

Н. Н. Самылкина

Московский педагогический государственный университет, г. Москва, Россия

И. А. Калинин

Московский государственный лингвистический университет, г. Москва, Россия

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДЛЯ ПОДГОТОВКИ К ОЛИМПИАДЕ ШКОЛЬНИКОВ ПО ИСКУССТВЕННОМУ ИНТЕЛЛЕКТУ*

Аннотация

Актуальность олимпиадной подготовки школьников по программированию определяется и тем, что олимпиадные достижения школьников по программированию позволяют реализовать альтернативный формат поступления в вуз, и тем, что такая подготовка обеспечивает раннее профессиональное самоопределение учащихся. Появилось большое количество новых олимпиадных треков в технологических отраслях с индивидуальным и командным участием школьников, однако учебно-методических материалов для подготовки к новым олимпиадам недостаточно. Зачастую контент новых олимпиад слабо структурирован, отсутствует концептуальная идея, соответствующая специализации олимпиады. Все это является проблемой для учителей при организации целенаправленной подготовки к олимпиадам Российского совета олимпиад школьников (РСОШ). В настоящее время Всероссийская олимпиада школьников (ВСОШ) по информатике содержит четыре профиля: программирование, искусственный интеллект, информационная безопасность и робототехника.

Структура и состав учебно-методического обеспечения для подготовки к олимпиаде школьников по искусственному интеллекту (ВСОШ и РСОШ) должны соответствовать ее специализации и этапам олимпиады. Анализ включенных в различные этапы олимпиады заданий прошлых лет позволяет учителям определить основные темы для подготовки новых заданий или для отбора и адаптации существующих. Таким образом, можно подготовить несколько учебных курсов различной сложности с типовыми тренировочными заданиями для каждого этапа олимпиады.

В данной статье приводятся примеры типовых заданий для каждого этапа олимпиады школьников по искусственному интеллекту с подробным разбором и решением.

Статья будет полезна учащимся, целенаправленно готовящимся к олимпиаде по искусственному интеллекту, учителям информатики, студентам педагогических вузов.

Ключевые слова: искусственный интеллект, олимпиада по ИИ, информатика, программирование на Python, курс по выбору, углубленный уровень, траектория обучения, тренировочные задания, этапы олимпиады.

Информация об авторах

Самылкина Надежда Николаевна, доктор пед. наук, доцент, профессор кафедры теории и методики обучения математике и информатике, Институт математики и информатики, Московский педагогический государственный университет, г. Москва, Россия; *e-mail:* nsamylkina@yandex.ru

Калинин Илья Александрович, канд. пед. наук, доцент, начальник управления информатизации, Департамент цифрового развития, Московский государственный лингвистический университет, г. Москва, Россия; доцент кафедры международной информационной безопасности, Институт информационных наук, Московский государственный лингвистический университет, г. Москва, Россия; *e-mail:* kalininilya@mail.ru

* Материалы к статье можно скачать на сайте ИНФО: https://infojournal.ru/journals/school/school_05-2025/

N. N. Samylkina

Moscow Pedagogical State University, Moscow, Russia

I. A. Kalinin

Moscow State Linguistic University, Moscow, Russia

TEACHING AND LEARNING RESOURCES FOR PREPARING SCHOOLCHILDREN FOR THE ARTIFICIAL INTELLIGENCE OLYMPIAD

Abstract

The relevance of preparing schoolchildren for programming competitions is determined by both the fact that competitive achievements in programming offer an alternative format for university admission and the fact that such training facilitates early professional self-determination. A large number of new Olympiad tracks have emerged in technology fields, with individual and team participation for school students. However, the teaching and learning materials for preparing for these new Olympiads are insufficient. The content of these new Olympiads is often poorly structured, lacking a conceptual framework that aligns with the Olympiad's specialization. This presents a challenge for teachers when organizing targeted preparation for Olympiads organized by the Russian Council of School Olympiads. Currently, the All-Russian Olympiad in Informatics has four profiles: programming, artificial intelligence, information security, and robotics.

The structure and composition of the teaching and methodological support for preparing for the AI Olympiad should correspond to its specialization and stages. An analysis of the tasks included in the various stages of the Olympiad from previous years allows us to identify key topics for developing new tasks or for selecting and adapting existing ones. This allows us to prepare several training courses of varying difficulty levels with standard practice tasks for each stage of the Olympiad.

The article provides examples of typical tasks for each stage of the AI Olympiad for schoolchildren, with detailed analysis and solutions.

This article will be useful for students specifically preparing for the Artificial Intelligence Olympiad, informatics teachers, and students of pedagogical universities.

Keywords: artificial intelligence, AI Olympiad, informatics, Python programming, elective course, advanced level, learning trajectory, training tasks, stages of Olympiad.

Information about the authors

Nadezhda N. Samylkina, Doctor of Sciences (Education), Docent, Professor at the Department of Theory and Methods of Teaching Mathematics and Informatics, Institute of Mathematics and Informatics, Moscow Pedagogical State University, Moscow, Russia; e-mail: nsamylkina@yandex.ru

Ilya A. Kalinin, Candidate of Sciences (Education), Docent, Head of Informatization Administration, the Department of Digital Development, Moscow State Linguistic University, Moscow, Russia; Associate Professor at the Department of International Information Security, Institute of Information Sciences, Moscow State Linguistic University, Moscow, Russia; e-mail: kalininilya@mail.ru

1. Актуальность олимпиадной подготовки по искусственному интеллекту

Олимпиадные достижения как альтернативная форма поступления школьников в российские вузы всегда были в центре внимания участников образовательных отношений, а в последние годы эта тема набрала пиковую популярность. Помимо прагматичных результатов для учащихся олимпиадная подготовка по информатике сама по себе является действенным инструментом ранней профилизации учащихся — она позволяет заложить фундаментальные основы будущей профессии в высокотехнологических отраслях экономики.

Инженерная подготовка для любой технологической отрасли начинается в школе, когда формируется мировоззрение ученика, когда он может сразу попробовать применить свои знания и умения для решения реальных задач. Такая возможность сейчас существует в области искусственного интеллекта и смежных с ним направлениях (информационная безопасность, робототехника и пр.).

