**«Разработка алгоритма на основе нейронной сети yolov8 для определения и аннотирования людей на видео для торговых центров к середине 2025 г.»**

Ученик 9-3 класса  
 МБОУ СОШ №17

Семенов Александр Алексеевич

г. Тверь, 2025

**Оглавление**

[Введение 2](#_heading=h.6ax4oyd325x9)

[Обзор рынка компьютерного зрения 4](#_heading=h.f64o12v3r7oy)

[Разработать алгоритм в виде блок-схемы работы программного кода 6](#_heading=h.218k3wn3r55i)

[Интерпретация алгоритма на основе нейронной сети yolov8 8](#_heading=h.jyzsi14rmqn2)

[Разработка пользовательского интерфейса 11](#_heading=h.acr9lpm72qmt)

[Произвести тестирование алгоритма в реальных условиях 12](#_heading=h.blvrybjxr5vr)

[Заключение 13](#_heading=h.6n6ledxdmfj)

[Список литературы 14](#_heading=h.kmpx3eqyppi)

# Введение

Нейронные сети и ИИ на данный момент в тренде, поэтому в рамках работы проекта была произведена разработка алгоритма на основе нейронной сети yolo v8.Yolo(You Once Look One) представляет собой нейросеть для определения различных объектов.V8 была выбрана за счет ее изученности и легкости в применении.

*Цель*: разработать алгоритм на основе нейронной сети yolo v8 для определения и аннотирования людей на видео для торговых центров к середине 2025 г.

*Задачи*:

* обзор рынка компьютерного зрения;
* разработать алгоритм в виде блок-схемы работы программного кода;
* интерпретация алгоритма на основе нейронной сети yolov8;
* разработка пользовательского интерфейса;
* произвести тестирование алгоритма в реальных условиях.

*Проблема*: увеличение прибыли за счет снижения экономических издержек;

*Актуальность:* обеспечение безопасности, автоматизация, снижение экономических издержек. В своем обращении указанную актуальность подчеркивал Президент российской федерации В.В. Путин: «Наличие собственных разработок нового поколения искусственного интеллекта – одно из ключевых условий научного, технологического и, что важно, мировоззренческого суверенитета нашей страны».

*Гипотеза*: алгоритм для определения и аннотирования людей на видео поможет увеличить прибыль за счет снижения экономических издержек.

Анализ области исследования: алгоритм будет реализован в области использования компьютерного зрения для различных потребителей для автоматизации рутинных процессов.

*Конечный продукт*: алгоритм для определения и аннотирования людей на видео.

*Целевая аудитория*:

* торговые центры;
* школы;
* частные охранные предприятия;
* МЧС.

*Предмет*: инструменты автоматизации рутинных процессов.

*Объект*: алгоритм компьютерного зрения.

*Методы*:

* статистический - совокупность приёмов, которые позволяют собирать, обобщать, представлять, анализировать и данные. В моей работе была проведена статистика о правильной работе алгоритма.
* экспериментальный - метод исследования, предполагающий цикличное наблюдения явления в определенных условиях и проведения опыта с целью проверки положений, выдвинутых в гипотезе.Были проведены проверки работы алгоритма в различных условиях;
* тестирование - процесс сбора и анализа субъективной информации от респондентов, целевой аудитории.Был выполнен тест о корректной работе алгоритма.

*Временные рамки:* планируется реализация в рамках 2025 г.

*Географические рамки:* г. Тверь,Тверская область.

