УДК 004

Реализация в Unity нейронных сетей для обучения объектов прохождению фарватера

Радионов Сергей Владимирович
Приамурский государственный университет им. Шолом-Алейхема
студент

Баженов Руслан Иванович
Приамурский государственный университет им. Шолом-Алейхема
к.п.н., доцент, зав. кафедрой информационных систем, математики и правовой информатики

Аннотация
В данной статье разработана программа, позволившая обучить нейронную сеть управлять объектами для прохождения фарватера, обходя препятствия и избегая столкновений. В ней наглядно представлен процесс и результат обучения в 3D графике. Также есть возможность ускорять обучения до 100 раз, чтобы обучение занимало как можно меньше времени.
Ключевые слова: нейросеть, машинное обучение, Unity, C#, ML Agents.

Implementation of neural networks in Unity for the training of objects passing the fairway

Radionov Sergey Vladimirovich
Sholom-Aleichem Priamursky State University
student

Bazhenov Ruslan Ivanovich
Sholom-Aleichem Priamursky State University
associate professor, head. Department of Information Systems, Mathematics and Law Informatics

Abstract
In this article, a program has been developed that has made it possible to train a neural network to control objects to pass the fairway, avoiding obstacles and avoiding collisions. It clearly shows the process and result of learning in 3D graphics. Also, there is the opportunity to accelerate the training up to 100 times, so that training takes as little time as possible.
Keywords: neural network, machine learning, Unity, C #, ML Agents.

В наше время все больше и больше используются нейронные сети для решения разных задач, например, распознание лиц в видео или изображении, автопилот у автомобилей, улучшение качества изображения, игровой

искусственный интеллект и прочее. Нейронные сети стали популярными, так как позволяют решать очень сложные задачи проще, чем классический алгоритмический подход, которым некоторые задачи и вовсе решить невозможно. На данный момент нейронные сети активно продвигаются крупными компаниями в своих разработках, например, автопилоты в «Tesla» и «Google», которые позволяют водителю заниматься своими делами и принимать управление автомобилем только при определенных ситуациях, постоянно совершенствующийся переводчик Google, FaceID от компании Apple. Все это говорит о том то нейронные сети – это перспективная технология, которую нужно изучать.

Целью является разработка программы, которая позволит обучить нейронную сеть управлять объектами для прохождения фарватера, обходя препятствия и избегая столкновений. В ней будет наглядно представлен процесс и результат обучения в 3D графике. Также будет возможность ускорять обучения до 100 раз, чтобы обучение занимало как можно меньше времени.

В статье М.А. Радушкевича рассмотрен метод анализа рисков инновационных проектов на основе нейронных сетей, которая в отличие от других методик не требует сложных расчетов и учитывает смешанную природу инновационных рисков. Тщательная оценка рисков может приблизить инвесторов и менеджеров проекта к успешному внедрению инновации и получению прибыли [1]. В.А. Частикова, Д.А. Картамышев, К.А. Власов разработали методику защиты от сетевых атак типа DDoS на основе механизма работы нейронных сетей. Для поиска наиболее эффективной архитектуры нейронной сети они привели ряд исследований. Архитектура, показавшая наилучший результат – нейронная сеть вида многослойный персептрон, обучающаяся методом обратного распространения ошибки, была реализована в виде программного комплекса в среде Microsoft Visual Studio [2]. О.Т. Данилова, Е.В. Трапезников в своей статье разработали модель анализирующую функцию безопасности в системе информационной защиты, на основе нейронной сети. В работе рассматривается способ анализа реакции системы защиты информации на дестабилизирующие факторы с применением нейронной сети как универсального механизма табличного задания объектной функции многих переменных с ассоциативной или ассоциативно-усредненной выборкой [3]. Рассматривая статью Л.В. Серебряной и В.В. Потараева можно увидеть применение искусственной нейронной сети в виде персептрона, сети Хопфилда и семантической сети для классификации текстовой информации. Изучаются процедуры обучения сетей, реализуются алгоритм обратного распространения ошибки в персептроне и алгоритм сходимости сети Хопфилда. Предлагается программное средство автоматической классификации текстов на основе разработанных моделей и алгоритмов. Оцениваются результаты работы программного средства [4]. Также можно выделить и другие исследования [5-7]. Не менее значимо имитационное моделирование и в англоязычном сегменте [8-9].

