

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Саратовский государственный технический университет имени Гагарина
Ю.А.»
Кафедра «Информационно-коммуникационные системы и программная
инженерия»

Проект
по дисциплине «Проектирование интеллектуальных систем»
на тему:
«Обучение нейронной сети распознаванию пластиковых бутылок на
изображении»

Выполнил:
Кудряшов Алексей Владимирович
м-ИВЧТ-21

Проверил: Ивженко Сергей Петрович

Саратов 2024

Содержание

Введение	4
1 Теоретическая часть	6
1.1 Нейронные сети	6
1.1.1 Что такое нейронные сети?	6
1.1.2 Основы работы нейронных сетей	6
1.1.3 Типы нейронных сетей	7
1.1.4 Обучение нейронной сети	8
1.2 YOLO (You Only Look Once)	11
1.2.1 Что такое YOLO?	11
1.2.2 Ultralytics YOLO11	11
1.2.3 Принцип работы YOLO	12
2 Практическая часть	14
2.1 Среда разработки	14
2.2 Подготовка датасета	14
2.2.1 Адаптация для формата YOLO	15
2.2.2 Редактирование изображений	16
2.2.3 Разделение датасета	16
2.2.4 Создание файла конфигурации	17
2.3 Обучение	17
2.3.1 Библиотека YOLO	17
2.3.1 Настройки обучения	17
2.3.2 Процесс обучения	19
2.3.3 Возможности дообучения	19
2.4 Результаты	20

Заключение	22
Список литературы.....	23

Введение

В последние годы обработка изображений и распознавание объектов на них становятся важными аспектами в различных областях, таких как безопасность, промышленность и экология. Одной из актуальных задач является автоматическое распознавание пластиковых бутылок на изображениях для различных целей: сортировка отходов, улучшение систем управления производственными процессами или анализ экологии. Применение нейронных сетей, таких как YOLO (You Only Look Once), позволяет эффективно и быстро решать задачи обнаружения объектов в реальном времени.

Использование нейронных сетей для распознавания объектов на изображениях является важной технологией для автоматизации различных процессов. Пластиковые бутылки являются одним из наиболее распространённых видов отходов, и их эффективное распознавание имеет значение как для сортировки мусора, так и для улучшения работы систем на основе компьютерного зрения. YOLOv11, как одна из наиболее эффективных моделей для задач распознавания объектов, позволяет достигать высокой точности при минимальных вычислительных затратах.

В процессе разработки решения необходимо собрать и подготовить данные (изображения пластиковых бутылок), а затем применить модель YOLOv11 для их обучения. Модель будет проходить через несколько этапов, включая детекцию объектов, оценку производительности и анализ результатов с применением различных метрик точности, таких как точность (precision) и полнота (recall).

Целью данного проекта является разработка и обучение нейронной сети, использующей архитектуру YOLOv11, для точного распознавания пластиковых бутылок на изображениях. Основные задачи включают:

1. Сбор и подготовка датасета с изображениями пластиковых бутылок.
2. Настройка и обучение модели YOLOv4 на этом датасете.

3. Оценка точности и производительности модели.
4. Разработка программного обеспечения для практического применения модели.

1 Теоретическая часть

1.1 Нейронные сети

1.1.1 Что такое нейронные сети?

Нейронная сеть — это математическая модель, вдохновленная работой человеческого мозга, которая используется для обработки данных и решения различных задач. Нейронные сети представляют собой систему взаимосвязанных узлов (нейронов), которые принимают входные данные, обрабатывают их и выдают результаты.

Наиболее часто используемыми являются перцептроны, свёрточные нейронные сети (CNN), рекуррентные нейронные сети (RNN), которые применяются в различных областях, таких как обработка изображений, видео, текста и т.д.

Можно объяснить нейронные сети через аналогию с нейронами в мозге: каждый нейрон получает сигналы от других нейронов и передает результат по своему выходу. Сигналы проходят через соединения (синапсы), вес которых можно обучить для точной передачи информации.

1.1.2 Основы работы нейронных сетей

Каждый нейрон получает несколько входных значений, которые комбинируются с помощью весов. Входные данные проходят через функцию активации, которая решает, будет ли нейрон активирован (выдаст ли он сигнал).

В процессе обучения сеть получает обучающие данные и корректирует веса нейронов с помощью алгоритма обратного распространения ошибки и метода градиентного спуска.

Слои нейронной сети: В нейронной сети могут быть несколько слоев (рисунок 1):

- **Входной слой:** получает исходные данные.
- **Скрытые слои:** обрабатывают данные.
- **Выходной слой:** выдает результаты.