Таблица 1. *Дорожная карта*

|  | 4.08.24-24.11.24 | 25.11.24 - 6.12.24 | 7.12.24 - 4.01.24 | 5.01.25 - 26.05.25 | 5.01.25 -  26.05.25 |
| --- | --- | --- | --- | --- | --- |
| 1 Этап. Подготовительный |  |  |  |  |  |
| 2 Этап.Разработка алгоритма в виде блок схемы |  |  |  |  |  |
| 3 Этап. [Интерпретация алгоритма](#_heading=h.jyzsi14rmqn2) |  |  |  |  |  |
| 4 Этап. Разработка пользовательского интерфейса |  |  |  |  |  |
| 5 Этап.Проведение тестирования |  |  |  |  |  |

*Бюджет*: стоимость специалиста текущего уровня за час равна 250 рублей,над этим проектом работали около 100 часов -это выходит 25000 руб. Далее покупка веб-камеры - 3000. Оптимальный компьютер для работы алгоритма 120000 р. В итоге издержки составили 148000 руб.

Минимальная комплектация включает в себя работу на 2-3 камерах,без подсчета людей,только триггерная линия.Стоимость тарифа - 10000 рублей

Средняя комплектация включает в себя работу на 10 камерах,с подсчетом людей.Стоимость этого тарифа - 35000 рублей

Полная комплектация включает в себя работу на максимально возможном количестве камер, с отслеживаем путей передвижения людей. Стоимость тарифа - 100000 руб.

Использованные ресурсы : разработка алгоритма велась на компьютере со следующими характеристиками:

* Процессор : Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz;
* Видеокарта : NVIDIA GeForce GTX 1060 6GB;
* Оперативная память 8GB DDR3.

Источники привлечения: был использован личный компьютер без дополнительных вложений

# 

# Обзор рынка компьютерного зрения

Компьютерное зрение - это область ИИ, создающая возможность компьютерам визуально воспринимать информацию с видео или с фото.

По данным на 2024 год объем рынка составлял примерно 26.25 млрд $ и прогнозируемые к 2030 году 50 млрд $, а в 2018 году имел объем в 2.9 млрд $.

Данное направление сейчас находится в тренде т.к. используется во многих различных отраслях.

Такие как:

* автоматизация производства;
* сельское хозяйство;
* медицина;
* безопасность населения.

Ведущие компании в этом направлении:

* Google: Google Cloud Vision API предоставляет инструменты для анализа изображений;
* Microsoft: Azure Cognitive Services включает Computer Vision API для анализа изображений и видео;
* Amazon: Amazon Recognition используется для распознавания объектов, лиц и текста;
* IBM: Watson Visual Recognition предлагает решения для анализа визуальных данных;
* NVIDIA: Разрабатывает GPU и платформы для задач компьютерного зрения (например, NVIDIA Jetson);

Ограничения компьютерного зрения:

* отсутствия законодательства во многих странах,что может привести к правовым искам;
* необходимость огромной базы данных для более точных результатов.Каптча,заполняемая вами при практически каждом заходе на сайт,является не проверка на робота,а материалами для обучения нейросети и создания баз данных.;
* вопрос об этичности использования нейронных сетей для определения лиц и для видеонаблюдения.

Компьютерное зрение является одной из ключевых технологий XXI века, оказывая влияние на множество аспектов нашей жизни. Благодаря прогрессу в области ИИ, появлению новых алгоритмов и снижению стоимости вычислений, рынок CV активно растет. Тем не менее, для дальнейшего прогресса необходимо перечисленные проблемы решить, чтобы обеспечить безопасное и эффективное использование этих технологий.

Ключевые конкурентные преимущества : алгоритм по сравнению с большинством работает при затрудненных условиях,а также на 3-5 % за счет использования модели нейронной сети YOLO.

