

ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ ВО НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

УДК 51-76

Отчет о программном проекте на тему:
Программные средства для моделирования обучения в лабиринте

Выполнили:

студент группы БПМИ209
Антонова Евгения Александровна

(подпись)

(дата)

студент группы БПМИ209
Наумова Евгения Ильинична

(подпись)

(дата)

Принял руководитель проекта:

Чернышев Всеволод Леонидович
Доцент
Факультета компьютерных наук НИУ ВШЭ /
Департамент больших данных и информационного поиска

(подпись)

(дата)

Москва 2023

Содержание

Аннотация	3
Ключевые слова	4
1 Введение	5
1.1 Актуальность	5
1.2 Новизна	6
1.3 Распределение задач	7
2 Обзор литературы	9
3 Модель машинного обучения	10
3.1 Предобработка данных	10
3.2 Обучение модели	12
3.3 Анализ полученных результатов	15
4 Графический интерфейс	19
4.1 Макеты для графического интерфейса	20
4.2 Реализация демонстрирования маршрутов животного	23
4.3 Описание использования	25
5 Заключение	29
Список литературы	30
Приложения	31

Аннотация

Данная работа посвящена обучению нейронной сети, которая способна генерировать маршруты прохождения лабиринта животными на основе реальных данных. В ходе исследования был проанализирован набор данных, содержащий информацию о маршрутах движения животных в лабиринтах, и разработан алгоритм обучения нейронной сети на этом наборе данных.

Кроме того, в работе представлена реализация графического интерфейса, который позволяет визуализировать процесс обучения животного и демонстрировать полученные результаты. Графический интерфейс разработан на основе современных технологий программирования и обеспечивает удобный доступ к настройкам и параметрам обучения, а также позволяет сохранять полученные результаты для дальнейшего анализа.

Результаты исследования показали, что обученная нейронная сеть способна генерировать эффективные маршруты прохождения лабиринта животными на основе реальных данных, что может быть полезным для дальнейших исследований в области поведения животных в условиях лабиринта.

Ключевые слова

- *Дискретный оператор* — это некоторая последовательность действий, которая повторяется с некоторой частотой (в результате обучения животного), и которую мы будем реализовывать с помощью алгоритма.
- *Графический интерфейс* — это система средств для взаимодействия пользователя с программой, созданная для визуализации всех необходимых объектов и функций в виде графических компонентов экрана.
- *Модель машинного обучения* — это файл, который использует методы искусственного интеллекта и обучен решать задачи за счёт применения знаний о решениях подобных задач, а также выявлять различные типы закономерностей.
- *Нейронная сеть* — математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма.
- *Трансформеры* — это тип нейронных сетей, который направлен на решение последовательностей с обработкой зависимостей.
- *Глубинное обучение* — это раздел искусственного интеллекта, который является частью более широкого семейства методов машинного обучения, которое основано на искусственных нейронных сетях с обучением представлению.

1 Введение

В последнее время было проведено большое количество исследований на темы, связанные с работой мозга и поведением животных в неизвестных для них средах. Научное сообщество хочет изучить вопрос принятия решений и научиться моделировать поведение различных типов животных с помощью программных средств.

Уже проводилось немало исследований на данную тему, изучались процессы поведения грызунов, рыб, обезьян и людей. В частности были предприняты попытки симулировать процесс поведения объекта, основанные на применении дискретных операторов на графе.

Не вызывает сомнений актуальность исследований искусственного интеллекта, а эта сфера развивается с огромной скоростью, и модели, созданные людьми, могут повторять довольно сложные закономерности и использовать их.

В отличие от предыдущих работ, идея нашего проекта заключается в большей автоматизации моделирования.

Есть предположение, что мозг при выборе действий также действует по некоторым правилам, которые пока что сложно перенести в компьютерный вид. Удачная модель, которая сможет близко к реальности предсказывать действия животного, поможет понять устройство мозга, а также даст возможность искусственно создавать эксперименты.

В случае успешного выполнения, наша работа будет полезна для исследования интеллектуальной деятельности различных объектов, а также в качестве инструмента, который будет экономить большое количество времени и снизит затраты на исследования. Более того, анализировать результаты экспериментов будет гораздо проще и удобнее, поскольку эти результаты будут появляться сразу на компьютере.

1.1 Актуальность

Создание модели машинного обучения для обучения животных в лабиринте и соответствующего графического интерфейса для демонстрации результатов имеет несколько актуальных причин.

Например, модель машинного обучения позволяет изучать и анализировать поведение животных в лабиринте. Такие исследования могут пролить свет на множество аспектов поведения животных, их способности к обучению, принятию решений и адаптации к новым ситуациям.

Также создание модели машинного обучения для животных в лабиринте может помочь разработать новые методы обучения и тренировки животных. Модель может помочь

идентифицировать наиболее эффективные стратегии и тактики, которые могут быть применены в реальных условиях обучения животных. Результаты исследований, проведенных с помощью модели машинного обучения, могут быть применены на практике. Например, они могут помочь в обучении служебных животных, тренировке лабораторных животных или улучшении понимания поведения диких животных в их естественной среде обитания.

С экономической точки зрения создание модели машинного обучения и графического интерфейса для обучения животных может привести к экономическим выгодам. Например, модель может быть использована для оптимизации процесса тренировки животных, что может снизить затраты на время и ресурсы, необходимые для обучения.

Также стоит вспомнить о популяризации науки, так как создание графического интерфейса для демонстрации результатов может помочь популяризировать исследования и научные достижения в области поведения животных. Люди могут визуально представить себе модель обучения животных и преодолевания лабиринтов, что может вызвать интерес у широкой аудитории.

Таким образом, актуальность сокращенно можно описать следующими пунктами:

- Исследования поведения животных имеют важное значение для понимания их когнитивных и психологических возможностей. Модель машинного обучения в лабиринте позволяет исследователям изучать и анализировать поведение животных в контролируемой среде, предоставляя новые данные и понимание их принятия решений.
- Поведенческие исследования животных могут помочь в разработке новых методов тренировки и улучшения адаптивности животных, что имеет применение в множестве областей, включая медицину, зоологию и сельское хозяйство.
- Реализация графического интерфейса поможет визуализировать результаты, которые были получены при помощи модели машинного обучения, а также показать реальные маршруты животных в лабиринтах.

1.2 Новизна

Модель машинного обучения была специально создана для разработки методов тренировки животных и анализа их поведения в контролируемой среде лабиринта.

Основной принцип работы модели состоит в том, что она использует машинное обучение для анализа и интерпретации данных, полученных от животных во время прохождения лабиринта. В начале обучения модель собирает данные о движении в лабиринте. Эти данные

затем обрабатываются и использованы для создания модели машинного обучения, которая может предсказывать будущее поведение животного на основе текущего контекста.

Одна из основных новаций этой модели заключается в использовании глубоких нейронных сетей, которые позволяют модели анализировать сложные шаблоны и зависимости в поведении животных. Также модель может учитывать контекстуальную информацию.

Для демонстрации результатов обучения и работы модели был разработан графический интерфейс. Графический интерфейс предоставляет пользователю удобный способ взаимодействия с моделью, отображая информацию о прохождении лабиринта животными и позволяя настраивать параметры обучения. Он также позволяет пользователю наблюдать за поведением животного в режиме реального времени и анализировать полученные данные.

Важно отметить, что эта модель и графический интерфейс имеют потенциал применения в различных областях, включая поведенческие исследования животных, разработку методов тренировки и повышения адаптивности животных, а также в области робототехники, где подобные методы могут быть использованы для разработки автономных систем навигации.

Таким образом, новизну сокращенно можно описать следующими пунктами:

- Создание модели машинного обучения для обучения животных в лабиринте представляет собой новый подход к изучению поведения животных. Эта модель комбинирует принципы машинного обучения, анализа данных и психологии животных, что открывает новые возможности для исследований.
- Разработка графического интерфейса для демонстрации результатов обучения является новым способом визуализации данных и взаимодействия с моделью. Он улучшает доступность и понятность результатов, делая их более доступными для широкой аудитории.

Поэтому необходимо понять возможно ли смоделировать поведение различных типов животных в лабиринте с помощью моделей машинного обучения. Также необходимо реализовать графический интерфейс, который поможет наглядно увидеть результат работы данной программы и даст возможность экономить время на разборе полученных результатов.

1.3 Распределение задач

Определим задачи и ответственных за них участников:

За Антоновой Евгенией были закреплены следующие задачи:

- 1 Написать код для предобработки данных.

- 2 Реализовать графический интерфейс для демонстрации маршрута животного.

Соответствующие разделы — 4.

За Наумовой Евгений закрепились следующие задания:

- 1 Обучить модель для предсказания маршрута животного в лабиринте.

- 2 Создание выборки из результатов для демонстрации работоспособности кода.

Соответствующие разделы — 3.

Также в проекте имелись общие задачи:

- 1 Изучить наработки прошлых годов.

- 2 Улучшить полученную модель с помощью метрик машинного обучения.

2 Обзор литературы

На данный момент эта тема была глубоко разобрана с теоретической точки в статье «Структура внешней среды как системообразующий фактор условнорефлекторного процесса» Никольской К.А., Дидык Л.А., Серебряковой Е.Р. [5]. В ней разобраны различные типы животных и их поведение в лабиринте, а также биологическое обоснование данных процессов.

