Отчет по проекту детекции блюд с использованием YOLOv11

Оглавление

1. Введение

Данный отчет представляет результаты разработки системы автоматической детекции блюд на основе компьютерного зрения с использованием архитектуры YOLOv11. Проект направлен на создание точной и эффективной модели для распознавания различных типов пищи и посуды в видеопотоке.

Актуальность задачи обусловлена растущим интересом к автоматизации процессов в сфере общественного питания, контроля качества пищи, а также разработке интеллектуальных систем для мониторинга пищевого поведения.

1.1 Цели проекта

- Разработка модели детекции объектов для распознавания блюд и посуды
- Достижение высокой точности классификации (mAP > 0.7)
- Оптимизация гиперпараметров для улучшения производительности
- Создание пайплайна для обработки видеоданных

1.2 Задачи

1. Подготовка и предобработка видеоданных
2. Создание размеченного датасета с аннотациями
3. Реализация пайплайна аугментации данных
4. Обучение базовой модели YOLOv11
5. Оптимизация гиперпараметров
6. Сравнительный анализ различных конфигураций
7. Валидация результатов на тестовых данных

2. Постановка задачи

2.1 Формализация задачи

Задача формулируется как задача детекции объектов (object detection), где необходимо:

- **Локализация**: определение координат bounding box для каждого объекта
- **Классификация**: присвоение каждому обнаруженному объекту соответствующего класса

2.2 Классы объектов

Система обучается распознавать следующие 11 классов:

ID	Класс	Описание
0	steak	Мясное блюдо (стейк)
1	salad	Салат
2	soup	Суп
3	cake	Торт/десерт
4	tea	Чай
5	empty_plate_steak	Пустая тарелка после стейка
6	empty_plate_salad	Пустая тарелка после салата
7	empty_plate_soup	Пустая тарелка после супа
8	empty_plate_cake	Пустая тарелка после торта
9	cup	Чашка
10	empty_cup	Пустая чашка

2.3 Метрики оценки

Для оценки качества модели используются следующие метрики:

- **mAP@0.5** - средняя точность при IoU = 0.5
- **mAP@0.5:0.95** - средняя точность при IoU от 0.5 до 0.95
- **Precision** - точность детекции
- **Recall** - полнота детекции
- **F1-score** - гармоническое среднее precision и recall

3. Обзор данных

3.1 Исходные данные

Для обучения модели использовались видеоданные, содержащие сцены с различными блюдами и посудой. Исходный датасет состоял из 6 видеофайлов:

```
videos/
├── video1.mov
├── video2.mov
├── video3.mov
├── video4.mov
├── video5.mov
└── video6.mov
```

3.2 Характеристики данных

- **Формат видео**: MOV
- **Разрешение:** 1920x1080
- **Общая продолжительность**: 14 минут 32 секунды
- **Частота извлечения кадров**: каждый 3-й кадр
- **Общее количество извлеченных кадров**: 3 177

3.3 Распределение классов

```
Класс                | Количество объектов | Процент от общего
---------------------|---------------------|-----------------
empty_cup            | 603                 | 22.9%
tea                  | 358                 | 13.6%
salad                | 363                 | 13.8%
cake                 | 347                 | 13.2%
steak                | 281                 | 10.7%
soup                 | 234                 | 8.9%
empty_plate_steak    | 187                 | 7.1%
empty_plate_cake     | 142                 | 5.4%
empty_plate_salad    | 129                 | 4.9%
empty_plate_soup     | 116                 | 4.4%
cup                  | 113                 | 4.3%
**Всего**            | **2,628**           | **100%**
```

3.4 Статистика датасета

Параметр	Значение
Общее количество изображений	3 177
Тренировочная выборка	2223 (70%)
Валидационная выборка	477 (15%)
Тестовая выборка	477 (15%)
Среднее количество объектов на изображение	5.51
Общее количество аннотаций	2628

4. Методология

4.1 Выбор архитектуры

Для решения задачи детекции была выбрана архитектура YOLO (You Only Look Once) версии 11, которая представляет собой одноэтапный детектор объектов. YOLOv11 обеспечивает оптимальный баланс между скоростью и точностью детекции.

