. Данный эксперимент показывает, что произойдёт, если обучающий набор уменьшить в 10 раз. Закономерно, что на маленьком обучающем наборе модели быстрее переобучаются. Однако если вовремя остановить обучение, то модели с уменьшенным набором данных показывают качество, сравнимое с моделями с обычным. На рисунках 26 и 27 представлены графики для базовой модели, на рисунках 28 и 29 – для квантизованной с $n_z = 10$. Можно сделать вывод о том, что наши модели способны обучаться и при меньшем количестве доступных данных.

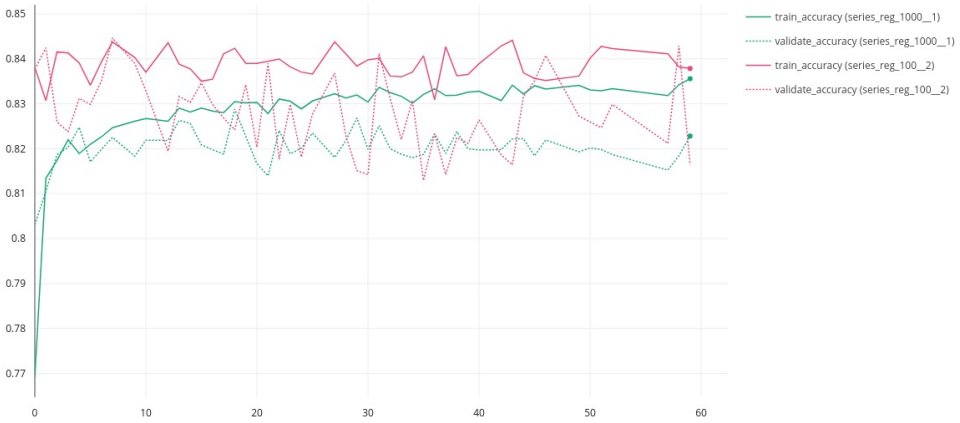


*Рис. 26: График точности предсказания на обучающей (сплошные линии) и валидационной (пунктирные линии) выборке для базовой модели с большой обучающей выборкой (зелёный) и маленькой (малиновый). По оси абсцисс отложены эпохи, поскольку размеры выборок, а значит, и количество батчей в них, разные.*


*Рис. 27: Сглаженный график точности предсказания на синтетических данных для базовой модели с большой обучающей выборкой (зелёный) и маленькой (малиновый).*

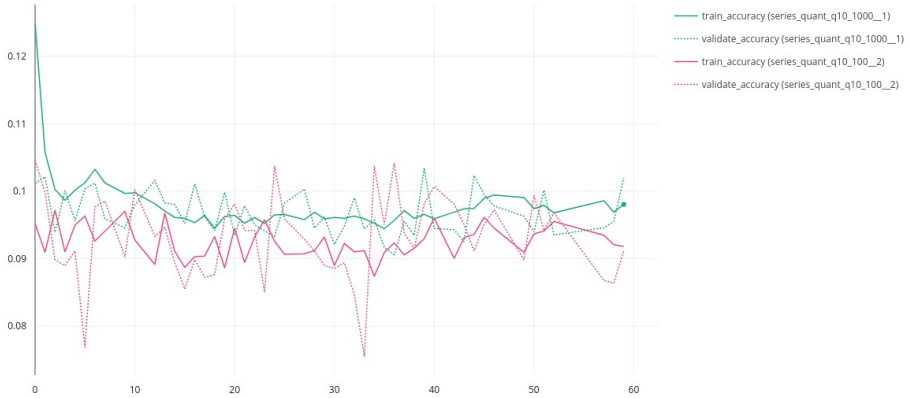


*Рис. 28: График точности предсказания на обучающей (сплошные линии) и валидационной (пунктирные линии) выборке для модели с $n_z = 10$ с большой обучающей выборкой (зелёный) и маленькой (малиновый). По оси абсцисс отложены эпохи, поскольку размеры выборок, а значит, и количество батчей в них, разные.*

