Санкт-Петербургский государственный университет

Математическое обеспечение и администрирование информационных систем

Группа 22.М05-мм

Приложение учета складских запасов на примере трубной продукции

Соболь Дарья Валерьевна

Отчёт по учебной практике
в форме «Производственное задание»

Научный руководитель:
старший преподаватель, к.т.н., М.Н. Смирнов

Санкт-Петербург
2023

Оглавление

Введение

Для повышения экономической эффективности, современные производства требуют качественного управления всеми внутренними процессами предприятия. Качественное и эффективное управление подразумевает автоматизацию отдельных повторяющихся процессов. В современных реалиях бизнес- и производственные процессы могут быть автоматизированы с помощью различных информационных технологий и конкретных программ, что способствует увеличению производительности, снижению издержек и улучшению общей конкурентоспособности предприятия.

Подсчет количества труб в процессе их изготовления на трубопрокатных предприятиях не единственная задача, которая может быть автоматизирована с помощью технологий компьютерного зрения. Однако, на примере этой задачи можно проследить, как технологии компьютерного зрения проникают во все сферы жизни человека. Эти технологии позволяют автоматизировать как классические бизнес-процессы, так и уникальные процессы производственных предприятий. В задачах производства технологии машинного обучения, а конкретно компьютерного зрения, с каждым годом применяется все чаще.

Трубная продукция обычно поставляется в цех предприятия в виде пакетов (рис. 1), если это не одна труба большого диаметра. Количество труб в упаковках может варьироваться, от нескольких факторов, включая диаметр трубы и спецификации заказа, что делает процесс подсчета труб в упаковке более сложным и трудоемким. Данный процесс в настоящее время требует активного вмешательства человека. Когда новая партия труб поступает на склад готовой продукции, сотрудник склада готовой продукции вынужден проводить ручной пересчет труб и сверку количества продукции с заявленными данными. Такое ручной подсчет и управление этапами логистики имеет свои недостатки, например: возможность ошибок в подсчете из-за человеческого фактора, замедление процессов отгрузки готовой продукции с завода и получения готовой продукции клиентом. Современные технологии распознавания объек-

тов на изображении позволяют реализовать потенциал для автоматизации этого процесса с использованием нейронных сетей, такой подход значительно повышает точность подсчета количества продукции и упрощает управление запасами на складе готовой продукции. Таким образом, необходим способ подсчета труб быстрее и эффективнее, что позволило бы снизить процент ошибок и уменьшить участие человека в процессе подсчета.

Рис. 1: Доставка труб для подсчета.

1. Постановка задачи

Целью данной работы является создание приложения для цифрового замера объектов трубной промышленности. Для обоснованного выбора обучающей модели требуется:

1. Провести обзор алгоритмов выделения объектов на изображениях;
2. Проанализировать и сравнить существующие модели для распознавания объектов на изображении;
3. Изучить методы оценки качества моделей.

Для выполнения обучения модели требуется:

1. Выбрать и разметить данные;
2. Подобрать оптимальную методику обработки обучающей выборки;
3. Произвести обучение модели не менее, чем на 20 эпохах, или до момента сходимости, в зависимости от того, что наступит раньше.

По завершении обучения необходимо проанализировать качество и производительность полученной модели, сравнить её с имеющимися аналогами, подготовить руководство по использованию модели. Для интеграции в приложение требуется:

1. Разработка интерфейса пользователя и интеграция обученной модели в приложение для цифрового замера объектов трубной промышленности;
2. Оптимизация приложения для обеспечения быстрой обработки изображений, особенно если требуется реальное времени или обработка большого объема данных;

После завершения проектирования архитектуры и реализации приложения важно провести тестирование и апробацию на реальных данных, получить обратную связь от пользователей.

2. Обзор

Компьютерное зрение (CV) представляет собой быстро развивающуюся область искусственного интеллекта (ИИ), которая стремится расширить возможности компьютеров в понимании и интерпретации содержимого цифровых изображений и видео. Эта область науки и технологии нацелена на то, чтобы дать компьютерам способность "видеть" и осознавать визуальные данные, получаемые от камер и датчиков.