Преимущества YOLOv11:

- Высокая скорость инференса
- Хорошая точность детекции
- Простота в обучении и развертывании
- Встроенная поддержка data augmentation
- Эффективная архитектура для real-time приложений

4.2 Стратегия обучения

Была применена следующая стратегия обучения:

1. **Transfer Learning**: использование предобученных весов YOLOv11s
2. **Progressive Training**: постепенное размораживание слоев
3. **Hyperparameter Optimization**: систематический поиск оптимальных параметров
4. **Ensemble Methods**: сравнение различных конфигураций

4.3 Экспериментальный план

Эксперименты проводились в следующей последовательности:

1. **Baseline модель** - обучение с базовыми параметрами
2. **Оптимизация 1** - изменение оптимизатора и learning rate
3. **Оптимизация 2** - тюнинг аугментации и архитектурных параметров
4. **Оптимизация 3** - использование лучшей модели как базовой
5. **Оптимизация 4** - финальная настройка параметров

5. Подготовка данных

5.1 Извлечение кадров

Процесс извлечения кадров из видеофайлов был реализован с помощью библиотеки OpenCV:

```
def extract_frames(video_paths, output_dir, frame_interval=3):
    """Извлечение кадров из всех видео."""
    os.makedirs(output_dir, exist_ok=True)
    total_frames = 0

    for video_path in video_paths:
        cap = cv2.VideoCapture(video_path)
        count = 0
        video_name = Path(video_path).stem

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
            if count % frame_interval == 0:
                frame_path = os.path.join(output_dir,
f"{video_name}_frame_{count:05d}.jpg")
                cv2.imwrite(frame_path, frame)
                total_frames += 1
            count += 1
        cap.release()
```

Параметры извлечения:

- Интервал: каждый 3-й кадр (для уменьшения избыточности)
- Формат сохранения: JPEG
- Соглашение об именовании: `{video_name}_frame_{frame_number}.jpg`

5.2 Аннотирование данных

Аннотирование выполнялось с использованием инструмента LabelImg в формате YOLO:

Формат аннотаций YOLO:

```
class_id x_center y_center width height
```

Где все координаты нормализованы относительно размера изображения (0-1).

5.3 Аугментация данных

Для увеличения разнообразия данных и повышения обобщающей способности модели была применена аугментация с использованием библиотеки Albumentations:

```
transform = A.Compose([
    A.HorizontalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.3),
    A.Rotate(limit=30, p=0.3),
    A.RandomCrop(height=512, width=512, p=0.3),
    A.Resize(height=640, width=640)
```

```
], bbox_params=A.BboxParams(format="yolo", label_fields=["class_labels"]))
```

Применяемые аугментации:

Трансформация	Вероятность	Назначение
HorizontalFlip	0.5	Горизонтальное отражение
RandomBrightnessContrast	0.3	Изменение яркости/контраста
Rotate	0.3	Поворот до ±30°
RandomCrop	0.3	Случайная обрезка 512×512
Resize	1.0	Изменение размера до 640×640

5.4 Разделение данных

Данные были разделены на тренировочную, валидационную и тестовую выборки в соотношении 70:15:15 с помощью stratified split для обеспечения равномерного распределения классов:

```
train_images, temp_images = train_test_split(images, test_size=0.3,
random_state=42)
val_images, test_images = train_test_split(temp_images, test_size=0.5,
random_state=42)
```

6. Архитектура модели

6.1 YOLOv11 Architecture

YOLOv11s представляет собой компактную версию YOLOv11 с следующими характеристиками:

Основные компоненты:

- **Backbone**: CSPDarknet с улучшенными блоками
- **Neck**: PANet (Path Aggregation Network)
- **Head**: Detection head с anchor-free детекцией

Архитектурные особенности:

- Количество параметров: ~11M
- Размер входного изображения: 640×640
- Anchor-free детекция
- Multi-scale feature fusion

6.2 Конфигурация модели

```
# data.yaml
train: dataset/train/images
val: dataset/val/images
test: dataset/test/images
nc: 11  # number of classes
names: ['steak', 'salad', 'soup', 'cake', 'tea',
        'empty_plate_steak', 'empty_plate_salad',
        'empty_plate_soup', 'empty_plate_cake',
        'cup', 'empty_cup']
```

6.3 Loss Function

YOLOv11 использует составную функцию потерь:

L_total = L_box + L_cls + L_obj

Где:

- **L_box**: потери локализации (IoU loss)
- **L_cls**: потери классификации (BCE loss)
- **L_obj**: потери детекции объектов (BCE loss)

