

Разработка программы для распознавания автомобилей на языке программирования Python

Звайгзне Алексей Юрьевич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье описывается процесс разработки программы для распознавания автомобилей на фото, видео, видео с камеры устройства в реальном времени. При разработке используется среда для разработки Pycharm на языке программирования Python. В результате работы будет создана программа способная распознавать автомобили на медиаметериалах.

Ключевые слова: Pycharm, нейронные сети, YOLO, видео, фото, tkinter

Development of a program for recognizing cars in the Python programming language

Zvaigzne Alexey Yurievich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article describes the process of developing a program for recognizing cars in photos, videos, and videos from the device's camera in real time. The development uses the Pycharm development environment in the Python programming language. As a result of the work, a program will be created capable of recognizing cars on media materials.

Keywords: Pycharm, neural networks, YOLO, video, photo, tkinter

1 Введение

1.1 Актуальность

Актуальность разработки программы для распознавания автомобилей на Python обусловлена стремительным развитием технологий компьютерного зрения и искусственного интеллекта, а также растущим спросом на автоматизированные системы анализа транспортных потоков. С увеличением количества автомобилей во всем мире возникает необходимость в эффективных решениях для управления дорожным движением, контроля парковок, обеспечения безопасности и оптимизации логистики. Современные библиотеки Python, такие как OpenCV, TensorFlow и PyTorch, вместе с передовыми моделями детекции объектов, включая

YOLO, позволяют создавать точные и быстрые системы распознавания, доступные для широкого круга разработчиков.

Программы для распознавания автомобилей находят применение в различных сферах, включая умные города, где они используются для автоматического взимания платы за проезд, мониторинга аварий и управления транспортными потоками. В бизнес-секторе такие системы помогают логистическим компаниям отслеживать транспорт, торговым центрам анализировать посещаемость, а правоохрательным органам выявлять нарушения. Кроме того, Python остается одним из наиболее подходящих языков для подобных задач благодаря простоте интеграции с нейросетевыми фреймворками, обширному сообществу и наличию готовых решений.

1.2 Обзор исследований

В своей статье К.Н. Трубин описывает методы и алгоритмы предназначенные для улучшения производительности и эффективности работы нейронных сетей [1]. В монографии К.А. Аржановой описывается использование нейросетей в продвижении и внедрении брендов [2]. В статье В.В. Ксефонтова рассматривается понятие нейронных сетей, принцип работы и обучаемость сетей [3]. В работе Б.С. Горячкина рассматриваются подходы к решению задач компьютерного зрения [4]. В статье А.М. Андодворцева обосновывается актуальность задачи сегментации объектов на изображениях в сфере машинного обучения, в частности, в классе задач машинного обучения по компьютерному зрению [5]. В работе Р.Н. Алифанова анализируется возможность использования нейросетей в пассивно-активных гидроакустических системах обнаружения и распознавания крабов [6]. В научной диссертации В.И. Шахуро рассматривается использование нейросетей в распознавании дорожных знаков на изображениях [7].

1.3 Цель исследования

Целью исследования является – Разработка программы для распознавания автомобилей на языке Python с использованием методов компьютерного зрения и глубокого обучения, обеспечивающей высокую точность детекции в реальном времени.

2 Материалы и методы

Программа разрабатывается на языке программирования Python в среде разработки Pycharm, с использованием готовой модели искусственного интеллекта распознавания элементов на изображении YOLO v8.

3 Результаты и обсуждения

Для начала работы создается новая среда для исключения конфликтов работы библиотек, в данном случае это Pure Python (рис. 1).