Системы компьютерного зрения используют мощные алгоритмы обработки изображений, которые позволяют компьютерам обнаруживать, классифицировать и анализировать объекты и их окружение на основе данных, полученных от камеры. Эти алгоритмы позволяют компьютерам распознавать образы, выделять особенности и структуры, а также извлекать ценную информацию из визуальных данных.

С каждым годом компьютерное зрение становится все более точным и мощным благодаря развитию глубокого обучения и нейронных сетей. Эти технологии позволяют компьютерам обучаться на больших объемах данных и выявлять сложные закономерности в изображениях, что приводит к значительному улучшению производительности систем CV[2].

Существует множество способов решения задач компьютерного зрения, и одним из наиболее эффективных является использование свёрточных нейронных сетей. Эти сети были специально разработаны для эффективного распознавания и обработки изображений. Они работают по принципу извлечения важных характеристик из изображения и последующей их обработки. На этой концепции основаны такие популярные алгоритмы, как YOLO и RCNN.

Хотя упомянутые алгоритмы, такие как YOLO и RCNN, обычно применяются для решения задачи распознавания нескольких объектов на изображении, использование свёрточной нейронной сети[8] в качестве основы метода решения задачи может привести к хорошим результатам. Даже если требуется распознать только один крупный объект на изображении, свёрточные нейронные сети могут быть полезны, по-

скольку они способны извлекать и анализировать важные особенности объекта, что поможет распознать его с высокой точностью[5].

2.1. Пороговые методы

Один из самых простых и эффективных способов решить задачу сегментации является использование порога как некоторого признака, который помогает разделить изображение на классы. Этот метод основан на определении некоторого порогового значения, которое позволяет разделить изображение на различные классы или регионы. При рассмотрении гистограмм изображений можно заметить, что все пиксели сгруппированы около нескольких основных центров. Используя пороговый метод, можно выделить эти группы, выбрав определенное значение порога T, и определить, какие точки на изображении принадлежат фону, а какие - объекту, на основе условия f(x, y) $>$ T. После этого можно, например, окрасить все точки объекта в черный цвет, а фона — в белый. Что позволит наглядно выделить объекты на изображении и облегчит их дальнейший анализ и обработку. Сегментация может быть произведена при помощи одного из двух подходов к пороговой сегментации: глобальный порог и адаптивный порог.

В первом случае выбор порога осуществляется один раз для всего изображения, после чего проходит по всем его точкам. Этот подход прост в реализации и подходит для изображений с однородной освещенностью и контрастом.

Адаптивный порог больше подходит для изображений с неравномерной освещенностью или изменяющимся контрастом. В данном методе пороговое значение выбирается индивидуально для каждой точки изображения, учитывая её окружение и контекст. Это позволяет более точно выделить объекты на изображении и улучшить качество сегментации[14].

2.2. Метод водоразделов

Алгоритм водоразделов (англ. watershed) является широко используемым методом сегментации изображений в компьютерном зрении и обработке изображений. Основная концепия водораздела состоит в том, что изображение рассматривается как топографическое рельеф, где интенсивности пикселей представляют высоту местности. Цель алгоритма - разделить изображение на регионы или сегменты на основе локальных минимумов и максимумов интенсивности изображения.

Алгоритм водоразделов начинает с рассмотрения каждого пикселя как отдельного водораздела или региона. Затем постепенно объединяются смежные водоразделы на основе градиентов интенсивности в изображении. Процесс объединения основан на концепции наводнения, где вода заливается в водоразделы и постепенно поднимается, пока не достигает границ между водоразделами. Эти границы, известные как линии водораздела, представляют границы сегментации между различными регионами на изображении.

Главным преимуществом алгоритма водоразделов является способность обрабатывать сложные структуры изображений, такие как перекрывающиеся или соприкасающиеся объекты. Он может точно сегментировать такие структуры, используя градиенты интенсивности и линии водораздела. Однако, алгоритм может быть чувствителен к шуму и может приводить к избыточной или недостаточной сегментации, если не контролируется должным образом[14].