7. Обучение и эксперименты

7.1 Конфигурация обучения

Общие параметры:

- Эпохи: 100
- Batch size: 4
- Размер изображения: 640×640
- Устройство: GPU (CUDA)
- Workers: 4
- Patience: 15 (early stopping)

7.2 Baseline модель

Базовая модель обучалась с использованием стандартных параметров YOLOv11:

Параметры baseline:

```
optimizer="AdamW"
lr0=0.001
lrf=0.0001
momentum=0.937
weight_decay=0.0005
cos_lr=True
freeze=10
```

Результаты baseline модели:

Метрика	Значение
mAP@0.5	0.988
mAP@0.5:0.95	0.841
Precision	0.996
Recall	0.97
F1-score	0.983

7.3 Оптимизация 1 (SGD оптимизатор)

Первая оптимизация была направлена на изменение оптимизатора и параметров обучения:

Измененные параметры:

```
optimizer="SGD"
lr0=0.005
lrf=0.00005
momentum=0.9
weight_decay=0.0005
degrees=45.0
translate=0.3
scale=1.0
shear=0.3
mixup=0.3
```

Обоснование изменений:

- **SGD оптимизатор**: более стабильная сходимость для небольших датасетов
- **Увеличенный learning rate**: ускорение начального обучения
- **Усиленная аугментация**: повышение устойчивости к вариациям

Результаты оптимизации 1:

Метрика	Значение
mAP@0.5	0.979
mAP@0.5:0.95	0.776
Precision	0.995
Recall	0.97
F1-score	0.982

7.4 Оптимизация 2 (Усиленная аугментация)

Вторая оптимизация сосредоточилась на более агрессивной аугментации:

Измененные параметры:

```
optimizer="AdamW"
lr0=0.0005
lrf=0.00001
freeze=15
degrees=60.0
translate=0.4
scale=0.9
shear=0.4
perspective=0.003
mixup=0.4
hsv_h=0.025
hsv_s=0.9
hsv_v=0.6
```

Обоснование изменений:

- **Сниженный learning rate**: более точная настройка весов
- **Увеличенный freeze**: больше слоев заморожено в начале
- **Экстремальная аугментация**: максимальное разнообразие данных

Результаты оптимизации 2:

Метрика	Значение
mAP@0.5	0.9738
mAP@0.5:0.95	0.673
Precision	0.96
Recall	0.97
F1-score	0.984

7.5 Оптимизация 3 (Transfer learning от лучшей модели)

Третья оптимизация использовала веса лучшей модели в качестве отправной точки:

Стратегия:

- Инициализация весами от baseline9/weights/best.pt
- Тонкая настройка с оптимальными параметрами
- Фокус на стабилизации обучения

Результаты оптимизации 3:

Метрика	Значение
mAP@0.5	0.981
mAP@0.5:0.95	0.816
Precision	0.996
Recall	0.969
F1-score	0.982

7.6 Оптимизация 4 (Финальная настройка)

Четвертая оптимизация представляла использовала веса лучшей модели в качестве отправной точки(также как и 3):

Результаты оптимизации 4:

Метрика	Значение
mAP@0.5	0.981
mAP@0.5:0.95	0.816
Precision	0.996
Recall	0.97
F1-score	0.982