В последние годы расширяется список олимпиад Российского совета олимпиад школьников (РСОШ) — в него включены олимпиады по искусственному интеллекту, информационной безопасности, криптографии, робототехнике и большое количество треков национальных технологических олимпиад (НТО) [4, 8]. Всероссийская олимпиада школьников (ВсОШ) по информатике с текущего года также содержит четыре профиля: программирование, искусственный интеллект, информационная безопасность и робототехника.

В соответствии с требованиями к организации этих олимпиад их учебно-методическое обеспечение должно быть доступным на сайтах олимпиад, однако чаще всего доступ предоставляют только участникам. Учащимся и педагогам непросто сделать первый шаг и сразу стать участниками или наставниками, всегда хочется заранее оценить свои возможности. Олимпиадного материала всегда недостаточно, возможности его самостоятельного создания могут быть ограничены, поскольку тематика не только новая, но и сложная.

При этом различаются подходы к постановке и тематике заданий на олимпиадах по искусственному интеллекту, проводимых разными вузами. Различается и формат проведения олимпиад — они могут быть

командными (с выделением индивидуальных достижений) и индивидуальными.

Поскольку профильные олимпиады ВсОШ и РСОШ по искусственному интеллекту являются новыми, а тематических направлений, относящихся к искусственному интеллекту, много, до настоящего времени наблюдается некий разброс мнений о том, каким должен быть непосредственно олимпиадный контент и, следовательно, типовые тренировочные задания [1, 4, 5].

2. Структура и состав учебно-методического обеспечения для подготовки к олимпиаде школьников по искусственному интеллекту

Олимпиадная подготовка используется в школе для развития способностей учащихся по математике, программированию и в некоторых других областях по выбору обучающихся, поэтому вполне обоснованно можно говорить о возможности реализации индивидуальных образовательных траекторий в соответствии с ФГОС общего образования [6, 7].

Выбрав программирование в качестве специализации, учащиеся и их родители оказываются перед задачей оценки имеющегося у подростка олимпиадного потенциала с точки зрения его возможной реализации.

В качестве обязательных отборочных шагов в условиях массовой школы необходимо обязательно пройти *школьный и муниципальный этапы Всероссийской олимпиады школьников по информатике (ВсОШ)*.

Прошедшим этот отбор учащимся можно целенаправленно заниматься олимпиадной подготовкой и начать с *годового интенсива по программированию на Python (в VII классе)*.

Далее наставник может включить в календарь школьных олимпиад *Всероссийскую олимпиаду школьников по искусственному интеллекту для VIII—XI классов из перечня РСОШ*, чтобы ученики попробовали свои силы в этом направлении. Первый год участия в этой олимпиаде в VIII классе в большей степени будет тренировочным.

При этом лучше *параллельно участвовать во Всероссийской олимпиаде школьников по информатике*, продвигаясь дальше по ее этапам и совершенствуя свои навыки спортивного программирования.

Структура и контент тренировочных заданий для подготовки к участию в любой олимпиаде по тематике и трудности заданий должны соответствовать этапам олимпиады. Подготовка к олимпиаде по искусственному интеллекту может быть реализована в очном курсе по выбору или во внеурочной деятельности с электронной поддержкой в виде дистанционного курса на любой удобной платформе.

Согласно предложенной концепции для первой Всероссийской олимпиады школьников по искусственному интеллекту, олимпиадные задания должны формировать индивидуальные траектории подготовки в области ИИ (тематическая преемственность и практическая направленность заданий с применением дата-сетов на 2-м и 3-м турах) [1, 11]. Интеллектуальные методы для решения задачи не должны содержаться в условии — они должны подбираться учащимися [2]. При этом используется рейтинговая оценка (как в известном профессиональном сообществе Kaggle.com) индивидуальных возможностей школьников в комплексном применении математических и цифровых навыков для реализации задач ИИ.

Олимпиада школьников по искусственному интеллекту состоит из трех этапов.

На первом этапе проверяются основные навыки программирования на Python, необходимые для решения задач ИИ, по темам, которые получают свое развитие на следующих этапах. Python — пока единственный доступный учащимся язык программирования, для которого есть полный набор средств решения задач искусственного интеллекта [4].

Целью первого этапа олимпиады является отбор участников, владеющих в необходимом объеме навыками решения задач на языке Python с использованием его стандартных функций и библиотек (например, регулярных выражений) [10]. Здесь можно проверить знания и умения по таким темам:

- Текстовый ввод и вывод.
- Переменные и основные типы данных: int, float, string.
- Условия, циклы, перебор.
- Классы и объекты: создание, точечная нотация, методы и свойства.
- Структуры данных: кортеж, список, словарь.
- Модули и пакеты.
- Работа с текстовыми файлами.
- Регулярные выражения.
- Функции и параметры.

На втором, основном, этапе олимпиады лучше всего использовать задания для выделения тех участников, которые действительно интересуются основными методами интеллектуальной обработки данных [12–14], специализируются на них или могут быстро с ними ознакомиться. Предусматривается возможность использования библиотек Numpy, Pandas, Pillow или OpenCV, sklearn, MathPlotLib. Задания оценивают умения применить специализированные библиотеки для решения ранее не встречавшейся задачи.

Необходимо использовать следующие классы задач:

- задачи классификации;
- задачи кластеризации;

- задачи выявления ассоциативных правил;
- задачи регрессии.

На третьем этапе участникам предлагается или одно задание, или два задания, из которых необходимо выполнить хотя бы одно.

В настоящее время олимпиадные задания на разных этапах могут быть перемешаны, т. е. уже на первом этапе может потребоваться специализированная библиотека или на втором этапе необходимо строить и обучать модель. Но при целенаправленной работе с учащимися по подготовке к олимпиаде по ИИ лучше придерживаться описанного содержания заданий различных этапов, чтобы учащиеся постепенно наращивали свои навыки — как математические, так и навыки программирования. При непосредственном участии в олимпиаде учащимся будет легче сориентироваться.

3. Разбор и решение тренировочных заданий для каждого этапа олимпиады

Задачи первого этапа олимпиады по искусственному интеллекту могут быть решены в любой среде программирования с поддержкой Python. Однако для решения последующих задач, более эффективного исследования и устранения ошибок в программном коде удобно использовать развитый дистрибутив, ориентированный на работу с языком Python и решение задач Data Science, — Anaconda.

