Описание технической архитектуры

программного обеспечения «Система поиска GigaFlex C1 Экстренный поиск»

Термины и определения

БВС (беспилотные воздушные суда): Автономные или дистанционно управляемые воздушные суда, не предназначенные для нахождения человека на борту во время полета.

Консоль: Интерфейс командной строки, позволяющий пользователю взаимодействовать с программным обеспечением без использования графического интерфейса.

Технологии искусственного интеллекта: Технологические решения и алгоритмы, имитирующие человеческий интеллект, используемые для автоматизации различных процессов, включая распознавание образов.

Linux (Ubuntu): Операционная система, основанная на ядре Linux и использующая рабочее окружение GNOME по умолчанию. Ubuntu 18.04 и выше обозначают конкретные версии этой операционной системы.

Conda: Система управления пакетами и средой, которая позволяет устанавливать, запускать и обновлять библиотеки и зависимости в различных языках программирования, таких как Python и R, в изолированных средах, называемых «кондой». Conda облегчает управление пакетами и зависимостями в рамках проектов, что способствует воспроизводимости и портативности кода.

Miniconda: Минимальная установка conda, которая включает только conda, Python, зависимости, которые используются conda, и несколько других полезных пакетов.

Python: Высокоуровневый язык программирования, ориентированный на повышение производительности разработчика и читаемости кода. Python поддерживает несколько парадигм программирования, включая структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование.

PyTurboJpeg: Библиотека для работы с изображениями в формате JPEG. PyTurboJpeg предоставляет Python-интерфейс для библиотеки TurboJPEG, что позволяет выполнять быстрое сжатие и декомпрессию

изображений, а также обеспечивает поддержку работы с изображениями высокого разрешения.

Intersection over Union (IoU): Метрика, используемая для оценки точности алгоритмов объектного обнаружения, измеряющая степень перекрытия между предсказанным и реальным ограничивающим прямоугольником. Значения варьируются от 0 (отсутствие перекрытия) до 1 (полное совпадение).

Пороговое значение (Threshold): Значение, используемое для определения, должен ли объект быть классифицирован как интересующий нас объект или отброшен. Используется для управления уровнем уверенности алгоритма в правильности классификации объекта.

Ограничивающий прямоугольник (Bounding Box): Прямоугольник, который рисуется вокруг обнаруженного объекта на изображении, показывая расположение и размеры объекта.

YOLO (You Only Look Once): является современным алгоритмом в области компьютерного зрения, специализирующимся на задачах детекции и классификации объектов на изображениях. Этот алгоритм уникален благодаря своей способности обрабатывать изображение за один просмотр, в результате чего достигается высокая скорость обработки при сохранении достаточной точности. Алгоритм разделяет входное изображение на множество ячеек сетки и для каждой ячейки параллельно предсказывает ограничивающие рамки и вероятности классов объектов. В отличие от традиционных методов, YOLO способен определять множество объектов различных классов на изображении, что делает его выдающимся инструментом в области машинного обучения и анализа изображений.

1. Общие сведения

Данный документ содержит описание технической архитектуры программного обеспечения «Система поиска GigaFlex C1 Экстренный поиск» (далее – ПО). Все исключительные права на ПО принадлежат Фонду НТИ (далее – Компания).

ПО не имеет пользовательского графического интерфейса, управление осуществляется с использованием консоли.

ПО разработано с целью автоматизации процесса поиска людей в условиях природной местности.

ПО способно автоматически идентифицировать человеческие фигуры на изображениях, полученных после полета беспилотных воздушных судов (БВС), используя технологии искусственного интеллекта.

Эффективность распознавания оптимизирована для работы на высотах от 45 до 70 метров. Максимальная эффективность достигается при высоте полета близкой к 45 метрам.

