Научная визуализация, 2025, том 17, номер 2, страницы 123 - 136, DOI: 10.26583/sv.17.2.09

Визуализация методов машинного обучения. Графическое программирование

Н.О. Шестерин[1]

Национальный исследовательский университет "Высшая школа эономики", Москва, Россия

[1] ORCID: 0000-0003-2134-8412, nshesterin@gmail.com

Аннотация

Технологии искусственного интеллекта и машинного обучения совершили фундаментальный скачок в своих возможностях за последний десять лет. Рост вычислительных мощностей и появление новых всё более эффективных методов машинного обучения позволяет ИИ не только выполнять типичные для них задачи в областях статистического анализа и оптимизации математических процессов, но и находить новые применения в смежных направлениях исследования, а также применяться в принципиально новых областях – как научного поиска, так и практических применений, в том числе на рынке, доступном массовому покупателю. Генерация изображений, аудио, анимаций, самообучение моделей управления роботизированных платформ и виртуальных механических моделей – эти и многие другие новые применения последних лет привели к медиа-буму вокруг ИИ и развитию интереса разработчиков и авторов самых разных областей и направлений.

При этом, методы разработки, исследования, тестирования и внедрения ИИ остались практически неизменными и до сих пор требуют знания языков программирования, библиотек машинного обучения, как и глубоких познаний и опыта непосредственно в области ИИ. Этот барьер специализации не только требует включения специалистов по машинному обучению в процесс разработки типичных для современных ИИ тривиальных программных приложений, но и лишает небольшие команды разработчиков и независимых авторов возможности использовать последние достижения этих технологий без существенных временных или даже денежных инвестиций в обучение.

Мной разработан прототип графического интерфейса, позволяющего пользователю без специального образования и без знания языков программирования разрабатывать и настраивать различные архитектуры методов машинного обучения, самообучения, а также тестировать эти методы как на широком спектре математических задач, так и в симуляциях физической среды. В этой статье, я даю краткое описание структуры, организации, принципа действия и возможностей этого интерфейса.

Ключевые слова: нейросети, искусственный интеллект, машинное обучение, блочное программирование, графический интерфейс.

1. Введение

Методы машинного обучения и в целом искусственного интеллекта (ИИ) сегодня применяются в широком спектре областей как научных исследований, так и практических задач, в том числе при разработке программных продуктов, рассчитанных на массового покупателя без специальных навыков и знаний. Число областей применения и сами по себе возможности методов ИИ значительно выросли за последние годы, что во многом обязано нескольким ключевым прорывам в областях глубоких нейросетей, самообучения, адверсариальных нейросетей и генеративных алгоритмов. Эти новые тех-

нологии развиваются с огромной скоростью, и новые достижения и открытия происходят в течение месяцев и даже недель. Последовавший за появлением высокоэффективных генеративных алгоритмов медиа-бум вокруг создаваемых ИИ медиа и второй бум, связанный с этим скачкообразным развитием технологий LLM (Large Language Models) или больших языковых моделей, привлекли внимание широкого спектра разработчиков из разных индустрий, часто далеких от ИИ и даже в целом индустрии информационных технологий. Все это сделало актуальным вопрос о потенциале визуализации методов машинного обучения и графического программирования.

Несмотря на практическую и теоретическую значимость проблемы, степень ее изученности невелика. Я в моей работе предпринимаю попытку компенсировать этот пробел.

2. Состояние индустрии на текущий момент

Сегодня методы машинного обучения и ИИ активно внедряются в медиаиндустрии, киноиндустрии, в онлайн-среде, в индустрии развлечений, в областях дизайна, маркетинга, а в контексте научного поиска – в биологии, медицине, химии, генетике, физике, астрономии, робототехнике, компьютерной безопасности. И это несмотря на то, что до недавнего времени ограничения в вычислительных мощностях и возможностях самих методов ИИ позволяли лишь специальные применения в областях компьютерного моделирования, статистического анализа и прочих в своём большинстве теоретических областях науки и техники. Вычислительные мощности современных компьютеров предоставляют возможность обывателям и независимым исследователям не только использовать, но и тренировать и даже разрабатывать методы машинного обучения на средних по возможностям общедоступных машинах или посредством облачных вычислений. Это, вместе с лавинным ростом интереса к искусственному интеллекту в широком спектре областей науки и техники, приводит к тому, что всё большее число людей начинает взаимодействовать с ИИ как в роли потребителя, так и в позиции разработчика.

На текущий момент времени существует несколько интерфейсов, рассчитанных на специалистов с минимальным опытом и даже обывателей, которые позволяют пользователю генерировать изображения, музыку, анимации, дип-фэйки и в целом эксплуатировать уже натренированные модели. Разработано большое число чат-ботов и интерфейсов LLM, дающих пользователям возможность генерировать тексты широкого спектра направленностей и существенных объемов. Существуют также программные реализации и сервисы, позволяющие внедрять некоторые методы ИИ в проекты в области IT с минимальным необходимым опытом разработки и без необходимости специального образования с фокусом на искусственный интеллект: сервисы Google, Amazon, Microsoft и других компаний предлагают генеративные ИИ для озвучки текста, Midjourney и Dall-E предоставляют возможность генерировать изображения по текстовому запросу, ChatGPT и схожие текстовые генераторы предлагают создание длинно форматных текстов разных стилей по заданию или для продолжения уже созданных отрывков.

Эти сервисы позволяют не только использовать результаты работы ИИ, но и встраивать его в веб-приложения и приложения для рабочего стола, чаще всего – с опорой на облачные вычисления. Однако для разработки собственных архитектур нейросетей разных методов до сих пор не существует признанных и доступных неспециалистам приложений и интерфейсов, не полагающихся на кодинг. Чаще всего работа с нестандартными архитектурами, тренировка нейросетей, и, конечно, разработка новых методов машинного обучения требует знакомства с языком программирования Python, библиотеками машинного обучения TensorFlow, Keras и другими. Эти знания и навыки нередко бывают необходимы специалистам из областей, в остальном не требующих знания программирования (например, в астрофизике большинство вычислений и ста-

тистический анализ данных могут производиться с помощью сервисов на подобии Wolfram Alpha или посредством псевдокода, однако для работы с ИИ практически обязательно знание Python). Это не только усложняет процесс обучения специалистов множества профессий (в том числе творческих), но и создаёт труднопреодолимый барьер для неспециалистов, у которых нет времени или возможности для часто платного обучения посредством онлайн-курсов без гарантии актуальности преподаваемого онлайн материала.

Я вижу две возможных причины отсутствия на рынке таких приложений. Первая причина связана с тем, с какой скоростью развиваются новые технологии в данной области – приложение, позволяющее работать с нейросетями (и другими типами ИИ), должно будет поддерживаться в актуальном состоянии и не уступать по вычислительной скорости более низкоуровневым решениям, таким как TensorFlow, или должно быть построено на базе таких решений, что в свою очередь сделает приложение зависимым от них. Вторая возможная причина состоит в сложности и многообразии устройства алгоритмов ИИ, что делает создание визуальной репрезентации или системы блочного программирования с целостной внутренней логикой проблематичным, а понимание такой системы – труднодоступным.

