Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4
по дисциплине ««Основы систем ИИ»

Тема Компьютерное зрение

Студент Батуев А.Г.

Группа ИУ7-36Б

Преподаватели Строганов Ю.В.

Москва, 2025

Содержание

ВВЕДЕНИЕ **4**

1 Аналитическая часть **5**

1.1 Определение объектов на изображении 5

1.2 Технологии компьютерного зрения 5

2 Конструкторская часть **7**

2.1 Обнаружение ромба 7

2.2 Обнаружение лошади 7

3 Технологическая часть **8**

3.1 Средства написания программы 8

3.2 Функции 8

4 Исследовательская часть **11**

4.1 Оборудование 11

4.2 Обнаруженные ромбы 11

4.3 Обнаруженные лошади 19

4.4 Вывод 19

ЗАКЛЮЧЕНИЕ **20**

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ **21**

ВВЕДЕНИЕ

Цель: с использованием библиотеки opencv необходимо реализовать поиск на фотографии лошади и ромба на изображении.

Задачи:

— найти на фотографии лошадь

— найти на фотографии ромб

1 Аналитическая часть

Компьютерное зрение представляет собой одну из ключевых областей искусственного интеллекта, направленную на автоматизацию задач, выполняемых человеческим зрением, таких как распознавание объектов, определение их местоположения, анализ движения и интерпретация сцены. Технологии компьютерного зрения находят широкое применение в различных сферах, включая автономные системы, медицинскую диагностику, промышленные системы контроля и робототехнику. Одной из фундаментальных задач этой области является определение объектов на изображении.

1.1 Определение объектов на изображении

Определение объекта на изображении в общем случае включает несколько этапов. Сначала выполняется предобработка данных, включающая устранение шумов, нормализацию изображения и его масштабирование для дальнейшего анализа. Затем происходит извлечение признаков, которое может быть основано на классических методах анализа изображения, таких как выделение краев, или на применении нейронных сетей, которые автоматически выделяют характерные черты объектов. Следующим этапом является классификация, где модель определяет принадлежность объекта к заданному классу, а также локализация, направленная на определение координат объекта в пространстве изображения. Для устранения ложных срабатываний и улучшения точности результатов на завершающем этапе применяется постобработка.

1.2 Технологии компьютерного зрения

OpenCV (Open Source Computer Vision Library) является широко применяемым инструментом для работы с изображениями и видео. Она предоставляет обширный набор функций для выполнения таких задач, как обработка изображений, анализ движения и распознавание объектов. Библиотека поддерживает интеграцию с современными технологиями машинного обучения, включая TensorFlow и PyTorch, что делает её подходящим выбором для реализации систем компьютерного зрения.

YOLO (You Only Look Once) представляет собой семейство моделей глубокого обучения, которые позволяют выполнять детектирование и классификацию объектов за один проход через нейронную сеть. Это достигается за счёт разделения изображения на сетку и одновременного предсказания классов и координат объектов. Высокая скорость работы делает YOLO особенно эффективным для задач реального времени.

Алгоритм NMS (Non-Maximum Suppression) используется для удаления избыточных ограничивающих прямоугольников, предсказанных для одного и того же объекта. NMS основывается на вычислении метрики IoU (Intersection over Union), которая определяет степень перекрытия прямоугольников. Прямоугольники с высокой степенью перекрытия, кроме одного с наивысшей вероятностью, удаляются, что позволяет значительно улучшить точность детекти-

рования.

Вывод

В ходе аналитического исследования были рассмотрены ключевые аспекты компьютерного зрения, включая процессы детектирования объектов, а также основные технологии, такие как библиотека OpenCV, алгоритмы YOLO и NMS. Данные инструменты предоставляют широкие возможности для реализации задач автоматического анализа изображений, включая поиск заданных объектов.

2 Конструкторская часть

Программа состоит из двух основных частей: обнаружение ромба и обнаружение лошади. Для каждого объекта используется индивидуальный подход, основанный на особенностях их визуальных признаков.

2.1 Обнаружение ромба

Алгоритм поиска ромбов в изображении основывается на анализе геометрических контуров. На первом этапе изображение преобразуется в оттенки серого для упрощения обработки. Затем применяется адаптивная бинаризация с использованием метода Гаусса, что позволяет выделить основные контуры объектов. Полученные контуры анализируются на предмет их формы. Для этого используется аппроксимация полигонов, определяющая количество углов у замкнутого контура.

Если контур имеет четыре угла и является выпуклым, выполняется дальнейшая проверка. Рассчитываются длины сторон и углы между ними, чтобы определить, соответствует ли фигура характеристикам ромба. Дополнительно проверяются диагонали фигуры на их соотношение длины, что позволяет отфильтровать ложные срабатывания.

