ORES

Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

АНАЛИЗ ПОДХОДОВ ИНТЕГРАЦИИ НЕЙРОСЕТИ В UNITY: ИССЛЕДОВАНИЕ СОВРЕМЕННЫХ ПРАКТИК И ИНСТРУМЕНТОВ

Сухоярский Денис Игоревич,
Студент, Донской Государственный Технический Университет,
кафедра «Медиатехнологии»
densuh68@gmail.com

Маевский Илья Александрович
Студент, Донской Государственный Технический Университет,
кафедра «Медиатехнологии»,
gagapool.55785@gmail.com

Аннотация

Данная статья посвящена анализу и систематизации современных подходов к интеграции нейронных сетей в Unity — одной из наиболее популярных платформ для разработки игр. Основное внимание уделено исследованию инструментов, таких как Unity ML-Agents Toolkit, Sentis, Muse, Barracuda и Unity Perception. Рассматриваются их функциональные возможности, особенности применения, а также ограничения, связанные с внедрением технологий искусственного интеллекта в игровые проекты.

Целью работы является формирование целостного представления о том, как современные нейросетевые технологии могут использоваться для создания инновационного игрового опыта.

Ключевые слова: ИИ, искусственный интеллект, разработка игр, НС, нейронные сети.

ANALYSIS OF NEURAL NETWORK INTEGRATION APPROACHES IN UNITY: RESEARCH OF MODERN PRACTICES AND TOOLS

Sukhoyarsky Denis Igorevich,
Student, Don State Technical University,
Department of «Media Technologies»
344000, Russian Federation, Rostov – on – Don, pl. Gagarina 1
E-mail: densuh68@gmail.com

Mayevsky Ilya Alexandrovich,
Student, Don State Technical University,
Department of «Media Technologies»
344000, Russian Federation, Rostov – on – Don, pl. Gagarina 1
E-mail: gagapool.55685@gmail.com

ABSTRACT

This article is devoted to the analysis and systematization of modern approaches to the integration of neural networks in Unity, one of the most popular platforms for game development. The main focus is on the study of tools such as Unity ML-Agents Toolkit, Sentis, Muse, Barracuda and Unity Perception. Their functionality, application features, and limitations associated with the implementation of artificial intelligence technologies in game projects are considered.

The purpose of the work is to form a holistic view of how modern neural network technologies can be used to create innovative gaming experiences.

Keywords: AI, artificial intelligence, game development, NS, neural networks.

Введение

Цель: исследовать существующие современные практики и инструменты для интеграции нейронной сети в игровой движок Unity.

Задачи: исследование ИИ технологий и инструментов, используемых в игровой индустрии, анализ способов для интеграции нейронной сети в игровой движок.

Стремительное развитие искусственного интеллекта (ИИ) и нейронных сетей оказало значительное влияние на индустрию разработки программного обеспечения и игр. Стало возможным создавать реалистичное поведение персонажей, динамичную адаптацию игровых условий и оптимизацию геймплея. Применение нейронных сетей в разработке видеоигр может способствовать ускорению разработки, а также повышению качества игры [1].

Данная статья нацелена на анализ подходов к интеграции нейронных сетей в Unity, исследование современных практик и инструментов, а также выявление их возможностей и ограничений. Основной целью является формирование целостного представления о применении технологий ИИ в Unity для игровых проектов. В статье рассматриваются ключевые инструменты, такие как Unity ML-Agents Toolkit, Sentis, Muse, Barracuda и Unity Perceptio. Их функциональные возможности и особенности применения.

Таким образом, статья представляет собой попытку систематизировать существующие знания о нейросетевых технологиях в Unity и оценить их значимость для разработчиков программного обеспечения.

Основная часть (методология, результаты)

Основные определения и термины

В начале дадим определения некоторым необходимым терминам и понятиям таким как нейронная сеть, языковая модель, для лучшего понимания дальнейшего материала статьи.

Нейронная сеть – это тип искусственного интеллекта, имитирующий работу человеческого мозга. Он состоит из множества узлов, или нейронов, которые обрабатывают информацию и передают ее другим нейронам. В то время как языковая модель нейронной сети (Neural Language Model) – это модель, использующая нейронные сети для предсказания следующего слова или фразы в последовательности текста. Она обучается на большом объеме текста и пытается предсказать наиболее вероятное следующее слово или фразу, учитывая предыдущую последовательность слов [2].