В рамках моего диссертационного исследования, мной разработан прототип программного решения, создающего графический интерфейс, позволяющий пользователю реализовывать нейросетевые и просто математические системы посредством блочного программирования с минимальным необходимым знанием нейросетевых алгоритмов и без необходимости кодинга. Это решение построено на базе созданной мной библиотеки машинного обучения на языке C++ и поэтому независимо от других программных продуктов в данной области. Разработанный мной программный продукт позволяет не только создавать произвольные нейросетевые архитектуры распространенных методов, но также обучать их локально, без использования облачных вычислений. Кроме того, программа позволяет тестировать эти решения на задаваемых пользователем математических задачах, а также в симулированной физической среде для тестирования виртуальных моделей роботизированных платформ, на подобии AI-gym и Mujoco.

3. Редактор нейросетевых алгоритмов

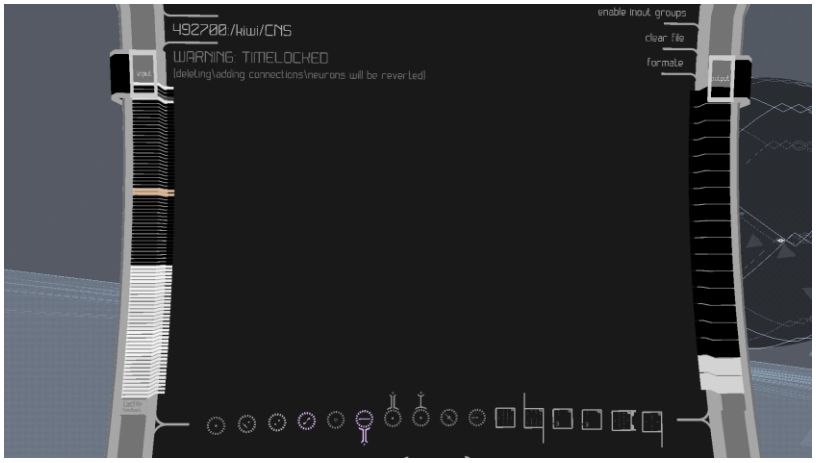

Рис. 1. Редактор нейросетевых алгоритмов.

Разработанное мной программное решение позволяет пользователю проектировать нейросетевые методы посредством графического интерфейса и взаимодействовать с ними посредством мыши и клавиатуры. Объекты данного интерфейса формализованы и визуализированы через пиктографические иконки двух типов – «Нейроны» и «Обособленные Обучаемые Юниты», внизу на рис. 1. Первый тип представляет собой визуализацию простейших математических функций: линейная функция, ReLU, Leaky

ReLU, гиперболический тангенс, генератор частот, генератор псевдослучайных чисел, «умножающий нейрон», «нейрон с памятью», и «возбуждаемый нейрон». Я опишу функции последних трёх видов «нейронов» далее в тексте. Второй тип объектов, «Обособленный Обучаемый Юнит» представляет собой визуализацию искусственной нейросети, работающей посредством машинного обучения. На данный момент, пользователю доступны шесть видов данного юнита: «Fully Connected Neural Net» - обыкновенная нейросеть, реализующая несколько методов машинного обучения в зависимости от заданной настройки, «Комбинирующий юнит», функционально идентичный первому, но позволяющий совмещать блоки данных от нескольких других юнитов, конволюционная нейросеть, «нейросеть с памятью», «нейросеть-ученик», реализующая методы DDPG и схожие методы самообучения, наконец – адверсариальная нейросеть.

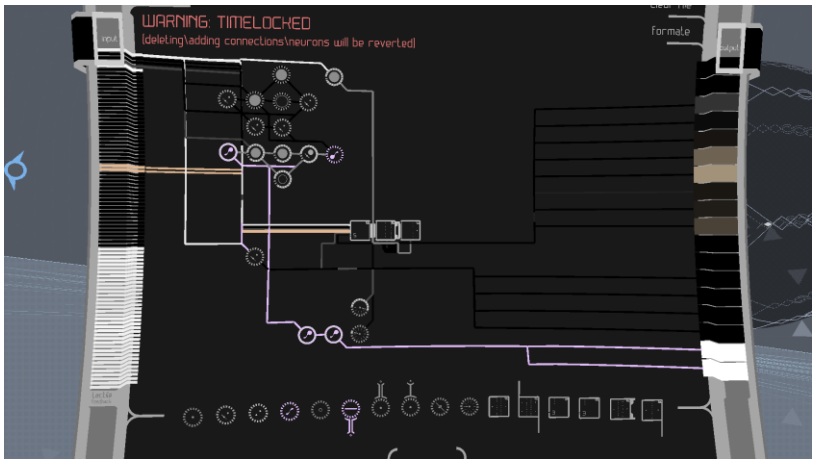

Рис. 2. Рабочее поле редактора.

Эти объекты располагаются пользователем в центральной рабочей области, изображенной на рис. 2 в произвольном порядке и месте в удобном пользователю виде. Передача информации между объектами производится посредством «связей». Так, связь между двумя «нейронами» позволяет создать комбинированную функцию – «нейрон», реализующий функцию ReLU, связанный с «нейроном» гиперболического тангенса функционально идентичен (1).

$$f(x) = Tanh(ReLU(x)) \quad (1)$$

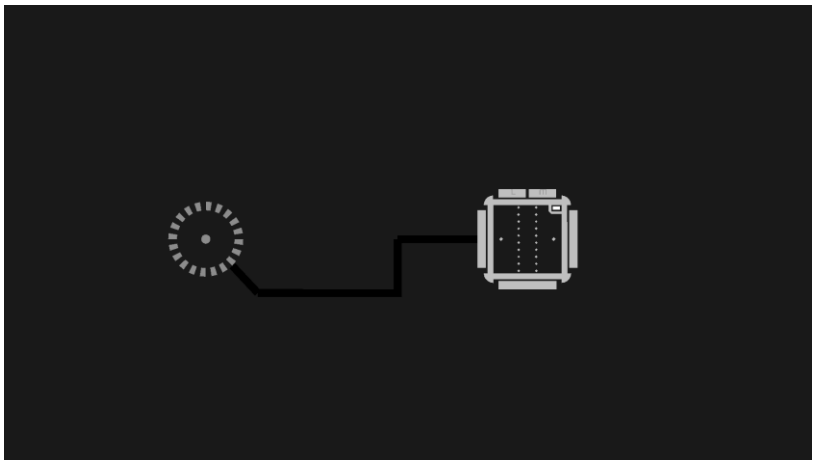
Рис. 3. "Нейрон" и "Обособленный Обучаемый Юнит" со связью между ними.

При создании связи между нейроном и «Обособленным Обучаемым Юнитом», изображенной на рис. 3, пользователь может выбрать между тремя типами связи, которые определяют, в какой «роли» информация будет передана нейросети или от

нейросети к «нейрону»: первый тип соединения (на рис. 3) подаёт значение функции «нейрона» на вход нейросети. У пользователя есть возможность создавать несколько соединений и передавать значения функций от нескольких нейронов, как массив данных, или подсоединить несколько «нейронов» к одному «входу», что идентично операции сложения.

