|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **СОДЕРЖАНИЕ**  ВВЕДЕНИЕ. 4  1. ПОСТАНОВКА ЗАДАЧИ. 5   1. РАЗРАБОТКА АЛГОРИТМОВ. 10 2. РАЗРАБОТКА ПРОГРАММЫ. 16 3. ТЕСТИРОВАНИЕ 24    1. Описание входных и выходных данных 24    2. Результаты тестирования 24   ЗАКЛЮЧЕНИЕ. 29  СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ… 30  ПРИЛОЖЕНИЕ А ТЕКСТ ПРОГРАММЫ  ПРИЛОЖЕНИЕ Б ГРАФИЧЕСКИЙ МАТЕРИАЛ | | | | | | | | | | |
|  |  |  |  |  | *КР.ПО4.**190344-01 81 00* | | | | | |
|  |  |  |  |  |
| *Изм* | *Лист* | докум № | П*одп.* | Д*ата* |
| *Разраб.* | | *Луд А.С.* |  |  | *Детекция объектов на*  *фотографиях с помощью глубоких сверточных нейронных сетей.*  *Пояснительная записка* | *Лит* | | | *Лист* | *Листов* |
| *Проверил* | | *Крощенко А.А.* |  |  |  | *К* |  | *3* | 30 |
|  | |  |  |  | *БрГТУ* | | | | |
| *Н. контр.* | | *Крощенко А.А.* |  |  |
| *Утв.* | |  |  |  |

Разработка и сравнительный анализ алгоритмов управления движением мобильного робота

по заданной траектории.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ВВЕДЕНИЕ**  В последнее время области применения машинного зрения быстро  расширяются, охватывая все больше различных сфер жизнедеятельности.  На сегодняшний день для реализации систем машинного зрения в основном применяются свёрточные нейронные сети, которые являются одним из видов искусственных нейронных сетей. Именно благодаря способности к обучению, искусственные нейронные сети используются для решения сложных задач классификации и регрессионного анализа.  Системы компьютерного зрения, использующие такой подход, обладают хорошей устойчивостью к изменениям масштаба, смещениям, поворотам, смене  ракурса и прочим искажениям. На основе свёрточных нейронных сетей реализовываются системы для беспилотного управления транспортными  средствами, распознавания лиц и выявления различных заболеваний на  медицинских снимках.  Целью данной работы является реализация алгоритма, способного распознавать объекты на изображении, а именно определять наличие и правильное местоположение таких объектов на изображении. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **4** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **1 ПОСТАНОВКА ЗАДАЧИ**  Объектом КР: является распознавание объектов на изображении.  Предмет КР: алгоритм локализации и классификации объектов на изображении с применением глубоких свёрточных нейронных сетей.  **Распознавание изображений (классификация)**  Распознавание изображения относится к задаче ввода изображения в нейронную сеть и присвоения какой-либо метки для этого изображения. Метка, которую выводит сеть, будет соответствовать заранее определенному классу. Может быть присвоено как сразу несколько классов, так и только один.  Если существует всего только один класс, обычно применяется термин «распознавание», тогда как задача распознавания нескольких классов часто называется «классификацией».  Подмножество классификаций изображений - является уже определением объектов, когда определенные экземпляры объектов идентифицируются как принадлежащие к определенному классу, например, животные, автомобили или люди.  **Функция извлечения**  Чтобы выполнить распознавание или классификацию изображений, нейронная сеть должна выполнить извлечение признаков. Признаки - это элементы данных, которые представляют максимальный интерес и которые будут передаваться по нейросети. В конкретном случае распознавания изображений такими признаками являются группы пикселей, такие как линии и точки, которые сеть будет анализировать на наличие паттерна.  Распознавание признаков (или извлечение признаков) - это процесс извлечения соответствующих признаков из входного изображения, чтобы их можно было проанализировать. Многие изображения содержат аннотации или метаданные, которые помогают нейросети находить соответствующие признаки. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **5** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Как нейронные сети учатся распознавать изображения**  Понимание о том, как нейронная сеть распознает изображения, поможет  вам при реализации модели нейронной сети, поэтому давайте кратко рассмотрим процесс распознавания изображений в следующих разделах. Извлечение признаков при помощи фильтров показано на рис 1.1    Рисунок 1.1 – Извлечение признаков при помощи фильтров  Первый слой нейронной сети принимает все пиксели в изображении. После того, как все данные введены в сеть, к изображению применяются различные фильтры, которые формируют понимание различных частей изображения. Это извлечение признаков, которое создает «карты признаков».  