2. Описание архитектуры приложения

2.1. Описание функций

В данном программном обеспечении не используются классы. Ниже приведены используемые функции:

Split_list: функция разделяет входной список на заданное количество подсписков. Размер подсписков определяется на основе длины входного списка и заданного количества подсписков. Параметры функции включают в себя lst и n. lst – это входной список, который будет разделен, тип данных - список любых объектов. n – это количество подсписков, на которое следует разделить входной список, тип данных – целое число. Функция возвращает список, состоящий из подсписков, на которые был разделен входной список, тип данных возвращаемого значения – список списков, при этом сохраняется тип данных элементов исходного списка.

run_infer: функция не принимает никаких аргументов на вход и выполняет следующие действия: сначала происходит сбор изображений, создается список images_list, который содержит пути ко всем изображениям в директории TEST_IMAGES_PATH с расширением .JPG, используется метод glob из модуля Path библиотеки pathlib. Затем список изображений images_list разделяется на n частей с помощью функции split_list. Далее, для каждой части списка из result запускается подпроцесс с использованием subprocess.Popen, каждый подпроцесс выполняет скрипт part_inference.py с путями изображений в качестве аргументов. Функция ожидает завершение всех запущенных подпроцессов с использованием метода communicate. После этого результаты, сохраненные в CSV-файлах в директории tmp_outputs, считываются и объединяются в один DataFrame с использованием функций pd.read_csv и pd.concat из библиотеки pandas. В конце объединенный DataFrame сохраняется в файл, путь к которому определен в SAVE_PATH. В функции используются также сторонние методы и модули, такие как Path.glob, subprocess.Popen, pd.read_csv и pd.concat, и должны быть определены такие параметры и константы, как TEST_IMAGES_PATH, SAVE_PATH и processes.

main.main: Функция main() является точкой входа в программу и не принимает никаких аргументов. Она отвечает за вызов функции run_infer(), которая выполняет ряд операций, связанных с изображениями, подпроцессами и файлами CSV, как было описано ранее.

calculate_intersection_area: функция возвращает тип float и вычисляет площадь пересечения двух прямоугольников. Параметры функции включают box1 и box2, которые представляют собой numpy массивы, содержащие координаты прямоугольников. Функция возвращает площадь пересечения двух прямоугольников и для выполнения использует библиотеку torch.

calculate_area: функция возвращает тип float и вычисляет площадь прямоугольника. Параметр функции box – это список координат прямоугольника. Функция возвращает площадь прямоугольника.

calculate_bbox_ios: функция возвращает тип float и вычисляет отношение площади пересечения к площади меньшего из прямоугольников. Параметры функции, pred1 и pred2, являются предсказаниями, содержащими информацию о прямоугольниках. Возвращаемое значение функции – это отношение площади пересечения к площади меньшего из прямоугольников.

has_match: функция возвращает булево значение и определяет, соответствует ли отношение площади пересечения к площади меньшего из прямоугольников заданному порогу. Параметры функции включают pred1 и pred2, которые являются предсказаниями, и match_threshold, который является пороговым значением. Функция возвращает булево значение, указывающее, превышает ли отношение площади пересечения пороговое значение.

get_merged_bbox: функция отвечает за получение объединенного прямоугольника из двух прямоугольников. Параметры функции, pred1 и pred2, являются предсказаниями и содержат информацию о прямоугольниках.

calculate_box_union: функция вычисляет объединение двух прямоугольников. В качестве параметров принимаются box1 и box2, которые содержат координаты прямоугольников. Функция возвращает координаты

прямоугольника, являющегося объединением двух исходных прямоугольников.

get_merged_score: функция возвращает максимальное значение из двух оценок (scores). Параметры функции, pred1 и pred2, являются предсказаниями и содержат оценки.

merge_object_prediction_pair: функция объединяет два предсказания в одно, сохраняя информацию о прямоугольнике и оценке. Параметры функции, pred1 и pred2, являются предсказаниями объектов. Функция возвращает объединенное предсказание. Все функции активно используют библиотеку torch для выполнения тензорных операций, что обеспечивает эффективные вычисления на CPU или GPU.