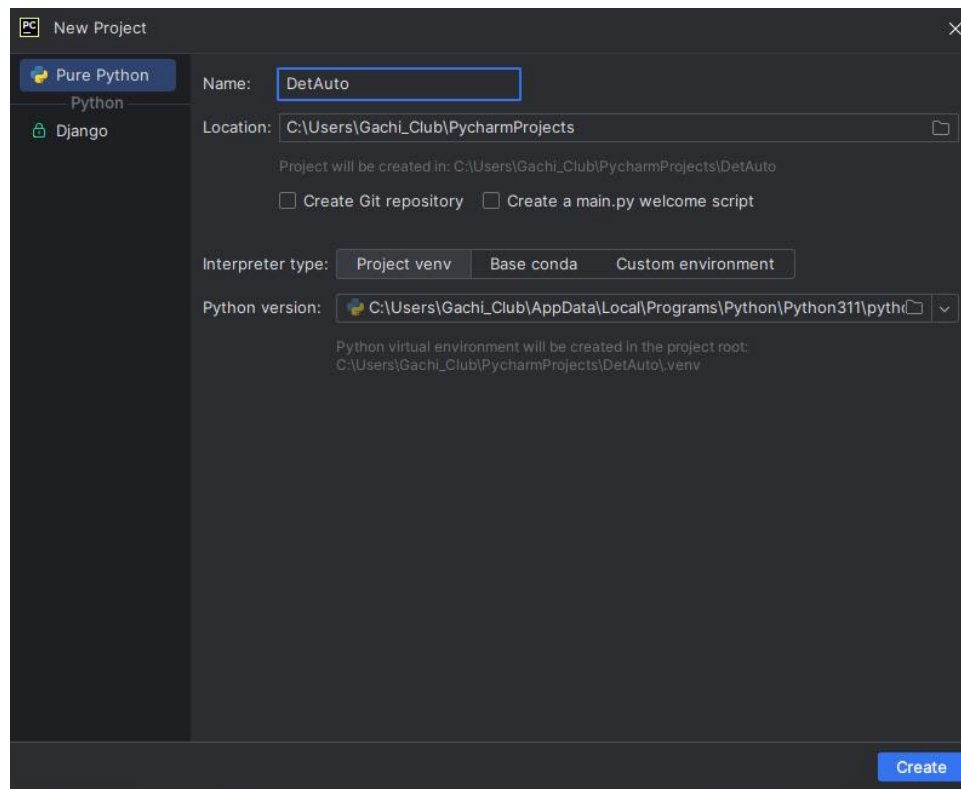


Рисунок 1. Создание проекта в PyCharm

После создания проекта необходимо установить необходимые для работы библиотеки через терминал командой:

```
pip install ultralytics opencv-python matplotlib
```

Ultralytics содержит в себе инструменты для работы с моделями нейросетей, в данном случае семейства YOLO. CV2 нужен для вывода результатов обработки визуально отображая полученный результат.

Для работы с изображениями в PyCharm создается файл detectPH.py содержащий код для работы с моделью (рис. 2).

```
detectPH.py x
1  from ultralytics import YOLO
2  import cv2
3
4  # Загрузка модели YOLOv8
5  model = YOLO("yolov8n.pt")
6
7  # Путь к фото (можно просто "car.jpg", если оно в той же папке)
8  image_path = "car.jpg"
9
10 # Запуск распознавания
11 results = model(image_path)
12
13 # Вывод результата
14 for r in results:
15     im_array = r.plot() # Прямоугольники вокруг автомобилей
16     cv2.imshow( winname: "Детектор машин", im_array)
17     cv2.waitKey(0)
18     cv2.destroyAllWindows()
```

Рисунок 2. Код для работы с фото

При компиляции данного кода в папке с проектом должен подгрузиться файл предобученной модели, в папке проекта должна находиться картинка car.jpg для обработки. Для проверки работоспособности было выбрано случайное изображение из интернета (Рис. 3).



Рисунок 3. Вывод результатов работы модели

Для проверки работы видео обработки в новом файле detectVid.py был описан код работы с камерой и видеофайлами (рис. 4).

```

detectVid.py x
1  from ultralytics import YOLO
2  import cv2
3
4  model = YOLO("yolov8n.pt") # Загрузка модели
5  #cap = cv2.VideoCapture("video.mp4") # Указать путь к видео
6  cap = cv2.VideoCapture(0) # Подключение камеры (0 – это встроенная камера)
7
8  while True:
9      ret, frame = cap.read() # Получение кадра с камеры
10
11     if not ret:
12         break # Если кадр не получен, выход
13
14     # Запуск распознавания (conf=0.7 – только уверенные обнаружения)
15     results = model(frame, conf=0.7)
16
17     # Рисует прямоугольники вокруг машин
18     annotated_frame = results[0].plot()
19
20     # Показывает результат
21     cv2.imshow( winname: "Car Detection", annotated_frame)
22
23     # Выходит по клавише ESC
24     if cv2.waitKey(1) == 27:
25         break
26
27 cap.release() # Отключает камеру
28 cv2.destroyAllWindows() # Закрывает все окна
29