Стоит отметить, что существует достаточно большое количество типов языковых моделей таких как: рекуррентные нейронные сети, долгая краткосрочная память, сверточные нейронные сети, трансформаторные модели, самообучающиеся архитектуры и т.д. Все они имеют свои преимущества и недостатки и выбор зависит от поставленной задачи [1].

Инференс нейронных сетей (inference) относится к процессу применения обученной модели для предсказания результатов на новых данных. Это включает использование архитектуры и параметров нейронной сети, полученных в ходе обучения, для выполнения таких задач, как классификация, регрессия, сегментация или другие типы анализа данных. Инференс применяется, например, в распознавании речи, обработке изображений, машинном переводе и многих других областях.

Теперь, когда все необходимые определения даны, перейдем к исследованию ИИ технологий.

Существующие ИИ технологии в Unity

Unity, одна из ведущих платформ для разработки игр и интерактивных сценариев. Она предлагает ряд технологий, связанных с искусственным интеллектом (ИИ), которые активно используются в разработке видеоигр.

Основными из них являются Unity ML-Agents, Unity Sentis, Unity Muse, Unity Perception и Barracuda. Проанализируем каждый инструмент:

Набор инструментов Unity ML-Agents Toolkit это проект с открытым исходным кодом, позволяющий исследователям и разработчикам создавать моделируемые среды с помощью редактора Unity и взаимодействовать с ними через API Python [7]. Набор инструментов предоставляет ML-Agents SDK, который содержит все необходимые функции для определения сред в редакторе Unity вместе с основными скриптами C# для построения обучающего конвейера. Три основных сущности в ML-Agents SDK — это датчики, агенты и академия. Эти компоненты позволяют напрямую указывать какие игровые объекты являются агентами, предоставляют им разнообразные датчики для получения информации (изображения, результаты ray-cast и т.д.) и определяют их поведение, которое ссылается на различные механизмы принятия решений, такие как ввод игрока, жестко запрограммированные сценарии, встроенные модели нейронных сетей или взаимодействие через API Python. Так же Unity ML-Agents Toolkit содержит ряд примеров сред одноагентных или многоагентных сценариев. Их предназначение использование в бенчмаркинге алгоритмов RL и IL или в качестве шаблона для создания собственных сред.

Сильными сторонами данного набора инструментов является высокая гибкость, позволяющая разработчикам динамически изменять среды в соответствии с различными сценариями обучения ИИ, такими как обучение с подкреплением. Она поддерживает крупномасштабные параллельные симуляции, значительно ускоряя обучение.

Однако слабые стороны включают крутую кривую обучения для новичков и ограничения в реальном обобщении из-за «разрыва с реальностью», когда агенты, обученные в симуляции, могут испытывать трудности в практических приложениях.

Unity Sentis — это инструмент, предназначенный для вывода нейросетей непосредственно внутри Unity Runtime [3]. Это позволяет разработчикам интегрировать функции искусственного интеллекта в игры и приложения без необходимости использования внешних серверов или инструментов.

Сильные стороны:

1. Локальный инференс: исключает задержки, связанные с сетевыми запросами.
2. Интеграция в Unity: упрощает разработку приложений с ИИ.
3. Поддержка ONNX: обеспечивает совместимость с популярными библиотеками, такими как TensorFlow и PyTorch.

Слабые стороны:

1. Ограничения производительности: зависит от возможностей устройства, на котором запускается приложение.
2. Новизна технологии: ограниченное количество готовых решений и документации.

Эти аспекты делают Sentis полезным, но с ограничениями для сложных проектов.

Unity Muse это инструмент для создания и генерации контента с использованием технологий искусственного интеллекта и машинного обучения, направленный в первую очередь на автоматизацию и адаптацию различных элементов игрового процесса [6].

Unity Perception — это мощный фреймворк, предназначенный для генерации синтетических данных, особенно для задач обучения моделей компьютерного зрения. Он позволяет настраивать сцены, добавлять аннотации, такие как сегментационные маски, bounding boxes и ключевые точки, что делает его полезным для создания больших наборов данных, необходимых для обучения и тестирования моделей [5].

Сильные стороны:

1. Генерация синтетических данных: Unity Perception позволяет создавать разнообразные и контролируемые наборы данных, что особенно важно при недостатке реальных данных.
2. Высокая степень кастомизации: возможность задавать параметры сцены и объектов обеспечивает гибкость для создания данных под конкретные задачи.
3. Интеграция с другими инструментами Unity: простота использования и интеграция с существующими фреймворками Unity упрощают процесс внедрения в проекты.