# Разработать алгоритм в виде блок-схемы работы программного кода

Рис. 1. Блок-схема работы кода

# 

Эта блок-схема детально,пошагово показывает работу алгоритма. Далее я бы хотел объяснить некоторые моменты:

* Ultralytics.check() проверяет правильность установки ultralytics`а;
* HOME является переменной для обозначения директории py файла;
* SOURCE\_VIDEO\_PATH принимает директорию изначального видео;
* Class\_Names\_Dict определяет имеющиеся объекты в выбранной модели;
* SELECTED\_CLASS\_NAMES определяет выбранные объекты а SELECTED\_CLASS\_ID их айди;
* LINE\_START и LINE\_END - это начало и конец триггерной линии;
* TARGET\_VIDEO\_PATH - конечный путь готового видео;
* инициализация ByTeTrack - запуск ByTeTrack и его калибровка;
* сброс ByTeTrack - сброс до начальных настроек;
* получение информации о видео - получение данных по типу: количество кадров,размеры видео т.д.;
* создание генератора кадров из видео - Генератор — это специальный объект в Python, который позволяет лениво (по запросу) выдавать данные, не загружая их все сразу в память;
* инициализация Line\_zone;box\_annotator;label\_annotator;line\_zone\_annotator; - LINE\_ZONE - объявление триггерной линии, box\_annotator - используется для рисования рамок,label\_annotator - создание текстовых меток,line\_zone\_annotator - создает визуальную линию для восприятия человеком ;
* определение функции callback - используется для обработки каждого кадра видео, добавляя визуальные аннотации (например, рамки, метки и линии) на основе результатов работы модели компьютерного зрения.;
* создание меток для объектов - данные об объекте;
* аннотирование кадра с помощью Box\_annotator и Labelannotator;
* возврат аннотированного кадра с Linezoneannotator;
* вызов sv.process\_video() - обрабатывает полное видео;
* вызов opencurfile() - вызывает пользовательский интерфейс.

Таким образом, эта блок-схема отлично и схематично показывает работу алгоритма.

# 

# Интерпретация алгоритма на основе нейронной сети yolov8

Код состоит из 153 строчек,весом в 10Кб, был спрограммирован на языке программирования Python,C. Код был выполнен в среде Pycharm. Далее будет представлен сам код с комментариями.

import os#библиотека “os” позволяет работать с файлами,которые находятся на компьютере

import ultralytics#библиотека “ultralytics” для обнаружения объектов, решает задачи отслеживания объектов, сегментации экземпляров

from tkinter import filedialog#с помощью метода filedialog можно определить директорию файлов

from tkinter import \*# с помощью tkinter я создаю интерфейс

from tkinter import ttk#

from os import startfile# импортирую метод открывания файла

import supervision as sv# инструмент для компьютерного зрения

from ultralytics import YOLO# загружаю YOLO

import numpy as np#

from tkinter.messagebox import showinfo,showerror

def opencurfile():# определяем функцию для вызова интерфейса

root = Tk()# создаем поле

root.title("")#

root.geometry("800x600")# определяем размеры поля

def open\_file():#определяем функцию для открытия файла

filepath = filedialog.askopenfilename()# задаем значение переменной,которое равно директории выбранного для открытия файла

if filepath != "":#если файл существует

startfile(filepath)#запускаем файл

open\_button = ttk.Button(text="Открыть файл", command=open\_file)# создаем кнопку для открытия файла

open\_button1 = ttk.Button(text = "Запустить аннотирование", command=lambda: opredvideo())# создаем кнопку для аннотирования всего видеo

open\_button1.pack(anchor=CENTER)# задаем ее местоположение

open\_button.pack(anchor=S,side = BOTTOM)# задаем ее местоположение

root.mainloop()# для отображения окна и взаимодействия пользователя с ним вызываем метод mainloop()

def proverka(window1,entry1, entry2, entry3, entry4,entry5,path):

#showinfo(title="аннотация", message='Нажмите')

annoturovanie(window1,entry1, entry2, entry3, entry4,entry5,path)

def annoturovanie(window1,entry1, entry2, entry3, entry4,entry5,path):

ultralytics.checks() # проверяем правильную установку “ultralytics”

HOME = os.getcwd() # задаем значение переменной, которое является директорией исполняемого файла .py

SOURCE\_VIDEO\_PATH = path # директория + название

entry5val = entry5.get()

model = YOLO(f'{temp1}') # выбор модели yolo

CLASS\_NAMES\_DICT = model.model.names # записываем имена которые в данной модели присутствуют