Также в процессе работы мы использовали наши наработки прошлых лет на подобные темы других студентов, а именно: «Эволюция поведения: моделирование поведения рыб в лабиринте с помощью дискретных операторов на графе» Антоновой Е. 2022 г., «Моделирование поведения крыс в лабиринте с помощью дискретных операторов на графе» Колб И.В. 2021 г., «Моделирование поведения крыс в лабиринте с помощью дискретных операторов на графе и кофеин» Копылова О.И. 2022 г., «Симулирование поведения мыши в лабиринте с помощью дискретных операторов на графе» Шитова Д.С. 2021 г., «Эволюция поведения: моделирование поведения обезьян в лабиринте с помощью дискретных операторов на графе» Наумовой Е.И. 2022 г.

Отличие всех имеющихся исследований от текущего в том, что они рассматривали причины появления некоторых моделей поведения животного, после чего пытались построить возможные маршруты. Наша идея заключалась в том, чтобы выбрать важные признаки и обработать данные для получения возможности их передачи модели машинного обучения, которая будет генерировать путь животного в лабиринте без необходимости нашего вмешательства, как это было ранее. Таким образом мы избавимся от необходимости выделения дискретных операторов (моделей поведения животного) вручную.

Как и в предыдущих исследованиях, мы рассматривали маршрут, по которому объект проходил лабиринт, в виде последовательности букв, где каждая отвечает за определенный отсек лабиринта.

Таким образом, на основе полученных данных мы сможем реализовать модель, которая будет генерировать возможные маршруты прохождения лабиринта разными типами животных.

3 Модель машинного обучения

Для начала, уделим внимание, почему мы решили перейти на обучение с помощью модели математического моделирования:

- 1 Гибкость и адаптивность: Модель машинного обучения способна обучаться на основе большого объема данных и адаптироваться к различным условиям и ситуациям. Она может находить скрытые закономерности и создавать более сложные стратегии прохождения лабиринта, чем те, которые могут быть заданы через дискретные операторы.
- 2 Повышение точности и эффективности: Модель машинного обучения может предоставить более точные и эффективные решения для генерации прохождения лабиринта. Она может учесть множество факторов, влияющих на прохождение, и оптимизировать свое поведение на основе полученной информации.

В целом, модель машинного обучения может предоставить более гибкий, адаптивный и эффективный подход к генерации прохождения лабиринта животным, в сравнении с реализацией через дискретные операторы.

3.1 Предобработка данных

Данным разделом занималась: Антонова Евгения.

Предобработка данных из лабиринта для машинного обучения включает в себя следующие шаги:

- 1 Загрузка данных: данные были представлены в формате CSV-файла и подгружены с помощью функции `pd.read_csv()`.
- 2 Очистка данных: данные могут содержать ошибки, пропущенные значения, выбросы и другие неточности. Эти неточности могут повлиять на качество обучения модели. Поэтому были произведены следующие действия:

2.1. Изначально данные были представлены в формате таблиц для различных лабиринтов с результатами прохождения их животными с указанием типа животного, номера опыта и эксперимента. Последовательности содержали буквы, соответствующие отсекам лабиринта, и знаки + и -, для обучения модели и понимания данных необходимо было удалить все знаки, не обозначающие отсеки лабиринта.

3 Преобразование данных: данные могут быть представлены в разных форматах, например, числовые значения могут быть представлены в виде строк. Поэтому необходимо преобразовать данные в соответствующий формат для машинного обучения.

3.1. В нашем случае, необходимо было разбить строку маршрута посимвольно, для того чтобы при токенизации каждой букве (каждому отсеку лабиринта) соответствовал отдельный токен.

4 Создание признаков: данные могут содержать информацию, которая может быть полезна для обучения модели. Например, в лабиринте могут быть определенные факторы, которые могут влиять на выбор пути. Поэтому необходимо создать новые признаки, которые будут учитывать эту информацию.

4.1. Поэтому мы создаем новый столбец из существующих, с информацией о типе животного, номера опыта и эксперимента.

5 Разделение данных: данные могут быть разделены на обучающую и валидационную выборки. Это позволит оценить качество обучения модели на валидационных данных, которые не использовались при обучении.

5.1. Мы разделили таблицу со всеми типами животных на разные, поскольку далее мы будем обучать модель на различных выборках для соответствующих типов. Также каждую из выборок поделили на обучающую и валидационную.

	input	path
0	fish 4 1	О О О О О И Л К А К Л Л К Ш Х Ц Щ З С Т
1	fish 4 2	О О О И И О О И К А Д С Р
2	fish 4 3	О М Л Л К Щ Ц Х Х Ц Щ З С С С Ж Л М О О И И И...
3	fish 5 1	О М М М М О О М К Г Щ Ц Ц Ц Х Ш А Ш Х Ц Ц Х Х...
4	fish 5 2	О М К Г Щ Ц Х Ш К Л Л К А Ч Ц Щ Г С С З Щ Ц Х...
...
29213	monkey 15 11	О М Я Г Я М И К А К И М Э Ж Т
29214	monkey 15 12	О И К А К И М Э Ж Т
29215	monkey 15 13	О И К А К И М Э Ж Т
29216	monkey 15 14	О И К А К И М Э Ж Т
29217	monkey 15 15	О И К К И М Я Г Я Э Ж Т

Рис. 3.1: Обработанные данные для модели.

3.2 Обучение модели

Данным разделом занималась: Наумова Евгения.

Сравнительный анализ:

Для дальнейшей работы, хотелось бы провести небольшой анализ вариантов возможной реализации, с помощью которой мы сможем добиться желаемых результатов.

Одним из вариантов была *SNN* [4] — спайковая нейронная сеть, которая. Но в процессе изучения данной модели, были выявлены несколько минусов для нашей задачи:

- 1 Вычислительные требования: Использование спайковых нейронных сетей требует высоких вычислительных ресурсов и по сравнению с традиционными нейронными сетями. Такие сети требуют моделирования динамики нейронов и передачи импульсов, что может потребовать большего объема вычислительных мощностей.
- 2 Ограниченная доступность данных: Обучение *SNN* требует большого объема данных о импульсах и динамике нейронов. Но в нашем проекте, это довольно затруднительно, поскольку количества данных для обучения мало.

Необходимо отметить, что спайковые нейронные сети также имеют свои преимущества, и их эффективность зависит от конкретной задачи и наличия соответствующих данных и ресурсов.

Еще один вариант, который был рассмотрен — *трансформеры* [2], в нашей задаче они имеют следующие преимущества:

- 1 Способность к обработке последовательностей: трансформеры хорошо подходят для работы с последовательностями данных, такими как серия действий, представляющих путь животного в лабиринте. Они могут моделировать зависимости между различными шагами и принимать во внимание контекст информации, что может быть полезным для определения наилучшего пути.
- 2 Гибкость и масштабируемость: трансформеры представляют собой гибкую архитектуру, которую можно настроить для различных задач. Они могут быть масштабированы на большие объемы данных и имеют потенциал для изучения сложных зависимостей в лабиринте, что может помочь модели разработать стратегию прохождения.
- 3 Самообучение: трансформеры могут использовать методы самообучения, такие как обучение с подкреплением или усиленное обучение, чтобы модель могла самостоятельно исследовать лабиринт и определить наиболее эффективные действия. Это позволяет модели обучаться на основе опыта и улучшать свои навыки с течением времени.

- 4 Обработка контекстной информации: трансформеры обладают способностью учиться на основе контекстной информации. Это означает, что модель может учитывать положение животного в лабиринте, информацию о препятствиях, цели и другие важные факторы при принятии решений. Это может помочь модели разрабатывать более интеллектуальные стратегии прохождения.

Обучение модели, реализация:

В данном разделе уделим внимание непосредственно реализации:

Сначала определены 3 модуля:

PositionalEncoding — модуль, который добавляет вложение позиций входных токенов в эмбединги. Он используется в модели для представления информации о позиции токена в последовательности. Это особенно важно для моделей, которые не имеют явного понятия о порядке токенов, таких как трансформер.

TokenEmbedding — модуль, который принимает тензор входных токенов и возвращает соответствующие эмбединги токенов. Он отображает каждый токен в векторное представление фиксированной размерности.

Input2PathTransformer — модуль, который использует архитектуру трансформера для кодирования и декодирования входных и выходных последовательностей. Он состоит из нескольких слоев энкодера и декодера, которые преобразуют последовательности токенов во вложения и обратно.

Для каждого модуля есть метод *init()*, который устанавливает все переменные и определяет архитектуру модели. Метод *forward()* определяет, как проходят данные через модель.

Также есть несколько вспомогательных функций:

generate_square_subsequent_mask — создает маску для предотвращения трансформера от обработки информации из будущих токенов. Она генерирует квадратную маску, в которой все элементы выше главной диагонали равны 0, а все элементы ниже диагонали равны 1. Это помогает модели рассматривать только предшествующую информацию при прогнозировании следующего токена.

create_mask — создает маски для исходных и целевых последовательностей, а также для заполнения (*padding*) для каждой последовательности. Она генерирует маску, в которой элементы, соответствующие пустым (*padding*) токенам, равны 0, а остальные элементы равны 1. Это позволяет модели игнорировать пустые токены при вычислении потерь и оценки модели.

sequential_transforms — принимает несколько функций, которые применяются по-

следовательно к текстовому вводу. Она удобна для применения нескольких преобразований к последовательности токенов.

tensor_transform — добавляет токены начала и конца последовательности (BOS и EOS) к тензору входных токенов. Это помогает модели определить начало и конец последовательности и корректно сгенерировать выход.

Train_epoch — В данном коде определена функция *train_epoch*, которая выполняет одну эпоху обучения модели *Input2PathTransformer*. Она принимает модель, оптимизатор, данные обучения и другие параметры, и обновляет веса модели, чтобы минимизировать потери на обучающем наборе данных.