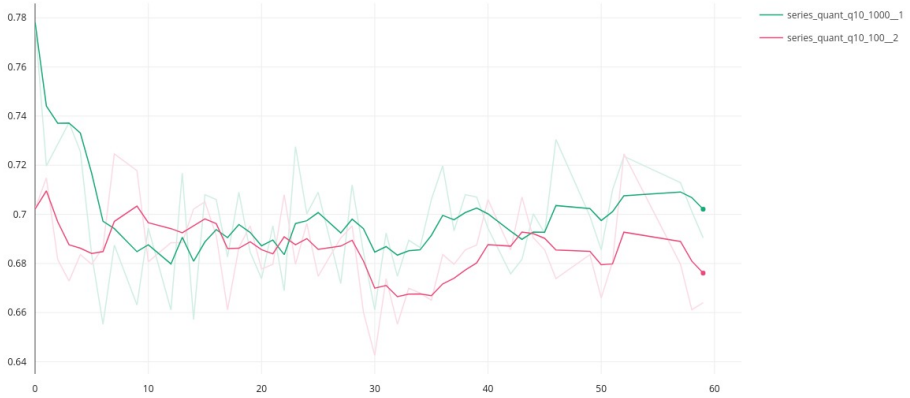


*Рис. 29: Сглаженный график точности предсказания на синтетических данных для модели с $n_z = 10$ с большой обучающей выборкой (зелёный) и маленькой (малиновый).*

**Подбор коэффициентов в функции потерь**

Функция потерь квантизованной модели состоит из трёх основных частей:

1. FPS-loss, градиенты которого переносятся и распространяются на параметры $\phi_2$
2. (опционально) FPS-loss, градиенты которого остаются на параметрах векторов квантизации
3. Функция потерь квантизации, отвечающая за близость выхода $\phi_2$ и ближайшего вектора квантизации. Градиенты распространяются и на параметры $\phi_2$, и на параметры векторов квантизации

Общая функция потерь составляется как взвешенная сумма этих трёх.

В представленном эксперименте коэффициенты перед ними пробегают все комбинации из значений $\{0.3, 1, 3\}$ для функций под номерами 1. и 2. и значений $\{0.03, 0.1, 0.3\}$ для функции под номером 3. Результаты видны на рисунке 30

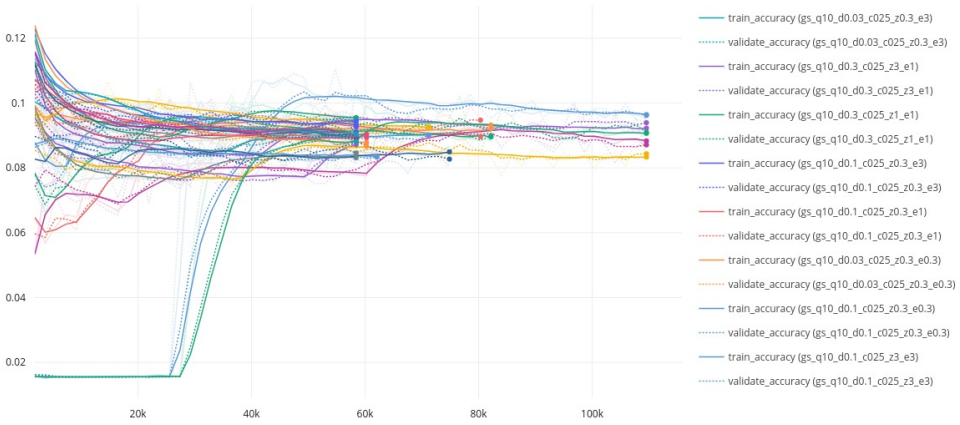


*Рис. 30: Сглаженный график точности предсказания на обучающей (сплошные линии) и валидационной (пунктирные линии) выборке для модели с $n_z = 10$ и разными вариантами коэффициентов в общей функции потерь. Некоторые линии короче, потому что обучение останавливалось, если долго не происходило улучшения качества на валидационной выборке.*