Этот процесс извлечения признаков из изображения выполняется с помощью «свёрточного слоя», и свертка просто формирует представление части изображения. Именно из этой концепции свертки мы получаем термин “Свёрточная нейронная сеть” (Convolutional Neural Network, CNN) — тип нейронной сети, наиболее часто используемый в классификации и распознавании изображений. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **6** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| После успешного применения глубоких свёрточных нейронных сетей для классификации изображений обнаружение объектов на основе технологии глубокого обучения также добилось большого прогресса. Новый алгоритм, основанный на глубоком обучении, значительно лучше традиционного. В настоящее время фреймворки обнаружения целей, основанные на глубоком обучении, можно разделить на две категории:  1) Двухкаскадные детекторы, такие как региональные CNN (R-CNN) и их варианты;  2) Одноступенчатые детекторы, такие как SSD, YOLO и его варианты.  **Детектор второго порядка**  Во-первых, генератор кадров-кандидатов используется для генерации разреженного набора кадров-кандидатов, и особенности извлекаются из каждого кадра-кандидата; затем классификатор области используется для прогнозирования категории области кадра-кандидата.  **Детектор первого порядка**  Непосредственно выполнять прогнозирование категории для объекта в каждой позиции на карте функций, не проходя этап классификации области второго порядка. Структура детекторов показана на рис 1.2    Рисунок 1.2 – Структура детекторов | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **7** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Вообще говоря, детекторы второго порядка обычно имеют лучшую производительность обнаружения и достигли лучших на данный момент результатов в общедоступных тестах, в то время как детекторы первого порядка экономят время и имеют большую применимость при обнаружении целей в реальном времени.  Основное различие между ними заключается в том, что двухэтапный алгоритм должен сначала сгенерировать предложение (поле предварительного выбора, которое может содержать объект для проверки), а затем выполнить детальное обнаружение объекта.  Одноэтапный алгоритм будет напрямую извлекать характеристики из сети для прогнозирования классификации и местоположения объекта.  YOLO рассматривает обнаружение цели как проблему регрессии и делит все изображение на фиксированное количество ячеек сетки (например, используя сетку 7 × 7). Каждая единица рассматривается как кадр-кандидат, а затем сеть определяет, есть ли в кадре-кандидате один или несколько объектов.  Деление изображения на ячейки показано на рис 1.3    Рисунок 1.3 – Деление изображения на ячейки | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **8** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Основываясь на тщательно разработанной облегченной архитектуре, YOLO может выполнять прогнозы со скоростью 45 кадров в секунду и может достигать скорости 155 кадров в секунду с более упрощенной магистральной сетью. Однако перед YOLO стоят следующие проблемы:  1)Для данного местоположения он может обнаруживать не более двух объектов, что затрудняет обнаружение небольших объектов и переполненных объектов.  2)Для прогнозирования можно использовать только последнюю карту функций, что не подходит для прогнозирования объектов нескольких размеров и соотношений сторон. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **9** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **РАЗРАБОТКА АЛГОРИТМОВ**   В данной работе необходимо реализовать алгоритм, который будет способен распознавать на изображении объекты. Процесс распознавания объекта на изображении включает в себя 2 этапа: нахождение местоположения объекта и присвоение ему конкретного класса, исходя из его свойств.  Иногда для решения задачи обнаружения объекта на изображении применяются такие подходы, как наложение цветовых фильтров на изображение, выделение и анализ контуров, сопоставление с шаблонами, а также выявление и поиск особых точек.  