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](#_Toc170139589)

[Введение 3](#_Toc170139590)

[1. Обзор методов определения нетиповых объектов. 4](#_Toc170139591)

[2. Создание окружения и настройка библиотек 5](#_Toc170139592)

[3. Создание программы по обнаружению объектов 10](#_Toc170139593)

# 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.

# 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.

# 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 командую 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 установлю 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).

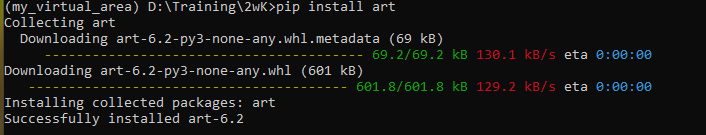


Рисунок 1 – Установка библиотек при помощи команды pip

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.

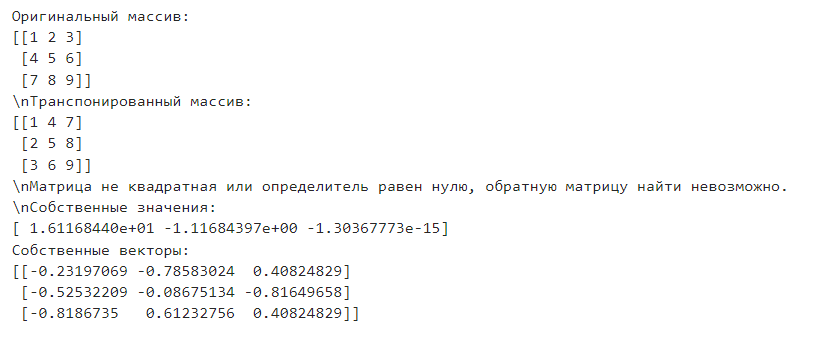


Рисунок 4 – Проверка работы библиотеки numpy

Проверка работы модуля tprint библиотеки art:

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

Вывод представлен на рисунке 5.

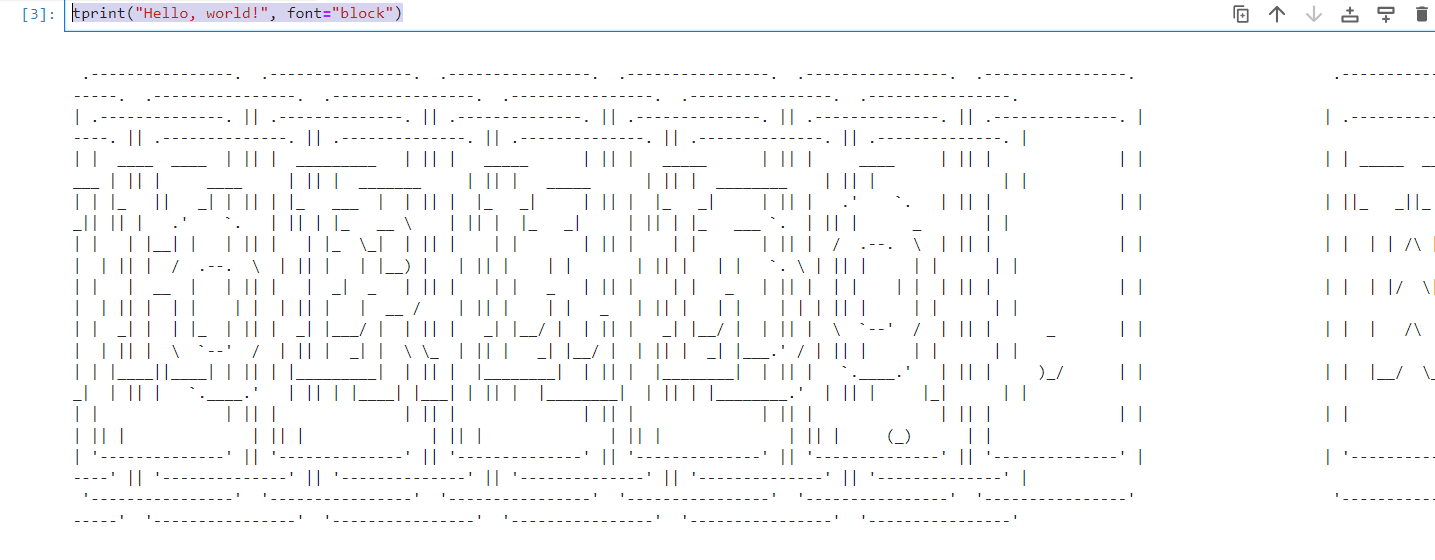


Рисунок 5 – Проверка работы модуля tprint

# 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)