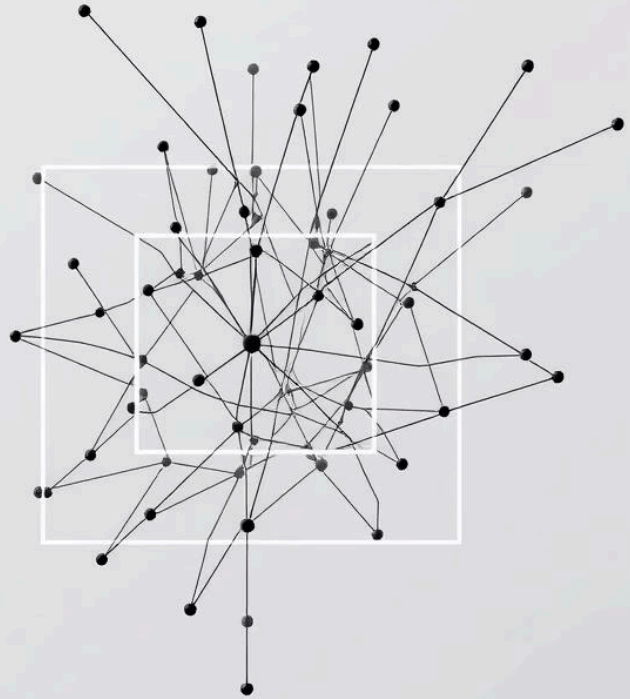
Оптимизация точности малых моделей детекции объектов

Плугин Олег 22503



Оптимизация точности малых моделей детекции объектов

Необходимо найти баланс между высокой точностью и эффективным использованием ресурсов

Постановка задачи: обнаружение объектов на ограниченных ресурсах

В задачах компьютерного зрения для мобильных и встроенных устройств, критически важно обеспечить высокую точность обнаружения объектов при сохранении скорости работы в реальном времени и минимальных вычислительных затратах.

Типовые Ограничения Устройств

Процессор

4-ядерный, тактовая частота 2.2 ГГц.

Оперативная память

До 4 ГБ ОЗУ.

Графический процессор

Обработка изображений размером до 1920 × 1080. Оптимизация под мобильные GPU-архитектуры.

Проблема и Цель Оптимизации

При таких аппаратных ограничениях использование крупных нейронных сетей (например, стандартных версий YOLO или Faster R-CNN) становится невозможным. Требуется упрощение архитектуры:

- Уменьшение числа слоев и их глубины.
- Сокращение количества фильтров (каналов).
- Снижение общего числа обучаемых параметров.

Основная проблема: Упрощение архитектуры нейронной сети приводит к значительному снижению точности (mAP) распознавания.

Ключевой целью является достижение точности, максимально приближенной к исходной, более крупной модели, при существенно меньшем размере панели и более высокой скорости работы.

Параметры задачи

Категория	Основные параметры	Влияет на
Архитектура сети	количество слоёв n, шаг карты признаков (stride feature map) $feat_{st}$, количество фильтров f	скорость работы (FPS) и точность (mAP)
Входные данные	размер входного изображения (W, H)	точность распознавания
Якори (anchors)	количество якорных рамок k и их размеры (w_a, h_a)	качество локализации объектов и IoU
Метрики оценки	mAP, $Precision$, $Recall$, IoU, FPS, T_1(время обработки одного кадра)	позволяет оценить скорость и точность работы модели

Обозначение метрик и параметров

- **Pooling** - операция уменьшения размера карты признаков, сохраняя наиболее важную информацию
- **Precision** - показывает, какая доля найденных моделью объектов действительно является корректной (в процентах или от 0 до 1)
- **Recall** - показывает, какая доля реальных объектов была найдена моделью (в процентах или от 0 до 1)
- **IoU (Intersection over Union)** - показывает, насколько точно предсказанные области покрывают исходные (от 0 до 1)
- **mAP (mean Average Precision)** - среднее значение точности по всем классам и порогам IoU. Основная метрика для сравнения моделей (в процентах)
- **FPS (frames per second)** - количество обрабатываемых кадров в секунду, показывает производительность (кадры/сек)

Формулировка задачи оптимизации

Данную задачу можно представить в виде задачи оптимизации.