2.3. Метод k-средних

Для сегментации можно использовать методы кластерного анализа, например, метод k средних. Эти методы предлагают новые и инновационные подходы к сегментации изображений, позволяя разделить все пиксели на k кластеров с минимальным отклонением точек кластеров от их центров. Метод строится на итеративном процессе, в котором каждый пиксель изображения присваивается к ближайшему кластеру на основе его цветовых характеристик. Затем, центры кластеров пере-

считываются на основе средних значений пикселей в каждом кластере, и процесс повторяется до достижения сходимости.

В контексте сегментации изображений, кластеры могут быть рассмотрены как различные цветовые области. Каждый кластер представляет собой группу пикселей с похожими цветовыми характеристиками, которые могут быть отнесены к одному и тому же объекту или региону на изображении. Путем разбиения всех N пикселей на k кластеров, мы можем достичь более точной и детализированной сегментации, где каждый кластер соответствует определенному цвету или области на изображении[14].

2.4. YOLO

YOLO (You Only Look Once) – это алгоритм компьютерного зрения, который обеспечивает высокую точность и эффективность в задаче обнаружения объектов на изображениях. Он основан на использовании сверточных нейронных сетей (CNN), которые позволяют извлекать важные признаки объектов из изображений.

Суть алгоритма YOLO[15] состоит в разделении изображения на равномерную сетку ячеек. Каждой ячейке CNN генерирует вероятность принадлежности к какому-то из классов объектов. Это позволяет алгоритму определить наличие объектов на изображении и их примерное местоположение. Ключевым шагом в работе YOLO является выбор активных ячеек, которые имеют вероятность класса выше заданного порогового значения. Эти активные ячейки используются для определения точного местоположения объектов путем предсказания координат ограничивающего прямоугольника (bounding box).

Главным из преимуществ YOLO является его способность обрабатывать изображения в режиме реального времени благодаря эффективной архитектуре. Это делает его применимым во многих областях, включая автономные транспортные средства, системы видеонаблюдения и робототехнику.

Однако, YOLO также имеет свои ограничения. Например, он мо-

жет столкнуться с трудностями в обнаружении маленьких объектов или объектов с низким контрастом. Также, при наличии перекрытия объектов, YOLO может испытывать трудности в точном определении границ каждого объекта.

2.5. RCNN

RCNN (Region-based Convolutional Neural Networks) представляет собой алгоритм по обнаружению и классификации объектов на изображениях. Этот метод основан на комбинации сверточных нейронных сетей (CNN) и региональных методов.

Алгоритм RCNN начинается с выделения пропозиций (region proposals) - потенциальных областей, где могут находиться объекты. Для этого используются методы, такие как Selective Search, которые генерируют кандидаты на основе текстурных, цветовых и пространственных характеристик изображения[13].

Затем каждая пропозиция подвергается обработке сверточной нейронной сетью, которая извлекает признаки из региона и преобразует его в фиксированный размер. Эти признаки затем подаются на вход классификатору, который определяет, к какому классу принадлежит объект в данной пропозиции.

Однако, обучение RCNN является сложным процессом, так как требуется большое количество вычислительных ресурсов и времени. В оригинальной версии RCNN каждая пропозиция обрабатывается независимо, что приводит к дублированию вычислений и замедлению процесса. Для решения этой проблемы были предложены улучшения, такие как Fast R-CNN и Faster R-CNN[3], которые вводят общую сверточную сеть для извлечения признаков и более эффективные методы обработки пропозиций. RCNN и его варианты демонстрируют высокую точность в обнаружении и классификации объектов на изображениях.

3. Оценки производительности и точности

Для оценки производительности и точности алгоритмов существует набор метрик, которые позволяют сравнить качество работы различных алгоритмов. В нашем случае, мы сталкиваемся с проблемой выбора подходящей метрики, так как нам необходимо оценить как качество обнаружения объектов из заданного списка классов, так и правильную классификацию этих объектов. Для задачи обнаружения объектов обычно используется метрика средней точности (Average Precision, AP), которая вычисляется на основе двух показателей: точности (precision) и полноты (recall)[17]. Рассмотрим возможные события при обнаружении объекта определенного класса:

- Истинно положительные события (true positives) - объект нужного класса был правильно обнаружен и заключен в обрамляющее окно.
- Ложно положительные события (false positives) - в обрамляющее окно было неправильно заключено что-то другое, не являющееся объектом нужного класса. Полнота и точность вычисляются отдельно для каждого класса, на котором обучен алгоритм.

