Проект по дисциплине "Глубинное обучение"

Чистяков Глеб
Группа 162
gichistyakov@edu.hse.ru

Цыбакин Александр
Группа 162
ayutsybakin@edu.hse.ru

1 Предисловие

Обучение с подкреплением (Reinforcement Learning) – это метод машинного обучения, при котором происходит обучение модели посредством взаимодействия со средой, не имея никаких знаний о ней, но имея возможность производить действия, переходить в новые состояния среды и получать вознаграждение в результате своих действий. Таким образом, модель обучается на своих ошибках, стараясь получать наибольшую награду. Такой подход набирает популярность, его применение все чаще можно встретить в решении разного рода задач из различных сфер, например, финансовом трейдинге или управлении беспилотным автомобилем. Однако, основным применением обучения с подкреплением являютя разнообразые игры по некоторым причинам, одной из которых является наличие конечного набора состояний среды.

Наш проект основан на статье “Learning to Act by Predicting the Future”[1], где описана имплементация и результы применения Direct Future Prediction-модели в игре Doom.

2 Описание подхода

В статье “Learning to Act by Predicting the Future”[1] авторы предлагают новую модель Future Prediction Networl(FPN-модель) и подход к ее обучению в задаче взаимодействия со средой. Такая задача решается подходами обучения с подреплением, в котором выделяют Агента и Среду. Агент взаимодействует со Средой и наоборот. Посредством этого взаимодействия происходит обучение Агента. У Агента также есть состояния относительно Среды, набор действий для перехода из одного состояния в другое и некоторая система вознаграждения.

Авторы обращают внимание на отличие их подхода от стандартного подхода обучения с подкреплением. Здесь они вместо стандартной системы вознаграждений используют “Сенсорный поток” и “Поток измерений”. “Сенсорный поток” – это то, что получает Агент (модель) в высокой размерности, например визуальные, аудиторные или иные виды данных (изображения, звук и т.д.). “Поток измерений” – это то, что получает модель в низкой размерности и относится к текущему состоянию Агента. Таковыми могут быть структурная целостность, уровень запасов и т.д. Для игр, например, это могут быть уровень жизни или боеприпасов.

3 Описание задачи

Модель действует в среде игры Doom. Для построения среды используется симулятор. Он является некоторым 3D-пространством с дополнениями в зависимости от сценария, например, возможно наличие объектов: аптечки, яд. На языке обучения с подкреплением: Агент – модель-игрок, среда – 3D-симулятор игры Doom. Набор действий Агента состоит из перемещения по карте симулятора, стрельбу, прыжки и прочее, что также зависит от сценария.

Как и в реальный игрок, Агент имеет вид от первого лица на Среду и индикаторы здоровья и оружия. Он должен действовать в соответствии с поступающей визуальной и метрической информацией как настоящий игрок. Здесь в качестве “Сенсорного потока” подается изображения пространства от первого лица, а в качестве “потока измерений” - уровень жизни и боеприпасов.

Всего авторы выделяют 4 сценария по разному уровню сложности:

1. D1 - игрок собирает комплекты аптечек в квадратной комнате
2. D2 - игрок собирает комплекты аптечек в лабиринте, избегая сосуды с ядом
3. D3 - игрок защищается от противников в лабиринте, собирая комплекты аптечек и оружия
4. D4 - игрок защищается от противников, собирая комплекты аптечек и оружия, в более сложном лабиринте (появляется разноуровневые локации)

4 Описание модели

В каждый момент времени у Агента есть наблюдения o_t и он производит некоторые дейсвия a_t из набора действий. Наблюдения имеют следующий вид: $o_t =< s_t, m_t >$, где s_t – объект, который приходит от “Сенсорного потока” в момент t, в нашем случае изображение игрового поля от первого лица, m_t – набор метрик от “Потока измерений” в момент t, в нашем случае такими метриками, например, могут быть: уровень жизни, боеприпасов, количество убитых врагов и другие. Эти метрики содержатся в векторе “измерений”.

Здесь вектор измерений m выделен отдельно от сенсорной составляющей s, так как основная идея заключается в предсказании значений этого вектора с некоторым временным смещением. Введем вектор f изменений текущего значения измерений от значений измерений с некоторым временным сдвигом $f =< m_{t+r_1} - m_t, ..., m_{t+r_1} - m_t >$, где $r_1, ..., r_n$ – временные сдвиги. Также есть вектор цели Агента, который отражает его текущий приоритет. Здесь приведем поясняющий пример. Пусть вектор измерений состоит из уровня жизни (hp level) и количества убитых врагов (enemy count), значит $m =< hplevel, enemycount >$. Пусть цель игрока была убить как можно больше врагов и тратить как можно меньше жизней, значит вектор цели может иметь вид $g = (-0.5, 1)$. Таким образом, задача сводится к максимизации функции $u(f, g) = g^T f$. То есть при заданной цели получить максимально возможные показатели вектора измерений.

