# Detection Transformers

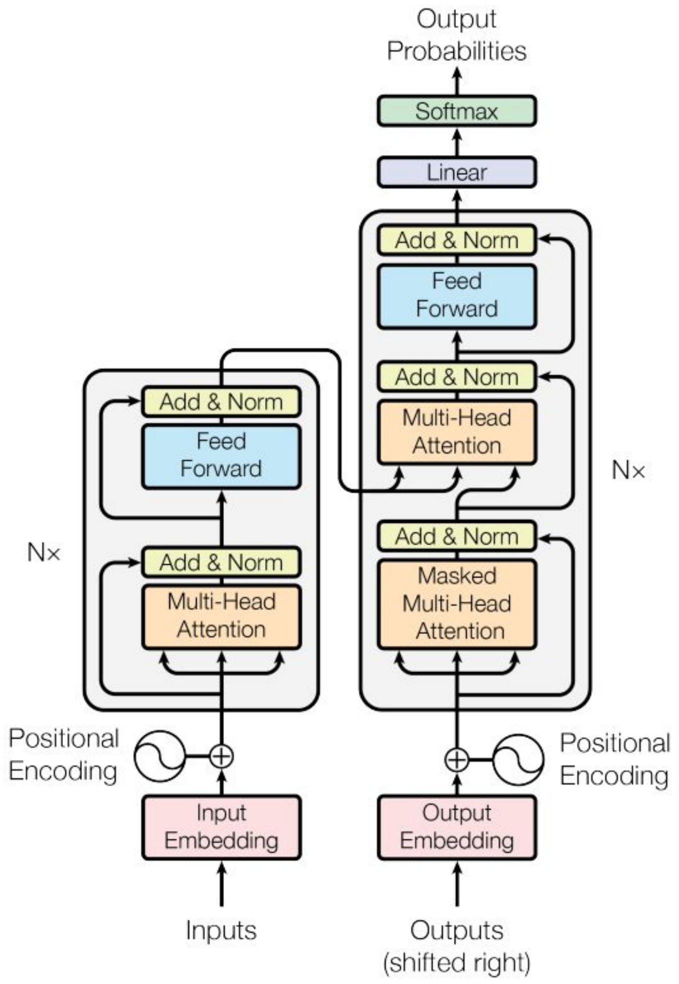
## 1. Transformer

Ashish Vaswani and Noam Shazeer and Niki Parmar and Jakob Uszkoreit and Llion Jones and Aidan N. Gomez and Lukasz Kaiser and Illia Polosukhin. Attention is all you need. [paper]

# Architecture -

- Input :
  [seq_len]
- Input Embedding :
  [seq_len, hidden_dim]
- Positional Encoding :
  Added to the input embeddings so that they also include (relative) positional information. More on this later.

* hidden_dim and d_model are used interchangeably.



## Architecture -

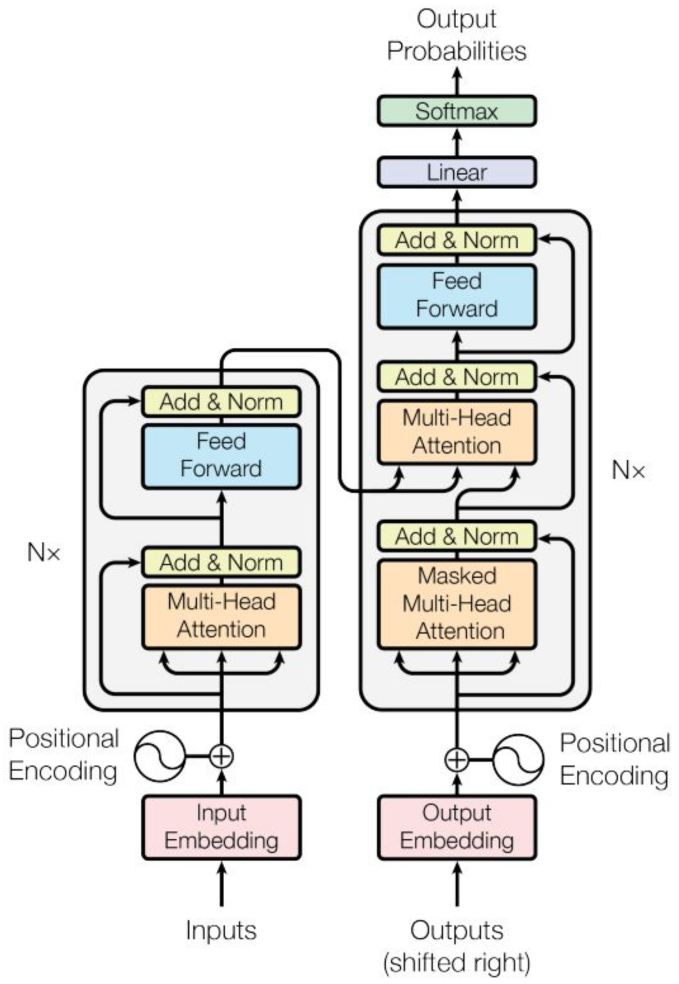
- Input to Decoder :

  [1 + seq_len]

  The extra token comes because we append <SOS> symbol at the start of the ground truth sequence.
- Ground truth for Decoder :

  [seq_len + 1]

  The extra token comes because we append <EOS> symbol at the end of the ground truth sequence.



# Self-Attention -

Self-Attention allows the model to relate words to each other.

In this simple case we consider the sequence length **seq** = 6 and $\mathbf{d_{model}} = \mathbf{d_k} = 512$.

The matrices **Q**, **K** and **V** are just the input sentence.

$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\text{softmax}\left(\frac{\underset{(6,\ 512)}{\mathbf{Q}} \times \underset{(512,\ 6)}{\mathbf{K^T}}}{\sqrt{512}}\right) =$$

| | YOUR | CAT | IS | A | LOVELY | CAT | Σ |
|---|---|---|---|---|---|---|---|
| YOUR | 0.268 | 0.119 | 0.134 | 0.148 | 0.179 | 0.152 | 1 |
| CAT | 0.124 | 0.278 | 0.201 | 0.128 | 0.154 | 0.115 | 1 |
| IS | 0.147 | 0.132 | 0.262 | 0.097 | 0.218 | 0.145 | 1 |
| A | 0.210 | 0.128 | 0.206 | 0.212 | 0.119 | 0.125 | 1 |
| LOVELY | 0.146 | 0.158 | 0.152 | 0.143 | 0.227 | 0.174 | 1 |
| CAT | 0.195 | 0.114 | 0.203 | 0.103 | 0.157 | 0.229 | 1 |

(6, 6)

* all values are random.

* for simplicity I considered only one head, which makes $d_{model} = d_k$.

