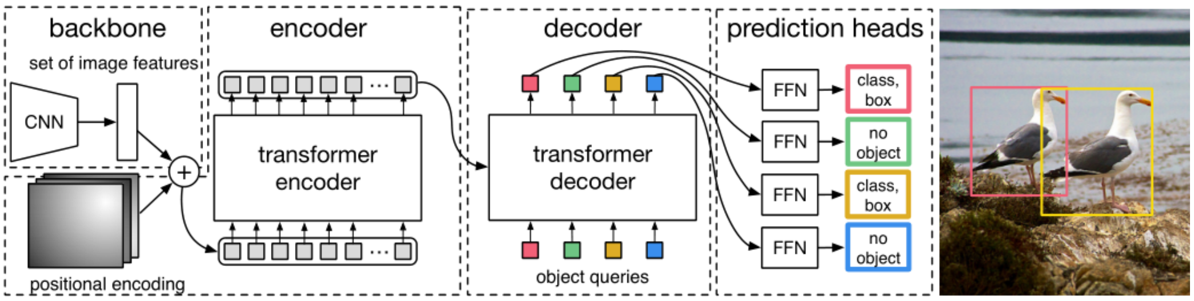
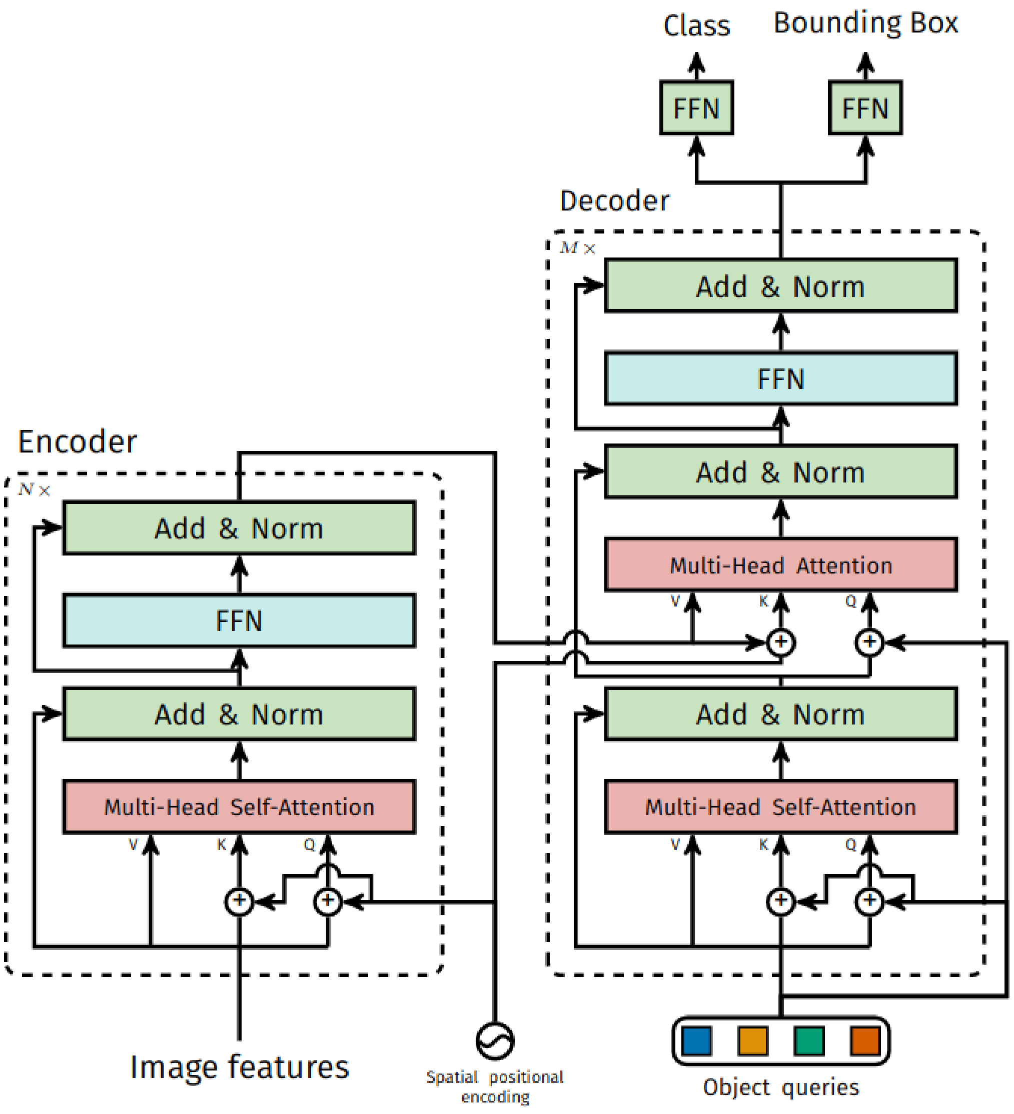
DETR