```

Рисунок 4. Код для работы с видео или камерой

Код, отвечающий за обработку видеофайла (файл должен находиться в папке проекта (Рис. 5):

```
cap = cv2.VideoCapture("video.mp4") # Указать путь к видео
```

Код, отвечающий за обработку видео с камеры (Рис. 6):

```
cap = cv2.VideoCapture(0) # Подключение камеры (0 — это встроенная камера).
```



Рисунок 5. Вывод результатов в реальном времени при работе с видео

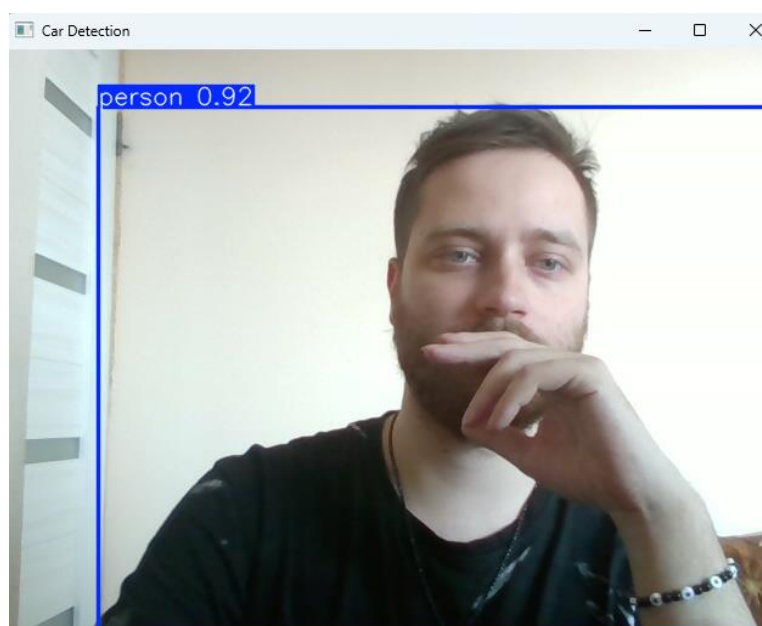


Рисунок 6. Вывод результатов с камеры

В данной модели по умолчанию уже есть необходимые настройки для работы с фото и видеоматериалами позволяющими давать результаты в реальном времени, остается только добавить интерфейс для работы с пользователем.

В рамках программы будет включен весь ранее описанный функционал, отключено отображение всех объектов кроме легковых автомобилей, грузовых машин, автобусов и мотоциклов. Для создания программы создается новый файл Python app.py. Для создания интерфейса используется библиотека tkinter, а для сборки готового проекта используется библиотека pyinstaller.