Пусть существуют параметры:

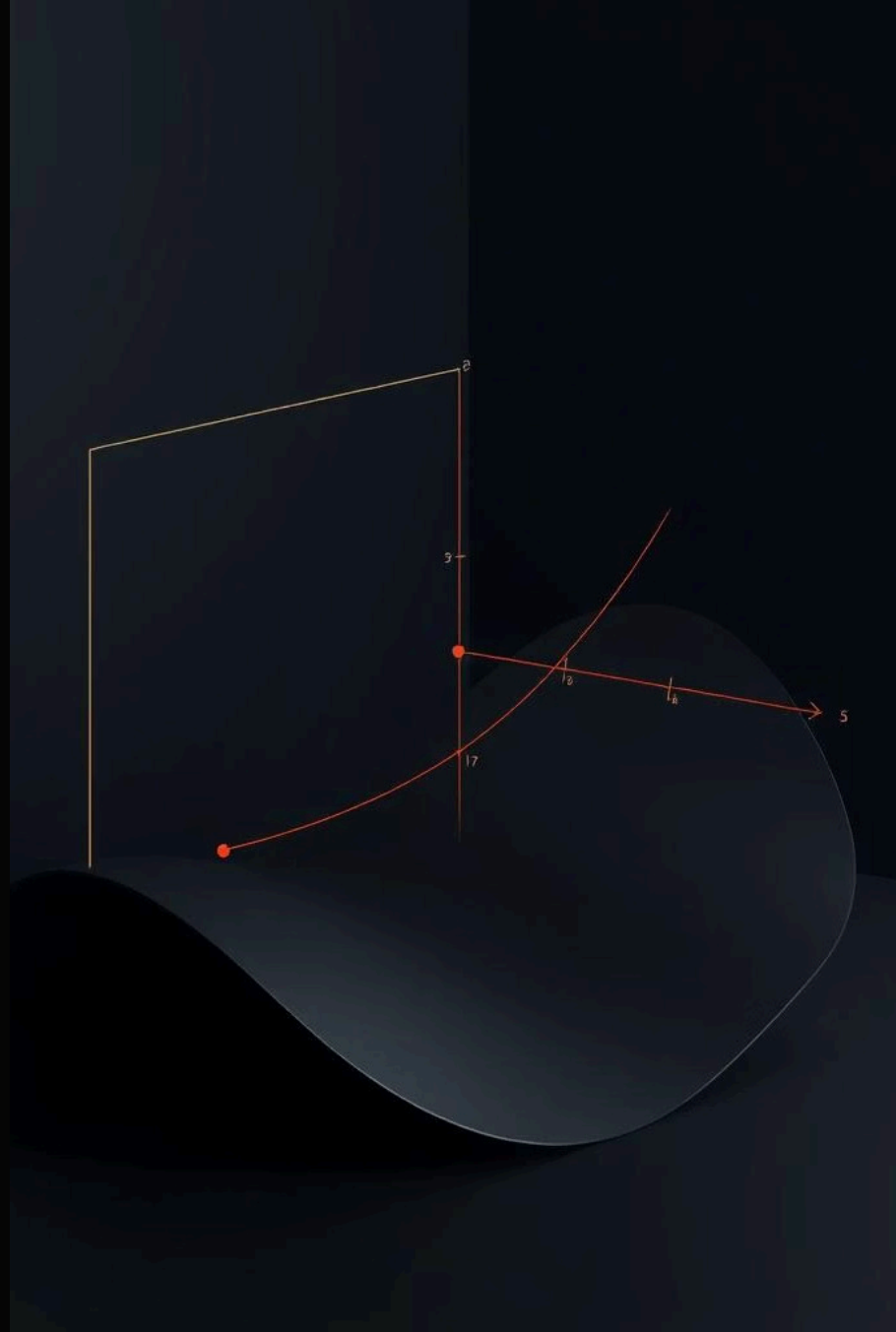
- W - ширина входного изображения (пиксели)
- H - высота входного изображения (пиксели)
- n - количество слоёв нейронной сети
- k - количество якорных рамок

