**Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.04.04 - Программная инженерия | |
| **Программа** | Разработка распределенных программных систем | |
| **Факультет** | КТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Кринкин К.В. |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

магистра

Тема: **РЕАЛИЗАЦИЯ И ИССЛЕДОВАНИЕ АЛГОРИТМА ГЕНЕРАЦИИ ТРАЕКТОРИЙ В ДИНАМИЧЕСКОЙ СРЕДЕ НА ОСНОВЕ OCTNet**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Губа Д. А. |
|  |  | *подпись* |  |  |
| Руководитель | к.т.н. |  |  | Заславский М.М. |
| *подпись* |  |  |
| Консультанты | ассистент |  |  | Жангиров Т.Р |
|  |  | *подпись* |  |  |
|  | к.э.н., доцент |  |  | Звонцов А.В. |
|  |  | *подпись* |  |  |

Санкт-Петербург

2021**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | Утверждаю | | | | | | | | |
|  | | | | Зав. кафедрой МО ЭВМ | | | | | | | | |
|  | | | | \_\_\_\_\_\_\_\_\_\_\_\_ Кринкин К.В. | | | | | | | | |
|  | | | | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. | | | | | | | | |
| Студент | Губа Д.А. | | | | | | |  | Группа | 5303 |
| Тема работы: Реализация и исследование алгоритма генерации траекторий в динамической среде на основе OCTNet | | | | | | | | | | | |
| Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), кафедра МО ЭВМ | | | | | | | | | | | |
| Исходные данные: разработка и реализации алгоритма генерации траекторий в динамической среде, на основе модели машинного обучения OCTNet. | | | | | | | | | | | |
| Содержание ВКР: «Введение», «Анализ существующих алгоритмов генерации траекторий и обоснование актуальности работы», «Подготовка данных для обучения алгоритма», «Модель OCTNet для статической среды», «Модель OCTNet для динамической среды», «Сравнение характеристик работы алгоритмов», «Оценка и защита результатов интеллектуальной деятельности», «Заключение». | | | | | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, презентация. | | | | | | | | | | | |
| Дополнительный раздел: Оценка и защита результатов интеллектуальной деятельности. | | | | | | | | | | | |
| Дата выдачи задания | | | | | | Дата представления ВКР к защите | | | | | |
| «08» февраля 2021 г. | | | | | | «01» июня 2021 г. | | | | | |
| Студент | |  |  | |  | | Губа Д. А. | | | | |
| Руководитель | | к.т.н. | | |  | | Заславский М.М. | | | | |
|  | |  | | |  | |  | | | | |
| Консультант | | ассистент | | |  | | Жангиров Т.Р | | | | |
|  | |  | | |  | |  | | | | |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Кринкин К.В. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Губа Д.А. |  | Группа | 5303 |
| Тема работы: Реализация и исследование алгоритма генерации траекторий в динамической среде на основе OCTNet | | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме генерации траекторий | 08.02 – 20.02 |
| 2 | Обзор и анализ характеристик существующих методов и алгоритмов | 21.02 – 05.03 |
| 3 | Проектирование и разработка алгоритма для проведения экспериментов | 05.03 – 27.04 |
| 4 | Проведение экспериментов | 28.04 – 29.04 |
| 5 | Оформление пояснительной записки | 30.04 – 15.05 |
| 6 | Оформление демонстрационного материала | 16.05 – 17.05 |
| 7 | Предзащита | 18.05 – 26.05 |
| 8 | Защита | 01.06.2021 |

|  |  |  |  |
| --- | --- | --- | --- |
| Студент |  |  | Губа Д. А. |
| Руководитель | к.т.н. |  | Заславский М.М. |

|  |  |  |  |
| --- | --- | --- | --- |
| Консультант | ассистент |  | Жангиров Т.Р |
|  |  |  |  |

**РЕФЕРАТ**

**Пояснительная записка:** 89 стр., 29 рис., 7 табл., 27 ист., 4 прил.

**Ключевые слова:** OCTNet, траектория, нейросеть, Tensorflow, модель, регрессия, обучение, генерация.

**Цель работы:** разработать и исследовать алгоритм машинного обучения для генерации траекторий на основе архитектуры сети OCTNet. Входными данными модели являются карты, закодированные в виде двумерного массива. На выходе модель выдает параметры для построения вероятностного распределения, из которого, при помощи семплирования, возможно сгенерировать необходимое количество траекторий.

Для разработки алгоритма был проведен анализ существующих решений в области генерации траекторий. Изучена архитектура OCTNet. Разработана статическая модель для проверки возможности генерации траекторий с помощью обученной модели. На основе статической модели реализована динамическая модель, для генерации маршрута по динамическим картам. Проведено сравнение реализованного алгоритма с аналогами.

**ABSTRACT**

In this paper, the principles of trajectory generation known at the moment are investigated and described. Generated dynamic map data to train the dynamic model. Algorithms based on the OCTNet machine learning architecture are implemented to generate trajectories in static and dynamic environments. Provided a mathematical description of ready-made machine learning models, on the basis of which the algorithm works. The results of the implemented algorithms are demonstrated. Comparison of the results of the work of similar approaches to the generation of trajectories with a work algorithm based on OCTNet is carried out.

**Содержание**

[ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 9](#_Toc72326489)

[ВВЕДЕНИЕ 11](#_Toc72326490)

[Актуальность 11](#_Toc72326491)

[Цель 12](#_Toc72326492)

[Задачи 12](#_Toc72326493)

[1. АНАЛИЗ СУЩЕСТВУЮЩИХ АЛГОРИТМОВ ГЕНЕРАЦИИ ТРАЕКТОРИЙ И ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАБОТЫ 13](#_Toc72326494)

[1.1. Критерии сравнения 14](#_Toc72326495)

[1.2. Существующие подходы к построению траекторий 16](#_Toc72326496)

[1.3. Сравнение аналогов 20](#_Toc72326497)

[2. ПОДГОТОВКА ДАННЫХ ДЛЯ ОБУЧЕНИЯ АЛГОРИТМА 23](#_Toc72326498)

[2.1. OCC-Traj120 23](#_Toc72326499)

[2.2. Увеличение размера набора данных 25](#_Toc72326500)

[3. МОДЕЛЬ OCTNet ДЛЯ СТАТИЧЕСКОЙ СРЕДЫ 26](#_Toc72326501)

[3.1. Вектор-признаков 26](#_Toc72326502)

[3.2. Кодирование траекторий 30](#_Toc72326503)

[3.3. Архитектура 36](#_Toc72326504)

[3.4. Обучение 40](#_Toc72326505)

[3.5. Генерация траекторий 43](#_Toc72326506)

[4. МОДЕЛЬ OCTNet ДЛЯ ДИНАМИЧЕСКОЙ СРЕДЫ 47](#_Toc72326507)

[4.1. Данные 47](#_Toc72326508)

[4.2. LSTM 50](#_Toc72326509)

[4.3. Архитектура 52](#_Toc72326510)

[4.4. Обучение MDN сети 53](#_Toc72326511)

[4.5. Предсказание следующего кадра 56](#_Toc72326512)

[4.6. Генерация траекторий 58](#_Toc72326513)

[5. СРАВНЕНИЕ ХАРАКТЕРИСТИК РАБОТЫ АЛГОРИТМОВ 62](#_Toc72326514)

[5.1. Результаты сравнения 63](#_Toc72326515)

[6. ОЦЕНКА И ЗАЩИТА РЕЗУЛЬТАТОВ ИНТЕЛЛЕКТУАЛЬНОЙ ДЕЯТЕЛЬНОСТИ 65](#_Toc72326516)

[6.1. Понятие интеллектуальной собственности 65](#_Toc72326517)

[6.2. Результаты интеллектуальной деятельности 65](#_Toc72326518)

[6.3. Характеристика видов интеллектуальной деятельности 66](#_Toc72326519)

[6.4. Объект интеллектуальной деятельности работы 71](#_Toc72326520)

[6.5. Нормативно-правовое акты, регулирующие вопросы защиты результатов интеллектуальной деятельности 71](#_Toc72326521)

[6.6. Описание процедуры подачи заявки на регистрацию объектов интеллектуальной деятельности 72](#_Toc72326522)

[6.7. Анализ возможного использования объекта интеллектуальной деятельности 74](#_Toc72326523)

[6.8. Оценка рыночной стоимости объекта интеллектуальной деятельности 75](#_Toc72326524)

[6.9. Выводы 78](#_Toc72326525)

[ЗАКЛЮЧЕНИЕ 79](#_Toc72326526)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 82](#_Toc72326527)

[ПРИЛОЖЕНИЕ А 85](#_Toc72326528)

[ПРИЛОЖЕНИЕ Б 86](#_Toc72326529)

[ПРИЛОЖЕНИЕ В 87](#_Toc72326530)

[ПРИЛОЖЕНИЕ Г 88](#_Toc72326531)

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

**OCTNet** – Occupancy Conditional Trajectory Network

**Модель** – исполняемый код, который обучен распознаванию определенных типов закономерностей. Обучение модели выполняется по некоторому набору данных с использованием алгоритма, который позволяет анализировать предоставленные данные и запоминать полученные результаты. Завершив обучение модели, вы сможете применить ее для принятия решений и выполнения прогнозов по данным, которые ранее не встречались.

**Нейронная сеть** – одно из направлений искусственного интеллекта, цель которого смоделировать аналитические механизмы, осуществляемые человеческим мозгом. Задачи, которые решает типичная нейросеть – классификация, предсказание и распознавание. Нейросети способны самостоятельно обучаться и развиваться, строя свой опыт на совершенных ошибках.

**Функция потерь** – функция, которая в теории статистических решений характеризует потери при неправильном принятии решений на основе наблюдаемых данных.

**Эпоха** – этап обучения, на котором весь набор данных для обучения прошел через сеть.

**Batch** – партия данных по которой проходит одна итерация обучения.

**TensorFlow** – библиотека с открытым исходным кодом, созданная Google, которая используется при разработке систем, использующих технологии машинного обучения.

**KRR** (Kernel Rigid Regression) – непараметрический статистический метод, позволяющий оценить условное математическое ожидание случайной величины. Его смысл заключается в поиске нелинейного отношения между парой случайных величин X и Y.

– в работе, при записи логарифма в следующем виде, подразумевается натуральный логарифм.

**Кроссвалидация** – метод оценки аналитической модели и её поведения на независимых данных

**Гиперпараметр** – параметры алгоритмов, значения которых устанавливаются перед запуском процесса обучения.

**Backpropagation** – метод вычисления градиента, который используется при обновлении весов многослойной нейронной сети. Основная идея этого метода состоит в распространении сигналов ошибки от выходов сети к её входам, в направлении обратном прямому распространению сигналов в обычном режиме работы.

**Функция активации** – функция, вычисляющая выходной сигнал нейрона и предающая ему нелинейные свойства.

**Нормализация** – это процесс, при котором все входные данные проходят процесс приведения к определённому интервалу, в рамках которого центрируется математическое ожидание и уменьшается дисперсия.

# ВВЕДЕНИЕ

Роботизированные агенты: такие как беспилотные автомобили, беспилотная строительная техника и роботы-помощники, сегодня активно развиваются и находят свое применение как в жизни обычных людей, так и на разнородных промышленных и военных предприятиях. Такие агенты способны работать в условиях, которые являются небезопасными для человека. Также, роботы способны выполнять свои задачи круглосуточно, без выходных и больничных. По этим причинам автоматизированные агенты становятся востребованы все в больших областях жизни человека. Рост популярности и количества роботов в жизни человека приводит к ужесточению требований безопасности систем, которые управляют данными машинами. Такие системы должны уметь прогнозировать движение робота на несколько шагов вперед и иметь запасные варианты движения в случае возникновения непредвиденной ситуации. Также, одним из требований к таким системам является их интегрированность в жизнь человека, так как неестественное перемещение роботов в пространстве может беспокоить окружающих и также приводит к аварийным последствиям.

## Актуальность

Для успешной и качественной работы роботизированных агентов управляющим системам необходимо принимать решения со скоростью, превышающей скорость работы человеческого мозга. Исходя из этого, алгоритмы, использующиеся для управления этими устройствами, должны работать очень быстро. Одним из критических алгоритмов для подобных устройств является алгоритм генерации траекторий. Для устройств, движение которых будет происходить среди обычных людей, правильная траектория движения критически важна. Выбор управляющим устройством аварийной или неестественной траектории может привести к аварийным последствиям. Таким образом важнейшими критериями построения траекторий в среде, где происходит движение других объектов, являются скорость работы и естественность сгенерированных траекторий.

## Цель

Поскольку на данный момент актуальна проблема быстрой и универсальной генерации естественных траекторий для среды с движущимися объектами, целью работы являлась реализация алгоритма, использующего модель машинного обучения, который позволяет генерировать возможные траектории маршрутов быстро и универсально, работает в динамической среде и генерирует маршруты, похожие на реальные.

## Задачи

Для достижения заданной цели были решены следующие задачи:

* Проведен анализ существующих аналогов для генерации траекторий;
* Подготовлены данные статических сред;
* Разработана модель для алгоритма, работающего в статических средах;
* Подготовлены данные динамических сред;
* Разработана модель для алгоритма, работающего в динамических средах;
* Проведено сравнение производительности готового решения с аналогами.

# АНАЛИЗ СУЩЕСТВУЮЩИХ АЛГОРИТМОВ ГЕНЕРАЦИИ ТРАЕКТОРИЙ И ОБОСНОВАНИЕ АКТУАЛЬНОСТИ РАБОТЫ

Для решения проблемы генерации траектории на текущий момент используются два семейства алгоритмов: алгоритмы на графах и алгоритмы машинного обучения.

Классические алгоритмы, такие как алгоритм Дейкстры и A\* можно использовать для генерации траекторий. В результате мы будем получать траектории, которые не обязательно будут похожи на реальные траектории в данной среде. Также, для использования классических алгоритмов нам необходимо уметь представлять любую карту в виде графа, что делает использование классических алгоритмов крайне ограниченным. Для динамических систем, необходимо постоянно изменять граф, что является дорогостоящей операцией.

Для решения задачи генерации траекторий используются также различные модели машинного обучения. Одни из наиболее популярных моделей, которые могут быть использованы для генерации траекторий - GAN, CVAE и KTM. В отличии от алгоритмов на графах, алгоритмы, с использованием обученных моделей могут обобщать опыт, полученный на обучающем наборе, на новые данные. Сложность генерации траекторий по обученной модели равна сложности перемножения вектора на матрицу, что является допустимым. В свою очередь приведенные в пример алгоритмы машинного обучения не имеют возможности получать информацию из динамической обстановки среды. Из-за отсутствия знаний о характере движения внутри среды и о ее динамике, существующие решения в сфере машинного обучения обладают достаточной точностью, или не способны строить подобные траектории. Проведем более детальное сравнение аналогов с разработанным решением.

## Критерии сравнения

Для сравнения качеств существующих подходов с реализованным необходимо выбрать критерии сравнения. Для алгоритма генерации траекторий важны следующие свойства:

* Универсальность;
* Динамичность;
* Схожесть;
* Время работы;
* Параллельность;
* Расходуемая память;
* Стабильность

Рассмотрим выбранные критерии более подробно.

### Универсальность

Данный оценочный критерий является бинарным. Решение может либо обладать данным свойством, либо нет. В рамках обзора считается, что решение универсально, если для любого нового входного значения алгоритм способен сгенерировать нужное количество траекторий без изменения реализации.

### Динамичность

В рамках обзора, под динамичностью понимается способность алгоритма получать на вход динамическое представление входных данных, по которому выдавать динамическое представление траекторий с заданной точностью.

### Схожесть

Одним из важнейших критериев для сгенерированных траекторий является схожесть с настоящими траекториями в данной среде. Данный критерий важен, поскольку одно из возможных применений алгоритма – модуль генерации маршрутов для роботизированных агентов. Характер движения в конкретной среде зависит от самой среды. Например, траектория движения автомобиля отличается от траектории движения человека и траектории движения морских судов, поэтому крайне важно, чтобы алгоритм имел возможность строить маршруты, которые свойственны траекториям конкретной среды.

### Масштабируемость

Для алгоритма генерации траекторий важным требованием является способность подстраиваться под размер входных данных. Например, алгоритм должен генерировать траектории как внутри помещения, так и в рамках всей планеты. Размер входных данных не должен ограничивать возможности готового решения.

### Время работы

Для роботизированных агентов, в работе которых планируется применять данный алгоритм, важно иметь адекватное время отклика. В данной работе за адекватное время принимается полиномиальное время работы, поскольку, в случае с генерацией траекторий в реальной жизни, экспоненциальное время может приводить к критическим последствиям.

### Параллельность

Реализация алгоритма должна быть устроена таким образом, чтобы время работы можно было сократить за счет распараллеливания. Такое свойство позволяет переносить работу на устройства с большим числом ядер, что приводит к многократному увеличению производительности.

### Расходуемая память

Память - ограниченный ресурс, поэтому важно, чтобы алгоритм не сильно возрастал по расходу памяти в зависимости от входных данных. Допустимым увеличением расхода памяти в зависимости от размера входных данных является линейный рост.

### Стабильность

Крайне важно, чтобы готовое решение выдавало как можно меньше недопустимых траекторий. Такое требование позволит сократить время на валидацию готовых маршрутов, что так же сказывается на отклике потенциального автоматизированного агента.

Основываясь на выбранных критериях, перейдём к сравнению аналогов.

## Существующие подходы к построению траекторий

Для обоснования актуальности работы, необходимо выделить основные существующие подходы, которые используются для генерации траекторий.

### A\*

А\* - алгоритм поиска кратчайшего пути в графе. Если представить все пространство как граф, каждый узел которого соединен со всеми своими смежными соседями, то можно использовать его для поиска маршрута в этом графе. Для генерации различных траекторий на каждый запрос можно добавить «шум» во входные данные. Например, можно случайным образом убирать ребра между узлами, или ввести веса и каждый раз задавать их случайным образом.

Плюсом такого подхода является простота реализации и широкая известность алгоритма. Также, для сгенерированного графа, при условии выбора подходящей эвристики, достигается полиномиальная скорость работы, что является допустимым. Алгоритм полностью стабилен и из всегда строит валидный маршрут.

Существующие минусы данного подхода делают классические алгоритмы неподходящими для решения данной задачи. Главный минус заключается в том, что существует необходимость модифицировать граф для каждой новой среды, а для динамической среды необходимо на каждое изменения про изводить обход графа с изменением ребер. Также, не соблюдается требование схожести сгенерированного и естественного маршрутов для конкретной среды.

### GAN

GAN - генеративная модель, которая может быть приспособлена к построению траекторий [1]. Сети GAN состоят из двух сетей: порождающей и различающей. Задача первой сети - сгенерировать данные, которые подаются на вход второй сети. Вторая сеть должна различать сгенерированные данные от истинных. GAN считается успешно обученным, когда распознающая сеть не в состоянии отличить истинные данные от сгенерированных первой сетью. Затем вторая сеть используется для генерации новых данных.

Плюсом подхода является его универсальность, масштабируемость, время работы и память. После обучения на тренировочных данных, модель способна генерировать траектории для новых данных, которые модель не видела до этого момента. Решение масштабируется на входные данные любого размера, при этом память при использовании растет линейно с размером входных данных. Время работы алгоритма с моделью GAN в худшем случае:

|  |  |  |
| --- | --- | --- |
|  |  | (1.1) |

где: – длина вектора признаков;

– количество нейронов в слое сети;

– количество слоев в сети.

Также плюсом является то, что любой алгоритм машинного обучения может быть распараллелен, поскольку предсказание строится при помощи перемножения матриц.

Существенным минусом использования модели GAN в алгоритме генерации траекторий является отсутствие схожести итоговых траекторий с действительными, а также частая генерация недопустимых траекторий. Также сети GAN крайне долго обучаются, что является большим минусом по той причине, что модели машинного обучения необходимо переобучать при каждом изменении требований. Большим минусом является и стабильность алгоритма – всего 10 из 100 построенных траекторий являются допустимыми.

### CVAE

CVAE сеть относятся к автоэнкодерам [2]. Такие сети ищут закономерности во входных данных и на их основе генерируют выходные. Данная архитектура сетей часто показывает хорошие результаты в алгоритмах по генерации.

Плюсом такого подхода является универсальность, поскольку автоэнкодерам не нужно понимание природы генерируемых данных, вся генерация основана на закономерностях во входных данных. Расход памяти и скорость работы (1.1) аналогичны GAN.

Минусом использования данной архитектуры сетей является то, что модель не учитывает природу генерируемых данных, а просто обобщает входные по некоторым закономерностям. Исходя из этого модель генерирует слишком много недопустимых траекторий, что исключает возможность ее применения в алгоритмах для движения автоматизированных агентов. Также данные сети крайне долго обучаются и требуют огромного количества данных для обучения. Стабильность данной модели такая же, как у GAN.

### KTM

KTM - модель машинного обучения основанная на MDN сетях [3]. MDNсеть - сеть смешанной плотности, которая способна предсказывать параметры некоторого вероятностного распределения [4]. Такой подход позволяет предсказывать не сами траектории, а параметры, на основании которых строится распределение, из которого методом семплирования получаются веса. Полученные веса передаются в модель линейной регрессии, которая может восстановить координаты траектории в любой момент времени.

