УДК: 004.896
ББК: 32.816.6
ГРНТИ: 28.23.37

Волощук Сергей Алексеевич, *кандидат физико-математических наук, доцент кафедры прикладной математики Института информационных технологий, ФГБОУ ВО «МИРЭА – Российский технологический университет», Москва, Россия. E-mail: vsa77@mail.ru*

Volochshuk Sergey A., *PhD in Physics and Mathematics, Associate Professor, Department of Applied Mathematics, Institute of Information Technologies, MIREA – Russian Technological University, Moscow, Russia.*
E-mail: vsa77@mail.ru

Михалев Иван Олегович, *студент, ФГБОУ ВО «МИРЭА – Российский технологический университет», Москва, Россия. E-mail: bat.maks@ro.ru*

Mikhalev Ivan O., *Student, MIREA – Russian Technological University, Moscow, Russia. E-mail: bat.maks@ro.ru*

Ледовская Екатерина Валерьевна, *кандидат технических наук, доцент кафедры прикладной математики Института информационных технологий ФГБОУ ВО «МИРЭА – Российский технологический университет», Москва, Россия. E-mail: ledovskaya@mirea.ru*

Ledovskaya Ekaterina V., *PhD in Engineering Sciences, Associate Professor, Department of Applied Mathematics, Institute of Information Technologies, MIREA – Russian Technological University, Moscow, Russia.*
E-mail: ledovskaya@mirea.ru

Разработка нейросетевого бота на онлайн мультиплеерной платформе в рамках обеспечения информационной безопасности

Аннотация. *В рамках обеспечения информационной безопасности в онлайн-мультиплеерных видеоиграх рассмотрено ПО, запрещённое пользовательским соглашением. Разработан игровой бот, основанный на нейронной сети, как инструмент имитации получения нечестного преимущества в онлайн-гейминге. С целью сбора данных о поведении подобных ботов разработано приложение для создания обучающих наборов данных к нейронным сетям детекции объектов на изображении. Разработано приложение для сбора низкоуровневых характеристик игровых сессий в виде данных от клавиатуры и мыши. В результате анализа на основе машинного обучения предобработанного датасета с данными игровых сессий человека и с использованием бота определены метрики игровой активности для эффективного распознавания бота. Предложено использовать разработку как модель в процессе обучения с возможностью студентам оценивать качество работы бота и двойственной ему системы мониторинга использования ботов, а также разрабатывать мероприятия, обеспечивающие эффективное распознавание подобного запрещённого ПО и совершенствование бота.*

Ключевые слова: *бот, нейронная сеть, машинное обучение, пользовательское соглашение, запрещённое программное обеспечение, датасет, аннотация, учебная модель, информационная безопасность, онлайн-гейминг.*

Development of a Neural Network bot on an Online Multiplayer Platform as Part of Information Security

Abstract. *As part of providing information security in online multiplayer video games, the SW prohibited by the user agreement is considered. A game bot based on a neural network has been developed as a tool to simulate gaining an unfair advantage in online gaming. In order to collect data on the behavior of such bots, an application has been developed to create training datasets for neural networks detecting objects in an image. An application has been developed to collect low-level characteristics of gaming sessions in the form of keyboard and mouse data. As a result of machine learning-based analysis of a pre-designed dataset with data from gaming sessions with human participation and using a bot, metrics of gaming activity were determined for effective bot detection. It is proposed to use the development as a model in the learning process with the opportunity for students to evaluate the quality of the bot and its dual monitoring system for the use of bots, as well as to develop measures to ensure effective detection of such prohibited SW and improvement of the bot.*

Keywords: *bot, neural network, machine learning, user agreement, software prohibited, dataset, annotation, education model, information security, online gaming.*

Введение

В последние десятилетия наблюдается взрывной рост IT-индустрии [2]. В сегменте онлайн-игр с каждым годом становится всё больше геймингговых (iGaming) и гемблинговых (iGambling) платформ, предлагающих весь спектр игровых услуг. Игровые онлайн-платформы – это сложноструктурированные системы с большим числом внутренних и внешних производственных связей и огромным (фантастическим по меркам 2000-х гг.) количеством онлайн клиентов: > 110 млн активных аккаунтов у PokerStars (iGambling) или > 160 млн зарегистрированных аккаунтов World of Tanks (iGaming) – при показателях количества пользователей, участвующих в одновременной игре, регулярно и в пиковые загрузки: сотни тысяч и более 1 млн, соответственно. Взаимодействие с клиентами, обеспечение справедливой и сбалансированной игровой среды, регулировка вопросов нарушения правил оператора обеспечивается пользовательским соглашением (ПС) предоставления онлайн услуг [13]. Причём некоторые нарушения ПС несут в себе состав уголовного преступления, например, кардинг[1].

Мультиплеерные онлайн-видеоигры собирают миллионы пользователей со всего мира, создавая уникальные виртуальные миры, где игроки могут взаимодействовать

[1] *кардинг* – мошенническая финансовая операция с использованием платёжной карты (счёта, е-кошелька) или их реквизитов, не инициированная или не подтверждённая легальным владельцем.

друг с другом в рамках заданных правил и сценариев. Однако, как и в любой конкурентной среде, возникают попытки получения недобросовестного преимущества и извлечения прибыли в ущерб другим игрокам и онлайн-оператору [4; 13] посредством использования программного обеспечения (ПО), запрещённого ПС, которое часто выражено в виде ботов, решающих определённый спектр задач вместо пользователя и/или имитирующих поведение игрока. Обсуждаемое в статье ПО в ПС обычно акцентированно выделяется в подвид – «ПО на основе ИИ»: программы[2] автоматически (или автоматизированно) принимающие решение (ПР) и/или выполняющие какие-либо действия за игрока (боты) либо подсказывающие оптимальные действия в текущей игровой ситуации (подсказчики) [13]. В свою очередь, эффективное выявление нарушений ПС – важная задача оператора онлайн-видеоигр, часто решаемая в реальном масштабе времени (РМВ).

Работа направлена на разработку специфического ПО как инструмента имитации получения нечестного преимущества в онлайн-мультиплеерных видеоиграх в рамках исследования возможностей мониторинга использования ботов для обеспечения информационной безопасности (ИБ) в онлайн-гейминге.