3.1. Оценки полноты и точности

Точность (precision) обозначает процент верных предсказаний для обрамляющего окна (true positives) среди всех результатов для этого класса.

$$Precision = \frac{True \quad positivies}{True \quad positivies + False \quad positivies}$$

Полнота (recall) обозначает процент найденных обрамляющих окон для данного класса, среди всех, представленных в ground truth этого изображения.[16]

$$Recall = \frac{True \quad positivies}{Number \quad of \quad ground \quad truth \quad boxes}$$

3.2. Метрика IoU

Пересечение по объединению (Intersection over Union)[18] – метрика, используемая для оценки точности работы алгоритмов, детектирующих объекты. Значение IoU равно частному площади пересечения предсказанного обрамляющего окна и окна-ответа из ground truth на площадь объединения этих окон:

$$IoU = \frac{Area \quad of \quad Overlap}{Area \quad of \quad Union}$$

3.3. Mean Average Precision

Метрика mAP (mean Average Precision) является средним значением Average Precision для всех классов. Average Precision вычисляется для каждого класса как среднее значение точности при различных значениях порога.

$$meanAveragePrecision(mAP) = \frac{\sum_0^Q AP(q)}{Q}$$

, где Q – количество запросов, что в данном случае соответствует количеству классов, на которых обучен алгоритм.

4. Обучение модели

4.1. Выбор платформы для обучения

Существует множество платформ, которые предоставляют возможности для обучения различных типов нейросетевых моделей.

Одной из таких платформ является PyTorch, которая является одной из наиболее распространенных библиотек для машинного обучения. PyTorch имеет открытый исходный код и предоставляет широкие возможности для решения задач, связанных с компьютерным зрением и обработкой естественного языка. Она также позволяет выстраивать высокоуровневые модели с использованием тензорных вычислений и глубоких нейронных сетей. Кроме того, PyTorch может быть использована в сочетании с системами aurodiff для создания более сложных и комбинированных моделей.

Tensorflow – самая широкоиспользуемая библиотека для машинного обучения, предоставляющая необъятный выбор всевозможных инструментов для предобработки, постобработки, обучения различных нейросетевых моделей и создания новых архитектур с нуля. Широкая пользовательская база и подробная документация уменьшает порог вхождения для понимания пользования данным фреймворком.

Keras – высокоуровневый API для Tensoflow, Deeplearning4j и Theano, использующий их на выбор в качестве бэкэнда. Имеет очень понятный для разработчика интерфейс и легкопереносимый код, что обеспечивает гибкость при написании ряда проектов.

Tensor2Tensor – библиотека моделей глубокого обучения, построенных на основе библиотеки TensorFlow. Включает модели классификации и генерации изображений, модели диалогов, модели для языкового моделирования, распознавания речи, машинного перевода, аннотирования изображений.

Все вышеупомянутые модели могут быть реализованы с использованием модели "YOLO". Для компиляции этой модели наиболее удобно использовать Keras в сочетании с Tensorflow, так как они предоставля-

ют широкий спектр возможностей для оптимизации модели, включая:

- Изменение архитектуры нейронной сети для увеличения ее описательной глубины или ускорения производительности путем уменьшения количества слоев и оптимизации вычислений. Использование различных параметров при обучении, включая batch normalization для ускорения процесса обучения.

- Максимальная производительность при использовании библиотеки CUDA и cudnn от NVIDIA для выполнения вычислений на графических процессорах.

- Подробная документация и поддержка со стороны сообщества пользователей и разработчиков.

- Большое количество проектов, доступных в открытом доступе, которые предоставляют опыт реализации различных нейросетевых моделей и оценку их адекватности в новых направлениях.

- Библиотека Tensorboard для мониторинга обучения, которая предоставляет инструменты для анализа процесса обучения и поиска возможных ошибок при выборе оптимизаторов, гиперпараметров или обучающей выборки.

- Проект разрабатывается и поддерживается Google, одной из крупнейших IT-компаний, и существует на рынке уже более 7 лет, занимая лидирующие позиции в области глубокого обучения.