Реализация траекторий в виде непрерывной функции позволяет строить маршруты различной длины, что делает данный подход универсальным и масштабируемым. Модель обучает данные для некоторого вероятностного распределения на основе известных траекторий, что позволяет генерировать траектории из того же вероятностного пространства. За счет данного подхода сгенерированные траектории максимально близки к реальным. Скорость работы KTN отличается от скорости GAN и CVAE

|  |  |  |
| --- | --- | --- |
|  |  | (1.2) |

где: – длина вектора признаков;

– длина генерируемой траектории;

– количество весов для регрессии;

– количество нейронов в слое сети;

– количество слоев в сети.

Полученная скорость работы (1.2) удовлетворяет требованиям. Большим плюсом является стабильность данной модели – на каждые 100 сгенерированных траекторий около 90 являются допустимыми

Данная архитектура сети имеет большой минус – неспособность работать в неизвестных средах. Дело в том, что вектор признаков, который подается на вход модели состоит из значений функции расстояния Хаусдорфа между траекториями – это означает, что обученная модель отлично справляется только в средах, где ранее наблюдалось движение. Данное ограничение делает невозможным применение KTM сетей в динамических средах. Рост расхода памяти так же не удовлетворяет поставленным ограничениям, поскольку растет квадратично.

### OCTNet

OCTNet – модель, которая заимствует идеи заложенные в KTM сети, но решает проблемы неконтролируемого роста памяти, необходимости иметь большой набор обучающих данных и неспособности работать с новыми данными. В качестве свойств модель строит вектор из расстояния Хаусдорфа между случайно выбранными картами из обучающей выборки.

Количество карт намного меньше, чем количество траекторий на карте, что существенно экономит память. Также в данной модели берутся только несколько случайных карт, что позволяет еще сильнее экономить память. Модель, как и KTM позволяет обучать параметры некоторого вероятностного распределения, что позволяет генерировать близкие к реальным траектории любой длины. Скорость работы также аналогична KTM, а потребление памяти растет линейно. Данная архитектура способна строить траектории в среде, в которой ранее не наблюдалось движения, поскольку обучена не на траекториях, а на схожести карт. Таким образом модель видит закономерности не в траекториях, а в картах, что делает ее применение возможным в динамических средах. Стабильность модели близка к KTM – на 100 сгенерированных траекторий 80 являются допустимыми.

## Сравнение аналогов

Выбранные решения для генерации траекторий в динамических средах были рассмотрены и сравнены между собой. Универсальность, динамичность, схожесть и параллельность являются бинарными признаками, означающими наличие или отсутствие данного свойства. Время работы и память описаны в нотации О большое. Стабильность рассчитана по формуле

|  |  |  |
| --- | --- | --- |
|  | , | (1.3) |

где: – количество сгенерированных допустимых траекторий;

– количество всех сгенерированных траекторий.

Значения стабильности определялись по формуле (1.3) по генерации 100 траекторий на 10 различных картах. В таблице табл. 1.1 представлено сравнение по указанным ранее критериям.

Таблица 1.1 - Сравнение алгоритмов-аналогов

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Универсальность** | **Динамичность** | **Схожесть** | **Время работы** | **Параллельность** | **Память** | **Стабильность** |
| A\* | + | - | - |  | - |  | 1.0 |
| GAN | + | + | + |  | + |  | 0.14 |
| CVAE | + | + | + |  | + |  | 0.11 |
| KTM | - | - | + |  | + |  | 0.78 |
| OCTNet | + | + | + |  | + |  | 0.73 |

Из данных в таблице табл. 1 видно, что существующие решения не решают описанную ранее проблему или решают ее не в полной мере. В отличии от аналогов, разработанное решение способно строить траектории в динамической среде, которые соответствуют реальным. Таким образом удовлетворяется критерий схожести. Поскольку обучение производится по картам фиксированного размера, решение отвечает требованиям универсальности, динамичности. Так как в основе алгоритма лежит использование модели машинного обучения, реализованное решения удовлетворяет требованиям по скорости и памяти. Также процесс построения траекторий может быть легко распараллелен так как в его основе лежит алгоритм умножения матриц. Исходя из этого, можно подтвердить актуальность данной работы.

# ПОДГОТОВКА ДАННЫХ ДЛЯ ОБУЧЕНИЯ АЛГОРИТМА

Модель OCTNet строится вокруг идеи сети MDN, которая является сетью обучения с учителем. Для обучения с учителем необходим обучающий набор данных большого размера. Данные должны содержать входные данные модели и соответствующий выход, на предсказание которого должна настраиваться модель.

Для обучения сети нужны карты различной местности, которые будут преобразованы в вектор признаков. Также, набор данных должен содержать соответствующие карте траектории, которые будут преобразованы в векторы весов для радиальных базисных функций.

## OCC-Traj120

На данный момент не существует обширного набора данных, содержащих необходимую информацию, поэтому лучшим вариантом для проведения обучения является набор данных Occ-Traj120 [5]. Данный набор содержит 400 карт комнат различной конфигурации, с построенными на них траекториями движения людей. Каждая карта содержит от 200 до 300 траекторий различной длины и формы. Карты закодированы в виде двумерного массива и выглядят следующим образом:

Для экономии пространства был продемонстрирован фрагмент карты, представляющий четверть оригинального размера. Нули на карте означают свободное место, а единицы представляют стены.

Траектории в наборе закодированы парами списков координат, где первый список отвечает координате , а второй . Длины траекторий лежат в пределах от 4 до 58, что видно на рисунке рис. 2.1:

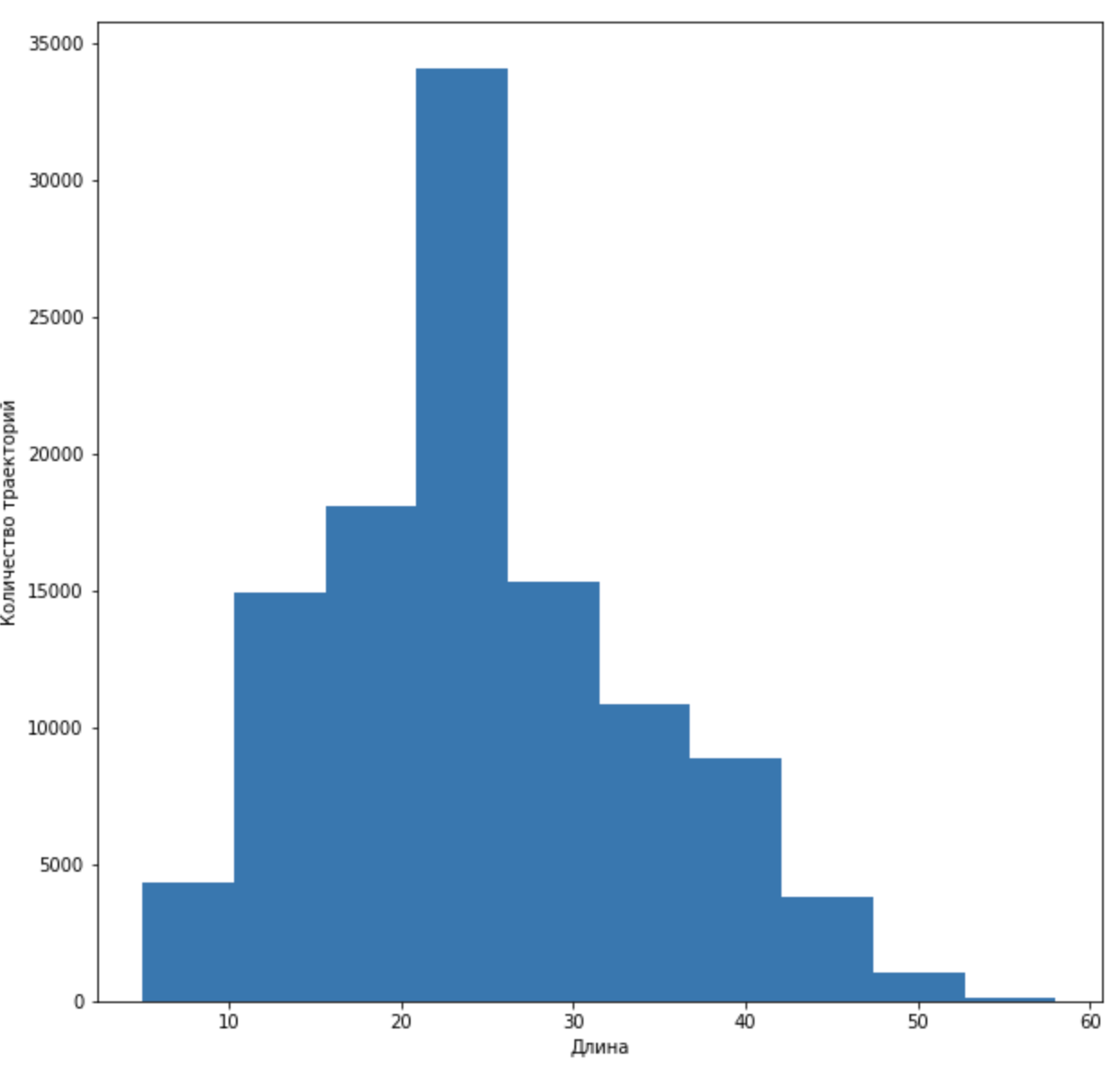


Рисунок 2.1 – Распределение длин траекторий

Все траектории должны лежать в пределах карты и не выходить за ее пределы, также на данные наложено дополнительное ограничение в виде минимального расстояния от стен, поскольку люди не ходят, прижавшись к стенам. Карта с траекториями представлена на рисунке рис. 2.2:

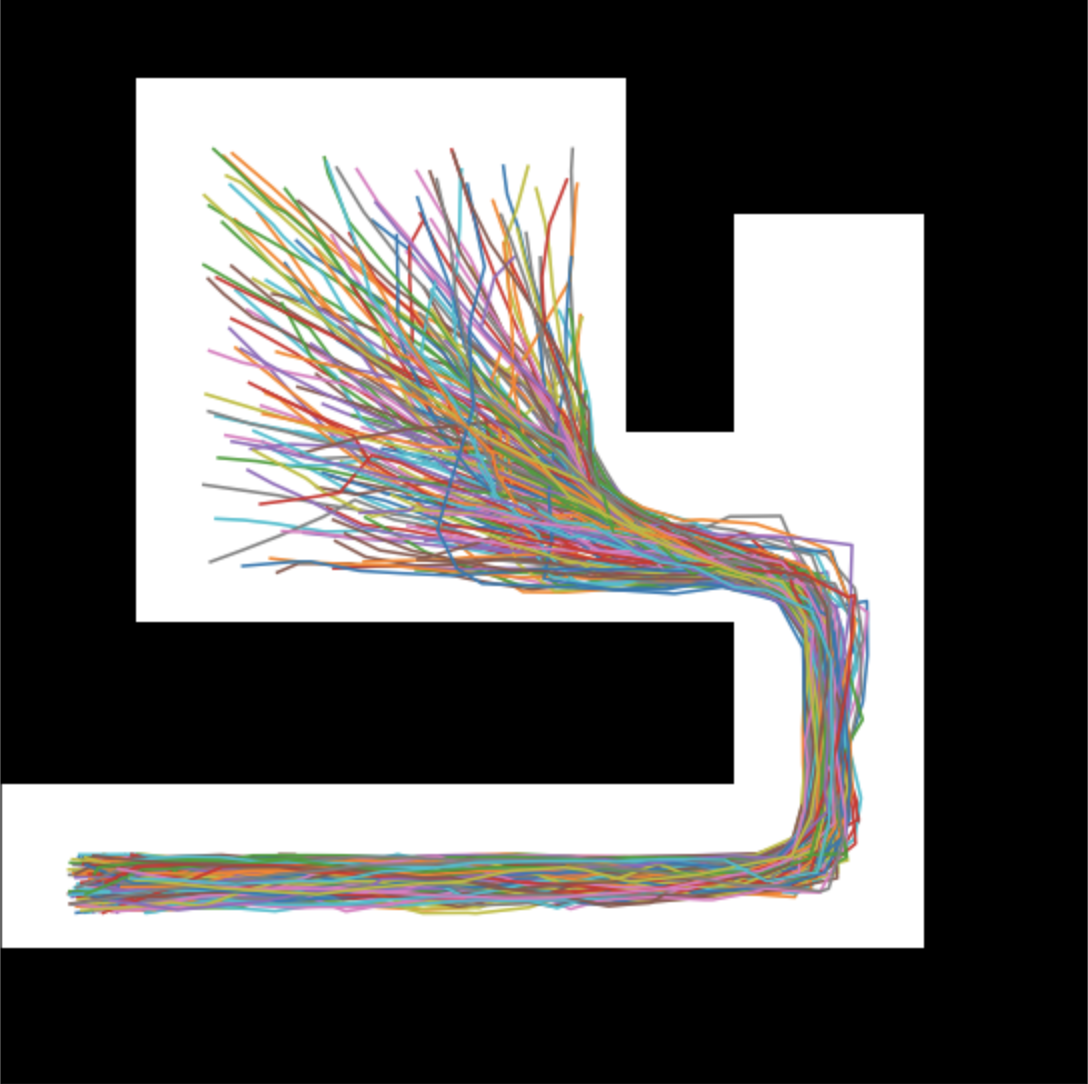


Рисунок 2.2 – Построение карты с траекториями

## Увеличение размера набора данных

Хотя Occ-Traj120 является отличным набором данных с реальными траекториями движения, он содержит слишком мало карт и траекторий для обучения. Однако, можно увеличить количество данных в 4 раза при помощи поворота карт и траекторий. Результат преобразования продемонстрирован на рисунке рис. 2.3.



Рисунок 2.3 – Трансформация карт

После обучения на подобных данных, алгоритм умеет генерировать траектории любой длины в помещениях с прямыми стенами. Для обучения алгоритма к другим типам местности, требуется предоставить набор данных различной конфигурации, чтобы модель могла обобщит их. Получение реальных данных различной конфигурации выходит за пределы данной работы, поэтому демонстрация возможностей алгоритма будет производится на картах подобной конфигурации.

# МОДЕЛЬ OCTNet ДЛЯ СТАТИЧЕСКОЙ СРЕДЫ

OCTNet берет многие идеи из архитектуры KTM. Например, кодирование траекторий. Абсолютно идентичное в обеих архитектурах. Кодирование входных данных, также, происходит подобным образом. Сеть в обоих случаях предсказывает параметры некоторого распределение, по которому можно воссоздавать веса для генерации траекторий. Рассмотрим OCTNet подробнее.

## Вектор-признаков

Как было описано ранее, OCTNet ищет закономерности между картами и траекториями, построенными на них. Карты в обучающей выборке приставлены в виде нулей и единиц, что не подходит для входных данных модели. Поскольку обучение модели – вычислительно сложная задача, обучение зачастую выполняется не на процессоре, а на видеокарте, возможности распараллеливания которой значительно выше. Такой способ обучения накладывает дополнительные ограничения на входные данные. Входные данные должны быть тензором фиксированной размерности. Таким образом модель не сможет обучаться на картах разной формы. Например, если одна карта имеет размеры а другая карта имеет размеры становится невозможным подать их на вход модели. Более того, из-за бинарности данных в исходных картах, обучение сети может быть бесконечно долгим и требовать огромных объемов данных. Для эффективного обучения требуется приводить данные в равномерную и репрезентативную форму. В архитектуре OCTNet предлагается представлять входные данные как вектор схожести между картами. Схожесть между картами рассчитывается при помощи расстояния между множествами. При помощи кроссвалидации выявлено, что лучшие результаты показывает сеть, входные данные которой превращены в вектор схожести при помощи расстояния Хаусдорфа [6]

|  |  |  |
| --- | --- | --- |
|  |  | (3.1) |

где: – расстояние Хаусдорфа;

– первое множество точек;

второе множество точек.

В подобном виде (3.1) расстояние Хаусдорфа не является симметричным, что является проблемой. Дело в том, что расстояние Хаусдорфа есть мера схожести между двумя множествами. В случае, когда она не симметрична получается ситуация, когда первая карта сильно похожа на вторую, но вторая слабо похожа на первую. Такой вектор свойств будет вводить сеть в заблуждение и негативно влиять на предсказания сети, делая ее практически необучаемой. Для решения данной проблемы берется средняя величина суммы прямого и обратного расстояний Хаусдорфа:

|  |  |  |
| --- | --- | --- |
|  |  | (3.2) |

где: – симметричное расстояние Хаусдорфа;

– расстояние Хаусдорфа.

Для того, чтобы нормировать данные, полученные из симметричного расстояния Хаусдорфа (3.2), к ним применяется функция ядра:

|  |  |  |
| --- | --- | --- |
|  |  | (3.3) |

где: – функция ядра;

– симметричное расстояние Хаусдорфа;

– масштабирующий гиперпараметр.

При помощи кроссвалидации был выбран размер масштабирующего гиперпараметра .

Применив функцию (3.3) ко всем картам в тестовом наборе, мы получим матрицу входных параметров сети, каждая строка которой – вектор-признак для одной карты.

|  |  |  |
| --- | --- | --- |
|  |  | (3.4) |

где: – вектор признаков модели;

– симметричное расстояние Хаусдорфа;

– кара из обучающего набора.

Алгоритм генерации матрицы признаков представлен в приложении А. Полученные данные будут подаваться на вход сети. Поскольку данные отражают схожесть карт со всеми другими, сеть будет строить предположение о траекториях основываясь на том, насколько текущая карта похожа на все остальные. Такое преобразование помогает перейти от нерепрезентативного входа, изображенного на рисунке рис. 3.1 к более репрезентативному входу с вероятностным распределением, изображенным на рисунке рис. 3.2.