Второй тип соединения интерпретирует передаваемую «нейроном» информацию, как целевое значение нейросети в случаях, когда тип нейросети, с которой работает пользователь, позволяет использовать целевые значения. Этот тип соединений действует по тому же принципу, что и входной тип – подключение нескольких нейронов к одному и тому же элементу эквивалентно операции сложения.

Третий тип соединения интерпретируется, как выходное значение нейросети – значения выходных нейронов нейросети будет передано подключенному «нейрону». Число элементов данного типа у одного «Обособленного Обучаемого Юнита» (справа на иконке на рис. 3) равно числу нейронов выходного слоя нейросети, а число элементов входного типа равно числу нейронов входного слоя нейросети (справа в центральном поле иконки на рис. 3). Число элементов целевых значений равно числу входных значений по очевидным причинам. Такая организация позволяет пользователям создавать нестандартные конфигурации нейросетей, например – самообучаемые нейросети с несколькими значениями награды.

В центре иконки представлены точечные элементы, визуализирующие нейроны всех слоев нейросети (в центре иконки юнита на рис. 3). Пользователь может менять число нейронов в каждом слое посредством треугольных интерактивных элементов сверху и снизу каждого ряда точек, а также число слоёв посредством таких же интерактивных элементов слева и справа на иконке. По выбору пользователя, эти точечные элементы могут оставаться однотонными или менять яркость в зависимости от того, какое значение выдают нейроны каждого слоя нейросети в процессе онлайн-работы в каждый момент времени. Эта опция, как и все другие настройки «Обособленных Обучаемых Юнитов», доступна в меню настроек, которое позволяет пользователю менять все существенные параметры нейросети, начиная от скорости обучения и заканчивая применяемыми алгоритмами оптимизации. Также, в зависимости от типа юнита, могут быть доступны дополнительные опции – например, число элементов буфера памяти для самообучающихся нейросетей, реализующих метод DDPG и подобные методы. Я более подробно остановлюсь на доступных пользователю параметрах далее в тексте.

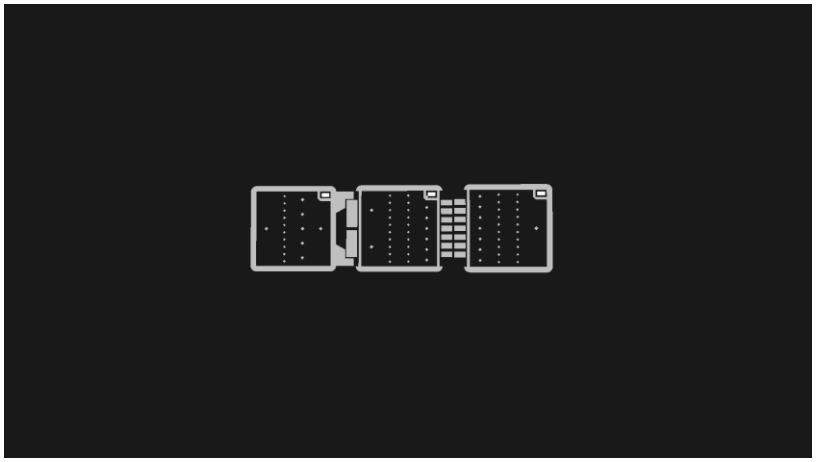

Рис. 4. Цепь "Обособленных Обучаемых Юнитов".

Как и «нейроны», обучаемые юниты могут быть подключены друг к другу посредством связей. В таком случае, значение или массив значений нейронов выходного слоя одной нейросети будут подаваться на вход или как целевые значения другой нейросети. Однако, обучаемые юниты могут также быть объединены в «цепи», как та, что

изображена на рис. 4. При расположении нескольких юнитов последовательно друг за другом, массив выходных значений выходного слоя нейронов одной нейросети также будет передан на вход другой. Разница между этими двумя типами соединений состоит в том, что «цепи» обучаемых юнитов функционируют и в режиме обучения. При тренировке нейросетей посредством метода градиентного спуска, градиенты, полученные на входе нейросети справа, будут переданы нейросети слева. Это позволяет пользователю создавать нейросети со слоями, выполняющими разнообразные функции и обучаемые с разными параметрами – каждый юнит и его внутренние слои будут иметь свою скорость обучения, обновление значений весов между внутренними слоями каждого юнита может быть выключено и т.д.

При объединении юнитов в цепь все включенные в неё юниты будут также иметь параметры, общие для всей цепи. Так, например, при обучении с применением минибэтчей (minibatch learning), число элементов в массиве данных для обучения будет общим для всех юнитов. Здесь я подробнее остановлюсь на принципе действия нескольких типов «Обособленных Обучаемых Юнитов» в цепи.

При включении юнита «Ученик» в цепь все юниты справа от него будут интерпретироваться как учитель или критик, состоящий из одного или нескольких юнитов с одним или несколькими внутренними слоями в каждом. Все юниты слева от «Ученика» также будут считаться частью ученика или актора («Actor»). Этот принцип организации визуализации позволяет не только работать с нейросетевыми архитектурами в удобном эргономичном виде, но и даёт пользователю интуитивное понимание того, как они будут функционировать.

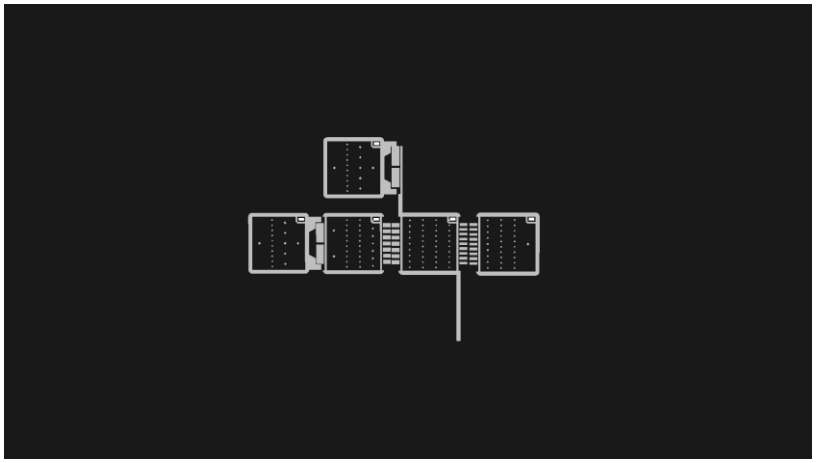

Рис. 5. Цепь юнитов с комбинирующим юнитом (третий по счету слева).