greedy_nmm: Параметры функции включают object_predictions_as_tensor, который является тензором torch, содержащим предсказания местоположения на изображении вместе с оценками предсказания класса, его форма [num_boxes,5], где каждый блок представлен своими координатами и оценкой. Другой параметр - match_metric, строковый тип с допустимыми значениями «IOU» или «IOS», используемый для сопоставления, и по умолчанию равен «IOU». Также есть параметр match_threshold типа float с дефолтным значением 0.5, который определяет порог перекрытия для сопоставляемой метрики, блоки с перекрытием больше этого порога будут рассматриваться для слияния. Функция возвращает словарь keep_to_merge_list типа Dict[int, List[int]], где каждый ключ – это индекс предсказания для сохранения, и каждое значение – это список индексов предсказаний для слияния. Функция применяет жадную версию слияния без максимума для уменьшения количества перекрывающихся ограничивающих рамок для данного объекта, делая это путем итеративного выбора предсказания с наивысшей оценкой и устранения других предсказаний, которые имеют высокое перекрытие с ним в соответствии с указанной метрикой сопоставления и порогом.

Xywh2xyxy: функция принимает параметр x, который может быть либо тензором torch.Tensor, либо массивом numpy.ndarray. Этот параметр содержит координаты прямоугольников в формате [x, y, w, h], где x, y – координаты центра прямоугольника, а w, h – ширина и высота прямоугольника. Функция возвращает y, который имеет тот же тип, что и вход x, и содержит координаты прямоугольников в формате [x1, y1, x2, y2], где x1, y1 – координаты верхнего левого угла, а x2, y2 – координаты правого нижнего угла. Основная задача функции - конвертировать координаты прямоугольников из формата [x, y, w, h] в формат [x1, y1, x2, y2]. Функция проверяет, является ли входной массив тензором PyTorch или массивом numpy, и создает копию входа для выходных данных. В процессе работы используются операции и методы, такие как torch.Tensor.clone(), который создает копию тензора PyTorch, isinstance(object, classinfo), который проверяет, является ли объект экземпляром указанного класса или кортежа классов, и numpy.copy(arr, order='K'), который возвращает массив-копию указанного объекта.

box_ios: функция вычисляет метрику пересечения относительно меньшего (Intersect Over Smaller – IOS) между двумя ограничивающими прямоугольниками. В качестве аргументов функция принимает два кортежа, box1 и box2, каждый из которых содержит четыре элемента, представляющих координаты ограничивающих прямоугольников (x1, y1, x2, y2). Здесь (x1, y1) – координаты левого верхнего угла, а (x2, y2) - координаты правого нижнего угла каждого прямоугольника. Также функция принимает опциональный аргумент eps, маленькое число, используемое для предотвращения деления на ноль при вычислении метрики IOS, со значением по умолчанию равным 1×10^{-7}. Функция возвращает значение метрики IOS, которое рассчитывается как отношение площади пересечения двух прямоугольников к площади меньшего из прямоугольников. Для выполнения вычислений функция использует стандартные операции и методы Python, не применяя специфические или сторонние функции или методы.

box_iou: функция вычисляет пересечение через объединение (Intersection over Union – IoU), или индекс Жаккара, двух наборов ограничивающих прямоугольников. Аргументы функции включают два тензора: box1 размерности (N, 4) и box2 размерности (M, 4), каждый из которых содержит координаты ограничивающих прямоугольников в формате (x1, y1, x2, y2). Также принимается опциональный аргумент eps, используемый для предотвращения деления на ноль с значением по умолчанию 1×10^{-7}. Функция возвращает тензор iou размерности (N, M), содержащий попарные значения IoU для каждой пары ограничивающих прямоугольников из box1 и box2.