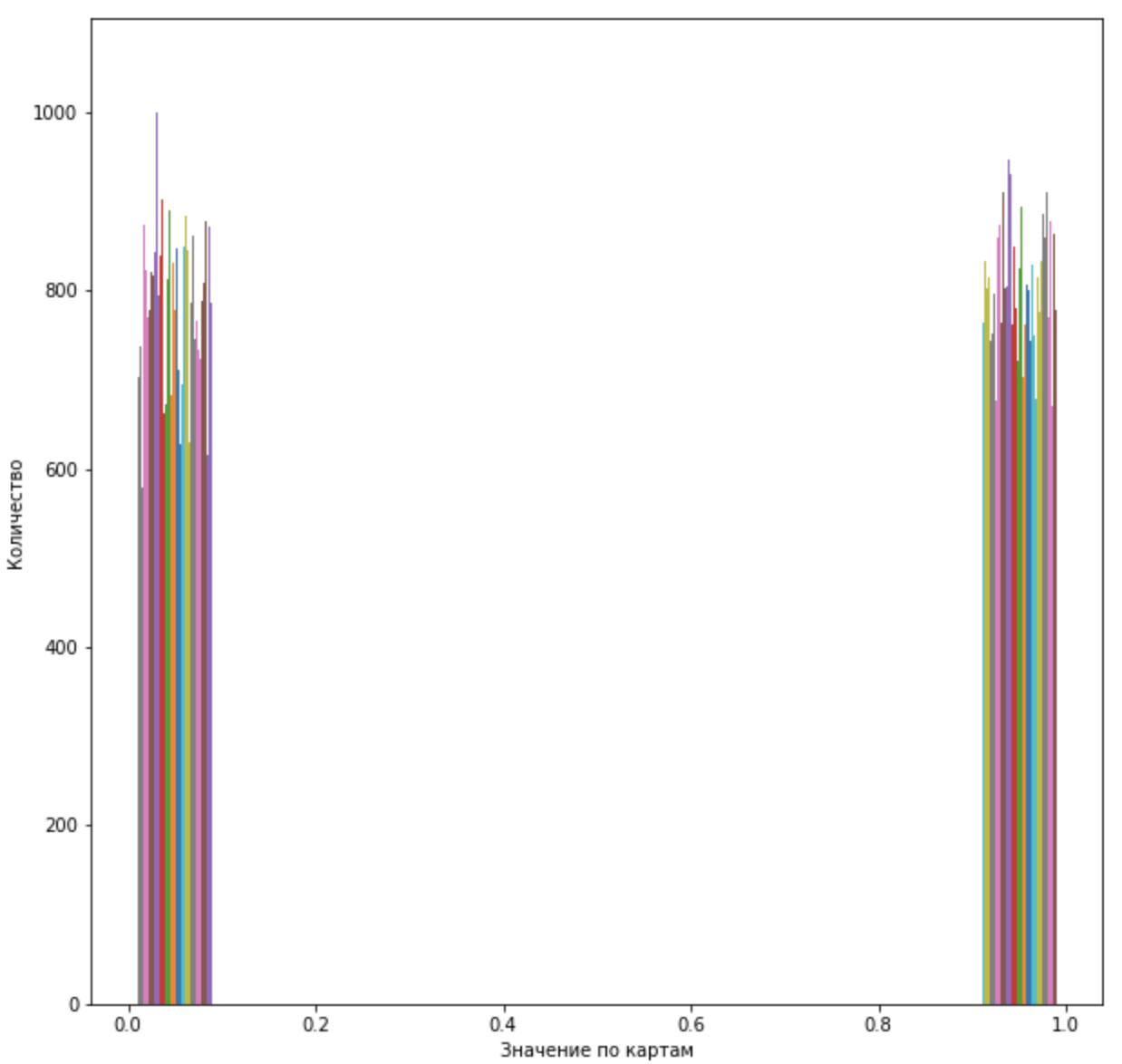


Рисунок 3.1 – Распределение параметров в векторе признаков по исходным картам

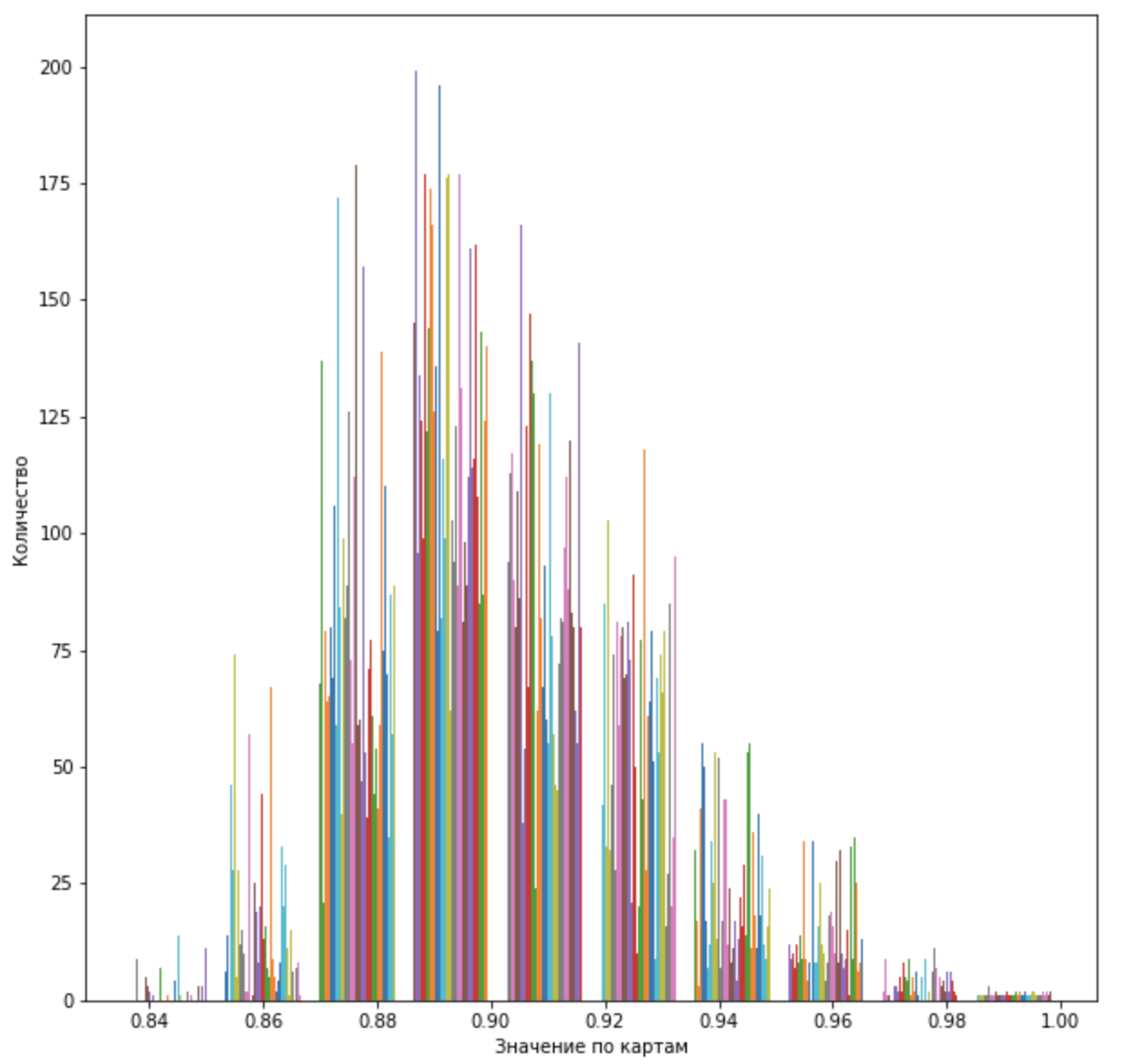


Рисунок 3.2 – Распределение параметров в векторе признаков по расстоянию Хаусдорфа между картами

В данной работе в качестве вектора признака берутся вычисленные расстояния между всеми картами набора. Такой подход используется так-как входных данных крайне мало и карты достаточно однотипны. В случае с набором, в котором содержится множество карт различного вида такой подход не является допустимым, поскольку длина вектора-признаков будет слишком быстро расти, что потребует увеличения количества нейронов в сети и приведет к увеличению времени обучения. В качестве вектора признаков требуется брать по несколько репрезентативных карт. Это означает, что в случае, когда у нас имеется сотня различных типов карт, для вектора признаков будет достаточно сравнивать любую карту набора с некоторыми фиксированными представителями каждого класса среды. Фиксированные представители класса могут быть выбраны случайным образом или по иным соображениям. Точность предсказаний сети в таком случае будет несколько ниже, чем в случае, когда вектор признаков является мерой расстояние между всеми картами, однако такой подход позволяет масштабировать систему к огромному количеству разновидностей карт.

## Кодирование траекторий

Целью работы является генерация траекторий произвольной длины, поэтому обучение также происходить по траекториям произвольной длины. Как было сказано ранее, обучение сети - сложный процесс, который требует множества матричных преобразований и вычисляется на видеокарте. Таким образом, траектории для обучения должны также иметь фиксированную длину.

В наборе данных траектории имеют различную длину, что делает обучение модели невозможным. Распределение и координат для одной карты представлено на рисунке рис. 3.3. Видно, что координаты не нормированы и не укладываются в простую модель распределения, что также негативно влияет на процесс обучения. Такое распределение может приводить к затуханию и взрыву градиента, из-за которого обучение становится невозможным.

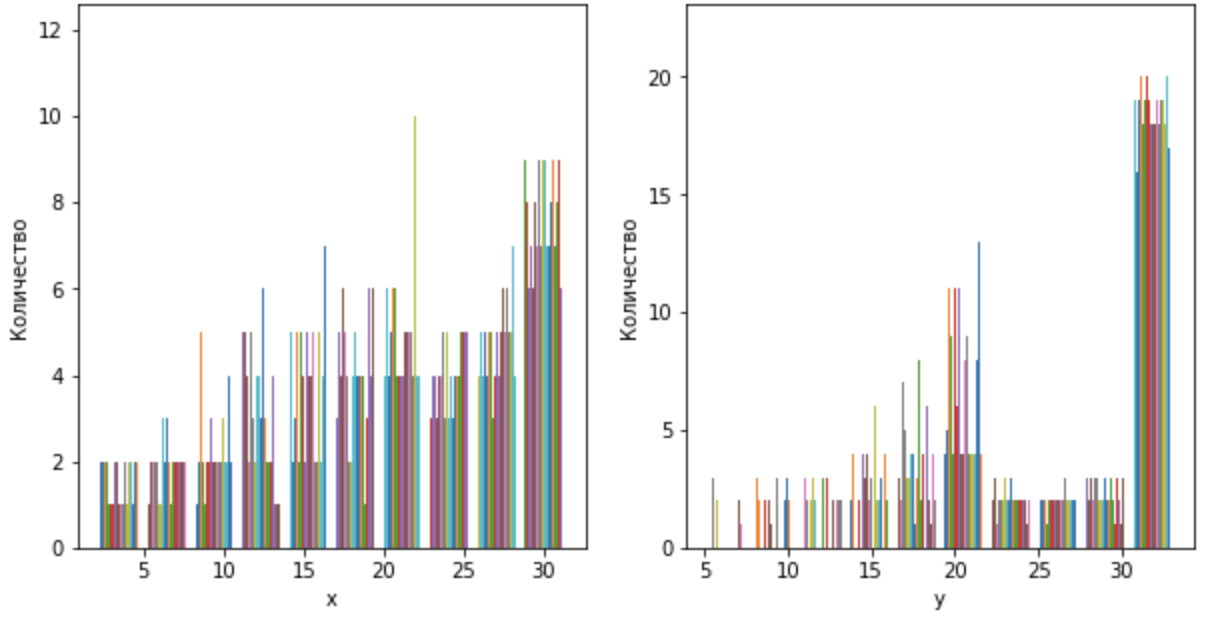


Рисунок 3.3 – Распределение координат траекторий на карте

Необходимо произвести математическое преобразование, которое позволит без потери точности встроить информацию о траектории в вектор фиксированного размера. Идея преобразования координат траекторий строится на представлении дискретных траекторий в некоторую непрерывную функцию. Непрерывная функция способна выдавать значения траекторий в заданный момент времени. В задачах машинного обучения существуют регрессионные модели, которые способны по некоторой числовой переменной, предсказывать значение неизвестной функции. Поскольку траектория — это совокупность функций и значения которых известны только в заданных точках, регрессионная модель может предсказывать значения данных функций, несмотря на то что сами уравнение функции нам не известны. Обученная регрессионная модель способна предсказывать значения функций в любой момент . Существует множество хорошо изученных регрессионных моделей, которые способны предсказывать значения функций, которые могут быть аппроксимированы линейными или полиномиальными функциями, но данные модели не подходят для предсказания функций траекторий. Дело в том, что линейная модель не способна описать нелинейную траекторию, а для использования полиномиальной модели необходимо хорошо понимать приблизительный характер функции, иначе обученная модель будет переобучаться или не обучаться. В теории регрессионных моделей существует прием, который позволяет заменить целевую функцию некоторым ядром, затем обучить модель предсказывать веса для этого ядра, после чего восстановить целевую функцию. Данное преобразование возможно, поскольку в качестве ядра принято брать фиксированные классы функций. Ядро не линейно, что позволяет линейной модели выходить за пределы линейного ограничения.

В данной работе в качества ядра выбрана квадратичная экспоненциальная базисная функция

|  |  |  |
| --- | --- | --- |
|  |  | (3.5) |

где: – параметр радиальной базисной функции;

– центрирующий параметр радиальной базисной функции;

– масштабирующий гиперпараметр.

При помощи кроссвалидации был выбран масштабирующий параметр для функции (3.5).

Чтобы преобразовать веса в вектор фиксированного размера, необходимо выбрать количество радиальных базисных функций для интерполяции. Чем больше количество функций, тем более гладкая получается предсказанная траектория.

Поскольку дискретные траектории имеют различную длину, нужно нормировать время движения таким образом, чтобы оно умещалось в интервал . Для этого преобразуем каждый шаг построения траектории в , которое лежит в нужном интервале по формуле

|  |  |  |
| --- | --- | --- |
|  |  | (3.6) |

где: – число шагов в траектории.

Таким образом по преобразованным временным интервалам (3.6) можно строить вектор значений радиально базисных функций

|  |  |  |
| --- | --- | --- |
|  |  | (3.7) |

где: –фиксированное количество радиально базисных функций;

– фиксированные временные интервалы.

Регрессионная модель, в которой обучение происходит не для целевой переменной, а для некоторого вектора ядер (3.7), а также присутствует регуляризация называется KRR (kernel rigid regression) [7], и ее задача описывается парой уравнений

|  |  |  |
| --- | --- | --- |
|  |  | (3.8) |
|  |  | (3.9) |

где: – вес ядра для переменной ;

– вес ядра для переменной ;

– дискретная переменная в момент времени

– дискретная переменная в момент времени

– вес ядра для в момент времени

– гиперпараметр регуляризации.

Задача, описанная (3.8) и (3.9) состоит в минимизации значения функций. Необходимо извлечь веса обученной модели, которые способны восстанавливать функции траекторий в непрерывном виде. После математического преобразования, получим уравнения для извлечения весов модели:

|  |  |  |
| --- | --- | --- |
|  |  | (3.10) |
|  | , | (3.11) |

где: – единичная матрица.

По полученным из (3.10) и (3.11) весам можно восстановить значения функций в любой момент времени

|  |  |  |
| --- | --- | --- |
|  |  | (3.12) |
|  |  | (3.13) |

где: – значение функции в момент времени ;

– значение функции в момент времени .

Для удобства представления данных в нейронную сеть векторы и объединяются в вектор . Проведя обучение по всем траекториям модели, получим набор весов фиксированной длины, по которым при помощи (3.12) и (3.13) можно восстановить траекторию в любой момент времени:

|  |  |  |
| --- | --- | --- |
|  |  | (3.14) |

где: – вектор весов для траектории номер .

Как было сказано ранее, от выбора количества компонент в векторе весов из матрицы (3.14), зависит гладкость аппроксимирующей функции. Результаты аппроксимации, при и изображены на рисунке рис. 3.4 и рисунке рис. 3.5 соответственно.

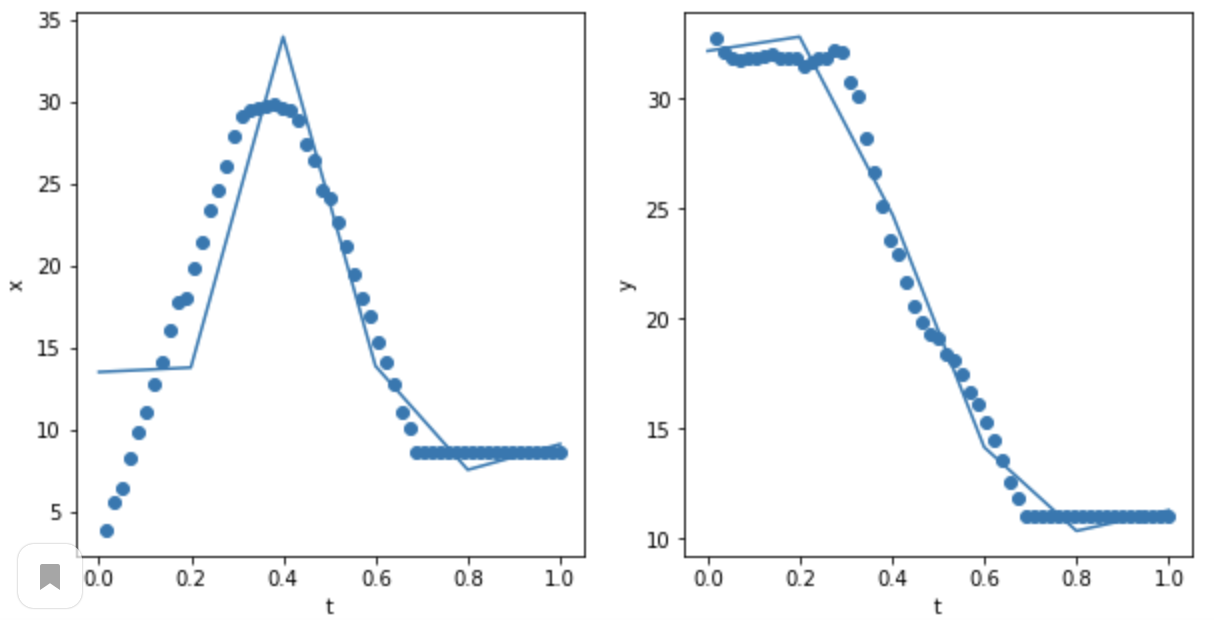


Рисунок 3.4 – Аппроксимация функций траекторий при

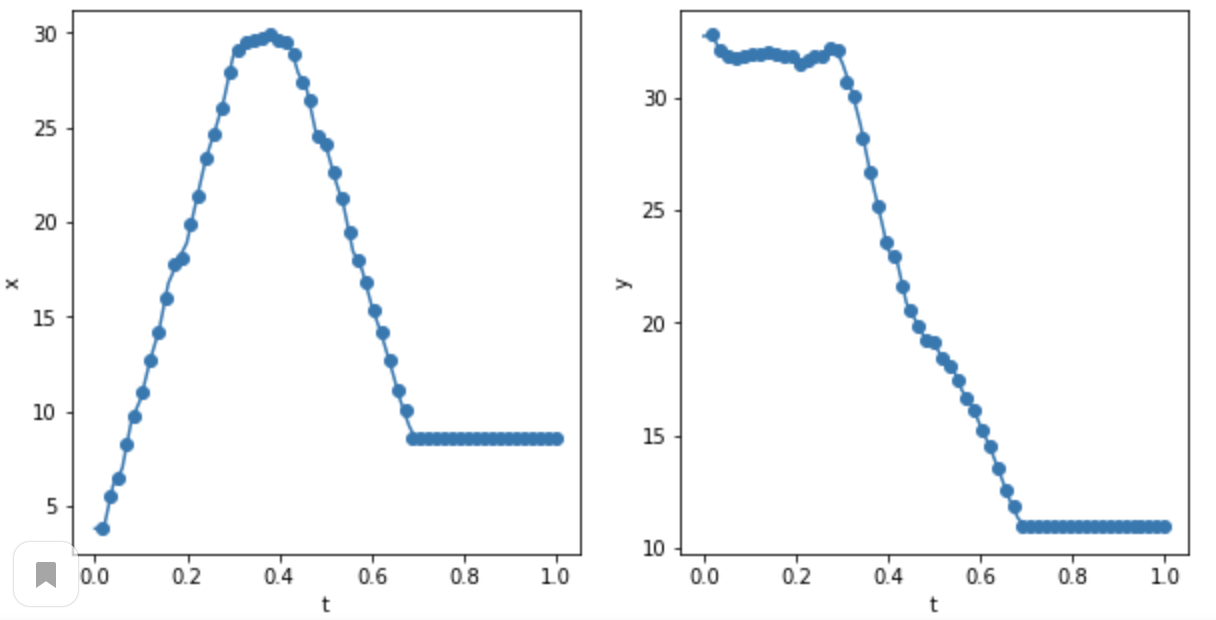


Рисунок 3.5 – Аппроксимация функций траекторий при

Как видно из рисунков, при большем значении функция получается более гладкой, однако нельзя брать слишком большое число, поскольку аппроксимация с помощью экспоненциальных радиальных базисных функций ломается в ситуации, когда количество дискретных точек меньше числа компонент аппроксимации. Для того, чтобы аппроксимировать траектории с параметром было выполнено преобразование траекторий. Ко всем траекториям, длина которых меньше в конец приписывается последнее значение дискретной траектории столько раз, чтобы длина вектора дискретных траекторий была равна . На рисунке рис. 3.5 видно, как последние точки траекторий дублируются, чтобы добиться гладкой аппроксимации. Восстановленные по формулам (3.12) и (3.13) траектории изображены на рисунке рис. 3.6.

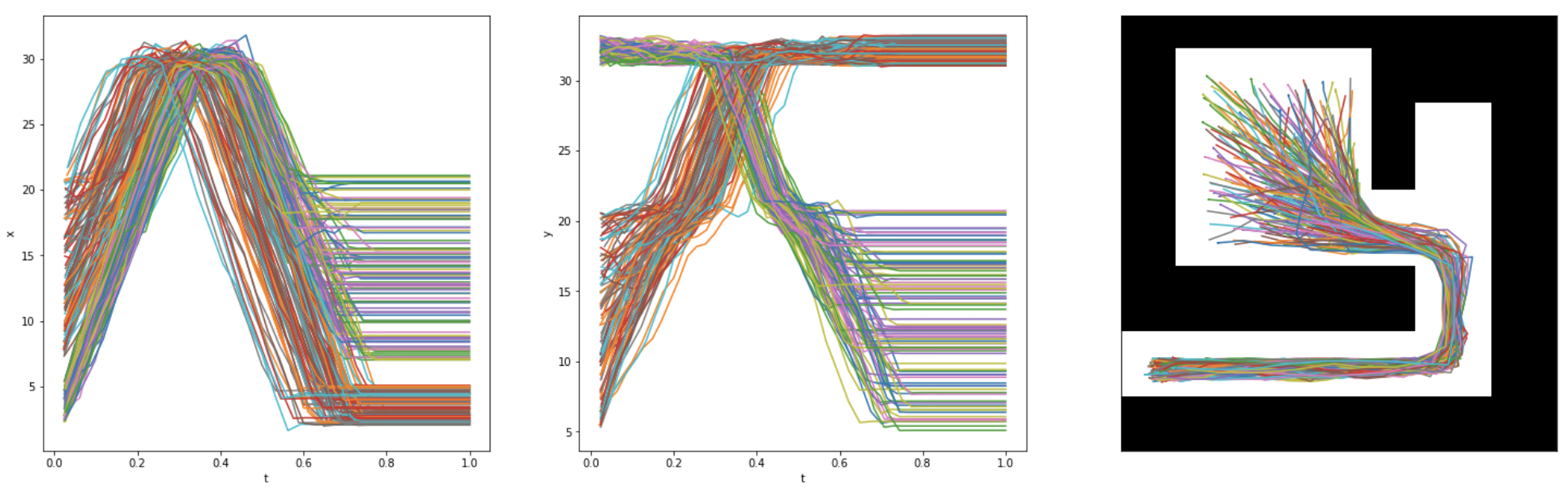


Рисунок 3.6 – Восстановленные траектории

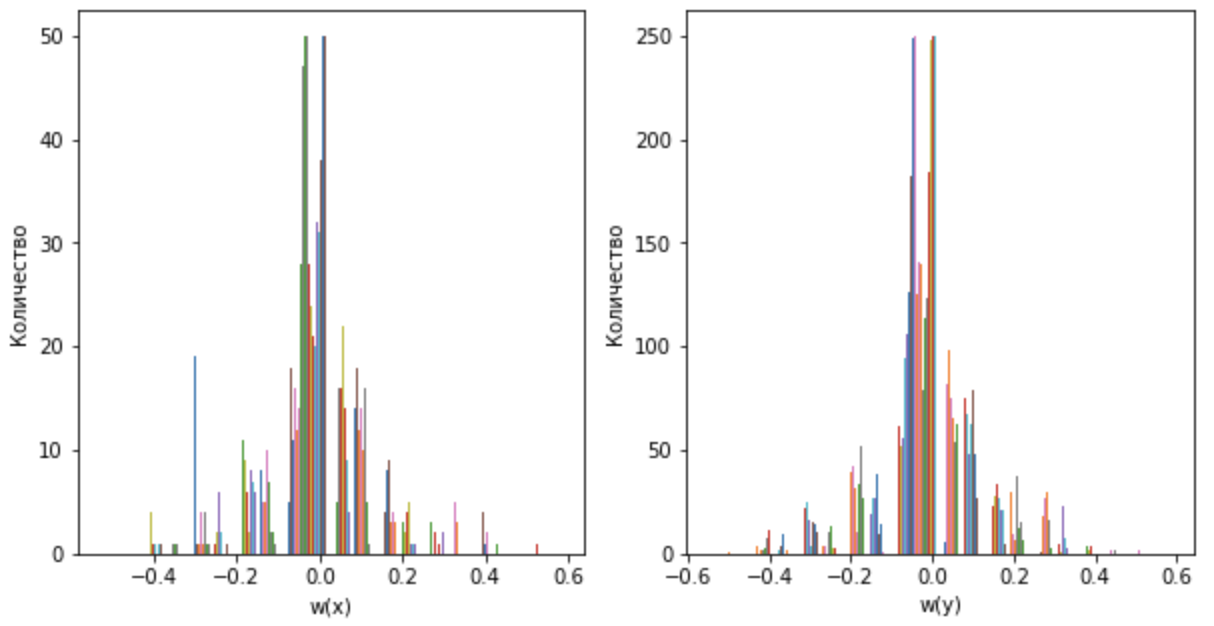
Представление траекторий в виде фиксированного вектора весов также положительно влияет на их распределение. Код встраивания траекторий в вектор фиксированной длины представлен в приложении Б. На рисунке рис. 3.7 показано распределение весов. 