4.2. Обучающая и тестовая выборки

Для обучения нейросетевых моделей необходима большая выборка изображений для получения эффективной модели. В качестве набора данных был выбран открытый датасет Pipe-Dataset[10], который был разбавлен примерами, собранными с производства. Всего было собрано более 1000 изображений труб.

Для аннотирования изображения использовался инструмент LabelImg[9], одно из его преимуществ – возможность генерации напрямую в формат yolo.

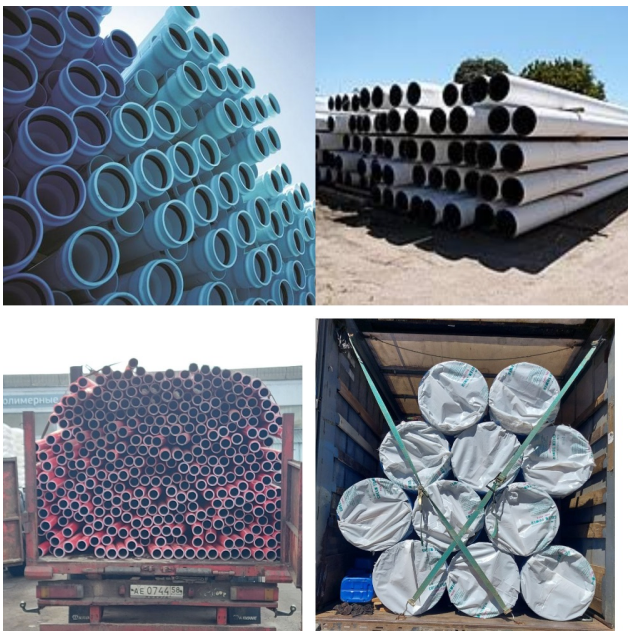

Рис. 2: Пример изображений из получившегося датасета.

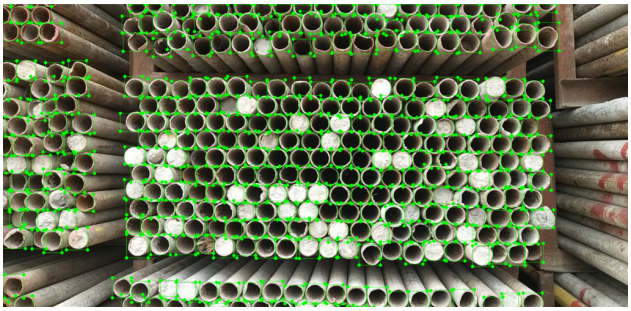

Рис. 3: Разметка изображений с помощью LabelImg.

4.3. Сравнение и выбор алгоритма

В данной работе были рассмотрены различные методы для распознавания трубной продукции с использованием компьютерного зрения. В качестве методов были выбраны пороговые методы, метод водоразделов, метод k-средних, YOLO, RCNN и FRCNN. Для оценки эффективности этих методов в данной задаче были использованы следующие критерии: полнота, точность, IoU, mAP.

Таблица сравнения по метрикам:

Алгоритм	**Precision**	**Recall**	**IoU**	**mAP**
Пороговый	0.70	0.75	0.62	0.70
Метод водоразделов	0.77	0.81	0.70	0.75
k-средних	0.82	0.85	0.73	0.81
RCNN	0.85	0.84	0.77	0.80
FasterRCNN	0.90	0.92	0.89	0.95
YOLO	0.91	0.93	0.91	0.95

Здесь YOLO показывает лучший результат по всем метрикам, что указывает на его высокую эффективность для поставленной задачи.

4.4. Реализация алгоритма для решения задачи распознавания и подсчёта труб

YOLOv5[4] представляет собой одну из последних версий алгоритма обнаружения объектов YOLO(You Only Look Once), который предлагает новую идею решения задачи обнаружения объектов путем преобразования ее в регрессионную задачу. Архитектура YOLOv5 отличается от предыдущих версий YOLO[1, 12, 11] и включает несколько нововведений для повышения точности и эффективности алгоритма.

Одним из нововведений YOLOv5 [7] является использование пакетной нормализации (batch normalization) [6] вместе со сверточными слоями для повышения точности и уменьшения возможности переобучения. Также была заменена магистральная сеть извлечения признаков на более мощную архитектуру Darknet53, что позволило лучше обнаруживать маленькие объекты.