Valid_epoch — В данной части кода определена функция *Valid_epoch*, которая выполняет оценку модели *Input2PathTransformer* на валидационном наборе данных.

Функция *Valid_epoch* принимает модель, пути к файлам с исходными и целевыми последовательностями ('*src_file*' и '*tgt_file*') и преобразование текста ('*text_transform*') и оценивает производительность модели, вычисляя потери и метрику качества.

Наконец, есть функция *plot_losses()*, которая отображает график потерь (*loss*) и перплексии (*perplexity*) для обучающих и проверочных примеров. Она используется для визуализации процесса обучения модели и оценки ее производительности.

3.3 Анализ полученных результатов

Данным разделом занималась: Наумова Евгения.

Также мы уделили внимание полученным результатам и сравнили их с нашей прошлой работой. Выбор метрик для обучения модели и оценки ее работы зависит от конкретной задачи и целей исследования. В случае обучения модели прохождения животным лабиринта выбор метрик обусловлен следующими соображениями:

- 1 Кросс-энтропия как функция потерь: кросс-энтропия является широко используемой функцией потерь для задач классификации, особенно когда речь идет о вероятностных моделях. В данном случае, если модель классифицирует действия животного в лабиринте, кросс-энтропия может быть хорошим выбором для измерения разницы между предсказанными и фактическими классами действий. Данную метрику мы использовали для обучения модели.
- 2 Метрика Левенштейна для оценки результатов: метрика Левенштейна измеряет минимальное количество операций (вставка, удаление, замена символов), необходимых для превращения одной строки в другую. В данном случае, если модель генерирует последовательность действий, метрика Левенштейна может быть использована для оценки сходства между сгенерированной последовательностью и ожидаемой последовательностью действий животного. Данную метрику мы использовали для окончательной оценки полученных результатов.

В результате мы получаем таблицу для каждого типа животного со столбцом, равным метрике Левейнштейна, следующего вида:

	input	path	generated	Lev
29002	monkey 3 1	ОМЭЖУЗГЗУЖЭЭЖУЗЗЗГЯЭЖУЗ...	ОИКАКИМЯГТ	24
29003	monkey 3 2	ОМЯГЯЗУЖЭМИЛЕР	ОИКАКИМЯГТ	13
29004	monkey 4 1	ЫОМЯГЗУЖЭЭУЗЯМИЛЕЕСДСЕБ...	ОИКАКИМЯГТ	30
29005	monkey 4 2	ОМЯЗУЖВВЭМИЛЕСДДСЕБЛИМВ...	ОИКАКИМЯГТ	26
29006	monkey 4 3	ОМЭВЖЖЭМИЛЕЕБЕР	ОИКАКИМЯГТ	13
...
29213	monkey 15 11	ОМЯГЯМИКАКИМЭЖТ	ОИКАКИМЯГТ	8
29214	monkey 15 12	ОИКАКИМЭЖТ	ОИКАКИМЭЖТ	1
29215	monkey 15 13	ОИКАКИМЭЖТ	ОИКАКИМЯГТ	3
29216	monkey 15 14	ОИКАКИМЭЖТ	ОИКАКИМЯГТ	3
29217	monkey 15 15	ОИККИМЯГЯЭЖТ	ОИКАКИМЭЖТ	5



Рис. 3.2: Результирующая таблица.

В отсортированном по столбцу с метрикой мы получаем такую таблицу:

	input	path	generated	Lev
28618	person 1 1	ЫОФХАДСШР	ЫОФХАДСШР	0
27629	person 1 1	ЫОИКАДСШР	ЫОФХАДСШР	2
27755	person 16 1	ЫОФХАХЦГЗУЩТ	ЫОФХАХЦГЗ	3
27758	person 19 1	ЫОФХАХЦГЗУЩТ	ЫОФХАХЦГЗ	3
27759	person 20 1	ЫОФХАХЦГЗУЩТ	ЫОФХАХЦГЗ	3
...
27768	person 3 1	ЫОФЦГЗУЩТЖВЭМЯГЦХАДСШЕБ...	ЫОФХАХЦГЗ	60
28421	person 1 1	ЫОИКАДСШРЕБЛИОЫОМЯГЗУЩТ...	ЫОФХАДСШР	90
28572	person 2 1	ЫОФХАХЦГЗУЩЖВЭМЯГЦФФХАК...	ЫОФХАХЦГЗ	95
28871	person 2 1	ЫОФХАДСШРЕБЛИКАДСШРЕБЛИ...	ЫОФХАХЦГЗ	109
28925	person 3 1	ЫОИКАДАХФОМЯГЗУЩЖВЭМОИИ...	ЫОФХАХЦГЗ	170

Рис. 3.3: Результирующая таблица. Отсортированная.

И результаты обучения в виде графиков:

Training 15/15: 100%  170/170 [00:40<00:00, 4.60it/s]
 Validation: 100%  43/43 [00:04<00:00, 12.21it/s]
 Epoch: 15, Train loss: 0.724, Val loss: 0.748, Epoch time = 40.594s

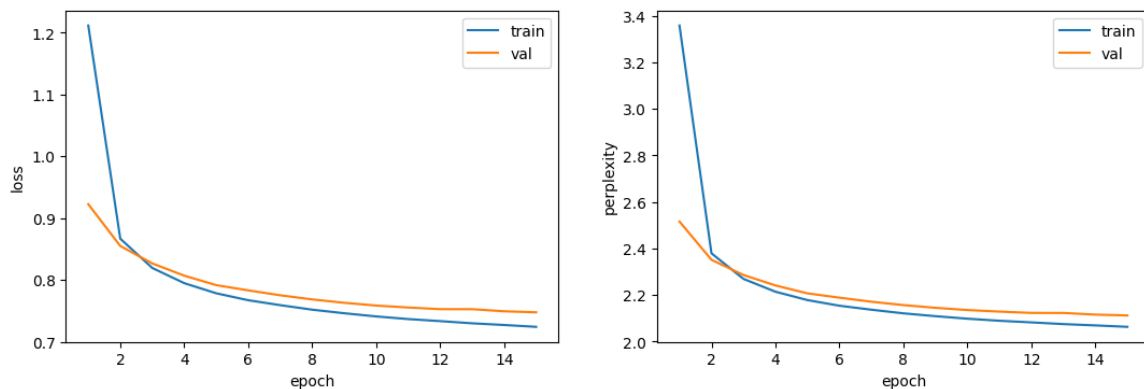




Рис. 3.4: График потерь. Крысы.

Training 15/15: 100%  10/10 [00:01<00:00, 5.97it/s]
 Validation: 100%  3/3 [00:00<00:00, 10.48it/s]
 Epoch: 15, Train loss: 0.533, Val loss: 0.611, Epoch time = 1.673s

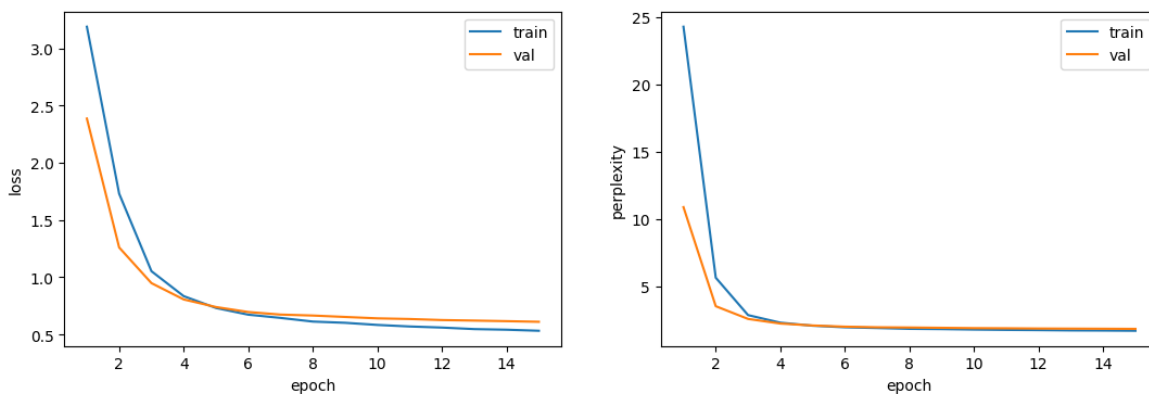




Рис. 3.5: График потерь. Люди.

Training 15/15: 100%  3/3 [00:00<00:00, 2.80it/s]
 Validation: 100%  1/1 [00:00<00:00, 14.07it/s]
 Epoch: 15, Train loss: 1.594, Val loss: 1.712, Epoch time = 0.764s

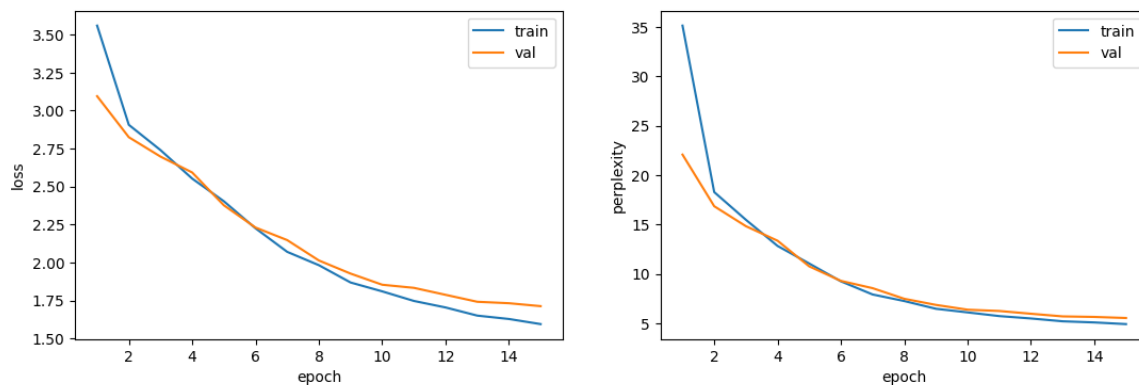


Рис. 3.6: График потерь. Рыбы.

