

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ»

(ФГБОУ ВО «ВГТУ», ВГТУ)

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И КОМПЬЮТЕРНОЙ  
БЕЗОПАСНОСТИ

КАФЕДРА КОМПЬЮТЕРНЫХ ИНТЕЛЛЕКТУАЛЬНЫХ ТЕХНОЛОГИЙ  
ПРОЕКТИРОВАНИЯ

КУРСОВОЙ ПРОЕКТ

по дисциплине «Машинное обучение и анализ данных»

Тема: «Определение нетиповых объектов по видеоряду»

Разработал студент

гр. МИИВТ -231

В.В. Котельников

подпись, дата

инициалы, фамилия

Руководитель

В.В. Ветохин

подпись, дата

инициалы, фамилия

Нормоконтролер

В.В. Ветохин

подпись, дата

инициалы, фамилия

Защищен

Оценка

дата

Воронеж 2024

## СОДЕРЖАНИЕ

СОДЕРЖАНИЕ .....	2
Введение.....	3
1. Обзор методов определения нетиповых объектов.....	4
2. Создание окружения и настройка библиотек .....	5
3. Создание программы по обнаружению объектов .....	10

## Введение

В современном мире анализ видеоданных играет ключевую роль в самых разных областях, от безопасности и наблюдения до автоматизации и улучшения пользовательского опыта. Одной из важнейших задач в этой области является определение и классификация объектов, которые отличаются от типичных или ожидаемых. Эти "нетиповые объекты" могут включать в себя все, что не соответствует обычному паттерну поведения или внешнего вида, например, необычные события или аномалии в поведении людей, транспортных средств и других элементов видеоряда.

Целью данной курсовой работы является разработка методов и алгоритмов, которые позволят эффективно идентифицировать такие объекты в автоматическом режиме. Для достижения этой цели будут исследованы существующие подходы к анализу видеоданных, включая машинное обучение и компьютерное зрение, а также разработаны новые методы, способные улучшить точность и скорость обработки данных.

## 1. Обзор методов определения нетиповых объектов.

### Традиционные методы:

- статистический анализ: использование статистических моделей для выявления аномалий в данных;
- пороговые методы: определение нетиповых объектов на основе заранее установленных пороговых значений;
- кластеризация: группировка данных с целью выявления объектов, которые не вписываются в общие кластеры;

### Методы машинного обучения:

- обучение без учителя: алгоритмы, такие как Isolation Forest или One-Class SVM, которые обучаются на "нормальных" данных и выявляют аномалии;
- обучение с подкреплением: использование наград и штрафов для обучения модели распознаванию нетиповых объектов;
- глубокое обучение: применение нейронных сетей, таких как свёрточные нейронные сети (CNN) для анализа видеоданных и выявления аномалий.

### Гибридные и инновационные подходы:

- совмещение методов: интеграция различных подходов для повышения точности определения;
- адаптивные системы: системы, способные самообучаться и адаптироваться к изменяющимся условиям в данных;
- интерактивное обучение: включение человеческого эксперта в процесс обучения модели для улучшения результатов.

## 2. Создание окружения и настройка библиотек

Для начала создаю папку (folder), где буду разворачивать рабочее окружение. Я создам папку Training (обучение) в которой также создам папку с именем своей учётной записи (2wK), в которой и буду разворачивать своё окружение. Для создания виртуального окружения нужно запустить команду строку, перейти в нужный каталог и ввести команды:

```
python -m venv my_virtual_area
```

После того как виртуальное окружение создано, можно устанавливать в него различные библиотеки, датасеты, клонировать репозитории и т.д. Чтобы установить нужные библиотеки нужно активировать виртуальное окружение при помощи команды:

```
.\my_virtual_area\Scripts\activate
```

После активации виртуального окружения устанавливаю библиотеки, которые мне понадобятся, а также классификатор. Библиотеки которые я использую:

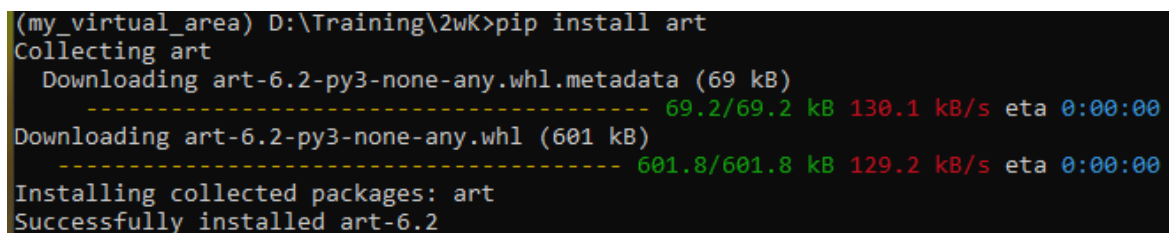
1) OpenCV (Open Source Computer Vision Library) — это библиотека программного обеспечения с открытым исходным кодом, предназначенная для компьютерного зрения и машинного обучения. OpenCV была создана для обеспечения общедоступной инфраструктуры для приложений компьютерного зрения и для ускорения использования восприятия машинами в коммерческих продуктах. Будучи библиотекой BSD-лицензированной, OpenCV позволяет использовать её код для коммерческих и исследовательских целей.

2) NumPy — это фундаментальная библиотека для научных вычислений в Python. Она предоставляет поддержку для создания и работы с

большими, многомерными массивами и матрицами, а также набор широко используемых математических функций для операций с этими массивами.

3) Библиотека ART (ASCII Art Library) — это инструмент Python для преобразования текста в ASCII-арт, то есть искусство создания картинок с помощью символов. Она позволяет пользователям легко добавлять стилизованный текстовый контент в консольные приложения или выводить его в текстовых файлах.

Установка данных библиотек производится при помощи команды `pip` (рисунок 1).



```
(my_virtual_area) D:\Training\2wK>pip install art
Collecting art
  Downloading art-6.2-py3-none-any.whl.metadata (69 kB)
----- 69.2/69.2 kB 130.1 kB/s eta 0:00:00
  Downloading art-6.2-py3-none-any.whl (601 kB)
----- 601.8/601.8 kB 129.2 kB/s eta 0:00:00
Installing collected packages: art
Successfully installed art-6.2
```

Рисунок 1 – Установка библиотек при помощи команды `pip`

Проверю что библиотеки установлены и работают. Для проверки библиотеки `opencv-python` напишу программу для загрузки изображения, его преобразования в оттенки серого и вывод этих изображений:

```
import cv2 as cv
```

```
#Загрузка изображения
```

```
image = cv.imread("test_image.jpg")
```

```
# Проверка, успешно ли было загружено изображение
```

```
if image is None:
```

```
    print("Не удалось загрузить изображение")
```

```
else:
```

# Преобразование изображения в оттенки серого

```
gray_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
```

# Отображение оригинального изображения

```
cv.imshow('Оригинальное изображение', image)
```

# Отображение изображения в оттенках серого

```
cv.imshow('Изображение в оттенках серого', gray_image)
```

# Ожидание нажатия клавиши перед закрытием окон

```
cv.waitKey(0)
```

```
cv.destroyAllWindows()
```

Вывод приложения показан на рисунках 2-3.



Рисунок 2 – Отображение оригинального изображения





Рисунок 3 – Отображение преобразованного изображения

Проверю работу библиотеки numpy:

```
# Создание двумерного массива
```

```
array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
```

```
# Вывод оригинального массива
```

```
print("Оригинальный массив:")
```

```
print(array)
```

```
# Транспонирование массива
```

```
transposed_array = array.T
```

```
print("\nТранспонированный массив:")
```

```
print(transposed_array)
```

```
# Вычисление обратной матрицы
```



```

# Проверка, является ли матрица квадратной и определенной
if array.shape[0] == array.shape[1] and np.linalg.det(array) != 0:
    inverse_array = np.linalg.inv(array)
    print("\nОбратная матрица:")
    print(inverse_array)
else:
    print("\nМатрица не квадратная или определитель равен нулю,
    обратную матрицу найти невозможно.")

# Вычисление собственных значений и собственных векторов
eigenvalues, eigenvectors = np.linalg.eig(array)
print("\nСобственные значения:")
print(eigenvalues)
print("Собственные векторы:")
print(eigenvectors)

```

Вывод программы показан на рисунке 4.

```

Оригинальный массив:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
\nТранспонированный массив:
[[1 4 7]
 [2 5 8]
 [3 6 9]]
\nМатрица не квадратная или определитель равен нулю, обратную матрицу найти невозможно.
\nСобственные значения:
[ 1.61168440e+01 -1.11684397e+00 -1.30367773e-15]
Собственные векторы:
[[-0.23197069 -0.78583024  0.40824829]
 [-0.52532209 -0.08675134 -0.81649658]
 [-0.8186735  0.61232756  0.40824829]]

```

Рисунок 4 – Проверка работы библиотеки numpy

Проверка работы модуля tprint библиотеки art:

```
tprint("Hello, world!", font="block")
```

Вывод представлен на рисунке 5.