Особого влияния на качество моделей нет, кроме двух случаев, когда коэффициент перед функцией потерь квантизации велик по сравнению с остальными. В таком варианте модель сильно штрафуется за расстояние между выходом $\phi_2$ и вектором квантизации, и в начале обучения она просто делает выход $\phi_2$ практически постоянным и близким к одному из векторов квантизации. Для того, чтобы отлавливать такие моменты, собиралась информация о том, какая доля выходов $\phi_2$ заменяется на каждый из векторов квантизации. На рисунке 31 виден переходный момент, когда модель из данного эксперимента перестаёт переводить все выходы $\phi_2$ в один и тот же вектор. Как правило,

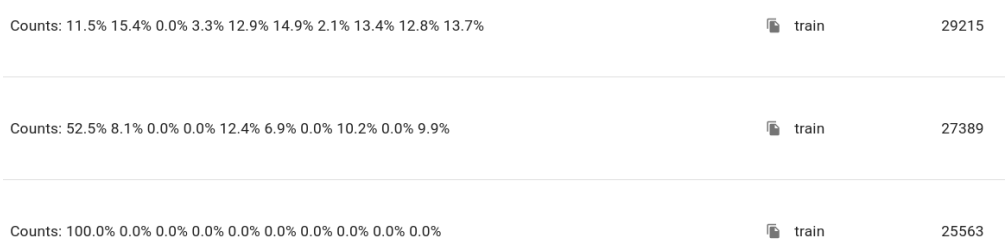


*Рис. 31: Записи о долях выходов $\phi_2$, отправляющихся в каждый из векторов квантизации, для эксперимента с большим коэффициентом перед функцией потерь квантизации. Справа указан номер шага обучения. Виден фазовый переход.*

во всех остальных экспериментах доли выходов распределялись по векторам практически равномерно.

Также для моделей с квантизацией велись записи таблиц попарных расстояний между обучаемыми $n_z$ векторами. Это сделано для проверки того, что, например, все векторы не сконцентрированы около одной точки. На рисунке 32 представлен пример такой таблицы. Никаких аномалий не наблюдается.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1.45 | 1.36 | 1.47 | 1.46 | 1.47 | 1.50 | 1.55 | 1.46 | 1.51 |
| 1 | 1.45 | 0 | 1.39 | 1.45 | 1.50 | 1.48 | 1.50 | 1.53 | 1.50 | 1.52 |
| 2 | 1.36 | 1.39 | 0 | 1.41 | 1.58 | 1.47 | 1.50 | 1.64 | 1.42 | 1.53 |
| 3 | 1.47 | 1.45 | 1.41 | 0 | 1.49 | 1.48 | 1.46 | 1.52 | 1.49 | 1.50 |
| 4 | 1.46 | 1.50 | 1.58 | 1.49 | 0 | 1.48 | 1.47 | 1.48 | 1.47 | 1.45 |
| 5 | 1.47 | 1.48 | 1.47 | 1.48 | 1.48 | 0 | 1.50 | 1.55 | 1.42 | 1.50 |
| 6 | 1.50 | 1.50 | 1.50 | 1.46 | 1.47 | 1.50 | 0 | 1.51 | 1.51 | 1.50 |
| 7 | 1.55 | 1.53 | 1.64 | 1.52 | 1.48 | 1.55 | 1.51 | 0 | 1.52 | 1.45 |
| 8 | 1.46 | 1.50 | 1.42 | 1.49 | 1.47 | 1.42 | 1.51 | 1.52 | 0 | 1.52 |
| 9 | 1.51 | 1.52 | 1.53 | 1.50 | 1.45 | 1.50 | 1.50 | 1.45 | 1.52 | 0 |

*Рис. 32: Попарные расстояния между векторами квантизации в одной из моделей.*

# 5 Способы применения модели

В данном разделе описаны потенциальные способы использования разработанных моделей для более эффективного обучения новых агентов.

Результаты оценки обученных моделей на синтетических данных показывают, что модели хорошо отличают экспертные действия от случайных. Это позволяет использовать их для имитационного обучения. Поскольку мы обучили функции $f$, предсказывающие по паре $(s, s' - s)$ то, насколько переход $s \to s'$ похож на экспертный, их можно использовать следующим образом: когда агент делает $n$-ый шаг в своей траектории $(s_0, s_1, ..., s_n)$, мы можем вычислить величину

$$r_n = \sum_{i=1}^{n} \alpha_i f(s_{n-i}, s_n - s_{n-i})$$

и использовать её напрямую для функции потерь или как дополнительную награду (подкрепление) для обучаемого агента. Данная величина (при некотором наборе положительных коэффициентов $\{\alpha_i\}$) является мерой того, насколько последний переход агента похож на экспертный, если усреднить краткосрочный и долговременный контекст. То есть она учитывает не только близкие пары состояний, но и далёкие.