Также в YOLOv5 был добавлен новый слой фокуса (focus layer), который заменил первые три слоя магистральной сети YOLOv3. Это позволило увеличить скорость обработки изображений при минимальных потерях в точности.

Кроме того, YOLOv5 является самой облегченной версией из предыдущих и использует фреймворк PyTorch вместо Darknet. Это упрощает разработку и интеграцию модели.

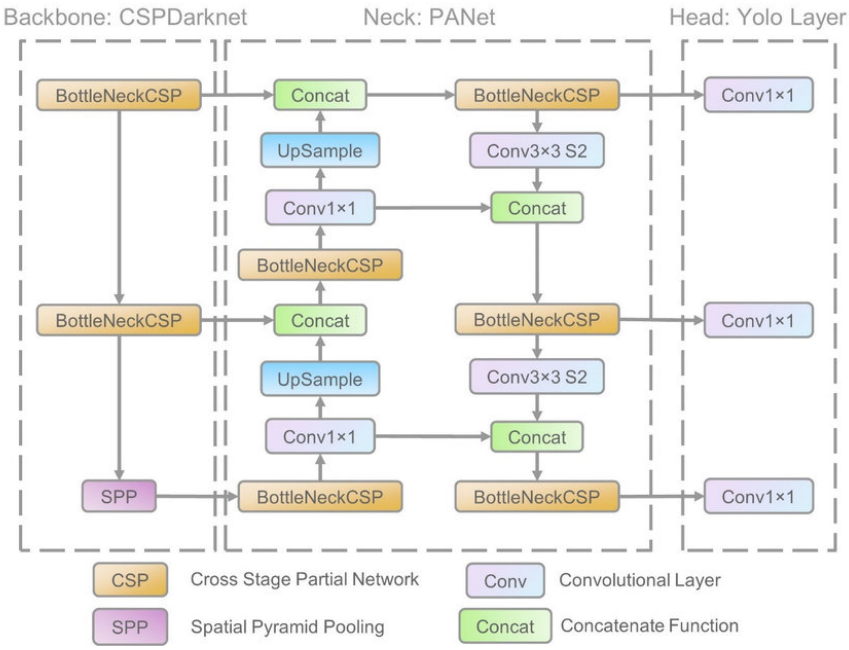

Рис. 4: Архитектура YOLOv5.

Для обучения и валидации моделей было использовано два графических процессора от Nvidia: GeForce 1070Ti и RTX 2080Ti. В качестве языка программирования был выбран Python и фреймфорк PyTorch. Главное преимущество этого выбора в том, что этот язык позволяет наиболее быстро и удобно писать скрипты, что особенно важно при работе с большими массивами данных, которые встречаются в области генерации тренировочных выборок для нейросетевых моделей. Все скрипты для конверсии и анализа были также написаны вручную. Все результаты работы можно вывести в отдельный файл для сбора статистики. Для реализации данных алгоритмов были использованы нативные библиотеки данного языка программирования: os, json, sys, argparse. В таблице ниже представлены гиперпараметры обучения модели.

Параметр	**Значение**
Количество эпох	20
Размер пакета	15
Скорость обучения	0,01
Сокращение веса	0.0005

Алгоритм YOLOv5 представлен в коде в виде класса yolov5. При

инициализации класса, передаются параметры onnx_path, confThreshold и nmsThreshold.

1. Метод init:

   - Принимает путь к модели ONNX (onnx_path) и устанавливает пороги уверенности (confThreshold) и пороги подавления немаксимумов (nmsThreshold);
   - Загружает модель ONNX и инициализирует классы объектов.

2. Метод detect:

   - Принимает входное изображение (srcimg) и флаг сохранения изображения с результатами (save_img);
   - Производит предобработку изображения и передает его в модель для получения выходных данных (outs);
   - Применяет алгоритм подавления немаксимумов к предсказаниям модели с использованием порога уверенности (confThreshold) и порога подавления немаксимумов (nmsThreshold);
   - Создает массив result_arr, в котором хранятся координаты и классы обнаруженных объектов;
   - Если флаг save_img равен True, то алгоритм визуализирует результаты на изображении, рисуя рамки вокруг обнаруженных объектов и выводя метки классов;
   - Возвращает изображение с нарисованными рамками (srcimg) и массив с результатами распознавания (result_arr).