Данные методы позволяют выделять образы, которые могут помочь определить класс объекта на изображении.  Машинное обучение является подразделом искусственного интеллекта – области, занимающейся разработкой систем, которые смогли бы справляться с задачами, решение которых считается исключительно прерогативой человека.    В настоящее время наиболее популярным способом классификации объектов на изображении является такой метод машинного обучения, как искусственные нейронные сети, а именно их специальная архитектура, называемая свёрточной нейронной сетью.  Данный подход обеспечивает частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и прочим искажениям, но основной особенностью свёрточных нейронных сетей является их обучаемость.  Недостатком свёрточных нейронных сетей является то, что они не способны классифицировать множество объектов на одном изображении, а для локализации возгорания необходимо распознавать несколько классов объектов.  Для задач такого рода существуют специальные алгоритмы на основе свёрточных нейронных сетей, такие как R-CNN, YOLO и SSD, которые позволяют производить локализацию и классификацию множества объектов на изображении. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **10** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Сверточная нейронная сеть является одной из наиболее популярных архитектур нейронных сетей, применяемая для анализа визуальных образов и классификации изображений. Данная архитектура спроектирована на основе принципа зрительной коры головного мозга, в которой отдельные участки клеток реагируют на возбуждения в своих конкретных областях поля зрения, называемых рецептивными полями. При восприятии различных образов, в зрительной коре активируются разные группы нейронов. В целом, архитектура сверточной нейронной сети позволяет выделять в данных различные признаки, начиная от очень простых и заканчивая более сложными.  Свёрточная нейронная сеть состоит из чередующихся слоев свертки и подвыборки, после которых следуют полносвязные слои. Архитектура свёрточной нейронной сети представлена на рис 2.1    Рисунок 2.1 – Архитектура свёрточной нейронной сети  Свёрточный слой всегда является первым в сверточной нейронной сети  и реализует операцию свёртки, которая отражает реакцию отдельного  нейрона на конкретную область поля зрения. На вход свёрточного слоя  подается изображение в виде матрицы пикселей, размерность которой  определяется размером изображения. Если изображение черно-белое, то на  вход подается одна матрица, каждый пиксель которой описывается  значением яркости в диапазоне от 0 до 255. Цветные изображения  представляются тремя матрицами, каждая из которых отвечает за один из  каналов цветовой модели RGB, значения матриц описывают интенсивность  цвета пикселей. В качестве нейрона выступает некое ядро свертки – матрица,  элементы которой являются весовыми коэффициентами. Глубина ядра  свертки должна быть такой же, как и глубина изображения, к примеру для  цветного изображения применяется ядро свертки, состоящее из трех матриц,  каждая из которых применяется для свертки по одному из каналов цветного  изображения. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **11** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размерность матрицы ядра свертки обычно варьируется от 3 до  7 и определяет количество признаков, которые необходимо объединить в  один новый признак более высокого уровня. Операция свертки начинается с  определения рецептивного поля, размер которого равен ядру свертки. Данное  поле скользит с некоторым смещением по исходному изображению и  пиксели изображения, попавшие в него, поэлементно перемножаются с  ядром свертки. Элементы новой матрицы, полученной в результате  перемножения, суммируются и образуют новый выходной пиксель.  Операция свертки представлена на рисунке 2.2    Рисунок 2.2 – Операция свертки  Рисунок 2.2 описывает операцию свертки для черно-белого  изображения. Свертка цветного изображения отличается лишь тем, что к  каждому цветовому каналу применяется одна из матриц, составляющих  трехмерное ядро свертки, после чего полученные матрицы суммируются в  одну результирующую матрицу. Операция свертки для цветного  изображения представлена на рисунке 2.3    Рисунок 2.3 – Свертка цветного изображения | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **12** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |
| Из-за того, что граничные пиксели исходного изображения не оказываются в центре рецептивного поля, матрица, полученная в результате свертки, имеет меньшую размерность, чем исходное изображение. Чтобы избежать постоянного сжатия изображения при выполнении свертки и сохранить информацию о его границах, вокруг изображения добавляется рамка из нулей, ширина которой зависит от размера ядра свертки. Данная операция называется заполнением (zero-padding). Свертка цветного изображения с заполнением представлена на рисунке 1.7    Рисунок 2.4 – Свертка цветного изображения с применением заполнения  Также, как и в многослойных нейронных сетях, в данной архитектуре присутствует смещение – значение, изменяемое в процессе обучения, которое складывается с каждым элементом выходной матрицы. Перед поступлением на следующий слой, полученная матрица проходит через функцию активации. Обычно в свёрточных слоях в качестве функции активации используют функцию усеченного линейного преобразования, которая преобразовывает все отрицательные значения в 0, а остальные не изменяет.  Итак, в результате свертки получается матрица, называемая картой признаков, на которой выделен конкретный признаки, такой как: различные прямые и кривые линии, полуокружности или более сложные признаки, в зависимости местонахождения свёрточного слоя в модели нейронной сети. Если первый свёрточный слой выявляет базовые признаки, то последующие свёрточные слои выявляют признаки все более и более высокого уровня, обобщая информацию на изображении. Свёрточные слои чаще всего используют более одного ядра свертки, каждый из которых идентифицирует конкретный признак, поэтому в результате свертки получается набор различных числовых матриц, число которых характеризует глубину карты признаков. Чем больше количество используемых ядер свертки, тем более подробную информацию можно получить об исходном изображении. После слоя свертки обычно следует слой подвыборки. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **13** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |
| Данный слой уменьшает пространственную размерность карты признаков, это достигается путем замены некоторой области пикселей одним пикселем. Необходимость этой операции объясняется тем, что важна именно информация о наличии конкретных признаков на изображении, а не их местоположение, поэтому оно сжимается до менее подробного, состоящего из доминирующих признаков. Кроме того, операция подвыборки уменьшает количество параметров сверточной сети, тем самым снижая вычислительную нагрузку. На практике используется всего два типа подвыборки: по среднему значению, в котором некоторая области пикселей объединяется в среднее значение интенсивности, и по максимальному значению, в котором та же область объединяется в пиксель с максимальной интенсивностью. Чаще всего используют подвыборку по максимальному значению, так как она более четко выделяет ключевые признаки, подавляя лишние шумы на изображении. При выполнении операции подвыборки учитываются два параметра: размер ядра подвыборки, скользящего по изображению, которое сворачивает попавшие в него пиксели до 1 пикселя и размер шага в пикселях, на которое сдвигается ядро подвыборки. Чаще всего используют пара значений параметров: 2x2, которая уменьшает размерность карты признаков в два раза. Пример выполнения операции подвыборки приведен на рисунке 2.5    Рисунок 2.5 – Пример выполнения подвыборки по максимальному значению | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **14** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |
| Для того, чтобы выявить сложные признаки на изображении, свёрточные сети содержат множество чередующихся слоев свертки и подвыборки. Таким образом, с каждой парой слоев свертки глубина карты признаков изменяется в зависимости от числа ядер свертки, а их размерность уменьшается.  Конечными слоями свёрточной нейронной сети являются полносвязные слои, с помощью которых происходит классификация объекта на изображении. Весовые коэффициенты данных слоев настраиваются в процессе обучения для определения метки класса на основе выявленных признаков.  