При включении в цепь «Комбинирующего Юнита», как та, что изображена на рис. 5, значения выходных слоёв нейросетей из двух независимых цепей объединяются в один массив данных, а градиенты, полученные в результате применения метода градиентного спуска, будут разделены на два массива данных соответственных размеров и переданы двум цепям слева от комбинирующего юнита. Схожий принцип действия осуществляется при подсоединении двух цепей на выходе юнита – выходные значения одной или двух цепей будут объединены в один массив, обработаны нейронами внутренних слоёв комбинатора, если таковые имеются, и затем разделены между двум цепями справа, размер каждого массива равный числу нейронов входного слоя в каждой из двух подключённых цепей. Такая система организации не только позволяет реализовывать нестандартные нейросетевые архитектуры (например, одна цепь (слева от комбинатора) может получать градиенты от двух независимых нейросетей со своими целевыми значениями (справа от комбинатора), но и реализовывать методы самообучения из «семейства» DDPG, требующие нескольких параллельно действующих нейросетей «учителей» или «учеников».

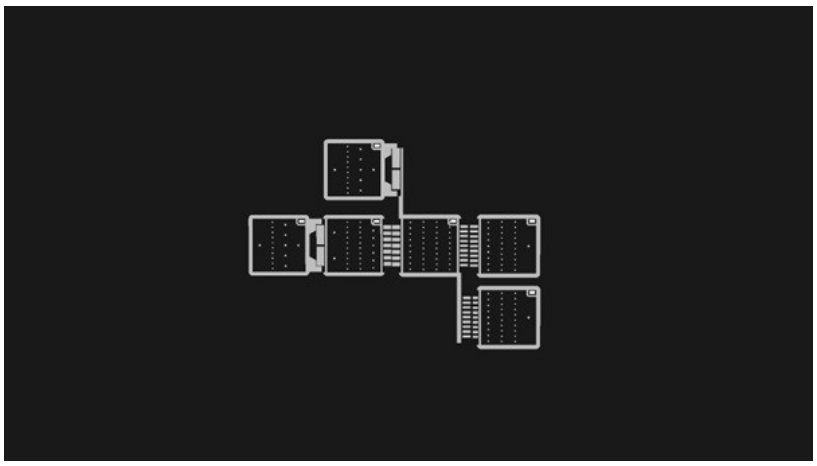

Рис. 6. Цепь юнитов, реализующая метод EAC.

Так, например, представленная на рис. 6 цепь реализует метод EAC (Explorer Actor-Critic), использующий архитектуру с двумя независимо обучаемыми «учениками» и «учителями». При подключении только одной цепи акторов, эта же конфигурация будет реализовывать метод TD3(Temporally Delayed Deterministic Gradients). Данный принцип визуализации нейросетевой архитектуры этих методов позволяет пользователю интуитивно понять, какие преобразования происходят с данными внутри алгоритма обучения, и как блоки информации передаются между составляющими его нейросетями.

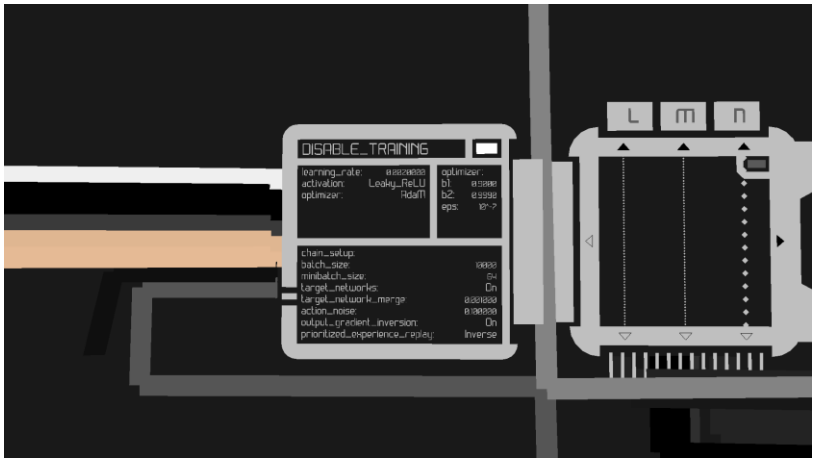

Рис. 7. Меню настроек обучаемого юнита.

«Обособленные Обучаемые Юниты» позволяют пользователю менять большинство существенных параметров обучения нейросети, а также применять широкий спектр методов оптимизации процесса обучения (меню изображено на рис. 7):

1. Learning rate (скорость обучения) – определяет скорость обновления значений весов и байесов нейронов нейросети на каждом шаге обучения;
2. Activation (функция активации) – определяет функцию активации нейронов внутренних слоёв нейросети. Доступен широкий набор типовых функций – ReLU, SoftMax, TanH, Sigmoid и т.д.;
3. Optimizer (Оптимизатор) – применяет к нейросети методы оптимизации моментов или дельты обновления весов нейросети – доступные методы: Momentum, AdaDelta, AdaGrad, AdaM, NadaM. Включение этой опции делает доступным дополнительное меню (на рис. 7 справа сверху иконки), в котором пользователь может редактировать их параметры;

4. Batch size (Размер бэтча) – параметр, общий для всех юнитов в сети, определяющий размер буфера памяти при обучении методами «семейства» DDPG. Когда параметр равен единице, обучение происходит без буфера памяти, в режиме онлайн;
5. Minibatch size (Размер минибэтча) – параметр, общий для всех юнитов в сети, задающий размер минибэтча. Когда параметр равен единице, обучение происходит на блоке данных из одного элемента на каждом шаге процесса;
6. Target networks (Целевые нейросети) - параметр, общий для всех юнитов в сети, специальный для методов машинного самообучения DDPG и подобных, добавляющий целевую нейросеть к «ученику» и «учителю», постепенно приращиваемую к основным и использующуюся для стабилизации обучения «учителя»;
7. Target networks merge (Приращение целевых нейросетей) – параметр, специальный для методов, использующих целевые нейросети. Параметр определяет степень приращения целевых сетей после каждого шага обучения;
8. Action noise (шум актора) – определяет амплитуду псевдослучайного значения, добавляемого на выход нейросети-ученика. Параметр специален для методов самообучения на основе пары нейросетей Actor-Critic;
9. Output gradient inversion (инверсия выходного градиента) – добавляет инверсию градиента выходного слоя нейросети при превышении максимального абсолютного значения;
10. Prioritized experience replay (приоритизированный проигрыш памяти) – метод оптимизации отбора элементов массива данных для обучения из буфера памяти методов «семейства» DDPG. Доступны три опции: первая опция – стандартная реализация метода, вторая и третья опции – разработанные мной альтернативные методы, показавшие сравнительно превосходную эффективность в серии экспериментов. Стандартный метод приоритизирует примеры для обучения с максимальным значением награды, тогда как созданные нами методы приоритизирует примеры с наименьшим значением награды и наибольшей девиацией от среднего значения награды.

При работе с «Обособленными Обучаемыми Юнитами» пользователь также имеет возможность подключать «нейроны» к специальным элементам юнитов, позволяющим «нейронам» менять параметры нейросети в зависимости от передаваемого ими значения. Эти элементы расположены сверху иконки юнита и в общем случае предоставляют возможность менять значение скорости обучения, амплитуду шума у самообучающихся нейросетей, а также блокировать обновление значений весов и байесов нейросети юнита.