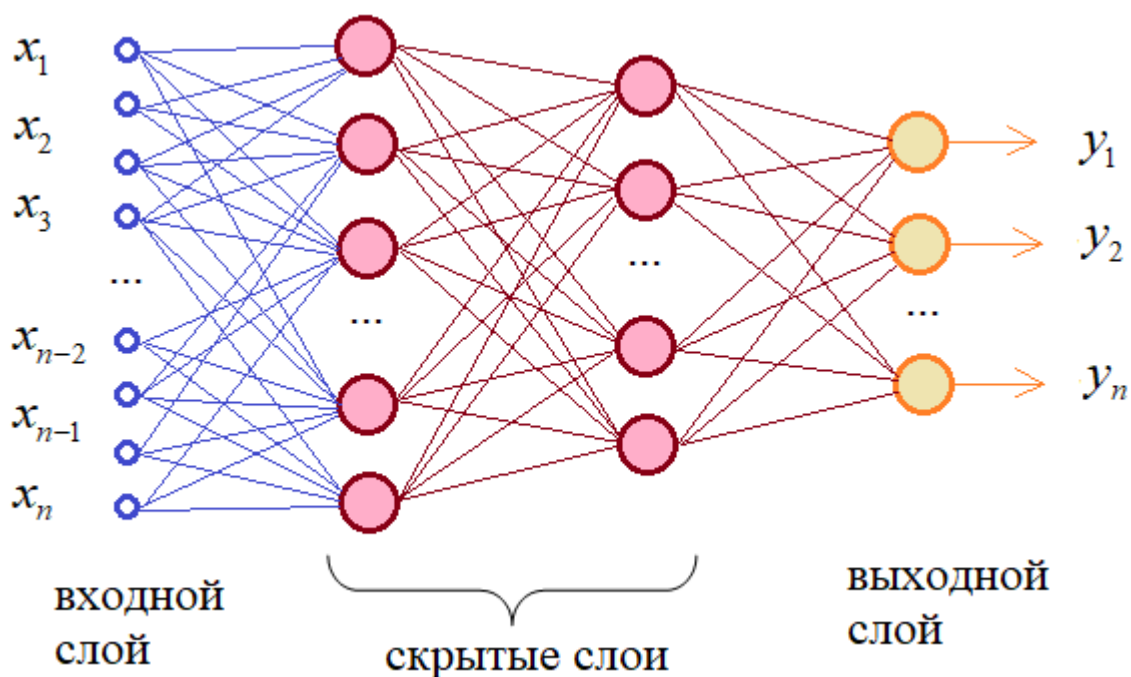


Рисунок 1 – Структура нейронной сети

Каждый нейрон в сети имеет вес, который регулируется в процессе обучения. Когда нейрон получает входные данные, эти данные умножаются на соответствующий вес, затем проходят через функцию активации. Функция активации определяет, будет ли нейрон активирован и, следовательно, передаст ли он информацию в следующий слой.

Функции активации играют ключевую роль в нейронных сетях, так как они помогают определить, будет ли нейрон активирован. Наиболее популярные функции активации:

- **Sigmoid** — ограничивает выходные значения от 0 до 1.
- **ReLU (Rectified Linear Unit)** — используется в большинстве современных сетей. Она заменяет все отрицательные значения на ноль, что позволяет ускорить обучение.
- **tanh** — схожа с sigmoid, но выходные значения варьируются от -1 до 1.

1.1.3 Типы нейронных сетей

Существует несколько типов нейронных сетей, каждый из которых оптимизирован для решения определённых задач:

- **Перцептрон** — простейшая нейронная сеть, которая состоит из одного слоя нейронов.

- **Свёрточные нейронные сети (CNN)** — используются для обработки изображений. Эти сети включают свёрточные слои, которые помогают выделять особенности изображений (например, края, углы и текстуры).
- **Рекуррентные нейронные сети (RNN)** — применяются для обработки последовательностей, таких как текст или временные ряды.

Для задач компьютерного зрения, таких как распознавание объектов, чаще всего применяют свёрточные нейронные сети (CNN), так как они эффективно извлекают особенности изображений.

1.1.4 Обучение нейронной сети

Обучение нейронной сети — это процесс корректировки её параметров (весов и смещений) с целью минимизации ошибки между предсказанным и реальным значением. Этот процесс включает несколько этапов, начиная с подачи обучающих данных и заканчивая оптимизацией модели для получения лучших результатов. Обучение нейронной сети может быть разделено на несколько ключевых этапов:

1. **Подача данных в сеть:** на первом этапе данные (например, изображения, текст или числовые данные) подаются во входной слой нейронной сети. Эти данные проходят через скрытые слои, где происходит их обработка, и в конечном итоге сеть генерирует предсказания на выходном слое.
2. **Обратное распространение ошибки (Backpropagation):** когда сеть делает предсказание, сравнивается его результат с истинным значением (например, меткой объекта на изображении). Разница между предсказанием и истинным значением называется ошибкой. Ошибка используется для корректировки параметров сети. Этот процесс называется **обратным распространением ошибки**. Он включает в себя следующие шаги:
 - Рассчитывается ошибка на выходе сети.

