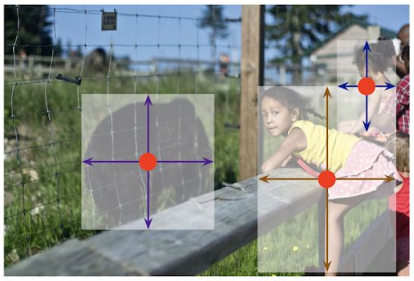
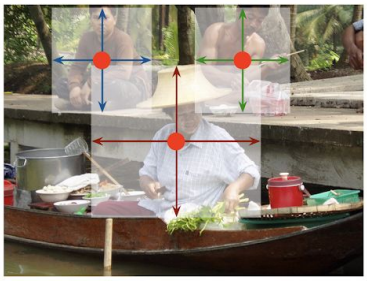
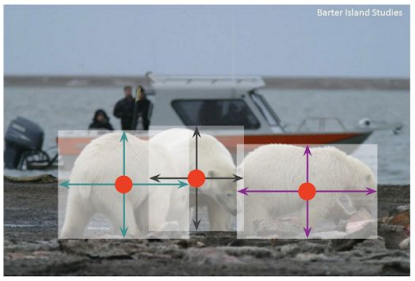
Objects as points: paper summary

Введение

Задача детекции широко используется в компьютерном зрении: в трекинге, детекции поз, и т. д. Однако большинство алгоритмов сейчас используют подход, в котором алгоритмами предлагается набор баундинг боксов, а затем из них выбираются и корректируются уже предсказания для каждого класса.

В данной статье этому подходу предлагается хорошая альтернатива. Каждый объект представляется как точка в центре своего баундинг бокса, и вся необходимая информация: размер объекта, класс и т. д. получаются из регрессии признаков именно в этой точке. То есть, задачу детекции сводят к нахождению точек.

Объекты как точки

Итак, на вход нейросети подается входное изображение $I \in R^{W \times H \times 3}$. Тогда выходом будет хитмап ключевых точек $Y^* \in [0, 1]^{W/R \times H/R \times C}$, где R - шаг выхода сети относительно входа, а C - число классов точек. $Y^*_{x, y, c} = 1$ соответствует найденной точке, а 0 - фону.

Для тренировки нейросети каждой точке из обучающей выборки $p \in R^2$ сопоставляется $p^* = p / R$, округленная вниз до ближайшего целого. Затем, чтобы из набора точек получить хитмап $Y \in [0, 1]^{W/R \times H/R \times C}$, все точки размываются по гауссиане с σ_p берется пропорционально размеру баундинг бокса. Если две гауссианы пересекаются, берется их максимум.

$$Y_{xyc} = \exp\left(-\frac{(x-\tilde{p}_x)^2+(y-\tilde{p}_y)^2}{2\sigma_p^2}\right)$$

Тренируем попиксельную логистическую регрессию с функцией потерь - focal loss, N - число точек на картинке, a и b - параметры focal loss:

$$L_k = \frac{-1}{N}\sum_{xyc}\begin{cases}(1-\hat{Y}_{xyc})^{\alpha}\log(\hat{Y}_{xyc}) & \text{if } Y_{xyc}=1\\ (1-Y_{xyc})^{\beta}(\hat{Y}_{xyc})^{\alpha} & \\ \log(1-\hat{Y}_{xyc}) & \text{otherwise}\end{cases}$$

Также, из-за дискретизации мы имеем небольшую ошибку из-за того, что размер выхода сети в R раз меньше размера входа. Для того, чтобы ее уменьшить, будем параллельно предсказывать отклонение $O^* \in R^{W/R \times H/R \times 2}$ с функцией потерь:

$$L_{off} = \frac{1}{N} \sum_{p} \left| \hat{O}_{\tilde{p}} - \left(\frac{p}{R} - \tilde{p} \right) \right|.$$

Для каждого баундинг бокса (x_1^k, x_2^k, y_1^k, y_2^k) сопоставим точку его центра: p_k как полусумму его минимальных и максимальных координат. Будем использовать Y^* для предсказания этих центров. Также с помощью регрессии будем находить размер баундинг боксов $s_k = (x_2^k - x_1^k, y_2^k - y_1^k)$, причем для всех классов в конкретной точке будем предсказывать один и тот же s_k. Функция потерь:

$$L_{size} = \frac{1}{N} \sum_{k=1}^{N} \left| \hat{S}_{p_k} - s_k \right|.$$

Итого, функция потерь полностью для нашего детектора:

$$L_{det} = L_k + \lambda_{size} L_{size} + \lambda_{off} L_{off}.$$

Инференс

Во время инференса мы сначала находим локальные максимумы на хитмапах отдельно для каждого класса - значения, которые больше всех своих 8 соседних пикселей. Берем топ 100 таких пиков. Для каждой найденной точки центра мы используем $Y^*_{x,\,y,\,c}$ как меру уверенности, а координаты баундинг бокса получаем по формуле:

$$(\hat{x}_i + \delta\hat{x}_i - \hat{w}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i - \hat{h}_i/2,$$
$$\hat{x}_i + \delta\hat{x}_i + \hat{w}_i/2,\ \ \hat{y}_i + \delta\hat{y}_i + \hat{h}_i/2),$$

где дельты, w и h берется из предсказанных сетью O^* и S^*.

Важно заметить, что поиск локальных максимумов на хитмапах является эффективной заменой NMS и может быстро считаться на GPU с помощью 3x3 max pooling операций.

Заключение

Таким образом, в данной статье представлена идея представления объектов в качестве точек в их центрах. Алгоритм получился очень простой, но в то же время эффективный и точный, без лишних постпроцессингов. Также, он может решать параллельно в один проход задачи нахождения поз, 3D ориентации, оценки глубины и др.