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

| | YOUR | CAT | IS | A | LOVELY | CAT |
|---|---|---|---|---|---|---|
| YOUR | 0.268 | 0.119 | 0.134 | 0.148 | 0.179 | 0.152 |
| CAT | 0.124 | 0.278 | 0.201 | 0.128 | 0.154 | 0.115 |
| IS | 0.147 | 0.132 | 0.262 | 0.097 | 0.218 | 0.145 |
| A | 0.210 | 0.128 | 0.206 | 0.212 | 0.119 | 0.125 |
| LOVELY | 0.146 | 0.158 | 0.152 | 0.143 | 0.227 | 0.174 |
| CAT | 0.195 | 0.114 | 0.203 | 0.103 | 0.157 | 0.229 |

(6, 6)

X

**V**

(6, 512)

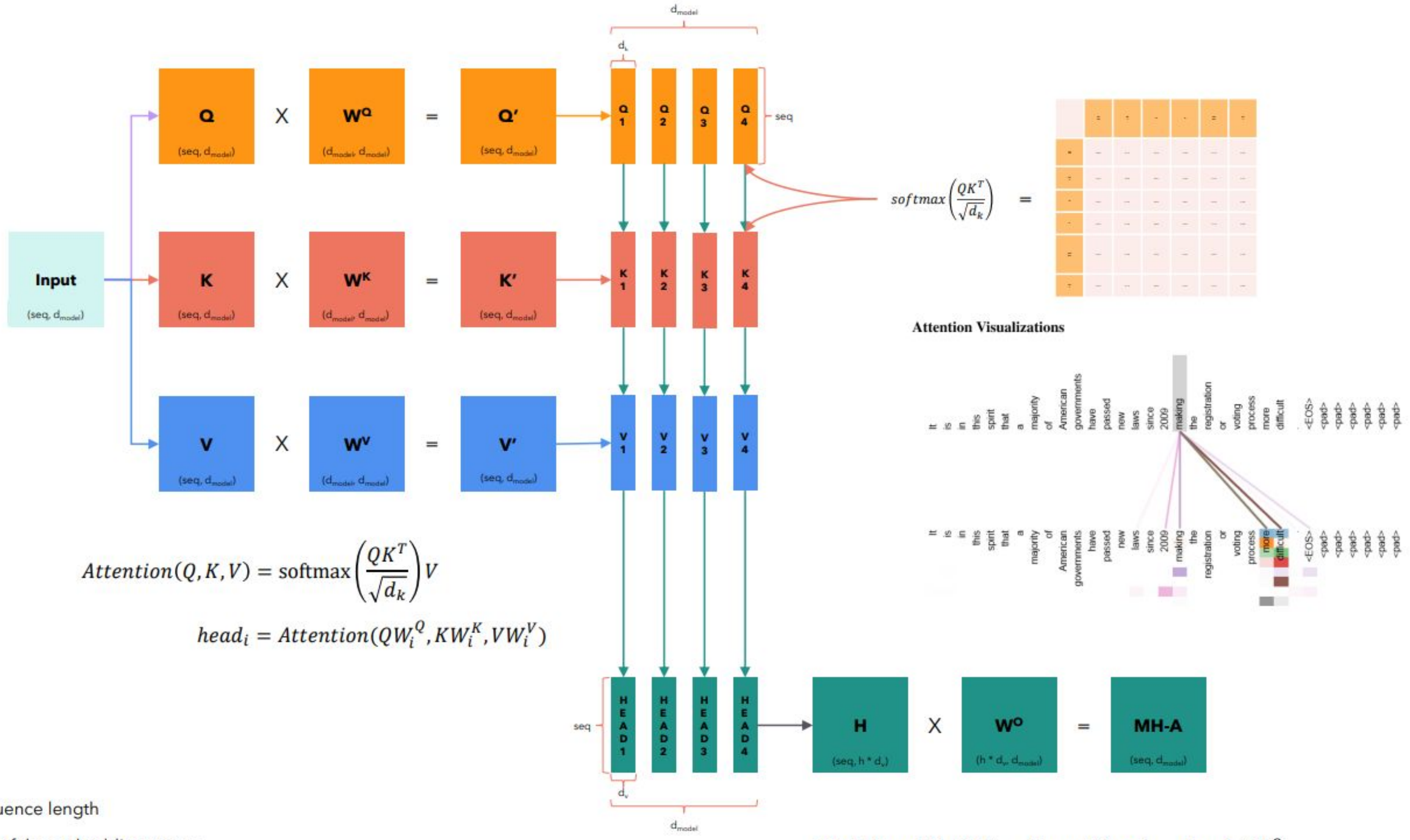
=

**Attention**

(6, 512)

Each row in this matrix captures not only the meaning (given by the embedding) or the position in the sentence (represented by the positional encodings) but also each word's interaction with other words.

# Multi-Head Attention -



$$Attention(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

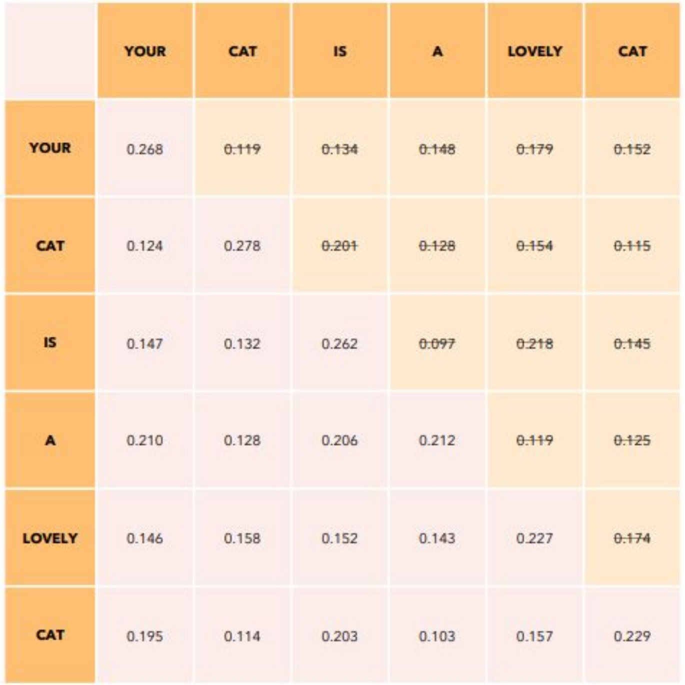
$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

$$MultiHead(Q,K,V) = Concat(head_1 \ldots head_h)W^O$$

| | |
|---|---|
| *seq* | = sequence length |
| $d_{model}$ | = size of the embedding vector |
| h | = number of heads |

## Masked Multi-Head Attention -

Our goal is to make the model causal: it means the output at a certain position can only depend on the words on the previous positions. The model **must not** be able to see future words.



| | YOUR | CAT | IS | A | LOVELY | CAT |
|---|---|---|---|---|---|---|
| YOUR | 0.268 | ~~0.119~~ | ~~0.134~~ | ~~0.148~~ | ~~0.179~~ | ~~0.152~~ |
| CAT | 0.124 | 0.278 | ~~0.201~~ | ~~0.128~~ | ~~0.154~~ | ~~0.115~~ |
| IS | 0.147 | 0.132 | 0.262 | ~~0.097~~ | ~~0.218~~ | ~~0.145~~ |
| A | 0.210 | 0.128 | 0.206 | 0.212 | ~~0.119~~ | ~~0.125~~ |
| LOVELY | 0.146 | 0.158 | 0.152 | 0.143 | 0.227 | ~~0.174~~ |
| CAT | 0.195 | 0.114 | 0.203 | 0.103 | 0.157 | 0.229 |

All the values above the diagonal are replace with **-∞** *before* applying the softmax, which will replace them with zero.