Machine Learning меняет способ, которым мы ожидаем получить интеллектуальное поведение из автономных агентов. Если в прошлом поведение было закодировано вручную, его все чаще преподают агенту (либо роботу, либо виртуальному аватару) посредством взаимодействия в учебной среде. Этот метод используется для изучения поведения для всего: от промышленных роботов, беспилотных летательных аппаратов и автономных транспортных средств до игровых персонажей и противников. Качество этой учебной среды имеет решающее значение для видов поведения, которые могут быть изучены, и часто возникают компромиссы того или иного вида, которые необходимо сделать. Типичным сценарием для обучения агентов в виртуальных средах является наличие единой среды и агента, которые тесно связаны. Действия агента изменяют состояние среды и предоставляют агенту вознаграждение.

Три основных типа объектов в любой учебной среде:

1. Агент. Каждый агент может иметь уникальный набор состояний и наблюдений, принимать уникальные действия в среде и получать уникальные награды за события в среде. Действия агента определяются мозгом, с которым он связан.

2. Мозг – каждый мозг определяет конкретное состояние и пространство действий и отвечает за определение того, какие действия будут выполняться каждым из его связанных агентов.

3. Академия. Объект Академии внутри сцены также содержит в качестве детей все мозги в среде.

Настройка обучения производится с помощью изменения следующих параметров:

1. Gamma – соответствует коэффициенту дисконтирования для будущих вознаграждений. Это можно рассматривать как то, как далеко в будущем агент должен заботиться о возможных вознаграждениях. В ситуациях, когда агент должен действовать в настоящем, чтобы подготовиться к вознаграждениям в отдаленном будущем, это значение должно быть большим. В случаях, когда вознаграждение является более непосредственным, она может быть меньше.

2. Lambda – соответствует параметру лямбда, используемому при вычислении обобщенной оценки преимуществ (GAE). Это можно представить себе, насколько агент полагается на свою текущую оценку стоимости при вычислении обновленной оценки стоимости. Низкие значения соответствуют большей оценке текущей оценки (которая может быть высокой смещенностью), и высокие значения соответствуют более реальным вознаграждениям, полученным в среде (что может быть большой дисперсией). Параметр обеспечивает компромисс между ними, и правильное значение может привести к более стабильному процессу обучения.

3. Buffer Size – соответствует тому, сколько накоплений (наблюдений агентов, действий и вознаграждений) должно быть собрано, прежде чем мы будем изучать или обновлять модель. Это должно быть кратно batch_size. Обычно более высокий buffer_size соответствует более стабильным обновлениям обучения.

4. Batch Size – это количество опытов, используемых для одной итерации обновления с градиентным спусками. Это всегда должно быть частью buffer_size. Если вы используете непрерывное пространство действий, это значение должно быть большим (порядка 1000). Если вы используете дискретное пространство действий, это значение должно быть меньше (в порядке 10 секунд).

5. Number of Epochs – количество проходов через буфер опыта при смене градиента. Чем больше значение batch_size, тем больше допустимо это сделать. Уменьшение этого будет обеспечивать более стабильные обновления за счет более медленного обучения.

6. Learning Rate – соответствует силе каждого этапа обновления спуска градиента. Обычно это должно быть уменьшено, если обучение неустойчиво, и вознаграждение не постоянно увеличивается.

7. Time Horizon – соответствует количеству шагов для сбора каждого агента перед добавлением его в буфер опыта. Когда этот предел достигнут до конца эпизода, для оценки ожидаемого вознаграждения от текущего состояния агента используется оценка стоимости. Таким образом, этот параметр торгуется между менее предвзятым, но более высокая оценка дисперсии (долговременный горизонт) и более предвзятая, но менее разнообразная оценка (короткий временной горизонт). В случаях, когда частые награды в эпизоде, или эпизоды являются чрезмерно большими, меньшее число может быть более идеальным. Это число должно быть достаточно большим, чтобы фиксировать все важные действия в последовательности действий агента.

8. Max Steps – соответствует тому, сколько шагов моделирования (умноженное на кадр-пропуска) выполняется во время учебного процесса. Это значение должно быть увеличено для обучения более сложным ситуациям.

9. Beta – соответствует силе регуляции энтропии, что делает политику «более случайной». Параметр гарантирует, что агенты должным образом исследуют пространство действия во время обучения. Это приведет к более случайным действиям. Бету нужно отрегулировать так, чтобы энтропия медленно уменьшалась вместе с увеличением вознаграждения. Если энтропия падает слишком быстро, увеличьте бета-версию. Если энтропия падает слишком медленно, уменьшите бету.