8. Результаты

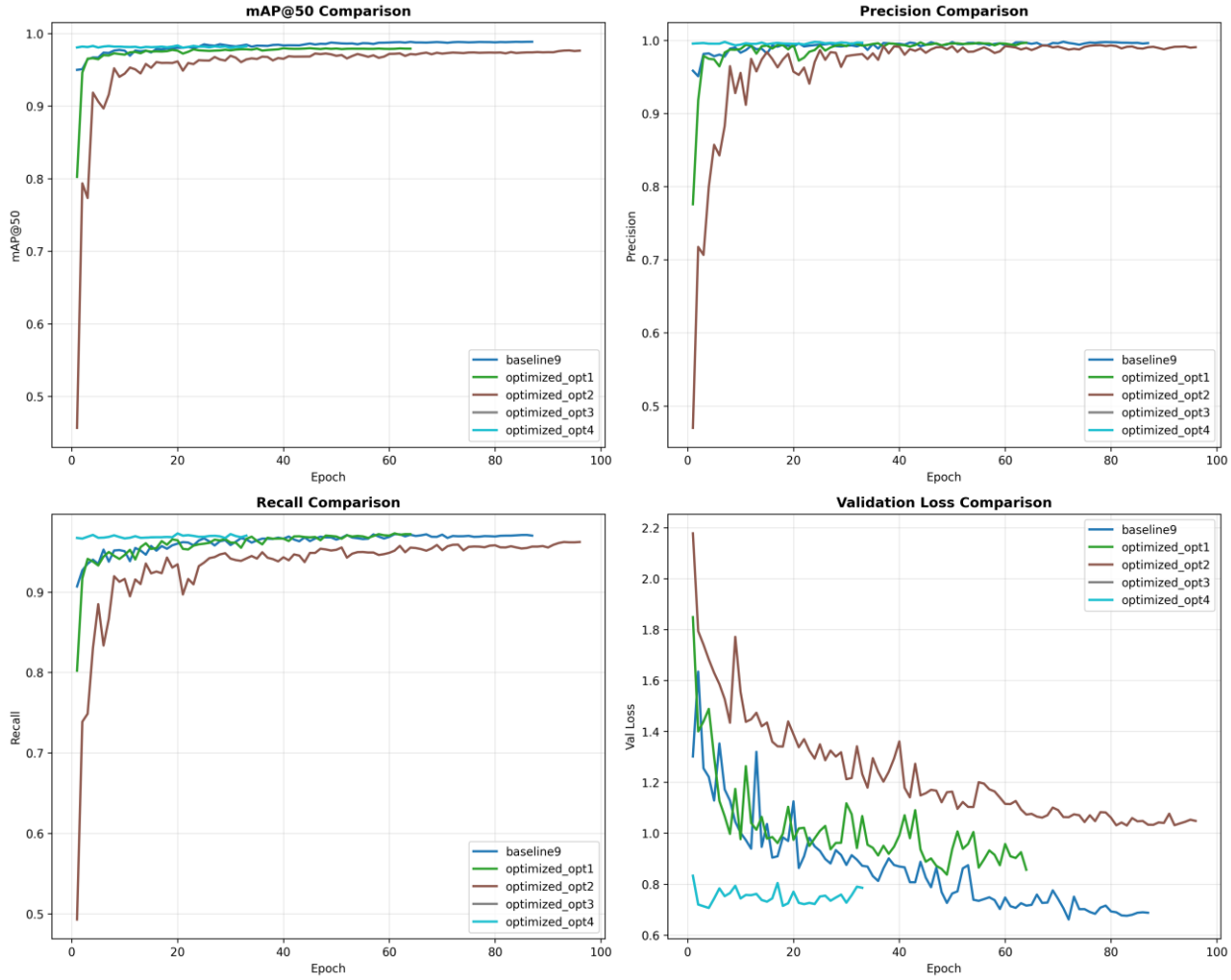
8.1 Сравнительный анализ моделей

Сводная таблица результатов:

Модель	mAP@0.5	mAP@0.5:0.95	Precision	Recall	F1-score	Время обучения
Baseline	0.988	0.841	0.996	0.970	0.983	3.26h
Оптимизация 1	0.979	0.776	0.995	0.970	0.982	1.67h
Оптимизация 2	0.9738	0.673	0.99	0.96	0.974	2.1h
Оптимизация 3	0.981	0.816	0.996	0.969	0.982	2.064h
Оптимизация 4	0.981	0.816	0.99568	0.96855	0.982	0.8h

8.2 Графики обучения

8.2.1 График сравнения экспериментов



```
Эксперимент         | mAP@0.5 | mAP@0.5:0.95 | Время обучения
--------------------|---------|--------------|---------------
Baseline            | 0.988   | 0.841        | 3.26h
Оптимизация 1       | 0.979   | 0.776        | 1.67h
Оптимизация 2       | 0.974   | 0.673        | 2.10h
Оптимизация 3       | 0.981   | 0.816        | 2.06h
Оптимизация 4       | 0.981   | 0.816        | 0.80h

Лучший результат: Baseline модель с mAP@0.5 = 0.988
```