В качестве временных сдвигов для предсказания изменений измерений берется 6 значений: 1, 2, 4, 8, 16 и 32.

Для предсказания значений вектора измерений будем использовать модель-аппроксиматор $F(o_t, a, g, \theta)$, здесь o – наблюдения Агента, a – действие Агента, g – цель Агента и θ – обучаемы параметры модели F.

Таким образом, по обученным параметрам θ модель может выбрать действие, которое дает наилучший прогнозируемый результат, т.е.:

$$a_t = \arg\max_{a \in A} g^T F(o, a, g, \theta)$$

5 Обучение модели

Модель F обучается на опыте, который получает Агент . Агент начинает взаимодействие со средой в течение эпизод, который длится фиксированное время или пока не будет выполнен некоторый критерий останова.

Итак, получаемый опыт можно представить в виде следующего набора данных D для обучения:

$$D =< o_i, a_i, g_i, f_i >_{i=1}^{N}$$

где $< o_i, a_i, g_i >$ – входные параметры модели, а f_i – целевая переменная (целевой вектор изменений измерений). Имея такой набор данных задача сводится к обучению с учителем и мы можем простую регрессионную лосс-функцию:

$$L(\theta) = \sum_{i=1}^{N} ||F(o, a, g, \theta) - f_i||^2$$

Так как происходят новые события, набор данных и модель изменяются самим Агентом. Здесь используется память последних M событий, из которых сэмплируется батч размера N на каждой итерации обучения. Параметры модели изменяются по прошествии каждых событий.

Авторы выделяют два режима обучения:

1. Фиксированная цель – вектор цели фиксирован на все время обучения.
2. Случайные цели – цели для каждого эпизода генерируются случайно.

В обоих случаях модель имеет политику ϵ-greedy, которая подразумевает жадное действие Агента в соответствии с режимом обучения с вероятностью $1 - \epsilon$ или выбор случайного шага с вероятностью ϵ. Сначала $\epsilon = 1$, затем значение снижается в соответствии с некоторым расписанием.

Агент учится и проверяется на эпизодах. Эпизод останавливает по достижении определенного количества шагов игры или нулевым уровнем жизни Агента.

6 Архитектура модели

Опишем архитектуру модели F. Как было отмечено раньше, F зависит от трех составляющих: первой является Сенсорная составляющая(s), вторая - вектор измерений(m), последняя - вектор цели(g). В соответствии с этим на входе модели есть три независимых модуля: модуль для Сенсорного составляющей S(s), модуль для Вектора измерений M(m) и модуль для Вектора цели G(g).

Опишем каждый модуль по отдельности:

1. Модуль сенсорной составляющей S(s). Как было упомянуто выше, Сенсорной компонентной является изображение. Поэтому для обработки этой составляющей используются сверточные слои.
2. Модуль вектора измерений M(m). Для обработки используются полносвязные слои.
3. Модуль вектора цели G(g). Аналогично состоит из полносвязных слоев.

Далее выходы всех модулей объединяются в один вектор (j), который дальше подается в две ветки: в ветку Ожиданий (E(j)) и ветку Действий (A(j)). Ветка Ожиданий предсказывает среднее каждого значения вектора измерений по всем действиям, а ветка Действий учитывает различия в действиях Агента, путем нормализации значений, т.е. вычитается среднее значение действий от всех действий. Таким образом, среднее значение выходов ветки Действий становится равно 0. Однако, вычитание среднего значения компенсируется суммой выходов предсказанного среднего значения из ветки Ожиданий. Поэтому, выход модели состоит из суммы выходов ветки Действий и Ожиданий, которые являются предсказаниями измерений для всех действий Агента. Размерность выхода модели $w \times dim(f)$.

Архитектура модели представлена на Рисунке 1.