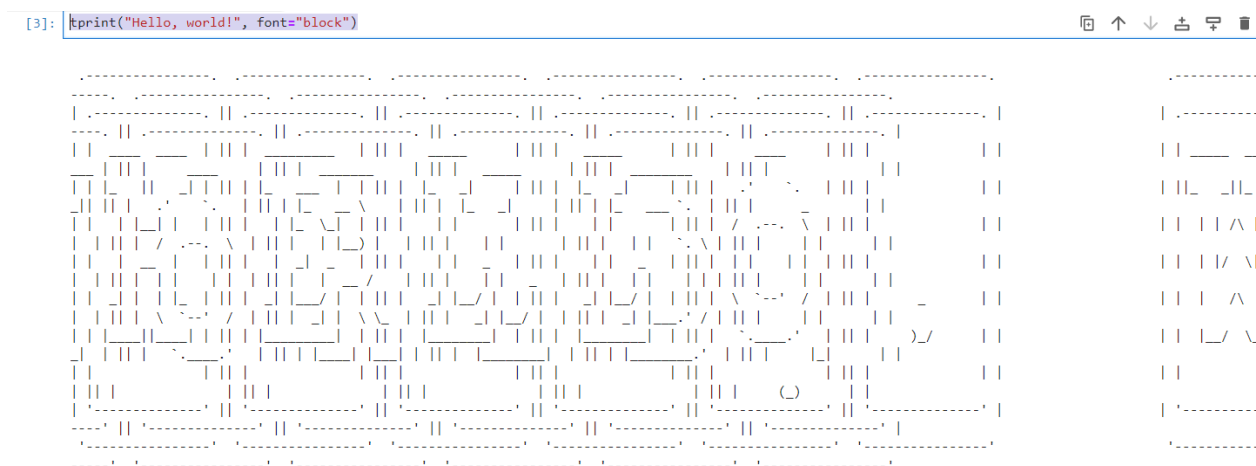


Рисунок 5 – Проверка работы модуля tprint

### 3. Создание программы по обнаружению объектов

Напишу функцию для применения YOLO. С её помощью будут определяться классы объектов на изображении, а также координаты их границ:

```
def apply_yolo_object_detection(image_to_process):
```

```
    """
```

Recognition and determination of the coordinates of objects on the image

```
:param image_to_process: original image
```

```
:return: image with marked objects and captions to them
```

```
    """
```

```
    height, width, _ = image_to_process.shape
```

```
blob = cv2.dnn.blobFromImage(image_to_process, 1 / 255, (608,  
608),
```

```
(0, 0, 0), swapRB=True, crop=False)
```

```
net.setInput(blob)
```

```
outs = net.forward(out_layers)
```

```
class_indexes, class_scores, boxes = ([[] for i in range(3)])
```

```
objects_count = 0
```

```
# Starting a search for objects in an image
```

```
for out in outs:
```

```
    for obj in out:
```

```
        scores = obj[5:]
```

```
        class_index = np.argmax(scores)
```

```
        class_score = scores[class_index]
```

```
        if class_score > 0:
```

```
            center_x = int(obj[0] * width)
```

```
            center_y = int(obj[1] * height)
```

```
            obj_width = int(obj[2] * width)
```

```
            obj_height = int(obj[3] * height)
```

```
            box = [center_x - obj_width // 2, center_y - obj_height // 2,
```

```
                  obj_width, obj_height]
```

```
            boxes.append(box)
```

```
            class_indexes.append(class_index)
```

```
            class_scores.append(float(class_score))
```

```
# Selection
```

```
chosen_boxes = cv2.dnn.NMSBoxes(boxes, class_scores, 0.0, 0.4)
```

```
for box_index in chosen_boxes:
```

```
    box_index = box_index
```

```
    box = boxes[box_index]
```

```

class_index = class_indexes[box_index]

# For debugging, we draw objects included in the desired classes
if classes[class_index] in classes_to_look_for:
    objects_count += 1
    image_to_process =
draw_object_bounding_box(image_to_process,
                           class_index, box)

    final_image = draw_object_count(image_to_process,
objects_count)
return final_image

```

Также добавлю функцию, которая будет обводить найденные на изображении объекты с помощью координат границ:

```

def draw_object_bounding_box(image_to_process, index, box):
    """
    Drawing object borders with captions
    :param image_to_process: original image
    :param index: index of object class defined with YOLO
    :param box: coordinates of the area around the object
    :return: image with marked objects
    """

    x, y, w, h = box
    start = (x, y)
    end = (x + w, y + h)
    color = (0, 255, 0)
    width = 2