## Feed Forward Network -

- Self-Attention is Linear operation.
- Output from MHA is passed through a two layer FFN with ReLU activation in between to introduce non-linearity, thus making the model more expressive and powerful.

## Prediction Head -

- Output from Decoder layer is passed through Linear Layer which predicts the logits over output vocab.

## Some Comments -

- Self-Attention is permutation invariant. Thus, we add positional encodings to the input.
- If we don’t want some positions to interact, we can always set the values to –∞ before applying the softmax in the attention score matrix and the model will not learn those interactions.
- The output shapes are exactly same as that of input for encoder (decoder) layer.

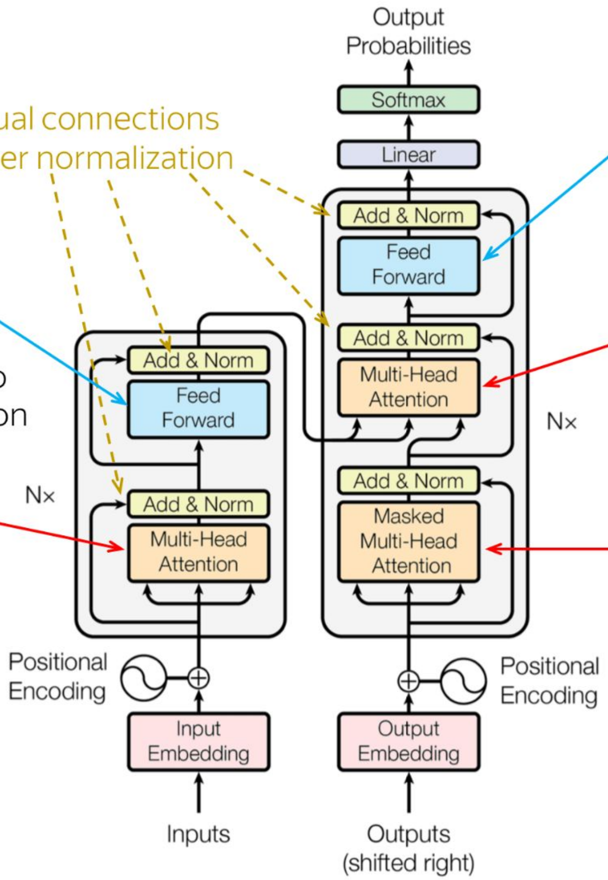
# Architecture -

Residual connections and layer normalization

Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Encoder self-attention: tokens look at each other

queries, keys, values are computed from encoder states



Feed-forward network: after taking information from other tokens, take a moment to think and process this information

Decoder-encoder attention: target token looks at the source

queries – from decoder states; keys and values from encoder states

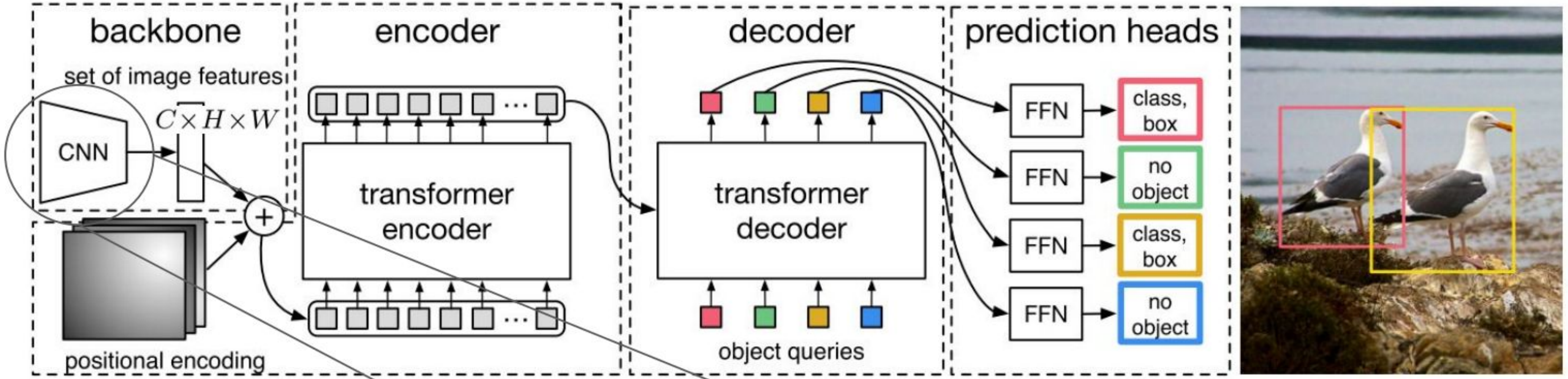
Decoder self-attention (masked): tokens look at the previous tokens

queries, keys, values are computed from decoder states

## 2. Detection Transformer (DETR)

Nicolas Carion and Francisco Massa and Gabriel Synnaeve and Nicolas Usunier and Alexander Kirillov and Sergey Zagoruyko. End-to-End Object Detection with Transformers. [paper]

## Architecture -



CNN Backbone:

$$f \in \mathbb{R}^{C \times H \times W}$$

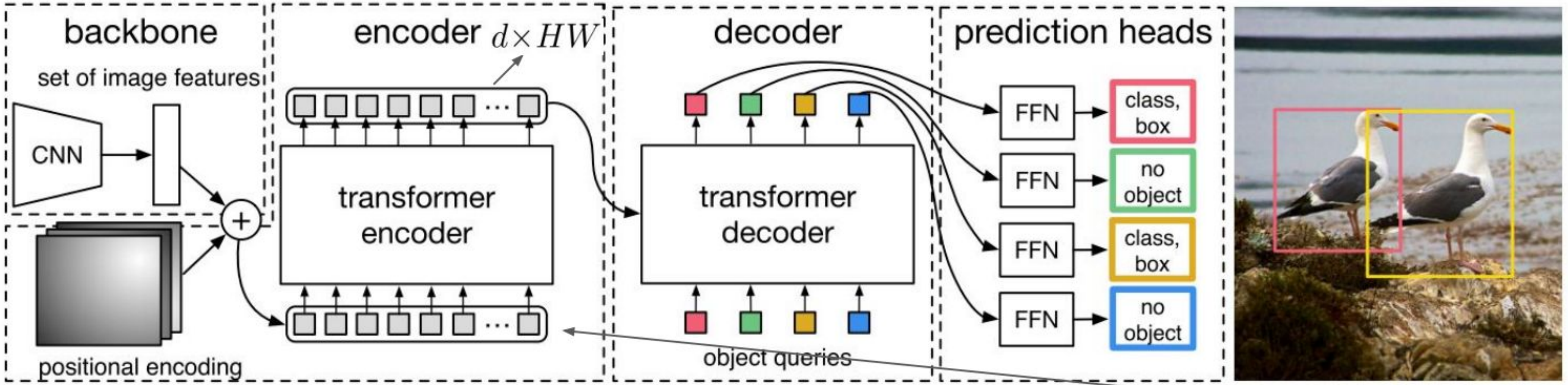
Activation Map $C \times H \times W$

$$x_{\text{img}} \in \mathbb{R}^{3 \times H_0 \times W_0}$$

Typically:

$$C = 2048 \; H, W = \frac{H_0}{32}, \frac{W_0}{32}$$

# Architecture -



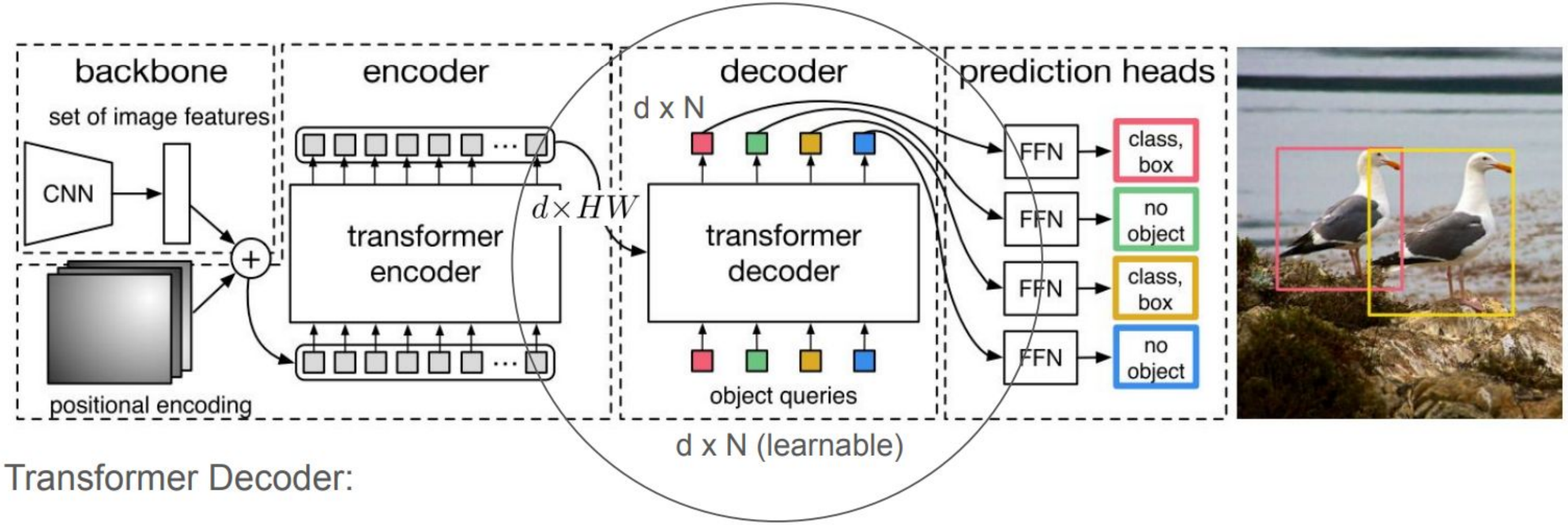
Transformer Encoder:

- Reduce channel dimension (1x1 Convolution)
- Flatten features into a sequential feature map
- Add positional encodings to input of each attention layer
- A multi-head self-attention module and a feed forward network

$$C\times H\times W \longrightarrow z_0 \in \mathbb{R}^{d\times H\times W}$$

$$\left.\begin{array}{c} d\times HW \\ + \\ \text{Positional Encoding} \end{array}\right\}$$

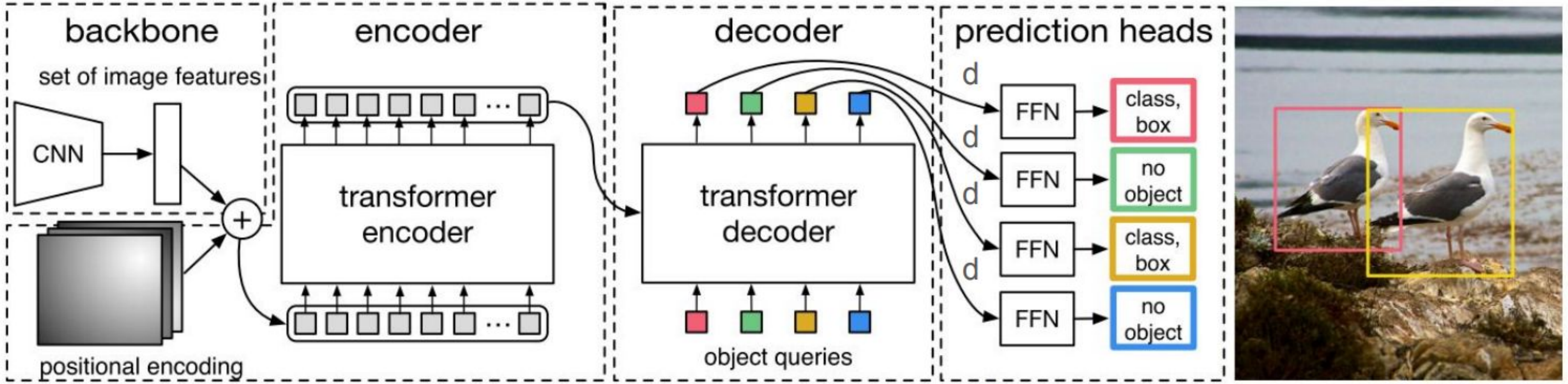
## Architecture -



Transformer Decoder:

- Use encoded representations, d x HW embeddings, from encoder as key and value
- Add N learnt positional encodings (object queries) to input of each attention layer
- Transforms N object queries into N output embeddings in parallel (non-autoregressive)
- Trained with auxiliary decoding loss to improve training

## Architecture -



d x N

Prediction Feed-forward networks (FFNs):

- For normalized center coordinates: 3-layer perceptron with ReLU activation, hidden dim d
- For class prediction: a linear projection layer with softmax
- Class prediction also predicts “no object”

## Hungarian Matching and Set Loss -

- Let us denote by $y$ the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ the set of $N$ predictions.
- We consider $y$ also as a set of size $N$ padded with $\varnothing$ (no object).
- To find a bipartite matching between these two sets we search for a permutation of $N$ elements $\sigma \in \mathfrak{S}_N$ with the lowest cost:

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

  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

## Hungarian Matching and Set Loss -

- Each element $i$ of the ground truth set can be seen as a $y_i = (c_i, b_i)$ where $c_i$ is the target class label (which may be $\varnothing$) and $b_i \in [0,1]^4$ is a vector that defines ground truth box center coordinates and its height and width relative to the image size.
- For the prediction with index $\sigma(i)$ we define the probability of class $c_i$ as $\hat{p}_{\sigma(i)}(c_i)$ and the predicted box as $\hat{b}_{\sigma(i)}$.
- With these notations we define:

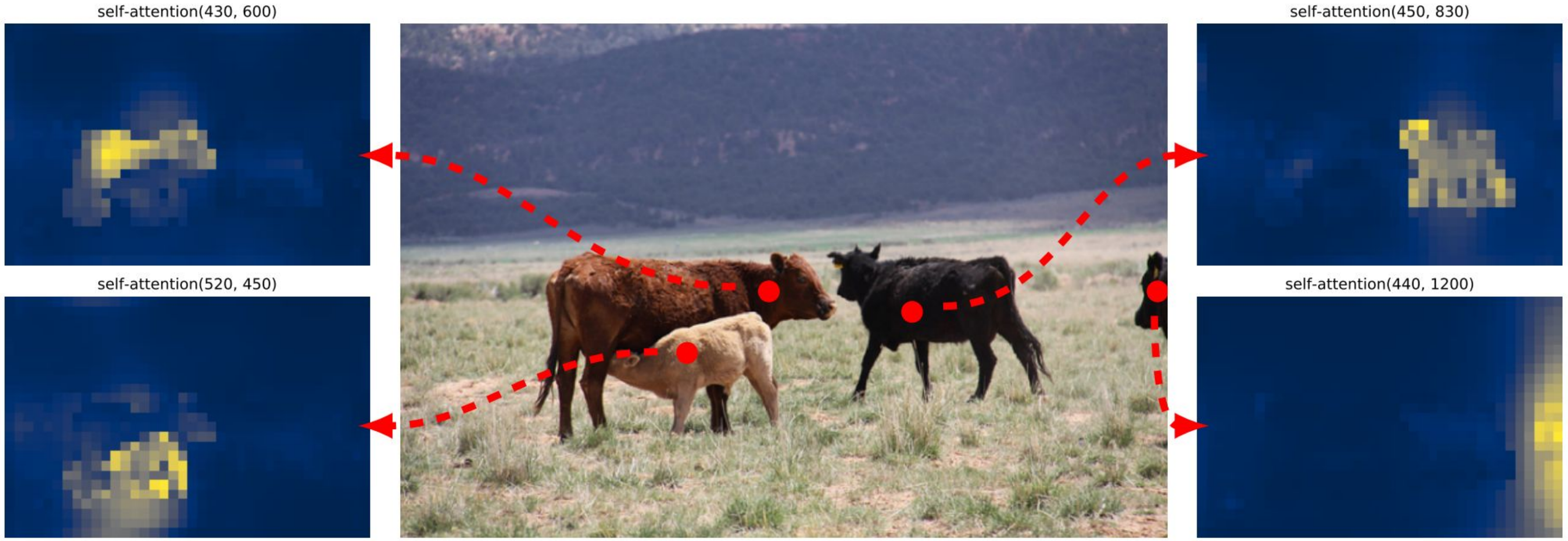
$$\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -\mathbf{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbf{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)}).$$

- We define the loss similarly to the losses of common object detectors:

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

where $\hat{\sigma}$ is the optimal assignment computed by the Hungarian algorithm.

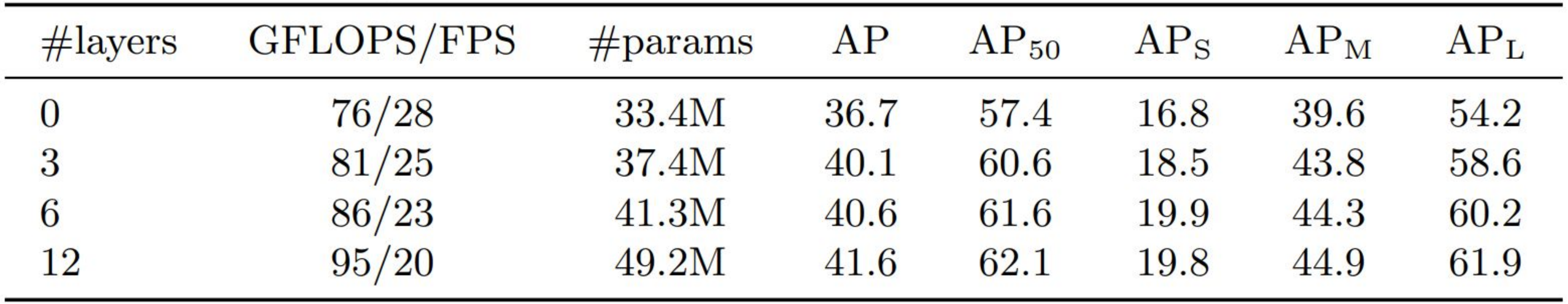
## Encoder Self Attention -



Visualization of attention scores from last encoder layer for a set of reference points.