Рисунок 3.7 – Распределение весов

Такое распределение хорошо тем, что оно центрировано относительно 0 и унимодальное. Такое представление весов ускоряет процесс обучения модели в несколько раз и сокращает объем требующихся для обучения данных.

## Архитектура

Архитектура MDN способна предсказывать параметры распределения [8]. Для решения задачи генерации траекторий больше всего подходит распределение Лапласа:

|  |  |  |
| --- | --- | --- |
|  |  | (3.15) |

где: – значение вероятности появления переданной траектории при заданной карте для компонента ;

– номер компонента распределения;

– количество весов в векторе траекторий;

– масштабирующий параметр распределения;

– вес из вектора весов траекторий;

– математическое ожидание;

– вектор признаков.

Выбранное распределение (3.15) показывает наилучшие результаты при кроссвалидации. Кроссвалидация – обучения сети на разных гиперпараметрах с последующим выбором лучших параметров. Конкурентами распределения Лапласса являются Нормальное распределение и распределение Стьюдента [9]. Поскольку распределение траекторий мультимодально, необходимо обучать не одно распределение, а смесь распределений:

|  |  |  |
| --- | --- | --- |
|  | , | (3.16) |

где: – итоговая вероятность построенной траектории при переданном векторе признаков;

– коэффициент, отвечающий за вклад каждой компоненты вероятности в суммарную;

– вероятность для отдельной компоненты.

На рисунке рис. 3.8 представлено наложение функций распределения Лапласса, которые вносят вклад в (3.16) поверх распределения координат траектории.

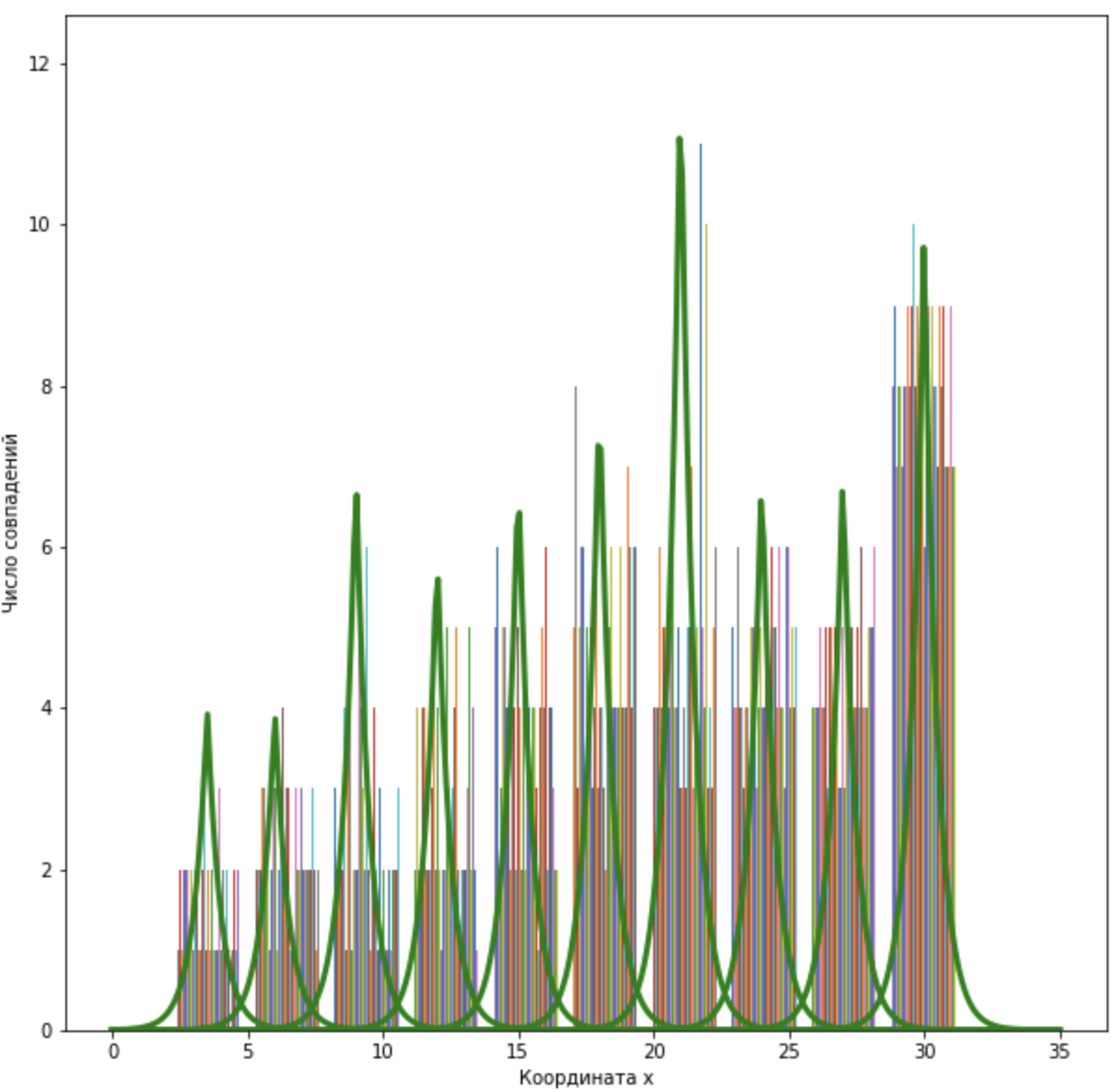


Рисунок 3.8 – Распределение Лапласса по траектории

Нейронные сети строятся по слоям [10]. Данные, попадая в очередной слой, перемножаются на матрицу весов данного слоя, после чего результат подставляется в функцию активации, которая добавляет нелинейность данных. В сети присутствуют и другие слои, без которых модель была бы не способна обучиться на имеющемся наборе данных. Рассмотрим слои, встречающиеся в модели.

### Полносвязные слои

Полносвязный слой представляет из себя слой, состоящий из нейронов указанного количества, каждый из которых связан с каждым нейроном предыдущего слоя. У каждой связи есть свой вес, отвечающий за вклад связи в итоговый результат. Когда данные прогоняются через сеть, происходит перемножение выходов предыдущего слоя со всеми весами связей, что является результатом работы нейрона. Однако, в таком случае модель была бы линейной, поскольку все операции в нейронах являются линейной комбинацией параметров. Такая модель не способна предсказывать нелинейные закономерности. Решить данную проблему помогает функция активации. Такая функция преобразует линейный результат суммы произведения весов в нелинейный. Существует огромное множество функций активации, но в данной сети используются такие функции как ReLU, exponential, softmax. Функция ReLU располагается во внутренних слоях модели, чтобы отсекать значения ненужных «слабых» нейронов. Функция softmax нужна, чтобы выход слоя в сумме был равен 1. А exponential активация нужна, чтобы обеспечить не отрицательность выходных параметров.

### Нормализация

Обучение модели связано с многократным выполнением различных математических операций. В связи с этим, существует риск того, что некоторые веса могут быть очень большими, в то время как другие очень маленькими. Такая проблема может приводить к непредсказуемым последствиям, например, взрывному росту модели, при котором веса становятся равны бесконечности. Также, возможен вариант того, что слишком маленькие веса при постоянном перемножении будут уходить в минус бесконечность. Для решения подобной проблемы существует нормализация. Данный слой сети принимает выходы предыдущего слоя и нормализирует их таким образом, чтобы данные стремились к более репрезентативному математическому ожиданию и дисперсии [11]. Такая модификация входных данных позволяет стандартизировать веса, способствует быстрому обучению и не приводит к выбросам.

### Dropout слои

В процессе обучения нейронных сетей, может произойти ситуация, когда некоторые нейроны начнут активироваться интенсивнее других и запоминать более конкретные закономерности. В результате получается хорошая точность на обучающих данных и плохая на тестовых. Такое поведение связано проблемой переобучения, которая характерна для всех архитектур нейронных сетей. Переобученная модель перестает обобщать предсказываемые данные, но начинает запоминать соответствие вход-выход, что приводит к тому, что модель будет отлично работать только с известными данными, поскольку запомнила их. Решить данную проблему помогает dropout слой [12]. Данный слой на каждую итерацию случайным образом деактивирует заданный процент связей в слое. Таким образом отпадает вероятность того, что некоторые нейроны будут обучаться быстрее. Сеть будет менять параметры более равномерна и не начнет запоминать тестовые варианты.

### Модель

Модель MDN должна предсказывать значения, которые необходимы для построения распределения вероятности. Поскольку было решено использовать распределение Лапласа, необходимо обучить параметры . Нейронная сеть для обучения описана на рисунке рис. 3.9:

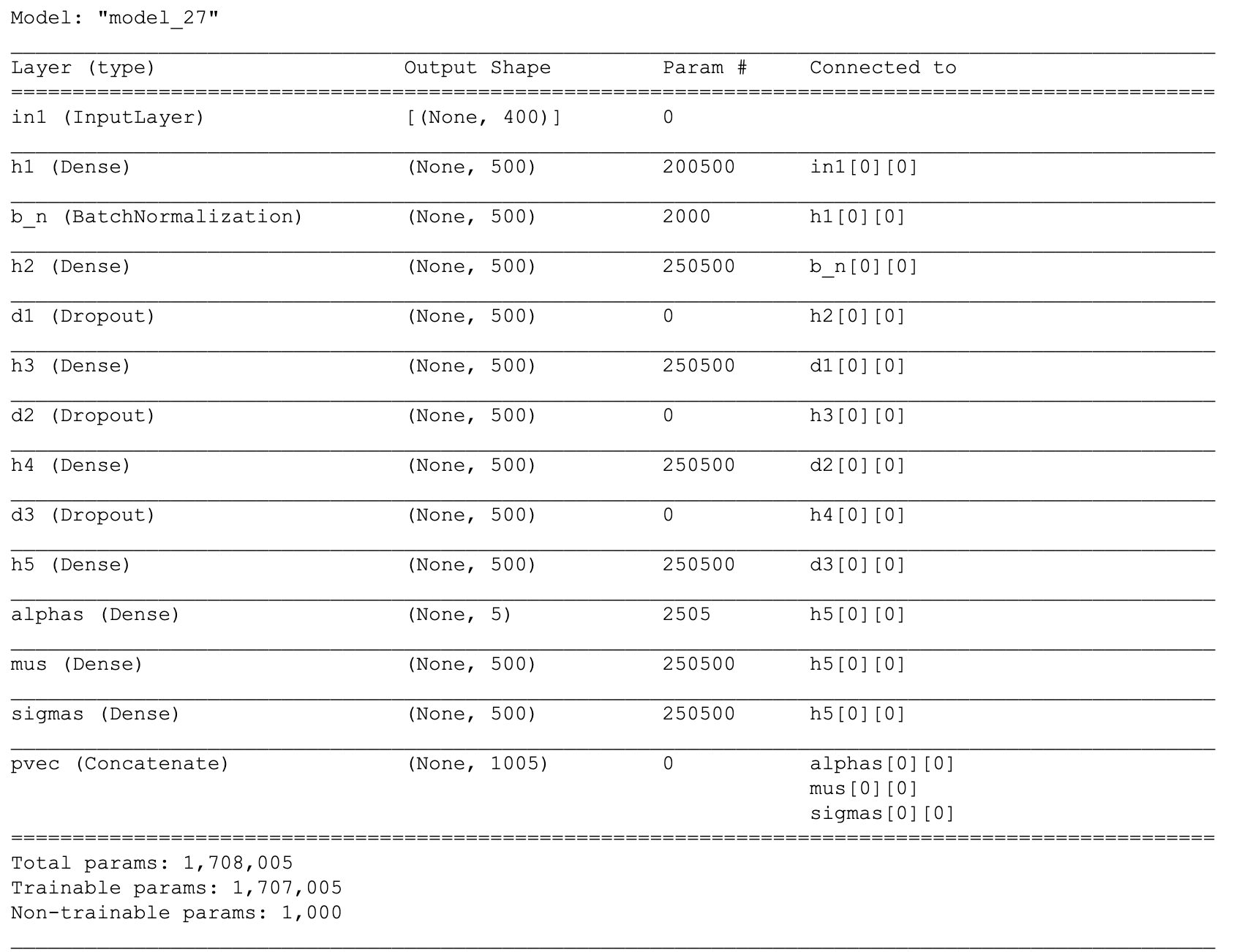


Рисунок 3.9 – Устройство обученной MDN сети

Сеть состоит из пяти полносвязных скрытых слоев, с функцией активации ReLU [13], после первого полносвязного слоя следует слой нормализации данных, который помогает избегать взрывов и выбросов. Все слои чередуются с dropout слоями, чтобы избежать переобучения. Слои «alphas», «mus» и «sigmas» выходят из последнего скрытого полносвязного слоя и служат для предсказания распределения. На выходе модели предсказания конкатенируются, чтобы в виде одного параметра попасть в функцию потерь.

## Обучение

Обучение модели производилось при помощи Tensorflow – Фреймворка машинного обучения от Google. Данный Фреймворк не выполняет операции напрямую, а представляет их в виде графа операций, который передается в модуль, отвечающий за распараллеливание. Данный модуль способен разбить заданный граф на графы меньшего размера, которые могут быть выполнены параллельно. Затем, Фреймворк передает неделимые графы для расчётов на процессор или видеокарту, в случае если архитектура видео карты поддерживает вычисления.

### Процесс обучения

Обучение происходит по эпохам. В конце каждой эпохи значение функции потерь записывается в историю, чтобы можно было оценить, насколько успешно модель обучилась.

Обучение модели состоит из нескольких этапов. На первом этапе на вход модели подаются векторы признаков в количестве, равном размеру batch выборки. Данные проходят по всем слоям сети, на каждом из которых с весами производится вычисление линейной комбинации, активация и другие операции. После прохода по всей сети, вычисляется предсказание модели, которое попадает в функцию ошибки. После вычисления функции ошибки выполняется ее обратное распространение [14].

Процесс обратного распространения ошибки выполняется при помощи оптимизатора, который указывает на сколько и в какую сторону необходимо сместить веса модели, чтобы уменьшить значение функции потерь. При обратном распространении ошибки вычисляются частные производные функции ошибки по каждому весу модели, чтобы определить вклад каждого веса в итоговый результат, после чего веса изменяются на размер, указанный оптимизатором в соответствии со своим вкладом в функцию ошибки [15].

Процесс повторяется для следующей batch выборки, но уже по измененным весам. Так продолжается, пока вся обучающая выборка, разбитая по batch выборкам, не пройдет через сеть. После прохода всех выборок завершается эпоха. Сеть обучается указанное число эпох, после чего обучение прекращается и модель считается обученной.

### Оптимизатор

На сегодняшний день известно множество способов оптимизации функций. Оптимизаторы бывают разных видов. Некоторые определяют шаг оптимизации при помощи поиска антиградиента функции, другие работают с поиском эвристик в значениях функции, также существуют оптимизаторы гладких функций, которые определяют вторую производную функции для определения шага оптимизации [16]. В данной работе используются оптимизатор, работающий методом градиентного спуска. Градиент показывает направление максимального роста функции, но так как оптимизатор должен минимизировать функцию потерь, при оптимизации вычисляется антиградиент. Фреймворк содержит множество реализованных алгоритмов оптимизации, поэтому нет необходимости реализовывать собственный алгоритм. В работе применяется оптимизатор Adam.

### Функция потерь

Функция потерь является важнейшей составляющей любой модели обучения. Именно функция ошибки способна сообщить насколько успешно работает модель и при помощи нее в модель вкладывается смысл. Нейронная сеть не имеет представления о то, для чего она обучается. Данные при проходе через сеть подвергаются математическим операциям, после чего функция ошибки, которая понимает для чего обучаются данные, сообщает сети о том, насколько плохо было сделано предсказание.

В работе нейронная сеть обучается, чтобы предсказывать параметры распределения Лапласа, которое описывает распределение весов траекторий. Чем лучше обучена модель, тем ближе находятся к истинным , а подбирается таким образом, чтобы покрыть разброс относительно центра распределения. Поскольку, как было сказано ранее, траектории имеет мультимодальное распределение – нужно обучать сразу распределений, каждое из которых входит в суммарное распределение с некоторым . Исходя из описанных выше условий функция ошибки для обучаемой модели принимает следующий вид:

|  |  |  |
| --- | --- | --- |
|  | , | (3.17) |

где: – количество карт в обучающей выборке;

– количество траекторий на карте ;

– количество компонент распределения.

Данная функция (3.17), после очередного прохождения данных по сети, вычисляет отрицательную функцию логарифмического правдоподобия. Функция правдоподобия в статистике служит для определения того, насколько правдоподобно предсказание для реальных данных. Поскольку в машинном обучении принято минимизировать функции потерь, в качестве ошибки принимается именно отрицательное правдоподобие, чтобы показать, насколько не правдоподобно предсказание модели. Поскольку обучение связано с многократным выполнением математических операций, значения функции. Ошибки могут неконтролируемо расти и выходить за пределы вычисляемых компьютером диапазонов. В таком случае функция вернет минус бесконечность и все обучение закончится ошибкой. Для того, чтобы избежать взрывного роста ошибки, правдоподобие вычисляется под логарифмом, который сглаживает значение функции.

Несмотря на то, что применяется отрицательное логарифмическое правдоподобие, функция ошибки все равно может неконтролируемо расти, что приведет к получению одного из недопустимых результатов: . Такое происходит по причине того, что в заданной функции потерь, используется многократное произведение данных, которые при значении >1 приводят к росту величин до , а при значениях <1 приводят к уменьшению величин до 0, что недопустимо для логарифма. Таким образом необходимо воспользоваться свойствами логарифма

|  |  |  |
| --- | --- | --- |
|  |  | (3.18) |
|  |  | (3.19) |

чтобы получить функцию, которая будет стабильна при выполнении операций. Таким образом, при помощи (3.18) и (3.19) функция ошибки задается в следующем виде:

|  |  |  |
| --- | --- | --- |
|  |  | (3.20) |

В таком виде функция ошибки вычисляется в сети на каждом шаге.

### Результаты обучения

Построенная сеть обучается 20 эпох. Результаты графика изменения ошибки (3.20) по эпохам представлены на рисунке рис. 3.10.

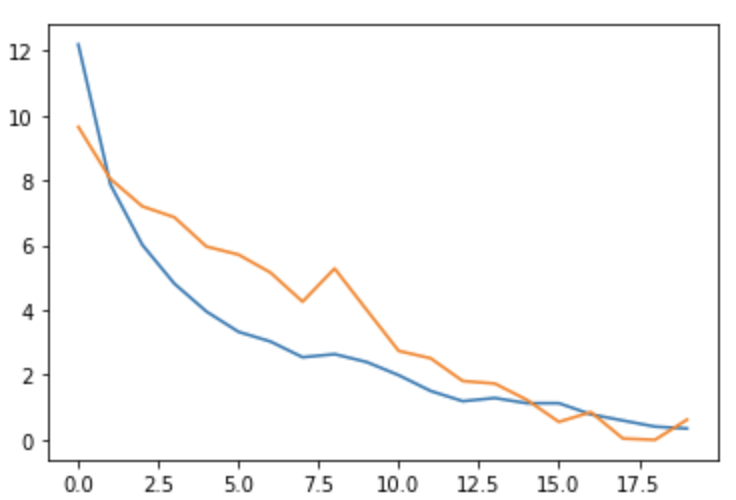


Рисунок 3.10 – График уменьшения ошибки по эпохам.

В результате обучения, модель приобретает возможность генерировать параметры для распределения Лапласа, на основании которого будут строиться траектории.

