Сделайте краткий обзор любой архитектуры для object detection. Проведите анализ: чем отличается выбранная вами архитектура нейронной сети от других? В чём плюсы и минусы данной архитектуры? Какие могут возникнуть трудности при применении этой архитектуры на практике?

Есть массив архитектур нейронных сетей, созданных для определения объектов в основном делятся:

1. на «двухуровневые», такие как RCNN, fast RCNN и faster RCNN, и
2. «одноуровневые», такие как YOLO.

«Двухуровневые» нейронные сети, перечисленные выше, используют так называемые «регионы» на картинке, чтобы определить, находится ли в этом регионе определенный объект.

Пример (для faster RCNN, которая является самой быстрой из перечисленных двухуровневых систем):

1. Подается картинка/кадр на вход
2. Кадр прогоняется через CNN для формирования feature maps Отдельной нейронной сетью
3. Определяются регионы с высокой вероятностью нахождения в них объектов
4. Регионы с помощью RoI pooling сжимаются и подаются в нейронную сеть, определяющую класс объекта в регионах

Два минуса:

1. Они не смотрят на картинку «полностью», а только на отдельные регионы
2. Относительно медленные.

На примере YOLO

Архитектура не имеет двух проблем выше.

Вообще архитектура YOLO в первых блоках не сильно отличается по «логике блоков» от других детекторов, то есть на вход подается картинка, дальше создаются feature maps с помощью CNN (правда в

YOLO используется своя CNN под названием Darknet-53), затем эти feature maps определенным образом анализируются), выдавая на выходе позиции и размеры bounding boxes и классы, которым они принадлежат.

Принцип YOLO:

Шаг 1: Картинки ресайз под размер 416x416 перед началом обучения нейронной сети, чтобы можно было их подавать батчами (для ускорения обучения).

Шаг 2: Делим картинку (пока что мысленно) на клетки на клетки размером 13x13 (тренд).

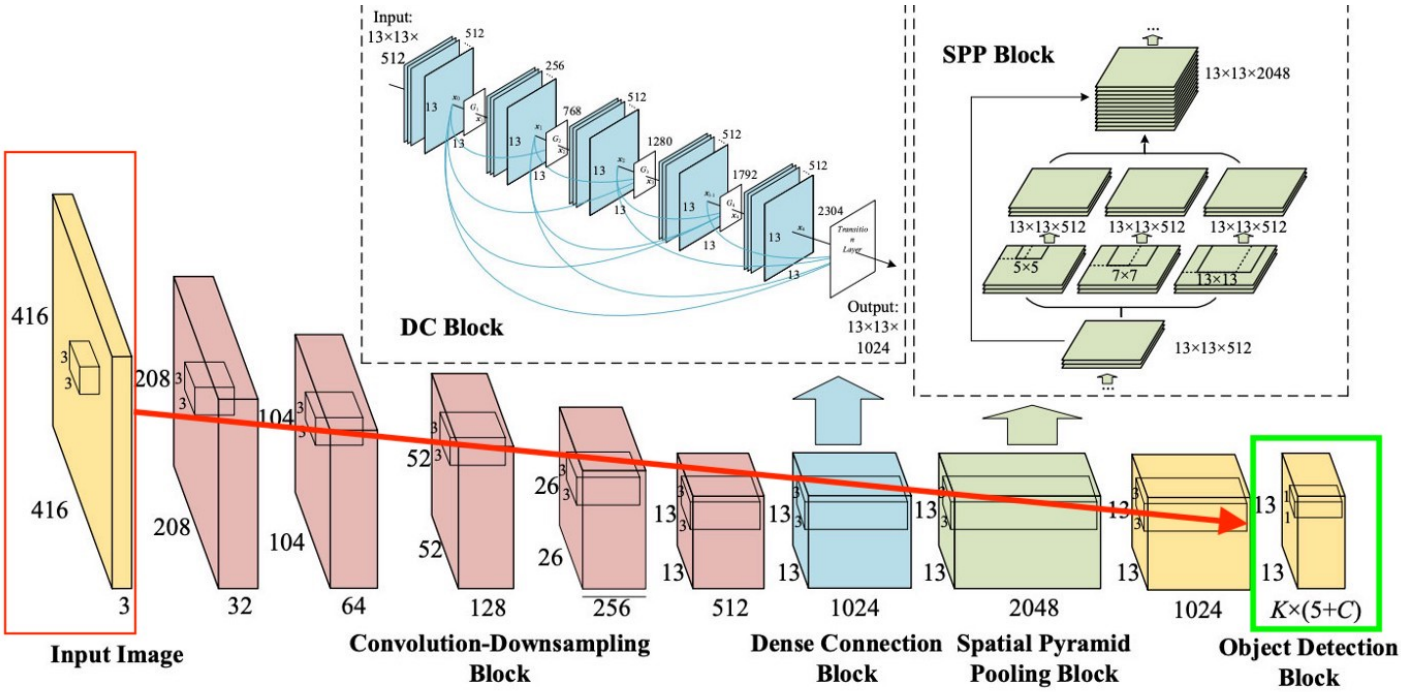
Фокусируемся на эти клеточках, на которые мы разделили картинку/кадр. Такие клетки, которые называются grid cells, лежат в основе идеи YOLO. Каждая клетка является «якорем», к которому прикрепляются bounding boxes. То есть вокруг клетки рисуются несколько прямоугольников для определения объекта (поскольку непонятно, какой формы прямоугольник будет наиболее подходящим, их рисуют сразу несколько и разных форм), и их позиции, ширина и высота вычисляются относительно центра этой клетки.

Они задаются в самом начале либо самим пользователем, либо их размеры определяются исходя из размеров bounding boxes, которые есть в датасете, на котором будет тренироваться YOLO (используется K-means clustering и IoU для определения самых подходящих размеров)

Шаг 3. Картинка из датасета прогоняется через нашу нейронную сеть (картинки в тренировочном датасете должны быть определенны позиции и размеры настоящих bounding boxes для объектов, которые есть на ней. Это называется «аннотация» и делается это в основном вручную).

Для каждой клетки, нужно понять две принципиальные вещи:

Какой из anchor boxes, из 3 нарисованных вокруг клетки, подходит больше всего и как его можно немного подправить для того, чтобы он хорошо вписывал в себя объект.



Подпись

ПРЕИМУЩЕСТВА

Скорость

НЕДОСТАТОК

Скорость за счет качества

Другими словами YOLO лучше, когда мы можем игнорировать небольшую неточность как пример мониторинг трафика в реальном времени.