Training 15/15: 100% 2/2 [00:00<00:00, 7.75it/s]
Validation: 100% 1/1 [00:00<00:00, 16.26it/s]
Epoch: 15, Train loss: 0.851, Val loss: 0.863, Epoch time = 0.204s

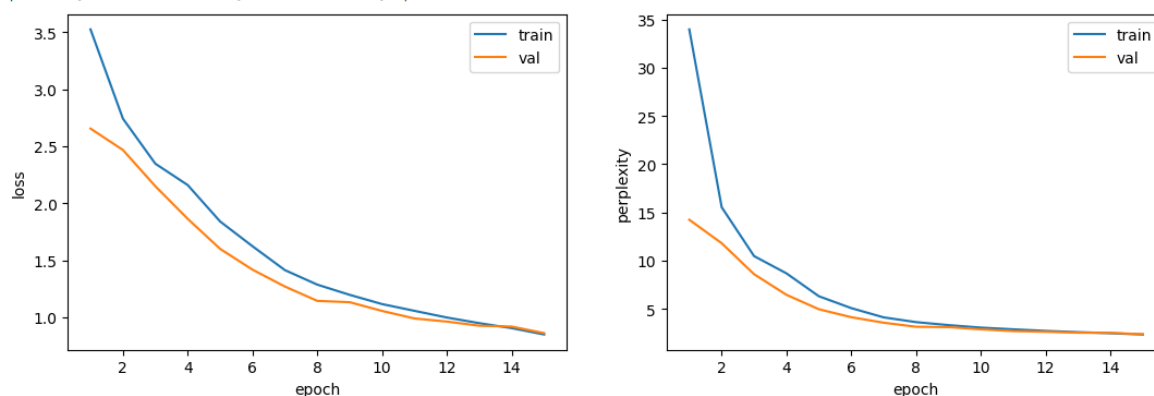


Рис. 3.7: График потерь. Обезьяны.

Таким образом, кросс-энтропия используется в процессе обучения модели для оптимизации ее параметров, а метрика Левенштейна используется для оценки качества работы модели и сравнения с ожидаемыми результатами. Такое комбинирование различных метрик полезно для получения более полного представления о работе модели и ее соответствии заданным целям.

В сравнении с прошлогодней работой наша программа продемонстрировала разные результаты в зависимости от типа животных. Для грызунов мы достигли минимального значения метрики Левенштейна, что указывает на высокую точность и успешное прохождение лабиринта. Это говорит о том, что модель машинного обучения хорошо справляется с анализом и навигацией в пространстве для этого конкретного вида животных.

Однако, в случае рыб и обезьян у нас недостаточно данных и, соответственно значимых результатов. Несмотря на это наша программа и графический интерфейс по-прежнему могут быть полезными инструментами для исследования поведения этих животных в будущих исследованиях, при условии дополнительного сбора и анализа данных.

Для людей наша программа достигла также высоких результатов. Мы располагали меньшим объемом данных в отличие от количества данных для крыс, однако всё равно получили довольно высокую точность.

Таким образом, можно заметить, что для обучения модели важным признаком является объем данных, так как при небольшой выборке невозможно с хорошей точностью получить результат.

4 Графический интерфейс

В нашей работе было уделено время для реализации визуализации прохождения животным различных лабиринтов. Всего для работы было представлено 3 лабиринта: Челнок, Никольской и Бережной.

Лабиринт Никольской использовался для 4 типов: грызунов, рыб, обезьян и людей:

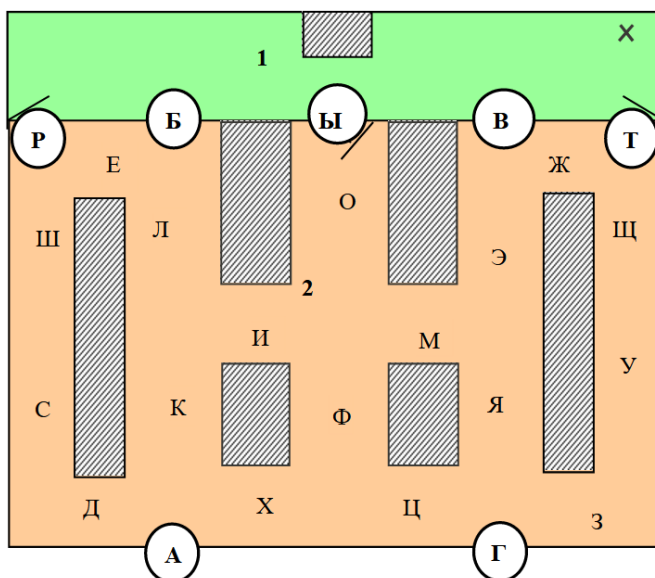


Рис. 4.1: Схема лабиринта. Никольская.

Лабиринт Бережной для грызунов:

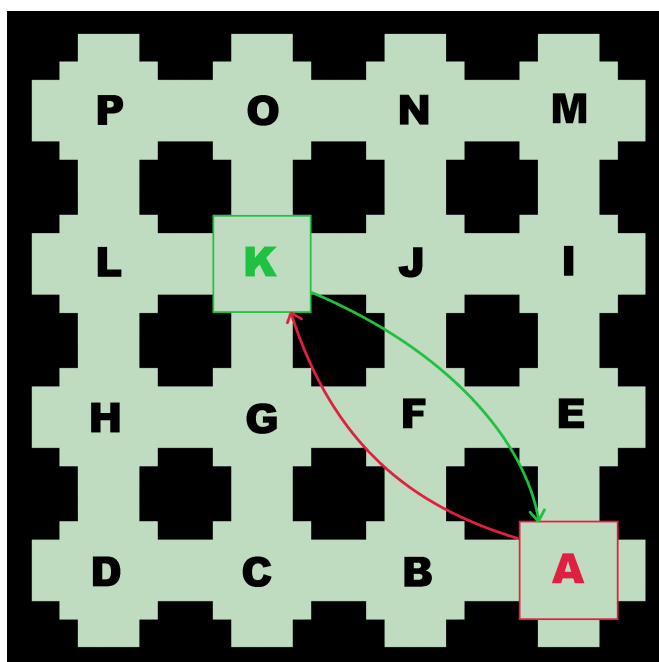


Рис. 4.2: Схема лабиринта. Бережной.

Лабиринт Челнок для грызунов:

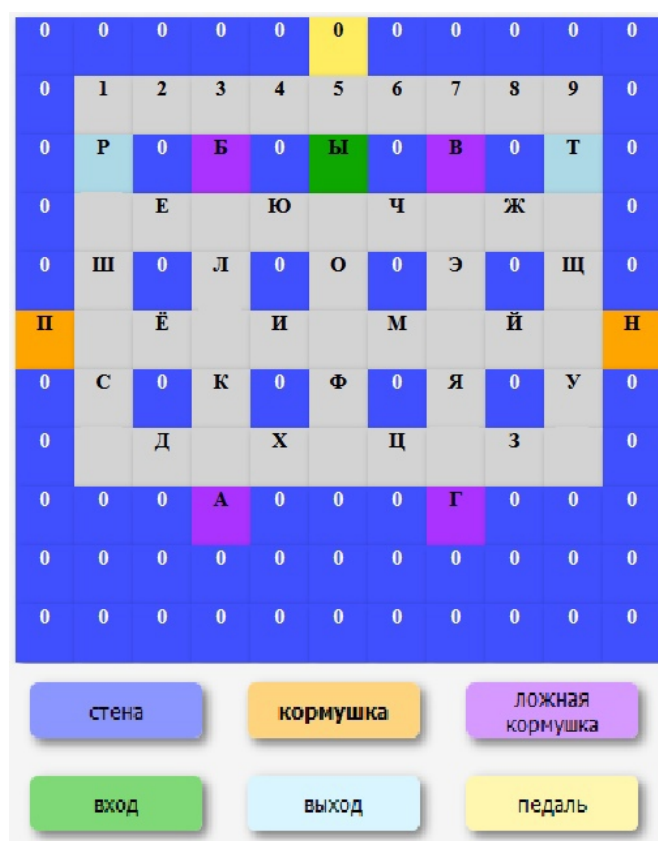


Рис. 4.3: Схема лабиринта. Челнок.

Основной акцент в нашей работе был сделан на работе с лабиринтом Никольской. Для остальных лабиринтов были реализованы прохождения только для реальных последовательностей, так как обучить модель машинного обучения с достаточной точностью было невозможно, поскольку количество реальных прохождений лабиринта было недостаточно.

4.1 Макеты для графического интерфейса

Данный раздел был выполнен совместно.

Создание макета графического интерфейса для лабиринта — это процесс проектирования пользовательского интерфейса, который должен обеспечивать простоту использования и понимания, а также соответствовать функциональным требованиям приложения.

Обозначим, что включает в себя создание макетов для графического интерфейса:

1 Исследование. В этом этапе было произведено изучение требований и целей проекта.