- Ошибка распространяется от выходного слоя к скрытым слоям, где для каждого нейрона вычисляется, как сильно он повлиял на ошибку.
- На основе этих данных обновляются веса и смещения каждого нейрона с помощью метода **градиентного спуска**.

3. **Градиентный спуск (Gradient Descent):** Градиентный спуск — это метод оптимизации, который используется для минимизации функции ошибки (или потерь). Это итеративный процесс, в котором вычисляется градиент функции потерь по отношению к каждому параметру сети, и затем параметры обновляются с шагом, пропорциональным этому градиенту. Процесс повторяется до тех пор, пока ошибка не станет минимальной или не достигнуты заранее установленные критерии сходимости.

4. **Методы оптимизации:** В стандартном градиентном спуске шаг обновления весов может быть фиксированным. Однако в реальных задачах для повышения эффективности используются усовершенствованные методы оптимизации:

- **Адаптивные методы (например, Adam, AdaGrad, RMSprop):** Эти методы автоматически изменяют скорость обучения в зависимости от того, как изменяются градиенты. Это помогает ускорить процесс обучения и предотвращает застревание в локальных минимумах.
- **Мини-батч градиентный спуск:** вместо того чтобы использовать всю обучающую выборку для вычисления градиента, часто используется её часть (мини-батч). Это снижает вычислительные затраты и позволяет улучшить сходимость.

5. **Функции потерь:** для обучения нейронной сети необходима функция потерь, которая измеряет, насколько далеко находятся предсказания модели от истинных значений. Для задач классификации часто

используется **кросс-энтропийная ошибка**, которая измеряет различие между истинным распределением меток и предсказанным. Для регрессионных задач применяется **среднеквадратичная ошибка** (MSE), которая вычисляет среднее квадратичное отклонение между предсказанными и реальными значениями.

6. **Регуляризация:** для предотвращения переобучения используется регуляризация — техники, которые ограничивают сложность модели. Один из распространённых методов — **L2-регуляризация** (или **Ridge-регуляризация**), которая добавляет штраф за большие значения весов в функцию потерь. Это заставляет модель предпочитать более простые решения.
7. **Этапы обучения:** Обучение нейронной сети состоит из нескольких этапов:
 - **Инициализация:** Начальные веса нейронной сети выбираются случайным образом или с использованием специализированных методов (например, Xavier или He initialization).
 - **Обучение:** Процесс, при котором сеть обновляет свои параметры, используя обучающие данные и метод градиентного спуска.
 - **Оценка:** после завершения обучения нейронную сеть проверяют на тестовых данных, которые не использовались в процессе обучения, чтобы убедиться, что модель не переобучена и способна обрабатывать новые данные.
8. **Переобучение и недообучение:** Одной из важных проблем при обучении нейронных сетей является переобучение (overfitting), когда модель слишком сильно подстраивается под обучающие данные и теряет способность обобщать на новые данные. Для борьбы с переобучением используются различные методы, такие как кросс-валидация, регуляризация, и использование более простых моделей.

Недообучение (underfitting) происходит, когда модель слишком простая и не может захватить все зависимости в данных.

9. **Тонкая настройка (Fine-tuning)**: после того как модель была обучена на большом объёме данных, её часто дообучают на более специфических задачах или с использованием меньшего объёма данных. Этот процесс называется **тонкой настройкой** и позволяет улучшить результаты модели для конкретных приложений.

1.2 YOLO (You Only Look Once)

1.2.1 Что такое YOLO?

YOLO (You Only Look Once) — это современная архитектура нейронной сети, предназначенная для решения задачи распознавания объектов в изображениях. Она была разработана с целью значительно ускорить процесс обнаружения объектов, обеспечивая высокую точность и эффективность. В отличие от традиционных методов, которые обрабатывают изображение в несколько этапов (например, выделение признаков, затем классификация), YOLO выполняет все операции за один проход через сеть, что делает ее гораздо быстрее и подходящей для реального времени.

Алгоритм YOLO применяет подход, который предполагает разбиение изображения на сетку (например, 13x13, 19x19), где каждая ячейка отвечает за предсказание объектов, расположенных в ее области. Это позволяет значительно сократить время обработки, так как для каждого объекта делается лишь один прогноз, а не несколько, как в традиционных методах, например, в R-CNN или его производных. В результате, YOLO значительно ускоряет процесс обработки, сохраняя высокую точность распознавания.