The detailed description of the transformer used in DETR, with positional encodings passed at every attention layer, is given in Fig. 10. Image features from the CNN backbone are passed through the transformer encoder, together with spatial positional encoding that are added to queries and keys at every multi-head self-attention layer. Then, the decoder receives queries (initially set to zero), output positional encoding (object queries), and encoder memory, and produces the final set of predicted class labels and bounding boxes through multiple multi-head self-attention and decoder-encoder attention. The first self-attention layer in the first decoder layer can be skipped.

DETR 的 cross-attention 有三个输入：**query, key, value**。

1. Query 由来自 decoder 中 self-attention 的输出 (content query: c_q) 和所有图片共享的 object query (spatial query: p_q, 在DETR中其实就是 object query o_q) 相加得到。
2. Key 由来自 encoder 的输出 (content key: c_k) 和对于 2D 坐标的位置编码 (spatial key: p_k) 相加得到。Value 的组成和 key 相同。

在这里，`content` 代表这个向量的内容和图像 (颜色、纹理等) 是相关的，而 `spatial` 代表这个向量它更多包含空间上的信息，他的内容和图像的内容无关。Attention 模块的输出，就是对 query 和 key 算一次内积得到注意力的权重，用这个权重给 value 进行加权。我们将这个过程写成下面的形式：

$$
\begin{aligned}
&(\mathbf{c}_q+\mathbf{p}_q)^{\mathrm{T}}(\mathbf{c}_k+\mathbf{p}_k) \\
&=\mathbf{c}_q^{\mathrm{T}}\mathbf{c}_k+\mathbf{c}_q^{\mathrm{T}}\mathbf{p}_k+\mathbf{p}_q^{\mathrm{T}}\mathbf{c}_k+\mathbf{p}_q^{\mathrm{T}}\mathbf{p}_k \\
&=\mathbf{c}_q^{\mathrm{T}}\mathbf{c}_k+\mathbf{c}_q^{\mathrm{T}}\mathbf{p}_k+\mathbf{o}_q^{\mathrm{T}}\mathbf{c}_k+\mathbf{o}_q^{\mathrm{T}}\mathbf{p}_k
\end{aligned}
\tag{1}
$$

Head of query-based has 2 steps:

1. **Label assignment**

DETR:

$$\hat{\sigma}=\underset{\sigma\in\mathfrak{S}_N}{\arg\min}\sum_{i}^{N}\mathcal{L}_{\mathrm{match}}(y_i,\hat{y}_{\sigma(i)}), \tag{1}$$

where $\mathcal{L}_{\mathrm{match}}(y_i,\hat{y}_{\sigma(i)})$ is a pair-wise *matching cost* between ground truth y_i and a prediction with index $\sigma(i)$. This optimal assignment is computed efficiently with the Hungarian algorithm, following prior work (*e.g.* [43]).

$$\mathcal{L}_{match}(y_i,\hat{y}_{\sigma(i)})=-\mathbb{I}_{\{c_i\neq\varnothing\}}\hat{p}_i(c_i)+\mathbb{I}_{\{c_i\neq\varnothing\}}\mathcal{L}_{box}(b_i,\hat{b}_{\sigma(i)})$$

2. **Compute loss**

DETR:

$$\mathcal{L}_{\mathrm{Hungarian}}(y,\hat{y})=\sum_{i=1}^{N}\left[-\log\hat{p}_{\hat{\sigma}(i)}(c_i)+\mathbb{1}_{\{c_i\neq\varnothing\}}\mathcal{L}_{\mathrm{box}}(b_i,\hat{b}_{\hat{\sigma}}(i))\right]$$