Были определены основные элементы интерфейса, его структура и расположение.

Основные элементы интерфейса, которые были реализованы:

- Входное окно.
- Схема самого лабиринта, которая отображается на экране.
- Управляющие элементы, такие как кнопки выбора лабиринта, в котором происходили эксперименты.
- Окно отображения текущего состояния “игры”, например, на каком этапе прохождения лабиринта находится животное.

2 Непосредственно создание самого макета.

Данный этап был реализован с помощью библиотеки Pygame [3]. Мы остановились в расширении 800 пикселей в ширину и 600 пикселей в длину. Спроектирован каждый лабиринт.

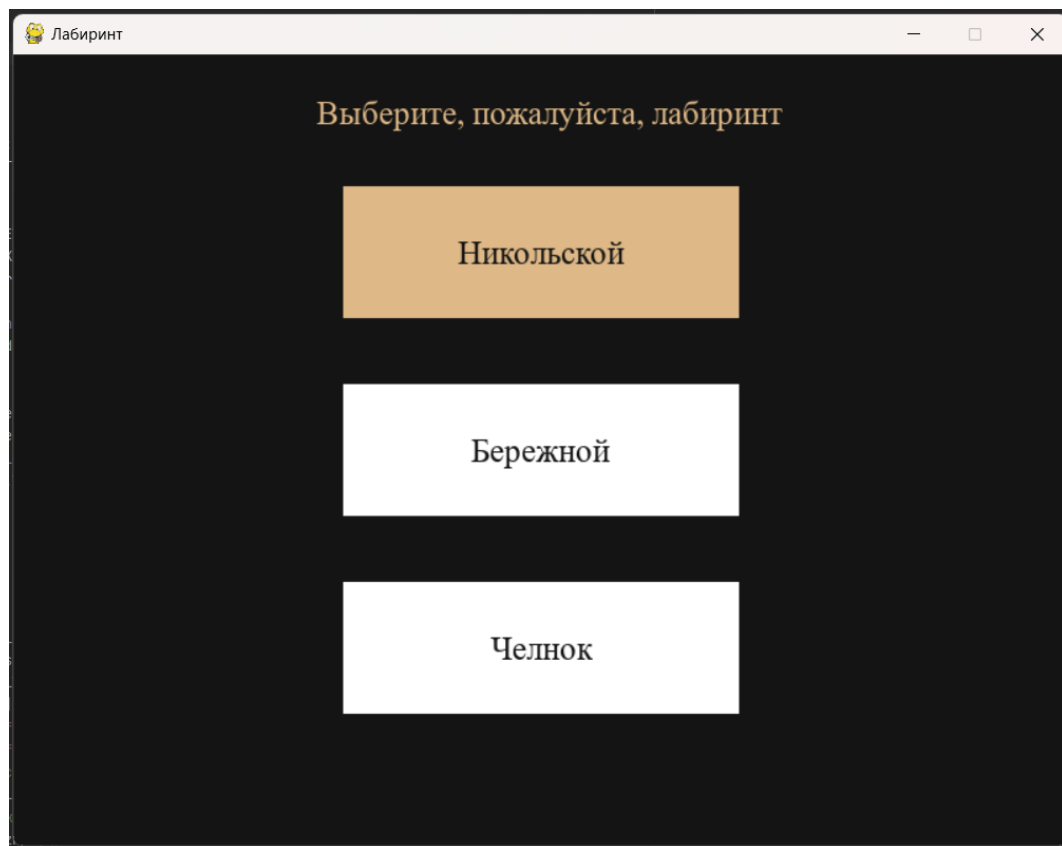


Рис. 4.4: Входное сообщение.

Далее были проработаны основные точки БЗЭ(вход-выход-кормушка), отмасштабировали схему лабиринта для заданной нами размерности.

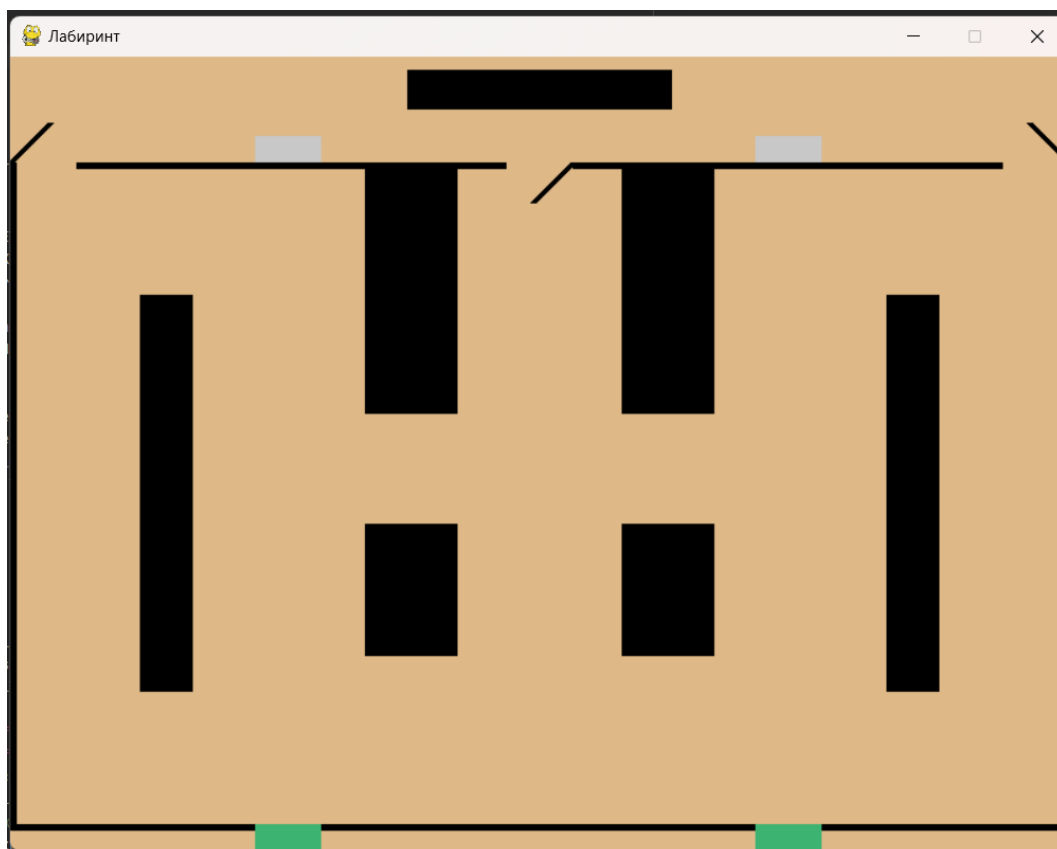


Рис. 4.5: Макет лабиринта. Никольская.

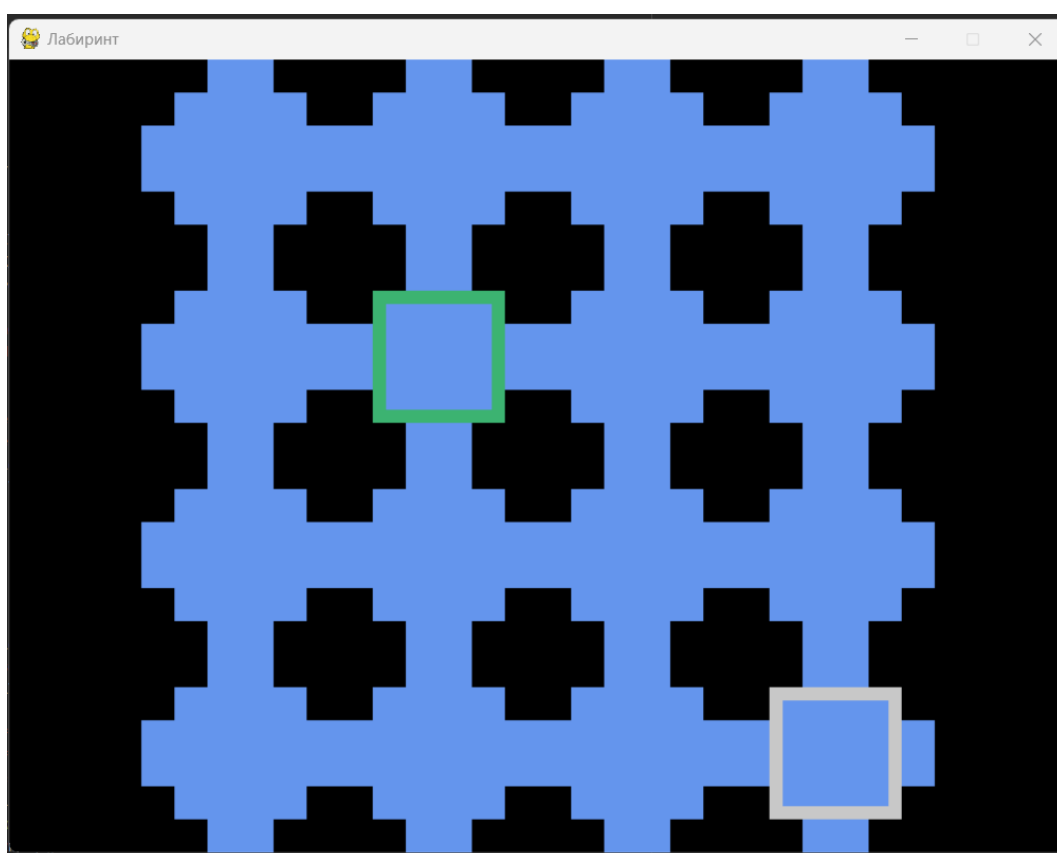


Рис. 4.6: Макет лабиринта. Бережной.