В процессе работы функция использует различные операции и методы из библиотеки PyTorch. В частности, используются функции torch.min() и torch.max() для вычисления минимального и максимального значения каждого элемента из входных тензоров соответственно. Методы тензора, такие как unsqueeze(), chunk(), clamp(), и prod(), также применяются для манипуляции тензорами, включая добавление размерности, разделение на части, обрезку значений и вычисление произведения элементов тензора по указанной размерности.

non_max_suppression: функция принимает на вход следующие аргументы: prediction, который является входным тензором содержащим ограничивающие рамки и прогнозируемые оценки; conf_thres, float значение, по умолчанию равное 0.25, которое является порогом уверенности для фильтрации ограничивающих рамок; iou_thres, float значение, по умолчанию равное 0.45, представляет собой порог Пересечения через Объединение (IoU), используемый для NMS; classes, список по умолчанию равный None, содержит специфические классы для выполнения NMS; agnostic, булево значение, по умолчанию равное False, определяет, будет ли выполняться классно-независимый NMS; multi_label, булево значение, по умолчанию равное False, определяет, будут ли учитываться несколько меток на ограничивающую рамку во время NMS; labels, кортеж по умолчанию пустой, содержит метки для

автоматической маркировки; max_det, целое число, по умолчанию равное 300, определяет максимальное количество детекций, разрешенных на изображение; nm, целое число, по умолчанию равное 0, указывает количество масок.

Функция использует различные операции и внешние методы, такие как torch.zeros для инициализации тензора, заполненного скалярным значением 0; torch.cat для конкатенации данной последовательности тензоров в данном измерении; torchvision.ops.nms для NMS на рамках на основе их IoU; box_iou для вычисления IoU рамок и torch.mm для выполнения умножения матриц. non_max_suppression выполняет NMS на предсказанных ограничивающих рамках, фильтруя ограничивающие рамки на основе порогов уверенности, спецификаций класса и порогов IoU, сохраняя наиболее уверенные прогнозы и удаляя перекрывающиеся рамки. Функция также обрабатывает классно-независимый и многометочный NMS, автоматическую маркировку и может объединять рамки на основе IoU и оценок, возвращая отфильтрованный набор ограничивающих рамок для каждого изображения в пакете.

read_image: функция принимает один аргумент: image_path, который является строковым путем к изображению, которое необходимо прочитать. Для выполнения своих операций функция использует методы torchvision.io.read_file и torchvision.io.decode_jpeg. Первый метод читает файл изображения и возвращает его в форме байтов, второй метод декодирует изображение в формате JPEG и возвращает его в форме тензора. Функция read_image читает изображение из файла, декодирует его и возвращает декодированное изображение в форме тензора на указанном устройстве (CUDA).

preprocess: функция принимает один аргумент: image, который является тензором изображения, подлежащим предобработке. Функция preprocess преобработывает входное изображение, преобразуя его в half precision (float16) и нормализуя значения пикселей, разделив их на 255. Это изменение

диапазона значений пикселей с 0-255 до 0-1, что является главным действием, выполняемым этой функцией в процессе предобработки изображения.

ImageDataset: класс наследуется от Dataset и представляет собой датасет для семантической сегментации изображений. Он содержит методы: init, который инициализирует объект датасета с использованием списка путей к изображениям и утилитой для работы с JPEG, принимает в качестве аргумента список строк, каждая из которых является путем к изображению, и инициализирует объект TurboJPEG для работы с изображениями в формате JPEG; len возвращает количество изображений в датасете; и getitem получает изображение по индексу, декодирует и предобрабатывает его, возвращая тензор изображения и его идентификатор, принимает индекс изображения в качестве аргумента и использует TurboJPEG для декодирования изображения, а также np.ascontiguousarray и torch.from_numpy для преобразования изображения в тензор. Изображения декодируются с использованием библиотеки TurboJPEG, а затем предобрабатываются и конвертируются в тензоры PyTorch.

sliding_inference: функция принимает на вход тензор изображения и применяет к нему модель скользящего окна с использованием метода скользящего окна с перекрытием. Изображение разбивается на несколько окон или патчей, к каждому из которых применяется модель, и затем результаты объединяются с использованием non-max suppression. Функция выполняет следующие операции: паддинг изображения, если его размер не кратен размеру окна, применение модели к каждому патчу, корректировка результатов с учетом положения окон на исходном изображении и применение non-max suppression для удаления дублирующихся или низкокачественных предсказаний. На выходе функция возвращает тензор, содержащий итоговые предсказания модели для входного изображения после применения метода скользящего окна и non-max suppression.