Анализ последних исследований и публикаций

Обращаясь к ПС, отметим, что в разделе «Положение о несправедливом преимуществе» запрещена любая нечестная деятельность. Под данную формулировку попадают любые действия, которые могут поставить 3-х лиц (т. е. игроков, не участвующих в подобной деятельности) в невыгодное по отношению к соперникам положение [13]. Выделяют основные нарушения ПС: кардинг, бонусхантерство, мультиаккаунтинг, чипдампинг, майнинг, эксплуатация уязвимостей оператора, использование стороннего софта (включая ботов), манипуляция рейтингом, неэтичное поведение, извлечение данных игрового профиля, гостинг, баттонинг, сговор, встречающиеся в разных категориях услуг онлайн-гейминга и гемблинга.

В контексте многопользовательских видеоигр боты автоматизируют игровые действия участника, решая широкий спектр задач, от простого перемещения персонажа до выполнения сложных стратегических операций: внутриигровая торговля, сбор предметов, участие в PvP- (игрок против игрока, *player via player*) и PvE- (игрок против окружения, *player via environment*) сражениях. Боты работают на основе различных технических подходов, включая чтение и анализ данных из памяти игры, использование ИИ и машинного обучения (machine learning – ML) для ПР и др. Боты классифицируются по различным критериям: их основным функциям, используемым ими технологиям, целям, для которых они созданы и наиболее распространённая классификация: по типу выполняемых задач:

[2] определения таких программ немного разнятся, но везде они однозначно под запретом.

– фарм-боты: автоматизируют процесс сбора игровых ресурсов; повышают уровень персонажа или накапливают внутриигровые активы;

– PvP-боты: автоматизируют участие в сражениях между игроками, используя заранее заданные стратегии или адаптируясь к действиям противника с помощью алгоритмов ML;

– квест-боты: автоматизируют выполнение игровых заданий, следуя определённому алгоритму действий;

– торговые боты: автоматизируют процессы торговли внутриигровыми предметами, часто используя алгоритмы для максимизации прибыли.

Для диагностики использования игроками подобного ПО разрабатываются античит-системы – специализированные программные решения, предназначенные для обнаружения и предотвращения использования средств автоматизации и других видов манипуляций с игровым процессом. Для маскировки своего участия боты используют сложные алгоритмы имитации человеческого поведения и взаимодействия с игровой средой. Современные античит-системы, реализующие классификацию и фильтрацию неаутентичных агентов, разрабатываются на основе передовых технологий, включая глубокое обучение и компьютерное зрение (computer vision – CV) [3; 14]. Так в исследовании NVIDIA предложен основанный на CV новый подход к обнаружению нечестной игры с захватом конечного состояния буфера кадра и обнаружения нелегальных наложений при глубоком ML [12; 14].

В настоящее время не существует универсального метода отслеживания работы подобного запрещённого ПО, вследствие чего актуально эмпирическое изучение способов выявления ботов. В рамках учебного процесса студенты смогут ознакомиться с составом и структурой бота, задать дизайн экспериментов и протестировать его работу, а на продвинутом уровне предложить свои соображения по разработке мероприятий для эффективного распознавания ботов.

Автоматизация игровых процессов, запрещённая ПС, требует от бота способности «видеть» и «понимать» игровое окружение для ПР и выполнения действий, имитирующих поведение человека, реализуемое такими подходами, как:

1) **Анализ данных из памяти игры**: получение доступа к данным непосредственно из памяти компьютера, на котором запущена игра – информации о состоянии игрового мира, положении объектов, здоровье персонажей и т. д. Однако многие современные игры и античит-системы разрабатываются с мерами защиты, затрудняющими или предотвращающими прямой доступ к памяти.

2) **Анализ графики**: анализ изображений на экране игрока для определения игровых элементов и состояний, основанный на CV и ML, включая определение объектов, чтение интерфейсов и распознавание текста, аналогично интерпретации визуальной информации человеком.

3) **Использование игровых программных интерфейсов приложений (API)**: некоторые игры предоставляют официальные API, которые позволяют получать информацию об игровом мире и действиях в нём[3].

4) Имитация игровых действий пользователя: нажатия клавиш, движения мыши и др. Это требует от ботов способности интерпретировать игровое окружение и принимать решения о соответствующих действиях в РМВ.

Исследования показывают, что обнаружение объектов в различных условиях, например, на дорогах или в контексте социальных сетей, может предложить ценные уроки и методологии для распознавания игрового окружения [10]. В частности, методы, основанные на использовании свёрточных нейронных сетей и долгой краткосрочной памяти, могут достигать высокой точности в задачах обнаружения. Эти технологии позволяют эффективно анализировать и интерпретировать визуальную информацию, что критически важно для создания автоматизированных агентов детекции объектов на изображении в видеоиграх [3].

При анализе ботами игрового окружения и, по его результатам, последующем ПР, используемые ими алгоритмы варьируются от простых условных операторов до сложных систем специфических стратегий, алгоритмов оптимизации, ML и др. Выделяют следующие виды алгоритмов:

– **простая условная логика**: базовые алгоритмы принятия решений, построенные на логическом следовании «если-то», где каждое действие активируется в ответ на конкретные условия в игровом мире;

– **алгоритмы поиска пути**: для навигации по игровому миру и достижения целей (перемещение к определённой локации, избегание объектов и др.) используются алгоритмы поиска оптимального пути: A*, Dijkstra и др. [6];

– **алгоритмы оптимизации**: для выполнения ботом сложной серии действий могут применяться эвристические и другие методы «обучения» бота наилучшим стратегиям поведения;

– **поведенческие деревья**: для управления более сложным поведением ботов может использоваться гибкая и масштабируемая иерархическая структура, содержащая условные операторы;

– **ML**: хотя традиционно ML не является распространённым инструментом для разработки читов и ботов для игр, некоторые продвинутые системы могут использовать модели ML для адаптации к изменениям в игровой среде или для улучшения стратегий поведения посредством итеративного обучения.

[3] обычно это связано с разработкой дополнений, а не ботов. Этот метод может быть использован в исследовательских или образовательных целях для анализа игровых процессов.

