

Использование нейронных сетей для поиска разнородных объектов на
Python

Ветрова Ольга,

Храмов Дмитрий Эдуардович, ассистент кафедры статистики и
кибернетики, ФГБОУ ВО РГАУ-МСХА им. К.А. Тимирязева

Москва 2023

Оглавление

ВВЕДЕНИЕ.....	3
ИССЛЕДОВАНИЕ В ПРЕДМЕТНОЙ ОБЛАСТИ	4
ПРЕДЛАГАЕМОЕ РЕШЕНИЕ	6
ЗАКЛЮЧЕНИЕ	10
СПИСОК ЛИТЕРАТУРЫ.....	11
ПРИЛОЖЕНИЕ А	12

ВВЕДЕНИЕ

В настоящее время существует задача сортировки разнородных объектов техническими средствами, использующими различные сенсоры и программы обработки получаемой информации. Одной из областей применения данной технологии является сортировка мусора.

Данная тема была выбрана ввиду возрастания важности проблем экологии и ограниченности ресурсов. Часть этих проблем решается с помощью технологий переработки отходов. Особенно следует выделить трудности с сортировкой отходов.

Проблема экологии и переработки мусора остро стоит в обществе. Потребность в переработке мусора и отходов возрастает с каждым годом.

Учитывая факт, что человек уже производит материалы, которые в природе могут разлагаться до 150 лет, вопрос их вторичного использования после переработки становится все более и более актуальным.

ИССЛЕДОВАНИЕ В ПРЕДМЕТНОЙ ОБЛАСТИ

Искусственные нейронные сети (ИНС) представляют собой математическую модель, а также её программное или аппаратное воплощение. Принцип построения основан на организации и функционирования биологических нейронных сетей, которые обычно ассоциируются с процессами человеческого мозга [3]. ИНС представляется в виде системы связанных и взаимодействующих между собой процессоров (нейронов).

Развитие аппарата искусственных нейронных сетей позволяет применять их повсеместно, не ограничиваясь одной какой-либо областью. Однако, на данный момент, ИНС не способны охватить все области и преимущественно решают лишь один тип задачи, на который они обучены. Помимо этого, остается еще множество сфер деятельности человека, где сети не раскрыли весь свой потенциал и не имеют широкого распространения. Одной из такой области является сортировка мусора.

Согласно статистическим данным, один человек в течение суток производит около 3-5 литров мусора. В месяц этот объем составляет до 120 литров, а за год 15000 литров. В масштабах планеты эти показатели являются огромными.

Очевидно, что переработка необходима, но главная задача и она же главная сложность переработки отходов является сортировка мусора. В данном случае идет речь о предварительной сортировке на соответствующие категории [1].

Можно сказать, что после сбора отходов предварительная сортировка представляет собой первичный этап переработки.

В настоящее время мусороперерабатывающие заводы используют роботизированные сортировочные установки и технологии распознавания объектов для определения типов отходов и их сортировки.

Благодаря технологии глубинного обучения [4] роботы-сортировщики используют специальную систему машинного зрения для распознавания

материалов, искусственный интеллект — для определения типа каждого материала и манипулятор — для выбора определенного типа отходов. По мнению Томаса Брукса, директора по технологиям компании BHS, которая производит роботизированных сортировщиков отходов Max-AI, данная технология играет важную роль в процессе повышения рентабельности перерабатывающих систем.

В настоящее время роботизированные сортировщики AMP Robotics используются например, на заводе AlpineRecycling [5].

Помимо этого, существует еще одна компания, которая предлагает свое роботизированное решение – ZenRobotics [9].

Эти усовершенствованные сортировщики используют то же машинное зрение, которое используют системы автоматизации производственных процессов, например, для изготовления микросхем. Однако отделить бумагу от пластика может быть не так просто, как кажется.

Такая система сортировки эффективна при работе с хорошо упорядоченными отходами, однако отходы в центрах переработки представляют собой кучи перемешанного мусора даже после механической сортировки.

Для того чтобы система смогла отделить картонную коробку от смятой консервной банки, которые появляются в разных местах конвейера и в разных положениях, необходимо создать большой репрезентативный набор данных.