second_sliding_inference: функция принимает тензор изображения в качестве аргумента и применяет второй раунд инференции со скользящим

окном для обнаружения объектов на изображении. Она создает окна с определенным размером и перекрытием, вычисляемыми исходя из размеров изображения, и возможно дополняет изображение для обеспечения полного покрытия скользящими окнами. Каждое окно или фрагмент проходит через модель second_sliding_model для получения прогнозов. Затем применяется NMS для удаления дублирующих прогнозов и сохранения наиболее уверенных прогнозов. Возвращаемый результат – это тензор full_predictions, содержащий координаты ограничивающих рамок, уровни уверенности и классовые метки для обнаруженных объектов. Используемые константы, такие как размер окна, перекрытие и пороги, должны быть предварительно определены или переданы функции в качестве аргументов.

half_image_inference: функция принимает один аргумент: image, который является тензором, представляющим изображение, для которого будет выполнена инференция. Функция выполняет инференцию на уменьшенной копии изображения, изменяя его размер до определенных констант HALF_MODEL_INPUT_HEIGHT и HALF_MODEL_INPUT_WIDTH. Для изменения размера изображения используется интерполяция с использованием функции torch.nn.functional.interpolate. После изменения размера, изображение передается модели для инференции с использованием функции model. Результаты инференции обрабатываются с использованием non_max_suppression, чтобы удалить дубликаты прогнозов и сохранить наиболее вероятные прогнозы. Координаты ограничивающих рамок масштабируются обратно к размерам оригинального изображения. Функция возвращает тензор all_boxes, который содержит скорректированные координаты ограничивающих рамок, а также уровни уверенности и классовые метки. В функции используются константы HALF_MODEL_INPUT_HEIGHT, HALF_MODEL_INPUT_WIDTH, THRESHOLD_FOR_FULLSIZE и IOU_THRES, которые должны быть предварительно определены или

переданы в функцию в качестве аргументов. Также предполагается, что модель model была правильно инициализирована.

full_image_inference: функция принимает в качестве аргумента изображение в формате тензора (image), которое должно иметь определенную форму и размер. В процессе работы функция масштабирует изображение, применяет к нему горизонтальное отражение и выполняет инференцию с использованием полной модели (full_model). После этого к результатам инференции применяется постпроцессинг, включающий NMS с использованием функции non_max_suppression, и координаты ограничивающих рамок масштабируются обратно. В результате функция возвращает тензор all_boxes, содержащий координаты ограничивающих рамок, скорректированные и приведенные в соответствие с оригинальными размерами изображения. В функции используются различные внешние функции и методы, такие как torch.nn.functional.interpolate для изменения размера изображения и предполагается использование внешней модели full_model. Также функция зависит от глобальных переменных, таких как FULL_MODEL_INPUT_HEIGHT, FULL_MODEL_INPUT_WIDTH и THRESHOLD_FOR_FULLSIZE, которые должны быть определены заранее.

small_image_inference: функция принимает тензор изображения (image) с определенной формой и размером в качестве аргумента. Функция изменяет размер этого изображения до размеров, установленных для «маленькой» модели, и выполняет инференцию с использованием «маленькой» модели (small_model). После этого к результатам применяется постпроцессинг, включая NMS, с использованием функции non_max_suppression, и координаты ограничивающих рамок масштабируются обратно к оригинальным размерам изображения. Возвращаемый результат, тензор all_boxes, содержит координаты ограничивающих рамок после масштабирования и применения NMS, приведенные в соответствие с оригинальными размерами изображения. Функция также использует внешние функции и методы, такие как torch.nn.functional.interpolate для изменения размера изображения и

предполагаемую внешнюю модель small_model для инференции. Функция зависит от глобальных переменных, таких как SMALL_MODEL_INPUT_HEIGHT, SMALL_MODEL_INPUT_WIDTH, THRESHOLD_FOR_FULLSIZE и IOU_THRES, которые должны быть определены заранее.