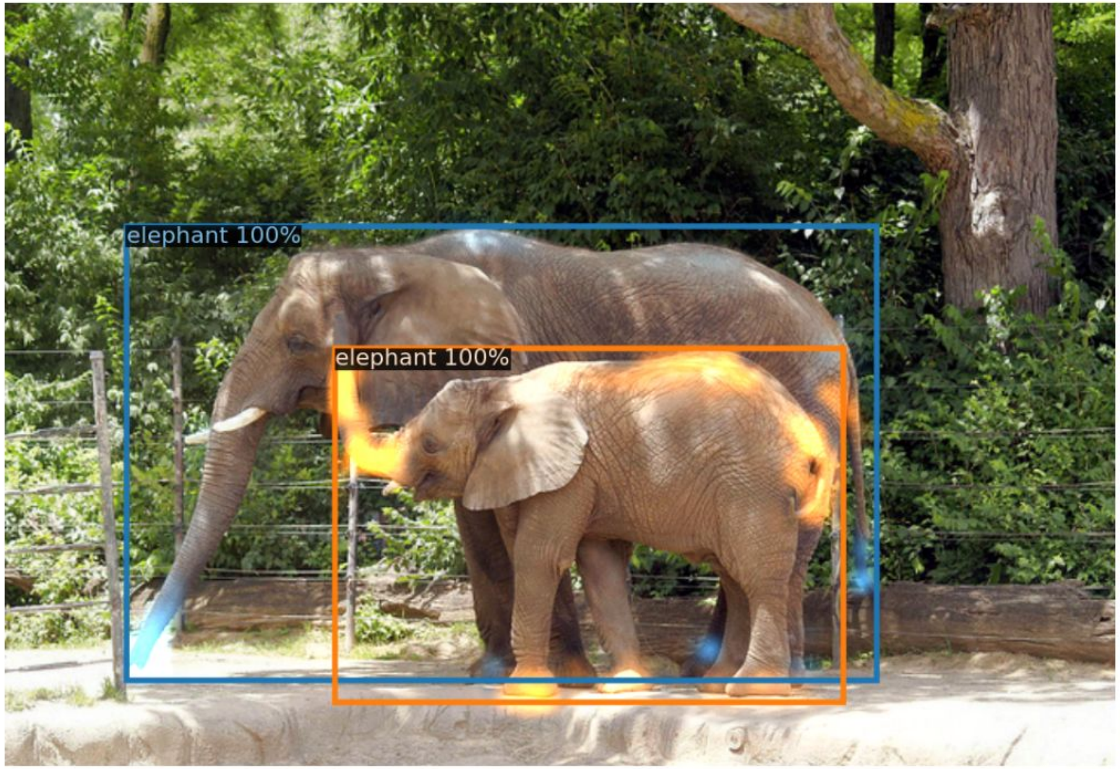
## Encoder Self Attention -

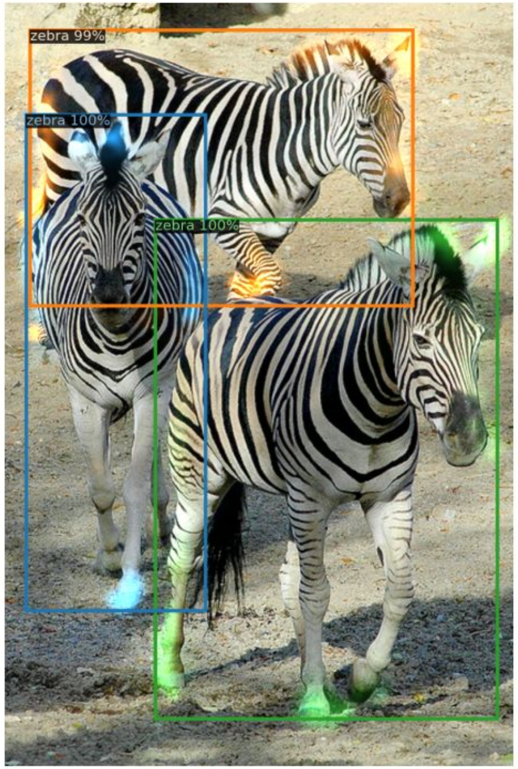
- Encoder is capable of separating object instances (from previous slide), which helps in object extraction and localization for the decoder.
- Following table shows the effect of varying the number of encoder layers.



| #layers | GFLOPS/FPS | #params | AP | $AP_{50}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| 0 | 76/28 | 33.4M | 36.7 | 57.4 | 16.8 | 39.6 | 54.2 |
| 3 | 81/25 | 37.4M | 40.1 | 60.6 | 18.5 | 43.8 | 58.6 |
| 6 | 86/23 | 41.3M | 40.6 | 61.6 | 19.9 | 44.3 | 60.2 |
| 12 | 95/20 | 49.2M | 41.6 | 62.1 | 19.8 | 44.9 | 61.9 |

# Decoder Cross Attention -

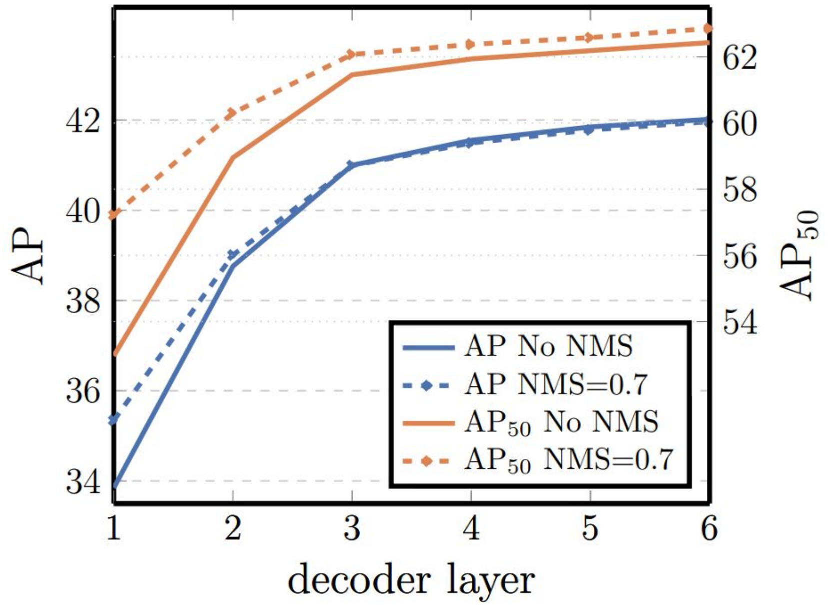




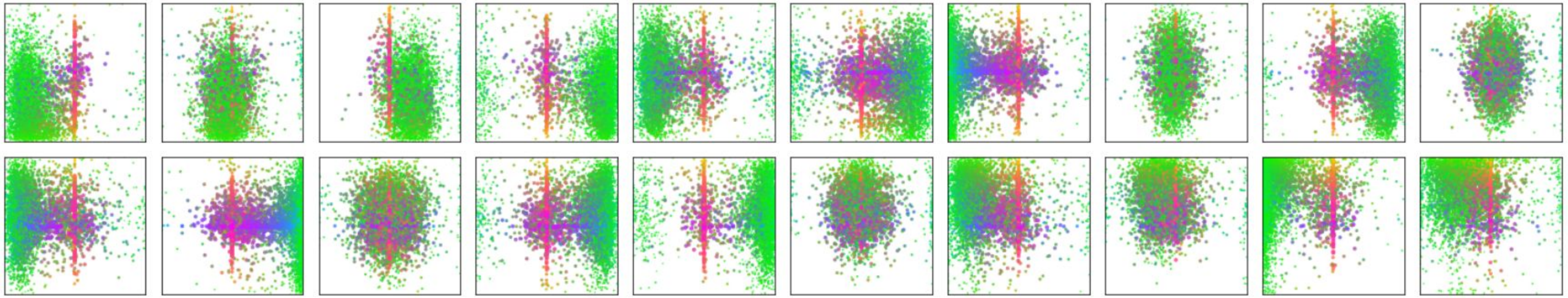
Visualization of attention scores from last decoder layer for every predicted object.

## Decoder Cross Attention -

- Decoder's cross attention is fairly local, attending to the object extremities.
- Once the encoder separates the object instances, the decoder only needs to attend to the extremities to extract the class and object boundaries.
- DETR by design does not need NMS, as the self attention in decoder inhibits the model to predict duplicate predictions.