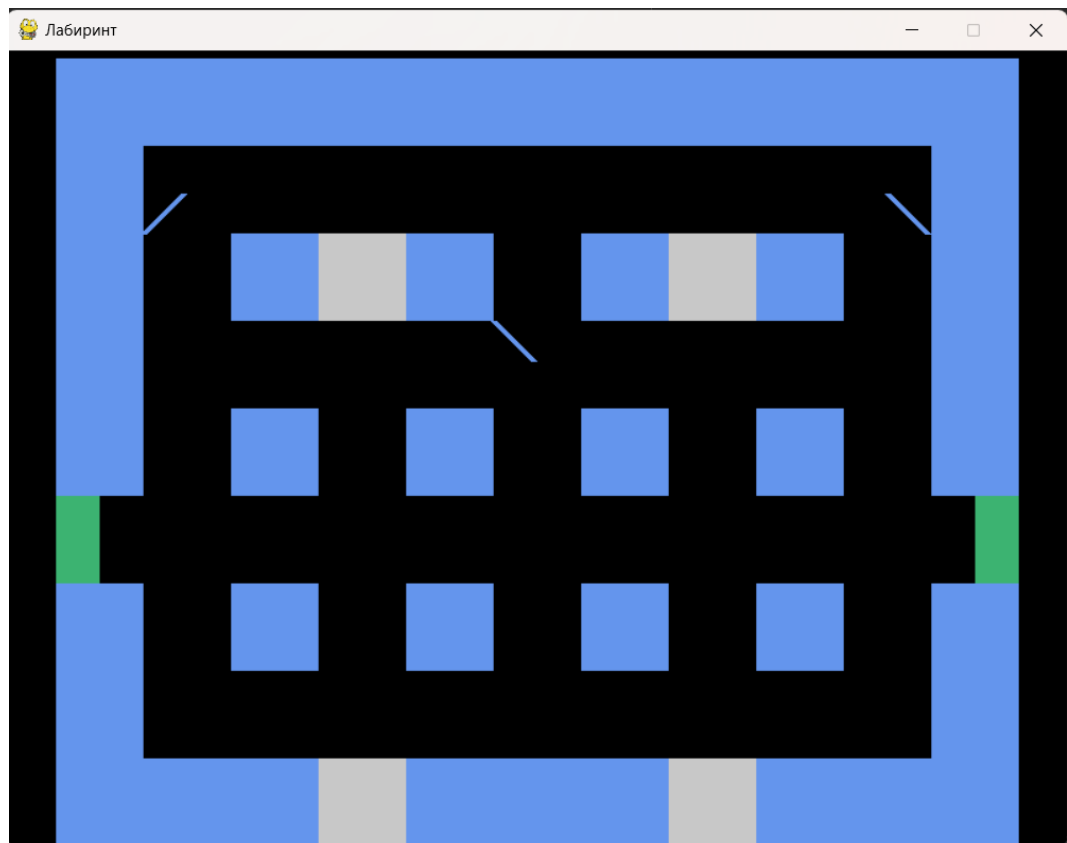


Рис. 4.7: Макет лабиринта. Челнок.

3 Тестирование созданного графического интерфейса.

После создания наброска макета, было проведено тестирование, чтобы определить, насколько удобен интерфейс для пользователя.

4 Внесение правок и внедрение.

После успешного тестирования и отладки графический интерфейс был готов к завершающей стадии процесса создания макета. Он был внедрен в конечный продукт и проект стал доступен для использования сторонними пользователями.

4.2 Реализация демонстраирования маршрутов животного

Данный раздел выполнила: Антонова Евгения.

На данном этапе мы переходим к реализации созданных нами макетов лабиринта в коде. Программа была написана с помощью библиотеки Pygame [3] в Python.

Непосредственно преимущества Pygame:

- 1 Простота использования: Pygame предоставляет простой и понятный интерфейс для создания графических приложений. Он предоставляет широкий спектр функций и инструментов для работы с графикой, звуком и пользовательским вводом.

- 2 Pygame работает на различных операционных системах, таких как Windows, macOS и Linux. Это обеспечивает переносимость приложений на разные платформы без необходимости значительных изменений в коде.
- 3 Широкие возможности графики: Pygame позволяет создавать и отображать графические объекты, такие как спрайты, изображения, фигуры и текст. Поэтому есть возможность легко управлять их положением, размером, цветом и другими атрибутами.
- 4 Игровая разработка: Pygame является популярным инструментом для разработки простых игр. Он предоставляет функциональность, необходимую для создания игрового цикла, управления спрайтами, обработки столкновений и многое другое.

В целом, Pygame является мощным инструментом для создания графического интерфейса, особенно если необходимо разрабатывать игры.

Реализация интерфейса:

Ниже приведено описание основных элементов кода для создания графического интерфейса с использованием Pygame:

- файл *path_.py* хранит в себе функции, основанные на DFS, для преобразования входной последовательности в виде строки в массив для передачи её на построение.
- файл *info_.py* хранит в себе константные переменные (словари и цвета), обозначающие список связности для каждого лабиринта, координаты в пикселях всех вершин лабиринтов и цвета, используемые при прорисовке лабиринтов.
- файл *build_.py* хранит в себе функции для построения самих лабиринтов. Здесь отрисовываются все стенки лабиринтов, входы и выходы, кормушки (истинные и ложные).
- файл *labirinths_.py* хранит в себе функции построения последовательностей, которых поочередно берутся из переданных данных, сгенерированных в части машинного обучения.
- основная часть *labirinth_VISUAL.ipynb* для запуска всей программы, где содержится класс *Button* для инициализации кнопок выбора лабиринта, инициализация окна Pygame и обработка действий пользователя.

Краткое описание шагов работы:

- 1 Инициализация окна.

- 2 Создание и отображение объектов.
- 3 Построение маршрутов.
- 4 Код для обработки событий Pygame, таких как нажатие клавиш, ввод данных с клавиатуры или закрытие окна.
- 5 Запуск программы.
- 6 Внедрение написанной программы в общий код для отработки построенных последовательностей.

В данном разделе представлена основная структура кода для создания графического интерфейса лабиринта с использованием Pygame.

4.3 Описание использования

Графический интерфейс, разработанный для демонстрации результатов обучения ранее, представляет пользователям удобный способ взаимодействия с моделью и анализа полученных данных.

Опишем использование графического интерфейса с точки зрения пользователя:

- 1 При запуске приложения пользователь увидит главное окно, в котором будет отображаться список лабиринтов для выбора соответствующего. Это позволит пользователю легко найти нужный лабиринт из доступных в приложении вариантов.
- 2 При нажатии кнопки с названием выбранного лабиринта, пользователь перейдет в новое окно с изображением самого лабиринта. Здесь пользователь сможет увидеть структуру лабиринта, его проходы, стены и возможные пути.
- 3 Переходя к этапу ввода данных, действия разделяются на две части, в зависимости от лабиринта.

3.1. Лабиринт Никольской. Пользователю будет предложено вписать тип животного и номер теста [1]. Эти данные могут быть необходимы для выполнения задачи или соревнования в рамках лабиринта. Например, пользователь может выбрать тип животного, которое будет искать выход из лабиринта, и указать номер теста для учета результатов. После нажатия клавиши ENTER начинает строиться наилучшее приближение моделью последовательности, а также соответствующий

ей реальный маршрут. При повторном нажатии клавиши строится следующая пара последовательностей. Между проходами можно изменять выбор животного и теста, это сделано для того, чтобы была возможность при разовом запуске окна провести полное обучение животного, то есть поочередно строить все номера тестов и прослеживать процесс обучения.

3.2. Лабиринт Челнок/Бережной. После ввода данных пользователем, программа будет строить маршрут последовательно. Она будет вычислять оптимальный путь от начальной точки до выхода из лабиринта, используя различные алгоритмы поиска пути. Также после нажатия клавиши ENTER начинают последовательно строиться реальные маршруты.

4 Далее программа переходит к визуализации.

4.1. В случае с работой в лабиринте Никольской программа будет визуально отображать как строится реальная последовательность движения (зеленого цвета), основанная на фактических перемещениях внутри лабиринта, и сгенерированная последовательность (красного цвета), которая показывает предполагаемые пути или варианты движения. Это поможет пользователю видеть разницу между оптимальным маршрутом и возможными альтернативными вариантами.

4.2. Если пользователь остановился на лабиринте Бережной или Челнок, то программа будет отрисовывать только сгенерированные нашей моделью маршруты.