Перед подачей полученной карты признаков на полносвязный слой, происходит объединение слоев карты признаков в вектор, где каждый элемент показывает вероятность присутствия конкретного признака на изображении. К примеру, чтобы изображение было классифицировано как лицо, вероятность присутствия таких признаков как глаза, рот и нос должна быть высокой.  Последний полносвязный слой содержит количество нейронов, равное количество определяемых классов и с помощью выбранной функции активации выводит вероятности присутствия каждого класса на изображении.  Таким образом, были рассмотрены особенности архитектуры свёрточной нейронной сети и разобран процесс извлечения признаков из изображения для классификации находящегося на нем объекта. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **15** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **РАЗРАБОТКА ПРОГРАММЫ**   До появления YOLO большинство способов обнаружения объектов пытались адаптировать классификаторы для детекции.  В YOLO же, обнаружение объектов было сформулировано как задача регрессии на пространственно разделенных ограничивающих рамок (bounding boxes) и связанных с ними вероятностей классов.  Так как YOLO необходимо только один взгляд на изображение, то метод скользящего окна не подходит в данной ситуации. Вместо этого, изображение будет поделено на сетку с ячейками размером S x S. Каждая ячейка может содержать несколько разных объектов для распознавания. Во-первых, каждая ячейка отвечает за прогнозирование количества bounding box’ов.  Также, каждая ячейка прогнозирует доверительное значение (confidence value) для каждой области, ограниченной bounding box’ом. Иными словами, это значение определяет вероятность нахождения того или иного объекта в данной области.  То есть в случае, если какая-то ячейка сетки не имеет определенного объекта, важно, чтобы доверительное значение для этой области было низким. Когда мы визуализируем все предсказания, мы получаем карту объектов и упорядоченных по доверительному значению, рамки(показано на рис 3.1).    Рисунок 3.1 – Карта объектов, упорядоченных по доверительному значению | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **16** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Во-вторых, каждая ячейка отвечает за предсказание вероятностей классов. Это не говорит о том, что какая-то ячейка содержит какой-то объект, только вероятность нахождения объекта.  Допустим, если ячейка предсказывает автомобиль, это не гарантирует, что автомобиль в действительности присутствует в ней. Это говорит лишь о том, что если присутствует объект, то этот объект скорее всего автомобиль.  В YOLO используются anchor boxes (якорные рамки / фиксированные рамки) для прогнозирования bounding box’ов.  Идея anchor box’ов сводится к предварительному определению двух различных форм. И таким образом, мы можем объединить два предсказания с двумя anchor box’ами (в целом, мы могли бы использовать даже большее количество anchor box’ов).  Эти якоря были рассчитаны с помощью датасета COCO (Common Objects in Context) и кластеризации k-средних (K-means clustering). Данные якорные рамки показаны на рис 3.2    Рисунок 3.2 – Якорные рамки | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **17** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Простое нахождение порогового значения избавит нас от прогнозов с низким доверительным значением. Для следующего шага важно определить метрику IoU (Intersection over Union / Пересечение над объединением). Эта метрика равняется соотношению площади пересекающихся областей к площади областей объединенных(рис 3.3).    Рисунок 3.3 – метрика IoU  После этого все равно могут остаться дубликаты, и чтобы от них избавиться нужно использовать “подавление не-максимумов” (non-maximum suppression). Подавление не-максимумов заключается в следующем: алгоритм берёт bounding box с наибольшей вероятностью принадлежности к объекту, затем, среди остальных граничащих bounding box'ов с данной области, возьмёт один с наивысшим IoU и подавляет его. Ввиду того, что все делается за один прогон, эта модель будет работать почти также быстро, как и классификация. К тому же все обнаружения предсказываются одновременно, что означает, что модель неявно учитывает глобальный контекст. Проще говоря, модель может узнать какие объекты обычно встречаться вместе, их относительный размер и расположение объектов и так далее(рис 3.4).    Рисунок 3.4 – метрика для YOLOv3 | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **18** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Что такое YOLO?