

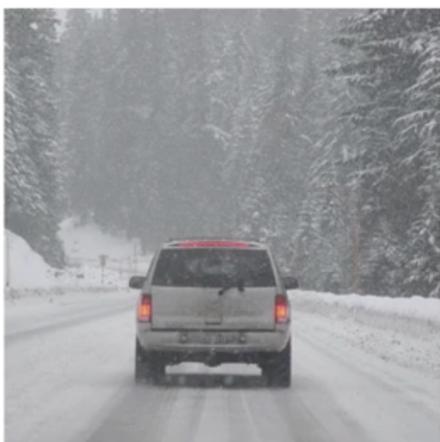
Урок 7.1 Детектирование объектов. Bounding box prediction. Детектор YOLO

Recap На прошлой неделе мы рассмотрели типичные архитектуры сверточных нейронных сетей: **LeNet**, **AlexNet**, **VGG-16**. Помимо задачи классификации изображений, мы рассмотрели задачу сегментации и архитектуру **UNet**. Также познакомились с основной идеи нейросети **ResNet**, а именно т.н. **Skip connection**.

На этом семинаре мы рассмотрим детектирование и локализацию объектов на изображении (bounding box prediction). Эта задача относится к задачи классификации, но не попиксельной (сегментация), а региональной. Мы указываем рамку, внутри которой находится объект заданного класса.

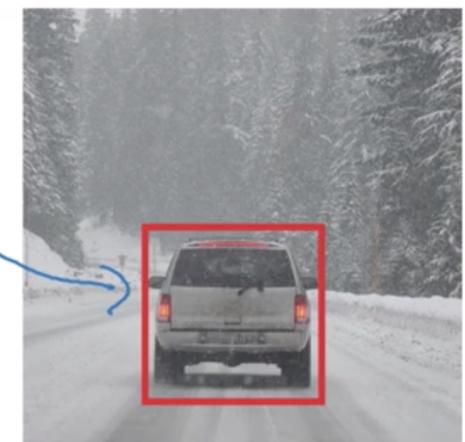
Локализация объектов

Image classification



"Car"

Classification with localization



"Car"

Сегодняшняя задача заключается в классификации объектов на изображении и указании координат этих объектов (см. рисунок сверху).

Используемая модель

От нашей модели требуется провести две операции:

- классифицировать объекты на изображении;
- указать координаты этих объектов.

Для начала рассмотрим, как будет выглядеть выходной вектор для поставленной задачи, а потом поймем, как получить такой вектор:

$$\hat{y}_{out} = (c_1, c_2, \dots, c_n, bx_1, bx_2, by_1, by_2)^T,$$

где c_i – вероятность, что на изображении объект i -ого класса, b_x, b_y – координаты этого объекта (обводящей рамки).

- Замечание* Мы пока считаем, что на изображении должен присутствовать РОВНО один объект, который надо локализовать и классифицировать.

Если мы не знаем, есть ли на изображении интересующий нас объект, то в выходной вектор надо добавить еще одну координату (бинарную), которая будет отвечать за то, есть ли на изображении объект или нет.