Как при проектировании ботов, так и при разработке моделей обнаружения их использования задействуются алгоритмы, корректирующие свои внутренние параметры. Эти алгоритмы максимизируют свою эффективность на основе входных данных и обратной связи. Метрикой оценки эффективности является mAP (mean Average Precision) – расширенная версия метрики AP, измеряющей среднюю точность (precision) по всем возможным порогам полноты (recall), используемая для оценки производительности алгоритмов обнаружения объектов нескольких классов. Precision – это доля правильно идентифицированных положительных результатов (TP, true positive) из всех положительно классифицированных, включая ложноположительные (FP, false positive):

$$Precession = \frac{TP}{TP + FP} \quad (1)$$

Recall – это доля правильно идентифицированных положительных результатов из всех реальных положительных случаев, включая ложноотрицательные (FN, false negative):

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

Для каждого класса объектов AP рассчитывается отдельно путём построения кривой Precision (Recall) и вычисления площади под этой кривой. Затем mAP вычисляет среднее значение этих AP по всем классам. При оптимизации параметров алгоритма ML используют агрегированный критерий качества – метрику F_1: среднее гармоническое Precision и Recall [5].

В настоящее время ресурсоёмкость и высокая цена проектирования и реализации игровых ботов на основе ML обусловлены несколькими причинами:

• **сложность разработки**: каждая подобная система разрабатывается под конкретный игровой проект, что требует времени и навыков работы с ML; помимо этого, может требоваться команда разработчиков для создания качественных наборов обучающих данных и тестирования;

• **существование более простых методов решения**: часто применяют более простой алгоритм без ML для решения конкретной задачи, который, в свою очередь, может отличаться от своих аналогов, что затруднит его обнаружение;

• **сложность приобретения**: боты с ML стоят дорого и, как правило, распространяются среди специфических узких групп людей; если подобное ПО продаётся без ограничений к отбору покупателей, то оно вскоре попадает в списки запрещённого, что позволяет античит-программам быстро его выявлять.

Постановка проблемы

Важная проблема, которая стоит перед оператором онлайн-игр, – это эффективность распознавания нарушений ПС клиентом (или группой клиентов), где оказать существенную поддержку может выявление аномалий в метриках игровых сессий и сравнение реализации игровых стратегий. Сравниваются стратегии в рамках правил со стратегиями с использованием ботов, чтобы определять спектры игровых ситуаций, в которых цена действия, реализованного с нарушением ПС, значительно превышает цену игрового действия в рамках правил. В этом случае можно будет уверенно обнаруживать нарушение. Для выявления алгоритмов и метрик, на основе которых возможен мониторинг использования запрещённых систем автоматизации, необходимо обладать эмпирической базой. Для этого требуется спроектировать и реализовать нейросетевой игровой бот.

Цель работы

Рассмотреть различные виды ПО, создаваемого для автоматизации игровых действий без непрерывного участия человека в мультиплеерных онлайн-видеоиграх. Разработать бот на основе ML, как инструмент моделирования нарушения ПС со стороны клиента, с целью создания необходимой эмпирической базы для выяснения условий эффективного обнаружения использования ботов. В рамках имитации использования запрещённого ПО протестировать бот в многопользовательской видеоигре. Представить разработку в виде модели для процесса обучения студентов по направлению «Прикладная математика» (CV, классификация, дизайн экспериментов, ML и оценка регрессионных моделей) и «Программирование» (архитектура и функционал ПО).

Изложение основного материала

Процесс разработки игрового бота состоит из нескольких этапов. Первый этап – обучение нейронной сети, входящей в модуль восприятия игрового пространства. При ML возникает проблема баланса времени, потраченного на получение качественно обученной модели, и степени вовлечённости человека в процесс обучения. При обучении с учителем для ML моделей создаётся три набора данных: тренировочный, валидационный и тестовый. Каждый набор состоит из двух директорий: изображений для обучения и аннотаций к ним. Аннотация – это координаты ограничивающих прямоугольников (bounding box), задающих область обнаружения для объектов необходимых классов. На рис. 1 приведено аннотированное изображение (скриншот из игры ARK: Survival Ascended), где два ограничивающих прямоугольника выделяют объект класса "iron_ore" (внутриигровой ресурс «железная руда»). Такой метод ML требует больших затрат времени и человеческого ресурса.

При использовании полунадзорного (или неконтролируемого) ML степень человеческого участия снижается, но повышается само время обучения. Для решения данной проблемы реализовано сочетание нескольких подходов к ML на примере варианта

YOLOv8m последней версии моделей YOLOv8 детекции объектов на изображении семейства YOLO [1], что позволяет использовать преимущества каждого из них.

Реализованы этапы ML: предобучение модели, циклическое переобучение и анализ результатов. На первом этапе использован метод обучения с учителем для получения предобученной модели. Следующий этап – применение полученной модели к большому количеству неаннотированных данных. Предобученная сеть генерирует предполагаемые аннотации на основе уже приобретённых знаний. Полученные аннотации находятся в диапазоне уверенности модели в них confidence score (CS): $CS \in (0,1)$ [1]. CS для каждого предсказанного bounding box: $CS = P_{OS} \cdot P_{CCP}$ отражает вероятности – P_{OS} того, что в прямоугольнике содержится объект, и насколько точно этот прямоугольник его обрамляет, и P_{CCP} – вероятность принадлежности объекта к классу при условии, что в прямоугольнике действительно есть объект.

Рис. 1. Пример аннотированного изображения с двумя объектами класса "iron_ore", выделенными двумя ограничивающими прямоугольниками

Далее итеративно: к оригинальному набору обучающих данных добавляется доля сгенерированных аннотаций – $A_a \in (0,1)$, и модель переобучается в режиме самообучения. Используется сжатие исходных изображений с аннотациями до 640 пикселов. Полученные модели оцениваются метрикой mAP. При увеличении эффективности модели датасет возвращается в исходное состояние, и процесс генерации аннотаций и их объединения с оригинальными данными с последующем переобучением повторяется. В случае уменьшения оценки mAP к оригинальным данным, помимо сгенерированных моделью аннотаций, добавляются аннотации, сделанные человеком посредством отбора и корректировки данных, полученных от модели. С целью сбора данных

через этот процесс несколько раз проходит оригинальная предобученная модель и её потомки.