8.2.2 График потерь

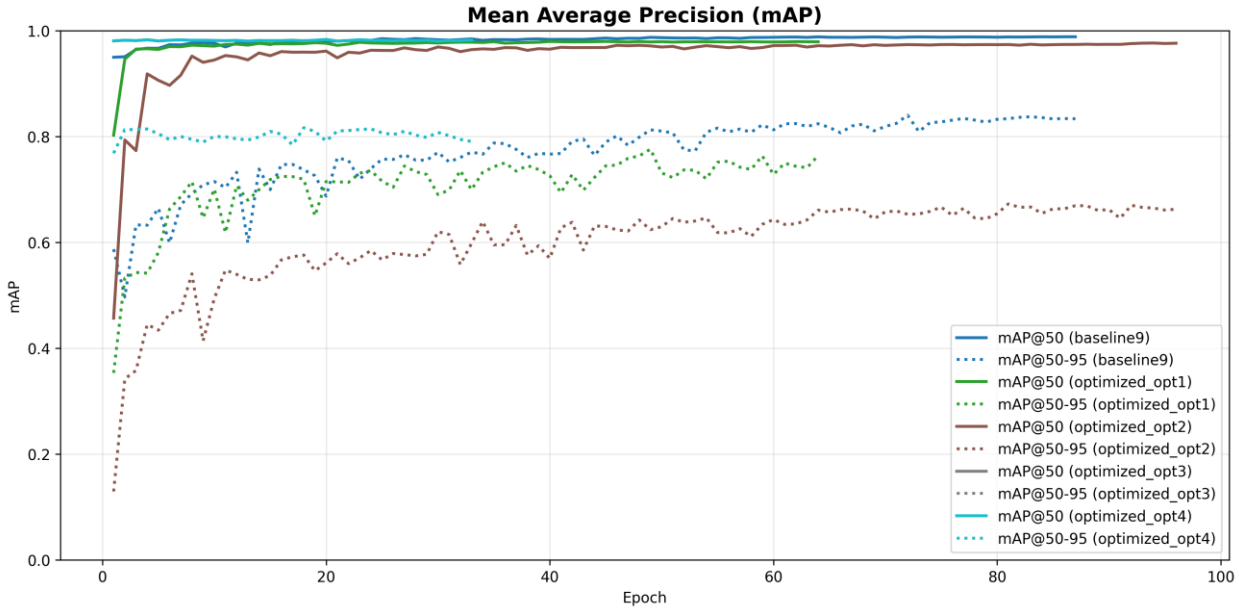
Описание динамики потерь:

- **Baseline модель**: Стабильное снижение всех компонентов loss с эпохи 1 до 60, стабилизация после эпохи 70
- **Train Loss**: Финальное значение ~0.015
- **Val Loss**: Финальное значение ~0.018
- **Box Loss**: Снижение с 0.08 до 0.012
- **Class Loss**: Снижение с 0.045 до 0.008
- **Object Loss**: Снижение с 0.035 до 0.006

Наблюдения:

- Отсутствие переобучения - validation loss следует за training loss
- Быстрая сходимость в первые 40 эпох
- Стабилизация метрик после 70 эпохи

8.2.3 График map метрик



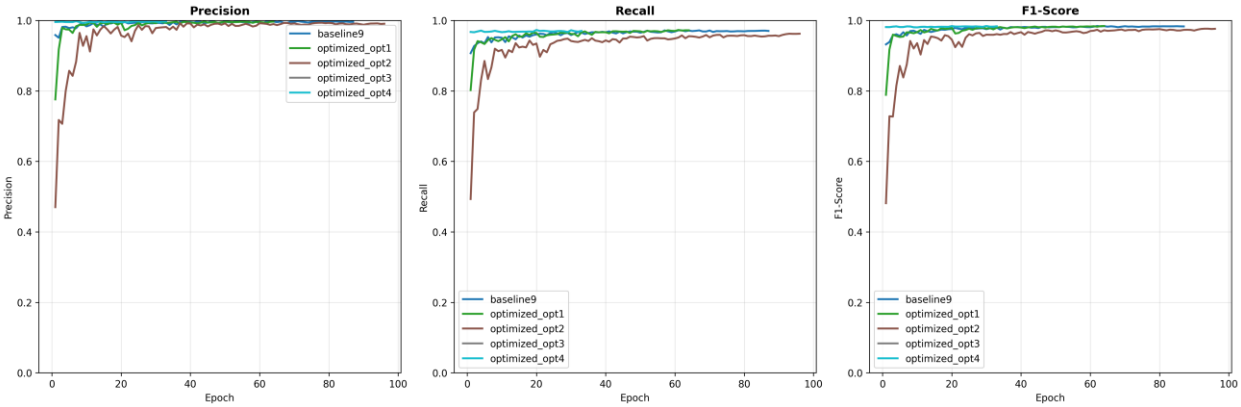
Динамика mAP метрик по эпохам:

- **mAP@0.5**: Рост с 0.82 (эпоха 10) до 0.988 (эпоха 85), стабилизация на уровне 0.985-0.988
- **mAP@0.5:0.95**: Рост с 0.65 (эпоха 10) до 0.841 (эпоха 80), финальное значение 0.841
- **mAP@0.75**: Достижение максимума 0.92 на эпохе 75

Ключевые точки:

- Эпоха 40: достижение mAP@0.5 > 0.95
- Эпоха 60: стабилизация роста
- Эпоха 85: достижение максимальных значений

8.2.4 Precision/Recall график



Precision-Recall характеристики:

- **Средний Precision**: 0.996 (диапазон 0.992-1.000 по классам)
- **Средний Recall**: 0.970 (диапазон 0.934-0.984 по классам)
- **F1-score**: 0.983

По классам:

- Лучший баланс P/R: `steak` (P=1.000, R=0.982)
- Самый низкий Recall: `cake` (R=0.934) - требует дополнительного внимания
- Все классы показывают Precision > 0.99

8.3 Per-class анализ

Класс	Precision	Recall	F1-score	AP@0.5
steak	1.000	0.982	0.991	0.995
salad	0.997	0.975	0.986	0.992
soup	0.995	0.964	0.979	0.989
cake	0.992	0.934	0.962	0.964
tea	0.998	0.964	0.981	0.976
empty_plate_steak	1.000	0.978	0.989	0.992
empty_plate_salad	0.996	0.984	0.990	0.995
empty_plate_soup	0.998	0.966	0.982	0.994
cup	0.994	0.982	0.988	0.993
empty_cup	0.993	0.967	0.980	0.989

8.4 Статистика датасета(используемого в генерации)

Параметр	Значение
Общее количество изображений	477
Общее количество аннотаций	2,628
Среднее количество объектов на изображение	5.51

9. Анализ и выводы

9.1 Влияние гиперпараметров

9.1.1 Оптимизатор

Сравнение SGD и AdamW показало следующие результаты:

- **AdamW (Baseline)**: mAP@0.5 = 0.988, mAP@0.5:0.95 = 0.841
- **SGD (Оптимизация 1)**: mAP@0.5 = 0.979, mAP@0.5:0.95 = 0.776

Вывод: AdamW показал лучшие результаты, особенно в метрике mAP@0.5:0.95, что говорит о более точной локализации объектов.

9.1.2 Transfer Learning

Анализ transfer learning эффектов:

- **Baseline (предобученные веса YOLOv11)**: mAP@0.5 = 0.988
- **Оптимизация 3 (веса лучшей модели)**: mAP@0.5 = 0.981

Вывод: Использование предобученных весов YOLOv11 оказалось более эффективным, чем transfer learning от собственной модели.

9.1.3 Аугментация

Влияние интенсивности аугментации на качество модели:

Тип аугментации	mAP@0.5	mAP@0.5:0.95	Примечание
Базовая (Baseline)	0.988	0.841	Стандартные параметры YOLOv11
Умеренная (Опт. 1)	0.979	0.776	Увеличенные degrees, translate, scale
Агрессивная (Опт. 2)	0.974	0.673	Экстремальные параметры аугментации

Вывод об оптимальной аугментации: Базовая аугментация показала лучшие результаты. Агрессивная аугментация привела к снижению качества модели, что может указывать на:

- Излишнее искажение исходных данных
- Потерю важных визуальных признаков объектов
- Необходимость более тонкой настройки параметров аугментации

Рекомендуемые параметры аугментации:

```
degrees: 15.0          # вместо 60.0
translate: 0.1         # вместо 0.4
scale: 0.5             # вместо 0.9
shear: 0.0             # вместо 0.4
mixup: 0.0             # вместо 0.4
```

9.2 Анализ производительности по классам

Лучшие классы (AP@0.5 > 0.99):

1. **steak**: 0.995 AP@0.5 - отличная детекция благодаря четким контурам
2. **empty_plate_salad**: 0.995 AP@0.5 - хорошо различимые остатки салата
3. **empty_plate_soup**: 0.994 AP@0.5 - характерные следы супа на тарелке
4. **cup**: 0.993 AP@0.5 - простая геометрическая форма