inference: функция, которая принимает аргумент image, тензор, содержащий изображение для обработки. Ожидается, что изображение имеет определенную форму и размер. Функция выполняет инференс, используя несколько методов, таких как обработка маленьких изображений, половина изображения и скользящее окно, и объединяет полученные ограничивающие рамки. Затем применяется жадный не-максимальный метод (greedy_nmm) для дополнительного объединения и выбора окончательных ограничивающих рамок. Функция возвращает тензор final_boxes, который содержит окончательные ограничивающие рамки после применения всех методов и постпроцессинга. Для работы функции необходимы внешние функции и методы, такие как torch.inference_mode(), small_image_inference, half_image_inference, second_sliding_inference, greedy_nmm, has_match, merge_object_prediction_pair, а также глобальные переменные, например, IOU_THRES.

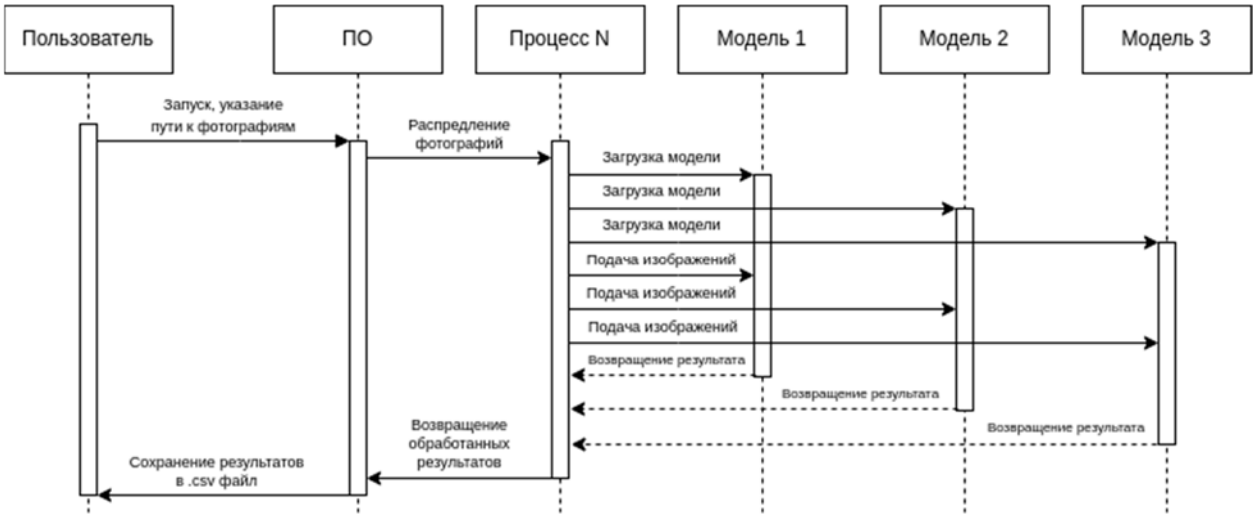
get_yolo_prediction: функция, принимающая два аргумента: image, тензор, содержащий изображение для обработки, и image_id, идентификатор изображения, используемый для связывания результатов с конкретным изображением. Функция получает ограничивающие рамки из функции inference, применяет глобальный порог и нормализует результаты. После этого функция формирует и возвращает список словарей, каждый из которых содержит информацию об одной ограничивающей рамке, включая идентификатор изображения, нормализованные координаты, ширину, высоту, метку и уверенность (score). Функция использует внешнюю функцию inference и ожидает наличие глобальной переменной GLOBAL_THRESHOLD. Также в коде функции могут присутствовать закомментированные строки,

использовавшиеся для отладки, и их можно удалить в окончательной версии функции.