Для минимизации риска переобучения модели [5] использован алгоритм оптимизации AdamW [11] и лимит в 100 эпох обучения. Если в этих рамках модель, превосходившая предыдущие, не была получена, то обучение прекращалось, в противном случае обучение заканчивалось по истечении часа.

Важным фактором, определяющим эффективность ML в рамках предложенного подхода, является выявление оптимального соотношения количества аннотаций, составленных человеком, к количеству аннотаций, сгенерированных моделью. При таком соотношении обеспечивается постепенное улучшение результатов детекции сети на двух классах объектов через несколько итераций переобучения, с минимизацией человеческого вмешательства, сочетая в себе преимущества обучения с учителем и без учителя, а также активного обучения. Такой подход позволяет использовать небольшое количество данных на начальном этапе и сохранять качество обучающего набора по мере развития модели.

На первом этапе ML получена предобученная модель, которая использовала тренировочный набор из 100 заранее подготовленных человеком аннотаций. В валидационном наборе было 50 сделанных человеком аннотаций. После завершения обучения модели mAP50-95 = 0,523. Эта точка фиксировалась, и на следующем этапе предобученная модель и её набор обучающих данных использовались как базовые.

На втором этапе выполнен эксперимент из 17 опытов для сбора данных о переобучении модели с различными исходными характеристиками. Прирост mAP50-95 составил в среднем 5,7 % при σ=3,6. Дан пример опыта проведённой серии: базовая предобученная модель применена к видеоролику, не содержащему элементов, входящих в её оригинальный обучающий набор. Таким образом получен набор новых аннотаций без участия человека. Из нового набора при A_a=0,1 и с условием уверенности модели в них $CS \in (0.7,1)$ случайным образом выбрано 10 (= 100·0,1) аннотаций. Полученные аннотации добавлены в исходный обучающий набор. На момент переобучения тренировочная выборка состояла из 110 фотографий с аннотациями, а валидационная – из 50.

Условия опыта: алгоритм оптимизации adamW, обучение длится в течение часа с ограничением на ожидание прироста оценки эффективности mAP в 100 эпох. По итогам опыта mAP50-95 увеличилась на 0,027 (или ≈5,2 %). Экспериментальные данные заносятся в БД, содержащую поля: mAP95 – метрика базовой модели; ΔmAP95 – полученный прирост при оптимизации модели; CSmin – минимальный порог для отбора аннотаций к переобучению; CSmax – максимальный порог отбора; photoAuto+ – процент добавленных изображений, сгенерированных исходной моделью; photoHuman+ – процент фото, добавленных в исходный датасет, сделанных человеком; origineData – количество аннотаций в датасете исходной модели.

На третьем этапе все полученные данные анализируются методом множественной линейной регрессии. В качестве зависимой переменной выступает ΔmAP95, в качестве независимых – остальные. Для выявления мультиколлинеарности используется корреляция Пирсона [7; 8]. Приведён пример анализа первых проведённых опытов: по тепловой карте матрицы корреляции видно, что присутствует большая корреляция (0,9) между значениями mAP95 и ΔmAP95, что логично, учитывая их отношение как родителя к потомку.

Были добавлены термы взаимодействия для исследования изменения взаимосвязи между ΔmAP95 и другими независимыми переменными при различных значениях mAP95, так как влияние одной переменной на зависимую переменную зависит от уровня другой переменной. Взаимодействие означает, что влияние одной независимой переменной не является постоянным, а меняется в зависимости от значения другой переменной. Это позволило выявить более сложные отношения между переменными. Оценка модели коэффициентом детерминации (R^2=0.898) свидетельствует о том, что около 90 % вариации переменной ΔmAP95 может быть объяснено включёнными в модель независимыми переменными и их взаимодействиями. Это подчёркивает сильную объяснительную способность разработанной модели. При этом необходимо учитывать, что полученные результаты – итог первых опытов и предварительного анализа. Часть переменных, включая термы взаимодействия, не показали статистическую значимость. Следует отметить, что взаимодействие переменной photoauto+ показало статистическую значимость ($p < 0,05$). Такой результат предполагает, что влияние этой переменной на ΔmAP95 меняется в зависимости от уровня mAP95. F-статистика = 10,78 при p-значение = 0,000277 указывает на общую статистическую значимость модели. Однако присутствие мультиколлинеарности, подтверждаемое высоким условным номером и малым собственным значением, требует дальнейшего анализа и, возможно, коррекции модели. Приведённое программно-методическое решение, реализовавшее первый этап ML для модуля восприятия, характеризуется пониженной ресурсоёмкостью по времени и человеческому участию.

Второй этап разработки бота – создание модуля ориентации – содержит решение задач:

• разработка системы считывания данных с маркеров[4], получаемых от модуля восприятия;

• разработка системы ориентации в пространстве по расстоянию до полученному от считывания данных с маркеров;

• разработка системы упрощённой карты игры, хранящей в себе ключевые особенности рельефа;

• разработка системы, проектирующей траекторию движения в зависимости от текущего местоположения игрока и топологии игровой среды.

[4] маркеры – это часть игровой механики, позволяющая отметить точку в пространстве игры, располагающуюся на экране пользователя и показывающую расстояние до отмеченной точки.

Третий этап – создание модуля контроллера – содержит решение следующих задач:

• разработка системы, определяющей расстояние до железной руды по площади и координатам её bounding box, определённых модулем восприятия;

• разработка системы ввода: системы, имитирующей человеческое управление;

• разработка необходимых алгоритмов действий в зависимости от полученных модулем данных.

При эксплуатации разработки в качестве учебной модели, а также усовершенствования мониторинга использования ботов, предполагается тестирование в условиях игровых сессий человека и с участием бота, анализ поведения и выявление характерных особенностей полученного нейросетевого игрового бота, анализ и интерпретация полученных результатов и разработка мероприятий по противодействию реальной эксплуатации подобного ПО в iGaming.

Разработан вид фарм-бота, в задачи которого входит процесс сбора ресурсов и их складирование, повторяющиеся циклически до поступления команды от пользователя о прекращении процесса. Использован язык программирования Python с подключением фреймворка Flask и библиотек threading, queue, openCV, numpy, pygetwindow, mss, ultralytics, asyncio, logging.

Архитектура фарм-бота

Архитектура бота содержит три коммуницирующих между собой модуля.