2.2 Обнаружение лошади

Для обнаружения лошади используется модель глубокого обучения YOLO (You Only Look Once). Программа загружает предварительно обученные веса и конфигурацию модели, после чего подготавливает входное изображение для обработки.

Модель предсказывает координаты объектов на изображении и их классы. Из всех предсказанных объектов отбираются только те, которые соответствуют классу *horse*. Для каждой найденной области вычисляется вероятность принадлежности к данному классу. На этапе постобработки применяется алгоритм подавления перекрывающихся прямоугольников (NMS), который удаляет избыточные рамки и оставляет только наиболее вероятные детекции.

Вывод

Разработанный алгоритм демонстрирует комплексный подход к решению задачи детектирования объектов на изображении. Использование классических методов обработки изображения для поиска ромбов и современных моделей глубокого обучения для обнаружения лошадей обеспечивает высокую точность и надёжность работы программы.

3 Технологическая часть

3.1 Средства написания программы

Для реализации программного обеспечения были использованы следующие средства:

— среда разработки PyCharm 2023 community edition [4]

— язык разработки Python 3.10 [5]

Используемые библиотеки:

— pathlib — для доступа к файлам системы и создания папок/файлов [2]

— numpy — для быстрой работы с массивами, сохранения промежуточных данных и связи их с другими библиотеками [1]

— cv2 — для работы с изображениями [3]

3.2 Функции

Листинг 3.1 — Определение ромба

```
def detect_diamonds(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    thresh = cv2.adaptiveThreshold(gray, 255, cv2.
        ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, 11, 2)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.
        CHAIN_APPROX_SIMPLE)
    diamonds_found = False
    for contour in contours:
        epsilon = 0.04 * cv2.arcLength(contour, True)
        approx = cv2.approxPolyDP(contour, epsilon, True)
        if len(approx) == 4 and cv2.isContourConvex(approx):
            sides = [np.linalg.norm(approx[i][0] - approx[(i+1) %
                4][0]) for i in range(4)]
            side_ratios = [sides[i] / sides[(i+1) % 4] for i in range
                (4)]
            if all(0.6 <= ratio <= 1.4 for ratio in side_ratios):
                angles = []
                for i in range(4):
                    vec1 = approx[i][0] - approx[(i-1) % 4][0]
                    vec2 = approx[i][0] - approx[(i+1) % 4][0]
                    cos_angle = np.dot(vec1, vec2) / (np.linalg.norm(
                        vec1) * np.linalg.norm(vec2))
                    angle = np.arccos(np.clip(cos_angle, -1.0, 1.0)) *
                        (180.0 / np.pi)
                    angles.append(angle)
```

```
                    if all(angle > 25 and angle < 165 for angle in angles)
                        :
                        acute_angles = sum(1 for angle in angles if 45 <=
                            angle <= 85)
                        obtuse_angles = sum(1 for angle in angles if 95 <=
                            angle <= 135)
                        if acute_angles == 2 and obtuse_angles == 2:
                            diag1_sq = np.sum((approx[0][0] - approx
                                [2][0])**2)
                            diag2_sq = np.sum((approx[1][0] - approx
                                [3][0])**2)
                            if not 0.9 <= diag1_sq / diag2_sq <= 1.2:
                                x, y, w, h = cv2.boundingRect(approx)
                                cv2.rectangle(image, (x, y), (x + w, y + h
                                    ), (255, 0, 0), 2)
                                diamonds_found = True
    if diamonds_found:
        return image
    else:
        return None
```

Листинг 3.2 — Определение лошади

```
def detect_horse(image, model_path, horse_class_id):
    net = cv2.dnn.readNet(model_path, model_path.replace(".weights", "
        .cfg"))
    blob = cv2.dnn.blobFromImage(image, scalefactor=1/255.0, size
        =(416, 416), swapRB=True, crop=False)
    net.setInput(blob)
    layer_names = net.getUnconnectedOutLayersNames()
    outputs = net.forward(layer_names)
    h, w = image.shape[:2]
    boxes = []
    confidences = []
    for output in outputs:
        for detection in output:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if class_id == horse_class_id and confidence > 0.2:
                center_x, center_y, width, height = detection[:4] * np
                    .array([w, h, w, h])
                x = int(center_x  width / 2)
```

```
                y = int(center_y  height / 2)
                boxes.append([x, y, int(width), int(height)])
                confidences.append(confidence)
    if len(boxes) == 0:
        return None
    indices = cv2.dnn.NMSBoxes(boxes, confidences, score_threshold
        =0.3, nms_threshold=0.7)
    if len(indices) > 0:
        for i in indices.flatten():
            box = boxes[i]
            x, y, w, h = box
            cv2.rectangle(image, (x, y), (x + w, y + h), (24, 143, 0),
                2)
            cv2.putText(image, f'{class_names[horse_class_id]}: {
                confidences[i] * 100:.2f}%',
                        (x + 10, y - 10), cv2.FONT_HERSHEY_DUPLEX, 1,
                            (0, 255, 0), 1)
    return image
```