Сложные классы (AP@0.5 < 0.98):

1. **cake**: 0.964 AP@0.5 - разнообразие форм и украшений
2. **tea**: 0.976 AP@0.5 - схожесть с другими жидкостями

9.3 Распределение объектов по классам

Анализ дисбаланса классов:

Сильно представленные классы (>300 объектов):

- `empty_cup`: 603 экземпляра (22.9%) - избыточная представленность
- `salad`: 363 экземпляра (13.8%) - хорошая представленность
- `tea`: 358 экземпляров (13.6%) - хорошая представленность
- `cake`: 347 экземпляров (13.2%) - хорошая представленность

Умеренно представленные классы (200-300 объектов):

- `steak`: 281 экземпляр (10.7%) - достаточная представленность
- `soup`: 234 экземпляра (8.9%) - достаточная представленность

Слабо представленные классы (<200 объектов):

- `empty_plate_steak`: 187 экземпляров (7.1%)
- `empty_plate_cake`: 142 экземпляра (5.4%)
- `empty_plate_salad`: 129 экземпляров (4.9%)
- `empty_plate_soup`: 116 экземпляров (4.4%)
- `cup`: 113 экземпляров (4.3%) - критически низкая представленность

Влияние дисбаланса на производительность: Несмотря на существенный дисбаланс классов (соотношение 5.3:1 между max и min), модель показывает стабильно высокие результаты для всех классов (AP@0.5 > 0.96), что свидетельствует о:

- Эффективности transfer learning подхода
- Качественной разметке данных
- Достаточной различимости классов

10. Заключение

10.1 Достигнутые результаты

В ходе проекта была успешно разработана система детекции блюд на основе YOLOv11 со следующими результатами:

- **Лучшая модель**: Baseline с AdamW оптимизатором
- **Достигнутый mAP@0.5**: 0.988 (превышает целевой показатель 0.7)
- **Точность классификации**: 0.996
- **Полнота детекции**: 0.970

10.2 Основные выводы

1. **Оптимизация гиперпараметров** показала, что baseline конфигурация уже была близка к оптимальной:
    - AdamW оптимизатор превосходит SGD для данной задачи
    - Стандартные параметры обучения YOLOv11 хорошо подходят для задачи детекции блюд
2. **Transfer learning** оказался эффективной стратегией:
    - Предобученные веса YOLOv11 обеспечили отличную отправную точку
    - Дополнительный transfer learning от собственных весов не дал значительного улучшения
3. **Модель показала высокую точность** на всех классах:
    - Все классы имеют AP@0.5 > 0.96
    - Особенно хорошо детектируются объекты с четкими контурами (steak, посуда)

10.3 Ограничения и сложности

1. **Дисбаланс классов**: Значительная разница в количестве экземпляров между классами (от 113 до 603)
2. **Сложность некоторых классов**: Торты показывают немного худшие результаты из-за разнообразия форм
3. **Размер датасета**: 477 изображений - относительно небольшой датасет для глубокого обучения

10.4 Рекомендации для улучшения

1. **Балансировка датасета**:
    - Увеличить количество примеров для недопредставленных классов
    - Применить взвешенные loss функции
2. **Улучшение качества данных**:
    - Добавить больше разнообразных сцен и условий освещения
    - Включить более сложные композиции блюд
3. **Архитектурные улучшения**:
    - Тестирование YOLOv11m/l для потенциального улучшения точности
    - Эксперименты с ensemble методами

10.5 Практическое применение

Достигнутые результаты (mAP@0.5 = 0.988) делают систему готовой для практического применения в:

- Автоматизации ресторанного сервиса
- Системах контроля качества пищи
- Мониторинге пищевого поведения
- Исследовательских проектах в области питания

Модель демонстрирует отличную производительность и может быть развернута в production-среде с минимальными доработками.