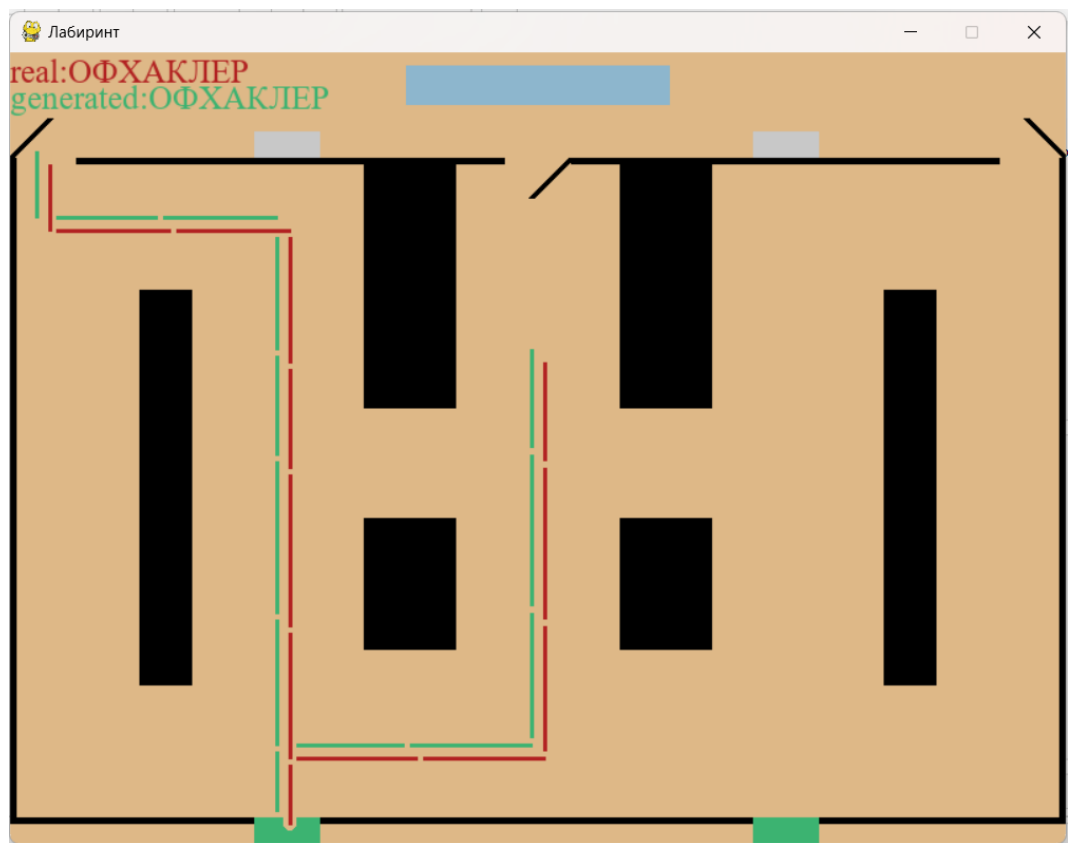


Рис. 4.8: Прохождение лабиринта. Никольская. Крысы. Итог.

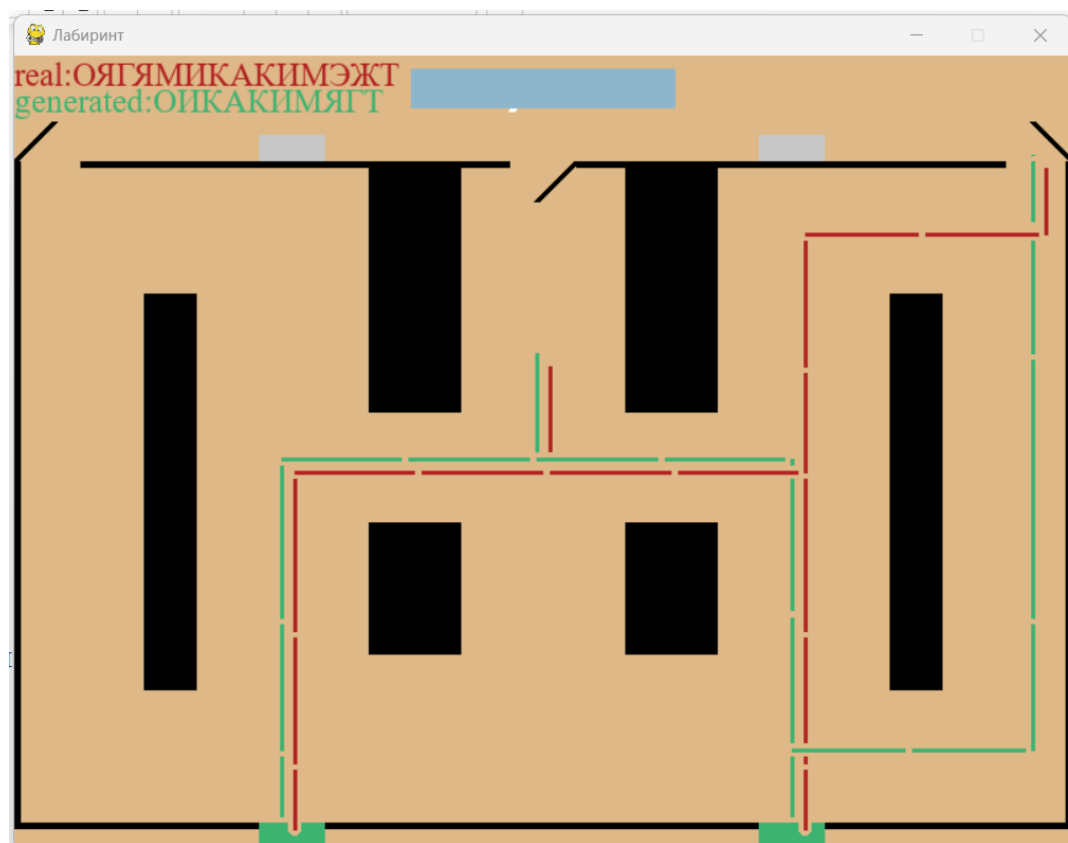


Рис. 4.9: Прохождение лабиринта. Никольская. Обезьяны. Итог.

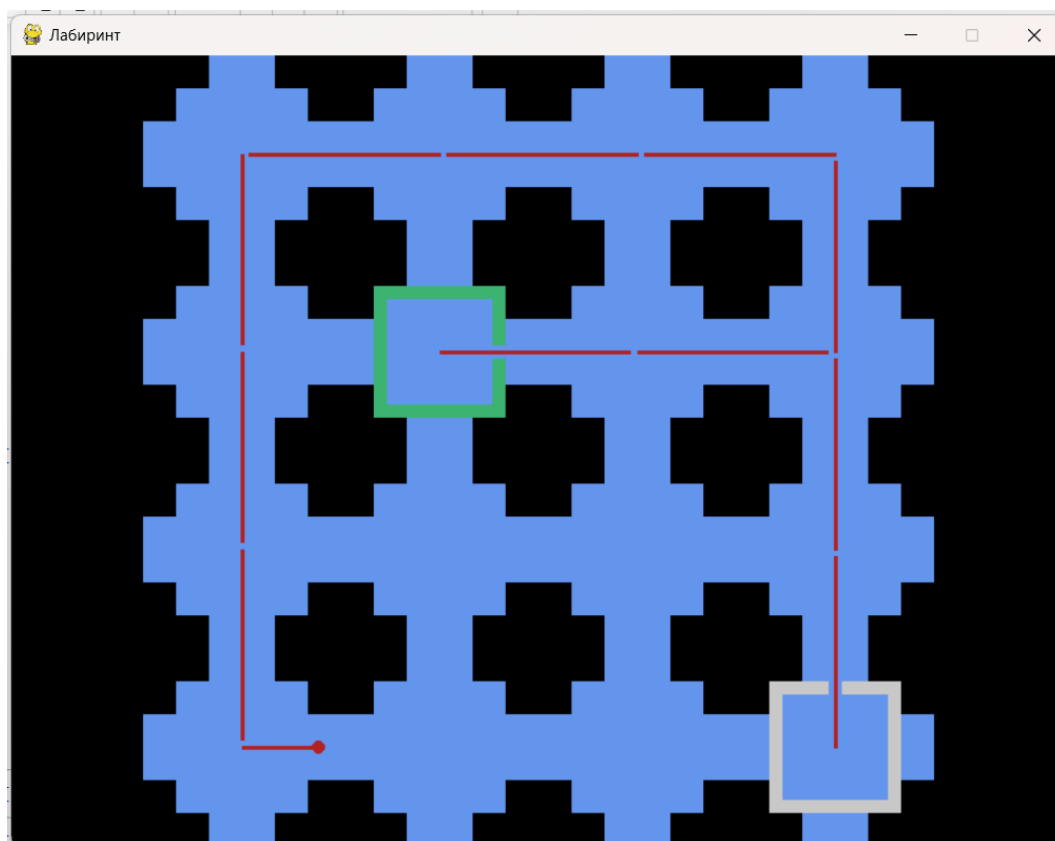


Рис. 4.10: Прохождение лабиринта. Бережной. В процессе.