```

```
final_image = cv2.rectangle(image_to_process, start, end, color,  
width)
```

```
start = (x, y - 10)
```

```
font_size = 1
```

```
font = cv2.FONT_HERSHEY_SIMPLEX
```

```
width = 2
```

```
text = classes[index]
```

```
final_image = cv.putText(final_image, text, start, font,  
font_size, color, width, cv2.LINE_AA)
```

```
return final_image
```

Также будет хорошо добавить вывод количества объектов:

```
def draw_object_count(image_to_process, objects_count):
```

```
    """
```

```
    Signature of the number of found objects in the image
```

```
:param image_to_process: original image
```

```
:param objects_count: the number of objects of the desired class
```

```
:return: image with labeled number of found objects
```

```
    """
```

```
start = (10, 120)
```

```
font_size = 1.5
```

```
font = cv2.FONT_HERSHEY_SIMPLEX
```

```
width = 3
```

```
text = "Objects found: " + str(objects_count)
```

```
# Text output with a stroke
```

```

        # (so that it can be seen in different lighting conditions of the
picture)
        white_color = (255, 255, 255)
        black_outline_color = (0, 0, 0)
        final_image = cv.putText(image_to_process, text, start, font,
font_size,
                                black_outline_color, width * 3, cv2.LINE_AA)
        final_image = cv.putText(final_image, text, start, font, font_size,
                                white_color, width, cv.LINE_AA)

    return final_image

```

Также нам нужна будет функция, которая обрабатывает видео по кадрам и выводит результат обработки на экран:

```

def draw_object_count(image_to_process, objects_count):
    """
    Signature of the number of found objects in the image
    :param image_to_process: original image
    :param objects_count: the number of objects of the desired class
    :return: image with labeled number of found objects
    """

    start = (10, 120)
    font_size = 1.5
    font = cv2.FONT_HERSHEY_SIMPLEX
    width = 3
    text = "Objects found: " + str(objects_count)

    # Text output with a stroke

```

```

        # (so that it can be seen in different lighting conditions of the
picture)
        white_color = (255, 255, 255)
        black_outline_color = (0, 0, 0)
        final_image = cv.putText(image_to_process, text, start, font,
font_size,
                                black_outline_color, width * 3, cv2.LINE_AA)
        final_image = cv.putText(final_image, text, start, font, font_size,
                                white_color, width, cv.LINE_AA)

    return final_image

```

Для того, чтобы не нагружать моё устройство обработкой каждого кадра, добавлю функцию обновления экрана при нажатии любой клавиши:

```

def start_video_object_detection(video: str):
    """
    Захват и анализ видео в режиме реального времени
    """

    while True:
        try:
            # Capturing a picture from a video
            video_camera_capture = cv.VideoCapture(video)

            while video_camera_capture.isOpened():
                ret, frame = video_camera_capture.read()
                if not ret:
                    break

            # Application of object recognition methods on a video
            frame from YOLO

```



```

        frame = apply_yolo_object_detection(frame)

        # Displaying the processed image on the screen with a
        reduced window size

        frame = cv.resize(frame, (1920 // 2, 1080 // 2))
        cv.imshow("Video Capture", frame)
        cv.waitKey(1)

    video_camera_capture.release()
    cv.destroyAllWindows()

except KeyboardInterrupt:
    pass

```

Ну и конечно же нужно написать функцию main, в которой настрою саму сеть:

```

if __name__ == '__main__':

    # Logo
    tprint("Object detection")
    tprint("by")
    tprint("2wK")

    # Loading YOLO scales from files and setting up the network
    net = cv.dnn.readNetFromDarknet("Resource/yolov4-tiny.cfg",
                                    "Resource/yolov4-tiny.weights")

    layer_names = net.getLayerNames()
    out_layers_indexes = net.getUnconnectedOutLayers()
    out_layers = [layer_names[index - 1] for index in
out_layers_indexes]

```

```
# Loading from a file of object classes that YOLO can detect  
with open("Resource/coco.names.txt") as file:
```

```
    classes = file.read().split("\n")
```

```
# Determining classes that will be prioritized for search in an image
```

```
# The names are in the file coco.names.txt
```

```
video = input("Path to video (or URL): ")
```

```
look_for = input("What we are looking for: ").split(',')
```

```
# Delete spaces
```

```
list_look_for = []
```

```
for look in look_for:
```

```
    list_look_for.append(look.strip())
```

```
classes_to_look_for = list_look_for
```

```
start_video_object_detection(video)
```