SELECTED\_CLASS\_NAMES = ['person'] # выбранные имена

SELECTED\_CLASS\_IDS = [ # определяем их айди

{value: key for key, value in CLASS\_NAMES\_DICT.items()}[class\_name]

for class\_name

in SELECTED\_CLASS\_NAMES

]

TARGET\_VIDEO\_PATH = f'{entry5val}.mp4' # конечное название файла,которое мы получим после обработки

byte\_tracker = sv.ByteTrack( # используем встроенную функцию для отслеживания людей

track\_activation\_threshold=0.25,

lost\_track\_buffer=30,

minimum\_matching\_threshold=0.8,

frame\_rate=30,

minimum\_consecutive\_frames=3)

byte\_tracker.reset() # перезапускаем байтетрек

video\_info = sv.VideoInfo.from\_video\_path(SOURCE\_VIDEO\_PATH) # записываем значение в переменную,которое является информацией о видео(количество кадров,размеры видео и т.д.)

width = video\_info.width

height = video\_info.height

if entry1.get().isdigit() and entry2.get().isdigit() and entry3.get().isdigit() and entry4.get().isdigit() :

entry1val = float(entry1.get())

entry2val = float(entry2.get())

entry3val = float(entry3.get())

entry4val = float(entry4.get())

else:

showerror(title='Ошибка',message="Указаны неверные значения для линии,значения выбраны по умолчанию")

entry1val = width \* 0.5

entry2val = 0

entry3val = width \* 0.5

entry4val = height

LINE\_START = sv.Point(entry1val, entry2val) # задаем начальные координаты линии

LINE\_END = sv.Point(entry3val, entry4val) # задаем конечные координаты линии

window1.destroy()

#generator = sv.get\_video\_frames\_generator(SOURCE\_VIDEO\_PATH) # создаем генератор для итерации видео

line\_zone = sv.LineZone(start=LINE\_END, end=LINE\_START) # используем метод для создания определяющей линии

box\_annotator = sv.BoxAnnotator(thickness=4) # рисование рамок

label\_annotator = sv.LabelAnnotator(text\_thickness=2, text\_scale=1.5,

text\_color=sv.Color.BLACK) # используется для создания текстовых меток

line\_zone\_annotator = sv.LineZoneAnnotator(thickness=4, text\_thickness=4,

text\_scale=2) # создает визуальную линию для восприятия человеком

def callback(frame: np.ndarray, index: int) -> np.ndarray:

results = model(frame, verbose=False)[0] # записываем результат

detections = sv.Detections.from\_ultralytics(results) # определяем и находим людей

detections = detections[np.isin(detections.class\_id, SELECTED\_CLASS\_IDS)] # записываем айди

detections = byte\_tracker.update\_with\_detections(detections) # обновляем слежение

labels = [

f"# {model.model.names[class\_id]} {confidence:0.2f}" # записываем информацию для отображении на рамках

for confidence, class\_id, #

in zip(detections.confidence, detections.class\_id, )

]

annotated\_frame = frame.copy() # копируем отформатированный кадр

annotated\_frame = box\_annotator.annotate( # создаем рамки для готового кара

scene=annotated\_frame, detections=detections)

annotated\_frame = label\_annotator.annotate(

scene=annotated\_frame, detections=detections, labels=labels)

line\_zone.trigger(detections) # проход через линию

return line\_zone\_annotator.annotate(annotated\_frame, line\_counter=line\_zone) # возвращаем результат

sv.process\_video( # #аннотируем все видео

source\_path=SOURCE\_VIDEO\_PATH,

target\_path=TARGET\_VIDEO\_PATH,

callback=callback

)

showinfo(title="аннотация", message='Видео аннотировалось')

def opredvideo():

window2 = Tk()

window2.title("Новое окно")

window2.geometry("800x600")