7 Описание входных данных

Как было сказано выше, входные данные в сеть состоят из трех частей:

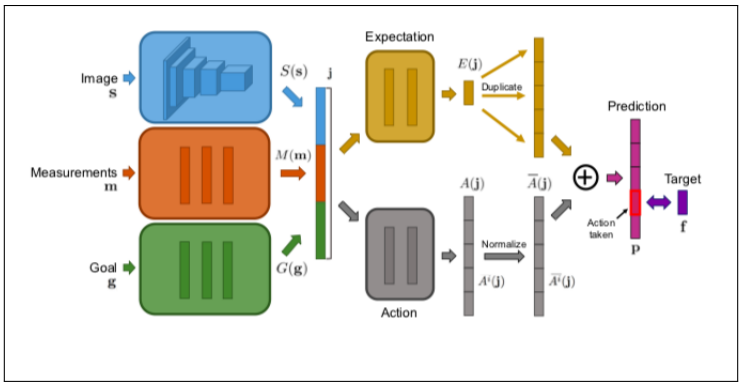

Рис. 1: Архитектура модели

1. Сенсорная составляющая Представляет из себя цветное изображение 3D-пространства размерностью 320x240 пикселей. На Рисунке 2 изображены примеры входных изображений для трех разных сценариев игры.

a b c

Рис. 2: Примеры входного изображения для сценариев: a) Сценарий D1, b) Сценарий D2, c) Сценарий D3

2. Измерения Представляет из себя вектор, содержащий некоторые метрики игры. В нашей задаче набор измерений зависит от сценария игры.
3. Цель Представляет из себя вектор с размерностью (размерность вектора временных сдвигов $\times$ размерность вектора измерений), так как нам в каждый момент времени из рассматриваемых временных сдвигов будет важен приоритет Агента.

7.1 Обработка данных

Обработке подлежит только изображение из “Сенсорного потока”. Сначала изображение переводится в градацию серого, а затемразмер изменяется на 84x84.

8 Взаимодействие с игрой

Игра Doom является интересной и популярной средой применения обучения с подкреплением. Изначально она предназначалась не для обучения моделей, а для реальных играков. Это значит, что среда игры создавалась как можно сложнее для привлечения большего интереса у играков. Все это услажняет обучения агента-бота, который может соревноваться с людьми. Однако, некоторые модели обучения с подкрепелением достойно показывают себя с уже существующими встроенными в игру врагами, как, например подход в рассматриваемой статье.

Взаимодействие с игрой проихсодит при помощи Python-API платформы VIZDoom. Платформа предназначена для исследований и обучения моделей обучения с подкреплением. Она предоставляет множество разнообразных возможностей. Помимо трансляции самого изображения игрового поля на каждом шаге игры, можно использовать "карту глубины"объектов на поле, для обучения моделей выбирать один из режимов игры:синхронный и асинхронный, производить пропуск изображения для снижения нагрузки вычислений и считать дополнительные входные признаки для улучшения представления среды. Также есть возможность модифицировать сами сценарии игры.

В нашей реализации используются не все возможности, предоставлемые платформой. Обучения происходит в асинхронном режиме, то есть игра не ждет действия игрока-агента для продолжения. При помощи реализованных функций в Python-API Агент получает входное изображение, вектора измерений, производит действия и получает награду.

Более подробно возможности платформы и эксперименты с ней описаны в статье "ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning"[2].

9 Результаты

Всего было рассмотрено 3 сценария игры, для каждого из которых была обучена FPN-модель и получены метрический результаты.

9.1 Сценарий D1

Сценарий D1 подразумевает нахождение в квадратной комнате, в которой на некотором расстоянии друг от друга распределены комплекты аптечек. Пример игрового поля изображен на Рисунке 2a. Уровень жизни Агента уменьшается на постоянную величину. Для того, чтобы выжить, Агенту необходимо перемещаться по комнате и собирать аптечки. Всего Агент имеет 3 возможных действия: перемещаться вперед, влево и вправо. Агент может комбинировать эти действия и использовать один из 8 возможных вариантов на каждом шагу.

Графики обучения и тестирования модели представлены в Приложении на рисунке 3.

Прежде всего посмотрим на значения лосс-функции. На графике (a) Рисунка 3 видно, что в начале обучения лосс уменьшается до значений порядка 0.0008, а затем не возрастает.

На графике (b) Рисунка 3 представлена зависимость количества эпизодов, которые были завершены раньше времени по причине смерти Агента, от эпох обучения. Видно, что количество таких эпизодов падает, а значит Агент учится "выживать".

На графике (c) Рисунка 3 предсталена зависимость среднего значения здоровья Агента в течении тестовых эпизодов. Видим, что средний уровень жизниимеет тенденцию к увеличению, что означает что Агент собирает больше аптечек.