В составе дистрибутива есть и популярная среда для решения задач интеллектуального анализа Jupyter Notebook, и интегрированная среда разработки Spyder, включающая в себя специализированный текстовый редактор, средства работы с интерпретатором Python, пошаговое исполнение, просмотр переменных и др.

3.1. Задание первого этапа олимпиады

Рассмотрим пример задачи первого этапа с возможным решением.

Условие задачи.

Напишите программу ввода целого числа (от –999999 до 999999) и вывода его текстовой формулировки на русском языке.

Разбор решения.

Часто встречающаяся задача, в рамках которой можно продемонстрировать средства стандартного ввода и вывода данных, работу с целыми числами и строками. В случае применения языка Python задача позволяет рассмотреть работу со списками, словарями, а также использование функций для структурирования кода.

Решение опирается на правила записи числительных в русском языке: фактически название числа конструируется из количества единиц (числа от 1 до 999), тысяч, миллионов и т. д. В рассматриваемой задаче к этой особенности добавляются также отрицательные числа (их запись предваряется словом «минус»).

Число от 1 до 999 преобразуется в текстовую форму как название количества сотен, десятков и единиц, например, «триста двадцать два». Этот принцип нарушается только для чисел от 11 до 19 — у них есть самостоятельные названия.

Таким образом, для решения задачи мы:

- 1) Выделяем знак числа.
- 2) Разбиваем число на две части — тысячи и единицы.
- 3) Каждую из этих частей записываем словами.

Соединяем части, вставив слово «тысяч», — конечно, если тысячи в числе вообще есть. Заметим, что в зависимости от числа тысяч у нас может меняться окончание слова: одна тысячаА, две (три, четыре) тысячИ, от 6 до 9 тысяч.

Решение задачи.

Подготовим функцию преобразования числа от 1 до 999 в текстовый вид. Принцип работы этой функции довольно прост: мы выделяем единицы, десятки и сотни, после чего преобразуем их в слова.

Слова (если есть нужные части) добавим в список, из которого потом и соберем число (листинг 1)*.

Единственной особенностью этой функции являются два дополнительных параметра — названия чисел 1 и 2 — для правильного использования в названии тысяч. По умолчанию это обычные названия для чисел от 1 до 999.

Для определения правильного склонения слова «тысяча» по количеству тысяч напишем несложную вспомогательную функцию (листинг 2).

Теперь можно написать и основную программу (листинг 3).

* Листинги можно скачать в архиве дополнительных материалов к статье на сайте ИНФО:
https://infojournal.ru/journals/school/school_05-2025/

```
def intToText( num = 0, oneName = 'один' , twoName = 'два' ):
# Словарь словарей для выбора диапазона названий по основанию деления и числу
    numText = { 1 : { 0 : 'ноль', 1 : oneName, 2 : twoName, 3 : 'три', 4 : 'четыре',
                      5 : 'пять', 6 : 'шесть', 7 : 'семь', 8 : 'восемь', 9 : 'девять' },
                20 : { 11 : 'одиннадцать', 12 : 'двенадцать', 13 : 'тринадцать',
                      14 : 'четырнадцать', 15 : 'пятнадцать', 16 : 'шестнадцать',
                      17 : 'семнадцать', 18 : 'восемнадцать', 19 : 'девятнадцать' },
                100 : { 1 : 'десять', 2 : 'двадцать', 3 : 'тридцать', 4 : 'сорок',
                      5 : 'пятьдесят', 6 : 'шестьдесят', 7 : 'семьдесят',
                      8 : 'восемьдесят', 9 : 'девяносто' },
                1000 : { 1 : 'сто', 2 : 'двести', 3 : 'триста', 4 : 'четыреста',
                      5 : 'пятсот', 6 : 'шестьсот', 7 : 'семьсот', 8 : 'восемьсот',
                      9 : 'девятьсот' } }

    result = [] # Список частей, изначально пустой
    if num //100 in numText[100]: # Если есть сотни
        result.append(numText[100][num//100]) # Добавляем название количества сотен
        num = num % 100 # Убираем все сотни из числа
    if num //10 in numText[10]: # Если есть десятки
        if num //10 == 1 and num %10 != 0: # Если остаток - от 11 до 19
            result.append(numText[20][num]) # Добавляем название от 11 до 19
            num = -1 # В этом случае преобразование закончено
        else:
            result.append(numText[10][num//10]) # Десятков больше 1, добавляем название
            num = num % 10 # выделяем единицы
    if num >=0 in numText[1]: # Если единицы есть
        result.append(numText[1][num])# Добавляем название единиц
    return ' '.join(result) # Собираем название из списка - соединяем названия по разрядам
                                через пробел в строку через пробел
```

Листинг 1

```
def getEnd1000 ( num = 1):
    num = num %100
    if num %100 >9 and num %100 <=20:
        return 'тысяч'
    else:
        num = num %10
        if num == 1:
            return 'тысяча'
        if num >= 2 and num <=4:
            return 'тысячи'
        return 'тысяч'
```

Листинг 2

```

digit = - 1000000
while digit < -999999 or digit > 999999: # Вводим число, соответствующее условиям
    digit = int( input('Введите целое число от -999999 до 999999:') )

sign = -1 if digit < 0 else 1 # Определяем и сохраняем знак
thousand = abs(digit) // 1000 # Количество тысяч
entities = abs(digit) % 1000 # Количество единиц до 1000

print ( ('минус ' if sign == -1 else ' ') + (intToText( thousand, oneName = 'одна' ,
twoName = 'две' ) + ' ' + getEnd1000(thousand) + ' ' if thousand > 0 else '') +
intToText(entities) ) # Собираем строку, учитывая знак и количество тысяч (если они есть)

```

Листинг 3

3.2. Задание для подготовки ко второму (основному) этапу

Для решения задач второго этапа необходима не только среда разработки, но и дополнительные библиотеки. В составе дистрибутива Anaconda есть большая часть этих библиотек, но если их все-таки не окажется, их можно будет установить стандартными средствами Python.

Приведем пример задачи основного этапа с разбором решения.

Условие задачи.

Вам предоставили две фотографии в формате JPEG: фотографию кота (единственный крупный объект на фото, контрастный, фон светлый) и фотографию некоторого другого помещения. Сделайте фотографию-коллаж, т. е. разместите фото кота на новом фоне. Программ редактирования изображений в вашем распоряжении нет, есть только Python с установленной библиотекой Pillow и программа просмотра графических файлов.

