Нагорных М.Э., Антонов А.А., Чернышёв С.А. Программное обеспечение для дистанционного управления приложениями посредством жестов кистей рук // Международный журнал информационных технологий и энергоэффективности. – 2021. – Т. 6 № 1(19) с. 34–41

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.058:004.93

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ДИСТАНЦИОННОГО УПРАВЛЕНИЯ ПРИЛОЖЕНИЯМИ ПОСРЕДСТВОМ ЖЕСТОВ КИСТЕЙ РУК

[1] Нагорных М. Э., [2] Антонов А.А., [3,4] Чернышёв С. А.

[1,2] Санкт-Петербургский государственный университет аэрокосмического приборостроения, Россия (190000, г. Санкт-Петербург, ул. Большая Морская, 67, лит. А), e-mail: [1] nagornykh_max@mail.ru, [2] grand.antonov@yandex.ru

[3] Санкт-Петербургский государственный университет промышленных технологий и дизайна, Россия (191186, г. Санкт-Петербург, ул. Большая Морская, 18)

[4] Санкт-Петербургский государственный экономический университет, Россия (191023, г. Санкт-Петербург, ул. Садовая, 21), e-mail: chernyshev.s.a@bk.ru

В статье рассмотрен процесс разработки программного обеспечения для дистанционного управления компьютером и приложениями с помощью жестов рук. Для решения этой задачи используются нейронные сети с глубоким обучением. В ходе работы произведен анализ существующих обученных нейронных сетей и выбор, с целью дальнейшего дообучения и внедрения в разрабатываемое приложение. Разработанное программное обеспечение проходило тестирование при взаимодействии с различными видами прикладных программ. В качестве одного из примеров, который приведен в завершении статьи, это его использование для управления персонажем в компьютерной игре.

Ключевые слова: нейронные сети, глубокое обучение, аннотирование, Tensorflow, Python

SOFTWARE FOR REMOTE CONTROL OF APPLICATIONS THROUGH HAND GESTURES

[1]Nagornykh M.E., [2] Antonov A.A, [3,4] Chernyshev S.A.

[1,2] Saint Petersburg State University of Aerospace Instrumentation, Russia (190000, Saint Petersburg, Bolshaya Morskaya st., 67, letter A), e-mail: [1] nagornykh_max@mail.ru, [2] grand.antonov@mail.ru

[3] Saint Petersburg State University of Industrial Technology and Design, Russia (191186, Saint-Petersburg, Bolshaya Morskaya st., 18)

[4] Saint Petersburg State University of Economics, Russia (191023, Saint Petersburg, Sadovaya st., 21), e-mail: chernyshev.s.a@bk.ru

Нагорных М.Э., Антонов А.А., Чернышёв С.А. Программное обеспечение для дистанционного управления приложениями посредством жестов кистей рук // Международный журнал информационных технологий и энергоэффективности. – 2021. –
Т. 6 № 1(19) с. 34–41

The article discusses the process of developing software for remote control of the computer and applications using hand gestures. Neural networks with deep learning are used to solve this problem. In the course of the work, the existing trained neural networks are analyzed and the choice is made, with the aim of further training and implementation in the application being developed. The software developed was tested when interacting with different types of applications. One example that is given at the end of the article is its use to control a character in a computer game.

Keywords: neural networks, deep learning, annotation, Tensorflow, Python.

Введение

В современном мире появляется все больше новых технологий, а с ними и новые возможности. Нейронные сети, например, уже используется практически во всех сферах жизни человека [1]. Одна из самых популярных задач для их применения - поиск и классификация объектов. Она присутствует в проектах распознавания лиц людей, рук и других частей тела. Однако, множество доступных проектов используют комбинацию из нейронных сетей и различных специальных технических средств: системы Kinect, стереопары и т.д. Но такие устройства, в большинстве случаев, являются дорогими или недоступными для обычных пользователей. Избежав использования дополнительных устройств и правильно обучив нейросеть, можно получить систему, применимую в большем спектре задач [2]. Ее можно использовать для определения жестов, рисования руками в информационном пространстве, распознавания азбуки глухонемых, и многом другом.

Учитывая современные тенденции, такую систему можно применить для управления пользовательским интерфейсом программ при помощи жестов кистей рук, избегая использования привычных устройств ввода.