ПО позволяет пользователю реализовывать широкий спектр современных методов машинного обучения, применение которых основано на результатах многочисленных исследований (1, 2, 3, 4, 5, 6 , 7, 8, 9, 10). Разработка методов самообучения на основе DDPG опиралась на работы ряда российских и зарубежных ученых (11, 12, 13, 14, 15, 16). Результатом является программная реализация методов DDPG и TD3, функционально идентичная поведению оригиналов, но отличающаяся в принципе организации данных буфера памяти и минибэтчей, что позволяет экономить объём хранимых данных при работе с множеством тестовых нейросетевых архитектур.

Как видно из списка выше, мой интерфейс обеспечивает пользователю доступ к большинству существенных для обучения нейросетей параметров и позволяет высокую степень гибкости в дизайне нейросетевых архитектур и методов. Эти параметры типичны для всех типов обучаемых юнитов. Следует отдельно отметить два специальных типа юнитов, обладающих уникальными функциями и параметрами:

1. «Обучаемый юнит с памятью» (тринадцатая иконка по счету слева на рис. 8) – юнит, сохраняющий массивы входных значений с предыдущих циклов и подающий их последовательно на вход своей нейросети, таким образом позволяя ей «помнить» состояния из произвольного определяемого пользователем числа предыдущих циклов;
2. «Запутанный обучаемый юнит» (четырнадцатая иконка по счету слева на рис. 8) - запутанная нейросеть, эффективная при работе с данными «многоуровневой» струк-

туры анализируемой информации (например, с изображениями). Юнит применяет нейросеть с размером входного слоя в несколько раз меньше числа входов юнита, последовательно, к каждой секции входного массива данных юнита.

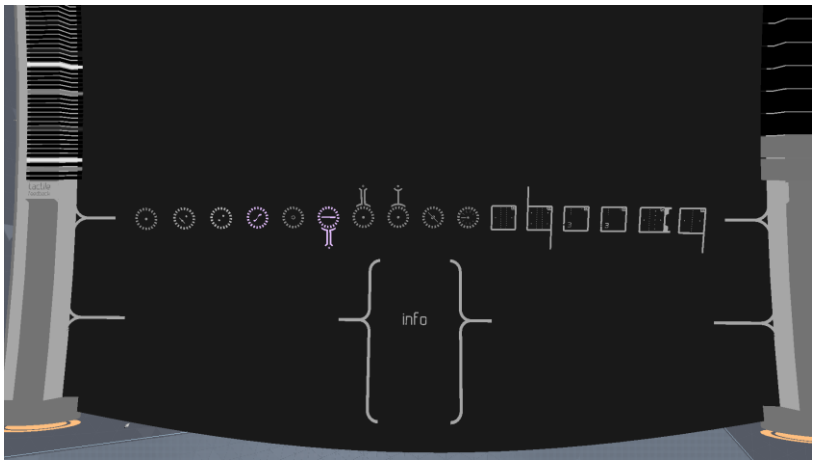

Рис. 8. Меню выбора объектов редактора.

Оба типа юнитов имеют дополнительный параметр, позволяющий пользователю определять число входных сегментов для запутанной нейросети и число запоминаемых предыдущих состояний для юнита с памятью.

4. Сравнительный анализ вычислительных возможностей программы

Мной был проведён сравнительный анализ моей программной реализации метода DDPG посредством данного интерфейса с идентичной с точки зрения нейросетевой архитектуры и параметров реализации на основе библиотеки машинного обучения TensorFlow. Данная контрольная реализация на языке Python выполнена студентом MIT Стивеном П. Спилбергом (Steven Spielberg P) (17). Его реализация использует стандартный алгоритм DDPG и показывает результаты эффективности обучения, схожие со стандартными. Она часто приводится пользователями онлайн-форумов, как шаблон для разработки собственных реализаций методов самообучения. Обе реализации использовали исключительно основной процессор компьютера без использования графического процессора. Обе реализации показали идентичные результаты при решении типовых экспериментальных задач машинного обучения библиотеки Mujoco, а именно: обратный маятник, маятник на тележке, Half-cheetah, Reacher3D. Обучение происходило в режиме онлайн, с частотой один цикл обучения на один шаг тестовой среды. После 10,000 тысяч циклов и после 100,000 циклов обучения, существенной разницы в среднем значении награды за один цикл не наблюдалось. Однако, моя реализация показала сравнительно лучшую производительность, в среднем выполнив на 12% больше циклов обучения, чем реализация TensorFlow за одинаковое время. В будущем, я планирую расширить возможности моей реализации и использовать в том числе ресурсы графических процессоров.

5. Экспериментальные возможности программы

Интерфейс позволяет пользователям реализовывать сложные, «составные» математические функции посредством «нейронов» и таким образом составлять задачи для решения нейросетями. В будущем, предполагается расширение этих возможностей посредством реализации «вложенных юнитов» - блоков из множества «нейронов», объединенных в один юнит и занимающих одну клетку, с несколькими входными и выходными элементами, подобно «Обособленным Обучаемым Юнитам». Для работы с

самообучающимися нейросетями и решения задач механической природы, мной разработана виртуальная среда с симуляцией физических законов на основе Ньютоновской механики. Эта среда позволяет пользователю тестировать методы самообучения на типовых виртуальных моделях с жесткими телами, а также разрабатываемых пользователями моделях с гибкими телами.

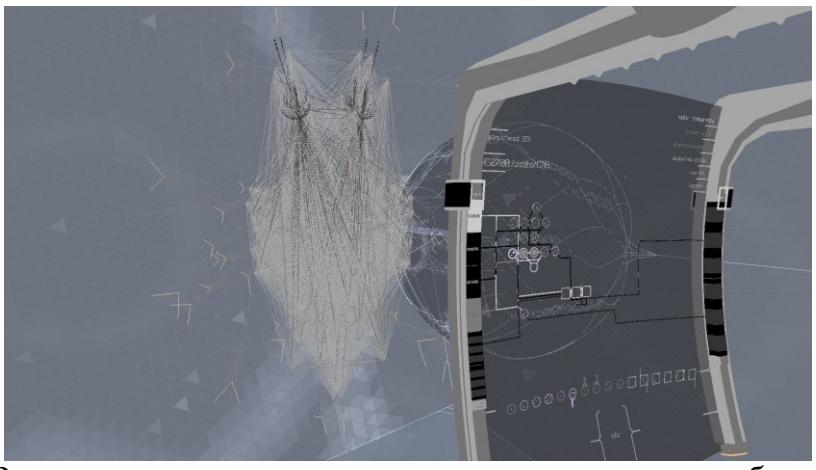

Рис. 9. Редактор и виртуальная механическая модель с гибким телом.

При работе с виртуальными моделями (слева на рис 9.), взаимодействующими с симуляцией физической среды, входные и выходные значения, передающие данные виртуальных сенсоров и активирующие виртуальные активаторы (приводы) представлены в редакторе, как два блока панелей, действующих подобно входным и выходным элементам «Обособленных Обучаемых Юнитов» слева и справа от главной рабочей поверхности редактора (редактор изображен справа на рис. 9).

