Модель нейронной сети, датасет.

Детектирование объектов происходит с помощью модели нейронной сети Yolov4 с использованием DL-фреймворка Darknet. Данная модель показывает наилучшие результаты по сравнению с другими моделями нейронных сетей.

В работе использовался датасет COCO.

Описание алгоритма детектирования и замены объекта в файле yolo_image.py

Переменная impath содержит путь до файла, объект в котором нужно детектировать и заменить. Производим инициализацию переменной net, которая будет содержать нейросеть. Функция detect принимает на вход изображение содержащее объект для детектирования и модель нейронной сети.

С помощью метода из библиотеки CV2 считываем изображение с самолётом по пути, которому передали и определяем высоту и ширину картинки. Также с помощью метода из библиотеки CV2 blobFromImage предварительно нормализуем картинку, которую подаём на вход в сеть и приводим её к размеру 416x416 пикселей. Затем прогоняем через нейросеть нашу картинку и записываем в переменную layers_outs полученные данные. Затем выполняем обход полученных данных и записываем их в списки scores, class_id, confidence. confidence содержит значения – то, насколько сеть уверенна что это тот или иной объект, которые мы затем сравниваем со значением переменной CONFIDENCE_THRESHOLD (CONFIDENCE_THRESHOLD = 0.2) – делаем такую проверку из-за особенности модели Yolo, чтобы отсеять побочные объекты. Если сеть уверенна больше чем на 20% в том что на картинке находится объект искомого класса и class_id == 4(т.е. на картинке самолёт), то создаётся бокс, в который также записываем координаты x, y, ширину и высоту.

Метод NMSBoxes устраняет боксы при их наложении, оставляя бокс с наивысшей степенью достоверности. Производим проверку на наличие корректных боксов и записываем его значения в переменные x, y, w, h и производим отрисовку бокса. Результат представлен на рис. 1.

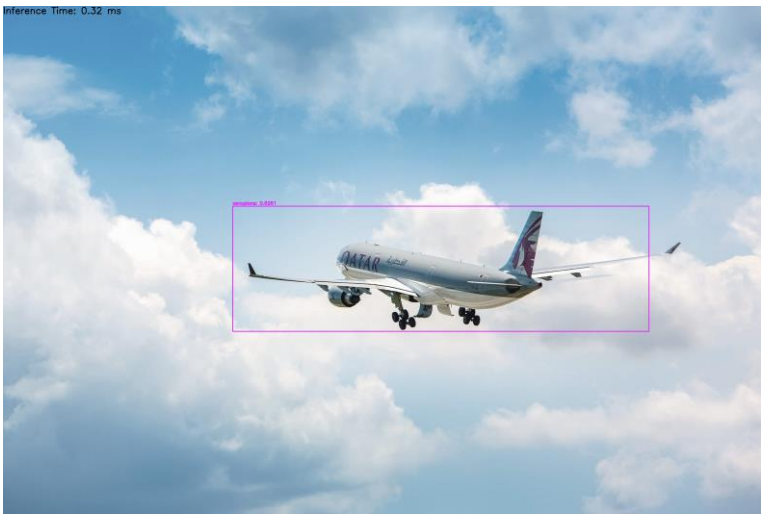

Рисунок 1 – Результат детектирвания искомого объекта

На следующем шаге произведём обработку области, в которой располагался самолёт. Бокс содержащий самолёт заливается чёрным и применяется алгоритм пороговой обработки библиотеки cv2 threshold, в результате получаем переменную th1, в которой хранится изображение с пороговым значением.

Рисунок 2 – Результат применения метода threshold

Затем применяем метод библиотеки cv2 inpaint для восстановления необходимого региона на изображении. Алгоритм inpaint был выбран на основе метода быстрого движения Telea. Алгоритм начинается с границы области, которую нужно закрасить и идет внутрь области, постепенно заполняя все границы сначала. Чтобы окрасить регион, требуется небольшая окрестность вокруг пикселя в окрестности. Этот пиксель заменяется нормализованной взвешенной суммой всех известных пикселей в окрестности. Больший вес придается тем пикселям, которые лежат рядом с точкой, рядом с нормалью границы, и пикселям, лежащим на контурах границы. Как только пиксель окрашен, он перемещается к следующему

ближайшему пикселю с помощью метода быстрого перехода. FMM гарантирует, что пиксели, расположенные рядом с известными пикселями, сначала окрашиваются, поэтому он работает как эвристическая операция вручную.

В результате получим новый фон.

Затем производим создание маски вертолёта. x_centred, y_centred содержат координаты середины бокса.

Описание алгоритма замены объекта:

Выполняются проверки для координат x_centred, y_centred и src_height, src_width (высота и ширина маски, которая содержит объект на который заменяем), inpainted_height, inpainted_width (высота и ширина маски, которая содержит закрашенный фон).

Если от координаты y_centred отнять половину длины маски с вертолётом и она выйдет за верхний край обрабатываемого изображения:

```
if (y_centered - src_height / 2) < 0:
```

То производим смещение центра бокса на столько единиц вниз, насколько маска вертолёта выходит за край обрабатываемого изображения с самолётом.

```
y = int(y_centered + abs(y_centered - src_height / 2))
```

Если к координате y_centred прибавить половину длины маски с вертолётом и она выйдет за нижний край обрабатываемого изображения:

```
elif (y_centered + src_height / 2) > inpainted_height:
```

То производим смещение центра бокса на столько единиц вверх, насколько маска вертолёта выходит за край обрабатываемого изображения с самолётом.

```
y = int(y_centered - (y_centered - src_height / 2))
```

Иначе оставляем координату середины бокса по оси *y* не тронутой. Аналогичное преобразование координат производится по оси *x*, только в том случае рассматриваются выходы за левый и правый край обрабатываемого изображения с самолётом.

После приведения координат к значениям, пригодным для вставки альтернативного объекта, используем алгоритм seamlessClone для вставки маски вертолёта на исходное изображение. Данный алгоритм бесшовного клонирования вставляет маску вертолёта на исходное изображение в область с изменённым после использования метода inpaint фоном, для того чтобы композиция выглядела бесшовной и естественной. Результат выполнения:

Рисунок 3 – Результат выполнения замены методом seamlessClone