Если агент учит модель мира, то есть учится по состоянию и действию предсказывать состояние, в которое он попадёт, то возможно оценить каждый вариант следующего состояния с помощью наших функций и выбрать действие с учётом результата.

Наша модель также обучает векторные представления для состояний и переходов. Большая часть моделей агентов обучения с подкреплением также использует векторные представления состояний, поскольку работать, например, с чистыми изображениями было бы слишком трудно ввиду их большой размерности. В таком случае появляется проблема холодного старта: пока разумные представления состояний ещё не обучены, агент не сможет извлекать полезную информацию из среды и учиться совершать правильные действия. Для решения этой проблемы возможно использовать предобученные векторные представления нашей модели в качестве инициализации векторных представлений агента. Поскольку наши модели не требуют взаимодействия со средой, такой подход может улучшить sample efficiency агента, то есть объём взаимодействия со средой, требуемый для обучения агента, что важно во многих задачах.

Наконец, квантизованная модель позволяет выделить небольшой набор из $n_z$ представлений переходов, которые играют важную роль в траекториях эксперта и могут являться большими действиями. Значит, эти представления возможно использовать для инициализации методов иерархического обучения с подкреплением, использующих большие действия, тоже для решения проблемы холодного старта. Например, в методе FuN[5] верхнему уровню иерархии, “менеджеру”, необходимо выдавать направление, в котором агенту необходимо

двигаться (в терминах латентного векторного пространства, в котором находятся представления состояний). На этом шаге можно квантизовать выход менеджера к ближайшему из предобученных векторов нашей модели.

# 6 Заключение

В данной работе исследованы достоинства и недостатки применения ранжирующей модели DSSM для извлечения крупных действий из наблюдений в траекториях эксперта.

Для этого разработана иллюстративная среда, позволяющая получить набор задач и экспертных траекторий, из которых возможно выделить крупные действия. Для обучения и тестирования (валидации) моделей собраны наборы экспертных данных.

Реализация и тестирование базовой модели позволило увидеть проблемы используемой функции потерь. А именно, тот факт, что её возможно оптимизировать, выучивая не релевантную нам информацию. Для борьбы с этим была разработана новая, более сложная версия модели, включающая операцию квантизации. Было проверено, что модель с квантизацией, действительно, обучается более полезным концептам.

Для оценки полученных моделей был собран тестовый набор данных, показывающих, насколько хорошо модели определяют экспертные действия. Были проведены эксперименты по подбору параметров, и все полученные модели, включая базовую, показали хорошее качество. что подтверждает, что все модели выделяют из данных полезную информацию и пригодны для имитационного обучения.

В завершение, были обсуждены возможные способы применения обученных моделей.

# 7 Благодарность

Автор благодарит Вячеслава Алипова за регулярное внимание и наставления к данной работе.

# 8 Список литературы

## Список литературы


[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[2] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

[3] Doina Precup. Temporal abstraction in reinforcement learning, 2000.

[4] Martin Stolle and Doina Precup. Learning options in reinforcement learning. volume 2371, pages 212–223, 08 2002.

[5] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017.

[6] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *CoRR*, abs/1805.08296, 2018.

[7] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

[8] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation, 2019.

[9] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement, 2019.

[10] Ashley D. Edwards, Himanshu Sahni, Yannick Schroecker, and Charles L. Isbell. Imitating latent policies from observation, 2019.

[11] Yusuf Aytar, Tobias Pfaff, David Budden, Tom Le Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube, 2018.

[12] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. cite arxiv:1606.01540.

[13] Po-Sen Huang, Xiaodong He, Jianfeng Gao, li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. pages 2333–2338, 10 2013.

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

[15] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[16] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013.

[17] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018.