10. Epsilon – соответствует допустимому порогу расхождения между старой и новой политиками при обновлении градиентного спуска. Установка этого значения мала приведет к более стабильным обновлениям, но также замедлит процесс обучения.

11. Number of Layers – соответствует тому, сколько скрытых слоев присутствует после входа наблюдения или после кодирования CNN визуального наблюдения. Для простых задач меньшее количество слоев, скорее всего, будет работать быстрее и эффективнее. Для освоения более сложных проблем управления может потребоваться больше слоев.

12. Hidden Units – соответствуют количеству единиц в каждом полностью подключенном слое нейронной сети. Для простых задач, где правильное действие представляет собой прямую комбинацию входов

наблюдения, это должно быть небольшим. Для задач, где действие представляет собой очень сложное взаимодействие между переменными наблюдения, это должно быть больше.

Для разработки приложения обучения нейросети проходить фарватер, управляя объектами выбраны следующие средства:

1. C# – объектно-ориентированный язык программирования.
2. Unity – межплатформенная среда разработки компьютерных игр.
3. Unity ML Agents – это плагин Unity с открытым исходным кодом, который позволяет игрокам и симуляторам служить средами для обучения интеллектуальных агентов.

Сначала необходимо открыть проект ML Agent в Unity, в котором содержаться примеры реализации нейронных сетей и скрипты, необходимые для их работы. Далее была создана сцена, в которой будет карта, имитирующая фарватер, и объекты в виде лодок, которыми впоследствии будет управлять нейросеть.

Для создания карты необходимо добавить на сцену объект Terrain, встроенными инструментами (Рис.1) придать ему нужные очертания, для красоты на Terrain были нанесены текстуры, загруженные бесплатно из Unity Asset Store. Это хранилище материалов для проектов Unity.

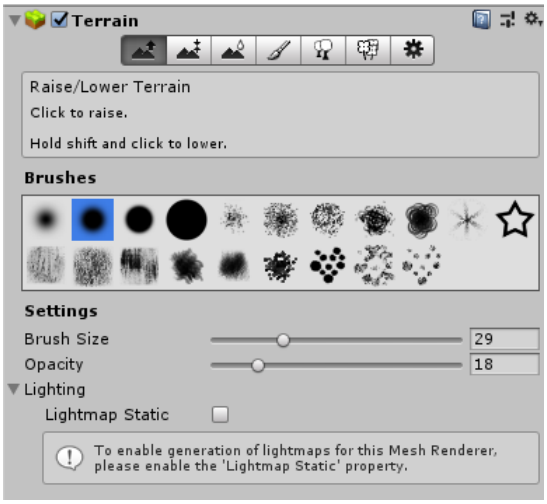

Рис.1. Инструментарий Terrain

Далее на сцене был создан Plane и размещен чуть выше Terrain, он будет имитировать воду. На него была наложена текстура из ранее загруженных (Рис.2).

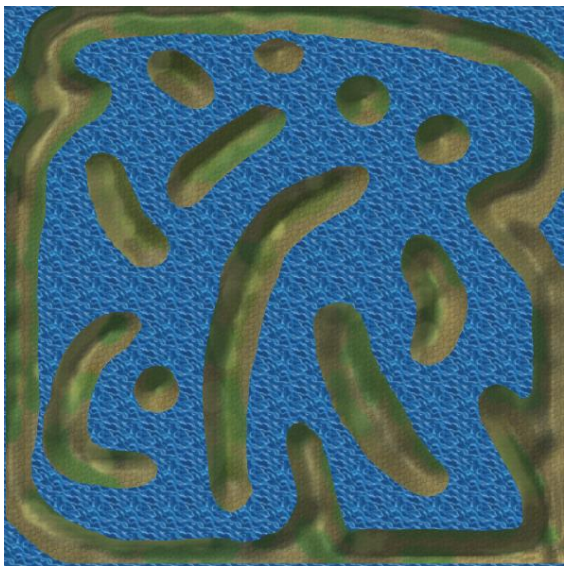

Рис. 2. Terrain с водой

Затем была создана 3D модель лодки, и импортирована в Unity. Далее был создан материал зеленого цвета, который был нанесен на лодку (Рис.3). Также на объект лодки был добавлен компонент Rigidbody, отвечающий за влияние физики на объект. Лодка была размножена до шести штук для увеличения скорости обучения.