Актуальность данной темы очень велика, так как возникают ситуации, в которых человек не может воспользоваться даже мышкой. Поэтому, цель работы заключается в создании универсального программного обеспечения, которое поможет сменить привычную всем парадигму, использования стандартных устройств ввода при управлении интерфейсом приложений, без применения специальных технических средств.

1. Выбор нейросети

Основным языком разработки проекта был выбран Python, из-за его ориентированности на нейросети и удобную работу с подключаемыми библиотеками [3]. Он позволяет быстро прототипировать идеи, что ускоряет процесс отработки методологии.

Для реализации проекта использовался дистрибутив Anaconda, который содержит в себе множество библиотек для работы с нейросетями и машинным обучением [4, 5]. В качестве средства для дообучения нейросети, использовался tensorflow[6]. Это библиотека с открытым исходным кодом предоставляющая разработчикам возможность работы с созданными ранее архитектурами нейросетей.

Для решения задач поиска и классификации жестов кистей рук на изображении отлично подходит модель обучения faster_rsnn, так как она дает высокие результаты при небольшом количестве данных и способна распознавать объекты в режиме реального времени [7].

В открытом доступе существует множество обученных нейросетей, поэтому, в основу проекта легла уже готовая, распознающая 5 жестов - 1-5 показанных пальцев. Не смотря на малое количество изначально распознаваемых объектов, она имеет хорошее качество поиска

и классификации как кистей рук, так и формируемых ими жестов. Кроме того, данная нейронная сеть является сверточной с архитектурой faster_rsnn, что полностью подходит под заданные требования.

Количество жестов, имеющихся в наличии недостаточно, чтобы обхватить основной спектр задач по управлению компьютером и приложениями. В связи с этим было принято решение - дообучить нейронную сеть до нужного числа распознаваемых и классифицируемых жестов.

2. Подготовка выборки

Первым этапом дообучения нейросети является формирование набора обучающих данных. Было добавлено 10 жестов рук, среди них: «большой палец вверх», «кулак», жест «окей», жест «коза», а оставшиеся являются их комбинациями. На каждый из них приходилось по 30 фотографий, с разным положением рук и фона. Пример подобного изображения продемонстрирован на рисунке 1.

Рисунок 1 - Пример изображения из выборки

Всего было добавлено порядка 430 изображений, 30% - тестовые, содержащие в себе как изображения жестов, так и посторонних объектов, которые составляли 5% от тестовой выборки.

Следующий этап - аннотирование изображений. Для этого использовалась программа LabelImg, которая позволяет выделить область на изображении, присвоить ей значение, а

после этого сохранить данные о всех действиях в файле формата xml. Процесс выполнения разметки изображений показан на рисунке 2.

Рисунок 2 - Пример разметки изображения

3. Дообучение нейросети

После аннотирования изображений началось дообучение нейросети. Этот процесс осуществлялся с помощью библиотеки tensorflow и происходил порядка 200000 итераций до точности не менее 0.05, что заняло примерно 39 часов. Процедура дообучения продемонстрирована на рисунке 3.

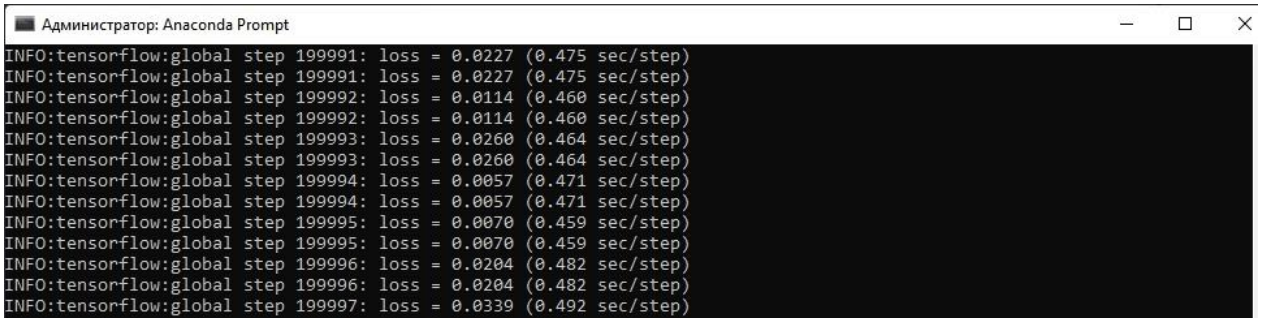

Рисунок 3 - Процесс дообучения нейросети

4. Тестирование нейросети

Тестирование нейросети происходило в приложении, которое в режиме реального времени на графический пользовательский интерфейс выводит видеопоток, ищет кисть в кадре и отрисовывает вокруг нее прямоугольник. Далее, определяется показанный жест, выводится его название, а также точность с которой он был распознан. Пример тестирования приложения продемонстрирован на рисунке 4.