Листинг код готовой программы:

```
from tkinter import *
from tkinter import filedialog
from ultralytics import YOLO
import cv2
from PIL import Image, ImageTk

class CarDetectorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Детектор машин")
        self.root.geometry("800x600")
        # Инициализация модели YOLO
        self.model = YOLO("yolov8n.pt")
        # Переменные состояния
        self.file_path = ""
        self.is_camera_active = False
        self.cap = None
        self.current_image = None
        self.original_size = (0, 0)
        # Настройка интерфейса
        self.create_widgets()
        self.setup_layout()
        # Обработчики событий
        self.root.protocol("WM_DELETE_WINDOW", self.on_close)
    def create_widgets(self):
        # Canvas для отображения изображения/видео
        self.canvas = Canvas(self.root, bg="gray", bd=0, highlightthickness=0)
        # Панель управления
        self.control_frame = Frame(self.root)
        self.btn_open = Button(self.control_frame, text="Выбрать файл",
command=self.open_file)
        self.btn_camera = Button(self.control_frame, text="Камера: Выкл",
command=self.toggle_camera)
        self.btn_start = Button(self.control_frame, text="Старт",
command=self.start_detection)
        self.btn_stop = Button(self.control_frame, text="Стоп",
command=self.stop_detection)
        self.label_file = Label(self.control_frame, text="Файл не выбран",
wraplength=400)
        self.label_status = Label(self.control_frame, text="Готово", fg="black")
    def setup_layout(self):
        # Настройка grid
        self.root.grid_rowconfigure(0, weight=1)
        self.root.grid_columnconfigure(0, weight=1)
        # Расположение элементов
```

```
self.canvas.grid(row=0, column=0, sticky="nsew")
self.control_frame.grid(row=1, column=0, sticky="ew", padx=10,
pady=5)
# Элементы управления
self.btn_open.pack(side=LEFT, padx=5)
self.btn_camera.pack(side=LEFT, padx=5)
self.btn_start.pack(side=LEFT, padx=5)
self.btn_stop.pack(side=LEFT, padx=5)
self.label_file.pack(side=LEFT, padx=5)
self.label_status.pack(side=RIGHT, padx=5)
# Обработчик изменения размера окна
self.canvas.bind("<Configure>", self.resize_handler)
def open_file(self):
    self.file_path = filedialog.askopenfilename(
        filetypes=[("Все файлы", "*.*"),
                    ("Изображения", "*.jpg *.png *.jpeg"),
                    ("Видео", "*.mp4 *.avi *.mov")])
    if self.file_path:
        self.label_file.config(text=f"Выбран файл: {self.file_path}")
        self.label_status.config(text="Готово", fg="black")
def toggle_camera(self):
    self.is_camera_active = not self.is_camera_active
    state = "Вкл" if self.is_camera_active else "Выкл"
    self.btn_camera.config(text=f"Камера: {state}")
def start_detection(self):
    self.stop_detection()
    if self.is_camera_active:
        self.cap = cv2.VideoCapture(0)
        self.process_video()
    elif self.file_path:
        if self.file_path.lower().split('.')[-1] in ['mp4', 'avi', 'mov']:
            self.cap = cv2.VideoCapture(self.file_path)
            self.process_video()
        else:
            self.process_image()
    else:
        self.label_status.config(text="Ошибка: не выбран источник!",
fg="red")
def process_image(self):
    frame = cv2.imread(self.file_path)
    if frame is not None:
        self.process_frame(frame)
        self.label_status.config(text="Обработка завершена", fg="green")
    else:
```

```
self.label_status.config(text="Ошибка загрузки изображения",
fg="red")
def process_video(self):
    if self.cap.isOpened():
        ret, frame = self.cap.read()
        if ret:
            self.process_frame(frame)
            self.root.after(10, self.process_video)
        else:
            self.stop_detection()
def process_frame(self, frame):
    #results = self.model(frame, conf=0.5)
    results = self.model(frame, classes=[2, 5, 7, 3])
    annotated_frame = results[0].plot()
    # Конвертация и сохранение оригинального размера
    img = cv2.cvtColor(annotated_frame, cv2.COLOR_BGR2RGB)
    self.original_image = Image.fromarray(img)
    self.original_size = self.original_image.size
    self.update_display()
def update_display(self):
    if not self.original_image:
        return
    canvas_width = self.canvas.winfo_width()
    canvas_height = self.canvas.winfo_height()
    if canvas_width <= 0 or canvas_height <= 0:
        return
    # Расчет масштаба
    width_ratio = canvas_width / self.original_size[0]
    height_ratio = canvas_height / self.original_size[1]
    scale = min(width_ratio, height_ratio)
    new_size = (
        int(self.original_size[0] * scale),
        int(self.original_size[1] * scale)
    )
    resized_image = self.original_image.resize(new_size,
Image.LANCZOS)
    self.current_image = ImageTk.PhotoImage(resized_image)
    # Центрирование изображения
    self.canvas.delete("all")
    x = (canvas_width - new_size[0]) // 2
    y = (canvas_height - new_size[1]) // 2
    self.canvas.create_image(x, y, anchor=NW, image=self.current_image)
def resize_handler(self, event=None):
    if self.original_image:
        self.update_display()
```

```
def stop_detection(self):
    if self.cap is not None:
        self.cap.release()
        self.cap = None
    self.canvas.delete("all")
    self.label_status.config(text="Остановлено", fg="black")
def on_close(self):
    self.stop_detection()
    self.root.destroy()
if __name__ == "__main__":
    root = Tk()
    app = CarDetectorApp(root)
    root.mainloop()
```

В процессе создания приложения было выявлено, что необходимо автоматически масштабировать размер окна под содержимое, а также изменять содержимое под размер рабочего окна.

В программе при запуске есть 3 кнопки и 1 переключатель состояния выбора камеры (рис 7).



Рисунок 7. Рабочее окно программы

Если установить состояние переключателя «Камера: Выкл» на «Камера: Вкл», то выбранные медиафайлы обрабатываться не будут. В будущем для доработки интерфейса приложения можно использовать более современные библиотеки или использовать другую среду разработки используя готовую модель нейросети.