Label1 = ttk.Label(window2,text= 'Выберите видео для аннотации')

Label1.pack(anchor= CENTER)

Button3 = ttk.Button(window2,text = "выбрать",command= lambda:filepath(window2,))

Button3.pack(anchor=CENTER)

def filepath(window2,):

filepath = filedialog.askopenfilename()

path = filepath

window2.destroy()

click(path,)

def n():

global temp1

temp1 = 'yolov8n.pt'

return (temp1)

def m():

global temp1

temp1 = 'yolov8m.pt'

return (temp1)

def x():

global temp1

temp1 = 'yolov8x.pt'

return (temp1)

def click(path):

window1 = Tk()

window1.title("Новое окно")

window1.geometry("800x600")

open\_button12 = ttk.Button(window1,text="запуск", command=lambda: proverka(window1,entry1, entry2, entry3, entry4,entry5,path,))

open\_button12.pack(anchor=CENTER) # задаем ее местоположение

inflabel = ttk.Label(window1,text = 'В первых двух полях необходимо указать начальные координаты триггерной линии,а в двух последних конечные координаты')

inflabel.pack(anchor= CENTER)

entry1 = ttk.Entry(window1)

entry1.pack(anchor=CENTER)

entry2 = ttk.Entry(window1)

entry2.pack(anchor=CENTER)

entry3 = ttk.Entry(window1)

entry3.pack(anchor=CENTER)

entry4 = ttk.Entry(window1)

entry4.pack(anchor=CENTER)

SOURCE\_VIDEO\_PATH = path # директория + название

video\_info = sv.VideoInfo.from\_video\_path(SOURCE\_VIDEO\_PATH)

resulutionlabel = ttk.Label(window1,text = f"Расширение видео : {video\_info.resolution\_wh} ")

resulutionlabel.pack(anchor=CENTER)

Label12 = ttk.Label(window1, text="Выберите модель")

Label12.pack(anchor=CENTER)

Button4 = ttk.Button(window1, text=f'yolov8n',command=n)

Button4.pack(anchor=CENTER)

Button5 = ttk.Button(window1, text=f'yolov8m',command=m)

Button5.pack(anchor=CENTER)

Button6 = ttk.Button(window1, text=f'yolov8x',command=x)

Button6.pack(anchor=CENTER)

Label13 = ttk.Label(window1, text="Выберите конечный путь файла")

Label13.pack(anchor=CENTER)

entry5 = ttk.Entry(window1)

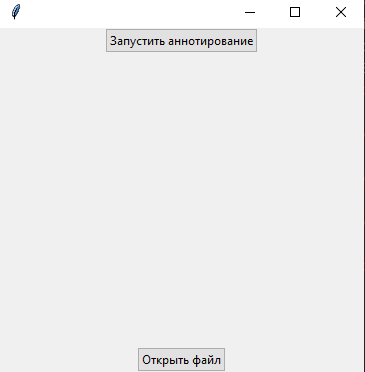
entry5.pack(anchor=CENTER)

opencurfile()#запускаем программу

# 

# Разработка пользовательского интерфейса

Пользовательский интерфейсе в проекте состоит из нескольких диалоговых окон. На первом окне есть две кнопки, с помощью одной можно открыть любой файл на компьютере,эта кнопка была выполнена с целью удобства открытия аннотированного видео,а с помощью второй можно запустить непосредственно аннотирование.

Рис. 2. Пользовательский интерфейс  


После нажатия верхней кнопки всплывает второе окно,на котором нужно выбрать желаемое видео для аннотации.