Слабые стороны:

1. Проблемы с переносимостью: модели, обученные на синтетических данных, могут демонстрировать ограниченную производительность на реальных данных из-за различий между синтетической и реальной средой.
2. Сложность создания правдоподобных данных: для некоторых задач генерация данных, максимально приближенных к реальным, может потребовать значительных усилий и ресурсов.

Стоит отметить, что использование синтетических данных и таких инструментов как Unity Perception значительно улучшают качество моделей, обучаемых на данных, которые трудно или дорого получить в реальном мире. Например, исследования на тему генерации синтетических данных рассматривают преимущества таких подходов, включая возможность борьбы с дисбалансом данных и их справедливым распределением.

Barracuda — это библиотека и нейросетевой процессор (NPU), разработанный Unity для выполнения инференса моделей машинного обучения непосредственно в среде Unity. Она поддерживает работу с обученными моделями, экспортированными в формате ONNX (Open Neural Network Exchange) [4], и используется для интеграции предсказаний моделей в игры или приложения. Данный инструмент поддерживает все целевые платформы Unity (ПК, консоли, мобильные устройства), оптимизирован для использования на устройствах с ограниченными ресурсами и совместим с большинством популярных фреймворков обучения, такими как TensorFlow и PyTorch.

Таким образом Barracuda особенно полезна для локального запуска моделей без необходимости использования внешних серверов.

Применение технологий в индустрии

Описанные выше технологии применяются в различных сферах разработки видеоигр. Unity ML-Agents и Barracuda используют для разработки адаптивных NPC, улучшая их поведение в играх.

Unity Sentis имеет более обширный функционал, так как Sentis предоставляет инструменты для внедрения машинного обучения прямо в игры и приложения. Его применяют как для генерации процедурных уровней, так и для улучшения поведения NPC.

Unity Muse используется для генерации различного контента, например для динамической генерации музыкальных треков, которые адаптируются к действиям игрока, что делает музыку более интерактивной и персонализированной.

Unity Perception может быть применен для создания данных для обучения, где требуется точная симуляция окружающей среды и поведения объектов. Например, обучения автономных транспортных средств или роботов в играх.

На чём можно разработать нейросеть для Unity

Разработка нейросети для Unity требует грамотного комбинирования использования Python и C#. Python выступает основным инструментом для создания и обучения моделей, в то время как C# обеспечивает их интеграцию и использование в Unity.

Создание и обучение нейросетей в Python.

Python является основным языком для разработки и обучения нейронных сетей благодаря множеству мощных библиотек и инструментов. Рассмотрим наиболее популярные библиотеки:

TensorFlow — это один из самых известных фреймворков с открытым исходным кодом, разработанный Google. Он поддерживает широкий спектр задач машинного обучения, включая обучение и инференс нейронных сетей.

Основные преимущества TensorFlow:

- Гибкость: поддержка как низкоуровневого программирования, так и высокоуровневых API (например, Keras).
- Производительность: использование GPU и TPU для ускорения вычислений.
- Совместимость с форматом ONNX, который позволяет экспортировать модели для интеграции в Unity.

Пример применения TensorFlow для задач в Unity — это создание модели, обученной на большом наборе данных, например, для генерации поведения NPC. Модель можно экспортировать в формате ONNX и интегрировать в Unity с помощью Unity Barracuda или Unity Sentis.

PyTorch — ещё один мощный фреймворк для машинного обучения, популярный среди исследователей и разработчиков благодаря своей гибкости и простоте. Он широко используется для разработки сложных моделей, таких как трансформеры, генеративные модели или модели компьютерного зрения.

Основные преимущества PyTorch:

- Интуитивность: динамическая вычислительная графика делает процесс разработки и отладки моделей более удобным.
- Широкий набор инструментов: PyTorch поддерживает библиотеки для работы с различными архитектурами нейросетей, такими как TorchVision (для обработки изображений) и TorchAudio (для работы со звуком).
- Совместимость с ONNX: модели, созданные в PyTorch, можно легко экспортировать для последующего использования в Unity.

Пример использования PyTorch — разработка модели распознавания объектов на основе изображений из игрового мира. После обучения модель можно экспортировать и интегрировать в Unity для анализа действий игрока или окружения.