3. Метод findPipes:

   - Принимает путь к модели ONNX (onnx_path), путь к входному изображению (img_path) и флаг сохранения изображения с результатами (save_img);

- Если путь к модели ONNX не указан, используется предустановленный путь;
- Инициализирует объект класса yolov5 с указанным путем к модели ONNX;
- Загружает входное изображение с помощью OpenCV;
- Получает ширину (width), высоту (height) и количество каналов (ch) изображения;
- Вызывает метод detect для обнаружения объектов на изображении и получения результатов;
- Если флаг save_img равен True, сохраняет изображение с результатами в указанную директорию;
- Возвращает ширину (width), высоту (height) и массив с результатами распознавания (result_arr).

Обучение модели производилось на протяжении 20 эпох. Время обучения составило 6,3 суток (152 часа), в среднем по 1 эпоха/7,65 час. К окончанию 20 эпохи модель продолжала медленно сходиться.

Рис. 5: Пример результата работы алгоритма YOLOv5.

Заключение

В ходе выполнения данной работы были выполнены следующие поставленные задачи:

- Выполнен обзор существующих алгоритмов;
- Изучены методы оценки качества моделей;
- Собраны и размечены данные;
- Произведено обучение модели YOLOv5, реализован алгоритм.

Список литературы

[1] A. Bochkovskiy C.-Y. Wang H.-Y. Mark Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. — 2020. — ISBN: arXiv:2004.10934v1.

[2] Alexe Bogdan, Deselaers Thomas, Ferrari Vittorio. V.: What is an object. — Vol. 73-80. — 2010. — 10. — P. 73–80.

[3] Ren Shaoqing, He Kaiming, Girshick Ross, Sun Jian. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. — 2016. — 1506.01497.

[4] A Forest Fire Detection System Based on Ensemble Learning / Renjie Xu, Haifeng Lin, Kangjie Lu et al. // Forests. — 2021. — 02. — Vol. 12. — P. 217.

[5] Szegedy Christian, Liu Wei, Jia Yangqing et al. Going Deeper with Convolutions. — 2014. — 1409.4842.

[6] Ioffe Sergey, Szegedy Christian. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. — 2015. — 1502.03167.

[7] J. Nelson J. Solawetz. YOLOv5 is Here: State-of-the-Art Object Detection at 140 FPS. — URL: https://blog.roboflow.com/yolov5-is-here/ (дата обращения: 16 июня 2023 г.).

[8] Kuo Weicheng, Hariharan Bharath, Malik Jitendra. DeepBox: Learning Objectness with Convolutional Networks // CoRR. — 2015. — Vol. abs/1505.02146. — arXiv : 1505.02146.

[9] LabelImg. — https://github.com/HumanSignal/labelImg.

[10] Pipe-Dataset. — https://www.kaggle.com/datasets/vijayshankar756/pipedataset/discussion/270644.

[11] Redmon Joseph, Farhadi Ali. YOLO9000: Better, Faster, Stronger. — 2016. — 1612.08242.

[12] Redmon Joseph, Farhadi Ali. YOLOv3: An Incremental Improvement. — 2018. — 1804.02767.

[13] Girshick Ross, Donahue Jeff, Darrell Trevor, Malik Jitendra. Rich feature hierarchies for accurate object detection and semantic segmentation. — 2014. — 1311.2524.

[14] Sarpe Adelina-Iulia. Image Segmentation with Clustering K-Means and Watershed Transform // 2010 Second International Conferences on Advances in Multimedia. — 2010. — P. 13–17.

[15] Redmon Joseph, Divvala Santosh, Girshick Ross, Farhadi Ali. You Only Look Once: Unified, Real-Time Object Detection. — 2016. — 1506.02640.

[16] deep_learning_object_detection. — https://github.com/hoya012/deep_learning_object_detection.

[17] ml-course-msu. — https://github.com/esokolov/ml-course-msu/blob/master/ML15/lecture-notes/Sem05_metrics.pdf.

[18] python_for_microscopists. — https://github.com/bnsreenu/python_for_microscopists.