## Генерация траекторий

Процесс генерации траекторий состоит из этапов семплирования, валидации и построения. При семплирование можно использовать несколько подходов - выбирать параметры с максимальным значением , случайным образом выбирать вектор компонент, перемножать на полученые параметры и брать среднее.

Выбор максимального приводит к тому, что выбирается всегда самый большой компонент распределения, который как правило соответствует только одному направлению траекторий, поэтому для покрытия траекториями всей карты такой способ не подходит. Вариант с усреднением параметров, приводит к сильным скачкам при семплирование, что делает большинство траекторий недопустимыми. Вариант со случайным выбором подходит лучше всего, поскольку каждый компонент отвечает за отдельное направление движений на карте.

Таким образом для построения траекторий необходимо выполнить следующие шаги:

* Сгенерировать случайное число от 0 до 1;
* Идти по массиву компонент, суммируя значения до тех пор, пока сумма не превысит сгенерированное число;
* Запомнить индекс, при котором наступило превышения порога;
* Взять из предсказанного вектора такие и которые соответствуют выбранному индексу;
* Инициализировать массив весов выбранными ;
* Пройти по всем весам прибавляя к ним случайное число из распределения Лапласа с центром в точке 0 и масштабирующим параметром равным выбранным ;
* Умножить вектор весов на радиальные базисные функции.

Повторив указанный выше алгоритм нужное число раз, можно сгенерировать требуемое число траекторий. Код для генерации траекторий представлен в приложении В. На рисунке рис. 3.11 представлено построение 50 траекторий, а на рисунке рис. 3.12 400 траекторий, что больше, чем на любой карте из обучающего набора.



Рисунок 3.11 – 50 сгенерированных траекторий на новой карте.

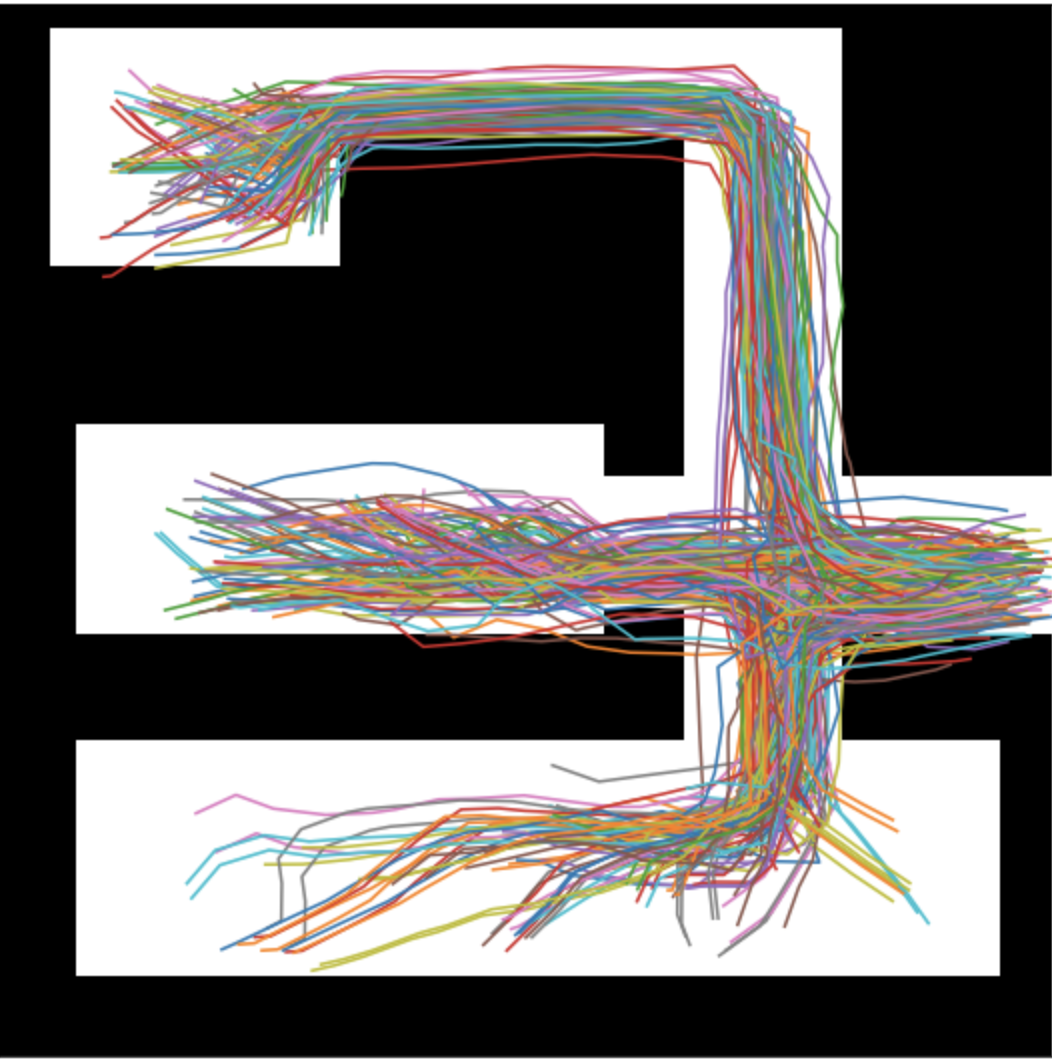


Рисунок 3.12 – 400 сгенерированных траекторий на новой карте.

На рисунке показаны все сгенерированные траектории, без применения валидации при построении. Чтобы отбросить все недопустимые траектории можно в момент перемножения весов на вектор радиальных базисных функций понять, что произошло пересечение и повторить семплирование траектории заново. На рисунке рис. 3.13 продемонстрировано применение валидации.

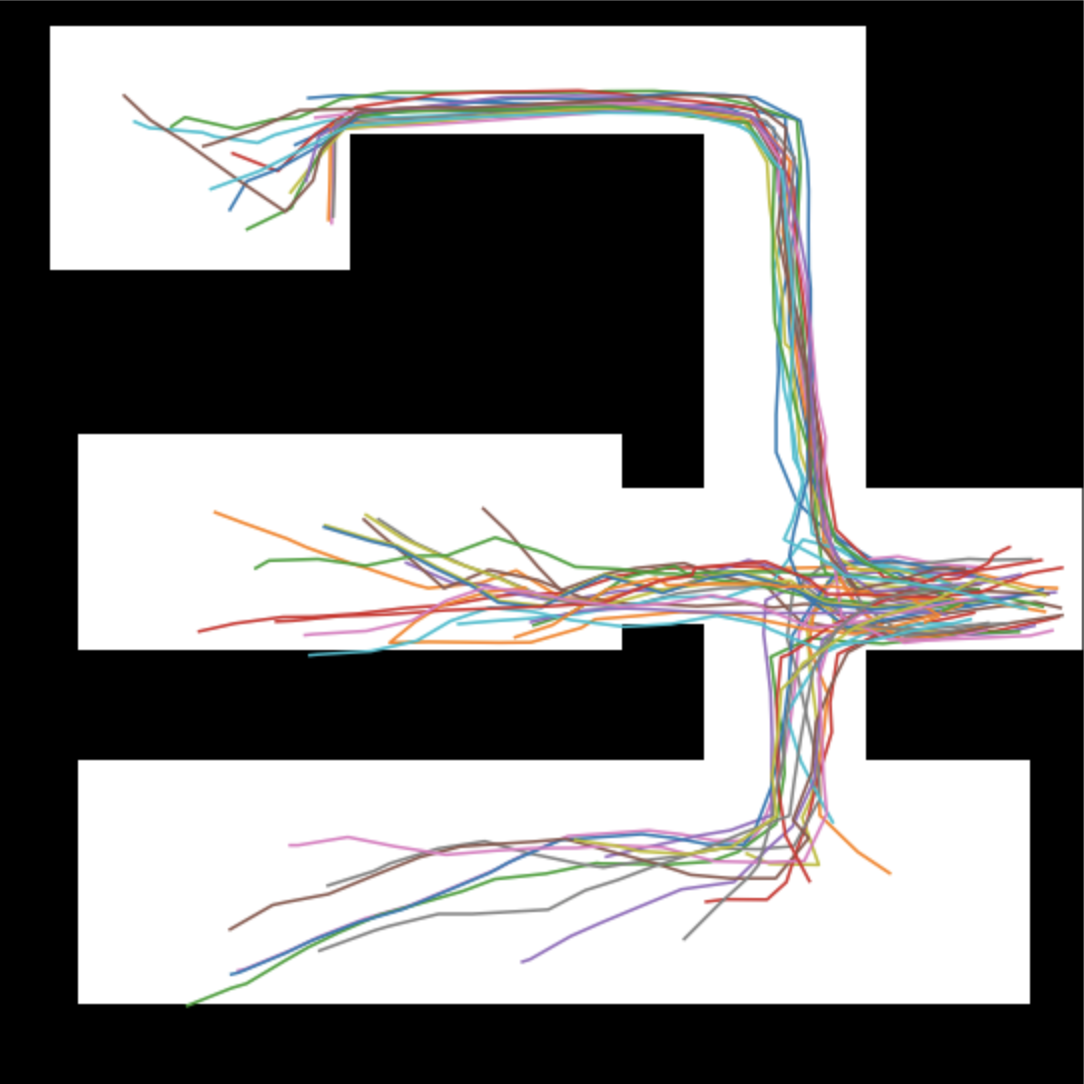


Рисунок 3.13 – 50 траекторий, прошедших валидацию.

Таким образом, обучив статическую модель OCTNet на сгенерированных с помощью поворота данных из набора Occ-Traj120, можно генерировать нужное количество траекторий, которые похожи на настоящие траектории движения в данной среде. Обученная модель способна строить корректное распределение для любых карт схожей конфигурации, состоящих из прямоугольных комнат и коридоров и имеющих размер . Для данных другой конфигурации необходимо единожды переобучить модель. Для того, чтобы модель была способна строить траектории на широком спектре пространств, требуется предоставить набор данных карт различной конфигурации с построенными на них траекториями и переобучить модель, как указывалось ранее.

# МОДЕЛЬ OCTNet ДЛЯ ДИНАМИЧЕСКОЙ СРЕДЫ

Для построения траекторий в динамической среде, необходимо обучать модель видео данных изображающих изменение карты и траекторий во времени. Архитектура сети также должна измениться, поскольку параметры распределения постоянно изменяются во времени.

## Данные

Для обучения динамической модели, нужны динамические данные. По той причине, что не существует хорошей базы даже для статических данных карт и траекторий, поиск динамических данных для обучения является проблемой. Поиски необходимых данных не дали результатов, поэтому единственным выходом является генерация динамических данных из статических.

### Фильтр

Для добавления динамики, на карту добавляется препятствие. Поскольку карта представлена в виде нулей и единиц, препятствие должно соответствовать формату данных. Фильтр представляет из себя двумерную матрицу такого же размера, что и карта. В фильтре все регионы отмечены как свободные и только в занятых участках располагаются единицы. Наложение фильтра на карту происходит за линейное от размеров карты время. Каждая ячейка фильтра сравнивается логической функцией ИЛИ с каждой ячейкой карты. Поскольку размеры фильтра и карты идентично, проверка наложения происходит за линейное время. Преобразование выглядит следующим образом:

При необходимости оптимизации можно кратно ускорить наложение фильтра, а также существенно экономить память. Суть оптимизации состоит в использовании битовых масок. Например, для кодирования карт можно брать 40 чисел типа long, которые проинициализированы нулем. Каждый long способен закодировать 64 бита информации, что делает его способным представить один ряд фильтра или карты с длиной до 64. Записывая в нужный бит числа long ноль или единицу, становится возможным закодировать каждый ряд карты или фильтра. Затем выполняя операцию логического ИЛИ между long из карты и long из фильтра получается ряд из результирующей карты. Операция ИЛИ между двумя числами типа long выполняется за константу, таким образом скорость наложения фильтра возрастает в несколько раз. В данной работе такая оптимизация не требуется по причине того, что карты маленькие. На рисунке рис. 4.1 продемонстрировано наложение фильтра-препятствия на карту с последующим его перемещением.



Рисунок 4.1 – Перемещение наложенного фильтра по карте

### Отсечение пересечений

При движении преграды по карте, необходимо отфильтровывать траектории, которые стали недопустимыми после перемещения препятствия. В общем случае для поиска пересечений отрезка и многоугольника используется алгоритм с заметающей прямой. В данной работе в качестве препятствия используется движущийся по карте квадрат, поэтому нет необходимости в универсальном подходе. Алгоритм поиска пересечений упрощается до поиска пересечения двух отрезков и состоит из следующих шагов:

* Преобразование фильтра в 4 отрезка;
* Преобразование траектории в последовательность отрезков, соединённых между собой;
* Проверка пересечения каждого отрезка траектории с каждым отрезком фильтра;
* Удаление из множества траекторий, один из отрезок которых имеет пересечение хотя бы с одним отрезком фильтра.

Для поиска пересечения отрезков выполняется проверка на пересечение двух прямых содержащих данные отрезки. В случае отсутствия пересечения прямых, отрезки однозначно не пересекаются, но в случае наличия пересечения прямых, отрезки не пересекаются, если точка пересечения не находится ни на одном из отрезков. Для того, чтобы траектории не прилипали к препятствию вплотную, вокруг фильтра имеется отступ, относительно которого рассчитывается пересечение. На рисунке рис. 4.2 изображен результат работы отсечения траекторий. Видно, что ни одна из траекторий не лежит внутри контейнера, который отвечает за отступ от препятствия.

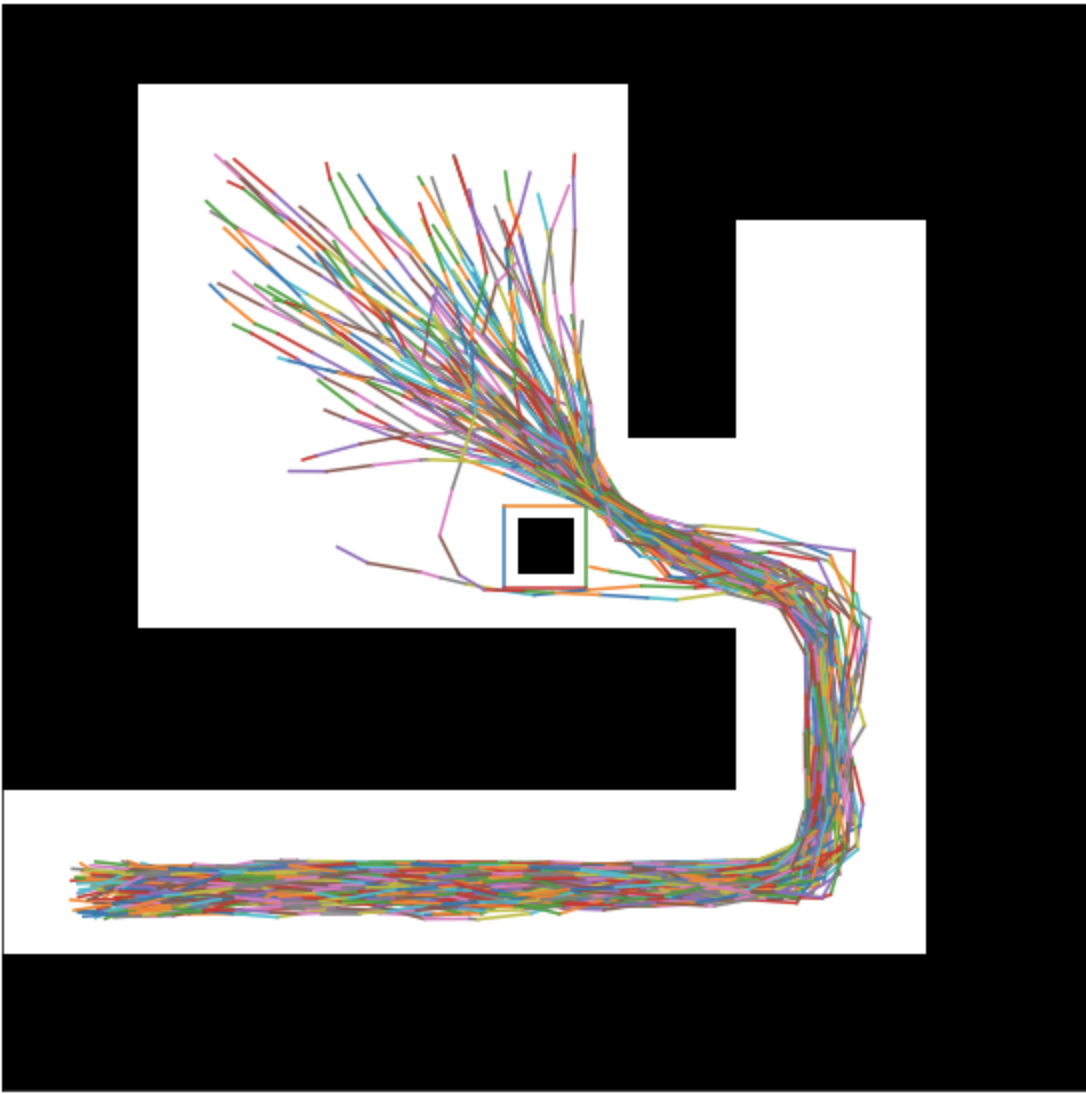


Рисунок 4.2 – Отсечение траекторий, пересекающихся с контейнером вокруг препятствия

### Генерация видео данных

Видео данные – статические данные, изменяющиеся во времени. Для генерации данных применяется фильтр, который смещается в нужном направлении, отсекая пересекающиеся траектории. В данной работе выполняется генерация видео длиной в двадцать кадров, на которых препятствие перемещается вертикально вверх из центра до верха карты. На рисунке рис. 4.3 представлена генерация первых 4 кадров видео.



Рисунок 4.3 – Первые 4 кадра сгенерированного видео

Генерация видео из статики является ресурсоемкой операцией, поэтому сгенерированные данные сохраняются на диск для дальнейшего использования. Алгоритм для создания видео данных представлен в приложении Г. Генерация видео данных для всего набора может занимать до суток. При помощи сгенерированных данных можно обучить модель для генерации траекторий в динамической среде.

## LSTM

LSTM сети долгой краткосрочной памяти. Данный тип сетей относится к рекуррентным нейронным сетям и широко используется в генеративных алгоритмах. Сети данной модели содержат в себе ячейки LSTM, которые принимают сразу несколько входных данных. В данную ячейку приходят данные с входного слоя, а также выходные данные данного слоя на предыдущей итерации. Таким образом сеть получает возможность применять не только входные данные для обучения, но также и запоминать результаты предыдущих этапов обучения. Такое устройство помогает сети видеть намного более сложные закономерности в данных. На рисунке рис. 4.4 представлена схема LSTM ячейки.



Рисунок 4.4 – LSTM ячейка

Данная ячейка содержит текущий вход модели, выход предыдущей итерации, а также порог для запоминания или забывания информации. Данные пороги нужны для того, чтобы сеть запоминала весомые закономерности и забывала незначащие, не загромождая входы следующих итераций ненужными данными.

Проблема данных сетей состоит в нескольких вещах. Данные сети больше обычных подвержены взрыву и затуханию градиента по причине того, что данные каждого предыдущего слоя приходят в следующий. Не смотря на защиту в виде порога забывания и порога запоминания рекуррентная архитектура не всегда устойчива в данным проблемам.

В данной работе LSTM ячейки используются для решения задачи предсказания следующего шага. Поскольку данной архитектуре требуется много данных для обучения, можно соединить выход MDN сети с генератором, который отдавать на вход LSTM сети. Данная комбинация позволит обучить параметры распределения для известных данных, что позволит сети интерполировать промежуточные состояния, затем подавать данные на вход LSTM для обучения сети экстраполировать данные.

## Архитектура

Для обучения MDN сети на видео данных, необходимо изменить форму сети. Можно обучать сеть тремя различными способами. Первый способ заключается в том, чтобы разбить динамические данные на множество кадров и обучать сеть аналогичным со статичными данными путем. Второй способ заключается в добавлении новой размерности к данным и независимому применению функции потерь к каждому экземпляру новой размерности с последующим усреднением результата. Третий вариант заключается в добавлении новой размерности к данным и обучении сети с новой размерностью, но применением функции потерь разу ко всем размерностям.

### Использование статической модели

Данный подход привлекателен тем, что не требует никаких модификаций по сравнению со статической моделью. Также идеологически данный подход является предпочтительным по той причине, что при разбиении по кадрам, получается среда с единственным препятствием. Обучившись на перемещении препятствий в различных комнатах, OCTNet должна строить траектории в новых комнатах, где ранее не наблюдалось движение. Однако данный подход имеет важный минус. Число карт при разбиении видео по кадрам составляет более 20000 различных конфигураций, что вызывает сложности при построении вектора признаков. Для решения данной проблемы можно ограничить количество карт, по которым будет составляться вектор признаков количеством равным 400, по аналогии со статической моделью. Однако, выход сети будет подаваться на вход другой сети, поэтому потеря качества предсказаний может стать более критичной. По данной причине использование простой статической модели и разбиение по кадрам не подходят для решения поставленной задачи.