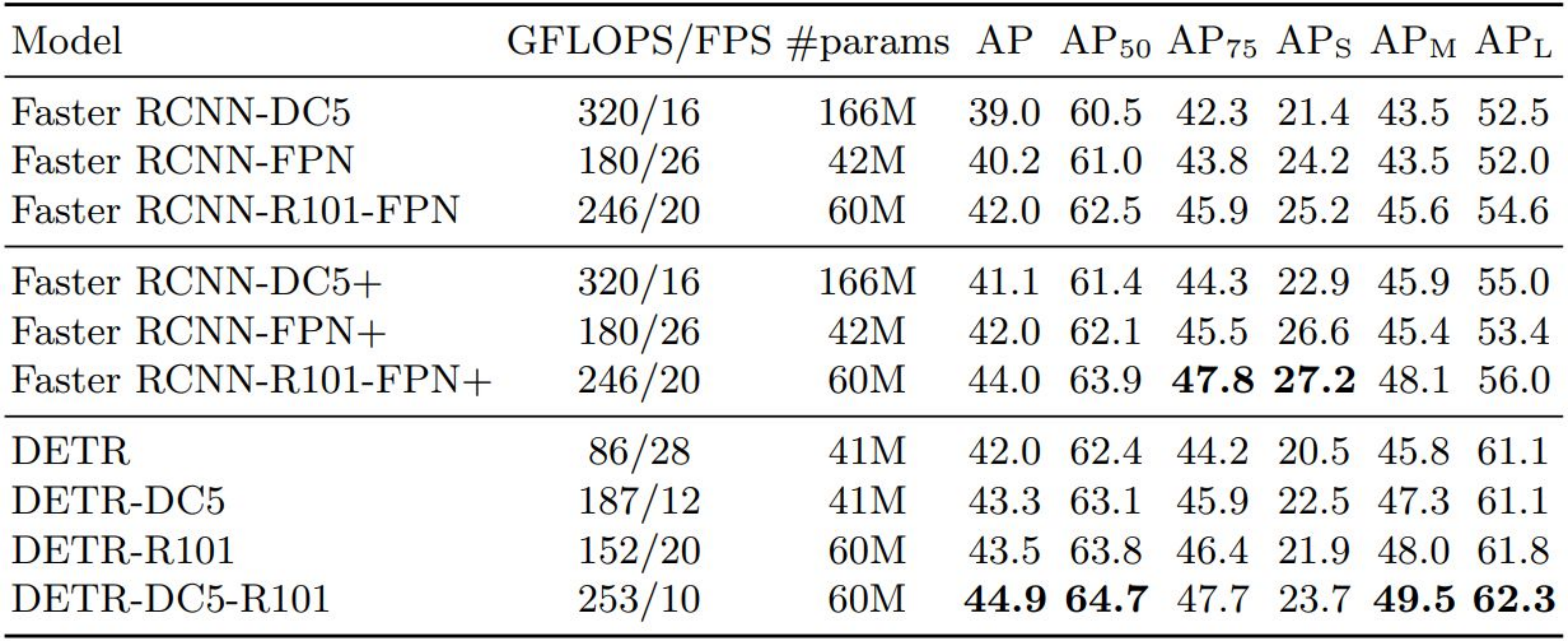


## Object Queries -



- Visualization of all box predictions on all images from COCO 2017 val set for first 20 (out of 100) object queries.
- Each point represents the box center, green for small objects, red for large horizontal objects and blue for large vertical objects.
- Each object query learns to specialize on certain areas and box sizes.

## Results -



| Model | GFLOPS/FPS | #params | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| Faster RCNN-DC5 | 320/16 | 166M | 39.0 | 60.5 | 42.3 | 21.4 | 43.5 | 52.5 |
| Faster RCNN-FPN | 180/26 | 42M | 40.2 | 61.0 | 43.8 | 24.2 | 43.5 | 52.0 |
| Faster RCNN-R101-FPN | 246/20 | 60M | 42.0 | 62.5 | 45.9 | 25.2 | 45.6 | 54.6 |
| Faster RCNN-DC5+ | 320/16 | 166M | 41.1 | 61.4 | 44.3 | 22.9 | 45.9 | 55.0 |
| Faster RCNN-FPN+ | 180/26 | 42M | 42.0 | 62.1 | 45.5 | 26.6 | 45.4 | 53.4 |
| Faster RCNN-R101-FPN+ | 246/20 | 60M | 44.0 | 63.9 | **47.8** | **27.2** | 48.1 | 56.0 |
| DETR | 86/28 | 41M | 42.0 | 62.4 | 44.2 | 20.5 | 45.8 | 61.1 |
| DETR-DC5 | 187/12 | 41M | 43.3 | 63.1 | 45.9 | 22.5 | 47.3 | 61.1 |
| DETR-R101 | 152/20 | 60M | 43.5 | 63.8 | 46.4 | 21.9 | 48.0 | 61.8 |
| DETR-DC5-R101 | 253/10 | 60M | **44.9** | **64.7** | 47.7 | 23.7 | **49.5** | **62.3** |

## Issues with DETR -

- Needs upwards of 300 epochs to train on COCO dataset.
- Low accuracy on small objects.
- Complexity is O(n^4) in image size.
- Number of possible object prediction is fixed before training.

## 3. Deformable DETR

Xizhou Zhu and Weijie Su and Lewei Lu and Bin Li and Xiaogang Wang and Jifeng Dai. Deformable DETR: Deformable Transformers for End-to-End Object Detection. [paper]

## Multi-Head Attention Recap -

$$\mathrm{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = [\mathrm{head}_1,\ldots,\mathrm{head}_h]\mathbf{W}_0$$

$$\text{where } \mathrm{head}_i = \mathrm{Attention}\left(\mathbf{Q}\mathbf{W}_i^Q,\mathbf{K}\mathbf{W}_i^K,\mathbf{V}\mathbf{W}_i^V\right)$$

Above $\mathbf{W}$ are all learnable parameter matrices.

$$\mathrm{MultiHeadAttn}(\overset{\text{query element}}{\boldsymbol{z}_q}, \overset{\text{key elements}}{\boldsymbol{x}}) = \underset{\text{Sum over attention heads}}{\sum_{m=1}^{M}} \overbrace{\boldsymbol{W}_m \Big[ \sum_{k\in\Omega_k} \underset{\text{Attention weights}}{A_{mqk}} \cdot \boldsymbol{W}'_m}^{\text{Learnable weights}} \underset{\text{Key feature}}{\boldsymbol{x}_k} \Big]$$

subjective to $\sum_{k\in\Omega_k} A_{mqk} = 1$

## Deformable Attention -

- Formulation

query element

key elements

$K$ is the total sampled key number ($K \ll HW$)