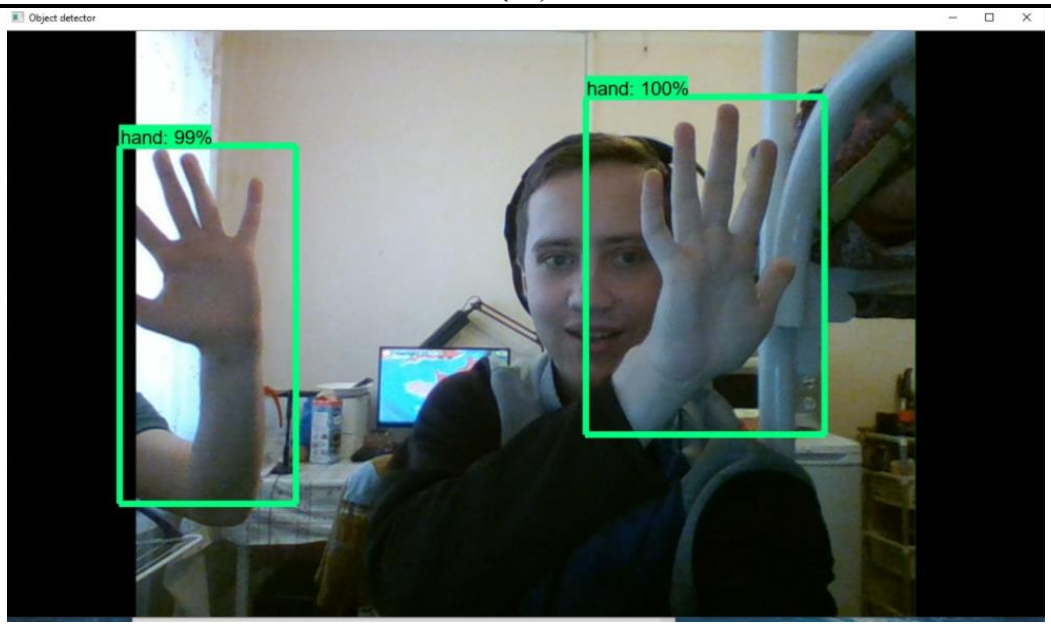

Рисунок 4 - Тестирование нейросети

5. Настройка взаимодействия с компьютером

Последний этап - создание программного обеспечения, основной функцией которого является синхронизация показанных жестов с действиями компьютера. Для решения этой задачи отлично подошла библиотека Python - PyAutoGUI. Она позволяет эмулировать работу клавиатуры, мыши и имеет следующие функции:

- Перемещение и нажатие мыши;
- Ввод текста;
- Отправка нажатий клавиш приложениям (например, для заполнения форм);
- Создание снимка экрана и поиск изображения (например, кнопки или флажка) на экране;
- Поиск окна приложения и его перемещение, изменение размера, разворот, свертка или его закрытие
- Отображение окон сообщений для взаимодействия с пользователем во время выполнения сценария автоматизации графического интерфейса.

В проекте, рассматриваемом в этой статье, данная библиотека используется только для эмуляции работы мыши и клавиатуры. Пример кода скрипта продемонстрирован ниже:

```
if (class_name[0][0] == self.MouseMove):
    xW = (coordinate[0][3] - (coordinate[0][3] - coordinate[0][2]) / 2)
    yH = (coordinate[0][1] - (coordinate[0][1] - coordinate[0][0]) / 2)
    if (math.fabs(xW - self.xW_lastPosition) > 200 or math.fabs(yH - self.yH_lastPosition) > 200):
        self.xW_lastPosition = xW
        self.yH_lastPosition = yH
    deltaX = xW - self.xW_lastPosition
    deltaY = yH - self.yH_lastPosition
    self.xW_lastPosition = xW
    self.yH_lastPosition = yH
    pyautogui.moveTo(x_mousePosition + (self.Sensitivity * (deltaX)),
                     y_mousePosition + (self.Sensitivity * (deltaY)))
```

В приведенном коде рассчитывается следующее положение курсора мыши на экране, в зависимости от смещения руки в кадре, при условии, что жест не изменялся.