Интеграция модели нейросети в Unity

Для внедрения нейросетей в игровой движок Unity существуют несколько инструментов, каждый из которых обладает своими особенностями, преимуществами и недостатками. Рассмотрим ключевые из них: Unity ML-Agents, Unity Barracuda и Unity Sentis.

При выборе инструмента следует учитывать сложность задачи, доступные ресурсы и потребности проекта. Использование этих технологий в совокупности позволяет создавать

игры с продвинутыми возможностями ИИ, улучшая качество и разнообразие игрового опыта.

Способы внедрения нейросетей в Unity

Рассмотрим общие шаги по внедрению нейросетей в Unity.

Экспорт и оптимизация моделей для Unity.

Шаги внедрения:

1. Разработка модели в Python. Начните с разработки и обучения модели с использованием фреймворков Python, таких как TensorFlow или PyTorch. После того как модель будет обучена, экспортируйте её в формат ONNX.
2. Подготовка модели для Unity. Используя инструменты Unity, такие как Barracuda или Sentis, загрузите модель в Unity. Здесь необходимо убедиться, что модель правильно импортирована и готова к использованию в игровых условиях.
3. Оптимизация модели. Для того, чтобы модель работала эффективно в игровых приложениях, необходимо её оптимизировать. Это может включать:

- Квантизацию модели для уменьшения её размера.
- Прунинг (удаление ненужных нейронов) для ускорения инференса.
- Уменьшение точности вычислений (например, переход от 32-битных к 16-битным вычислениям).

Интеграция с игровыми объектами. Интегрируйте модель с игровыми объектами, используя C# скрипты. Это может включать обработку данных с помощью модели, принятие решений в реальном времени или управление поведением NPC.

Тестирование и улучшение. После внедрения модели в игру проведите тестирование для проверки её работы. Если модель не удовлетворяет требованиям по производительности или точности, повторите шаги оптимизации.

Особенности: Этот подход требует предварительной оптимизации моделей для работы в Unity, но позволяет интегрировать высокоэффективные нейросети в игры с минимальными затратами на обучение и настройку.

Теперь, когда определены общие шаги, рассмотрим подробнее второй этап внедрения. Для этого опишем шаги по применению Sentis и Barracuda.

Sentis

Для начала работы с Sentis его необходимо установить в Unity.

Шаги по установке Sentis в проект Unity:

1. Создайте новый проект Unity или откройте существующий.
2. Чтобы открыть менеджер пакетов, перейдите в "Окно" > "Менеджер пакетов".
3. Нажмите "+" и выберите "Добавить пакет по имени"....
4. Введите Com.unity.sentis.
5. Нажмите "Добавить", чтобы добавить пакет в свой проект.

Чтобы использовать Sentis для запуска нейронной сети в Unity необходимо выполнить следующие действия:

1. Необходимо подключить пространство имён.

Пример кода:

«using Unity.Sentis;».

2. Загрузить файл модели нейронной сети. Для этого необходимо экспортировать модель в формат ONNX, а после добавить файл модели в папку "Ресурсы" окно "Проект".

Пример кода:

«ModelAsset modelAsset = Resources.Load("model-file-in-assets-folder") as ModelAsset; var runtimeModel = ModelLoader.Load(modelAsset);».

3. Создать входные данные для модели. Для этого необходимо использовать Tensor API для создания тензора с данными для модели.

Пример кода:

«Texture2D inputTexture = Resources.Load("image-file") as Texture2D; Tensor<float> inputTensor = TextureConverter.ToTensor(inputTexture);

int [] array = new int [] {1,2,3,4};

Tensor<int> inputTensor = new Tensor<int> (new TensorShape(4), array);».

4. Создать механизм логического вывода (worker).

Пример кода:

«Worker worker = new Worker (runtimeModel, BackendType.GPUCompute);».

5. Запустите модель с входными данными для вычисления результата (логического вывода). Чтобы запустить модель следует использовать метод «Schedule».

Пример кода:

«worker.Schedule(inputTensor);».

Извлеките результаты. Для этого можно использовать метод «PeekOutput».

Пример кода:

«Tensor<float> outputTensor = worker.PeekOutput() as Tensor<float>;».

Barracuda

Принцип применения Barracuda очень схож с Sentis поэтому он будет описан вкратце.