Box loss Similarly to [41,36], we use a soft version of Intersection over Union in our loss, together with a ℓ_1 loss on $\hat{b}$:

$$\mathcal{L}_{\mathrm{box}}(b_{\sigma(i)},\hat{b}_i)=\lambda_{\mathrm{iou}}\mathcal{L}_{\mathrm{iou}}(b_{\sigma(i)},\hat{b}_i)+\lambda_{\mathrm{L1}}||b_{\sigma(i)}-\hat{b}_i||_1\,, \tag{9}$$

where $\lambda_{\mathrm{iou}},\lambda_{\mathrm{L1}}\in\mathbb{R}$ are hyperparameters and $\mathcal{L}_{\mathrm{iou}}(\cdot)$ is the generalized IoU [38]:

$$\mathcal{L}_{\mathrm{iou}}(b_{\sigma(i)},\hat{b}_i)=1-\left(\frac{|b_{\sigma(i)}\cap\hat{b}_i|}{|b_{\sigma(i)}\cup\hat{b}_i|}-\frac{|B(b_{\sigma(i)},\hat{b}_i)\setminus b_{\sigma(i)}\cup\hat{b}_i|}{|B(b_{\sigma(i)},\hat{b}_i)|}\right). \tag{10}$$

Deformable DETR

将原来的**全局注意力**计算改为**参考点(p)周围的局部(Δp_{mqk})注意力计算**，并采用多尺度特征 。

以qkv角度来说：query不是和全局每个位置的key都计算注意力权重，而是**对于每个query，仅在全局位置中采样部分位置的key，并且value也是基于这些位置进行采样插值得到的**，最后将这个**局部&稀疏**的注意力权重施加在对应的value上。

公式差异

$$\mathrm{MultiHeadAttn}(\boldsymbol{z}_q,\boldsymbol{x})=\sum_{m=1}^{M}\boldsymbol{W}_m\big[\sum_{k\in\Omega_k}A_{mqk}\cdot\boldsymbol{W}_m'\boldsymbol{x}_k\big],$$

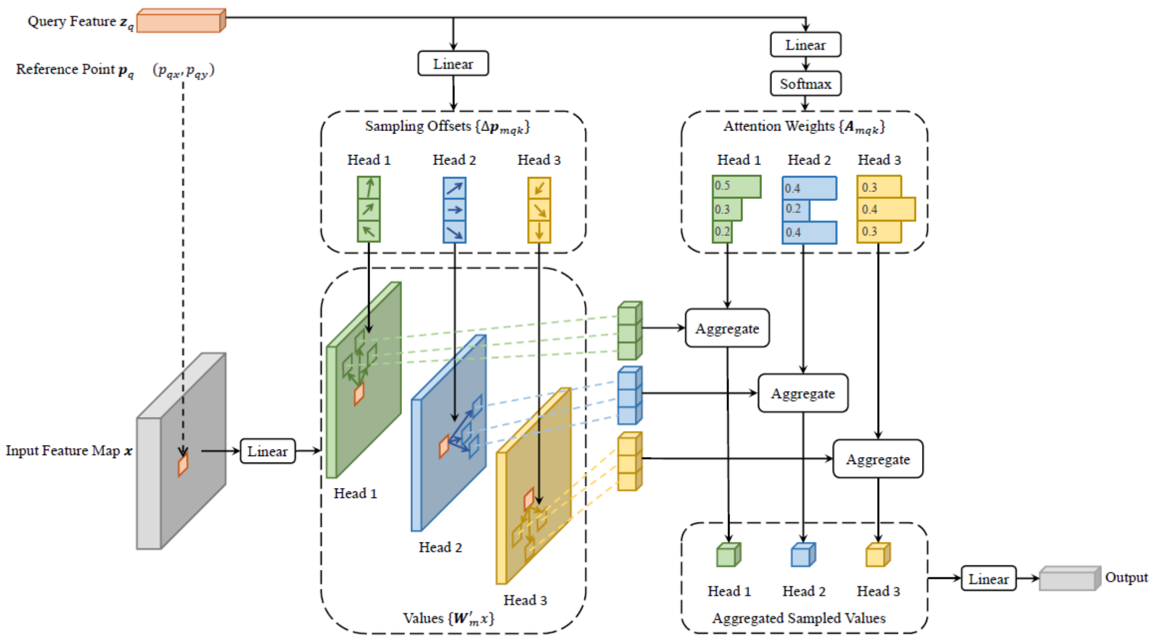
$$\text{DeformAttn}(\boldsymbol{z}_q, \boldsymbol{p}_q, \boldsymbol{x}) = \sum_{m=1}^{M} \boldsymbol{W}_m \Big[\sum_{k=1}^{K} A_{mqk} \cdot \boldsymbol{W}'_m \boldsymbol{x}(\boldsymbol{p}_q + \Delta \boldsymbol{p}_{mqk}) \Big],$$