1.2.2 Ultralytics YOLO11

YOLO11 это последняя итерация в Ultralytics YOLO серии детекторов объектов в реальном времени, заново определяющая возможные возможности благодаря передовой точности, скорости и эффективности. Основываясь на впечатляющих достижениях предыдущих версий YOLO, YOLO11 вносит

значительные улучшения в архитектуру и методы обучения, что делает его универсальным выбором для широкого спектра задач компьютерного зрения.

Основные характеристики:

- Улучшенное извлечение признаков: на сайте YOLO11 используется усовершенствованная архитектура "позвоночника" и "шеи", которая расширяет возможности извлечения признаков для более точного обнаружения объектов и выполнения сложных задач.
- Оптимизирован для повышения эффективности и скорости: на сайте YOLO11 представлены усовершенствованные архитектурные решения и оптимизированные конвейеры обучения, обеспечивающие более высокую скорость обработки данных и оптимальный баланс между точностью и производительностью.
- Большая точность при меньшем количестве параметров: Благодаря усовершенствованию конструкции модели, YOLO11m достигает более высокой средней точности (mAP) на наборе данных COCO, используя при этом на 22 % меньше параметров, чем YOLOv8m, что делает его эффективным с вычислительной точки зрения без ущерба для точности.
- Адаптируемость к различным средам: YOLO11 можно легко развернуть в различных средах, включая пограничные устройства, облачные платформы и системы, поддерживающие графические процессоры NVIDIA, что обеспечивает максимальную гибкость.
- Широкий спектр поддерживаемых задач: Будь то обнаружение объектов, сегментация экземпляров, классификация изображений, оценка позы или ориентированное обнаружение объектов (OBB), YOLO11 разработан для решения разнообразных задач компьютерного зрения.

1.2.3 Принцип работы YOLO

YOLO использует принцип единого предсказания, разделяя изображение на сетку и предсказывая сразу несколько параметров для каждой ячейки сетки:

1. **Разбиение изображения на сетку:** каждый объект на изображении ассоциируется с одной или несколькими ячейками сетки. Сетка делится на области, например, 13x13 или 19x19. Каждая ячейка будет прогнозировать несколько параметров, таких как координаты ограничивающей рамки (bounding box), вероятность наличия объекта и его класс.
2. **Предсказание классов и координат рамок:** каждая ячейка сети предсказывает, присутствует ли объект в ее области и где он находится. Это делается через предсказание координат ограничивающих рамок, а также вероятность того, что объект является принадлежащим определенному классу (например, "пластиковая бутылка").
3. **Алгоритм подавления несущественных предложений (Non-Maxima Suppression):** после того как сеть сделала предсказания для всех ячеек, применяется алгоритм подавления несущественных предложений (NMS), чтобы устранить избыточные или перекрывающиеся рамки. Это помогает исключить лишние предсказания и оставляет только наиболее точные и уверенные рамки для каждого объекта.

2 Практическая часть

2.1 Среда разработки

Google Colaboratory (или просто Colab) — это бесплатная интерактивная облачная платформа, разработанная компанией Google, предназначенная для работы с кодом, анализа данных и выполнения задач машинного обучения. Она предоставляет пользователям онлайн-версию блокнота Jupyter с интеграцией Google Диска, что позволяет работать над проектами, не устанавливая дополнительные программы или библиотеки на локальную машину.

Преимущества:

При разработке проекта по обучению нейронной сети для распознавания пластиковых бутылок Google Colab был выбран благодаря следующим особенностям:

- Бесплатный доступ к GPU/TPU: Обучение YOLOv11 требует значительных вычислительных ресурсов. Использование GPU в Colab значительно ускоряет процесс обучения модели, снижая время на настройку экспериментов.
- Удобство работы с большими наборами данных: Интеграция с Google Диском позволяет легко загружать, обрабатывать и сохранять изображения, используемые в проекте.
- Универсальность среды: Возможность писать, тестировать и отлаживать код, визуализировать результаты и документировать проект в одном интерфейсе.

2.2 Подготовка датасета

В рамках проекта по обучению нейронной сети YOLOv4 для распознавания пластиковых бутылок был использован и адаптирован датасет [TrashBox](#), доступный в открытом доступе. Этот набор данных включает изображения различных видов отходов (рисунок 2), включая пластиковые бутылки, что делает его подходящим для целей исследования.