Такие виртуальные механические модели позволяют реализовывать как типовые эксперименты самообучающихся сетей (маятник, обратный маятник, маятник на тележке и т.д.) из баз экспериментальных сред типа Mujoco, так и уникальные механические модели из облака точек с гибкими структурными связями между точками.

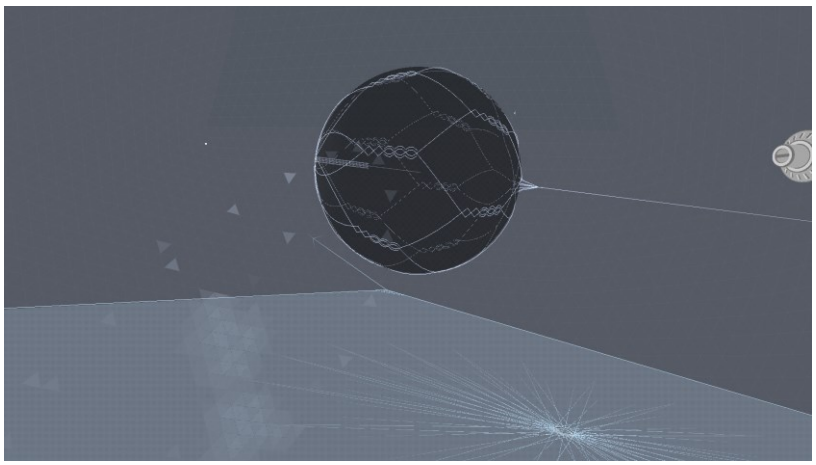

Рис. 10. Геометрические примитивы тестовой среды.

Среда также позволяет реализовывать геометрические примитивы – сферы (сверху в центре на рис. 10), плоскости (снизу в центре на рис 10.) и их комбинации. Создание такой сложной и многофункциональной среды для экспериментов отчасти обусловлено тем, что моя программа написана на основе моей собственной библиотеки машинного обучения и не может использовать стандартизированные среды типа Mujoco. Это ограничение можно охарактеризовать как недостаток, однако депрекация, прекращение поддержки или в целом доступа к стандартным средам типа AI-gym и Mujoco по-

следних лет показывает, что использование сторонних программных продуктов для экспериментов влечет за собой серьезный риск утраты функциональности по прошествии нескольких лет и даже месяцев. Таким образом, полагаясь на мою собственную экспериментальную среду, обеспечивается независимость и долголетие моего программного продукта как инструмента для исследования и обучения.

При разработке экспериментальной среды, в расчет принимались экспериментальные результаты работ, направленных на анализ потенциала онлайн-обучения в среде с высокой степенью зашумленности сигнала (18, 19, 20, 21, 22, 23), что особенно существенно, применительно к виртуальным механическим моделям с гибкими телами. В расчет также принимались результаты исследований технологий передачи сигнала (15, 23). Трехмерная визуализация экспериментальной среды разрабатывалась и будет в дальнейшем обновляться с опорой на работу «Основные принципы построения визуальных моделей данных на примере интерактивных систем трехмерной визуализации» А.А. Захаровой (24). В рамках проекта был также разработан механизм синхронной передачи сигналов между «нейронами», обучаемыми юнитами и объектами среды в пределах одного шага тестовой среды.

При работе в экспериментальной среде, для наблюдения за тестовыми моделями, пользователь может использовать свободно передвигающуюся камеру обзора, управляемую посредством мыши и клавиатуры. Несколько тестовых моделей могут быть загружены одновременно, тестирование, тренировка и редактирование моделей и нейросетевых решений может проводиться в виртуальном пространстве ПО параллельно. Данная реализация позволяет визуализировать большое число физических объектов (до 64 одновременно загруженных параллельно обучаемых шаблонных моделей и до 32 параллельно обучаемых моделей с гибкими телами из облака точек) и создана, полагаясь на принципы эффективной трехмерной визуализации данных, описанных П. Васёвым и С. Поршневым в статье «An Experience of Using Cinemascience Format for 3D Scientific Visualization» (25).

ПО позволяет отслеживать, просматривать, регистрировать и сравнивать экспериментальные результаты, представляя регистрируемые значения награды самообучающихся нейросетей, целевые значения «Обособленных Обучаемых Юнитов» посредством графов. Представление графической информации реализуется в соответствии с основными принципами формализованной визуализации, описанными Д. Манаковым и В. Авербухом в статье «Верификация визуализации» (26), а именно: добавлена опция нормализации информации, представляемой на графах, опция создания сплайнов, отображения как абсолютного значения награды или целевого значения нейросети, так и его первой производной. Также, добавлена опция отображения последних 50%, 25% и 5% каждого цикла обучения, для уменьшения влияния на визуализацию данных об эффективности процесса обучения шума от рандомизации начальных условий каждого цикла симуляции.

6. Формальное описание ПО.

Данный программный продукт разработан на языке C++ в среде разработки Visual Studio (Visual Studio 2019) для платформы Windows PC (Windows 10+, 64bit). Для эффективных матричных вычислений при работе ИИ и обработке физической симуляции использовалась библиотека Armadillo. Для реализации объемного звука в экспериментальной среде использовалась библиотека OpenAL. Для реализации графического интерфейса, использовалась библиотека OpenGL. Данное ПО является коммерческим проектом и зарегистрировано на платформе Steam. В настоящее время, готовится выпуск бета-версии ПО на рынок.

Перечень функциональных возможностей ПО:

1. Создание, редактирование и тестирование логических, математических цепей и нейросетей посредством графического пользовательского интерфейса;

2. Разработка комплексных, модульных нейросетевых архитектур с независимыми параметрами обучения для разных модулей посредством графического интерфейса;

3. Обучение нейросетей, в том числе онлайн-самообучение посредством взаимодействия с экспериментальной средой;

4. Экспериментальная среда, реализующая физическую симуляцию механических систем на основе Ньютоновской механики – для обучения методов DDPG и подобных;

5. Реализация методов DDPG, TD3, EAC самообучения, нейросетей типа feed-forward, адверсариальных и запутанных нейросетей. Реализация разработанных пользователем методов обучения и нейросетевых архитектур посредством графического интерфейса и блочного программирования с высокой степенью свободы эксперимента;

6. Визуализация активности нейросетей при онлайн-работе: визуализация активности нейронов нейросетей, юнитов логических цепей, активности входных и выходных слоёв нейросетей, значений награды и целевых значений;

7. Реализация трёхмерных экспериментальных сред из графических примитивов, возможность обзора экспериментальной среды посредством «свободной» камеры, с управлением посредством мыши и клавиатуры;

8. Возможность присвоения выбранных пользователем клавиш для управления функциями ПО;

9. Возможность создания виртуальных механических моделей для тестирования методов самообучения посредством встроенного редактора;

10. Возможность загрузки и сохранения файлов нейросетевых архитектур, виртуальных механических моделей и экспериментальных сред, возможность автоматической генерации нейросетевых архитектур для типовых задач на основе встроенных шаблонов;

11. Возможность регистрации, просмотра и сохранения результатов экспериментов – динамики значений награды (в виде графов), выполнения тестовых критериев эксперимента, определяемых пользователем.