Разбор решения.

Библиотека Pillow предназначена для работы с изображениями в растровом формате. Таким образом, изображение, которое мы прочитаем, будет набором точек, и где именно там кот, как он выглядит, мы заранее знать не можем. Тем более у нас нет возможности точно описать контур кота.

Для решения таких задач в рамках библиотеки у нас есть инструмент «Маска», который позволяет нам определить меру прозрачности каждой точки на изо-

бражении. Инструмент применяется при наложении двух изображений. Если точка полностью прозрачна, при наложении маски ее будет не видно (т. е. будет видна точка другого изображения). Если точка полностью непрозрачна, то будет видна точка основного изображения.

Общий алгоритм, который можно применить, используя графический редактор, таков:

- 1) Подбираем разницу, которая позволяет нам выделить фон (на основании того, что фон светлый и кот от него заметно отличается).
- 2) Создаем новую картинку, на которой точки кота — белые, а точки фона — черные.
- 3) Размываем эту картинку (чтобы не возникло резкой границы между котом и новым фоном).
- 4) Накладываем кота с маской на новый фон.

Для решения задачи:

- 1) Загружаем файл с изображением кота.
- 2) Подбираем границу и создаем черно-белое изображение (на котором выделится фон).
- 3) Очищаем изображение от мелких артефактов — например, когда в шерсти кота есть почти белые шерстинки.
- 4) Загружаем новый фон.
- 5) Вставляем на него изображение кота с подготовленной маской.

Решение задачи.

Решение представлено в листинге 4.

Результат получится примерно такой, как показано на рисунке 1.

```

from PIL import Image, ImageFilter, ImageOps

filename_cat = "cat3.jpg"

with Image.open(filename_cat) as img_cat:
    img_cat.load()

threshold = 210 # Пороговое значение
# Создаем заготовку маски - отмечаем черным все светлые точки
step_1 = img_cat.point(lambda x: 0 if x > threshold else 255)
# Конвертируем изображение в черно-белое без оттенков
step_1 = step_1.convert("1")

```

Листинг 4

```

# Шаг 2 - убираем мелкие (в одну-две точки) артефакты
# Первый фильтр - убирает мелкие белые точки, выбирает минимальный уровень
step_2 = step_1.filter(ImageFilter.MinFilter(3))
# Второй фильтр - убирает мелкие черные точки, выбирает максимальный уровень
cat_mask = step_2.filter(ImageFilter.MaxFilter(3))
# Полученную маску конвертируем в градации серого
cat_mask = cat_mask.convert("L")
# Размываем изображение - оно изменится только на границах черного и белого
cat_mask = cat_mask.filter(ImageFilter.BoxBlur(10))

# Загружаем новый фон
filename_monastery = "monastery.jpg"
with Image.open(filename_monastery) as img_monastery:
    img_monastery.load()

# Вставляем на новый фон изображение, уменьшив его до 20% исходного размера
# с наложением подготовленной маски
img_monastery.paste(
    img_cat.resize((img_cat.width // 5, img_cat.height // 5)),
    (1300, 750),
    cat_mask.resize((cat_mask.width // 5, cat_mask.height // 5)),
)

# Показываем результат
img_monastery.show()

```

Листинг 4 (окончание)

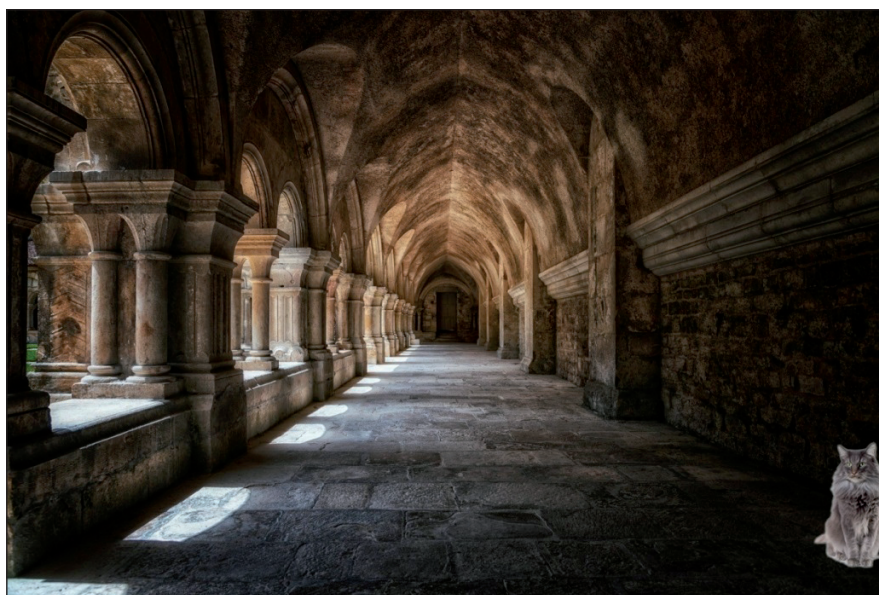


Рис. 1. Результат решения задачи второго этапа

3.3. Задание третьего (заключительного) этапа олимпиады и подходы к его решению

Для подготовки к заключительному этапу олимпиады могут понадобиться следующие специализированные библиотеки:

- SciKitLearn;
- TensorFlow / PyTorch;
- средства визуализации.

В качестве примера задачи заключительного этапа предлагается рассмотреть одну задачу машинного зрения.

Условие задачи.

Программист Никодим любит программировать дома, обедая бутербродами. Бутерброды стоят на отдельном столе.

Кот Бит программиста Никодима тоже любит бутерброды.

Когда Никодим погружается в любимую работу, он не успевает вовремя сказать «Брысь!» своему коту и часто остается без бутербродов. Он обзывает кота Мегабайтом, но все равно огорчается.

Наблюдать за котом и бутербродами Никодим не может, потому что он работает. Говорить Биту «Брысь!» просто так (когда кот просто ходит вдоль стола или сидит на табуретке) нельзя, потому что кот тогда перестанет на это реагировать.

Никодим установил старый ноутбук с камерой для наблюдения за бутербродами, записал свой голос, но ему все равно приходится отвлекаться от работы, чтобы нажать на кнопку, — и кот успевает (за четыре секунды) стащить колбасу.