Чтобы использовать Barracuda для запуска нейронной сети в Unity необходимо выполнить следующие действия:

1. Экспортируйте свою нейронную сеть из фреймворка Pytorch, TensorFlow или Keras в формат файла ONNX.
2. Добавьте файл. onnx в свой проект: он будет работать как обычный ресурс.
3. Загрузите модель из ресурса.
4. Создайте механизм логического вывода (worker).
5. Запустите модель.
6. Извлеките результаты.

Экспорт кода в формат ONNX

Загрузка обученной модели состоит из двух этапов:

1. Экспорт обученной модели из внешнего инструмента, такого как Pytorch, TensorFlow или Keras, в формат ONNX.
2. Импорт и загрузка ONNX-ресурса. Чтобы загрузить ONNX-модель, добавьте ее файл .onnx в папку ресурсов вашего проекта. Модель будет импортирована и отобразится как ресурс типа NNModel.

Экспортировать модель в ONNX легко, поскольку пример такого кода можно найти в документации самих библиотек, а также и в Unity.

Использование ML-Agents

Шаги внедрения:

1. Подготовка среды. Установите Unity ML-Agents, который включает Python API для обучения агентов. Для этого вам нужно настроить Unity, Python и необходимые зависимости с TensorFlow или PyTorch.
2. Разработка модели и обучение. Создайте модель нейросети с использованием Python, и обучите её для выполнения конкретной задачи в игровом мире. В Unity вам нужно настроить среду, которая будет взаимодействовать с вашим агентом, чтобы он мог учиться через симуляции.
3. Интеграция с Unity. Для того, чтобы обученная модель могла взаимодействовать с игровыми объектами в Unity, используйте API Unity ML-Agents. Этот инструмент позволяет создавать агентов, которые могут взаимодействовать с игровым миром и улучшать своё поведение через обучение с подкреплением (RL).
4. Тестирование и улучшение модели. После интеграции модели проведите тестирование, чтобы убедиться, что агент ведёт себя должным образом в игровом мире. Если необходимо, улучшите модель, настройте её гиперпараметры и продолжите обучение.

Особенности: Этот метод подходит для обучения моделей в реальных игровых условиях. Основное внимание уделяется взаимодействию агентов с игровым миром через обучение с подкреплением.

Unity Muse

Muse интегрирован в экосистему Unity и предлагает средства для ускорения разработки игр, включая генерацию текстов, 3D-моделей, визуальных эффектов и анимаций.

- Шаги внедрения Unity Muse:

1. Установка Unity Muse в проект

- Muse интегрирован в Unity Editor. Чтобы начать работу с инструментом, убедитесь, что у вас установлена последняя версия Unity с поддержкой Muse.
- Откройте Unity Editor, перейдите в “Window > Package Manager”, найдите пакет Unity Muse и добавьте его в проект.

2. Создание контента с использованием Muse

- Muse использует текстовые запросы (prompts) для генерации контента. Например, вы можете запросить создание персонажа, описав его основные характеристики.
- В редакторе Unity Muse предоставляется интерфейс, где разработчики вводят текстовые запросы и получают визуальный или текстовый контент.

3. Интеграция с игровыми объектами

- Сгенерированный контент автоматически добавляется в проект Unity. Для использования его в игре необходимо интегрировать результат с игровыми объектами, используя стандартные средства Unity (C# скрипты или редакторские инструменты).

4. Тестирование и оптимизация
- После генерации контента важно протестировать его на совместимость с игровым окружением и убедиться в его корректной работе. Если необходимо, проведите доработку полученного результата.

Особенности внедрения Muse: Unity Muse в значительной степени автоматизирует процессы генерации контента. В отличие от инструментов, таких как Barracuda или Sentis, Muse не требует предварительного обучения моделей, так как работает с уже встроенными возможностями. Основное отличие — упрощение начальных этапов разработки за счёт генерации готовых игровых элементов. Это делает его идеальным выбором для ускоренной разработки прототипов игр.