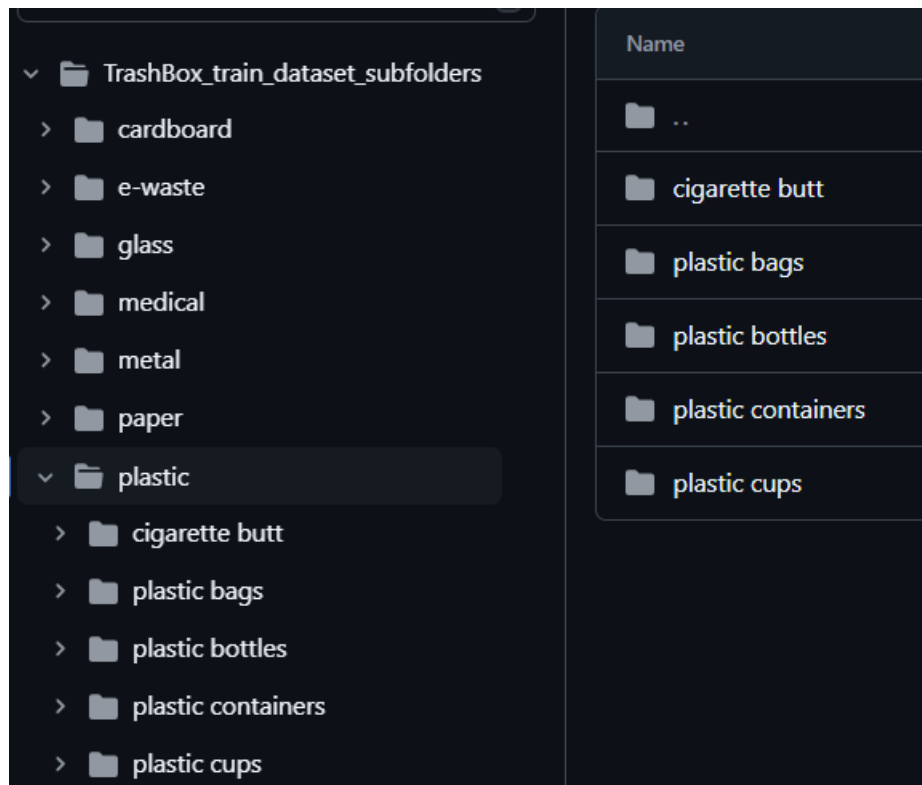


Рисунок 2 - Категории датасета

2.2.1 Адаптация для формата YOLO

- Исходный формат меток датасета был преобразован в формат, подходящий для YOLO, с использованием программы [LabelImg](#) (рисунок 3).

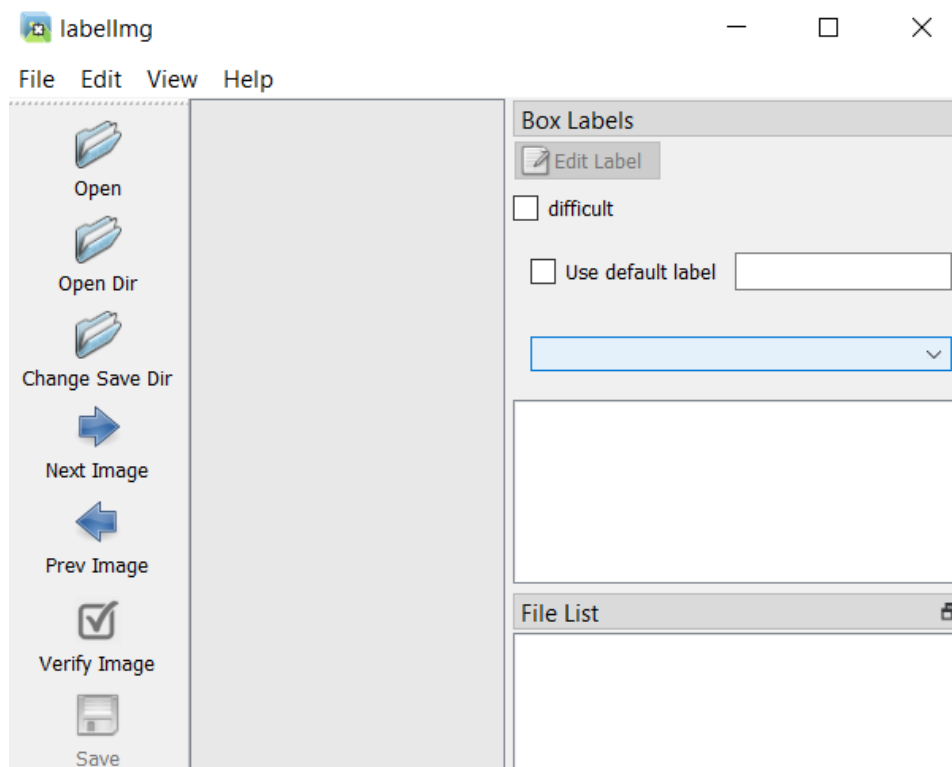


Рисунок 3 - Интерфейс программы

- LabelImg — это удобный инструмент для разметки изображений, который позволяет вручную отмечать объекты на изображениях и сохранять метки в формате YOLO.