Поскольку искусственным интеллектом Никодим не занимается, он просит написать ему программу, которая будет получать фото стола с бутербродами и выдавать слово «Брысь!», если кот уже лезет на стол, не более чем за три секунды. Если кота нет (или он не лезет на стол), программа должна выдавать прочерк: «-----».

Возможны следующие комбинации (части задачи):

Часть 1. В кадре — стол (а на столе тарелка с бутербродами), надо реагировать на кота на столе. Вокруг стола темный фон, на котором камера практически не позволяет различить кота. Коту на столе нечего делать.

Часть 2. В кадре не только стол, но и табуретка (иногда с котом), поэтому надо реагировать, только если край кота приближается к бутербродам. Бутерброды лежат на тарелке, тарелка круглая (но фото делается сбоку) и стоит на одном и том же месте. Никодим обещает фото-образец.

Часть 3. Все то же самое, что и в пункте 2, но тарелку Никодим может ставить в разные места. Никодим обещает только не двигать стол и ноутбук.

За каждый пункт начисляется 10 очков. На реакцию у программы есть три секунды, иначе кот успевает стащить колбасу.

Разбор решения.

Основная трудность задачи — отсутствие точного шаблона «кот» и способа его поиска. Именно такая задача относится к задачам машинного зрения, т. е. решается средствами искусственного интеллекта.

С практической точки зрения это задача выделения (т. е. сегментации) объектов на фотографии и их последующей классификации. Самостоятельно подготовить и обучить нейронную сеть для таких целей — задача трудная и требующая больших ресурсов, но мы можем воспользоваться уже обученной сетью.

Одна из наиболее развитых и эффективных свободно доступных сетей — семейство сетей YOLO, на момент решения этой задачи была доступна версия 8 семейства сетей, в частности, сеть сегментации и классификации изображений, предобученная на наборе изображений COCO.

Стоит отметить, что для работы это семейство сетей потребует загрузки довольно большого объема дополнительных средств и библиотек (например, PyTorch) — общий объем будет около 1 Гб.

При наличии такого средства задача в первой части может быть решена так:

- 1) Загрузить и подготовить к работе сеть.
- 2) Получить изображение.
- 3) Выполнить сегментацию и классификацию объектов, отбросив все сегменты с вероятностью менее 0,5 (т. е. маловероятные).
- 4) Если среди найденных сегментов есть сегмент класса «cat» (на столе кот), надо реагировать.

Решение части 1.

Решение представлено в листинге 5.

```
from ultralytics import YOLO
import cv2

# Загружаем модель среднего размера для сегментации
# При первом обращении она будет загружаться с сайта разработчика
model = YOLO("yolov8m-seg.pt")
# Загружаем названия классов объектов
yolo_classes = list(model.names.values())
classes_ids = [yolo_classes.index(clas) for clas in yolo_classes]
# Минимальная вероятность распознавания
conf = 0.5

# Будем продолжать работу, пока будут вводить имена файлов
cont = True

while cont:
    file = input('Введите имя файла:')
    cont = (file != '') # Если ввели пустую строку, прекращаем работу
    if cont:
        img = cv2.imread(file) # Читаем изображение
        hasCat = False # Есть кот? Пока не нашли...
    # Сегментируем изображение и классифицируем найденное
    results = model.predict(img, conf=conf)
    for result in results: # Перебираем все результаты
        for mask, box in zip(result.masks.xy, result.bboxes):
            # Проверяем: не кот ли нашелся?
            hasCat = hasCat or (result.names[int(box.cls[0])] == 'cat')
    print('Брысь!' if hasCat else '----') # Кот в кадре есть. Выгнать.
```

Решение части 2.

Вторая часть задачи несколько сложнее. Кот может присутствовать в кадре, но нам надо оценить расстояние от него до тарелки. Оптимальный способ — найти центр объекта «тарелка» (также это можно сделать заранее, возможно, даже вручную) и определить расстояние от крайней точки кота до этого центра.

Недостатком такого решения будет то, что если тарелка стоит ближе к краю стола, то такая окружность будет захватывать лишнее — например, кусок стены.

Поэтому более универсальным (хотя и трудоемким) решением будет описать контур границы многоугольником, т. е. цепочкой координат. При таком подходе расстояние не вычисляется, а определяется: входит ли точка кота в многоугольник?