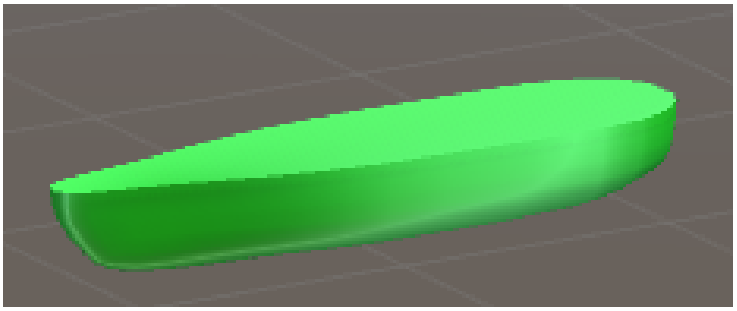

Рис. 3. 3D модель лодки

Для визуализации столкновений был создан кубик с красным материалом, который будет создаваться на месте столкновения лодки с поверхностью Terrain. В настройках физики проекта убрана возможность столкновения лодки с лодкой, благодаря чему возможно проводить обучение с несколькими объектами. На карте было создано несколько пустых объектов, которые будут являться точками возрождения. С визуальной частью закончено.

Далее идет техническая часть. Были созданы следующие скрипты:

1. ShipAcademy – скрипт, содержащий класс наследующийся от класса Academy плагина ml agents. В нашем случае в нем ничего менять не нужно.

2. ShipAgent – скрипт, отвечающий за взаимодействие агентов с мозгом.

На скрипте ShipAgent стоит остановиться поподробней. Этот скрипт содержит класс, наследуемый от класса Agent из плагина. В нем необходимо переопределить метод CollectObservations(), отвечающий за сбор входных

данных для нейросети. В этом методе необходимо задать входные данные как аргумент функции AddVectorObs(). В нашем случае это дистанция до поверхности в пяти направлениях: 0, 5, 30, -5, -30 градусов относительно переда лодки (Рис.4). Определяется она с помощью класса Raycast, в котором настроено реагирование только на слой Terrain. Также необходимо переопределить метод AgentAction(), отвечающий за действие объекта на выходные данные нейронной сети. В нашем случае это поворот лодки влево, если выходное данное меньше -0,3, и поворот вправо, если больше 0,3, в противном случае действие не требуется, и лодка будет идти вперед. Зададим награду с помощью функции SetReward() за успешный шаг 0,1, а за столкновение с поверхностью -1 и вызов метода Done(). Помимо это нужно переопределить метод AgentReset(), который определяет действие после вызова метода Done(). Здесь укажем, что нужно поставить лодку на случайную из точек возрождения.

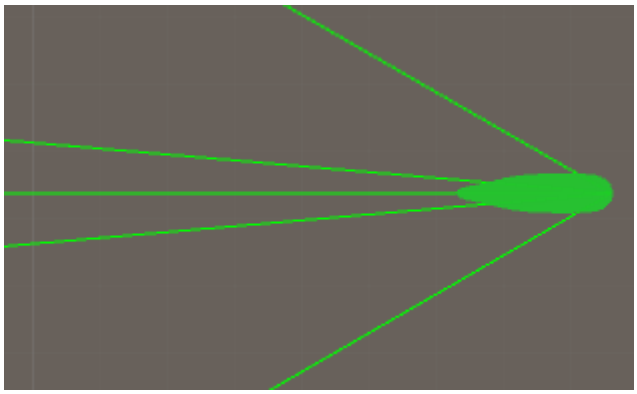

Рис. 4. Направления Raycast

На сцене был создан пустой объект «ShipAcademy», на который добавлен одноименный скрипт. Дочерним к нему создан «ShipBrain», которому присвоен скрипт Brain из плагина. Настроим скрипт под наш проект (Рис.5).

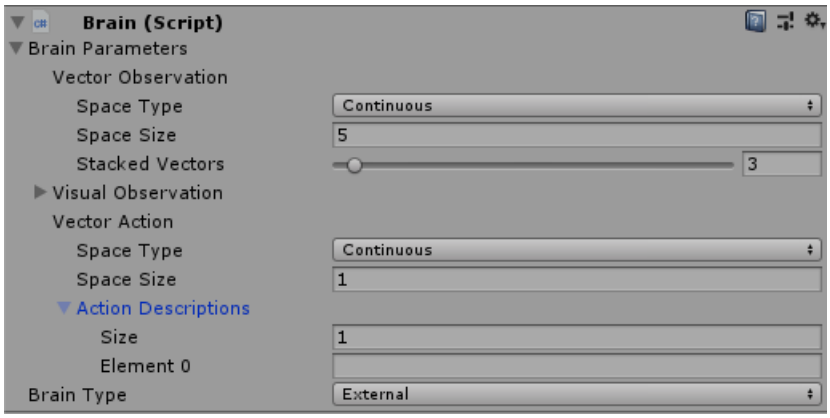

Рис. 5. Настройки Brain

Описание настроек Brain:

1. Space Type определяет какого формата данные, continuous – действительные числа.

2. Space Size – количество параметров.

3. Stacked Vectors – число предыдущих векторных наблюдений, которые будут складываться и использоваться совместно для принятия решений. Оставляем по умолчанию.

4. Brain Type – тип мозга, external – внешний, т.е. подключается к Python API.

5. Раздел Vector Observation – входные параметры.

6. Раздел Vector Action – выходные параметры.

Далее необходимо скомпилировать созданный проект и перейти к настройке обучения нейронной сети с помощью Python API.

Для обучения нейронной сети была установлена среда python, также установлены все необходимые библиотеки. Помимо этого, необходимо загрузить необходимые скрипты из раздела плагина Unity ml agents на GitHub [10].

В файле настроек обучения были заданы следующие параметры, подобранные эмпирическим способом исходя из рекомендаций разработчиков плагина:

- Batch Size – 500;
- Beta – 5,0e-3;
- Buffer Size – 5000;
- Epsilon – 0,1;
- Gamma – 0,99;
- Hidden units – 32;
- Lambda – 0,95;
- Learning rate – 3,0e-5;
- Max steps – 1,0e6;
- Number of Epochs – 7;
- Number of Layers – 2;
- Time Horizon – 248.

Для начала обучения был запущен скрипт learn.py с помощью командной строки Windows PowerShell командой «python learn.py ShipImitation –train», где ShipImitation – это название скомпилированного проекта. В консоли отобразилось, что подключение прошло (Рис.6).

```
PS C:\Users\sergo\Desktop\ml-agents-master\python> python learn.py ShipImitation --train
INFO:unityagents:{'--curriculum': 'None',
 '--docker-target-name': 'Empty',
 '--help': False,
 '--keep-checkpoints': '5',
 '--lesson': '0',
 '--load': False,
 '--run-id': 'ppo',
 '--save-freq': '50000',
 '--seed': '-1',
 '--slow': False,
 '--train': True,
 '--worker-id': '0',
 '<env>': 'ShipImitation'}
INFO:unityagents:
'ShipAcademy' started successfully!
Unity Academy name: ShipAcademy
        Number of Brains: 1
        Number of External Brains : 1
        Lesson number : 0
        Reset Parameters :

Unity brain name: ShipBrain
        Number of Visual Observations (per agent): 0
        Vector Observation space type: continuous
        Vector Observation space size (per agent): 5
        Number of stacked Vector Observation: 1
        Vector Action space type: continuous
        Vector Action space size (per agent): 1
```

Рис. 6. Результат ввода команды в консоль

После этого открылось созданное ранее приложение и начался процесс обучения (Рис.7). В это время в консоль выводились данные о состоянии обучения, а именно средняя награда агентов. Для наглядности по этим данным составлен в график (Рис.8)