Несмотря на то, что искусственный интеллект только недавно начал использоваться для сортировки отходов, приверженцы этой технологии уверены, что он может найти множество других применений помимо контроля качества. Мусороперерабатывающие заводы представляют собой лабиринты конвейеров и систем сортировки, работающих по схеме, цель которой максимально эффективно распределять поступающие отходы в зависимости от их типов.

Использование данной технологии в будущем может значительно повысить эффективность и рентабельность мусороперерабатывающих заводов.

ПРЕДЛАГАЕМОЕ РЕШЕНИЕ

Поскольку создание полноценной нейронной сети, которая будет способна сравниться или даже превзойти существующие скоростные параметры готовых нейронных сетей – задача многолетней разработки в области математики, которая, в данный момент, не оправдывает вложенные усилия. Исходя из вышесказанного, целесообразнее пользоваться уже готовыми библиотеками.

В качестве языка программирования был выбран Python, поскольку он обладает рядом преимуществ и имеет обширную коллекцию библиотек для нашей задачи.

В настоящее время существует несколько крупных библиотек и программного обеспечения, которые можно использовать для дальнейшего построения модели:

- ImageAI – это библиотека Python, созданная для того, чтобы дать возможность разработчикам, исследователям и студентам создавать приложения и системы с автономными возможностями Deep Learning и Computer Vision с использованием простого и небольшого количества кода. Предоставляет очень мощные, но простые в использовании классы для обучения самых современных алгоритмов глубокого обучения [7].
- OpenCV (Open Source Computer Vision Library) – библиотека программного обеспечения для компьютерного зрения и машинного обучения с открытым исходным кодом. OpenCV был создан для обеспечения общей инфраструктуры для приложений компьютерного зрения и для ускорения использования машинного восприятия в коммерческих продуктах [8].
- Симулятор робота V-REP с интегрированной средой разработки основан на распределенной архитектуре управления: каждый объект / модель может индивидуально управляться с помощью встроенного скрипта, плагина, узла ROS или BlueZero, удаленного клиента API или пользовательского интерфейса, решения. Это делает V-REP очень универсальным и идеально подходящим для применения с несколькими роботами. Контроллеры могут быть написаны на C

/ C ++, Python, Java, Lua, Matlab или Octave [6]. Также данное программное обеспечение имеет большую библиотеку с разнообразными роботизированными моделями, манипуляторами, чипами, сенсорами. Дополнительно имеется возможность импортирования объектов из других источников. Пример интерфейса изображен на рисунке 1.

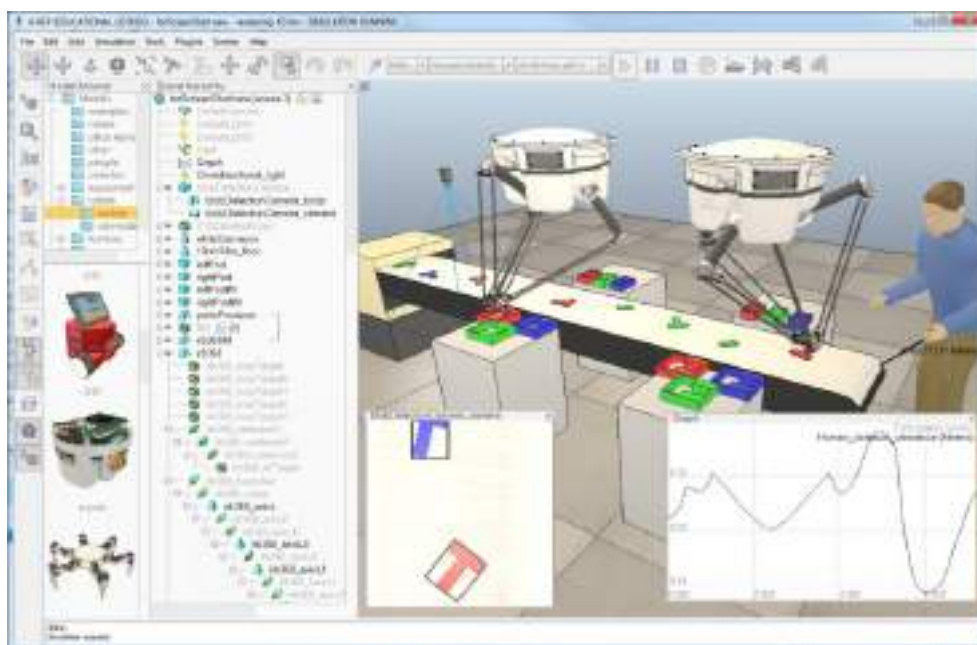


Рисунок 1 – Интерфейс V-геп

Рассмотрев существующие решения и прототипы, а также возможности программных средств, можно сделать несколько выводов, чтобы выделить общее и разработать собственную концепцию.