**  YOLO (вы смотрите только один раз) - это метод/способ обнаружения объектов. Это алгоритм/стратегия, лежащая в основе того, как код собирается обнаруживать объекты на изображении. Более ранние рамки обнаружения, рассматривали разные части изображения несколько раз в разных масштабах и применяли технику классификации изображений для обнаружения объектов.  Этот подход медленный и неэффективный. YOLO использует совершенно другой подход. Он просматривает все изображение только один раз,  проходит через сеть один раз и обнаруживает объекты.  **OpenCV DNN модуль**  Модуль DNN (Deep Neural Network) изначально был частью opencv\_contrib. Он был перемещен в основной филиал opencvс, предоставив пользователям возможность делать выводы по предварительно обученным моделям глубокого обучения в самом OpenCV.(Стоит отметить, что dnn Модуль не предназначен для обучения. Это просто для вывода на изображения/видео.)  Изначально поддерживались только модели Caffe и Torch. В течение этого периода добавляется поддержка различных фреймворков/библиотек, таких как TensorFlow.  Поддержка YOLO/DarkNet была добавлена ​​недавно. Я собираюсь использовать модуль OpenCV dnn с предварительно обученной моделью YOLO для обнаружения общих объектов.  **Non-maximal Suppression** — метод, который подавляет перекрывающиеся ограничивающие прямоугольники,  у которых вероятности для обнаружения объекта меньше максимальной. В основном это достигается в два этапа:  1.Выбираем ограничивающую рамку с наибольшей достоверностью (то есть вероятностью).  2.Затем сравниваем её с вероятностями всех других ограничивающих прямоугольников и удаляем те, которые имеют высокий IoU. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **19** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **IoU** (Intersection over Union или пересечение над объединением) — это метод, используемый в Non-maximal Suppression для сравнения того, насколько близки два разных ограничивающих прямоугольника. Это просто показано на следующем рисунке 3.5    Рисунок 3.5 – IoU  Чем выше IoU, тем ближе ограничивающие рамки. IoU, равное 1, означает, что две ограничивающие рамки совпали, а IoU, равное 0, означает, что они даже не пересекаются. В результате мы будем использовать пороговое значение IoU 0,5 (которое задано в начале урока) и это означает, что будет удалятся любая ограничивающая рамка со значением ниже значения ограничивающей рамки с максимальной вероятностью. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **20** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |
| Наш проект сегодня состоит из 4 каталогов и двух скриптов Python. Справочники (в порядке важности):  **yolo-coco/** : Файлы модели детектора объектов YOLOv3 предварительно обучены (на основе набора данных COCO). Они были обучены командой Darknet.  **images/** : Эта папка содержит изображения, на которых мы будем выполнять обнаружение объектов для целей тестирования и оценки.  **video/** : После обнаружения объектов с помощью YOLO на изображениях мы будем обрабатывать видео. Этот каталог содержит видео для тестирования.  **output/** : Выходные видео, которые были обработаны YOLO и аннотированы ограничительными полями и именами классов, могут находиться в этой папке.  Мы рассматриваем два скрипта Python — yolo.py и yolo-video.py .  Первый сценарий предназначен для изображений, а затем мы возьмем то, что мы узнаем, и применим это к видео во втором сценарии.  При разработке мы будем использовать уже готовую сеть с весами предобученными на COCO датасете. В нем представлено 80 различных классов.  Следовательно наша нейронка будет способна распознать 80 различных объектов.  Теперь разберем четыре аргумента командной строки. Аргументы командной строки обрабатываются во время выполнения и позволяют нам изменять входные данные нашего скрипта из терминала. Наши аргументы командной строки включают:  **--image** : Путь к входному изображению. Мы будем обнаруживать объекты на этом изображении с помощью YOLO.  **--yolo** : Базовый путь к каталогу YOLO. Затем наш скрипт загрузит необходимые файлы YOLO, чтобы выполнить обнаружение объектов на изображении.  **--confidence** : Минимальная вероятность фильтрации слабых обнаружений. Я присвоил этому значение по умолчанию 50% (0,5 ), но вы должны не стесняться экспериментировать с этим значением.  **--threshold** : Это наш порог подавления без максимумов со значением по умолчанию 0,3 . Вы можете прочитать больше о подавлении не-максим здесь.  После синтаксического анализа args переменная теперь представляет собой словарь, содержащий пары ключ-значение для аргументов командной строки. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **21** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |
| **Аргументы командной строки(argparse)**  Сценарий требует четыре входных аргумента:  1)входное изображение  2)Конфигурационный файл YOLO  3)Предварительно подготовленные веса YOLO  4)Файл, содержащий имена классов  Пример:   1. Для изображения:   python yolo.py --image images/city.jpg --yolo yolo-coco     1. Для видео:   python yolo-video.py --input video/Aircraft.mp4 --output output/Aircraft.avi --yolo yolo-coco  Далее мы загружаем метки класса COCO, на которые была обучена наша модель YOLO:  labelsPath=os.path.sep.join([args["yolo"], "coco.names"])  LABELS = open(labelsPath).read().strip().split("\n")  Инициализируем список цветов для представления каждой возможной метки класса:  np.random.seed(42)  COLORS=np.random.randint(0,255,size=(len(LABELS),3),dtype="uint8")  Пишем производные пути к весам YOLO и конфигурации модели:  weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])  configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])  Загрузить наш детектор объектов YOLO, обученный на наборе данных COCO (80 классов):  print("[INFO] loading YOLO from disk...")  net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)  Загружаем наше входное изображение и захватываем его пространственные размеры:  image = cv2.imread(args["image"])  (H, W) = image.shape[:2]  Определяем только имена \*выходных\* слоев, которые нам нужны из YOLO:  ln = net.getLayerNames()  ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()] | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **22** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |
| Создаём большой двоичный объект из входного изображения, а затем выполняем прямой проход детектора объектов YOLO, дающий нам наши ограничители исвязанные вероятности:  blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),  swapRB=True, crop=False)  net.setInput(blob)  start = time.time()  layerOutputs = net.forward(ln)  end = time.time()  Показываем информацию о времени выполнения детекции на YOLO:  print("[INFO] YOLO took {:.6f} seconds".format(end - start))  Применяем подавление без максимумов для подавления слабых, перекрывающихся границ коробки:  idxs=cv2.dnn.NMSBoxes(boxes,confidences,args["confidence"],args["threshold"])  После всех обработок и ограничений показываем выходное изображение:  cv2.imshow("Image", image)  Для видео аналогично, только мы инициализируем видеопоток, указываем на видеофайл и размеры рамки:  vs = cv2.VideoCapture(args["input"])  writer = None  (W, H) = (None, None)  И пробуем определить общее количество кадров в видеофайле:  prop = cv2.cv.CV\_CAP\_PROP\_FRAME\_COUNT if imutils.is\_cv2() \  else cv2.CAP\_PROP\_FRAME\_COUNT  total = int(vs.get(prop))  print("[INFO] {} total frames in video".format(total))  И при выводе записываем выходной кадр на диск в рамке:  writer.write(frame) | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **23** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1. **ТЕСТИРОВАНИЕ**    1. **Описание входных и выходных данных**   Формами входных данных являются аргументы командной строки, основой которых является входное изображение.  Формой выходных данных является изображение/видео с обнаруженными на них объектами в рамке случайного цвета и значением точности совпадения.   * 1. **Результаты тестирования**   Среда тестирования – ПК, процессор Intel Core i5-7200U с частотой 2.5 - 2.71 ГГц, ОЗУ 8 ГБ, тип системы: 64-разрядная OC Windows 10 Домашняя. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **24** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 1:** «Распознавание объектов на изображении»  Ожидаемый результат: объект, который находится на изображении будет обнаружен(выделен в рамку случайного цвета со значение точности совпадения).  Описание: пользователь задаёт аргументы командной строки. Данные этих аргументов передаются в исходный код программы, который в дальнейшем выполняет процесс распознавания(см. рисунки 4.1.1 — 4.1.6).  