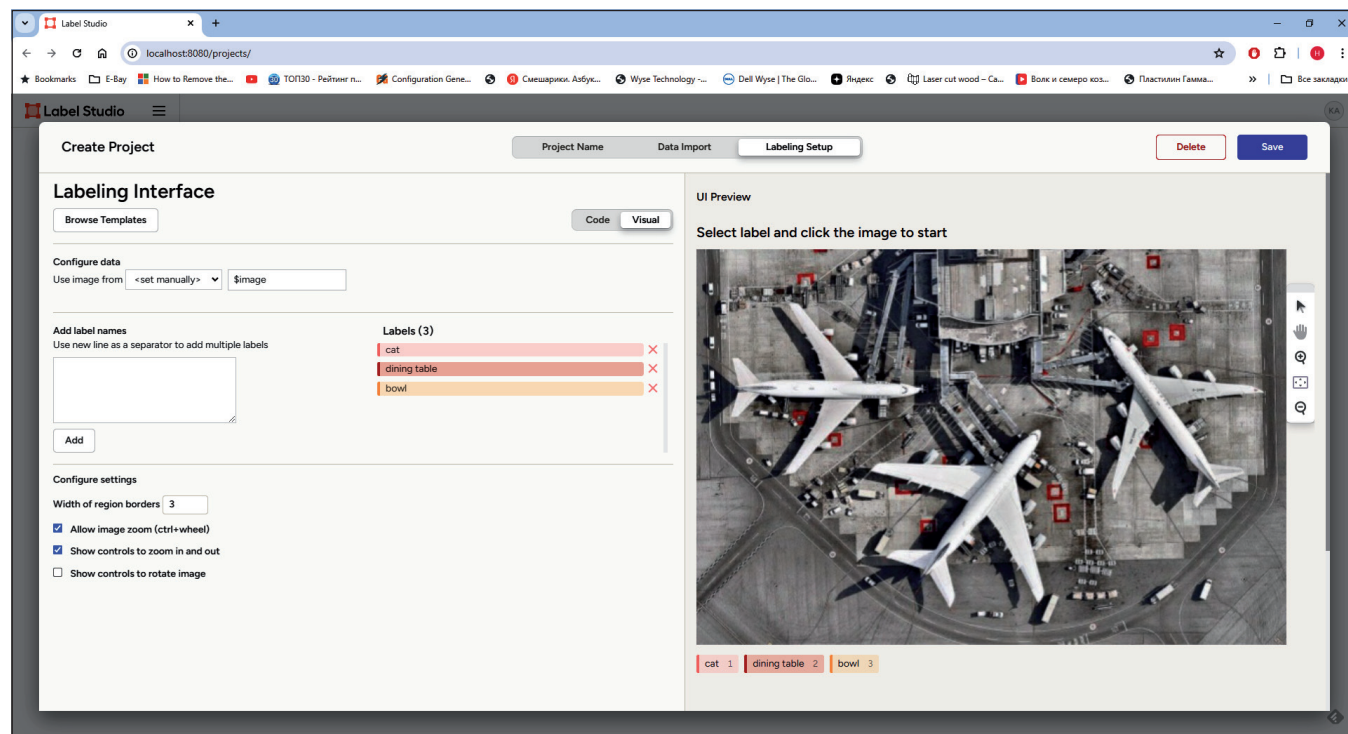


Рис. 2. Выбор категории разметки

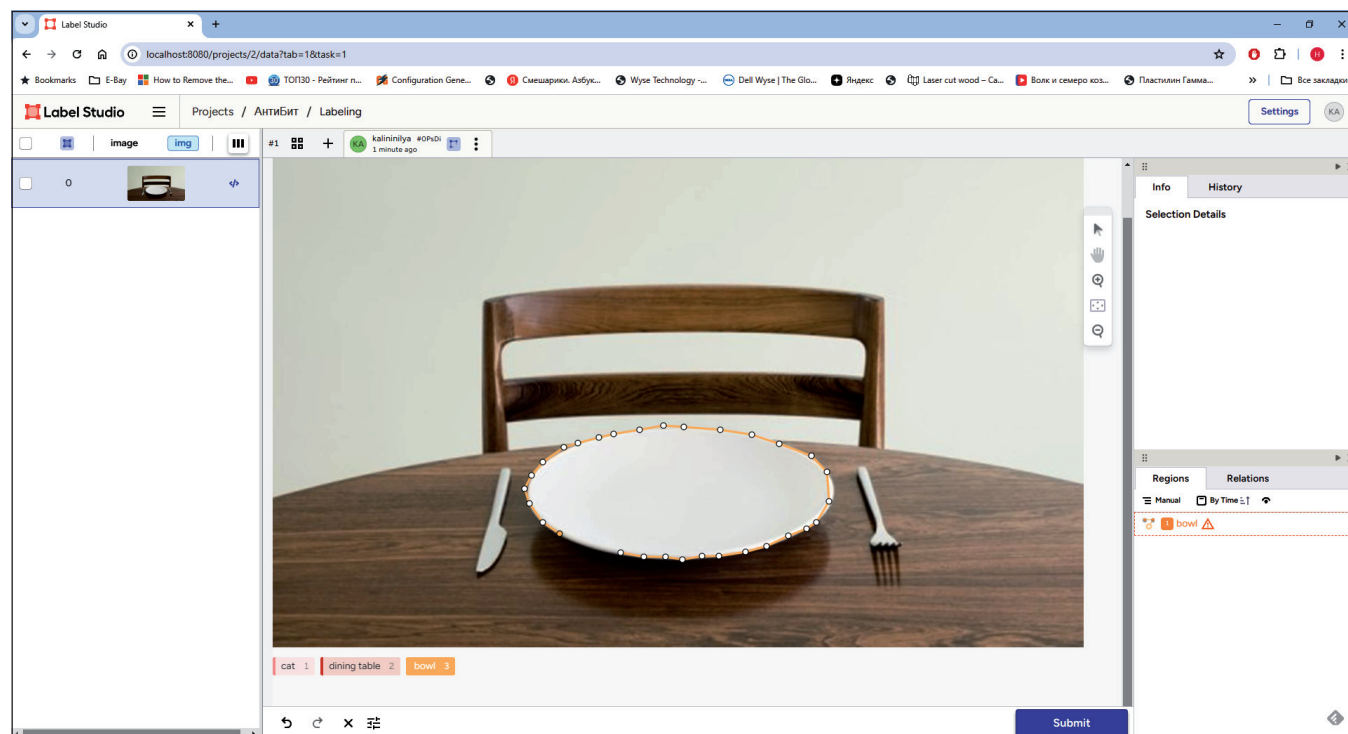


Рис. 3. Разметка сегмента для тарелки

Для предварительной разметки — описания контуров — можно установить и использовать свободно доступное программное обеспечение Label Studio (<https://labelstud.io/>).

Система разметки устанавливается в соответствии с инструкциями на сайте как приложение Python и доступна через браузер на машине пользователя.

После начала работы создадим проект, взяв за основу задачи компьютерного зрения семантическую разметку многоугольниками.

При создании проекта заранее определим категории разметки: Dinning Table (*англ.* обеденный стол) и Bowl (*англ.* тарелка) (именно такие есть в списке категорий YOLO) (рис. 2).

Результат сохраним. Теперь откроем фотографию, выберем категорию и поставим точки по контуру объекта. Последнюю точку совместим с первой и получим размеченный объект — тарелку (рис. 3).

Аналогично выделим стол.

Формат предназначен для обучения, но мы в данном каталоге найдем файл с координатами и импортируем его в нашу программу. Координаты в файле записываются подряд (X1,Y1, X2, Y2 и т. д.), так что мы преобразуем результат в словарь списков координат (листинг 6). Список приведем не полностью, поскольку без исходной фотографии он в любом случае только иллюстрация.

Обратите внимание: координаты заданы не в пикселях (потому что размеры распознаваемых картинок могут быть очень разными), а в десятичных дробях от 0 до 1.

