Описание работы скрипта

СТОРОННИЕ БИБЛИОТЕКИ (numpy, pandas) НЕ ИСПОЛЬЗОВАЛИСЬ

1. Бэкграунд. Нам нужно определить порог confidence для каждого класса одежды, при котором значение f1 score для этого класса будет наибольшим. f1 score предполагает расчет таких параметров, как precision и recall. Т.е. f1 = 2 * precision * recall / (precision + recall). При этом, precision = TP / (TP + FP), а recall – TP / (TP + FN), где
TP = True positive
FP = False positive
FN = False negative
Т.о., просматривая записи лога, нам нужно накопить статистику по этим параметрам для каждого предмета одежды.

2. [Предварительно]. Опишем функцию **iuo**, которая считает площадь перекрытия прямоугольников обнаружения.

3. [Предварительно]. Опишем функцию, которая для заданной строки из лога формирует информацию о предметах, которые реально были на фотографии (ground truth) и предметах, которых обнаружил алгоритм (detected). Результаты будем хранить в виде списка, где в качестве элементов будут словари, содержащие данные об объектах. Примеры: для ground truth запись в списке будет иметь следующий вид: {‘class’:’upper_body’, ‘box’:(222,10,34,14)}. Для detected объектов - {‘class’:’upper_body’, ‘box’:(220,15,40,12), ‘conf’:0.341}. Можно использовать другие структуры, но так нагляднее.

4. [Предварительно]. Опишем функцию, которая фильтрует список detected-объектов - если confidence их обнаружения будет меньше заданной, они удаляются.

5. Опишем основную функцию, которая будет просматривать каждую строку в логе и извлекать необходимые данные о случаях TP, FP, FN. В качестве входного параметра она будет брать значение порога обнаружения.
Работать она должна примерно так:
5.1. Берем строку и получаем списки ground truth и detected согласно пункту 3.
5.2. Фильтруем список detected в соответствии с нужным уровнем confidence.
5.3. [Опционально]. Вероятно, следовало бы отфильтровать список detected, произведя поиск дубликатов обнаружения одних и тех же предметов, а также дубликатов обнаружения несуществующих предметов. Но т. к. критерии такой фильтрации не определены и анализ лога показывает, что фильтрация дубликатов видимо уже производилась на уровне параметров алгоритма детектирования, мы этого делать не будем.
5.4. Получив списки ground truth и detected, составим матрицу следующего вида: номера столбцов будут соответствовать классам ground truth, номера строк — классам detected. Далее, каждой ячейке присвоим значение 1, если классы одинаковы и площадь пересечения больше 0.5. В противном случаем, запишем туда 0. Например:

	Upper_body	**Blouse**	**Jacket**
Upper_body	1 (тот же класс + оверлап > 0.5)	0	0
Upper_body	0 (класс-то тот же, но box обнаружения совсем в другом месте)	0	0
Blouse	0	1	0

Blazer	0	0	0

Из этой таблицы можно извлечь информацию о TP, FP и FN:
- Для каждой строки, если там есть 1, то это TP
- Для каждой строки, если там только нули — это FP
- Для каждого столбца, если там только нули — это FN

Эти данные записываем в словарь, содержащий в качества ключей — наименования каждого класса, а в качестве значений — словарь, содержащий ключи TP, FP, FN. После анализа каждой строки соответствующие ключи увеличиваются.

Просмотрели все строки — записали все, что нужно в словарь. Дальше для каждого ключа в словаре результатов считаем значение f1 с помощью функции, которую тоже отдельно описываем. Возвращаем результаты работы функции в виде еще одно словаря, с ключами — названиями классов и значениям f1 для каждого ключа.

6 Теперь собственно, как на найти ОПТИМАЛЬНОЕ значение уровня confidence для каждого класса предметов одежды, при котором f1 будет максимальным? Предлагается простой вариант — будем запускать функцию, описанную в пункте 5, меняя каждый раз значение порога обнаружения в диапазоне от 0 до 1, с определенным шагом. Результаты f1 для каждого шага будем записывать. А в конце для каждого класса найдем, при каком значении порога f1 было максимально. Каждая итерация будет занимать около 1.5 с. Что вполне приемлемо для 40000 записей. 20 шагов (точность 0.05) будет выполняться за половину минуты.