注：$x(p_q + \Delta p_{mqk})$ 代表基于采样点位置插值出来的value

> **实现方式**

将参考点映射到不同尺度特征图，同时给予scale-level embedding

在多个level上采样，每个level采k个，共采样$l \times k$



> Encoder

Transformer encoder input: feature map + position embedding + scale-level embedding

1. 参考点(p_q)：每个特征图中点的坐标经过归一化 or **多尺度特征点的归一化坐标**，由objdect query直接通过linear预测输出
2. 偏移量(Δp_{mqk})：query经过linear输出,初始的采样点位置相当于会分布在参考点3x3、5x5、7x7、9x9方形邻域
3. A_{mqk} ：不经过QK相乘，直接linear输出
4. Query(z_p)：object query(特征图点) + query embedding(position embedding + scale-level embedding)
5. Key or Value：$l \times k$ 个采样点+位置编码
6. Attention全是MS-Self-Attention（MS代表多尺寸），

> Decoder

将参考点坐标映射（re-scales）到各尺度特征层

Self-attention （学习各个object目标之间的关系）

1. *query or key*: object query+query embedding
2. *value*: object query（注意不需要位置嵌入哦）

Cross-attention （object queries从encoder输出的feature map中提取特征（key，value））

只将decoder cross-attention模块替换为多尺度可变形注意力模块，而decoder self-attention模块仍沿用Transformer本身的。

1. *query*: object query+query embedding

2. *key:* encoder输出的多尺度特征图像素点（输入query的feature直接回归LK个偏差，LK个贡献图）
3. *value*: 由Encoder编码的feature经过线性变换得到（在query基础上根据偏差聚合特征，作为value）