Алгоритм работы мы уже описали, но есть особенность: если мы будем использовать готовые средства определения внутренней точки (в составе SymPy — библиотеки символьной математики для Python), то время работы программы для одной фотографии превысит пять секунд, что неприемлемо.

```
borders = {
    'bowl' : [ (0.34948096885813146,0.7715327784531937), (0.33044982698961933,0.750490975404470
2), (0.3157439446366782,0.7182268773964275), (0.3122837370242214,0.6831572056485551), (0.3209
342560553633,0.6565042551201721), (0.3347750865051903,0.6298513045917891), (0.36072664359861
59,0.6003927803235761), (0.3780276816608996,0.5919760591040868), (0.40397923875432523,0.5807
537641447676), (0.4221453287197232,0.5737398297951931), (0.4541522491349481,0.56532310857570
37), (0.4835640138408304,0.5569063873562143), (0.5086505190311419,0.5597119610960442), (0.553
6332179930795,0.5653231085757037)],
    'dinning table' : [ (0.02076124567474045,0.6941924623585524), (0.050173010380622794,0.67876
18067894886), (0.07958477508650515,0.670345085569999), (0.11764705882352938,0.65772000374076
5), (0.1375432525951557,0.6479004956513608), (0.1548442906574394,0.6422893481717011), (0.1929
0657439446363,0.6366782006920415), (0.2404844290657439,0.6226503319928925), (0.2647058823529
411,0.6100252501636585), (0.3019031141868512,0.6058168895539138)]
}
```

Листинг 6

```
def angleToLine( p , a,b ): # Угол из точки на отрезок
    ax,ay = a[0] - p[0], a[1] - p[1]
    bx,by = b[0] - p[0], b[1] - p[1]

    if (ax == 0.0 and ay == 0.0) or ( bx == 0.0 and by == 0.0 ):
        return 0 # Точка на вершине

    len_a, len_b = math.sqrt( ax*ax+ay*ay ), math.sqrt( bx*bx+by*by )
    scalar_prod = ax*bx+ay*by
    cos_gamma = scalar_prod/( len_a * len_b ) # Косинус угла

    # Вернуть угол с учетом знака векторного произведения
    return ( -1 if (ax*by-ay*bx) <0 else 1) * round(math.acos( cos_gamma ),10)

def innerPoint ( p, poly ):

    angle = 0
    lenP = len(poly)
    for i in range( lenP ):
        angle += angleToLine(p , poly[i] , poly[ 0 if i== (lenP-1) else i+1 ])
    print(angle)
    return abs(round(angle/math.pi,6)) == 2 # Если угол 2Pi - точка внутри
```

Листинг 7

Таким образом, нам понадобится своя реализация алгоритма определения внутренней точки многоугольника (причем форма многоугольника может быть любой). Самый распространенный алгоритм решения такой задачи — трассировка луча (подсчет пересечений из нашей точки), но этот алгоритм имеет целый ряд осо-

бенностей, из-за которых плохо работает для сложных многоугольников.

Воспользуемся более универсальным (хотя и более медленным) алгоритмом суммирования углов (листинг 7).

Таким образом, наша программа выглядит примерно так, как представлено в листинге 8.

```
from ultralytics import YOLO
import math
from PIL import Image

def angleToLine( p , a,b ): # Угол из точки на отрезок
    ax,ay = a[0] - p[0], a[1] - p[1]
    bx,by = b[0] - p[0], b[1] - p[1]

    if (ax == 0.0 and ay == 0.0) or ( bx == 0.0 and by == 0.0 ):
        return 0 # Точка на вершине

    len_a, len_b = math.sqrt( ax*ax+ay*ay ), math.sqrt( bx*bx+by*by )
    scalar_prod = ax*bx+ay*by
    cos_gamma = scalar_prod/( len_a * len_b ) # Косинус угла

    # Вернуть угол с учетом знака векторного произведения
    return ( -1 if (ax*by-ay*bx) <0 else 1) * round(math.acos( cos_gamma ),10)

def innerPoint ( p, poly ):

    angle = 0
    lenP = len(poly)
    for i in range( lenP ):
        angle += angleToLine(p , poly[i] , poly[ 0 if i== (lenP-1) else i+1 ])
    print(angle)
    return abs(round(angle/math.pi,6)) == 2 # Если угол 2Pi - точка внутри

borders = {
    'bowl' : [ (0.34948096885813146,0.7715327784531937),(0.33044982698961933,0.750490975404470
    2),(0.3157439446366782,0.7182268773964275),(0.3122837370242214,0.6831572056485551),(0.3209
    342560553633,0.6565042551201721),(0.3347750865051903,0.6298513045917891),(0.36072664359861
    59,0.6003927803235761),(0.3780276816608996,0.5919760591040868),(0.40397923875432523,0.5807
    537641447676),(0.4221453287197232,0.5737398297951931),(0.4541522491349481,0.56532310857570
    37),(0.4835640138408304,0.5569063873562143),(0.5086505190311419,0.5597119610960442),(0.553
    6332179930795,0.5653231085757037)],
    'dinning table' : [ (0.02076124567474045,0.6941924623585524),(0.050173010380622794,0.67876
    18067894886),(0.07958477508650515,0.670345085569999),(0.11764705882352938,0.65772000374076
    5),(0.1375432525951557,0.6479004956513608),(0.1548442906574394,0.6422893481717011),(0.1929
    0657439446363,0.6366782006920415),(0.2404844290657439,0.6226503319928925),(0.2647058823529
    411,0.6100252501636585),(0.3019031141868512,0.6058168895539138)]
}

# Загружаем модель среднего размера для сегментации
# При первом обращении она будет загружаться с сайта разработчика
model = YOLO("yolov8m-seg.pt")
# Загружаем названия классов объектов
yolo_classes = list(model.names.values())
classes_ids = [yolo_classes.index(clas) for clas in yolo_classes]
# Минимальная вероятность распознавания
conf = 0.5
```

Листинг 8

```

# Будем продолжать работу, пока будут вводить имена файлов
cont = True
while cont:
    file = input('Введите имя файла:')
    cont = (file != '') # Если ввели пустую строку, прекращаем работу
    if cont:
        img = Image.open(file) # Читаем изображение
        xS, yS = img.size
        # Запретные зоны
        alarmZone = {}
        for zone in borders:
            # пересчитать зоны под файл
            alarmZone[zone] = [ (int(b[0]*xS),int(b[1]*yS)) for b in borders[zone] ]
        hasCat = False # Есть кот? Пока не нашли...
# Сегментируем изображение и классифицируем найденное
results = model.predict(img, conf=conf)
for result in results: # Перебираем все результаты
    for mask, box in zip(result.masks.xy, result.bboxes):
        # Проверяем: не кот ли нашелся?
        if result.names[ int(box.cls[0]) ] == 'cat':
            alarm = False # Надо выгонять этого кота?
            for point in mask:
                point = (int(point[0]),int(point[1]))
                for zone in alarmZone: # Кот в зоне тревоги?
                    alarm = alarm or innerPoint( point, alarmZone[zone] )
            hasCat = hasCat or alarm # Какой-то кот в зоне тревоги?
        if hasCat :
            break
    print( 'Врысь!' if hasCat else '----') # Выгнать, если кот в запретной зоне

```

Листинг 8 (окончание)

Решение части 3.

