Annex II

The computational complexity of the YOLO model is a summation of the complexity of individual layers L, i.e.

$$C = \sum_{i=1}^{N} C(L_i)$$

As YOLO is a combination of different types of layers, their types and complexities are discussed below:

Convolutional Layers

Routing Layers

Shortcut Layers

Upsampling Layers

YOLO Layers

0.1 Convolutional Layers

In convolutional layers, the convolution operation is followed by an activation operation. Now in the convolution operation, for each element of an input feature map $I \in \mathbb{R}^{H \times W \times C}$, convolution with a filter $K \in \mathbb{R}^{H_k \times W_k \times C}$ is defined as:

$$\phi(m,n) = \sum_{i=0}^{H_k-1} \sum_{j=0}^{W_k-1} \sum_{n=0}^{C} I_n(m+i, n+j) \cdot K_n(i,j) \tag{1}$$

where $H, W,$ and C denote the height, width, and number of channels of the input feature map, respectively, and H_k and W_k represent the height and width of the filter kernel.

Each convolution operation combines multiplication and summation. Thus, for an input I convolved with N filters, the computational complexity in the ith layer of a model is:

$$C(CL_i) = 2 \times (H \times W \times C \times H_k \times W_k) \times N \tag{2}$$

Now, in convolutional layers, mostly used activation function is Leaky Rectified Linear Unit or ReLU which can be mathematically expressed as:

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha x, & x < 0 \end{cases} \tag{3}$$

where α is a small constant for negative values. It takes arround 3 FLOPS (one comparison, one multiplication of α with negative values, and one selection)

Thus, the computational complexity for the activation function in the ith convolutional layer is

$$C(AL_i) = 3 \times H \times W \times C \times N \tag{4}$$

Thus, total computational cost of ith convolutional layer is :

$$C(ConL_i) = C(CL_i) + C(AL_i) \tag{5}$$

0.2 Routing, Shortcut, and Upsampling Layers

Routing and Shortcut layers do not contribute directly to computational complexity, as they simply add outputs from previous layers without extra computation. In the Nearest Neighbor Upsampling method, pixel values from the input tensor are duplicated to form a larger spatial grid. This process does not involve floating-point arithmetic and is purely a memory operation.

0.3 Prediction Layers

The design of the final output (or prediction) layers varies across different detectors. In this paper, we use the YOLOv3 model; therefore, the computational complexity of its prediction layers is calculated in terms of FLOPs (Floating Point Operations). The major operations performed by the layer are considered as follows:

1. Bounding Box Prediction

For each anchor in each grid cell, the model predicts:

- 4 bounding box attributes: center (x, y), width w, and height h.
- 1 objectness score.
- C class probabilities.

Total predictions per anchor:

$$P_{\text{per anchor}} = 5 + C$$

2. Grid and Anchor Coverage

Total predictions across all anchors and grid cells:

$$P_{\text{total}} = H \times W \times A \times P_{\text{per anchor}}$$

Where:

- $H \times W$: Grid size (e.g., 13×13).
- A: Number of anchors (e.g., 3, as per mask configuration).

3. Floating Point Operations

a. Sigmoid Activation

Applied to each prediction (objectness + class probabilities).
computational complexity for sigmoid activation function of the:

$$C(AL_j) = P_{\text{total}} \times \text{Cost}_{\text{sigmoid}} \tag{6}$$

Sigmoid approximately requires 4 FLOPs (1 exponential + 1 division), so:

$$C(AL_j) \approx P_{\text{total}} \times 4 \tag{7}$$

b. Bounding Box Transformation

Each bounding box requires additional operations to compute:

- Scaled offsets (e.g., $b_x = \sigma(t_x) + c_x$)
- Scaling widths and heights by anchor sizes

computational complexity per bounding box transformation:

$$C(TL_j) = H \times W \times A \times 4 \tag{8}$$

Combining all operations, the computational complexity of the jth prediction layer can be written as :

$$C(PredL_j) = C(AL_j) + C(TL_j) \tag{9}$$

0.4 Combined Computational Complexity

Combining all the computational complexity costs of all the layers, the total computational complexity of a detector branch can be summarized as

$$C(B) = \sum_{i=1}^{X}(C(CL_i) + C(AL_i)) + \sum_{j=1}^{Y}(C(AL_j) + C(TL_j)) \tag{10}$$