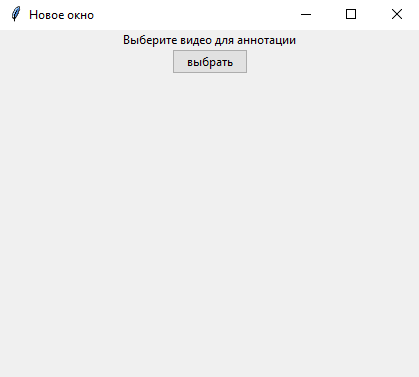


Рис.3.Пользовательский интерфейс

После выбора всплывает третье окно,на котором нужно указать дополнительные параметры для аннотации.

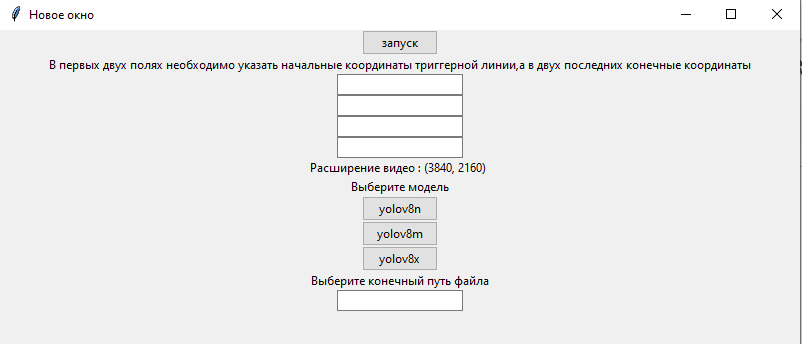


Рис.4.Пользовательский интерфейс

Для реализации пользовательского интерфейса была использованы библиотеки “os” и “tkinter”.Библиотека “os” позволяет работать с файлами,а в конкретном случае для открывания файла.Библиотека “tkinter” создает непосредственно интерфейс, т. е. поле,кнопку,заголовок а также возможность закрытия диалогового окна и его свертывания.

# Произвести тестирование алгоритма в реальных условиях

В ходе проверок было выявлено несоответствие оборудования для исполнения алгоритма.После первого пробного запуска компьютер загрузился и не мог повторно выполнить программу.Это свидетельствует об устаревшем оборудовании.

Далее были проведены эксперименты в различных условиях. В затемненных условиях программа работает стабильно. Цели не теряются и правильно аннотируются. В условиях частичного закрытия объекта наблюдения алгоритм продолжает корректную работу. При быстром движении объекты обнаруживаются, но иногда пропадают при быстром смене ракурса. При сильном замыливании видео работа происходит штатно, люди аннотируются.