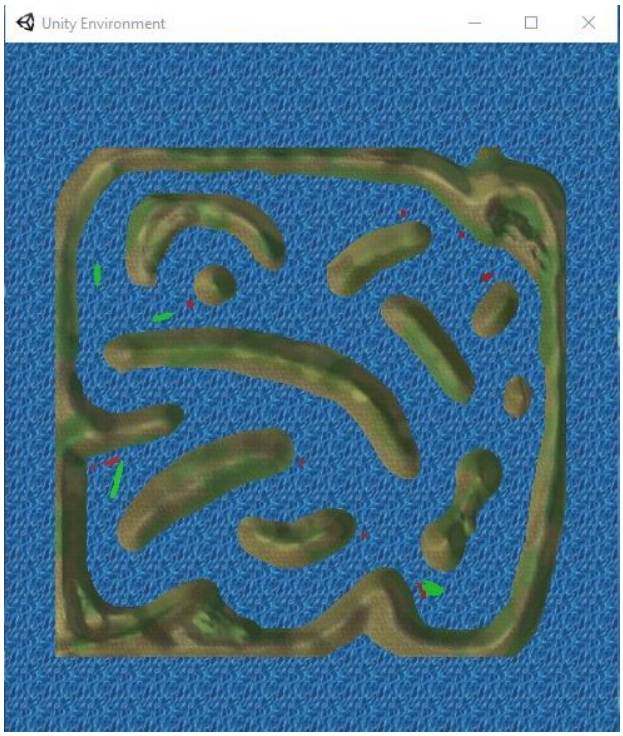

Рис. 7. Нейросеть в процессе обучения

Рис. 8. График средней награды

Проанализировав график можно заметить, что обучение достигло предела на середине пути. В этот момент было визуально оценено состояние обучения нейросети в приложении. Было замечено что в отличии от начала обучения, когда лодки абсолютно никак не реагировали на препятствие и постоянно сталкивались с ним, в середине обучения они уже довольно успешно избегали столкновений, но все еще далеко до идеала.

Исходя из этого можно сделать вывод что нужно очень точно подобрать параметры обучения. При удачном подборе параметров потребуется достаточно много времени для идеального обучения, когда лодки перестанут сталкиваться с поверхностью.

Было проведено еще несколько экспериментов с разными значениями параметров обучения нейросети, которые были занесены в таблицу (Табл.1), также было сокращено время обучения в 2 раза.

Таблица 1 – Подбор наилучших параметров обучения

№	Epsilon	Hidden units	Learning Rate	Number of Epochs	Number of Layers	Time Horizon	Max mean Reward
1	0,1	32	3,0e-5	7	2	248	16,1
2	0,2	32	3,0e-5	7	2	248	16,9
3	0,2	64	3,0e-5	7	2	248	19,5
4	0,2	64	3,0e-4	7	2	248	21,8
5	0,2	64	3,0e-4	10	2	248	15,2
6	0,2	64	3,0e-4	3	2	248	22
7	0,2	64	3,0e-4	3	4	248	20,2
8	0,2	64	3,0e-4	3	3	248	19,8
9	0,2	64	3,0e-4	3	2	512	18,2
10	0,2	64	3,0e-4	3	2	128	19,4

В экспериментах были протестированы отклонения от начальных значений. При повышении максимального значения среди средних наград при

обучении, параметр сохранялся. При понижении, бралось значение, отклоненное в другую сторону, при отрицательном результате оставлялось прежним. Из таблицы можно заметить, что наиболее удачным оказался шестой эксперимент.

Библиографический список

1. Радушкевич М.А. Анализ рисков инновационных проектов на основе нейронных сетей // Вестник факультета управления СПбГЭУ. 2017. № 1-2. С. 250-254.
2. Частикова В.А., Картамышев Д.А., Власов К.А. Нейросетевой метод защиты информации от ddos-атак // Современные проблемы науки и образования. 2015. № 1-1. С. 183.
3. Данилова О.Т., Трапезников Е.В. Разработка модели, анализирующей функцию безопасности в системе информационной защиты, на основе нейронной сети // Информационное противодействие угрозам терроризма. 2015. № 24. С. 24-29.
4. Серебряная Л.В., Потараев В.В. Методы классификации текстовой информации на основе искусственной нейронной и семантической сетей // Информатика. 2016. № 4. С. 95-103.
5. Зубричев Н.В., Ащепков Ф.А. Нейросеть как помощник автоматизации игровой анимации // В сборнике: Современные технологии: актуальные вопросы, достижения и инновации Сборник статей X Международной научно-практической конференции. Под общей редакцией Г.Ю. Гуляева. 2017. С. 44-46.
6. Зубричев Н.В., Ащепков Ф.А. Обзор областей применения нейросетей // В сборнике: Концепция динамического равновесия в новых технологиях сборник статей Международной научно-практической конференции. 2017. С. 33-36.
7. Зубричев Н.В. Новый вид нейронной сети для распознавания изображений – капсульная нейронная сеть // В сборнике: Современные технологии: актуальные вопросы, достижения и инновации Сборник статей XII Международной научно-практической конференции. В 2-х частях. Под общей редакцией Г.Ю. Гуляева. 2017. С. 106-108.
8. Murru N., Rossini R. A Bayesian approach for initialization of weights in backpropagation neural net with application to character recognition //Neurocomputing. – 2016. – Т. 193. – С. 92-105.
9. Dickson P. E. et al. An Experience-based Comparison of Unity and Unreal for a Stand-alone 3D Game Development Course //Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education. – ACM, 2017. – С. 70-75.
10. Unity ML-Agents // GitHub URL: https://github.com/Unity-Technologies/ml-agents (дата обращения: 27.08.2018).