Unity Perception

➢ Шаги внедрения Unity Perception:

1. Установка Unity Perception
- Как и другие пакеты Unity, Perception добавляется через “Package Manager”. Откройте Unity Editor, перейдите в “Window > Package Manager”, найдите Unity Perception и добавьте его в проект.
2. Создание синтетических данных
- Perception предоставляет инструменты для создания виртуальных сцен с генерацией синтетических изображений. Используйте шаблоны, входящие в пакет, или создавайте собственные сцены, задавая параметры камеры, освещения и расположения объектов.
- Добавьте аннотации для объектов. Perception автоматически генерирует метаданные (например, bounding boxes, instance segmentation masks и keypoints).
3. Интеграция с игровыми механиками
- Если синтетические данные нужны для обучения моделей, интегрируйте Perception с Unity ML-Agents или другими инструментами для создания динамических сцен. Это позволяет моделям обучаться на большом количестве данных, имитирующих реальные условия.
4. Экспорт данных
- Сгенерированные наборы данных экспортируются в формате, удобном для использования в сторонних фреймворках машинного обучения (например, COCO или Pascal VOC).
5. Обучение моделей на сгенерированных данных
- Используйте синтетические наборы данных для обучения моделей в TensorFlow, PyTorch или других библиотеках. После обучения модели можно интегрировать их обратно в Unity с помощью Barracuda или Sentis.

Особенности внедрения Perception: Perception отличается от других инструментов тем, что его основная цель — не инференс или обучение модели, а создание данных для последующего обучения [8]. Однако этапы интеграции Perception схожи с Barracuda и Sentis в том, что он использует экспортированные данные или модели для интеграции в игровой проект. В отличие от Muse, Perception ориентирован на разработчиков, которые работают с компьютерным зрением и нуждаются в больших объёмах аннотированных данных.

Сравнение Muse и Perception с другими инструментами

- Общие черты:
- Установка через Package Manager.
- Интеграция с игровыми объектами через Unity Editor и C# скрипты.
- Использование дополнительных библиотек или инструментов для подготовки контента.
- Ключевые отличия:
- Muse фокусируется на генерации игрового контента (например, текстов и моделей) и не требует внешних данных или обучения моделей.
- Perception предназначен для создания синтетических данных и тесно связан с задачами компьютерного зрения, что делает его полезным на этапе подготовки модели к обучению.

Выводы

В данной статье был проведен глубокий анализ существующих подходов и инструментов для разработки нейронных сетей и их интеграции в игровой движок Unity. Быстрое развитие искусственного интеллекта открыло новые возможности для игровой индустрии, сделав более доступным создание сложного игрового поведения, адаптивных игровых механик и персонализированного контента. Unity, как одна из ведущих платформ разработки, предоставляет широкий спектр инструментов для работы с нейросетями, включая Unity ML-Agents Toolkit, Unity Sentis, Unity Muse, Unity Perception и Barracuda. Каждый из них имеет свои уникальные особенности и области применения, что позволяет разработчикам выбирать оптимальные решения в зависимости от задач проекта.

Рассмотренные технологии демонстрируют широкий спектр возможностей:

Unity Muse и Unity Perception действительно играют вспомогательную роль в экосистеме Unity:

- Unity Muse — помогает автоматизировать творческие процессы, такие как генерация контента (например, текстур, моделей или анимаций), но не занимается выполнением инференса или обучением моделей.
- Unity Perception — предназначен для создания синтетических данных (изображений, анотаций), которые затем используются для обучения моделей в сторонних инструментах (например, PyTorch или TensorFlow). Таким образом, Perception — это инструмент подготовки данных.

С другой стороны:

- Unity Sentis и Unity Barracuda — это основные инструменты для интеграции обученных нейросетей в Unity. Они непосредственно работают с моделями, созданными и обученными в Python (например, в PyTorch или TensorFlow), импортируя их в Unity через формат ONNX. Эти инструменты позволяют выполнять инференс нейросетей, оптимизированный для игровых условий.

Главное различие:

- Muse и Perception — инструменты поддержки и подготовки, которые помогают в создании контента (Muse) или данных для обучения (Perception). Они работают на этапе подготовки или создания вспомогательных материалов.

- Sentis и Barracuda — участвуют на этапе исполнения обученных моделей внутри Unity, обеспечивая их работу с игровыми объектами.

Если говорить проще, Muse и Perception помогают создать условия для работы нейросетей, а Sentis и Barracuda — запускают нейросети внутри игр. Unity ML-Agents можно отнести к инструментам обучения нейросетей. В отличие от Unity Muse и Perception, которые выступают скорее вспомогательными средствами, и Sentis/Barracuda, которые служат для инференса обученных моделей, ML-Agents используется именно для процесса обучения внутри симуляций в Unity.

Несмотря на значительные достижения, существует ряд ограничений, связанных с применением ИИ в Unity. К ним относятся сложности настройки инструментов, высокая зависимость производительности от характеристик устройства и недостаточная адаптация обученных в симуляции моделей к реальным сценариям. Кроме того, новизна некоторых технологий, таких как Sentis, ограничивает доступность документации и готовых решений, что может создавать трудности для разработчиков.