7. Потенциальные применения разработки

Данная интерактивная визуализация нейросетевых архитектур и экспериментальных сред может применяться для обучения работы с искусственным интеллектом и основных принципов данной области исследования как специалистами смежных сред, не имеющими опыта работы с ИИ, так и простыми обывателями. Программа предлагает подробный глоссарий, серию инструкций и типовых задач и готовых реализаций нейросетевых архитектур. Возможна автоматическая генерация решения для механических моделей посредством заранее созданных разработчиком или пользователями шаблонов. После завершения работы над программой и выпуска её на рынок, я планирую выложить в открытый доступ библиотеку машинного обучения, что позволит использовать данную разработку в других проектах. Данная визуализация и сопутствующий редактор нейросетей является уникальным, эргономичным и визуально индикативным способом репрезентации абстрактных, часто контринтуитивных процессов и математических методов, на момент создания данной статьи данная программа не имеет аналогов на рынке.

8. Значимость исследования

На текущий момент, большинство научных достижений и показательных технологических результатов в области искусственного интеллекта совершаются крупнейшими зарубежными и интернациональным корпорациями и принадлежащими им исследовательскими группами. Моя программная реализация и подобные ей проекты могут «демократизировать» данное поле деятельности и обеспечить как конкурентные условия в затрагиваемой ИИ рыночной нише, так и возможность участия в разработке, продвижении и развитии данных технологий небольшими командами и независимы-

ми разработчиками, что в том числе позволит более активно внедрять данные технологии в область креативных индустрий – интерактивные медиа, онлайн среды, видеоигры и т.д.; позволит находить новые решения задач в этих индустриях (генерация медиа, создание независимых агентов игровых сред на основе самообучающихся нейросетей и т.д.) студиями без существенных финансовых и трудовых ресурсов.

9. Заключение

Искусственный Интеллект прочно вошел во многие области научного исследования, в творческие области, во многие, часто несвязанные с IT индустрии. Число областей применения несомненно будет расти, как будет расти и число новых всё более эффективных методов. На текущий момент, на рынке не существует доступных простому обывателю и специалистам смежных сфер исследования программных продуктов и визуализаций, дающих схожий набор функций и возможностей с низкоуровневыми библиотеками машинного обучения, требующими навыки кодинга.

Мной создано программное решение, визуализирующее нейросетевые архитектуры, математические функции и позволяющее пользователю работать с ними и проектировать собственные архитектуры посредством эргономичного минималистичного графического интерфейса. Данная программная реализация предлагает методы машинного обучения, зарекомендовавшие себя как наиболее эффективные и универсальные – в том числе методы, созданные в последние годы и месяцы (EAC). С точки зрения вычислительной эффективности, программа показала себя почти идентично со стандартной реализацией на базе TensorFlow, с небольшим преимуществом.

Разработанная визуализация может применяться как средство обучение специалистов, неспециалистов, хоббиистов, а также для экспериментов и разработки новых нейросетевых архитектур и методов управления механическими моделями. Эта и подобные реализации может способствовать демократизации и внедрению технологий ИИ в различные области научного поиска и цифровых индустрий.

Список литературы

1. Deisenroth M., Rasmussen C.E. A model-based and data-efficient approach to policy search / // Proceedings of the 28th International Conference on machine learning, 2011, pp. 465–472.E
2. Foster D.J., Matthew A. Reverse replay of behavioural sequences in hippocampal place cells during the awake state // Nature, Vol. 440, 2006, pp. 680–683.
3. Glascher J., Nathaniel D, Peter D. States versus rewards: Dissociable neural prediction error signals underlying model-based and model-free reinforcement learning // Neuron, Vol. 66, 2010, pp. 585–595.
4. Van Hasselt H. Double Q-learning // Advances in Neural Information Processing Systems, 2010, pp. 2613–2621.
5. Hinton G. E. To recognize shapes, first learn to generate images // Progress in brain research, Vol. 1652007, 2015, pp. 535–547.
6. Koutnık J., Schmidhuber J., Faustino G. Online evolution of deep convolutional network for vision-based reinforcement learning // From Animals to Animats, Vol. 13, 2014. pp. 260–269.
7. Koutnık J., Schmidhuber J., Faustino G. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning // Proceedings of the 2014 conference on Genetic and evolutionary computation, 2014, pp. 541–548.
8. Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems, 2012, pp. 1097–1105.
9. Larochelle H., Murray I. The neural autoregressive distribution estimator // AISTATS, Vol. 6, 2011, p. 622.

10. Heess N., Wayne Gr., Silver D. Learning continuous control policies by stochastic value gradients // Advances in Neural Information Processing Systems, 2015. pp. 2926–2934.
11. Wawrzynski P., Tanwani A.K. Autonomous reinforcement learning with experience replay // Neural Networks, Vol. 41, 2013. pp. 156–167.
12. Wawrzynski P. Real-time reinforcement learning by sequential actor–critics and experience replay // Neural Networks, Vol. 22, 2009, pp. 1484–1497.
13. Watkins Chr., Peter D. Q-learning // Machine learning, Vol. 8, 1992. pp. 279–292.
14. Hafner R. Riedmiller Martin Reinforcement learning in feedback control // Machine learning. Vol. 84, 2011, pp. 137–169.
15. Lecun Yann, Bottou L., Bengio Y. Gradient-based learning applied to document recognition // Proceedings of the IEEE, Vol. 86, 1998. pp. 2278–2324.
16. Spielberg P.S. Continuous control with deep reinforcement learning - Deep Deterministic Policy Gradient (DDPG) algorithm implemented in OpenAI Gym environments // Github. (https://github.com/stevenpjg/ddpg-aigym)
17. Schulman J., Mnih V., Kavukcuoglu K., Silver D. Gradient estimation using stochastic computation graphs // Advances in Neural Information Processing Systems, 2015, pp. 3510–3522.
18. Van Seijen Harm, Sutton R. Planning by prioritized sweeping with small backups // Proceedings of The 30th International Conference on Machine Learning, 2013. pp. 361–369.
19. Sun Yi, Ring M., Schmidhuber J. Incremental basis construction from temporal difference error // Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 481–488.
20. Todorov E., Erez T., Tassa Y.. A physics engine for model-based control // Intelligent Robots and Systems, 2012, pp. 5026– 5033.
21. Uhlenbeck G.E. Ornstein L.S. On the theory of the brownian motion // Physical review. Vol. 36, 1930, pp. 47-51.
22. Wawrzynski P. Control policy with autocorrelated noise in reinforcement learning for robotics // International Journal of Machine Learning and Computing, Vol. 5, 2015. pp. 91–95.
23. Guo Xiaoxiao, Satinder S., Lee H. Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning // Advances in Neural Information Processing Systems. Vol. 27, 2014. pp. 3338–3346.
24. Захарова А.А., Шкляр А. Основные принципы построения визуальных моделей данных на примере интерактивных систем трехмерной визуализации // Научная визуализация. Vol. 6, 2014. 2. pp. 62–73.
25. Vasev P., Porshnev S., Forghani M., Manakov D., Bakhterev M., Starodubtsev. I. An Experience of Using Cinemascience Format for 3D Scientific Visualization // Научная визуализация. Vol. 13, 2021. 4, pp. 127–143.
26. Манаков Д., Авербух В. Верификация визуализации // Научная визуализация. Vol. 8, 2016. 1. pp. 58–94.