part_inference.main: функция, которая не принимает аргументы. Эта функция предназначена для обработки изображений и получения прогнозов с использованием модели YOLO. Изображения берутся из определенного пути, указанного в TEST_IMAGES_PATH, загружаются и обрабатываются с использованием модели YOLO. Прогнозы, полученные от модели YOLO, сохраняются в DataFrame и экспортируются в CSV-файл. В функции используются следующие внешние функции и методы: Path из модуля pathlib для работы с путями файлов и директорий, ImageDataset - пользовательская функция или класс, служит для создания набора данных изображений, torch.utils.data.DataLoader используется для загрузки и обработки батчей изображений, get_yolo_prediction - пользовательская функция, служит для получения прогнозов от модели YOLO, pd.DataFrame используется для создания DataFrame из результатов прогнозирования, uuid4 из модуля uuid используется для генерации уникального идентификатора, который добавляется к имени экспортируемого CSV-файла. Функция ничего не возвращает, но в процессе её выполнения создается CSV-файл с результатами прогнозирования.

2.2. Диаграмма последовательностей

Диаграмма последовательности программного обеспечения «Система поиска GigaFlex C1 Экстренный поиск» приведена на рисунке:



Пользователь запускает ПО, указывая путь к фотографиям, полученным с БВС DJI MAVIC 3E. ПО принимает фотографии в качестве входных данных, далее распределяет их между заданными процессами (Процесс N). В каждом процессе сначала фотографии загружаются из постоянной памяти в оперативную память, после чего в память видеокарты загружаются три модели нейронных сетей: «best2368_3520», «bestwithneg3520» и «newdataset3648_5504». Каждая модель последовательно обрабатывает поданное на нее изображение. Результаты работы каждой модели собираются в единый список, где они фильтруются и конвертируются в специфический формат, включающий идентификатор изображения, координаты, метки и оценки («image_id», «xc», «yc», «w», «h», «label», «score»). После завершения всех обработок, обработанные данные возвращаются в ПО, которое, в свою очередь, сохраняет их в формате .csv и предоставляет пользователю в качестве выходных данных.

2.3. Диаграмма прецендентов

Диаграмма прецедентов программного обеспечения «Система поиска GigaFlex C1 Экстренный поиск» приведена на рисунке:

Пользователь начинает работу с программой, запуская ее через консоль и указывая путь к фотографиям, полученным с БВС DJI MAVIC 3e. Программа автоматически принимает этот вход, обрабатывает фотографии, деля их между заданным количеством процессов, обычно двумя. Каждый из этих процессов считывает фотографии с диска в оперативную память, после чего загружает в

память видеокарты три модели нейронных сетей: best2368_3520, bestwithneg3520 и newdataset3648_5504.

После загрузки моделей, изображения последовательно передаются через каждую из нейронных сетей для обработки. Полученные результаты собираются в один список, где они фильтруются и конвертируются в определенный формат, соответствующий стандартным процессам YOLO (object detection). В этом формате каждый объект на изображении идентифицируется с помощью «image_id», «xc», «yc», «w», «h», «label», «score». После завершения всех процессов обработки и фильтрации, результаты сохраняются в файл с расширением .csv, который пользователь может использовать для дальнейшего анализа или обработки.

3. Описание архитектурного стиля

В разработке программы был использован стиль кодирования, основанный на PEP 8, что является общепринятым стандартом написания кода на Python. Это включает в себя соблюдение отступов в четыре пробела, ограничение максимальной длины строки 79 символами, а также использование понятных имен переменных и функций, описывающих их назначение и функциональность. Комментарии и строковые документации (docstrings) использовались в некоторых функциях для облегчения понимания кода и его последующей модификации или расширения.

В отношении паттернов проектирования, основной упор сделан на функциональное программирование, учитывая, что основная часть кода организована с использованием функций. Это способствует модулярности и переиспользуемости кода, так как функции могут быть легко заменены или модифицированы без воздействия на остальные части программы. Для управления данными использовался класс Dataset из библиотеки Torch, который является примером применения паттерна «Стратегия», где конкретная реализация алгоритма выбирается на этапе выполнения программы. Это позволяет легко изменять способ обработки и представления данных без изменения остальной части программы.

Также стоит отметить, что некоторые функции могут быть реализованы как чистые функции, что упрощает их тестирование и отладку, а также повышает надежность программы за счет уменьшения побочных эффектов.

Для связи с разработчиками писать на почту ntifundsoft@nti.fund