2.2.2 Редактирование изображений

- Было вручную обработано (рисунок 4) около 150 изображений для уточнения меток. Это включало проверку существующих аннотаций, добавление недостающих меток и исправление ошибок.

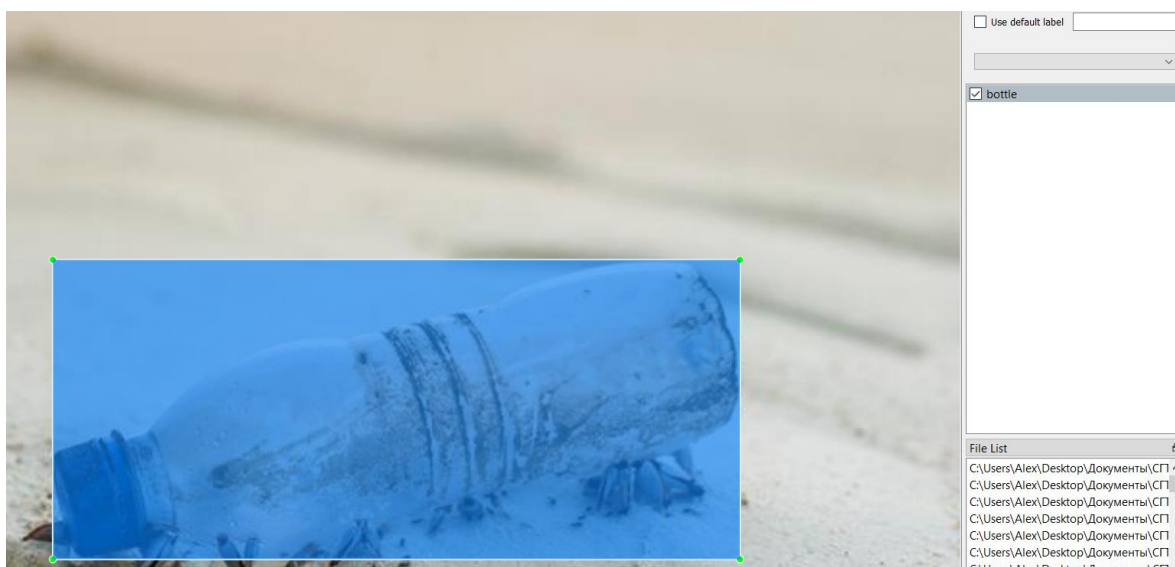


Рисунок 4 - Создание разметки

2.2.3 Разделение датасета

- Для обучения модели датасет был разделён на три подмножества при помощи кода (рисунок 5):

```
#Utility function to move images
def move_files_to_folder(list_of_files, destination_folder):
    for f in list_of_files:
        try:
            shutil.move(f, destination_folder)
        except:
            print(f)
            assert False

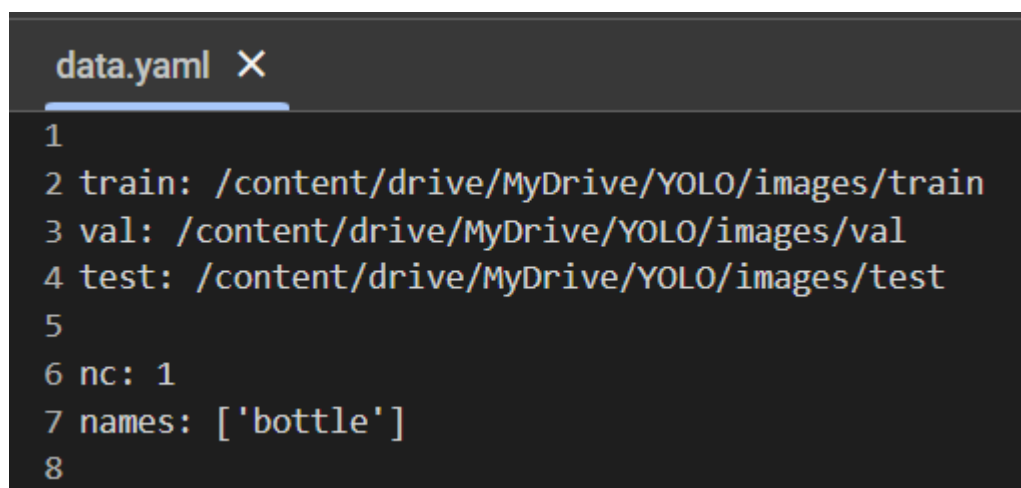
# Move the splits into their folders
move_files_to_folder(train_images, 'images/train')
move_files_to_folder(val_images, 'images/val/')
move_files_to_folder(test_images, 'images/test/')
move_files_to_folder(train_annotations, 'annotations/train/')
move_files_to_folder(val_annotations, 'annotations/val/')
move_files_to_folder(test_annotations, 'annotations/test/')
```

Рисунок 5 - Функция разделения изображений

- **Обучающая выборка (train):** 80% изображений. Используется для обучения модели.
- **Валидационная выборка (val):** 10% изображений. Используется для проверки качества модели во время обучения.
- **Тестовая выборка (test):** 10% изображений. Используется для финальной оценки производительности модели.