   每一个value都通过多尺度deformable attention module汇聚了特征，设计检测头预测的是以key为中心的bbox偏置

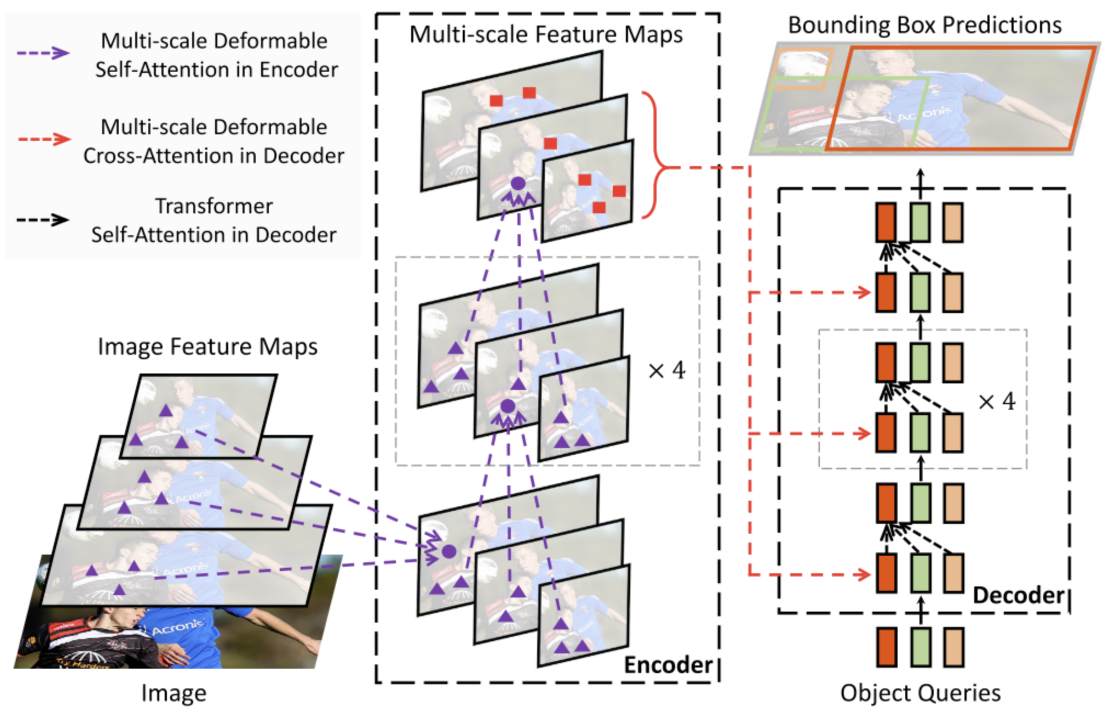
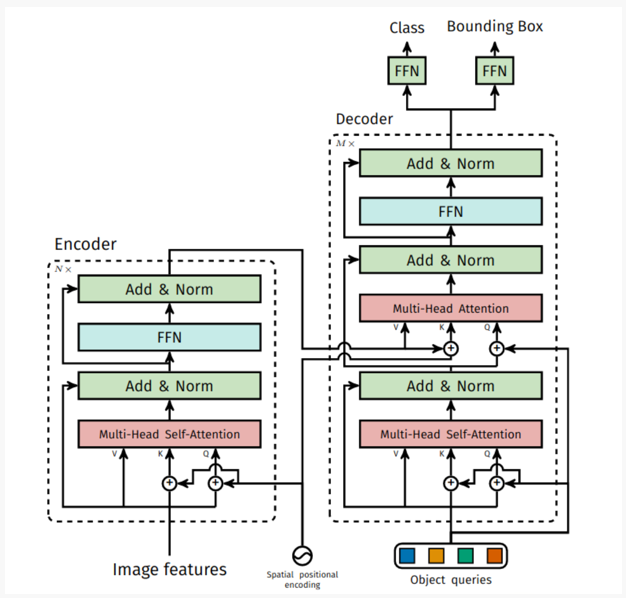
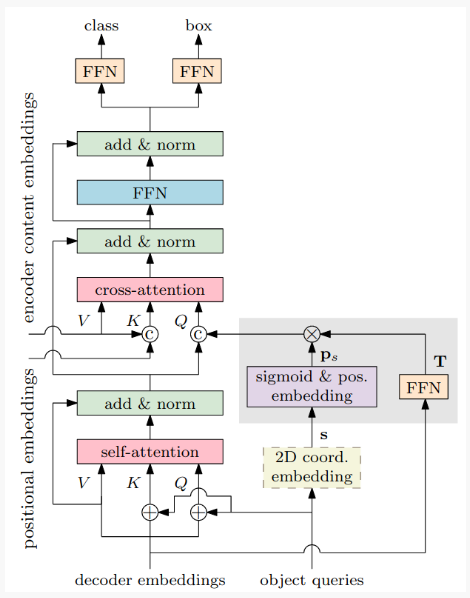

Multi-scale Deformable Self-Attention in Encoder
Multi-scale Deformable Cross-Attention in Decoder
Transformer Self-Attention in Decoder
Multi-scale Feature Maps
Bounding Box Predictions
Image Feature Maps
× 4
× 4
Image
Encoder
Decoder
Object Queries

Figure 1: Illustration of the proposed Deformable DETR object detector.

Reference:

1. code and note
2. paper notes

Conditional DETR

 Class Bounding Box FFN FFN Decoder M × Add & Norm FFN Add & Norm Multi-Head Attention Add & Norm Multi-Head Self-Attention Encoder N × Add & Norm FFN Add & Norm Multi-Head Self-Attention V K Q Image features Spatial positional encoding Object queries	 class box FFN FFN add & norm FFN add & norm cross-attention V K Q encoder content embeddings positional embeddings $\mathbf{p}_s$ sigmoid & pos. embedding T FFN s 2D coord. embedding add & norm self-attention V K Q decoder embeddings object queries
DETR	Conditional DETR

主要区别在于cross-attention

1. 将`content query` $\oplus$ `spatial query`变为concat
2. 将`content`(关注目标图像内容)和`spatial`(关注目标空间位置)解耦
3. 从decoder embedding中提取与reference point的偏移量 T，类似于anchor框与中心点偏移，具体来说通过$FFN(decoder\ embedding)$来实现；
4. 再将从object query中预测得出的s（2维坐标点类似与anchor中心点）映射到与saptial positional encoding相同的正弦位置编码空间中得 p_s，将 p_s 和 T 相乘得到spatial query。

Reference：

1. paper note with author

Efficient DETR

在 backbone 出来的 dense feature map (`shape: [256, h, w]`)上预测 top-k score 的 proposals, 用这些top-k score(`shape: [2, h, w]`)的idx来对索引 proposal 的 2-d 或者 4-d 坐标用于初始化 reference point. 用同样的idx索引选择 top-k 的 feature (`shape: [256, h, w]`)来初始化 object query.