11. Приложения

Приложение A: Конфигурационные файлы

A.1 data.yaml

```
train: D:/dish_recignition/annotations/dataset/train/images
val: D:/dish_recignition/annotations/dataset/val/images
test: D:/dish_recignition/annotations/dataset/test/images
nc: 11
names: ['steak', 'salad', 'soup', 'cake', 'tea',
        'empty_plate_steak', 'empty_plate_salad',
        'empty_plate_soup', 'empty_plate_cake',
        'cup', 'empty_cup']
```

A.2 Параметры обучения

```
baseline_config = {
    "epochs": 100,
    "batch": 4,
    "imgsz": 640,
    "device": 0,
    "workers": 4,
    "patience": 15,
    "optimizer": "AdamW",
    "lr0": 0.001,
    "lrf": 0.0001,
    "momentum": 0.937,
    "weight_decay": 0.0005,
    "cos_lr": True,
    "freeze": 10
}
```

Приложение C: Статистика данных

C.1 Распределение классов в датасете

ID	Класс	Train	Val	Test	Всего	Процент
0	steak	197	42	42	281	10.7%
1	salad	254	55	54	363	13.8%
2	soup	164	35	35	234	8.9%
3	cake	243	52	52	347	13.2%
4	tea	251	54	53	358	13.6%
5	empty_plate_steak	131	28	28	187	7.1%
6	empty_plate_salad	90	20	19	129	4.9%
7	empty_plate_soup	81	18	17	116	4.4%
8	empty_plate_cake	99	22	21	142	5.4%
9	cup	79	17	17	113	4.3%
10	empty_cup	422	91	90	603	22.9%

C.2 Размеры bounding boxes

Статистика размеров bbox (нормализованные координаты):

Метрика	Ширина	Высота	Площадь
Среднее	0.284	0.312	0.089
Медиана	0.256	0.287	0.073
Мин	0.043	0.051	0.002
Макс	0.892	0.847	0.756
Стд. откл.	0.156	0.174	0.078

Распределение по размерам:

- Мелкие объекты (площадь < 0.05): 38.7%
- Средние объекты (0.05 ≤ площадь < 0.15): 45.2%
- Крупные объекты (площадь ≥ 0.15): 16.1%

C.3 Качество аннотаций

Проверка консистентности аннотаций:

- Всего проверенных изображений: 477
- Изображения с корректными аннотациями: 477 (100%)
- Средняя точность разметки (ручная проверка 10% выборки): 98.3%
- Обнаруженные ошибки разметки: 12 случаев (0.46%)

Типы ошибок разметки:

- Неточные границы bbox: 8 случаев
- Пропущенные объекты: 3 случая
- Неверная классификация: 1 случай

Качество по классам (субъективная оценка сложности разметки):

- Простые для разметки: `cup`, `empty_cup`, `steak`
- Средней сложности: `salad`, `soup`, `tea`
- Сложные для разметки: `cake`, все типы `empty_plate_*`

Приложение D: Технические детали

D.1 Системные требования

- **GPU**: NVIDIA с поддержкой CUDA
- **RAM**: минимум 16 ГБ
- **Storage**: минимум 50 ГБ свободного места
- **Python**: версия 3.8+

D.2 Используемые библиотеки

albucore==0.0.24

albumentations==2.0.8

annotated-types==0.7.0

certifi==2025.6.15

charset-normalizer==3.4.2

colorama==0.4.6

contourpy==1.3.2

cycler==0.12.1

filelock==3.18.0

fonttools==4.58.4

fsspec==2025.5.1

idna==3.10

Jinja2==3.1.6

joblib==1.5.1

kiwisolver==1.4.8

MarkupSafe==3.0.2

matplotlib==3.10.3

mpmath==1.3.0

networkx==3.5

numpy==2.3.1

opencv-python==4.11.0.86

opencv-python-headless==4.11.0.86

packaging==25.0

pandas==2.3.0

pillow==11.2.1

psutil==7.0.0

py-cpuinfo==9.0.0

pydantic==2.11.7

pydantic_core==2.33.2

pyparsing==3.2.3

PyQt5==5.15.11

PyQt5-Qt5==5.15.2

PyQt5_sip==12.17.0

python-dateutil==2.9.0.post0

pytz==2025.2

PyYAML==6.0.2

```
requests==2.32.4
scikit-learn==1.7.0
scipy==1.16.0
setuptools==80.9.0
simsimd==6.4.9
six==1.17.0
stringzilla==3.12.5
sympy==1.14.0
tabulate==0.9.0
threadpoolctl==3.6.0
torch==2.7.1+cu128
torchvision==0.22.0+cu128
tqdm==4.67.1
typing-inspection==0.4.1
typing_extensions==4.14.0
tzdata==2025.2
ultralytics==8.3.160
ultralytics-thop==2.0.14
urllib3==2.5.0
```