Scientific Visualization, 2025, volume 17, number 2, pages 123 - 136, DOI: 10.26583/sv.17.2.09

Visualization of Methods of Machine Learning. GUI Programming

N.O. Shesterin[1]

National research university "HSE" (Higher School of Economics), Moscow, Russia

[1] ORCID: 0000-0003-2134-8412, nshesterin@gmail.com

Abstract

The technologies of artificial intelligence and machine learning have made a fundamental leap in their capabilities in the last five years. The growth of processing power and the emergence of more and more effective methods of machine learning allows AI to not just solve the most typical tasks associated with the field, such as statistical analysis and optimization of mathematical processes, but also to find new applications in related fields of research, as well as practical applications, including those on the free market, available to the mass consumer. Image generation, audio, animation, self-learning models of control of robotic platforms and virtual mechanical models – these and many more novel applications of the recent years have led to a media-boom around AI and a growing interest from developers and authors from various fields and industries.

That being said, the methods for developing, research, testing, and integration of AI have largely remained unchanged and still require the knowledge of programming languages, machine learning libraries, as well as a deep understanding and experience specifically in the narrow field of AI. This barrier of specialization not only demands inclusion of machine learning specialists in the development process of otherwise trivial computer applications, typical for the field of AI, but also prevents small teams and independent developers from using the latest advances in these technologies without significant monetary and time investments into studying the subject.

I offer a novel solution to this issue in the form of a prototype graphical interface that allows the user without technical education and without the need for knowledge of programming languages to develop and tune various architectures of neural nets and other machine learning methods, methods of unsupervised machine learning, and to test these methods on a wide range of experimental tasks – from mathematical equations to controlling virtual mechanical models in a simulated physical environment. In this article, I give a brief description of its structure and organisation, its fundamental principles of operation, and the capabilities of this GUI.

Keywords: neural net, artificial intelligence, machine learning, block programming, graphic interface.

References

1. Deisenroth M., Rasmussen C.E. A model-based and data-efficient approach to policy search // Proceedings of the 28th International Conference on machine learning, 2011, pp. 465–472.E
2. Foster D.J., Matthew A. Reverse replay of behavioural sequences in hippocampal place cells during the awake state // Nature, Vol. 440, 2006, pp. 680–683.
3. Glascher J., Nathaniel D, Peter D. States versus rewards: Dissociable neural prediction error signals underlying model-based and model-free reinforcement learning // Neuron, Vol. 66, 2010, pp. 585–595.
4. Van Hasselt H. Double Q-learning // Advances in Neural Information Processing Systems, 2010, pp. 2613–2621.

5. Hinton G. E. To recognize shapes, first learn to generate images // Progress in brain research, Vol. 1652007, 2015, pp. 535–547.
6. Koutnık J., Schmidhuber J., Faustino G. Online evolution of deep convolutional network for vision-based reinforcement learning // From Animals to Animats, Vol. 13, 2014. pp. 260–269.
7. Koutnık J., Schmidhuber J., Faustino G. Evolving deep unsupervised convolutional networks for vision-based reinforcement learning // Proceedings of the 2014 conference on Genetic and evolutionary computation, 2014, pp. 541–548.
8. Krizhevsky A., Sutskever I., Hinton G.E. Imagenet classification with deep convolutional neural networks // Advances in neural information processing systems, 2012, pp. 1097–1105.
9. Larochelle H., Murray I. The neural autoregressive distribution estimator // AISTATS, Vol. 6, 2011, p. 622.
10. Heess N., Wayne Gr., Silver D. Learning continuous control policies by stochastic value gradients // Advances in Neural Information Processing Systems, 2015. pp. 2926–2934.
11. Wawrzynski P., Tanwani A.K. Autonomous reinforcement learning with experience replay // Neural Networks, Vol. 41, 2013. pp. 156–167.
12. Wawrzynski P. Real-time reinforcement learning by sequential actor–critics and experience replay // Neural Networks, Vol. 22, 2009, pp. 1484–1497.
13. Watkins Chr., Peter D. Q-learning // Machine learning, Vol. 8, 1992. pp. 279–292.
14. Hafner R. Riedmiller Martin Reinforcement learning in feedback control // Machine learning. Vol. 84, 2011, pp. 137–169.
15. Lecun Yann, Bottou L., Bengio Y. Gradient-based learning applied to document recognition // Proceedings of the IEEE, Vol. 86, 1998. pp. 2278–2324.
16. Spielberg P.S. Continuous control with deep reinforcement learning - Deep Deterministic Policy Gradient (DDPG) algorithm implemented in OpenAI Gym environments // Github. (https://github.com/stevenpjg/ddpg-aigym)
17. Schulman J., Mnih V., Kavukcuoglu K., Silver D. Gradient estimation using stochastic computation graphs // Advances in Neural Information Processing Systems, 2015, pp. 3510–3522.
18. Van Seijen Harm, Sutton R. Planning by prioritized sweeping with small backups // Proceedings of The 30th International Conference on Machine Learning, 2013. pp. 361–369.
19. Sun Yi, Ring M., Schmidhuber J. Incremental basis construction from temporal difference error // Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 481–488.
20. Todorov E., Erez T., Tassa Y.. A physics engine for model-based control // Intelligent Robots and Systems, 2012, pp. 5026– 5033.
21. Uhlenbeck G.E. Ornstein L.S. On the theory of the brownian motion // Physical review. Vol. 36, 1930, pp. 47-51.
22. Wawrzynski P. Control policy with autocorrelated noise in reinforcement learning for robotics // International Journal of Machine Learning and Computing, Vol. 5, 2015. pp. 91–95.
23. Guo Xiaoxiao, Satinder S., Lee H. Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning // Advances in Neural Information Processing Systems. Vol. 27, 2014. pp. 3338–3346.
24. Zakharova A., Shklyar A. Basic principles of data visual models consruction, by the example of interactive systems for 3D visualization// Scientific visualization. Vol. 6, 2014. 2. pp. 62–73.
25. Vasev P., Porshnev S., Forghani M., Manakov D., Bakhterev M., Starodubtsev. I. An Experience of Using Cinemascience Format for 3D Scientific Visualization // // Scientific visualization. . Vol. 13, 2021. 4, pp. 127–143.
26. Manakov D., Averbukh V. Verification of visualization // // Scientific visualization. Vol. 8, 2016. 1. pp. 58–94.