Во-первых, необходимо использовать камеры для распознавания объектов. Есть два вида установки: до манипуляторов и непосредственно возле манипулятора или на нем. Эффективность установки можно установить опытным путем.

Во-вторых, в рассмотренных примерах используется обычно от одной до трех «рук». Помимо количества различаются их тип и способ установки – картезианский и параллельный.

Картезианская конструкция – самый простой вариант стационарной системы. Малое количество шарниров и сочленений обеспечивает жесткость конструкции и высокие показатели точности.

Недостатками данной конструкции являются: не самая удобная форма обслуживаемого пространства; низкое соотношение обслуживаемого и занимаемого роботом пространств; ограниченное ориентирование инструмента и невозможность работы в местах с затрудненным доступом.

Плюс параллельных манипуляторов в том, что исполнительное звено имеет жёсткость, которая во много раз превышает жёсткость последовательного манипулятора, за счёт этого точность манипулятора очень высока. Также конструкция очень легка за счёт того, что звенья работают на растяжение и сжатие, поэтому звенья могут быть лёгкими.

Главный недостаток в том, что рабочая зона из-за особенности конструкции может становиться меньше. Также в параллельном манипуляторе двигатели должны работать без ошибок, а иначе может произойти заклинивание или даже поломка манипулятора.

Также стоит обратить внимание на то, что такие технологии значительно увеличивают общую эффективность предварительной сортировки в сравнении с ручным способом. Считается, что в самом ближайшем будущем, уменьшенные копии подобных машин будут использоваться повсеместно, в том числе и в быту, что облегчит весь процесс переработки и позволит снизить общие издержки [2].

Для сравнения второго пункта необходимо будет провести ряд экспериментов и опробовать на деле каждый вариант, чтобы выбрать наиболее оптимальный, который будет удовлетворять в скорости работы и обучения, а также точности в распознавании объекта.

Помимо этого, главным аспектом является нейронная сеть, точнее ее обучение. Для того, чтобы ИНС научилась определять подходящий объект, необходимо подготовить обширную базу исходных данных, на которых тренируется сеть. Минимальный объем изображений на один объект, в лучшем случае, составляет 1000 штук, а учитывая сложный характер работы их потребуется больше.

Для обучения ИНС был написан скрипт (Приложение, Скрипт Python для обучения ИНС).

Главной функцией, от которой зависит тип модели, является `setModelTypeAs`, их существует 4 типа:

- `setModelTypeAsSqueezeNet()` – функция устанавливает тип модели обучающего экземпляра в модель SqueezeNet;
- `setModelTypeAsResNet()` – функция устанавливает тип модели обучающего экземпляра, который вы создали, в модель ResNet;
- `setModelTypeAsInceptionV3()` – функция устанавливает тип модели обучающего экземпляра, который вы создали, в модель InceptionV3;
- `setModelTypeAsDenseNet()` – функция устанавливает тип модели обучающего экземпляра, который вы создали, в модель DenseNet.

Особенности работы и точности данных моделей будут проанализированы после того, как сформируется база данных объектов. На данном этапе, сложно дать оценку, поскольку на текущем этапе исследования у всех моделей имеется схожая погрешность в работе.

Пример разработанного скрипта, для определения объектов представлен в Приложении (Скрипт Python для распознавания видео с помощью OpenCV).

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы был проведен анализ предметной области, найдены программные средства для построения виртуального прототипа. Для тестирования работоспособности скрипта ИНС была обучена на небольшом объеме данных и способна определять объект лишь в некоторых позициях.