Листинг 3.3 — Обработка изображения

```
def proceed_image(image_path, model_path):
    horse_class_id = class_names.index("horse")
    image = cv2.imread(image_path)
    image_copy = image.copy()
    diamond = detect_diamonds(image)
    image_with_boxes = detect_horse(image_copy, model_path,
        horse_class_id)
    if image_with_boxes is not None:
        output = RES_DIR / "horses" / image_path.name
        output.parent.mkdir(parents=True, exist_ok=True)
        cv2.imwrite(str(output), image_with_boxes)
    if diamond is not None:
        output = RES_DIR / "diamonds" / image_path.name
        output.parent.mkdir(parents=True, exist_ok=True)
        cv2.imwrite(str(output), diamond)
```

Вывод

Перечисленные технологии и функции позволяют реализовать программу для обнаружения ромбов и лошадей в изображениях.

4 Исследовательская часть

Цель исследования — обнаружить на фотографии лошадей и ромбы.

4.1 Оборудование

Характеритстики ноутбука:

— процессор intel-core i5-12500H [7]

— ОЗУ 16 Гб DDR4

— ОС Windows 11 [6]

4.2 Обнаруженные ромбы

Рисунок 4.1 — Изображение 1

Рисунок 4.2 — Изображение 5

Рисунок 4.3 — Изображение 10

Рисунок 4.4 — Изображение 11

Рисунок 4.5 — Изображение 12

Рисунок 4.6 — Изображение 13

Рисунок 4.7 — Изображение 15

Рисунок 4.8 — Изображение 16

Рисунок 4.9 — Изображение 17

Рисунок 4.10 — Изображение 19

4.3 Обнаруженные лошади

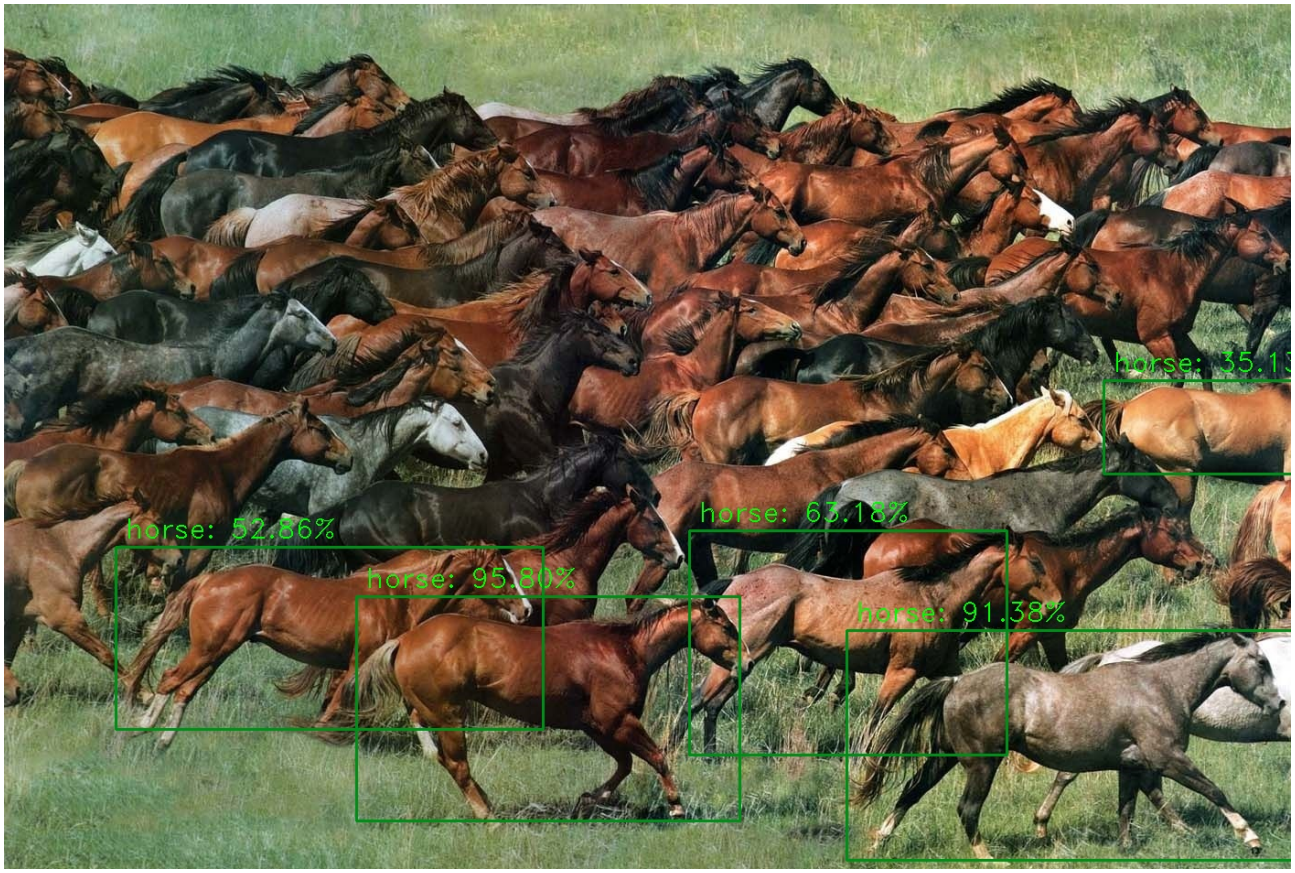

Рисунок 4.11 — Изображение 1

4.4 Вывод

Полученные изображения в общем случае показывают состоятельность данных метод, хотя и с далеко не идеальными результатами, часть из которых можно объяснить тем, что, например, для поиска лошадей используется лишь часть изображения подходящего для cv2 формата (в данном случае 416x416), что не дает полной картины.

Что касается ромбов, то результаты далеки от идеальных, многие объекты были ошибочно обнаружены, а в некоторых случаях очевидные ромбы (с точки зрения человека) не были обнаружены, хотя правильные результататы, в некоторых случаях, были получены.

ЗАКЛЮЧЕНИЕ

Разработанный алгоритм продемонстрировал возможность решения задачи автоматического поиска заданных объектов на изображениях с использованием библиотеки OpenCV. Несмотря на общее подтверждение состоятельности предложенного метода, проведённый анализ результатов выявил ряд ограничений и проблем, которые необходимо учитывать.