На графике (d) Рисунка 3 предсталена зависимость среднего значения награды, получаемой за тестовые эпизоды. Видно, что это значение растет, а значит с каждой эпохой Агент стремится увеличить свою награду, что соответствует подходу обучения.

Результат работы модели этого режима в статье дослигает среднего значения уровня жизни за эпизод 97.7+-0.4 при обучении на 70M шагах. Наша модель показывает

результат в 55.1 на 5K шагах (на больших объемах обучить не узалось из-за нехватки графической памяти).

9.2 Сценарий D2

Сценарий D2 явялется усложненной версией D1. Здесь игрок помещен не в квадратную комнату, а в лабиринт, что является ограничением для перемещений. Еще одним отличием от сценария D1 является наличие сосудов с ядом, при подборе которых уровень жизни уменьшается на некоторое значение. Пример игрового поля для сценария D2 изображен на Рисунке 2b. Уровень жизни игрока попрежнему уменьшается на постоянную величину, а по игровому лабиринту так же, как и в сценарии D1, на некотором расстоянии друг от друга распределены аптечки. В распоряжении игрока имеются 3 действия и 8 возможных комбинаций. Теперь игроку для того, чтобы выжить, необходимо не только поднимать аптечки, но и избегать сосуды с ядом, перемещаясь по более сложной структуре игрового поля.

Графики обучения и тестирования модели представлены в Приложении на рисунке 4.

Посмотрим на значения лосс-функции на графике (a) Рисунка 4. Видим похожую синуацию, как и в режиме D1: начала лосс уменьшается, а потом не возрастает.

Аналогично первому режиму, количество завершенных эпизодов за эпоху падает. Это мы можем набоюдать на графике (b) Рисунка 4.

Средние значение здоровья Агента в течении тестовых эпизодов изображено на графике (c) Рисунка 4.Наблюдаем небольшое уменешение после 3 и 4 эпизода, но потом резкий подъем показателя.

На графике (d) Рисунка 4 предсталена зависимость среднего значения награды, получаемой за тестовые эпизоды. Тут, в отличии от первого режима, явного роста не наблюдается.

Результат работы модели этого режима в статье дослигает среднего значения уровня жизни за эпизод 84.1+-0.6 при обучении на 70M шагах. Наша модель показывает результат в 52.5 на 5K шагах.

9.3 Сценарий D3

Самой сложной модификацией игры является сценарий D3. В нем добавлен элемент стрельбы, то есть у игрока появляется оружие и вооруженные враги. Враги могут передвигаються по полю и атаковать игрока. По полю распределены комплекты аптечек и наборы с боеприпасами, так как по мере стрельбы заканчиваются и боеприпасы. Здесь уровень жизни уменьшается только от аттак врагов. Игрок нахобится в лабиринте, который подобен лабиринту из сценария D2. Пример игрового поля изображен на Рисунке 2c. Игрок есть 8 базовых действий: предвигаться вперед, назад, влево, вправо, стрелять влево, вправо, вперед и бег. При их комбинации получается 256 возможных вариантов действий. Задачей игрока теперь является выжить до конца игрового эпизода, убив при этом максильное количество врагов.

Графики обучения и тестирования модели представлены в Приложении на рисунке 5.

Значения лосс-функции скачкообразно растут в начале обучения, но потос лосс начинает пстепенно и постоянно уменьшаться. Это мы можем увидеть на графике (a) Рисунка 5.

Ступенчато уменьшается количество завершенных эпизодов за эпоху. Это мы видим на графике (b) Рисунка 5.

График (c) Рисунка 5 показывает хороший рост реднего значение здоровья Агента в течении тестовых эпизодов.

Также, на графике (d) Рисунка 5 проиллюстрирован рост показателей среднего значения награды, получаемой за тестовые эпизоды, что говорит об обучении Агента.

Зависимость среднего количества фргов от тестовых эпизодов игры изображена на Рисунке 6 из Приложения. Из него можно сделать вывод, что хоть количество фрагов и невелико, видна тенденция к росту, что означает, что Агент учится убивать монстнов.

Результат работы модели этого режима в статье дослигает 33.5+-0.4 фрагов за эпизод при обучении на 70M шагах. Наша модель показывает результат в 1.4 на 5K шагах. Это обуславливается тем, что в связи с большим количеством вариантов действий, Агент не успевает качественно обучиться на малом количестве итераций, то есть не успевает опробовать каждое действие и понять какое и в каком случае павышают награду.