Функция потерь

Если бинарная координата равна единице, то

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^N (\hat{y}_i - y_i)^2,$$

если бинарная координата равна нулю, то

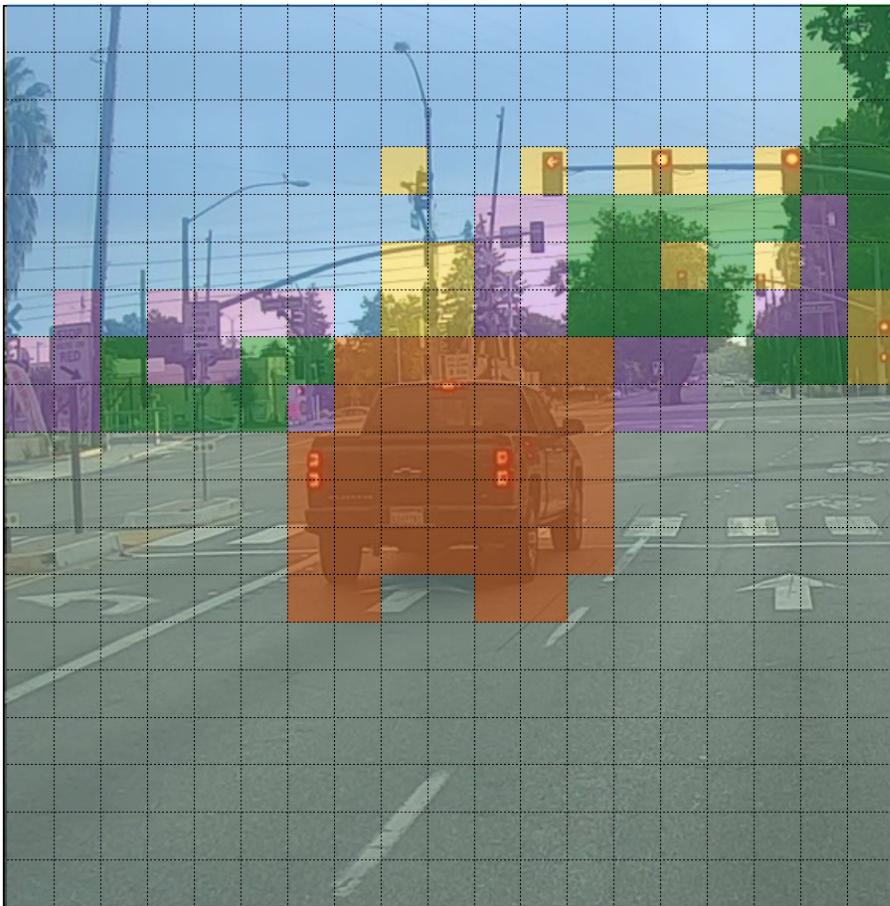
$$\mathcal{L}(y, \hat{y}) = (\hat{y}_0 - y_0)^2.$$

▼ Детектирование объектов

Если на изображении может присутствовать несколько объектов разных классов, то задача локализации и классификации всех этих объектов называется задачей детектирования.

Sliding window detection

Самый простой подход - обучить классификатор (например, на основе CNN), пройтись по маленьким участкам изображения, после чего отдельно отклассифицировать каждый такой участок.