### Независимое применение функции потерь

При данном подходе функция потерь вычисляется независимо для каждого кадра и усредняется перед выполнением обратного распространения ошибки. Плюсом данного подхода является более контролируемый рост ошибки, поскольку количество перемножающийся параметров не возрастает относительно статической модели. Функция потерь принимает следующий вид:

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

Данный подход имеет существенный минус, при обратном распространении вклад каждого веса модели в ошибку (4.1) будет вычисляться именно по усредненной ошибке, что замедляет обучение. Данная проблема не критична, поэтому такое устройство подходит для рассмотрения в качестве первой сети.

### Применение функции потерь ко всему выходу

Данный способ построения сети заключается в том, что функция потерь будет рассчитываться по всему выходу модели сразу. Данный подход позволяет более интенсивно модифицировать веса модели и делать более уверенные шаги оптимизации на каждой итерации. Главный минус такого подхода заключается в возможности возникновение затухания или взрыва градиента. Функция потерь в данном случае принимает вид:

|  |  |  |
| --- | --- | --- |
|  |  | (4.2) |

По причине возможного возникновения взрыва или затухания градиентов и весов модели после вычисления ошибки (4.2) данный подход не является предпочтительным, хотя и предоставляет более быстрое обучение.

## Обучение MDN сети

В качестве модели для обучения была выбрана модель MDN с добавлением новой размерности и подсчетом функции потерь по каждому кадру независимо с последующим усреднением. Такой подход хоть и замедляет обучение, но является самым стабильным из всех.

Вектор признак кодируется для каждого кадра по отдельности. Таким образом вектор признаков становится матрицей признаков, каждая строка которой отвечает за один кадр. Таким образом размер матрицы признаков для каждого элемента составляет . Вычисление расстояния Хаусдорфа происходит с фиксированным набором карт. Поскольку данная операция выполняется для каждого кадра независимо, качество обучения не падает.

Траектории кодируются аналогичным со статической моделью образом, с единственным различием в том, что к данным добавляется размерность, отвечающая за номер кадра. Также количество траекторий выравнивается под максимальный размер. Такое преобразование нужно для того, чтобы на вход сети приходили тензоры одинаковой формы. Если некоторая карта содержит меньше траекторий, то траектории в ней дублируются до максимального количества выборки.

Также важно удалить из выборки видео, не содержащие траекторий. Такое может получиться в ходе генерации данных, например, когда преграда полностью перекрывает коридор. Такие видео являются неестественными аномалиями, которые будет негативно влиять на модель в процессе обучения.

На рисунке рис. 4.5 изображено строение модели MDN для обучения динамических данных. Данная модель отличается от статического варианта тем, что количество параметров увеличилось кратно количеству кадров для обучения, также изменилась и функция ошибки.

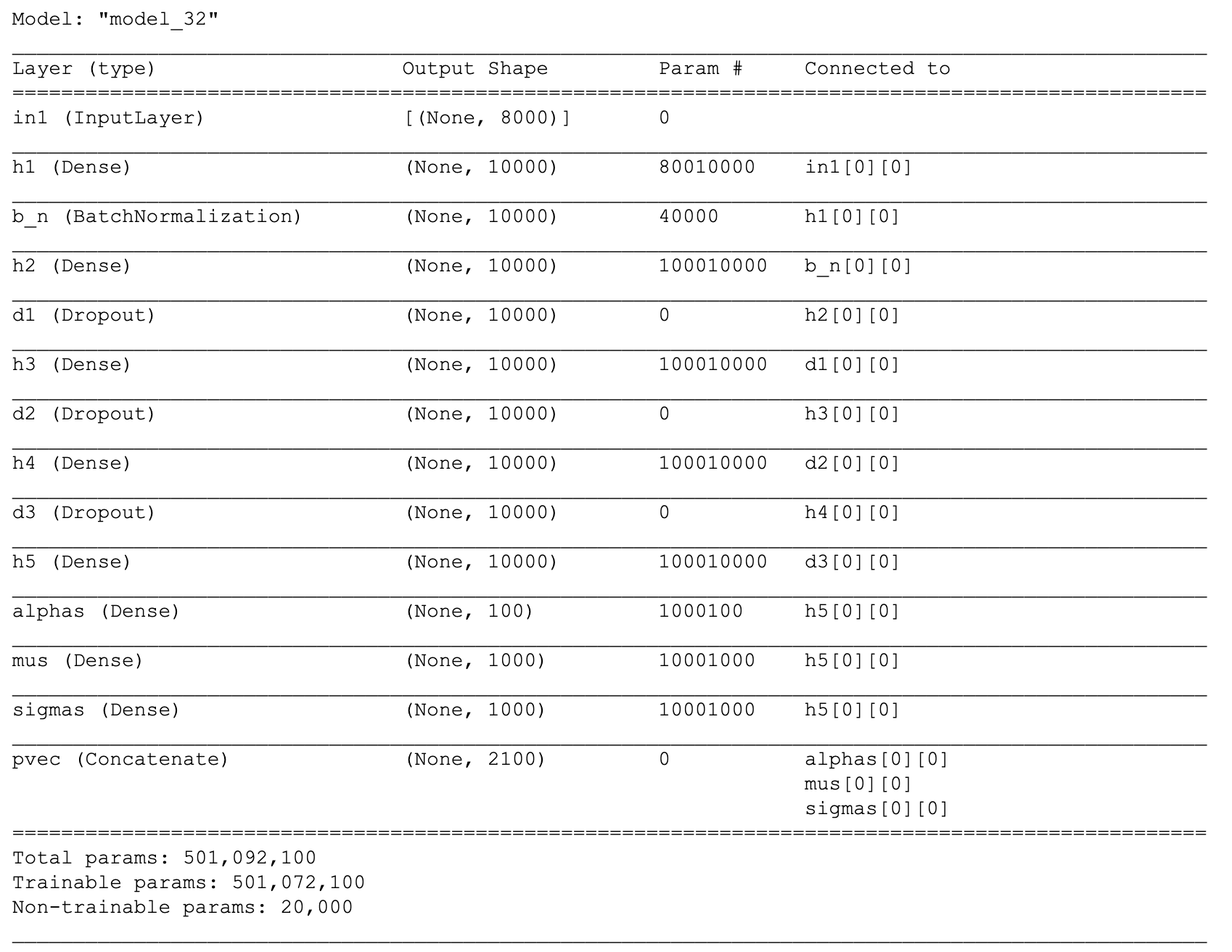


Рисунок 4.5 – Модель обучения для динамической среды

Поскольку данных стало значительно больше, нет необходимости повторять обучение на протяжении 20 эпох. Для обучения сети на динамических данных сеть обучалась на протяжении 10 эпох. На рисунке рис. 4.6 представлен график уменьшения ошибки сети по ходу обучения.

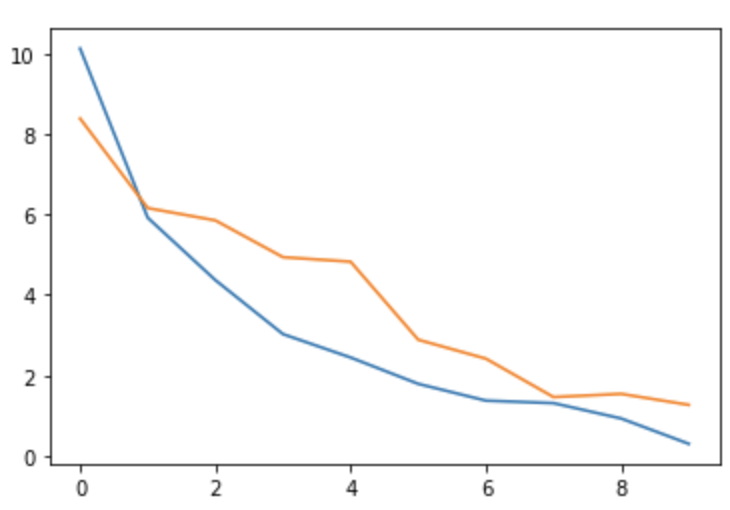


Рисунок 4.6 – Значение функции ошибки по эпохам

## Предсказание следующего кадра

MDN сеть, обученная на тренировочных данных способно обобщать траектории, которые находятся в пределах движения обучающих данных. Например, сеть, обученная на данных с квадратом, движущимся до середины карты будет стремительно терять предиктивную способность при дальнейшем движении квадрата. Для решения данной проблемы используются сети предсказания следующего кадра. Полносвязные и сверточные нейронные сети не способны решать подобные задачи по той причине, что характер данных может не задаваться последним известным кадром. Для извлечения характера изменения кадра используются рекуррентные нейронные сети, которые владеют информацией не только о текущем состоянии кадра, но и о всех необходимых состояниях, произошедших ранее. Использование рекуррентных сверхточных сетей ограничивается вычислительными способностями. Обучение таких сетей часто приводит к затуханиям и взрывам градиентов. Наиболее стабильной архитектурой рекуррентных сетей является LSTM. При помощи порогов забывания и запоминания данная сеть меньше других склонна к возникновению вычислительных ошибок. На рисунке рис. 4.7 изображено строение LSTM сети для обучения предсказания следующего кадра. На вход данной сети поступают данные в виде последовательности карт с построенными на них траекториями. Модель предсказывает запрошенное количество следующих кадров.

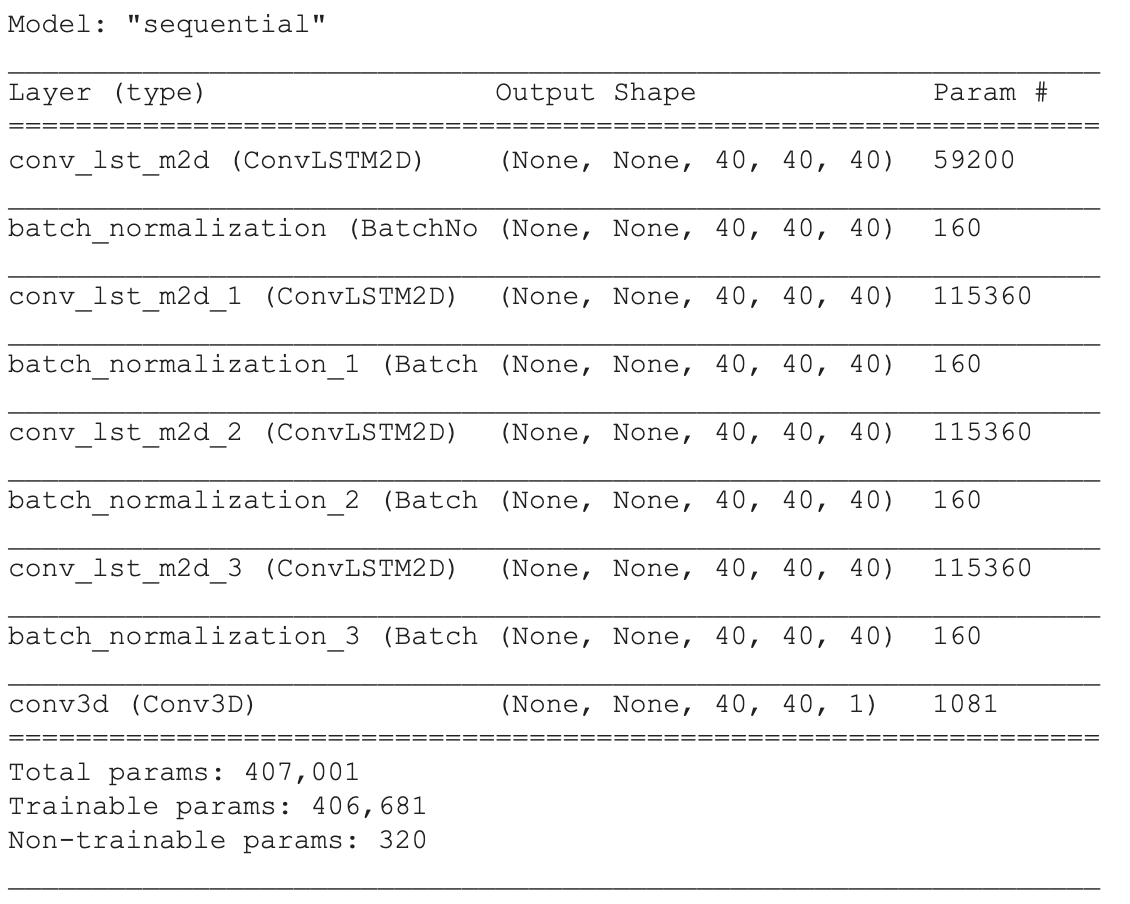


Рисунок 4.7 – Строение LSTM сети

Обучение рекуррентных сетей происходит крайне долго из-за огромного количества входных данных. Так происходит по той причине, что даже при существовании порога запоминания и забывания в ячейку сети могут попадать данные из нескольких десятков предыдущих итераций. На этапе обратного распространения ошибки необходимо вычислять частные производные также по всем весам предыдущих итераций. Известно, что для качественного обучения сети предсказывающий следующие кадры, необходимо несколько тысяч эпох обучения. В данной работе предпринималась попытка обучить сеть на протяжении 200 эпох, однако после завершения 9 эпохи было принято решение остановить обучения и проводить эксперименты с моделью, обученной до данного состояния. Такое решение было принято из-за длительности обучения одной эпохи, которая занимает около 40 минут, на машине с 32 гигабайтами оперативной памяти и процессором Intel i7. Процесс изменения функции потерь по эпохам обучения представлено на рисунке рис. 4.8.

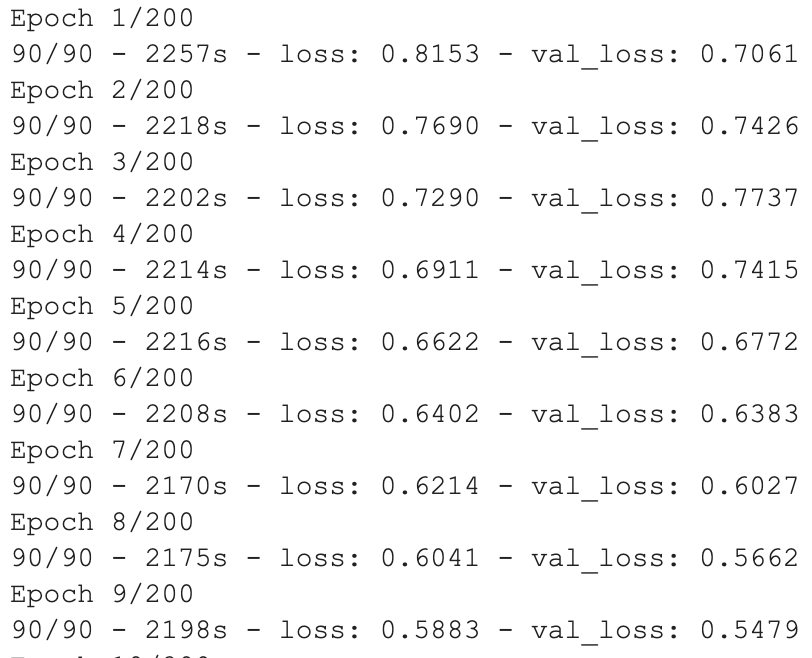


Рисунок 4.8 – Обучение сети на протяжении 9 эпох

После обучения модели на протяжении 9 эпох, модель способна предсказывать следующие кадры, однако на данном этапе они содержат большое количество шума. После качественного обучения сети кадры становятся более чистыми.

## Генерация траекторий

Генерация траекторий в динамической среде выполняется при помощи использовании двух обученных моделей.

Модель на основе MDN способна с высокой точностью генерировать траектории в виде непрерывной функции для карт на которых ранее не наблюдалось движение. Данная модель имеет ограничение для динамических систем. Чтобы алгоритм работал, необходимо обучить сеть на всех движениях преград по карте. В случае, когда движение преграды выходит за пределы обучающей выборки, модель быстро теряет предиктивную способность и начинает генерировать большое количество недопустимых маршрутов. В рамках небольших территорий данное ограничение не приносит больших проблем, поскольку в рамках промышленного склада, можно задать все возможные движения автоматизированных агентов по карте, после чего обучить модель. Однако для больших карт, для которых предоставление всех вариантов движения не представляется возможным, необходимо использовать LSTM модель, которая обучена на предыдущих кадрах и способна предсказывать следующий.

Данная модель принимает на вход кадры сгенерированных траекторий в рамках известного движения преград, чтобы предсказать запрошенное количество следующих кадров.

Алгоритм генерации траекторий в динамической среде в общем случае изображен на рисунке рис. 4.9.

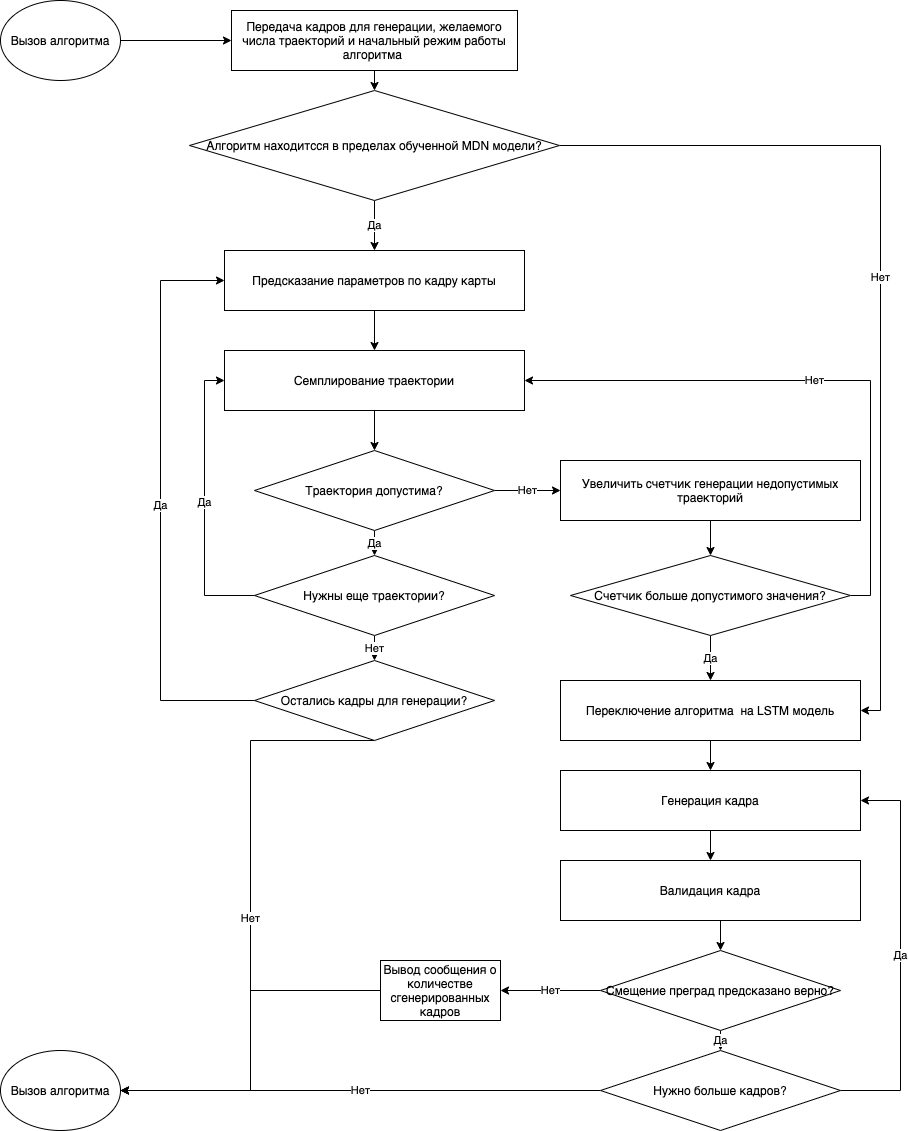


Рисунок 4.9 – Алгоритм генерации траекторий в динамической среде

Таким образом алгоритм работает в двух состояниях. Предпочтительным вариантом генерации траекторий является режим работы MDN, поскольку в данном режиме траектории предсказываются в виде непрерывной функции и могут быть семплированы до нужной длины. Также на выходе при данном режиме работы сразу получатся траектории в виде координат. Режим работы с LSTM моделью достаточно рискованный, поскольку система предсказывает изображение следующего кадра с построенными траекториями. Данное изображения сравнивается при помощи расстояния Хаусдорфа с переданным кадром и в случае заданного критерия сходства принимается или отвергается. В случае принятия предсказания, необходимо запустить алгоритм заметающей прямой по сгенерированной карте, чтобы прервать изображения траекторий в координаты. В данном режиме работы алгоритм не обязательно сгенерирует нужное количество кадров. В случае, когда модель предсказывает карту, которая не похожа на переданный кадр, предсказания останавливаются, а модель выдает то количество кадров, которое смогла предсказать с заданным критерием уверенности. Таким образом, запрашивая у алгоритма нужное количество кадров и траекторий в каждом кадре, передается только пожелание по результатам работы. Алгоритм старается выдать результат максимально близкий к запрашиваемому, однако в случае невозможности продолжать работу алгоритм останавливается и выдает имающиеся на данный момент сгенерированные кадры.

На рисунке рис. 4.10 изображен результат работы алгоритма в режиме MDN.