Заключение

Весь разработаный код находится в репозитории vizdoom-project. Видео с резутьтатами работы модели можно посмотреть здесь.

Исходя из видео, можно сделать вывод, что Агент еще плохо распознает предметы по краям экрана. Заметны подергивания в сторону аптечек, но они прекращаются, как только аптечки оказывтся на краю изображения. В последнем режиме, Агент делает осторожные движения, но еще плохо обучен на поражение цели.

Источники

[1] Alexey Dosovitskiy, Vladlen Koltun. "Learning to Act by Predicting the Future" ICLR 2017.

[2] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, Wojciech Jas kowski."ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learninge"

Приложение

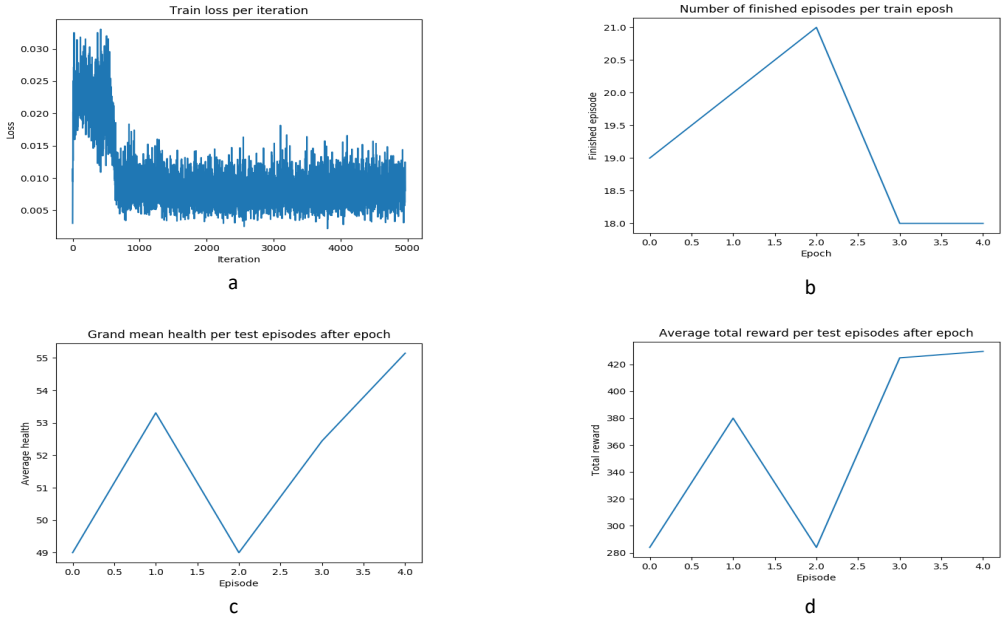

Рис. 3: Графики обучения и теста для Сценария D1: a) Зависимость лосса-фунции от итерации обучения, b) Зависимость числа досрочно завершенных эпизодов от эпохи обучения, c) Зависимость среднего значения уровня жизни от тестовой эпохи, d) Зависимость среднего значения общей награды от тестовой эпохи