Предложенное решение позволяет нам перейти к третьей части задачи.

Сложность этой части в том, что нужно не просто воспользоваться готовыми контурами, а сначала найти контуры стола и тарелки.

При этом стандартная сеть далеко не всегда распознает на фотографиях тарелку (bowl) и стол (dinner table), особенно в ситуации, когда от них видны только куски. Сеть необходимо дообучить.

Дообучение сети выполняется отдельной программой, и для этого потребуется:

- 1) Подобрать (или сделать) серию фотографий (скорее всего, хватит 25–30 штук) стола и тарелки с бутербродами.
- 2) Разметить эти фотографии (т. е. вручную максимально точно определить границы объектов) и указать новые классы объектов на ней в уже рассмотренной нами Label Studio.
- 3) Экспортировать данные и подготовить программу дообучения типовой сети. (Решение подобной задачи описано, например, в статье «Обучите YOLOv8 на пользовательском наборе данных»^{*}.)
- 4) Провести обучение, в том числе подобрать метапараметры и выбрать количество эпох обучения.
- 5) Сохранить результат.

- 6) В программе распознавания не выбрать готовые контуры, а отобрать контуры объектов нужных классов. Пользуясь тем, что стол стоит на месте, его контур можно и зафиксировать.

При появлении «кота» на фото мы просто проверим, попадают ли точки «кота» в объекты «стол» и «тарелка».

4. Заключение

При отборе и разработке контента тренировочных заданий для целенаправленной подготовки к олимпиаде по искусственному интеллекту были учтены экспертные оценки и запросы педагогов, а также эмпирические результаты подготовки будущих учителей информатики в Институте математики и информатики Московского педагогического государственного университета за последние годы. Деление курса на модули в соответствии с целями каждого этапа олимпиады позволяет пошагово и целенаправленно наращивать у школьников необходимые навыки программирования, использования интеллектуальных методов и готовых сервисов при решении олимпиадных задач.

Финансирование

Работа выполнена в рамках государственного задания Министерства просвещения Российской Федерации (тема № 124052100092-0 («Вариативное обучение основам искусственного интеллекта в общем образовании на основе интегративного подхода»)).

^{*} <https://habr.com/ru/articles/714232/>

Список источников

1. Григорьев С. Г., Калинин И. А., Самылкина Н. Н. Система заданий для первой всероссийской олимпиады школьников по искусственному интеллекту // Информатика и образование. 2022. Т. 37. № 3. С. 12–20. EDN: FYGJPK. DOI: 10.32517/0234-0453-2022-37-3-12-20.
2. Джарратано Д., Райли Г. Экспертные системы: принципы разработки и программирование: 4-е изд. М.: Вильямс, 2007. 1152 с.
3. Джоши П. Искусственный интеллект с примерами на Python. СПб.: Диалектика, 2019. 448 с.
4. Каракозов С. Д., Самылкина Н. Н., Салахова А. А., Самохвалова Е. А. Вариативное обучение основам искусственного интеллекта в общем образовании на основе интегративного подхода: монография. М.: МПГУ, 2024. 350 с.
5. Левченко И. В. Основные подходы к обучению элементам искусственного интеллекта в школьном курсе информатики // Информатика и образование. 2019. № 6 (305). С. 7–15. EDN: UC1HSM. DOI: 10.32517/0234-0453-2019-34-6-7-15.
6. Примерная основная образовательная программа основного общего образования. Одобрена решением федерального учебно-методического объединения по общему образованию, протокол 6/22 от 15.09.2022 г. https://legalacts.ru/doc/primernaja-osnovnaja-obrazovatel'naja-programma-osnovnogo-obshchego-obrazovanija-odobrena-resheniem_2/
7. Примерная основная образовательная программа среднего общего образования. Одобрена решением федерального учебно-методического объединения по общему образованию, протокол 2/16-з от 28.06.2016 г. <https://legalacts.ru/doc/primernaja-osnovnaja-obrazovatel'naja-programma-srednego-obshchego-obrazovanija-odobrena-resheniem/>
8. Рыжова Н. И., Трубина И. И., Королева Н. Ю., Филимонова Е. В. Искусственный интеллект как актуальный тренд содержания обучения информатике в условиях цифровизации // Преподаватель XXI век. 2022. № 2-1. С. 11–22. EDN: ZGIENM. DOI: 10.31862/2073-9613-2022-2-11-22.
9. Самылкина Н. Н., Калинин И. А. Подготовка школьников к олимпиаде по искусственному интеллекту // Сибирский педагогический журнал. 2024. № 6. С. 31–39. EDN: CGSJTI. DOI: 10.15293/1813-4718.2406.03.
10. Фридл Дж. Регулярные выражения. СПб.: Символ-Плюс, 2008. 608 с.
11. Ясницкий Л. Н. Введение в искусственный интеллект. М.: Академия, 2008. 176 с.
12. Agrawal R., Srikant R. Fast algorithms for mining association rules in large databases // Proc. of the 20th Int. Conf. on Very Large Data Bases (VLDB). Santiago, Chile. 1994. P. 487–499. <https://vldb.org/conf/1994/P487.PDF>
13. Stoyanov S., Glushkova T., Papancheva R. Artificial Intelligence. Problem solving through search. Bourgas, Logic Programming LLC Art Publishing House, 2019. 312 p.
14. Stoyanov S., Glushkova T., Papancheva R. Source intellect. Knowledge representation through logic. Bourgas, Logic Programming LLC Art Publishing House, 2021. 248 p.