Результаты детектирования лошадей показывают, что использование моделей глубокого обучения, таких как YOLO, позволяет эффективно находить объекты данного класса. Однако ограничение входного изображения размером 416x416 пикселей, накладываемое архитектурой сети, приводит к потере некоторых деталей сцены, что может снижать качество обнаружения.

С точки зрения детектирования ромбов, применение классических методов обработки изображения дало неоднозначные результаты. Хотя часть объектов была успешно обнаружена, алгоритм продемонстрировал чувствительность к помехам и вариациям формы объектов. В ряде случаев наблюдались ложные срабатывания, а также пропуски очевидных ромбов, что может быть связано с особенностями реализации метода аппроксимации контуров и выбора пороговых значений для проверки геометрических характеристик.

Таким образом, для повышения точности и надёжности алгоритма требуется провести дополнительные исследования. В частности, возможным направлением улучшения является использование более мощных архитектур глубокого обучения для обработки ромбов или гибридного подхода, комбинирующего классические методы с нейронными сетями. Исследовать масштабирование изображений на производительность алгоритма, а также оптимизировать пороговые параметры для уменьшения числа ложных срабатываний.

Подводя итог, предложенный метод обладает потенциалом для решения задач компьютерного зрения, однако требует дальнейшей доработки для достижения более высоких показателей точности и универсальности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. NumPy Documentation URL: https://numpy.org/doc/ (Дата обращения 22.01.2025)
2. Документация Pathlib https://docs.python.org/3/library/pathlib.html (Дата обращения 22.01.2025)
3. OpenCV Documentation https://docs.opencv.org/4.x/ (Дата обращения 22.01.2025)
4. Среда разработки Pycharm 2023 Community edition URL: https://www.jetbrains.com/ru-ru/pycharm/download/other.html
5. Язык разработки Python 3.10 URL: https://www.python.org/downloads/release/python-3100/
6. ОС Windows 11 URL: https://www.microsoft.com/ru-ru/software-download/windows11
7. Intel i5-12500H URL: https://www.intel.com/content/www/us/en/products/sku/96141/intel-core-i512500h-processor-18m-cache-up-to-4-50-ghz/specifications.html?wapkw=12500h