Рисунок 4.10 – Генерация траекторий в режиме работы с моделью MDN

В случае, когда возможно обучить модель на всем допустимом диапазоне движения объекта по картам, точность сгенерированных траекторий получается достаточно высокой. Такой способ генерации является предпочтительным. В случае, когда не представляется возможным обучить систему на всем диапазоне движения преград, допускается обучение только на некотором диапазоне движений. Однако, все равно должны быть отражены все возможные характеры движения объектов на карте, иначе модель не будет иметь никакого представления о том, в какую сторону отклоняется распределение координат траекторий движения. В рамках данной работы LSTN сеть обучена до низкой точности, поэтому не представляется возможным продемонстрировать генерацию траекторий. Однако, данная архитектура правильно предсказывает характер движения двух фильтров, которые в обучающем наборе двигаются с низа карты до середины. На рисунке рис. 4.11 продемонстрирована генерация следующего кадра, который находится за пределами обучающей выборки.



Рисунок 4.11 – Генерация следующего кадра движения фильтров.

Таким образом алгоритм будет работать в диапазоне известного движения агентов по карте, и имеет потенциал работы за пределами известных диапазонов движения.

# СРАВНЕНИЕ ХАРАКТЕРИСТИК РАБОТЫ АЛГОРИТМОВ

После реализации алгоритма для генерации траекторий в статической и динамической среде, необходимо сравнить производительность реализованного решения с рассмотренными ранее аналогами. В качестве аналогов в статической среде алгоритмы с моделями машинного обучения GAN, CVAE, KTM. В качестве аналогов в динамических средах рассматриваются GAN и CVAE.

Показатели модели будут сравниваться по средней скорости построения 100 траекторий на 100 картах. Также будет сравниваться среднее расстояние между каждой из 100 сгенерированных траекторий и ближайшей действительной траекторией. Поскольку частота генерации недопустимых траекторий является важной характеристикой генеративных алгоритмов, необходимо также сравнить выбранных кандидатов по данной метрике.

В работе не реализуются сет GAN, CVAE и KTM поскольку их реализация требует больших усилий и выходит за рамки работы. В открытых источниках содержится информация о всех интересующих нас метриках. На рисунке рис. 5.1 представлена генерация 100 траекторий сетью GAN.



Рисунок 5.1 – Результат генерации 100 траекторий сетью GAN

Сеть CVAE строит схожие с GAN траектории, поскольку также основывается на архитектуре состязающихся сетей. Построение траекторий моделью KTM отличается, поскольку происходит не на основании изображения, а строится по наблюдаемым траекториям. Результат генерации нескольких сотен траекторий представлен на рисунке рис. 5.2.

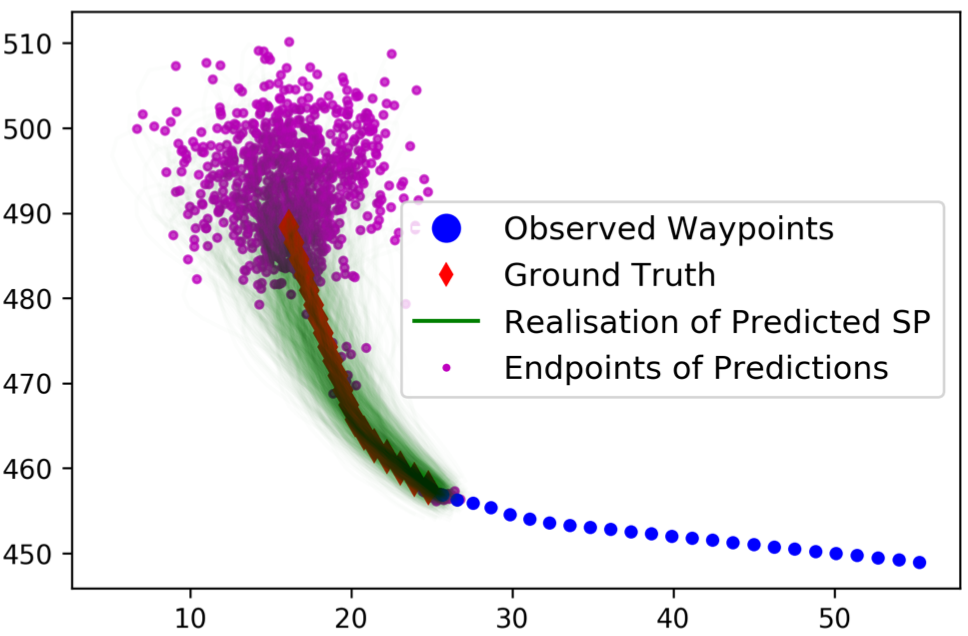


Рисунок 5.2 – Результат генерации траекторий при помощи модели KTM

## Результаты сравнения

Реализованные модели OCTNet были запущены для построения 100 траекторий на 100 различных картах. Рассмотренные аналоги и результаты тестирования реализованной модели занесены в таблицу табл. 5.1 и таблицу табл. 5.2.

Таблица 5.1 – Характеристики алгоритмов в статических средах

|  |  |  |  |
| --- | --- | --- | --- |
| Название подхода | Hausdorf | Frechet | Доля недопустимых траекторий |
| **Алгоритм с моделью OCTNet** | **1.86** | **2.00** | **0.24** |
| Алгоритм с моделью GAN | 10.48 | 14.67 | 0.88 |
| Алгоритм с моделью CVAE | 11.79 | 16.66 | 0.91 |
| Алгоритм с моделью KTM | 1.84 | 1.95 | 0.21 |

По результатам сравнения видно, что модели GAN и CVAE не подходят для генерации траекторий. Модель KTM показала себя лучше всего, однако из-за неспособности предсказывать траектории в неизвестных средах и высокого потребления памяти данная модель не подходит для построения траекторий в динамической среде. Для сравнения точности в динамической среде берутся OCTNet, GAN и GAN. Стоит отметить, что каждая динамическая карта содержит 20 кадров, для которых необходимо построить траектории. Реализованная модель была обучена на всей траектории движения, поэтому генерация происходит полностью в режиме работы MDN сети.

Таблица 5.2 – Характеристики алгоритмов в динамических средах

|  |  |  |  |
| --- | --- | --- | --- |
| Название подхода | Hausdorf | Frechet | Доля недопустимых траекторий |
| **Алгоритм с моделью OCTNet** | **2.11** | **2.34** | **0.30** |
| Алгоритм с моделью GAN | 15.64 | 16.20 | 0.99 |
| Алгоритм с моделью CVAE | 15.88 | 17.01 | 0.99 |

По результатам, представленным в таблице видно, что ни один из аналогов не составляет модели OCTNet конкуренции при генерации маршрутов в динамической среде. Таким образом реализованный алгоритм лучше всего подходит для генерации близких к реальным траекторий в динамической среде.

# ОЦЕНКА И ЗАЩИТА РЕЗУЛЬТАТОВ ИНТЕЛЛЕКТУАЛЬНОЙ ДЕЯТЕЛЬНОСТИ

## Понятие интеллектуальной собственности

Понятие интеллектуальная собственность определено в Конвенции об учреждении Всемирной организации интеллектуальной собственности — интеллектуальная собственность это права, относящиеся к литературным, художественным и научным произведениям, исполнительской деятельности артистов, звукозаписи, радио и телевизионным передачам, изобретениям во всех областях человеческой деятельности, научным открытиям, промышленным образцам, товарным знакам, знакам обслуживания, фирменным наименованиям и коммерческим обозначениям, защите против недобросовестной конкуренции, а также все другие права, относящиеся к интеллектуальной деятельности в производственной, научной, литературной и художественной областях [17].

В Российской Федерации понятие «Интеллектуальная собственность» регламентируется 1225-й статьей ГК РФ. Из текста статьи следует, что интеллектуальная собственность — это охраняемые законом результаты интеллектуальной деятельности и средства индивидуализации [18].

## Результаты интеллектуальной деятельности

Согласно статье 1225-й четвертой части Гражданского кодекса [19], результатами интеллектуальной деятельности и приравненными к ним средствами индивидуализации юридических лиц, товаров, работ, услуг и предприятий, которым предоставляется правовая охрана являются:

* произведения науки, литературы и искусства;
* программы для электронных вычислительных машин (программы для ЭВМ);
* базы данных;
* исполнения;
* фонограммы;
* сообщение в эфир или по кабелю радио- или телепередач (вещание организаций эфирного или кабельного вещания);
* изобретения;
* полезные модели;
* промышленные образцы;
* селекционные достижения;
* топологии интегральных микросхем;
* секреты производства (ноу-хау);
* фирменные наименования;
* товарные знаки и знаки обслуживания;
* наименования мест происхождения товаров;
* коммерческие обозначения.

Никакие иные объекты на сегодняшний день в не считаются видами интеллектуальной собственности в соответствии с ГК РФ.

В зарубежных странах существуют свои перечни объектов интеллектуальной собственности, не совпадающие с тем, что действует в Российской Федерации.

## Характеристика видов интеллектуальной деятельности

Все виды интеллектуальной собственности, перечисленные выше делятся на 3 группы:

* результаты интеллектуальной деятельности;
* средства индивидуализации;
* иные виды интеллектуальной собственности.

К первой группе относится 7 видов интеллектуальной собственности, ко второй группе относится 4 вида, а к третьей 5 видов.

Результаты интеллектуальной деятельности имеют автора и являются материальным объектом интеллектуальной собственности.

К средствам индивидуализации относятся нематериальные объекты, которые указывают на другие объекты, выделяют, индивидуализируют. Некоторые средства индивидуализации указывают не на объект, а на определенный субъект права. Вместе с тем средства индивидуализации не являются результатами интеллектуальной деятельности - у них нет авторов.

К иным видам интеллектуальной собственности относятся объекты, которые не принадлежат к первым двум группам.

### Произведения науки, литературы и искусства

Данный вид интеллектуальной собственности относится к группе «результаты интеллектуальной деятельности». Объекты, относящиеся к данному виду, часто именуются «авторские произведения». Они должны являться результатами самостоятельного творческого труда автора.

Произведения науки, литературы и искусства - самая распространенная и самая ценная категория интеллектуальной собственности. Правовая охрана «авторских произведений» возникает с момента их создания, без необходимости выполнения государственной регистрации. Данный вид интеллектуальной собственности является одним из самых многочисленных благодаря простоте возникновения права. Самыми ценными произведениями являются компьютерные программы, песни, а также фильмы и телепрограммы. Основные нормы о правовой охране произведений сосредоточены в главе 70 ГК РФ - «Авторское право».

### Исполнения

Исполнения включают в себя результаты деятельности артистов-исполнителей. Результаты данного вида интеллектуальной собственности всегда являются творческим продуктом. Их правовая охрана возникает автоматически, без регистрации.

Право авторства на исполнения принадлежат артисту-исполнителю. Нормы правовой охраны исполнений указаны во втором параграфе главы 71 ГК РФ. Данный вид интеллектуальной собственности принадлежит к группе «результаты интеллектуальной деятельности».

### Изобретения

Изобретения относятся к группе «результаты интеллектуальной деятельности». К изобретениям относятся технические новинки и технические решения различных видов. Результаты данного вида интеллектуальной собственности должны быть новыми и неочевидными.

Все права, в том числе и право авторства на изобретение, возникают только в результате государственной регистрации, которая осуществляется в Федеральной службе по интеллектуальной собственности - Роспатент.

### Полезные модели

В отдельную группу, именуемую «полезные модели» относят «малые изобретения». К полезным моделям зачастую относятся технические новинки, которые являются частью какого-либо устройства. Правовые нормы и правила регистрации данного вида интеллектуальной собственности аналогичны изобретениям. Относятся к группе «результаты интеллектуальной деятельности».

### Промышленные образцы

Промышленные образцы — это объекты, имеющие внешний вид промышленного изделия, содержащий художественные черты.

Относятся к группе «результаты интеллектуальной деятельности» и охраняются, как и изобретения. Нормы, касающиеся правовой охраны изобретений, полезных моделей и промышленных образцов, сосредоточены в главе 72 ГК РФ «Патентное право».

### Селекционные достижения

Селекционные достижения относятся к группе «результаты интеллектуальной деятельности». К данному виду интеллектуальной собственности относятся новые сорта растений и породы животных, полученные селекционным путем.

Являются результатами творческой деятельности. Получают правовую защиту после государственной регистрации. Основные нормы об охране селекционных достижений содержатся в главе 73 ГК РФ «Право на селекционное достижение».

### Топологии интегральных микросхем

Данный вид интеллектуальной собственности относится к группе «результаты интеллектуальной деятельности». К топологии интегральных микросхем относятся объемные системы, определяющие взаимное расположение отдельных элементов электронной микросхемы. Устройство микросхемы должно быть оригинальными. Правовая охрана возникает автоматически, без государственной регистрации. Основные нормы об их правовой охране - в главе 74 ГК РФ «Право на топологии интегральных микросхем».

### Фирменные наименования

Объекты данного вида относятся к группе «средства индивидуализации» — это названия коммерческих юридических лиц. Часто, объекты данного вида интеллектуальной собственности именуются «фирма». Фирма указывается в учредительных документах юридического лица при его регистрации. Самостоятельно фирма не может быть зарегистрирована. Нормы, касающиеся правовой охраны фирменных наименований, содержатся в § 1 главы 76 ГК РФ.

### Товарные знаки

Товарные знаки это - охраняемые гражданским правом обозначения товаров, результатов работ и услуг. Могут быть зарегистрированы в Роспатенте, либо во Всемирной организации интеллектуальной собственности. Имеют очень большое значение, как средства рекламы на товарных рынках. Основные нормы, касающиеся правовой охраны товарных знаков, - в § 2 главы 76 ГК РФ. Является средством индивидуализации.

### Наименования мест происхождения товаров

К данному виду относятся охраняемые гражданским правом обозначения какого-либо географического объекта, где производятся традиционные товары, обладающие особыми качествами. Является средством индивидуализации. Охраняются так же, как и товарные знаки.

### Коммерческие обозначения

Объекты данного вида относятся к группе «средства индивидуализации». К коммерческим обозначениям относятся: обозначения предприятий, служащие для индивидуализации этих комплексов. Охрана объектов данного вида возникает автоматически с начала использования. Государственная регистрация для данного вида интеллектуальной собственности не требуется.

### Фонограммы

К фонограммам относятся записи звуков. Охрана возникает сразу после записи, автоматически, без регистрации. Фонограмма не считается результатом творческой деятельности. Авторов у фонограммы быть не может, но всегда есть изготовитель фонограммы. Правовая охрана фонограмм описана в § 3 главы 71 ГК РФ.

### Радио- и телепередачи

Правовая охрана возникает автоматически с момента вещания у организации, осуществляющей вещание.

### Большие базы данных

Охрана возникает автоматически. Владельцем базы данных считается ее изготовитель. Изготовитель не считается творцом или автором. Правовая охрана на данный момент не прописана.

### Первые публикации произведений

К объектам данного вида интеллектуальной собственности относятся публикации, которые не охраняются авторским правом, или если срок охраны истек.

### Секреты производства (ноу-хау).

Секреты производства — это различные технические, производственные и другие сведения, в основном - в научно-технической сфере, хранящиеся в секрете и не являющиеся общедоступными. Охрана предоставляется любому правообладателю. Для возникновения охраны не требуется выполнять регистрацию. Основные нормы о правовой охране секретов производства зафиксированы в главе 75 ГК РФ «Право на секрет производства (ноу-хау)».

## Объект интеллектуальной деятельности работы

Результатом выполнения выпускной квалификационной работы является код компьютерной программы, способный обучить модель генерировать траектории, которые свойственные для определенной среды. Данный алгоритм будет полезен для работы роботизированных систем. Код программы уникален и не имеет аналогов.

Компьютерный код относится к произведениям науки, литературы и искусства, поэтому правовая охрана интеллектуальной собственности возникает с момента их создания, без необходимости выполнения государственной регистрации. Нормы о правовой охране компьютерного кода описаны в главе 70 ГК РФ - «Авторское право».

## Нормативно-правовое акты, регулирующие вопросы защиты результатов интеллектуальной деятельности

Среди основных законов, регулирующих правовую защиту интеллектуальной собственности, можно выделить следующие:

1. Конституция РФ. Закрепляет обязанность закона охранять интеллектуальную собственность. (пункт 1, статья 44, глава 2) [20]
2. Гражданский кодекс РФ. (Часть четвертая) Определяет понятия интеллектуальной деятельности и описывает права на результаты интеллектуальной деятельности и средства индивидуализации.
3. Кодекс РФ об административных правонарушениях. (Раздел II. Особенная часть) [21] Декларирует административные нарушения и ответственность за них, в отношении следующих объектов:
   * авторских и смежных прав, изобретательских и патентных прав;
   * установленного порядка патентования объектов промышленной собственности в иностранных государствах;
   * непредставления первичных статистических данных;
   * незаконного использования средств индивидуализации товаров.
4. Налоговый кодекс РФ. (Часть вторая) [22] Содержит информацию об уплате государственной пошлины: размер, плательщики, порядок и сроки уплаты, льготы, основания и порядок возврата, особенности предоставления отсрочки или рассрочки уплаты пошлины. [Глава 25.3]
5. Уголовный кодекс РФ. (Главы 19 и 22) [23] Содержит описание уголовных правонарушений и ответственности в отношении:
   * нарушения авторских и смежных прав;
   * нарушения изобретательских и патентных прав;
   * незаконного использования средств индивидуализации товаров.

Среди законов общей нормативной базы:

1. Федеральный закон Российской Федерации «Об информации, информатизации и защите информации» от 25.10.95 г. Регулирует отношения, возникающие по поводу информации, в том числе несанкционированного ее использования. [24]
2. Федеральный закон Российской Федерации «О коммерческой тайне». Описывает понятия коммерческой тайны и мероприятия по охране информации, являющейся коммерческой тайной. [25]

## Описание процедуры подачи заявки на регистрацию объектов интеллектуальной деятельности

Поскольку объектом интеллектуальной собственности выпускной работы является программный код, нет необходимости производить государственную регистрацию данной собственности так как право на собственность возникает сразу после написания программного кода.

Проводить правовую защиту результатов работы в виде программного кода затруднительно, поскольку под защиту попадает сам исходный код программы, который не несет в себе пользы. Итоговая программа, полученная из исходного кода, не попадает под «авторское право», поэтому аналогичная программа, написанная на другом языке программирования не является нарушением авторского права. В таком случае доказывать факт нарушения «авторского права» необходимо защищать в судебном порядке, где необходимо доказать первенство возникновения «авторского права» и предоставить факты нарушения.

Для доказательства существования права программу следует зарегистрировать в Роспатенте. Для этого необходимо выполнить следующее:

1. Подготовить документы;
   1. Заявление о государственной регистрации программы для ЭВМ с указанием правообладателя, а также автора, если он не отказался быть упомянутым в качестве такового;
   2. Депонируемые материалы, идентифицирующие программу для ЭВМ или базу данных, включая реферат;
   3. Подтверждение согласия на обработку персональных данных субъектов персональных данных, указанных в заявлении о государственной регистрации программы для ЭВМ или базы данных;
   4. Документ, подтверждающий согласие автора на указание сведений об авторе, указанных в заявлении;
   5. Доверенность, подтверждающая полномочия представителя заявителя;
   6. Документ, подтверждающий уплату государственной пошлины в размере и порядке, предусмотренных Налоговым кодексом Российской Федерации.
2. Оплатить пошлину за государственную регистрацию;
   1. Для физических лиц в размере 3000 рублей;
   2. Для юридических лиц в размере 4500 рублей.
3. Подать документы в Роспатент через ФИПС, ЕГПУ или лично.

## Анализ возможного использования объекта интеллектуальной деятельности

Реализованный алгоритм генерирует возможные траектории в динамических системах, которые должны подаваться на вход управляющему процессу автономного агента. На основании траектории, робот должен строить стратегический маршрут движения, который будет уточняться за счет одометрии и лидаров. Данная траектория будет характерной для данной среды, например если модель обучается на движении складских роботов, то траектории движения будут аналогичны движению погрузчиков, если алгоритм обучился на движении человека, то траектория будет похожей на человеческую. Данная способность алгоритма походить на других агентов среды обеспечивает возможность эффективно перемещаться в пространстве.

В качестве вероятного примера использования рассматривается система роботизированного склада. Такие системы отказываются от использования человеческой рабочей силы в пользу роботизированных агентов. Данный подход позволяет экономить на оплате труда, поддерживать режим работы 24/7, практически не имеет простоев, исключает опасность получения травм или гибели сотрудников. В данный момент роботы перемещаются по складу при помощи алгоритмов, в которых зашиты допустимые маршруты. Такой подход создает простои в работе всей системы, когда разрешенные маршруты заблокированы. Реализованный алгоритм способен самостоятельно генерировать траектории в динамической системе, что решает данную проблему и сокращает количество и время простоев.

Возьмем для примера компанию METRAROBOTICS, которая предоставляет услуги по комплексной роботизации складских помещений. Стоимость услуг за год работы для среднего склада составляет 6000000 рублей, при этом по статистике система простаивает до получаса каждые сутки. Таким образом за год 182.5 часа простоя. Стоимость часа простоя составляет 685 рублей/час. Таким образом за год работы система простаивает на . Разработанное решение позволяет избавиться от простоя и сэкономить данную сумму. Затраты на внедрение алгоритма в систему не требуется, поскольку внедрение осуществляется за счет обновления системы.

## Оценка рыночной стоимости объекта интеллектуальной деятельности

Для оценки рыночной стоимости объекта интеллектуальной деятельности используется метод капитализации доходов, основанный на следующей формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (6.1) |

где: – текущая стоимость будущих доходов (рыночная стоимость объекта оценки);

Д – прогнозируемый годовой чистый доход;

К – коэффициент капитализации, определяемый как разница между ставкой дисконтирования (R) и прогнозным приростом дохода (g).