2.2.4 Создание файла конфигурации

Для корректной работы модели YOLO был создан файл data.yaml файла (рисунок 6), который включает информацию о путях к обучающим и тестовым данным, а также перечень классов объектов, доступных для детекции.



```
data.yaml X
1
2 train: /content/drive/MyDrive/YOLO/images/train
3 val: /content/drive/MyDrive/YOLO/images/val
4 test: /content/drive/MyDrive/YOLO/images/test
5
6 nc: 1
7 names: ['bottle']
8
```

Рисунок 6 – Файл конфигурации

2.3 Обучение

2.3.2 Библиотека YOLO

Для обучения модели YOLOv11 в данном проекте применялась библиотека Ultralytics YOLO. Эта библиотека предоставляет удобные инструменты для работы с различными версиями YOLO, включая подготовку данных, настройку гиперпараметров и процесс обучения.

2.3.1 Настройки обучения

Обучение модели (рисунок 7) проводилось на облачной платформе Google Colaboratory, что обеспечило доступ к графическим процессорам (GPU), необходимых для ускорения вычислений.

```

#Обучение. 50 эпох заняло примерно 4 часа
from ultralytics import YOLO
import os

# Путь к файлу данных
data_path = '/content/drive/MyDrive/YOLO/data.yaml'

# Параметры тренировки
n_epochs = 50
verbose = False
rng = 0
validate = True

# Указываем директорию для сохранения результатов в Google Drive
save_dir = '/content/drive/MyDrive/YOLO/runs'
os.makedirs(save_dir, exist_ok=True)

# Загрузка модели
model = YOLO('yolo11s.pt')

# Тренировка
results = model.train(
    data=data_path,
    epochs=n_epochs,
    lr0=0.001,
    verbose=verbose,
    seed=rng,
    val=validate,
    project=save_dir, # Указываем путь для сохранения результатов

```

Рисунок 7 - Код для обучения модели

- **Количество эпох:** Модель была обучена в течение 50 эпох, что позволило достичь стабилизации метрик производительности.
- **Начальная скорость обучения:** Гиперпараметр `lr0` был установлен на уровне 0.001. Такой выбор начальной скорости обучения помог модели эффективно находить минимум функции ошибки.
- **Время обучения:** Обучение заняло приблизительно 4 часа.

2.3.2 Процесс обучения

- **Загрузка данных:** на этапе обучения использовались данные, разделённые на тренировочные и валидационные выборки (описаны в предыдущей главе).
- **Настройка гиперпараметров:** В дополнение к скорости обучения, для обучения были использованы следующие параметры:
 - **Размер пакета данных (batch size):** 16.
 - **Функция потерь:** комбинация для предсказания координат объектов, их классов и оценки перекрытия рамок.
- **Мониторинг метрик:** В процессе обучения отслеживались ключевые метрики (рисунок 8):

```
Ultralytics 8.3.49 Python-3.10.12 torch-2.5.1+cu121 CPU (Intel Xeon 2.20GHz)
YOLO11s summary (fused): 238 layers, 9,413,187 parameters, 0 gradients, 21.3 GFLOPs
val: Scanning /content/drive/MyDrive/YOLO/labels/val.cache... 15 images, 0 backgrounds, 0
      Class      Images  Instances   Box(P          R      mAP50  mAP50-95)
        all         15           43        1      0.905      0.981      0.863
Speed: 2.5ms preprocess, 582.7ms inference, 0.0ms loss, 0.6ms postprocess per image
Results saved to runs/detect/val
0.8629675612060715
```

Рисунок 8 - Итоговая метрика модели

- **Loss (потери):** ошибка, которую модель минимизирует на каждом этапе.
- **mAP (mean Average Precision):** показатель точности распознавания объектов.
- **Recall и Precision:** метрики полноты и точности модели.

2.3.3 Возможности дообучения

Одним из преимуществ использования библиотеки Ultralytics YOLO является возможность дообучения модели (fine-tuning). Это позволяет улучшить производительность модели за счёт добавления новых данных или увеличения количества эпох (рисунок 9).

```

# Дообучение предыдущей модели(по необходимости)
from ultralytics import YOLO

# Load the model with the previously trained weights
model = YOLO('/content/drive/MyDrive/YOLO/runs/train/weights/last.pt')

# Train the model with custom dataset
model.train(
    data="/content/drive/MyDrive/YOLO/data.yaml",
    epochs=50,
    imgsz=640,
    batch=64,
    workers=0,
    project="/content/drive/MyDrive/YOLO",
    name="results"
)

```

Рисунок 9 - Код для дообучения модели

Для дообучения можно:

- Добавить новые изображения в тренировочную выборку.
- Изменить гиперпараметры, например, снизить начальную скорость обучения для более точной настройки модели.