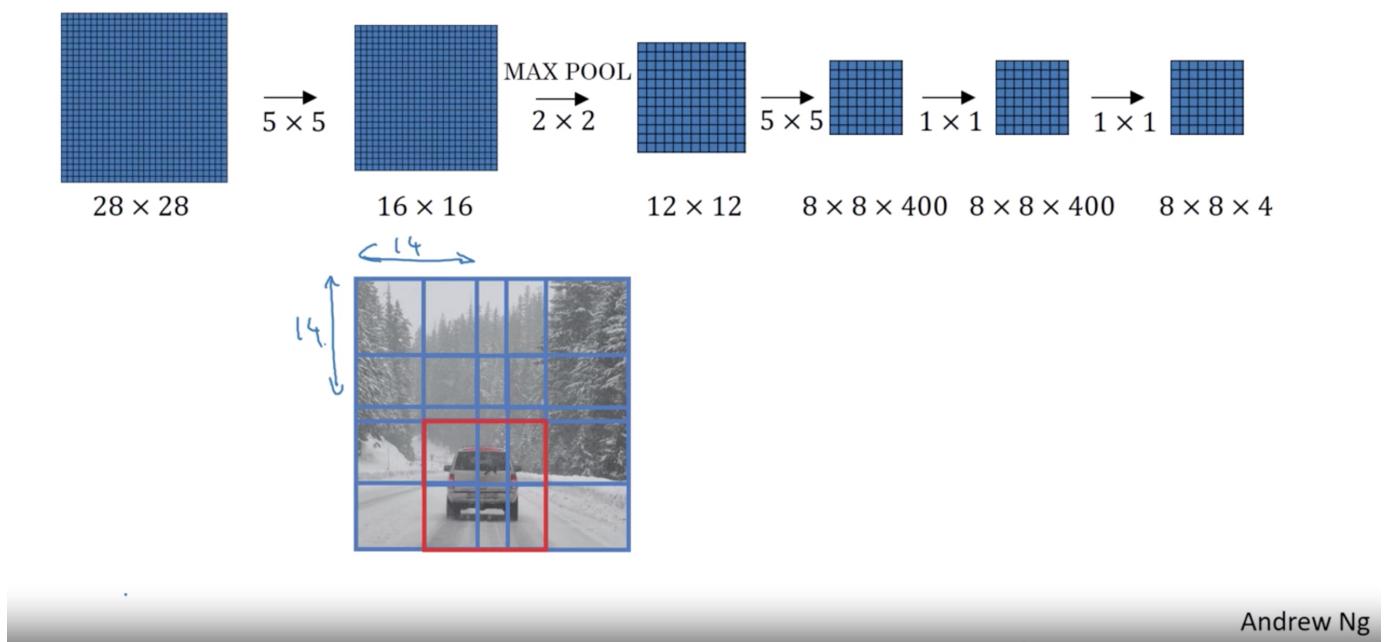


Основная проблема такого подхода - это вычислительная сложность и то, что объект может "вылезать" за пределы одного квадратика.

Использование CNN вместо скользящего окна

Как мы помним, при применении операции свертки, можно уменьшать размерность изображения (в сочетании с пулингом). На этом факте основано применение сверточной нейросети для детектирования объектов: мы в качестве последнего слоя выбираем не fc-слой, а сверточный низкой размерности, в котором каждая координата отвечает соответствующему региону на входном изображении, а значение этих координат - классу, к которому относится тот регион (точнее, объекты в том регионе).

Convolution implementation of sliding windows



Andrew Ng

Поиск обводящей рамки (bounding box prediction)

К сожалению, нет никакой гарантии, что интересующий нас объект влезет в выбранный регион, причем только в него. Поэтому необходимо менять модель.

Алгоритм YOLO

Paper: "[You Only Look Once: Unified, Real-Time Object Detection by J. Redmon et al. 2016](#)"

YOLO (you only look once) - алгоритм, который заключается в следующем:

- разбиваем изображения на фрагменты 19×19 (оригинальное исполнение), после чего применяем классификатор с описанным выходным вектором y_{out} ;
- каждый объект на изображении относим к тому или иному фрагменту (объект попадает в ту ячейку, куда попадает его центр);
- применяем КО ВСЕМУ изображению сверточную нейросеть, на выходе получаем тензор размера $19 \times 19 \times 8$.

YOLO algorithm



Для ячеек с объектами (левая и правая) мы формируем соответствующие векторы y_{out} , бинарная координата у них будет равна единице:

$$y_{out}^1 = (1, bx_1^1, bx_2^1, by_1^1, by_2^1, 0, 1, 0)^T$$
$$y_{out}^2 = (1, bx_1^2, bx_2^2, by_1^2, by_2^2, 0, 1, 0)^T.$$

Последние три координаты отвечают за класс объекта (в данном случае - автомобиль).

Поиск bounding box. Intersection over union

В алгоритме YOLO обводящая рамка кодируется 4-мя координатами следующим образом: $(b_{xcent}, b_{ycent}, b_h, b_w)$, причем каждая координата отсчитывается в относительных координатах и нормирована на размер фрагмента.

Метрика, которая часто используется в подобных задачах, называется IoU (intersection over union):

$$IoU(A, B) = \frac{\mu(A \cap B)}{\mu(A \cup B)},$$

то есть отношение площади пересечения к площади объединения. В качестве площадей используются площади истинной рамки и предсказанной.

Предсказание считается верным, если $IoU > \text{threshold}$ (например, 0.5).

Non - max suppression

Возможна проблема - это то, что один объект может быть обведен в рамки несколько раз. Чтобы побороться с этой проблемой, используется non-max suppression: если есть много пересекающихся рамок (overlapping frames) одного класса, то

- выбирается та, у которой вероятность обнаружения объекта этого класса наибольшая;
- ищем рамку, отвечающую тому же классу, у которой IoU с первой максимальен;
- удаляем остальные рамки с высоким IoU (выше порога).

Функция потерь алгоритма YOLO

Функция потерь в алгоритме YOLO выглядит следующим образом:

$$\begin{aligned} \mathcal{L}(y, \hat{y}) = & \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B I_{ij}^{obj} [(\sqrt{w}_i - \sqrt{\hat{w}_i})^2 \\ & \sum_{j=0}^B I_{ij}^{noobj} (C_i - \hat{C}_i)^2 + \sum_{i=0}^{S^2} \sum_{c \in classes} I_{ij}^{noobj} (p_i(c) - \hat{p}_i(c))^2 \end{aligned}$$