Первый модуль: восприятие игрового пространства. В состав входит нейросеть, распознающая объект «железная руда» и маркеры. Модуль собирает следующую информацию:

• площадь прямоугольников, созданных нейросетью, для обозначения железной руды;

• координаты ограничивающих прямоугольников относительно монитора;

• скриншот области, попадающей в ограничивающий прямоугольник, содержащей маркер.

Второй модуль: ориентация бота в пространстве. Получает скриншоты маркеров от модуля восприятия, с которых считывается надпись под маркером, содержащая текущее расстояние от игрока до заранее отмеченных точек. Содержит упрощённую карту игрового пространства, где точки, полученные из игры, задают систему координат и позволяют боту знать текущую позицию, не обращаясь к файлам игры. Информация в модуле после обработки:

• текущее местоположение игрового персонажа;

• необходимое направление движения.

Третий модуль является контролирующим. В зависимости от данных, полученных им от модуля восприятия пространства и модуля ориентации, даёт команды к последовательностям действий и управляет остальными модулями. Например, отключая

их по необходимости для оптимизации использования ресурсов компьютера. Такая архитектура позволяет избежать взаимодействия с файлами игры, используя нейросеть в модуле восприятия, и маскирует работу бота от античит-систем мониторинга.

Порядок взаимодействия компонентов фарм-бота приведён на рис. 2 в виде диаграммы последовательности и состоит из четырёх приложений:

1. configurator.py: содержит класс Configurator, управляющий конфигурационными параметрами системы: FPS (частота кадров), масштаб изображения, порог площади объекта и порог расстояния. Этот класс предоставляет методы для установки и получения текущих значений этих параметров. Используется очередь config_queue для обмена изменениями конфигурации между потоками.

Основные методы: set_fps устанавливает значение FPS; set_scale устанавливает масштаб изображения; set_area_threshold устанавливает порог площади объекта; set_distance_threshold устанавливает порог расстояния; get_fps, get_scale, get_area_threshold, get_distance_threshold возвращают текущие значения параметров; get_config_updates возвращает список обновлений конфигурации из очереди.

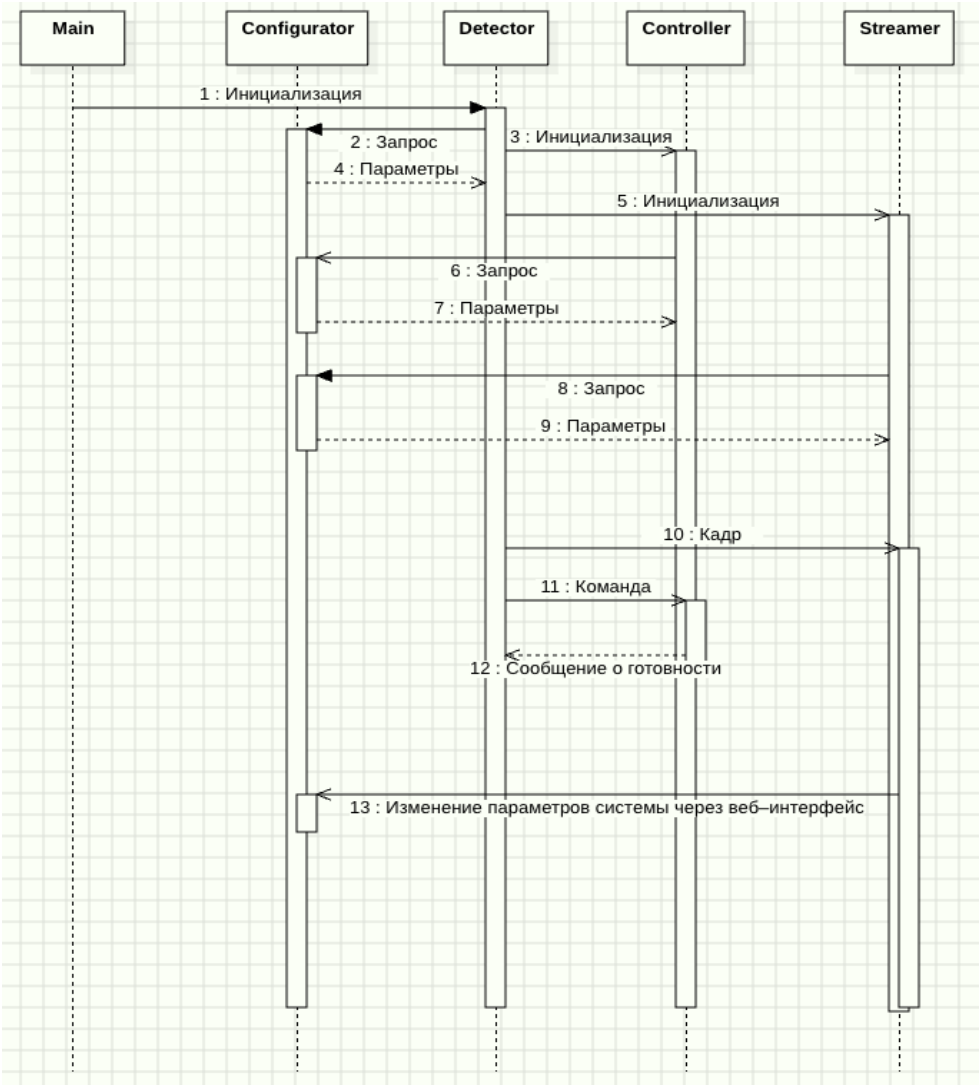

Рис. 2. Диаграмма последовательности фарм-бота

2. controller.py: содержит класс Controller, управляющий командами перемещения камеры и другими действиями. Получает команды через очередь controller_queue и генерирует соответствующие ответы. Реализована функция start_controller (controller_queue) для инициализации и запуска контроллера в отдельном потоке.

Основные методы: generate_commands генерирует команды управления на основе полученных данных; run запускает основной цикл обработки команд.

3. detector.py: главный файл проекта, отвечающий за захват и обработку видеопотока, а также взаимодействие с другими компонентами системы. Используются синхронные задачи для захвата и обработки видеопотока, а также для запуска контроллера и стримера в отдельных потоках. Реализована основная логика обработки видео с использованием YOLO и отслеживания объектов.