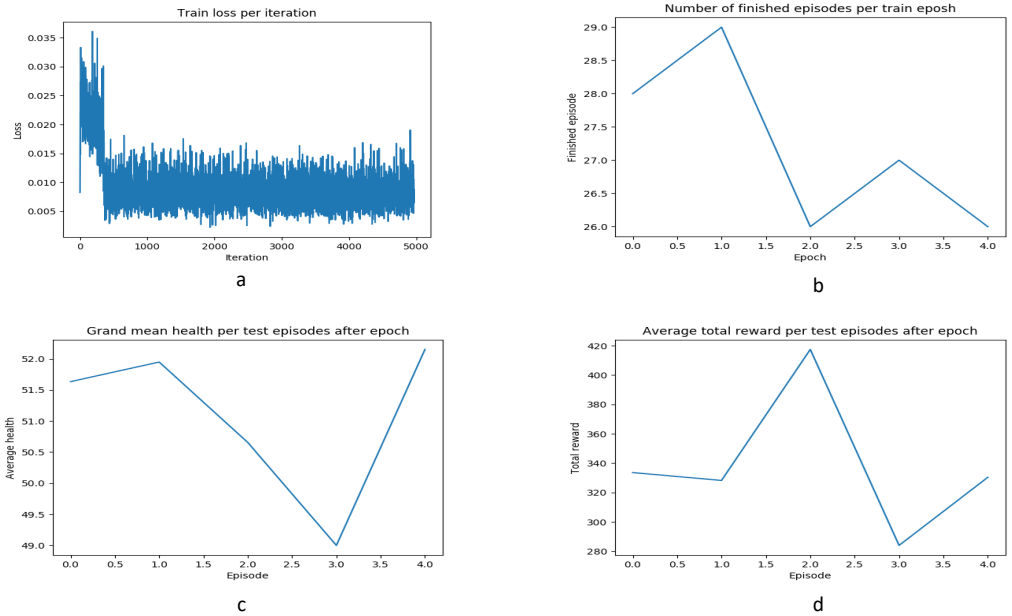

Рис. 4: Графики обучения и теста для Сценария D2: a) Зависимость лосса-фунции от итерации обучения, b) Зависимость числа досрочно завершенных эпизодов от эпохи обучения, c) Зависимость среднего значения уровня жизни от тестовой эпохи, d) Зависимость среднего значения общей награды от тестовой эпохи

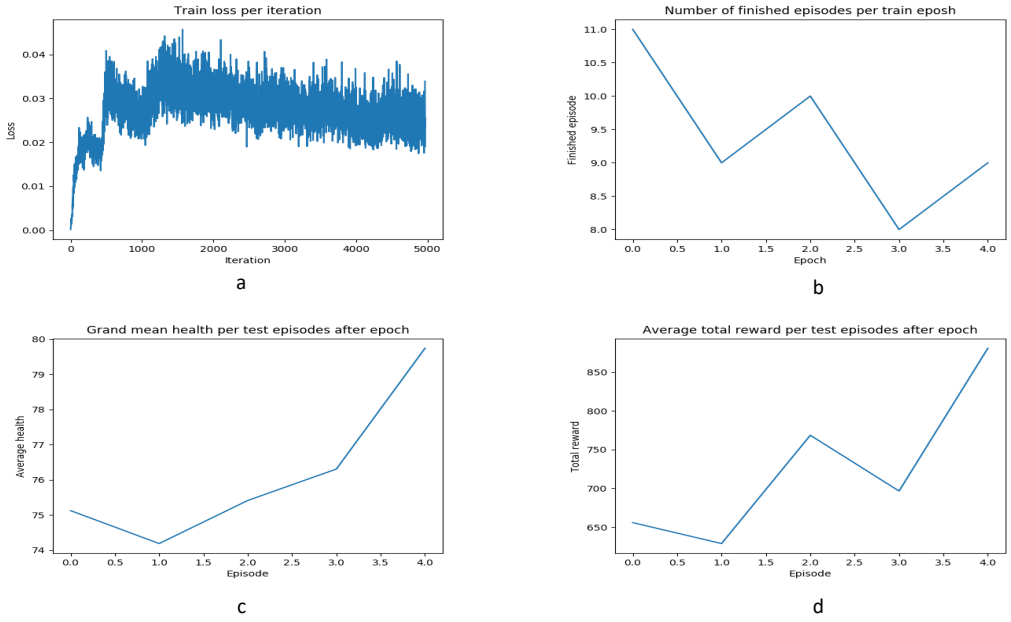

Рис. 5: Графики обучения и теста для Сценария D3: a) Зависимость лосса-фунции от итерации обучения, b) Зависимость числа досрочно завершенных эпизодов от эпохи обучения, c) Зависимость среднего значения уровня жизни от тестовой эпохи, d) Зависимость среднего значения общей награды от тестовой эпохи

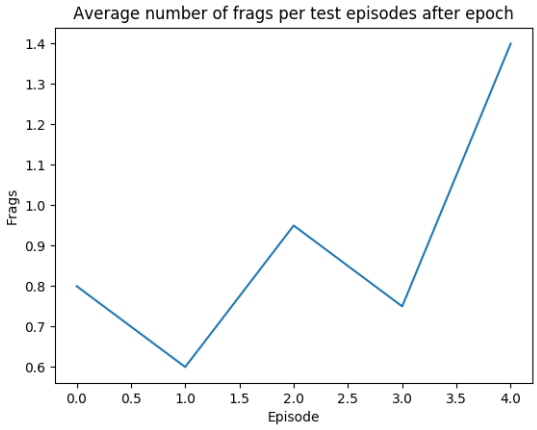

Рис. 6: График зависимости среднего количества убитых врагов от тестовой эпохи