В настоящее время проводится сбор достаточного по объему банка данных для обучения нейронной сети, чтобы повысить ее эффективность и количество объектов, которые она сможет определять.

Выбранная для визуализации данного алгоритма программа V-per, имеет в своем составе библиотеку моделей реальных роботов, которые могут быть использованы при создании физической модели для сортировки мусора.

СПИСОК ЛИТЕРАТУРЫ

1. Бобович Б.Б. Переработка промышленных отходов: Учебник для вузов. / Б.Б. Бобович - М.: «СП Интермет Инжиниринг», 1999. - 445 с.
2. Глазунов, В.А. Пространственные механизмы параллельной структуры. / Глазунов В.А., Колискор А.Ш., Крайнев А.Ф. - М. Наука, 1991. – 94 с.
3. Круглов, В.В. Искусственные нейронные сети. Теория и практика. / В. В. Круглов, В.В. Борисов - М.: Горячая линия - Телеком, 2001. - 382 с
4. Николенко С.И. Глубокое обучение. Погружение в мир нейронных сетей. / С.И. Николенко - Питер, 2018.- 480 с.
5. AMP Robotics [Электронный ресурс]. <https://www.amprobotics.com/>, дата обращения 14.02.2023
6. Coppelia Robotics V-REP [Электронный ресурс]. <http://www.coppeliarobotics.com/>, дата обращения: 18.03.2023
7. ImageAI [Электронный ресурс]. <http://imageai.org/>, дата обращения: 20.02.2023
8. OpenCV [Электронный ресурс]. <https://opencv.org/>, дата обращения: 02.03.2023
9. ZenRobotics [Электронный ресурс]. <https://www.terex.com/zenrobotics/>, дата обращения 13.02.2023

ПРИЛОЖЕНИЕ А

Скрипт Python для обучения ИНС:

```
from imageai.Prediction.Custom import ModelTraining
model_trainer = ModelTraining()
model_trainer.setModelTypeAsResNet()
model_trainer.setDataDirectory(r"D:/SomeFilesForTest/AI-dataset")
model_trainer.trainModel(num_objects=2,num_experiments=2,
batch_size=32, show_network_summary=True)
```

Скрипт Python для распознавания видео с помощью OpenCV:

```
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import cv2

ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
                help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
                help="path to Caffe pre-trained model")
ap.add_argument("-c", "--confidence", type=float, default=0.2,
                help="minimum probability to filter weak detections")
args = vars(ap.parse_args())
CLASSES = ["bottle", "cup", "person "]
COLORS = np.random.uniform(0, 255, size=(len(CLASSES), 3))
print("[INFO] loading model...")
```

```

net = cv2.dnn.readNetFromCaffe('MobileNetSSD_deploy.prototxt.txt',
'MobileNetSSD_deploy.caffemodel')

print("[INFO] starting video stream...")

vs = VideoStream(src=0).start()

time.sleep(2.0)

fps = FPS().start()

while True:

    frame = vs.read()

    frame = imutils.resize(frame, width=400)

    (h, w) = frame.shape[:2]

    blob = cv2.dnn.blobFromImage(cv2.resize(frame, (300, 300)),
        0.007843, (300, 300), 127.5)

    net.setInput(blob)

    detections = net.forward()

    for i in np.arange(0, detections.shape[2]):

        confidence = detections[0, 0, i, 2]

        if confidence > args["confidence"]:

            idx = int(detections[0, 0, i, 1])

            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])

            (startX, startY, endX, endY) = box.astype("int")

            label = "{}: {:.2f}%".format(CLASSES[idx],
confidence * 100)

            cv2.rectangle(frame, (startX, startY), (endX, endY),
                COLORS[idx], 2)

            y = startY - 15 if startY - 15 > 15 else startY + 15

            cv2.putText(frame, label, (startX, y),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, COLORS[idx], 2)

            cv2.imshow("Frame", frame)

            key = cv2.waitKey(1) & 0xFF

```

```
        if key == ord("q"):
            break
        fps.update()
fps.stop()
print("[INFO] elapsed time: {:.2f}".format(fps.elapsed()))
print("[INFO] approx. FPS: {:.2f}".format(fps.fps()))
cv2.destroyAllWindows()
vs.stop()
```