Важным аспектом разработки нейросетей для Unity является сочетание Python и C#: первый используется для обучения и оптимизации моделей, а второй – для их интеграции в игровую среду. Выбор таких фреймворков, как TensorFlow или PyTorch, позволяет использовать все преимущества современных методов машинного обучения и интегрировать их в Unity с помощью формата ONNX. Для успешного внедрения нейросетей критически важны этапы оптимизации, такие как квантизация и прунинг, что обеспечивает баланс между производительностью и качеством работы моделей.

Таким образом, использование нейросетевых технологий в Unity открывает перед разработчиками новые горизонты для создания инновационных игровых проектов. Однако успешное применение данных технологий требует тщательной подготовки, анализа задач и ресурсов проекта, а также умения работать с инструментами Unity и современными методами машинного обучения. Систематизация подходов, представленных в статье, помогает разработчикам сформировать комплексное представление о возможностях интеграции нейросетей в Unity, что делает её ценной для дальнейшего развития технологий ИИ в игровой индустрии.

Список литературы:

1. Сухоярский Д. И., Маевский И. А., Савельев И. М. ИССЛЕДОВАНИЕ ЯЗЫКОВЫХ МОДЕЛЕЙ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ГЕНЕРАЦИИ СЦЕНАРИЕВ ИГРОВОГО КВЕСТА //Оригинальные исследования. – 2023. – №12. – С. 146.
2. Сухоярский Д. И., Маевский И. А. ИССЛЕДОВАНИЕ СПОСОБОВ ПРИМЕНЕНИЯ ЯЗЫКОВОЙ МОДЕЛИ НЕЙРОННОЙ СЕТИ ДЛЯ РАЗРАБОТКИ ИГРЫ В ИГРОВОМ ДВИЖКЕ UNITY //Оригинальные исследования. – 2024. – №4. – С. 236.
3. Френч Ф. и др. Творческое использование OpenAI в образовании: примеры из разработки игр //Мультимодальные технологии и взаимодействие. – 2023. – Т. 7. – №. 8. – С. 81.
4. Пьердикка Р. и др. Глубокая реальность: фреймворк с открытым исходным кодом для разработки приложений дополненной реальности на основе ИИ // Экспертные системы с приложениями. – 2024. – Т. 249. – С. 123530.
5. Каули Н., Рекуперо Д. Р. Синтетическое дополнение данных для классификации действий на видео с использованием Unity //IEEE Access. – 2024. – т. 12. – С. 156172-156183. – doi: 10.1109.

6. Пешевич А. С. Инновационный искусственный интеллект в Unity. – 2024. – С. 275-277.
7. Савид Й. и др. Моделирование автономного вождения с использованием обучения с подкреплением: сравнительное исследование фреймворка ML-агентов Unity //Информация. – 2023. – Т. 14. – № 5. – С. 290.

References:

1. Sukhoyarsky D. I., Mayevsky I. A., Savelyev I. M. RESEARCH OF LANGUAGE MODELS OF NEURAL NETWORKS FOR GENERATING GAME QUEST SCENARIOS //Original Research. – 2023. – No. 12. – P. 146.
2. Sukhoyarsky D. I., Mayevsky I. A. RESEARCH OF WAYS TO APPLY A NEURAL NETWORK LANGUAGE MODEL FOR GAME DEVELOPMENT IN THE UNITY GAME ENGINE //Original research. – 2024. – No. 4. – P. 236
3. French F. et al. Creative use of OpenAI in education: case studies from game development //Multimodal Technologies and Interaction. – 2023. – T. 7. – №. 8. – P. 81.
4. Pierdicca R. et al. DeepReality: An open source framework to develop AI-based augmented reality applications //Expert Systems with Applications. – 2024. – T. 249. – P. 123530.
5. Cauli N., Recupero D. R. Synthetic Data Augmentation for Video Action Classification Using Unity //IEEE Access. – 2024. – vol. 12. – P. 156172-156183. – doi: 10.1109.
6. Peshevich A. S. Innovative artificial intelligence in Unity. – 2024. – P. 275-277.
7. Savid Y. et al. Simulated autonomous driving using reinforcement learning: A comparative study on unity's ML-agents framework //Information. – 2023. – T. 14. – №. 5. – P. 290.