Reference points are 2-d tensors that represent predictions of box centers and belong to the location information of an object container.

不同Reference points初始化方式，对于1-decoder最终效果影响差异巨大，但对cascade-decoder影响不大，所以采用1-decoder和特殊的初始化方式来减小和cascade-decoder之间的gap。

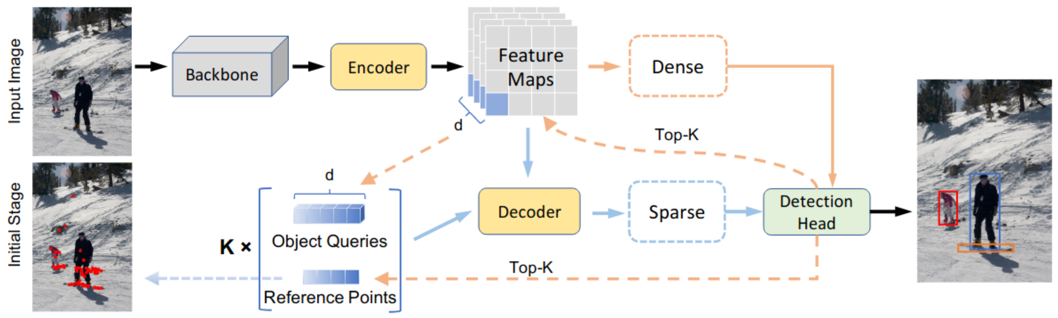


Figure 5. Architecture of Efficient DETR. Top-K denotes the topk selection method. For example, the top scored indices of anchors from the dense part are used to select object queries from encoder, and select reference points from region proposals.

Anchor DETR

1. 改变位置编码，对于同一区域多个目标，同一个点采用多个模式(N_p)编码，采用sin编码，$Q_f^i \in R^{N_A \times 1 \times C} \rightarrow Q_f^i \in R^{N_A \times N_q \times C}$

$$Q_p = g(Pos_q), K_p = g(Pos_k)$$

$$Q_p = \text{Embedding}(N_q, C).$$

$$Q_f^i = \text{Embedding}(N_p, C)$$

$$Q = Q_f^{init} + Q_p.$$

2. Row-Column Decoupled Attention；将2D注意力解耦为x和y两个1D注意力，减少内存消耗(即用一维的gobal pooling消除纵向或者横向的维度)

$$
\begin{aligned}
A_x &= \text{softmax}(\frac{Q_x K_x^T}{\sqrt{d_k}}), A_x \in \mathbb{R}^{N_q \times W}, \\
Z &= \text{weighted\_sumW}(A_x, V), Z \in \mathbb{R}^{N_q \times H \times C}, \\
A_y &= \text{softmax}(\frac{Q_y K_y^T}{\sqrt{d_k}}), A_y \in \mathbb{R}^{N_q \times H}, \\
Out &= \text{weighted\_sumH}(A_y, Z), Out \in \mathbb{R}^{N_q \times C},
\end{aligned}
$$

where

$$
\begin{aligned}
Q_x &= Q_f + Q_{p,x}, \quad Q_y = Q_f + Q_{p,y}, \\
Q_{p,x} &= g_{1D}(Pos_{q,x}), \quad Q_{p,y} = g_{1D}(Pos_{q,y}), \\
K_x &= K_{f,x} + K_{p,x}, \quad K_y = K_{f,y} + K_{p,y}, \\
K_{p,x} &= g_{1D}(Pos_{k,x}), \quad K_{p,y} = g_{1D}(Pos_{k,y}), \\
V &= V_f, \quad V \in \mathbb{R}^{H \times W \times C}.
\end{aligned}
$$