Полученный результат:    Рисунок 4.1.1 – Аргументы входных данных    Рисунок 4.1.2 – Входное изображение    Рисунок 4.1.3 – Результат обработки | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **25** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Рисунок 4.1.4 – Аргументы входных данных    Рисунок 4.1.5 – Входное изображение    Рисунок 4.1.6 – Результат обработки  Вывод: результат соответствует ожиданиям. На заданных изображениях были найдены(car, traffic light, и dog) с вполне высокой точностью. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **26** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Тест 2:** «Распознавание объектов на видео»  Ожидаемый результат: объект, который находится на кадрах, разбитых при обработке видео, будет обнаружен(выделен в рамку случайного цвета со значение точности совпадения) в течении всего видео.  Описание: пользователь задаёт аргументы командной строки. Данные этих аргументов передаются в исходный код программы, который в дальнейшем выполняет процесс распознавания(см. рисунки 4.2.1 — 4.2.5).  Полученный результат:    Рисунок 4.2.1 – Аргументы входных данных    Рисунок 4.2.2 – Кадр из видео    Рисунок 4.2.3 – Кадр из видео | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **27** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Рисунок 4.2.4 – Результат обработки    Рисунок 4.2.5 – Результат обработки  Вывод: результат соответствует ожиданиям. На протяжении всего видео, на кадрах, разбитых при обработке видео, был обнаружен(aeroplane) с вполне высокой точностью. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **28** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ЗАКЛЮЧЕНИЕ**  В данной курсовой работе исследовался вопрос локализации объектов на изображении.  Для этого было необходимо было выбрать и реализовать алгоритм на основе свёрточных нейронных сетей, способный распознавать различные классы объектов.  Был проведен анализ существующих методов распознавания объектов на изображении и сделан вывод, что для локализации объектов наиболее подходящими являются методы, основанные на машинном обучении, а именно свёрточные нейронные сети.  Были рассмотрены основные принципы работы искусственных нейронных сетей и особенности их построения. Кроме того, был подробно разобран процесс применения свёрточных нейронных сетей для классификации объекта на изображении.  Для локализации и классификации объектов на изображении было решено использовать алгоритм You Look Only Once(YOLO)v3, в основу которого входят свёрточные нейронные сети.  Данный алгоритм был выбран поскольку он обнаруживает объекты за один прямой проход нейронной сети и является лучшим в плане быстродействия, что является немало важным аспектом при распознавании объектов в режиме реального времени.  Также были описаны основные принципы работы алгоритма YOLOv3 и произведена его реализация для локализации объктов на изображении на языке Python.  Таким образом, был реализован алгоритм на основе сверточных нейронных сетей, способный обнаруживать на изображении различные классы объектов с довольно высокой точностью. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **29** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**   1. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам. 2. ГОСТ 19.504-79. Единая система программной документации ЕСПД. Руководство программиста. Требования к содержанию и оформлению. 3. ГОСТ 19.701-90. ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.   4. ГОСТ 19.005-85. ЕСПД.  Р-схемы алгоритмов и программ. Обозначения условные графические и правила выполнения.   1. ГОСТ 19.101-77. ЕСПД. Виды программ и программных документов. 2. ГОСТ 19.102-77. ЕСПД. Стадии разработки. 3. ГОСТ 19.103-77. ЕСПД. Обозначения программ и программных документов. 4. ГОСТ 19.401-78. ЕСПД. Текст программы. Требования к содержанию и оформлению. 5. ГОСТ 19.402-78. ЕСПД. Описание программы. 6. Бурков А. Машинное обучение без лишних слов / Андрей Бурков -   Питер СПб, 2020. – 192 c.   1. ГОСТ 7.1-2003. Система стандартов по информации, библиотечному и   издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. | | | | | | |
|  |  |  |  |  | *КР.ПО4.190344-01 81 00* | *Лист* |
|  |  |  |  |  |
| **30** |
| *Изм* | *Лист* | *№ докум.* | *Подп.* | *Дата* |