В функции `process_frame`, в фильтрации указаны конкретные классы:

- 2 – легковые автомобили;
- 3 – мотоциклы;

- 5 – автобусы;
- 7 – грузовики.

Это нужно для исключения остальных объектов, которые не требуются в задаче, но модель все равно их распознает, не передавая обработчикам на отрисовку.

Программа работает аналогично ранее отработанным отдельным элементам обработки фото, видео и видео с камеры, выдавая расчеты по нажатию кнопки «Старт», кнопка «Стоп» необходима для остановки обработки записи (рис. 8-10).

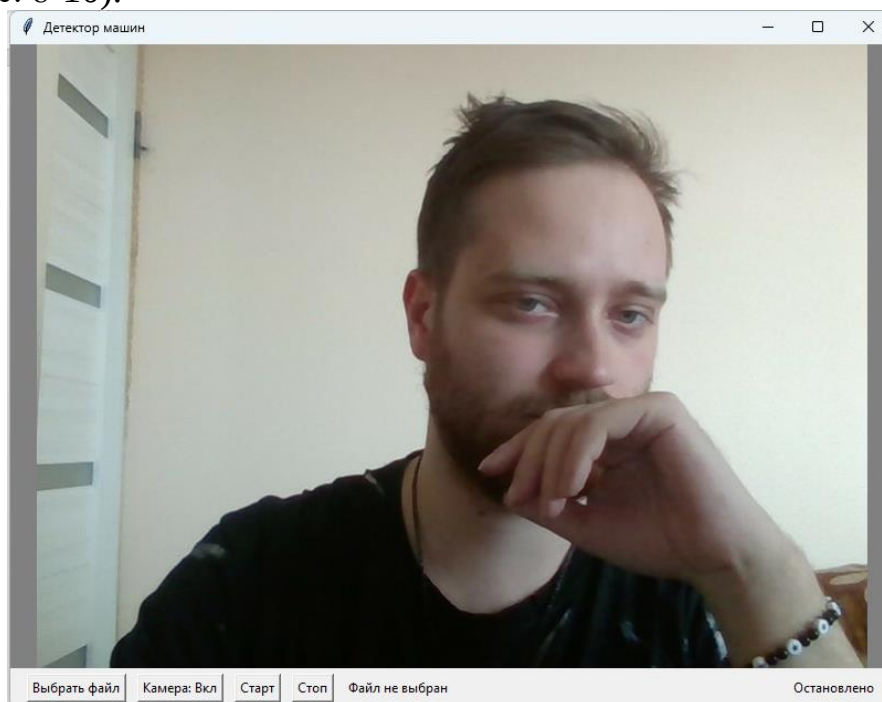


Рисунок 8. Обработка видео с камеры устройства

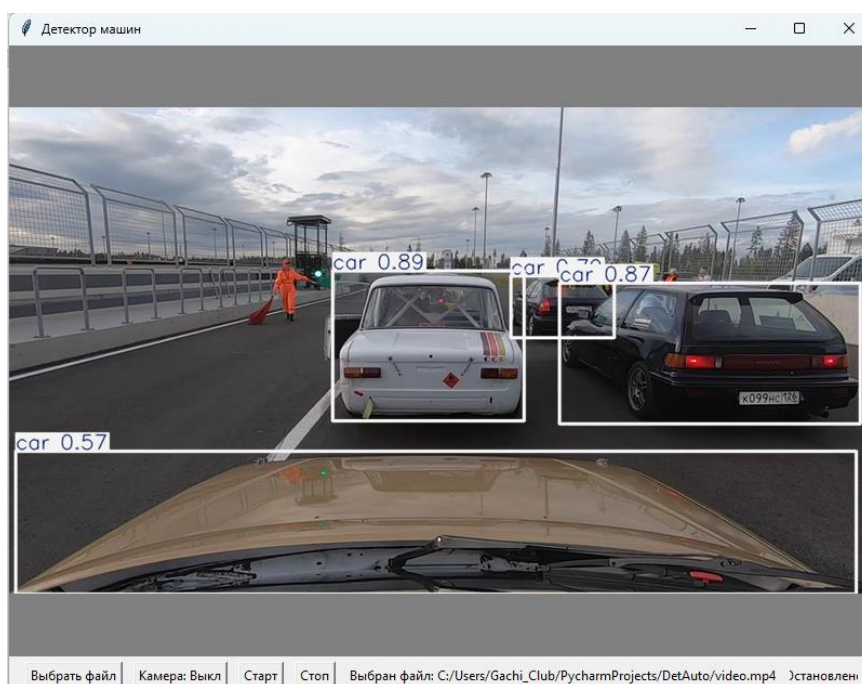


Рисунок 9. Результат распознавания объектов при обработке видеофайла

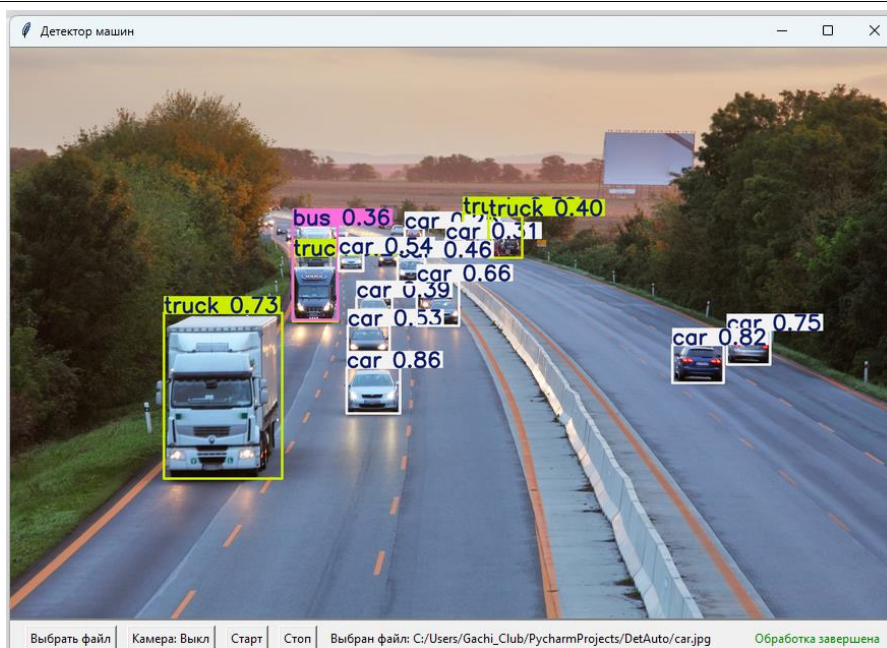


Рисунок 10. Результат распознавания объектов на изображении

В программе нет возможности обработки номеров, для этого необходимо использование дополнительной модели, нацеленной на распознавание номеров.

Выводы

Это может использоваться в дальнейшем в шиномонтажной промышленности для распознавания своих клиентов сокращая необходимое время обработки на ЭВМ в момент внесения сведений. Данная программа нуждается в доработке в части дальнейшего обучения моделей: YOLOv8n распознаванию марки автомобилей, добавить и обучить модель по распознаванию госномеров, разобраться с проблемой обработки входных данных ускорив процесс обработки и повысив точность полученных данных.