Для взаимодействия с пользовательским интерфейсом в программе необходимо знать позиции ключевых окон приложения на экране или комбинаций клавиш для их вызова. Поиск этих окон осуществлялся с помощью метода сопоставления с шаблоном, предоставляемый библиотекой компьютерного зрения OpenCV.

6. Тестирование в приложении

Для демонстрации работы приложения отлично подошла компьютерная игра Mario. Для управление игровым процессом не требуется много клавиш, поэтому каждому игровому действию соответствует свой жест: движение вправо – жест «два пальца», движение влево – жест «3 пальца», движение вверх – жест «коза», движение вниз – жест «окей», прыжок персонажа – жест «кулак», выстрел – жест «большой палец вверх».

Подобное управление хоть и непривычно, но не мешает игровому процессу. Примеры некоторых действий в игре можно наблюдать на рисунках 5 и 6, на первом показана навигация в меню, а на втором прыжок персонажа.

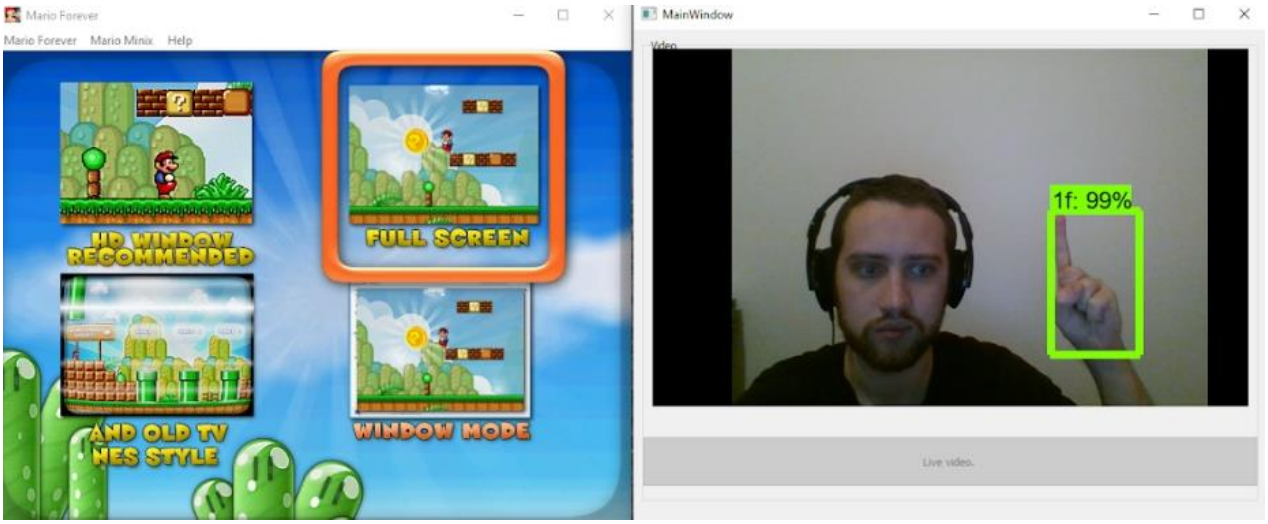

Рисунок 5 - Выполнение навигации по меню

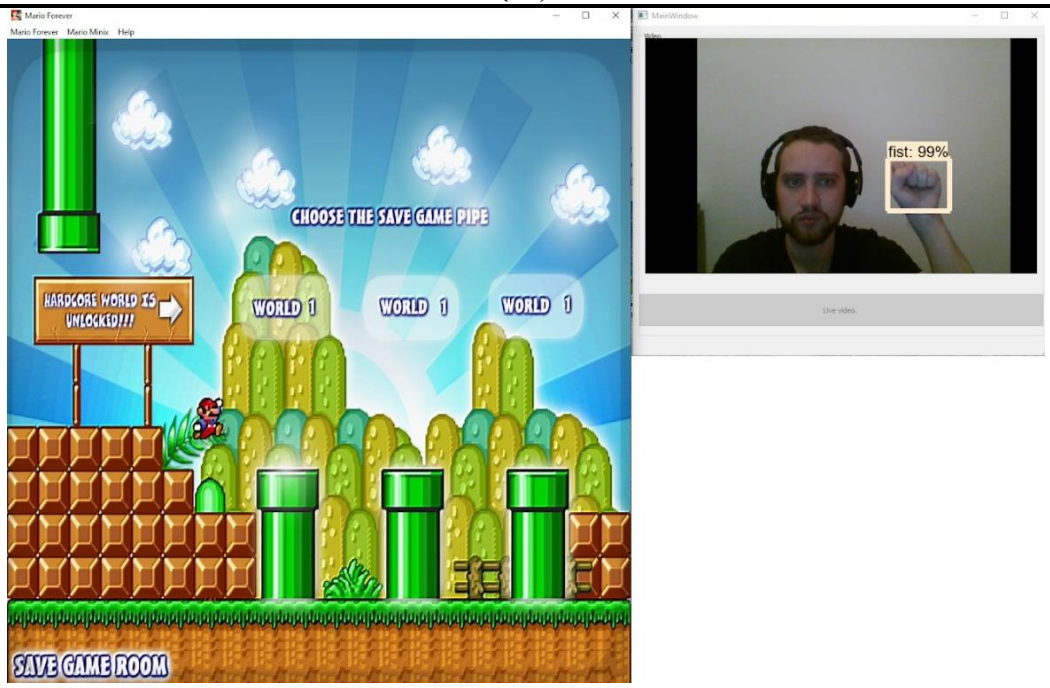

Рисунок 6 - Выполнение прыжка в игре

Выводы

Представленное в статье программное обеспечение успешно справляется с поставленной задачей. Особенно, это касается ситуаций, не требующих быстрого и обильного нажатия клавиш. Существование такого приложения уже доказывает, что управление пользовательским интерфейсом компьютера может происходить и без использования мыши и клавиатуры, а с помощью обычной веб-камеры, которая имеется практически у всех.

Разработанное приложение может являться основой для более сложных систем.