Dynamic DETR

encoder：用中间层的deformable 偏移作用到所有level feature map上，用SE layer通道注意力加权一波，relu再cat

ROI：ROI window内局部信息，缺少全局特征

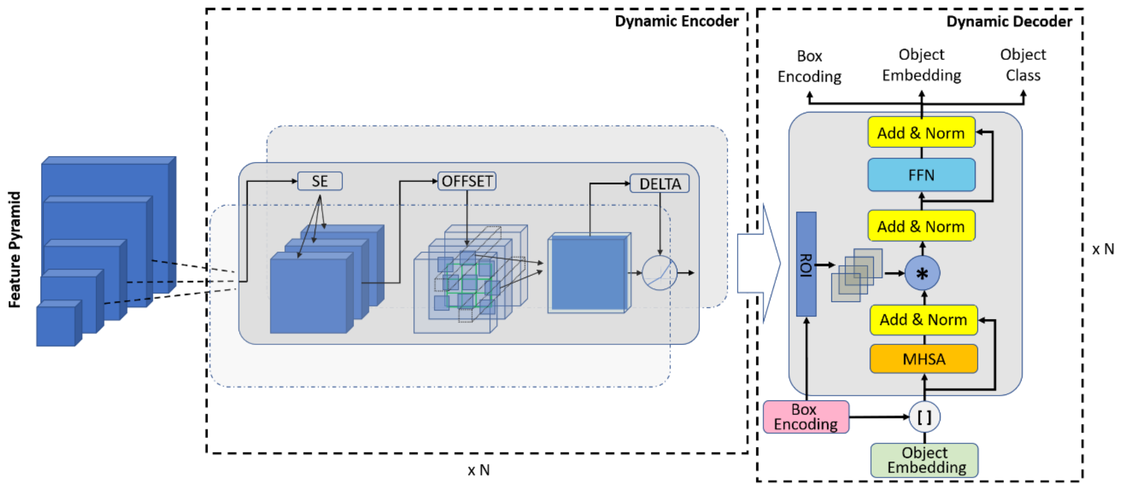


Figure 2. The architecture overview of the proposed approach. Our Dynamic DETR coherently combines a dynamic convolution-based encoder and a dynamic Transformer-based decoder.

Sparse R-CNN

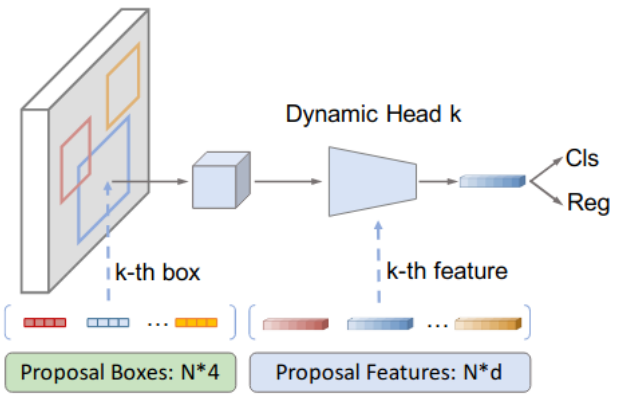


Figure 3 – An overview of Sparse R-CNN pipeline. The input includes an image, a set of proposal boxes and proposal features, where the latter two are learnable parameters. The backbone extracts feature map, each proposal box and proposal feature are fed into its exclusive dynamic head to generate object feature, and finally outputs classification and location.

1. N个anchor类似于anchor DETR中多模式
2. 动态refine box，head本质作attention

```
def dynamic_instance_interaction(pro_feats, roi_feats):
    # pro_feats: (N, C)
    # roi_feats: (N, S*S, C)

    # parameters of two 1x1 convs: (N, 2 * C * C/4)
    dynamic_params = linear1(pro_features)
    # parameters of first conv: (N, C, C/4)
    param1 = dynamic_params[:, :C*C/4].view(N, C, C/4)
    # parameters of second conv: (N, C/4, C)
    param2 = dynamic_params[:, C*C/4:].view(N, C/4, C)

    # instance interaction for roi_features: (N, S*S, C)
    roi_feats = relu(norm(bmm(roi_feats, param1)))
    roi_feats = relu(norm(bmm(roi_feats, param2)))

    # roi_feats are flattened: (N, S*S*C)
    roi_feats = roi_feats.flatten(1)
    # obj_feats: (N, C)
    obj_feats = linear2(roi_feats)
    return obj_feats
```

Figure 4 – Pseudo-code of dynamic instance interaction, the k-th proposal feature generates dynamic parameters for the corresponding k-th RoI. `bmm`: batch matrix multiplication; `linear`: linear projection.