Основные функции и задачи: захват видеопотока с помощью mss и pygetwindow; обработка изображений с использованием OpenCV и numpy; обнаружение и отслеживание объектов с помощью модели YOLOv8 из библиотеки ultralytics; взаимодействие с Configurator для получения и применения настроек конфигурации; взаимодействие с Controller для управления движением камеры и других действий.

4. streamer.py: содержит класс Streamer, который отвечает за потоковую передачу видео через веб-интерфейс, используя Flask. Класс обрабатывает кадры из очереди frame_queue, применяя текущие настройки конфигуратора. Реализована функция start_streamer(frame_queue, configurator, config_queue) для инициализации и запуска стримера и веб-приложения Flask для предоставления видеопотока и интерфейса настройки параметров.

Основной метод: get_frame получает кадр из очереди и изменяет его размер в соответствии с текущими настройками конфигурации.

Спроектирован веб-интерфейс адаптированной настройки реализованного фармбота и разработки перспективной системы мониторинга с 4 видами настройки системы: Set FPS меняет количество обрабатываемых ботом кадров из прямого видеопотока игры; Set Scale меняет масштаб обрабатываемых изображений, что напрямую сказывается на производительности системы; Set Area Threshold изменяет порог области реагирования на ограничивающую рамку; Set Distance Threshold изменяет дистанцию реагирования.

Три основных этапа процесса работы бота:

1) Выбор ближайшего объекта: объект считается ближайшим, если его ограничивающая рамка по площади превышает ограничивающие рамки остальных объектов в кадре. В веб-интерфейсе это отображается как синяя линия, проведённая от центра кадра до центра выбранного объекта (рис. 3).

2) Наведение камеры: бот начинает следовать к объекту, по необходимости корректируя направление движения. В веб-интерфейсе – это зелёная траектория, проведённая от центра кадра (место нахождения харвестера (сборщика ресурсов) до объекта, к которому следует бот (рис. 4).

3) центрирование и добыча ресурса (как только бот достаточно приблизился к объекту): происходит за счёт поворота камеры и взгляда на объект под разными ракурсами.

Основные этапы 1÷3 циклически повторяются до поступления команды к завершению от пользователя либо при достижении лимита игрового инвентаря.

Рис. 3. Веб-интерфейс приложения Streamer. Выбор ближайшего объекта: линия проведена от центра кадра до центра выбранного объекта

Рис. 4. Веб-интерфейс приложения Streamer. Предполагаемая траектория движения до следующего объекта – кривая линия

Отметим, что программные решения для создания аннотаций изображений играют ключевую роль в подготовке данных для обучения нейросетей, особенно в области CV. Основная функция таких инструментов – обеспечение пользователей интерфейсом для маркировки объектов на изображениях, что позволяет создавать тренировочные наборы данных для алгоритмов детекции.

Предобработка и анализ данных игрового поведения

Собраны данные об игровых сессиях человека и с участием бота. Для удобства сбора данных написано приложение, считывающие все входящие сигналы от клавиатуры и мыши и вносящее их в формируемые датасеты с игровыми действиями – «keyboard_data» и «mouse_data», соответственно. После сбора данных полученные датасеты предобработаны для анализа игрового поведения. Поскольку одновременно записывались метрики, связанные и с поведением мыши, и с нажатием клавиш, данные разделены и проанализированы отдельно. Датасет «keyboard_data» предварительно приведён в строчный вид, так как, например, при зажатии «Shift» данные о нажатии какой-либо кнопки могли быть искажены. Применён метод "one-hot-encoding" [9] для преобразования категориальных данных в числовой формат и проведена минимакс-нормализация данных [5]. Для модели множественной логистической регрессии сделаны выводы по матрице корреляции Пирсона:

• большинство признаков не имеют значительной корреляции между собой, что указывает на независимость признаков и позволяет избежать проблем мультиколлинеарности. Умеренная корреляция между некоторыми признаками и целевой переменной (player_type: человек – 0, бот – 1) указывает на полезность этих признаков для классификации;

• независимость признаков помогает модели избежать переобучения и улучшить обобщающую способность. Исходя из матрицы корреляции Пирсона для mouse_data, сделаны аналогичные выводы, за исключением умеренной отрицательной корреляции между event_time и mouse_y, что может указывать на особенности игрового передвижения мыши при выполнении определённой задачи с течением времени.

Коэффициенты множественной логистической регрессии для соответствующих признаков, полученные при ML на датасетах keyboard_data и mouse_data, приведены в табл. 1 «Коэффициенты множественной логистической регрессии, полученные при ML на датасетах keyboard_data и mouse_data. Округление до 10^{-3}».

Таблица 1

keyboard_data						mouse_data	
event_time	-0,512	d	-0,318	alt	0,239	event_time	-0,904
duration	-1,44	k	0,462	ctrl	1,176	mouse_x	-1,237
space	0,178	s	-0,434	↓	0,229	mouse_y	-3,223
enter	0,18	w	-1,098	esc	-0,302		
a	-0,713	x	0,12	shift	-0,211		
d	-0,145	←	0,4	tab	0,239		

Анализ коэффициентов для признаков keyboard_data:

1. event_time и duration: отрицательные коэффициенты указывают на то, что с увеличением event_time и duration вероятность того, что действие принадлежит боту, уменьшается. Это значит, что боты, как правило, действуют быстрее и с меньшей продолжительностью действий по сравнению с человеком.

2. Клавиши: отрицательные коэффициенты для клавиш «w», «a», «s», и «d», «Esc», «Shift» в порядке значимости указывают на то, что эти клавиши чаще нажимаются человеком, чем ботом; положительные коэффициенты для «Ctrl», «k», «←», «Tab», «Alt», «↓» указывают на то, что эти действия чаще выполняются ботом.

Выводы по анализу датасета keyboard_data: базовая модель хорошо распознаёт ботов (высокая полнота для класса 1: 97 %), но плохо распознаёт людей (низкая полнота для класса 0: 8 %).

Анализ коэффициентов для признаков mouse_data:

1) event_time: -0,904. Увеличение времени события связано с уменьшением вероятности того, что во время игровой сессии используется бот.

2) mouse_x: -1,237. Увеличение значения координаты X движения мыши связано с уменьшением вероятности того, что игрок – бот. Это может означать, что боты склонны использовать более ограниченные диапазоны по оси X.