В последнее время большую популярность набирает дополненная реальность, однако, несмотря на большое количество специализированных контроллеров, не один из них не может воспроизвести точную модель руки. Поэтому внедрение разработанной технологии может позволить посмотреть на решение этой проблемы с другой стороны.

Список литературы

1. S.Haykin. Neural Networks and Learning Machines. 3rd Edition. Pearson, 2018.
2. Редько В.Г. Эволюция, нейронные сети, интеллект: Модели и концепции эволюционной кибернетики / В.Г. Редько. - М.: Ленанд, 2019. - 224 с.
3. Любанович Билл. Простой Python. Современный стиль программирования / Билл Любанович. - М.: Питер, 2016. - 480 с.
4. Anaconda python дистрибутив, официальный веб-сайт. — Электронный ресурс. — https://www.anaconda.com. — дата обращения: 20.12.2020.
5. Гудфеллоу Я., Бенджио И., Курвилль А. − Глубокое обучение. ДМК, Москва, 2018 г.
6. Pramod Singh, Avinash Manure. Learn TensorFlow 2.0 -NY: Apress, 2020. -177p.
7. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Neural Information Processing Systems (NIPS), 2015.

References:

1. S.Haykin. Neural Networks and Learning Machines. 3rd Edition. Pearson, 2018.
2. Redko V.G. Evolution, neural networks, intelligence: Models and concepts of evolutionary cybernetics / V.G. Redko. - M.: Lenand, 2019. - 224 p.
3. Lubanovich Bill. Simple Python. Modern programming style / Bill Lubanovich. - M.: Peter, 2016. - 480 p.
4. Anaconda python distribution, official website. Available at: https://www.anaconda.com. (accessed 20 December 2020).
5. Goodfellow J., Benjio I., Curville A. - Deep Learning. DMK, Moscow, 2018.
6. Pramod Singh, Avinash Manure. Learn TensorFlow 2.0 -NY: Apress, 2020. -177p.
7. S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” in Neural Information Processing Systems (NIPS), 2015.