2.4 Результаты

После завершения обучения модель была протестирована на тестовом наборе данных (рисунок 10), состоящем из 15 изображений. Для оценки производительности были использованы метрики точности (Precision), полноты (Recall), средневзвешенной точности (mAP) при различных порогах (mAP0.5 и mAP0.5:0.95).

```

from ultralytics import YOLO

# Загрузка обученной модели
model = YOLO('/content/drive/MyDrive/YOLO/runs/train/weights/best.pt')

# Оценка на тестовом наборе
metrics = model.val(data='/content/drive/MyDrive/YOLO/data.yaml', split='test')
print(metrics)

```

Рисунок 10 - Код для оценки на тестовом наборе

Основные результаты (рисунок 11):

```
val: Scanning /content/drive/MyDrive/YOLO/labels/test... 15 images, 0 backgrounds, 0 corr
      Class      Images  Instances   Box(P       R    mAP50  mAP50-95):
      all         15         22    0.872    0.955    0.959    0.844
Speed: 7.1ms preprocess, 790.9ms inference, 0.0ms loss, 3.1ms postprocess per image
```

Рисунок 11 - Результаты оценки

- Точность (Precision): 0.872
- Полнота (Recall): 0.955
- Средневзвешенная точность (mAP@0.5): 0.959
- Средневзвешенная точность при разных порогах (mAP@0.5:0.95): 0.844

Эти значения показывают высокую способность модели определять пластиковые бутылки на изображениях. Значения Precision и Recall свидетельствуют о том, что модель минимизировала как ложные срабатывания, так и пропуски объектов. Высокий показатель mAP подтверждает качественное распознавание объектов на тестовых изображениях.

В результате обучения нейронной сети YOLOv11 на подготовленном датасете была достигнута способность модели распознавать пластиковые бутылки на изображениях. На рисунке 11 ниже приведены примеры распознавания объектов на тестовом изображении.



Рисунок 12 - Распознавание бутылок

Заключение

В ходе выполнения работы была успешно решена задача обучения нейронной сети YOLOv11 для распознавания пластиковых бутылок на изображениях. Процесс включал все этапы разработки: от подготовки данных до получения и анализа результатов.

Для обучения был использован датасет, переработанный под формат YOLO с помощью инструмента LabelImg. Подход включал ручную разметку около 150 изображений, разделение данных на тренировочную, валидационную и тестовую выборки (в пропорции 80:10:10), а также создание конфигурационного файла data.yaml. Модель обучалась на платформе Google Colaboratory, используя мощные вычислительные ресурсы GPU, что позволило провести 50 эпох за 4 часа.

Результаты тестирования модели на тестовом наборе данных показали высокую эффективность работы. Средневзвешенная точность (mAP@0.5) составила 0.959, а mAP@0.5:0.95 достигла значения 0.844. Такие показатели подтверждают способность модели не только точно локализовать пластиковые бутылки на изображениях, но и правильно классифицировать их даже в сложных условиях, таких как низкая освещённость или наличие перекрывающихся объектов.

Данная работа подтверждает, что нейронные сети, такие как YOLOv11, являются мощным инструментом для решения задач компьютерного зрения. Полученные результаты и подходы могут быть использованы для дальнейшего развития проектов, направленных на автоматизацию процессов сортировки отходов, мониторинг окружающей среды и другие прикладные задачи в области устойчивого развития. Возможности дообучения модели и расширения тренировочного набора данных открывают перспективы для повышения её точности и адаптации к новым задачам.

Список литературы

1. Документация по модели YOLO11 [Электронный ресурс]. — URL: <https://docs.ultralytics.com/ru/models/yolo11/> (дата обращения: 08.11.2024).
2. TrashBox Dataset [Электронный ресурс]. — URL: <https://github.com/nikhilvenkatkumsetty/TrashBox> (дата обращения: 25.11.2024).
3. Введение в YOLO: как работает один из самых быстрых алгоритмов распознавания объектов [Электронный ресурс] // Хабр. — URL: <https://habr.com/ru/articles/709432/> (дата обращения: 02.12.2024).
4. YOLO: You Only Look Once [Электронный ресурс] // pjreddie.com. — URL: <https://pjreddie.com/darknet/yolo/> (дата обращения: 15.11.2024).
5. Обзор YOLO11: Улучшения и особенности [Электронный ресурс] // Viso.ai. — URL: <https://viso.ai/computer-vision/yolov11/> (дата обращения: 06.12.2024).