Необходимо набрать такой набор параметров (W, H, n, k) при котором:

- точность $mAP \rightarrow$ max
- время обработки $T_1 \rightarrow$ min, т.е. частота кадров $FPS \rightarrow$ max (т.к. $FPS = 1/T_1$)

при выполнении следующих ограничений:

- $FPS(W, H, n, k) \geq FPS_{min}$
- $n < n_{max}$



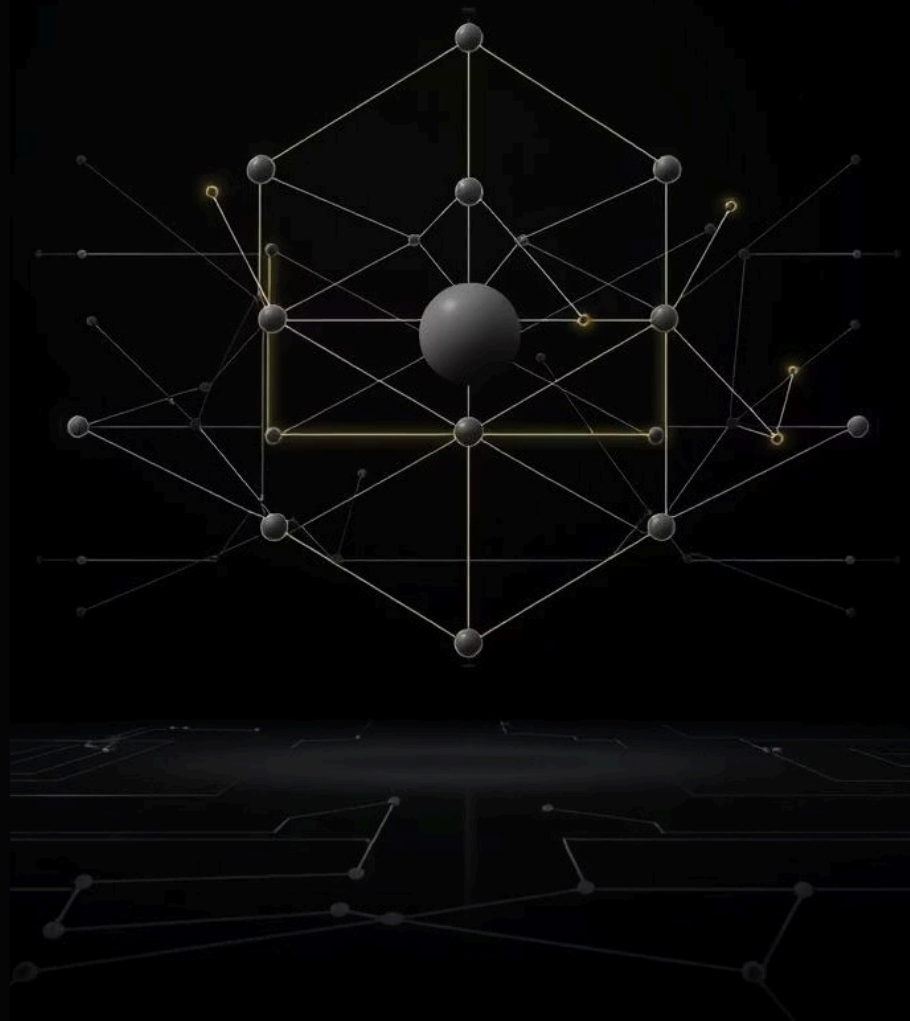
Алгоритм оптимизации сети: Анализ архитектуры и изменение входных данных