3) mouse_y: -3,223. Увеличение значения ординаты связано с уменьшением вероятности использования бота. Это может указывать на то, что боты имеют тенденцию двигаться в более ограниченных диапазонах по оси Y.

Все коэффициенты отрицательные, т. е. увеличение значений всех признаков (время события, координаты X и Y движения мыши) связано с уменьшением вероятности использования бота. Это говорит о склонности ботов к более ограниченным движениям мыши и более коротким интервалам между действиями.

Получены оценки качества классификации при количестве наблюдений для класса 0 (человек): 3263, для класса 1 (бот): 1871. Доля правильно предсказанных наблюдений из общего числа наблюдений: Accuracy ≈ 66,07 %. Для класса 0 (человек): Precision ≈ 68 %; для класса 1 (бот): Precision ≈ 58 %. Для класса 0: Recall ≈ 89 %; для класса 1: Recall ≈ 25 %.

На рис. 5 приведена ROC-кривая для классификации при ML на датасете mouse_data. Видно, что в начале кривая показывает высокую точность при низкой полноте, т. е. модель сначала предсказывает очень точно, но ловит только небольшое количество положительных примеров. Стабилизация: после начального всплеска кривая стабилизируется и остаётся примерно на одном уровне. Это показывает, что модель способна поддерживать баланс между Precision и Recall на относительно стабильном уровне при различных порогах вероятности. Падение: начиная с Recall ≈ 0,8, Precision начинает заметно снижаться. Это типичное поведение, когда модель начинает ошибаться чаще при попытке поймать больше положительных примеров.

Выводы по анализу датасета mouse_data: базовая модель хорошо распознаёт класс 0 с высокой полнотой Recall ≈ 89 %, но плохо распознаёт класс 1 с низкой полнотой Recall ≈ 25 %. Это указывает на склонность модели предсказывать класс 0; модель при использовании оптимального порога, метрики для класса 1 значительно улучшились (Recall ≈ 80 %), что указывает на лучшее распознавание ботов. Метрики для класса 0 ухудшились (Recall ≈ 55 %), что указывает на увеличение ложноположительных срабатываний.

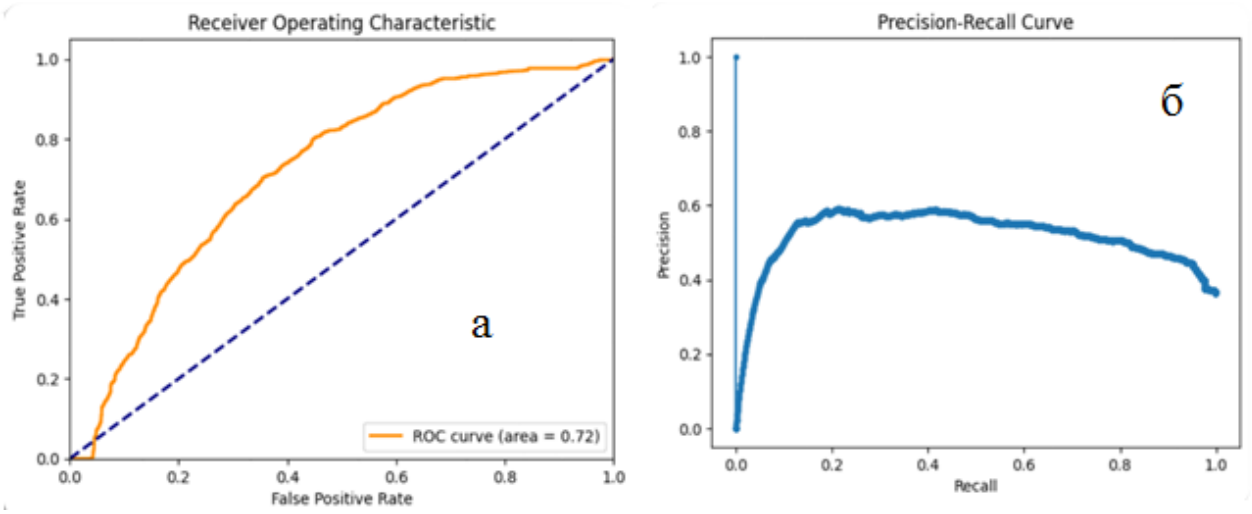

Рис. 5. Классификации ML на датасете mouse_data: а) ROC-кривая; б) зависимость точности предсказания от полноты – Precision (Recall)

Применение оптимизации порога помогает достичь лучшего баланса между распознаванием людей и ботов. Эти метрики можно использовать для дальнейшего улучшения модели, например, путём добавления новых признаков или использования более сложных моделей классификации.

Заключение

Рассмотрены различные виды ботов и их возможности предоставления нечестного преимущества в многопользовательских онлайн-видеоиграх. Для обеспечения ИБ спроектирован и реализован фарм-бот на основе ML, как инструмент для моделирования нарушения ПС со стороны клиента, для онлайн-мультиплеерной видеоигры ARK: Survival Ascended. Фарм-бот использован для создания необходимой эмпирической базы при подготовке эффективных способов обнаружения использования ботов. В составе разработки реализованы:

– модуль восприятия игровой топологии, основанный на ML;

– модуль ориентации, включающий подсистему считывания данных с маркеров, получаемых от модуля восприятия;

– модуль контроллер, включающий определение маршрута следования по данным, получаемых от модуля восприятия и модуля ориентации, а также подсистему

ввода, имитирующую человеческое управление на основе разработанных базовых алгоритмов игровых действий в зависимости от полученных данных.

Выполнены предобработка и анализ данных, собранных с клавиатуры и мыши, как результатов игрового поведения нейросетевого бота и человека, с целью выявления возможностей дифференцировать действия человека и бота. Для этого использованы методы множественной логистической регрессии, визуализации данных и оптимизации порогов классификации. Предложены метрики и критерии распознавания игрового поведения с использованием бота. Разработка представлена в виде учебной модели для студентов с целью демонстрации решения задач классификации, CV, дизайна экспериментов, проектирования и реализации нейронных сетей, архитектуры и функционала прикладного ПО.

Результаты работы применимы для решения задач идентификации ботов, в т. ч. использующих ML для обеспечения честной игры в iGaming, а также в других сферах, требующих высокого уровня ИБ многопользовательских систем, предотвращая возможные случаи мошенничества и кибератак.