Ставка дисконтирования R рассчитывается следующим образом:

|  |  |  |
| --- | --- | --- |
|  | , | (6.2) |

где: – ключевая ставка Центрального Банка Российской Федерации (текущее значение);

– риск внедрения в производство товаров/услуг;

– риск неудачного продвижения продукта (товара/услуг) на рынок;

– риск низкой влиятельности при распределении совокупных выгод от реализации товара/услуг;

– риск высоких расходов, неэффективных инвестиций.

На момент осуществления расчетов (11.05.2021 г.) ключевая ставка Центрального Банка Российской Федерации составляет 5%. [26]

Риск внедрения в производство складывается из показателей, представленных в таблице табл. 6.1.

Таблица 6.1 – Факторы риска внедрения в производство

|  |  |  |
| --- | --- | --- |
| Фактор | Характеристика | Значение |
| Известность имени в нескольких отраслях | Неизвестное имя | 5% |
| Опыт применения ОИС в производстве | Нет | 5% |
| Наличие научной известности | Есть научные труды | 4% |

Исходя из представленных значений показатель риска внедрения в производство составляет:

|  |  |  |
| --- | --- | --- |
|  |  | (6.3) |

Риск неудачного продвижения продукта на рынок складывается из показателей, представленных в таблице табл. 6.2.

Таблица 6.2 – Факторы риска неудачного продвижения продукта на рынок

|  |  |  |
| --- | --- | --- |
| Фактор | Характеристика | Значение |
| Степень разработки | НИР, НИОКР | 4% |
| Степень коммерциализации в бизнесе | Начальная стадия | 4% |
| Наличие и объемы рынка | Несколько рынков | 2% |
| Рекламная кампания | Не ведется | 5% |
| Маркетинговая политика в части реализации продукции | Отсутствие плана реализации | 5% |

Исходя из представленных значений показатель риска неудачного продвижения продукта на рынок составляет:

|  |  |  |
| --- | --- | --- |
|  |  | (6.4) |

Риск низкой влиятельности при распределении совокупных выгод от реализации товара/услуг складывается из показателей, представленных в таблице табл. 6.3.

Таблица 6.3 – Факторы риска низкой влиятельности при распределении совокупных выгод от реализации товара/услуг

|  |  |  |
| --- | --- | --- |
| Фактор | Характеристика | Значение |
| Степень использованности ОИС в отдельных элементах товарной продукции | Использование в одном основном элементе | 3% |
| Стратегия ценообразования продукта с ОИС | В стадии разработки | 4% |
| Наличие лицензий | Нет | 5% |
| Менеджмент | Отсутствие специализированного менеджмента | 5% |
| Надежность участников проекта | Общие интересы участников проекта | 2% |

Исходя из представленных значений показатель риска низкой влиятельности при распределении совокупных выгод от реализации товара/услуг составляет:

|  |  |  |
| --- | --- | --- |
|  |  | (6.5) |

Риск высоких расходов, неэффективных инвестиций складывается из показателей, представленных в таблице табл. 6.4.

Таблица 6.4 – Факторы риска высоких расходов, неэффективных инвестиций

|  |  |  |
| --- | --- | --- |
| Фактор | Характеристика | Значение |
| Опыт применения аналогичных ОИС на предприятии | Нет | 5% |
| Стратегия ценообразования продукта с ОИС | Нет | 5% |
| Наличие на рынке конкурентного контрафакта | Есть в небольшом количестве | 3% |
| Наличие на рынке конкурентных ОИС | Среднеконкурентные ОИС | 4% |

Исходя из представленных значений показатель риска высоких расходов, неэффективных инвестиций составляет:

|  |  |  |
| --- | --- | --- |
|  |  | (6.6) |

Уровень инфляции в России за 2020 год составил 4.91%. [27]

Учитывая все определённые показатели, текущая стоимость будущих доходов складывается из (6.1), (6.2), (6.3), (6.4), (6.5), (6.6) и составляет:

## Выводы

В ходе работы, проделанной в рамках оценки и защиты результатов интеллектуальной деятельности был определен объект интеллектуальной собственности, полученный в результате выпускной квалификационной работы. Модель, полученная в результате выполнения выпускной работы, является результатом интеллектуальной деятельности, которая защищается «авторским правом». Была рассчитана прогнозируемая годовая доходность объекта интеллектуальной собственности равная 124830 руб. Текущая стоимость будущих доходов предприятия, использующего данный объект, составляет

# ЗАКЛЮЧЕНИЕ

В рамках проделанной работы был разработан и исследован алгоритм для генерации траекторий в статической и динамической среде.

Перед реализацией работы был проведен обзор и анализ существующих подходов к построению траекторий и разобраны алгоритмы, которые реализуют данные подходы. Обзор аналогов показал, что существует две разновидности подходов к построению траекторий: с использованием классических алгоритмов на графах, а также с использованием алгоритмов с моделями машинного обучения. Из каждого класса алгоритмов были выбраны наиболее соответствующие требованиям решения. В качестве алгоритма из класса классических был выбран алгоритм A\*, который после незначительных доработок можно применять для генерации маршрутов. Из семейства алгоритмов с моделями машинного обучения были выбраны алгоритмы использующие модели GAN, CVAE и KTM. Сопоставление отобранных аналогов поставленным критериям показало, что A\* не подходит для работы в динамической среде из-за необходимости модификации графа, GAN и CVAE не подходят из-за низкой стабильности и требований к большому количеству обучающих данных, а модель KTM, хотя и строит траектории близкие к реальным, не способна работать в динамической среде так как строит предсказания только на основе наблюдаемых траекторий. Таким образом была подтверждена актуальность работы.

Реализация модели OCTNet для статических данных требовала наличие обучающей выборки, в качестве которой был выбран набор данных Occ-Traj120. В наборе содержатся карты и соответствующие им траектории. При помощи поворота, количество карт и траекторий для обучения было увеличено в 4 раза. На модифицированных данных была обучена статическая модель. При обучении потребовалось модифицировать функцию ошибки таким образом, чтобы ее вычисление не приводило к увеличению данных до , или уменьшению до . После обучения сети на протяжении 20 эпох, ошибка снизилась до допустимого уровня. Были построены графики процесса обучения, для проверки отсутствия недообучения или переобучения. На основании графиков был сделан вывод о корректности обученной модели. Продемонстрирована генерация траекторий по обученной модели.

Для реализации динамической модели OCTNet требовались видео данные карт и изменяющихся траекторий. Из-за отсутствия подобных данных, был реализован генератор динамических карт. Генератор работает по принципу наложения фильтра преград на карту и отсечения траекторий, которые пересекаются с фильтром. Путем смещения фильтра с отсечением пересечений на каждом шаге получается несколько кадров, которые описывают одну динамическую карту. По полученным динамическим картам была обучена модель OCTNet с изменением размерностей и функции ошибок. Было выявлено ограничение сети, которое заключается в том, что для динамических данных модель генерирует допустимые траектории только в известном диапазоне смещения препятствий. Если на вход сети приходит ранее неизвестная карта со знакомым движением преграды – сеть справляется с задачей, но в случае, когда даже на известной карте преграда смещается дальше изученного диапазона, сеть стремительно теряет точность. Данная проблема не была решена в рамках данной работы в полной мере. Одним из вариантов решения проблемы является обучение модели на полном диапазоне движущихся объектов, что восстанавливает предиктивные свойства модели. В качестве второго подхода рассматривается создание LSTM сети, на вход которой подается результат работы MDN сети. На основании архитектуры рекуррентных сетей была обучена модель, которая способна предсказывать следующий кадр по предыдущим. Был сделан поход к обучению сети, но из-за высоких требований к ресурсам, которые необходимы для обучения сеть была обучена на протяжении 9 эпох, что в несколько сотен раз меньше необходимого. Однако, после подачи смешения фильтров на вход обученной сети, модель смогла предсказать следующий кадр с сильным размытием, что дает повод рассматривать такой подход в дальнейшем.

После реализации статической и динамической модели было проведено сравнение характеристик системы с отобранными ранее аналогами. Результаты сравнительной характеристики показали, что разработанный алгоритм превосходит аналоги в задаче генерации траекторий, которые являются естественными для среды.

Дальнейшими шагами по развитию алгоритма генерации траекторий являются приспособление алгоритма к трехмерным пространствам, полноценное обучение LSTM сетей и проверка качества их работы, генерация более обширного набора данных со средами различных характеристик, а также генерация обширной базы динамических карт.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Jianmin B, Dong C, Fang W, Houqiang L , Gang H CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training // arXiv. 2018;
2. Weiming Z, Tin L, Lionel O, Gilad F, Fabio R OCTNet: Trajectory Generation in New Environments from Past Experiences // arXiv. 2019;
3. Nicolas V, Oleksandr Z, Theodoros T, Priya G, Zachary D, William S, Sven V, Andrew A, Albert C, Tensor Comprehensions: Framework-Agnostic High-Performance Machine Learning Abstractions // arXiv. 2018;
4. T. Lai, W. Zhi, and F. Ramos, “Occ-traj120: Occupancy maps with associated trajectories,” CoRR, 2019;
5. Yaser S. A., Malik M.I. Learning From Data. Hardcover. 2011;
6. Дашкевич А.А. Исследование моделей сверточных автоэнкодеров для выделения признаков в наборах стереоизображений // Вестник НТУ ХПИ.2017. №50 (1271). URL: https://cyberleninka.ru/article/n/issledovanie-modeley-svertochnyh-avtoenkoderov-dlya-vydeleniya-priznakov-v-naborah-stereoizobrazheniy (дата обращения: 01.05.2021);
7. Кожевников М.М., Чумаков О.А., Илюшин И.Э., Господ А.В. Генерация программных траекторий для роботов-манипуляторов точечной контактной сварки в среде с препятствиями // Доклады БГУИР. 2017. №7 (109). URL: https://cyberleninka.ru/article/n/generatsiya-programmnyh-traektoriy-dlya-robotov-manipulyatorov-tochechnoy-kontaktnoy-svarki-v-srede-s-prepyatstviyami (дата обращения: 11.05.2021);
8. Введение в Генеративно-состязательные сети (GAN – Genereative Adversarial Networks). Id-lab. URL: <https://id-lab.ru/posts/developers/vvedenie-v-generativno-sostyazatelnye-seti-gan-genereative-adversarial-networks/> (дата обращения 11.03.2021);
9. Баймухамедов М.Ф., Баймухамедова А.М. Обучение ядерных нейронных сетей // АВУ. 2016. №11 (153). URL: https://cyberleninka.ru/article/n/obuchenie-yadernyh-neyronnyh-setey (дата обращения: 24.04.2021);
10. Z. Wang, B. Shakibi, L. Jin, and N. de Freitas. Bayesian multi-scale optimistic optimization. In *AISTATS*, 2014;
11. Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa. Functional gra- dient motion planning in reproducing kernel hilbert spaces. In Robotics: Science and Systems, 2016;
12. Николенко С, Кадурин А, Архангельская Е Глубокое обучение. Погружение в мир нейронных сетей. СПб. Питер, 2018;
13. Марсикано К., Стюарт К., Филлипс Б. Android. Программирование для профессионалов. Питер, 2017;
14. Нишант Ш Машинное обучение и TensorFlow. СПб.: Питер, 2018;
15. Гудфеллоу Я., Бенджио И., Курвилль А. Глубокое обучение. СПб.: ДМК Пресс, 2018;
16. Фрвансуа Ш Глубокое обучение на Python. СПб.: Питер, 2018;
17. Интеллектуальная собственность в международном праве. // Юридический справочник URL: https://jurlib.blogspot.com (дата обращения: 11.05.2021);
18. Интеллектуальная собственность. // Правовое обеспечение. URL: <https://zakon.ru> (дата обращения: 11.05.2021).;
19. Гражданский кодекс Российской Федерации (часть четвертая)" от 18.12.2006 N 230-ФЗ (ред. от 30.12.2020) (с изм. и доп., вступ. в силу с 17.01.2021) // КонсультантПлюс: справочно-правовая система [Офиц. сайт]. URL: http://www.consultant.ru/ (дата обращения 11.05.2021);
20. Конституция Российской Федерации" (принята всенародным голосованием 12.12.1993 с изменениями, одобренными в ходе общероссийского голосования 01.07.2020) // КонсультантПлюс: справочно-правовая система [Офиц. сайт]. URL: http://www.consultant.ru/ (дата обращения 11.05.2021);
21. Кодекс Российской Федерации об административных правонарушениях от 30.12.2001, № 195-ФЗ. // Роспатент Федеральная служба по интеллектуальной собственности URL: <https://rospatent.gov.ru/ru/documents/kodeks-rf-ob-administrativnyh-ravonarusheniyah-izvlecheniya/download> (дата обращения 11.05.2021);
22. Налоговый кодекс Российской Федерации (Часть вторая). От 05.08.2000, № 117-ФЗ. // Роспатент Федеральная служба по интеллектуальной собственности URL: https://rospatent.gov.ru/ru/documents/nalogovyy-kodeks-rossiyskoy-federacii-chast-vtoraya (дата обращения 11.05.2021);
23. Уголовный кодекс Российской Федерации от 13.06.1996, № 63-ФЗ. // Роспатент Федеральная служба по интеллектуальной собственности URL: https://rospatent.gov.ru/ru/documents/ugolovnyy-kodeks-rf-izvlechenie/download (дата обращения 11.05.2021);
24. Федеральный закон от 27.07.2006 N 149-ФЗ (ред. от 09.03.2021) "Об информации, информационных технологиях и о защите информации" (с изм. и доп., вступ. в силу с 20.03.2021) // КонсультантПлюс: справочно-правовая система [Офиц. сайт]. URL: http://www.consultant.ru/ (дата обращения 11.05.2021);
25. Федеральный закон от 29.07.2004 N 98-ФЗ (ред. от 09.03.2021) "О коммерческой тайне" // КонсультантПлюс: справочно-правовая система [Офиц. сайт]. URL: http://www.consultant.ru/ (дата обращения 11.05.2021);
26. Ключевая ставка Банка России // Центральный банк Российской Федерации URL: <https://www.cbr.ru/hd_base/KeyRate/> (дата обращения 11.05.2021);
27. Инфляция в России // Уровень инфляции в России URL: <https://уровень-инфляции.рф> (дата обращения 11.05.2021).

# ПРИЛОЖЕНИЕ А

**Код генерации матрицы признаков**

def get\_hausdorff\_matrix(data):

data\_map = [d.map for d in data]

X\_map\_arr = []

l = 50

for i, m\_i in enumerate(data\_map):

X\_map\_arr.append([])

for j, m\_j in enumerate(data\_map):

d\_1 = directed\_hausdorff(m\_i, m\_j)[0]

d\_2 = directed\_hausdorff(m\_j, m\_i)[0]

d = (d\_1 + d\_2)/(2)

s = np.exp(-((d\*\*2)/(2\*l)))

X\_map\_arr[i].append(s)

return np.array(X\_map\_arr)

# ПРИЛОЖЕНИЕ Б

**Код встраивания траекторий в векторы фиксированной длины**

def k(tau, length):

b = 5

radial\_taus = np.array([(i+1)/length for i in range(length)])

input\_taus = np.array([tau for i in range(length)])

return np.exp(-((np.abs(input\_taus - radial\_taus))/(2\*b))).reshape(1, length)

def embed\_trajs(data, comp=COMPONENTS, lmbd=0):

result = []

for d in data:

maps = []

for t in d.trajectories:

taus = np.array([(i+1)/len(t.x) for i, \_ in enumerate(t.x)])

k\_k = LA.inv(lmbd\*np.eye(comp) + reduce(lambda x,y: x + y, [np.matmul(k(tau,comp).T, k(tau,comp)) for tau in taus]))

x\_k = reduce(lambda x,y: x + y, [ t.x[i] \* k(tau,comp) for i, tau in enumerate(taus)])

w\_x = k\_k.dot(x\_k.T)

y\_k = reduce(lambda x,y: x + y, [ t.y[i] \* k(tau,comp) for i, tau in enumerate(taus)])

w\_y = k\_k.dot(y\_k.T)

w\_con = np.concatenate((w\_x, w\_y), axis=None)

maps.append(w\_con)

result.append(np.array(maps))

return np.array(result, dtype=object)

# ПРИЛОЖЕНИЕ В

**Код генерации траекторий по предсказанию модели**

for entry in X\_stand[:1]:

input1 = np.array([entry])

prediction = model.predict(input1[:1])

alph, mu, sig = slice\_parameter\_vectors\_exp(prediction, 5)

alph = alph.reshape(-1)

mu = mu.reshape(-1, 10)

sig = sig.reshape(-1, 10)

t = np.arange(0, 10)

w\_pred\_xs = []

w\_pred\_ys = []

for i in range(100):

randNum = randNum=np.random.uniform(low=0.0, high=1.0)

best = 0

cumulativeVal = 0

cumulativeValUpper = 0

for i in range(len(alph)):

cumulativeValUpper+=alph[i]

if((randNum>cumulativeVal)and (randNum<=cumulativeValUpper)):

best=i

cumulativeVal+=alph[i]

changed\_mu = mu[best]

changed\_sig = sig[best]

changed\_mu\_x = changed\_mu[:5]

changed\_mu\_y = changed\_mu[5:]

changed\_sig\_x = changed\_sig[:5]

changed\_sig\_y = changed\_sig[5:]

w\_pred\_x = []

w\_pred\_y = []

for i, m in enumerate(changed\_mu\_x):

res\_x = (m + laplace.rvs(loc=0, scale=changed\_sig\_x[i]))\*max\_w

w\_pred\_x.append(res\_x)

for i, m in enumerate(changed\_mu\_y):

res\_y = (m + laplace.rvs(loc=0, scale=changed\_sig\_y[i]))\*max\_w

w\_pred\_y.append(res\_y)

w\_pred\_xs.append(w\_pred\_x)

w\_pred\_ys.append(w\_pred\_y)

w\_pred\_xs\_np = np.array(w\_pred\_xs)

w\_pred\_ys\_np = np.array(w\_pred\_ys)

# ПРИЛОЖЕНИЕ Г

**Код генерации динамических карт**

def filter\_to\_shapes(filt):

vert = []

for i\_r, r in enumerate(filt):

if len(np.unique(r)) > 1:

vert.append(i\_r) #add top and bottom

filt\_t = filt.T

for i\_r, r in enumerate(filt\_t):

if len(np.unique(r)) > 1:

vert.append(i\_r) #add left and right

#vert is [top, bottom, left, right]

vert[0] = vert[0] - 1

vert[1] = vert[1] + 1

vert[2] = vert[2] - 1

vert[3] = vert[3] + 1

return np.array([

[

[vert[2], vert[1]],

[vert[2], vert[0]]

], #left

[

[vert[2], vert[0]],

[vert[3], vert[0]]

], #top

[

[vert[3], vert[0]],

[vert[3], vert[1]]

], #right

[

[vert[3], vert[1]],

[vert[2], vert[1]]

] #bottom

])

def is\_dot\_in\_shape(x, y, shape):

mp1 = [(shape[0][0] - x), (shape[0][1] - y)]

mp2 = [(shape[1][0] - x), (shape[1][1] - y)]

return np.dot(mp1, mp2) <= 0

def line\_intersection(line1, line2):

xdiff = (line1[0][0] - line1[1][0], line2[0][0] - line2[1][0])

ydiff = (line1[0][1] - line1[1][1], line2[0][1] - line2[1][1]) #Typo was here

def det(a, b):

return a[0] \* b[1] - a[1] \* b[0]

div = det(xdiff, ydiff)

if div == 0:

return False

d = (det(\*line1), det(\*line2))

x = det(d, xdiff) / div

y = det(d, ydiff) / div

return is\_dot\_in\_shape(x,y,line1) and is\_dot\_in\_shape(x,y,line2)

def trajectory\_to\_shapes(trajectory):

x = trajectory.x

y = trajectory.y

lines = []

for i in range(len(x) - 1):

l\_x\_1 = x[i]

l\_x\_2 = x[i+1]

l\_y\_1 = y[i]

l\_y\_2 = y[i+1]

lines.append([[l\_x\_1, l\_y\_1], [l\_x\_2, l\_y\_2]])

return np.array(lines)

def is\_traj\_intersect\_filter(traj, filt):

result = False

for filt\_shape in filt:

for traj\_shape in traj:

inter = line\_intersection(filt\_shape, traj\_shape)

result = result or inter

return result

def generate\_maps(data, amount): #generate amount+amount\*20 maps and trajectories

generated = []

if amount > len(data):

amount = len(data)

for num in range(amount):

start\_time = time.time()

d = data[num]

map\_ini = np.array(d.map, dtype='int32')

traj\_ini = d.trajectories

generated.append(DataUnit(Map(map\_ini.tolist()), traj\_ini))

for step in range(20):

filt = move\_filter(dot\_filter(), step)

map\_mod = apply\_filter(map\_ini, filt)

traj\_shapes = []

for t in traj\_ini:

traj\_shapes.append(trajectory\_to\_shapes(t))

filter\_shapes = filter\_to\_shapes(filt)

traj\_mod = []

for i\_t, t in enumerate(traj\_shapes):

if not is\_traj\_intersect\_filter(t, filter\_shapes):

traj\_mod.append(traj\_ini[i\_t])

generated.append(DataUnit(Map(map\_mod.tolist()), traj\_mod))

print('{}/{} ({} s)'.format((num+1), amount, (time.time() - start\_time)))

return generated