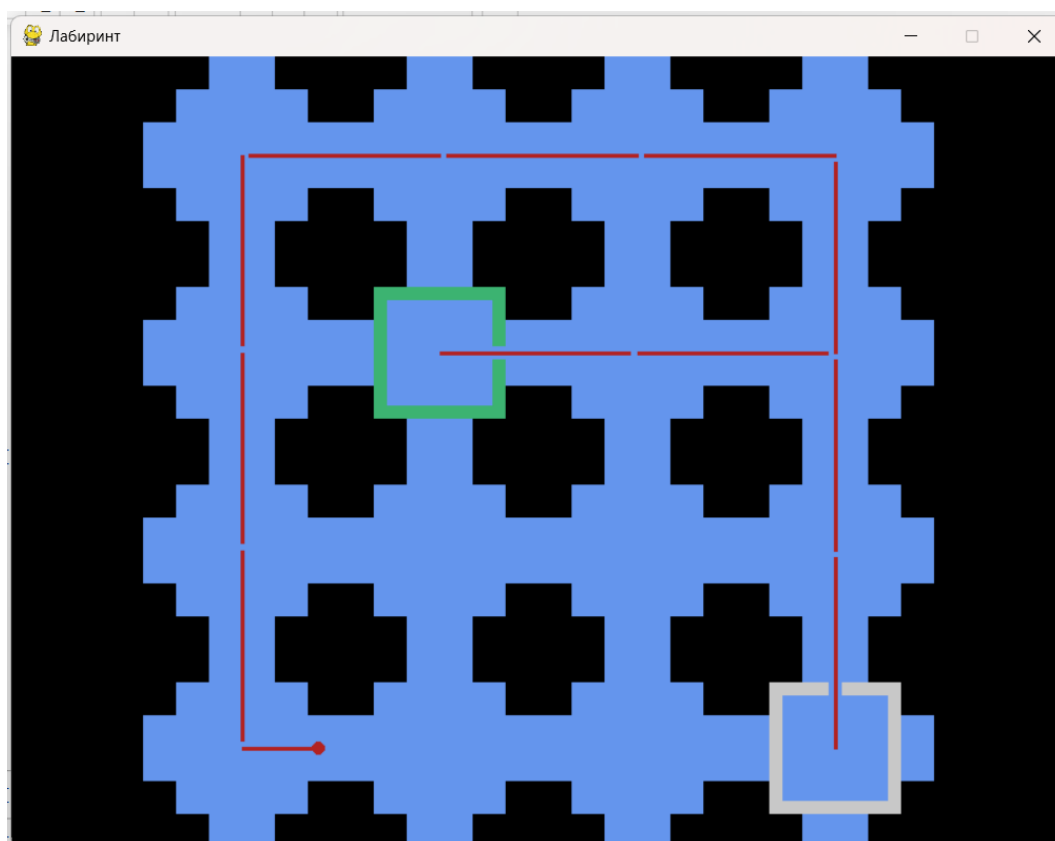


Рис. 4.11: Прохождение лабиринта. Челнок. В процессе.

5 Заключение

Нашей целью было разработать эффективную систему, которая позволит моделировать освоение животным лабиринтов без необходимости внешнего управления или обучения.

С помощью методов машинного обучения, таких как глубокое обучение, мы создали модель, способную адаптироваться к различным лабиринтам и находить оптимальные пути к цели. Эта модель была обучена на большом наборе данных, содержащем информацию о различных лабиринтах и правильных путях к выходу.

Графический интерфейс, который мы разработали, предоставляет удобный способ взаимодействия с моделью и наблюдения за процессом прохождения лабиринта животными. Он включает в себя функции визуализации, отображения текущего состояния и пути, а также возможности настройки параметров модели и лабиринта.

Наши эксперименты показали, что модель машинного обучения успешно применяется к прохождению лабиринта животными. Животные, прошедшие обучение на данной модели, продемонстрировали значительное улучшение в навигации и способности находить выход из лабиринта. Это открывает новые возможности для исследований в области поведения животных и улучшения их познавательных способностей.

В целом, мы убеждены, что успешная реализация модели машинного обучения для прохождения животным лабиринта и соответствующего графического интерфейса открывает новые перспективы в области исследования поведения животных.

Список литературы

- [1] Raphael Holzer Revision Copyright 2019. *Work with text*. URL: https://pygame.readthedocs.io/en/latest/4_text/text.html (дата обр. 26.02.2023).
- [2] PyTorch Copyright 2023. *LANGUAGE TRANSLATION WITH NN.TRANSFORMER AND TORCHTEXT*. URL: https://pytorch.org/tutorials/beginner/translation_transformer.html (дата обр. 04.03.2023).
- [3] pygame v2.3.0 documentation. *Pygame Front Page*. URL: <https://pyga.me/docs/> (дата обр. 26.02.2023).
- [4] Ильюшин Е. А. Евграфов В. А. “Спайковые нейронные сети”. В: *International Journal of Open Information Technologies, ISSN 2307-8162 vol. 9 no. 7*. 2021.
- [5] Дидык Л. А. Никольская К. А. “Структура внешней среды как системообразующий фактор условнорефлекторного процесса”. В: *Сравнительная физиология высш. нервн. деят. человека и животных*. Изд-во Наука, 1990, с. 69—90.

Приложения

В данном разделе внимание будет уделено основным функциям, которые были описаны в разделах **3.2** и **4.2**

- Код, относящийся к разделу разработки модели машинного обучения (обучение и валидация на одной эпохе):

```
def process_epoch(model, optimizer, dataloader, tqdm_desc): # ?
    optimizer

    model.train() if model.training else model.eval()
    total_loss = 0
    total_batches = len(dataloader)

    for src, tgt in tqdm(dataloader, desc=tqdm_desc):
        src, tgt = src.to(DEVICE), tgt.to(DEVICE)

        tgt_input = tgt[:-1, :]
        src_mask, tgt_mask, src_padding_mask, tgt_padding_mask =
            create_mask(src,
                        tgt_input)

        logits = model(src, tgt_input, src_mask, tgt_mask,
                        src_padding_mask,
                        tgt_padding_mask,
                        src_padding_mask)

        tgt_out = tgt[1:, :]
        loss = loss_fn(logits.reshape(-1, logits.shape[-1]),
                        tgt_out.reshape(-1
                        ))

        total_loss += loss.item()

    if model.training and optimizer is not None:
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    average_loss = total_loss / total_batches
    return average_loss
```

- Код, относящийся к разделу разработки графического интерфейса (подготовка пути для визуализации):

```

def path_Nikolskaya(path, maze_Nikolskaya):
    path_res = []
    for k in range(len(path) - 1):
        nodes = maze_Nikolskaya.keys()
        unvisited = {node: None for node in nodes}
        visited, current, currentDistance = {}, path[k], 0
        unvisited[current] = currentDistance
        distance = 1
        all_path = list(list())
        while True:
            for neighbour in maze_Nikolskaya[current]:
                if neighbour not in unvisited: continue
                newDistance = currentDistance + distance
                if unvisited[neighbour] is None or unvisited[
                    neighbour]
                    >
                    newDistance
                    :
                    unvisited[neighbour] = newDistance
                    all_path.append([current, neighbour])
            visited[current] = currentDistance
            del unvisited[current]
            if not unvisited: break
            candidates = [node for node in unvisited.items() if
                           node[1]]
            current, currentDistance = sorted(candidates, key=
                                               lambda x: x[1]
                                               )[0]

        ans_path = ""
        now = path[k + 1]
        all_path_rev = list(reversed(all_path))
        for i in range(len(all_path_rev)):
            if all_path_rev[i][1] == now:
                ans_path += all_path_rev[i][1]
                now = all_path_rev[i][0]

        ans_path += path[k]
        path_res.append(ans_path[::-1])
    return path_res

```


- Код, относящийся к разделу графического интерфейса (построение лабиринта на примере лабиринта Бережной):

```
def build_Berezhnoy(screen):
    screen.fill(BLACK)
    pygame.display.update()
    # rects
    for i in range(4):
        r1 = pygame.Rect(150 + i * 150, 0, 50, 600)
        pygame.draw.rect(screen, BLUE, r1, 0)
        r2 = pygame.Rect(100, 50 + i * 150, 600, 50)
        pygame.draw.rect(screen, BLUE, r2, 0)

    # squares
    for i in range(4):
        for j in range(4):
            sq = pygame.Rect(125 + i * 150, 25 + j * 150, 100, 100)
            pygame.draw.rect(screen, BLUE, sq, 0)

    # enter
    enter = pygame.Rect(125 + 150, 25 + 150, 100, 100)
    pygame.draw.rect(screen, GREEN, enter, 10)

    # korm
    korm = pygame.Rect(125 + 3 * 150, 25 + 3 * 150, 100, 100)
    pygame.draw.rect(screen, GRAY, korm, 10)
```

- Код, относящийся к разделу графического интерфейса (построение маршрута на примере лабиринта Бережной):

```
build_Berezhnoy(screen)
pth_ = path_Berezhnoy(df_Berezhnoy.path[now], maze_Berezhnoy)
for i in range(len(pth_)):
    for j in range(len(pth_[i]) - 1):
        k = 0
        x_start, y_start = coordinates_Berezhnoy(pth_[i][j])
        x_finish, y_finish = coordinates_Berezhnoy(pth_[i][j + 1])
        while y_start != y_finish or x_start != x_finish:
            do_y_start = ((y_start < y_finish) - (y_start >
                                                                    y_finish))
```

```

do_x_start = ((x_start < x_finish) - (x_start >
                                                    x_finish))

pygame.draw.circle(screen, RED, (x_start, y_start), 5)
pygame.display.update()
pygame.draw.circle(screen, BLUE, (x_start, y_start), 5
                        )

y_start += do_y_start
x_start += do_x_start
k += 1
pygame.draw.line(screen, RED, (
    x_start - k * do_x_start, y_start - k * do_y_start
),
                    (x_start, y_start), 3)
clock.tick(100)

now += 1

```

- Код, относящийся к разделу графического интерфейса (инициализация кнопки для выбора):

```

def __init__(self, x, y, width, height, buttonText='Button',
                onclickFunction=None,
                onePress=False):

    self.x = x
    self.y = y
    self.width = width
    self.height = height
    self.onclickFunction = onclickFunction
    self.onePress = onePress
    self.alreadyPressed = False
    self.Text = buttonText
    self.fillColors = {
        'normal': '#ffffff',
        'hover': FONT,
        'pressed': '#333333',
    }

    self.buttonSurface = pygame.Surface((self.width, self.height))
    self.buttonRect = pygame.Rect(self.x, self.y, self.width, self
                                    .height)

    self.buttonSurf = font.render(buttonText, True, (20, 20, 20))
    objects.append(self)

```