Список литературы

1. Изучите Ultralytics YOLOv8 // Документация Ultralytics YOLO. URL: https://docs.ultralytics.com/ru/models/yolov8 (дата обращения: 04.11.2024).

2. Мировой IT-рынок // Сетевое издание TAdviser. 16.07.2025. URL: http://www.tadviser.ru/index.php/Статья:ИТ_(мировой_рынок) (дата обращения: 04.11.2024).

3. *Николенко С. И.*, *Кадурин А. А.*, *Архангельская Е. О.* Глубокое обучение. Погружение в мир нейронных сетей. СПб.: Питер, 2022. 480 с.

4. Правила игровых серверов по ARK Survival Ascended // Zmey Edition. Игровые сервера для вашего комфорта. URL: https://zmeyedition.ru/pages/ASA_Rules (дата обращения: 04.11.2024).

5. *Рашка С.*, *Лю Ю.*, *Мирджалили В.* Машинное обучение с PyTorch и Scikit-Learn / пер. с англ. Астана: Фолиант, 2024. 688 с.

6. A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games / S. R. Lawande, G. Jasmine, J. Anbarasi [et al.] // Applied Sciences. 2022. Vol. 12 (11). URL: https://www.mdpi.com/2076-3417/12/11/5499 (дата обращения: 04.11.2024).

7. *Etaga H. O.*, *Ndubisi R. C.*, *Oluebube N. L.* Effect of Multicollinearity on Variable Selection in Multiple Regression // Science Journal of Applied Mathematics and Statistics. 2021. Vol. 9, Iss. 6. P. 141–153.

8. *Menard S.* Applied logistic regression analysis. 2nd Edition. London: Sage Publications, 2001. 128 p.

9. One Hot Encoding in Machine Learning // GeeksforGeeks. URL: https://www.geeksforgeeks.org/ml-one-hot-encoding/?ysclid=m5nt6680eo624976558 (дата обращения: 04.11.2024).

10. Potholes and traffic signs detection by classifier with vision transformers / S. K. Satti, G. N. V. Rajareddy, K. Mishra [et al.] // Scientific Reports. 2024. Vol. 14. URL: https://www.nature.com/articles/s41598-024-52426-4 (дата обращения: 04.11.2024).

11. Reference API. Torch optim. AdamW // PyTorch. URL: https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html (дата обращения: 04.11.2024).

12. Robust Vision-Based Cheat Detection in Competitive Gaming / A. Jonnalagadda, I. Frosio, S. Schenider [et al.] // Research.nvidia.com. 24.03.2021. URL: https://research.nvidia.com/publication/2021-03_robust-vision-based-cheat-detection-competitive-gaming (дата обращения: 04.11.2024).

13. Terms and conditions // Ark-unity.com. URL: https://ark-unity.com/terms-of-service (дата обращения: 04.11.2024).

14. What Is Deep Learning? // Oracle.com. 20.08.2022. URL: https://www.oracle.com/europe/artificial-intelligence/machine-learning/what-is-deep-learning (дата обращения: 04.11.2024).

References

1. Izuchite Ultralytics YOLOv8. In: Dokumentatsiya Ultralytics YOLO. *Available at*: https://docs.ultralytics.com/ru/models/yolov8 (accessed: 04.11.2024).

2. Mirovoy IT-rynok. In: Setevoe izdanie TAdviser. 16.07.2025. *Available at*: http://www.tadviser.ru/index.php/Статья:ИТ_(мировой_рынок) (accessed: 04.11.2024).

3. Nikolenko S. I., Kadurin A. A., Arkhangelskaya E. O. *Glubokoe obuchenie. Pogruzhenie v mir neyronnykh setey*. St. Petersburg: Piter, 2022. 480 p.

4. Pravila igrovykh serverov po ARK Survival Ascended. In: Zmey Edition. Igrovye servera dlya vashego komforta. *Available at*: https://zmeyedition.ru/pages/ASA_Rules (accessed: 04.11.2024).

5. Raschka S., Liu Yu., Mirjalili V. *Mashinnoe obuchenie s PyTorch i Scikit-Learn*. Transl. from English. Astana: Foliant, 2024. 688 p. (In Russian)

6. Lawande S. R., Jasmine G., Anbarasi J. et al. A Systematic Review and Analysis of Intelligence-Based Pathfinding Algorithms in the Field of Video Games. *Applied Sciences*. 2022, Vol. 12 (11). *Available at*: https://www.mdpi.com/2076-3417/12/11/5499 (accessed: 04.11.2024).

7. Etaga H. O., Ndubisi R. C., Oluebube N. L. Effect of Multicollinearity on Variable Selection in Multiple Regression. *Science Journal of Applied Mathematics and Statistics*. 2021, Vol. 9, Iss. 6, pp. 141–153.

8. Menard S. *Applied logistic regression analysis*. 2nd Edition. London: Sage Publications, 2001. 128 p.

9. One Hot Encoding in Machine Learning. In: GeeksforGeeks. *Available at*: https://www.geeksforgeeks.org/ml-one-hot-encoding/?ysclid=m5nt6680eo624976558 (accessed: 04.11.2024).

10. Satti S. K., Rajareddy G. N. V., Mishra K. et al. Potholes and traffic signs detection by classifier with vision transformers. *Scientific Reports*. 2024, Vol. 14. *Available at*: https://www.nature.com/articles/s41598-024-52426-4 (accessed: 04.11.2024).

11. Reference API. Torch optim. AdamW. In: PyTorch. *Available at*: https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html (accessed: 04.11.2024).

12. Jonnalagadda A., Frosio I., Schenider S. et al. Robust Vision-Based Cheat Detection in Competitive Gaming. In: Research.nvidia.com. 24.03.2021. *Available at*: https://research.nvidia.com/publication/2021-03_robust-vision-based-cheat-detection-competitive-gaming (accessed: 04.11.2024).

13. Terms and conditions. In: Ark-unity.com. *Available at*: https://ark-unity.com/terms-of-service (accessed: 04.11.2024).

14. What Is Deep Learning? In: Oracle.com. 20.08.2022. *Available at*: https://www.oracle.com/europe/artificial-intelligence/machine-learning/what-is-deep-learning (accessed: 04.11.2024).