Библиографический список

1. Турбин К.Н., Петухов А.А. Методы оптимизации работы нейросетей. Рязань: Рязанский государственный радиотехнический университет им. В.Ф. Уткина, 2018.
2. Аржанова К. А. Разработка программ внедрения нейросетей в деятельность компаний. М.: Общество с ограниченной ответственностью "Русайнс", 2024. 160 с.
3. Ксенофонтов В. В. Нейронные сети //Проблемы науки. 2020. №. 11. С. 59.
4. Горячкин Б. С., Китов М. А. Компьютерное зрение //E-scio. 2020. №. 9 (48). С. 317-345.
5. Аднодворцев А. М. и др. Создание интеллектуальных систем принятия решений на базе технологии свёрточных нейронных сетей //Фабрика будущего: переход к передовым цифровым, интеллектуальным производственным технологиям, роботизированным системам для

- отраслей пищевой промышленности. 2022. С. 25-32.
6. Алифанов Р.Н. Пассивно-активная гидроакустическая система обнаружения и распознавания крабов с использованием нейросетей: дис. канд. тех. наук: 05.18.17. Владивосток, 2010. 197 с.
 7. Шахуро В. И. Распознавание редких дорожных знаков с использованием синтетических обучающих выборок: дис. канд. физ.-мат. наук: 05.13.17. М., 2021. 93 с.