Рис. 5. пример работы кода

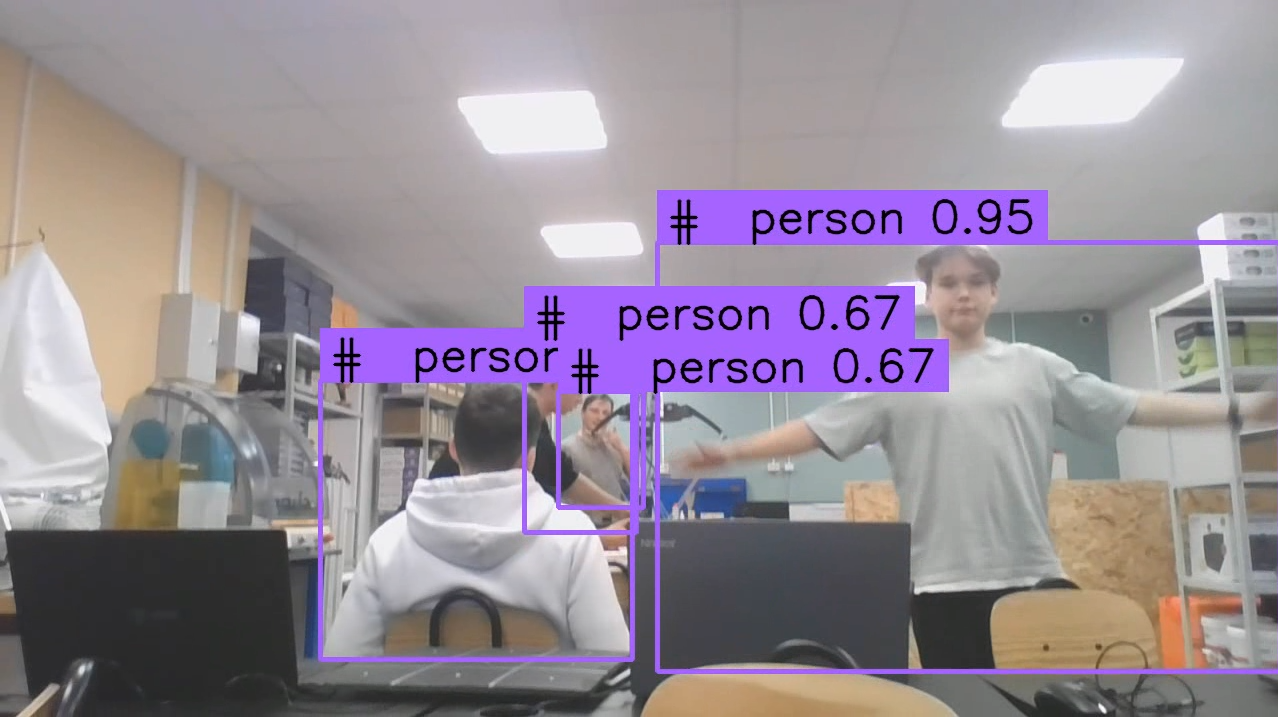
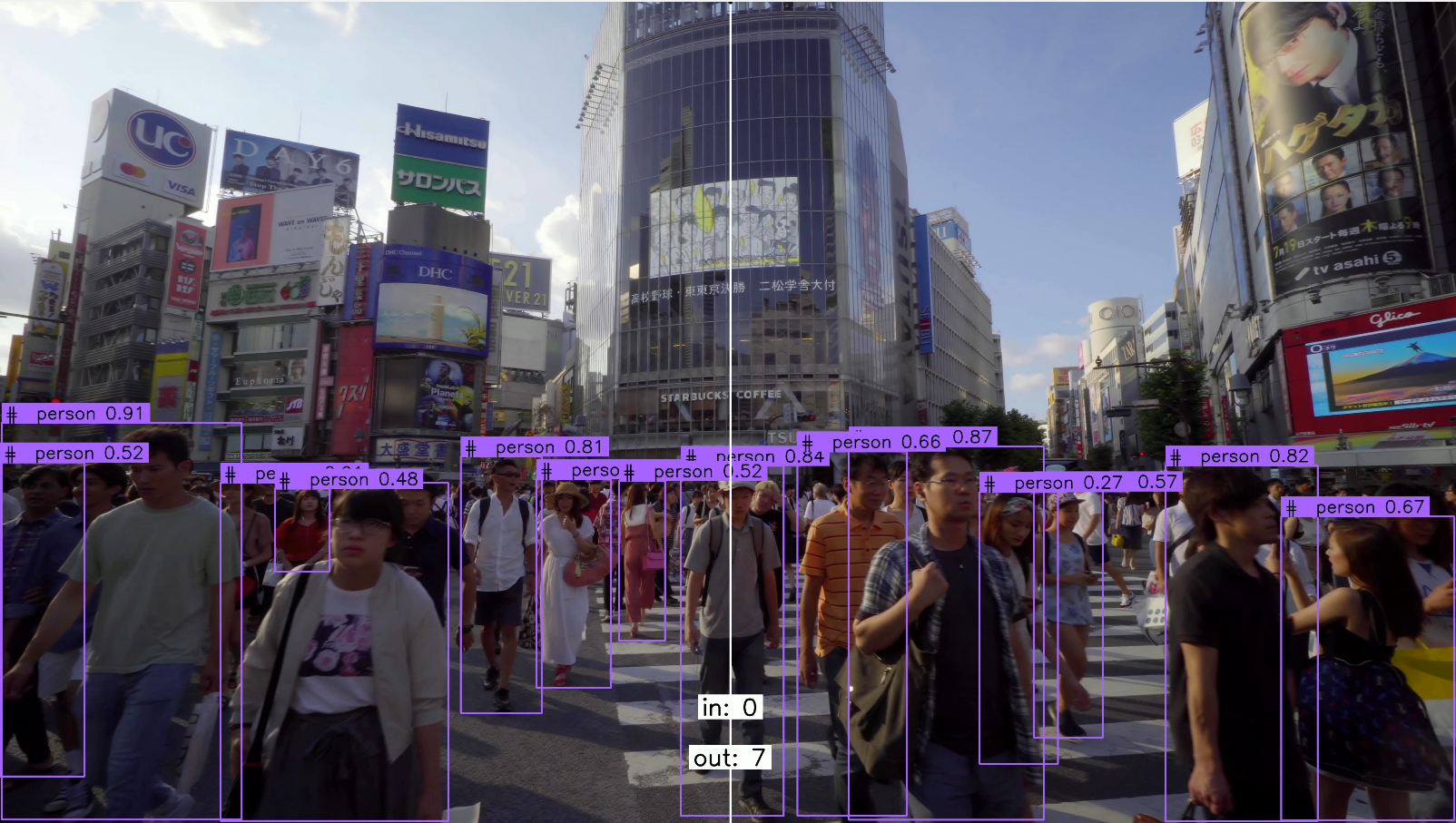