Анализ архитектуры

- Зафиксировать начальное количество слоев как n_0
- Уменьшить количество слоев до $n < n_0$

Изменение входных данных

1. Изменить размер входного изображения $W_n \times H_n$
2. Оценить изменения mAP и FPS
3. Отметить зависимость: $mAP \propto f_1(W_n, H_n)$, $T_1 \propto f_2(W_n, H_n)$, $FPS = 1/T_1$ (T_1 — время обработки обратно пропорционально частоте кадров). f_1 — функция, описывающая зависимость точности от разрешения входного изображения, f_2 — функция, описывающая зависимость времени от разрешения входного изображения.



Алгоритм оптимизации сети: Оптимизация якорных рамок

$$(w_r, h_r) = (W_n / W_i \times w_t, H_n / H_i \times h_t)$$

Формула обеспечивает пропорциональное масштабирование якорных рамок под новый размер.

$$(w_c, h_c) = k - means(w_r, h_r)$$

$$\bar{S}(k) = 1/n \sum_{i=1}^{n} S(i)$$

$$S_k = (b - a)/max(a, b),$$

1. Масштабировать рамки по формуле:
   - (w_t, h_t) - граничные рамки исходного значения (пиксели)
   - (w_r, h_r) - ремасштабированные рамки (пиксели)
   - (W_n, H_n) - исходное разрешение входа сети (пиксели)
   - (W_i, H_i) - ширина и высота входного изображения (пиксели)
2. Провести кластеризацию k-средних по ширине и высоте рамок:
   - (w_c, h_c) - k центров кластеров, соответствующие усредненным размерам объектов
3. Найти оптимальное количество кластеров k с помощью силуэта средних коэффициентов:
   - Проэкспериментировать со значением k от 3 до 15
   - При каждой итерации вычислять средний силуэт по всем точкам:
     - n - общее количество элементов, участвующих в кластеризации
     - a - среднее внутрикластерное расстояние,
     - b - ближайшее межкластерное расстояние
   - Максимум среднего $\bar{S}(k)$ определит оптимальное значение k

Алгоритм оптимизации сети: Нормализация якорей и оценка результатов

Нормализация якорей

Нормализуем размеры якорных рамок

$$(w_a, h_a) = (w_c / feat_{st}, h_c / feat_{st})$$

- w_a, h_a - размеры рамок в масштабе feature map, то есть в тех единицах, с которыми работает последний слой модели
- $feat_{st} = 2^n$ - шаг карты признаков, где сеть содержит n слоев max pooling

Это необходимо, чтобы размер якорей соответствовал масштабу признаков на последнем уровне сети.

Оценка результатов

1. Измерить метрики $Precision$, $Recall$, mAP, FPS, IoU
2. Проанализировать зависимости параметров
   - увеличение W_n, H_n → рост mAP, падение FPS
   - увеличение k → рост IoU
   - уменьшение n → рост FPS, падение точности mAP
3. Сделать выводы, удовлетворяют ли текущая конфигурация и метрики целевым ограничениям

Заключение

Представленный подход позволяет подбирать архитектурные и входные параметры модели под доступные вычислительные ресурсы.

Высокая точность

Достижение высокой точности распознавания [1].

Оптимизированная сложность

Значительно меньшая сложность модели, что важно для систем компьютерного зрения, работающих на мобильных и малопроизводительных устройствах.

Источники

1. Tarek Teama, Hongbin Ma, Ali Maher, and Mohamed A. Kassab. 2019. Real Time Object Detection Based on Deep Neural Network. In Intelligent Robotics and Applications: 12th International Conference, ICIRA 2019, Shenyang, China, August 8–11, 2019, Proceedings, Part IV. Springer-Verlag, Berlin, Heidelberg, 493–504.