Рис.6. пример работы кода



# Заключение

Был рассмотрен рынок компьютерного зрения: в этом проекте была выбрана модель Yolo (You Once Look One), которая представляет собой нейросеть для определения различных объектов.V8 была выбрана за счет ее изученности и легкости в применении.

Разработан алгоритм в виде блок-схемы работы программного кода: блок схема помогает понять пошаговую работу проекту. Интерпретирован алгоритм на основе нейронной сети yolov8: алгоритм работает корректно в большинстве случаев,возможна оптимизация. Разработан пользовательский интерфейс: пользовательский интерфейс находится в начальном положении,планируется доработка. Произведено тестирование алгоритма в реальных условиях: тесты выявили корректность работы алгоритма,а также несоответствие оборудования для полноценной работы алгоритма.

# Список литературы

* Распознавание лиц и силуэтов людей, автомобилей и номерных знаков // Ntechlab URL:<https://ntechlab.ru/> (дата обращения: 06.07.2025).
* Компьютерное зрение в 2024 году // Хабр URL: https://habr.com/ru/companies/otus/articles/810207/ (дата обращения: 06.07.2025).
* Финансы // Altindex URL: https://altindex.com/ (дата обращения: 06.07.2025).
* Рынок компьютерного зрения вырастет на 81% и достигнет стоимости в $47 млрд к 2030 году // https://finance.rambler.ru/economics/52859288-rynok-kompyuternogo-zreniya-vyrastet-na-81-i-dostignet-stoimosti-v-47-mlrd-k-2030-godu/ (дата обращения: 06.07.2025).
* DeepSORT: Deep Learning to Track Custom Objects in a Video // DeepSort URL: https://nanonets.com/blog/object-tracking-deepsort/ (дата обращения: 06.03.25).
* Real-time Tracking via On-line Boosting. In Proceedings British Machine Vision Conference (BMVC). 2006. vol. 1. Р. 47–56. // Online Tracking Booster URL: https://www.researchgate.net/publication/221259753\_Real-Time\_Tracking\_via\_On-line\_Boosting (дата обращения: 06.03.25).
* The MIL algorithm trains a classifier in an online manner to separate the object from the background // Mil Tracker URL: https://docs.opencv.org/4.x/javadoc/org/opencv/video/TrackerMIL.html (дата обращения: 06.03.25).