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

reference point

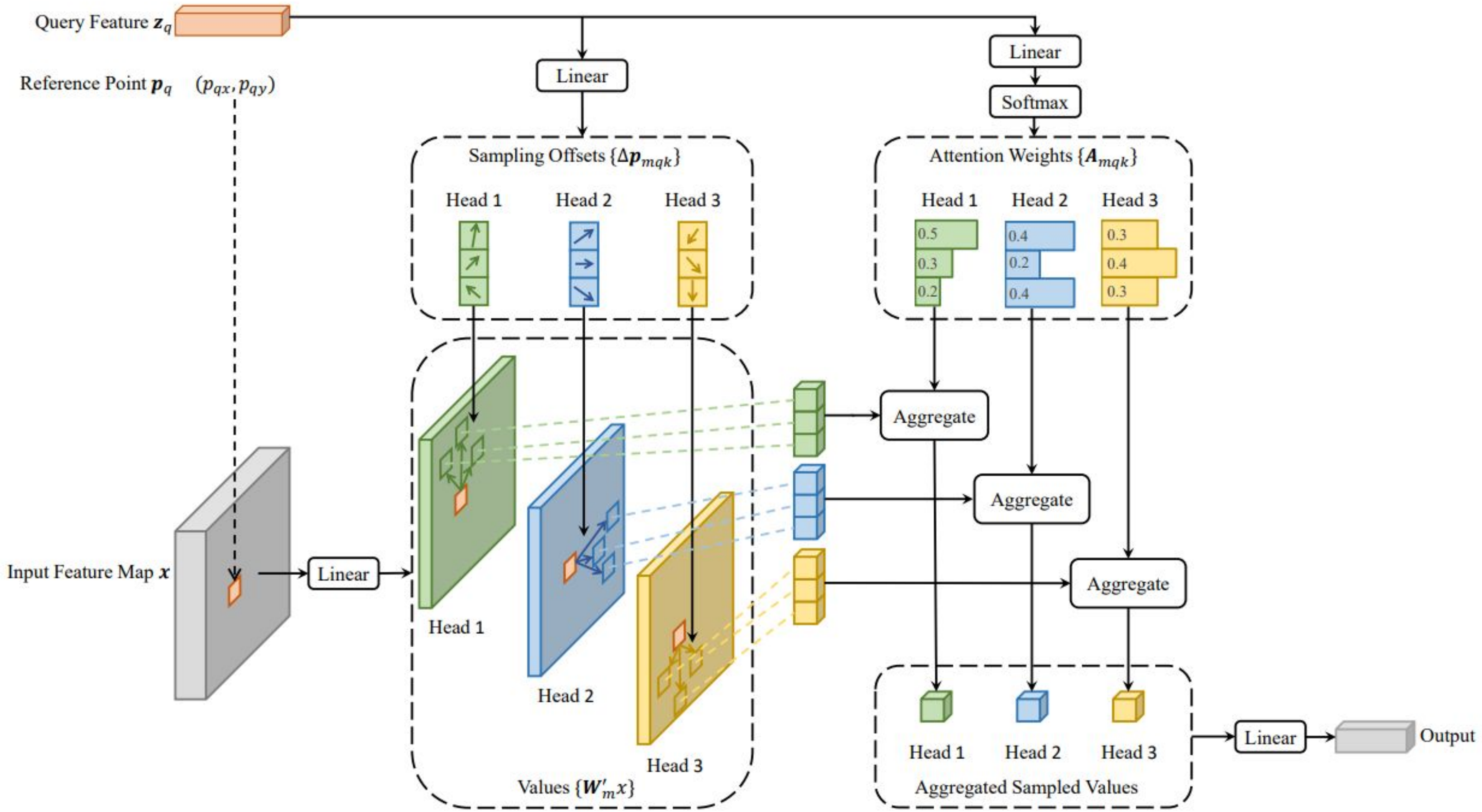
Sum over attention heads

Attention weights

sparsely sampled key feature

- Reference point
  - In encoder: the query point itself
  - In decoder: predicted from object query embedding

# Deformable Attention -



## Multi-Scale Deformable Attention -

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

$$\Downarrow$$

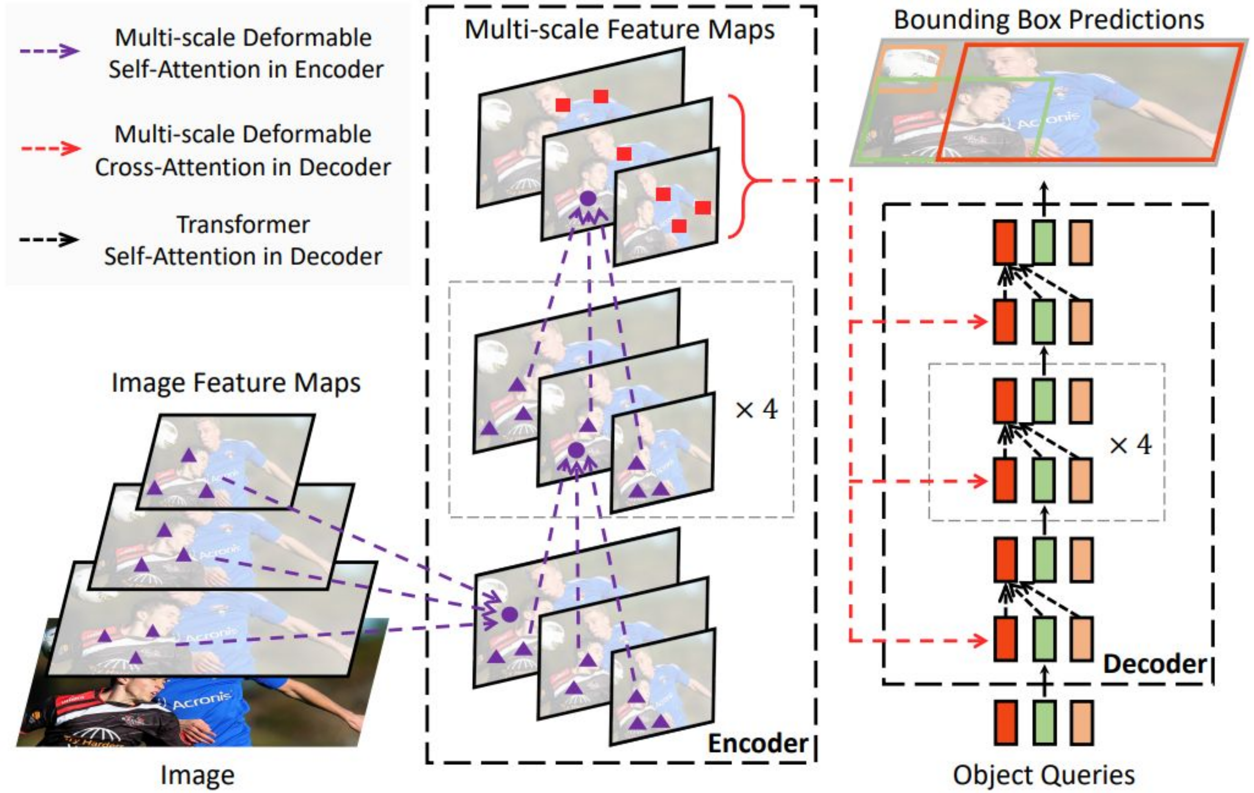
$$\text{MSDeformAttn}(\boldsymbol{z}_q, \hat{\boldsymbol{p}}_q, \boxed{\{\boldsymbol{x}^l\}_{l=1}^{L}}) = \sum_{m=1}^{M} \boldsymbol{W}_m \Big[ \boxed{\sum_{l=1}^{L} \sum_{k=1}^{K}} A_{mlqk} \cdot \boldsymbol{W}'_m \boldsymbol{x}^l(\phi_l(\hat{\boldsymbol{p}}_q) + \Delta \boldsymbol{p}_{mlqk}) \Big],$$

$\phi_l(\hat{\boldsymbol{p}}_q)$ normalized coordinate (ranged in [0, 1]) of $\hat{\boldsymbol{p}}_q$

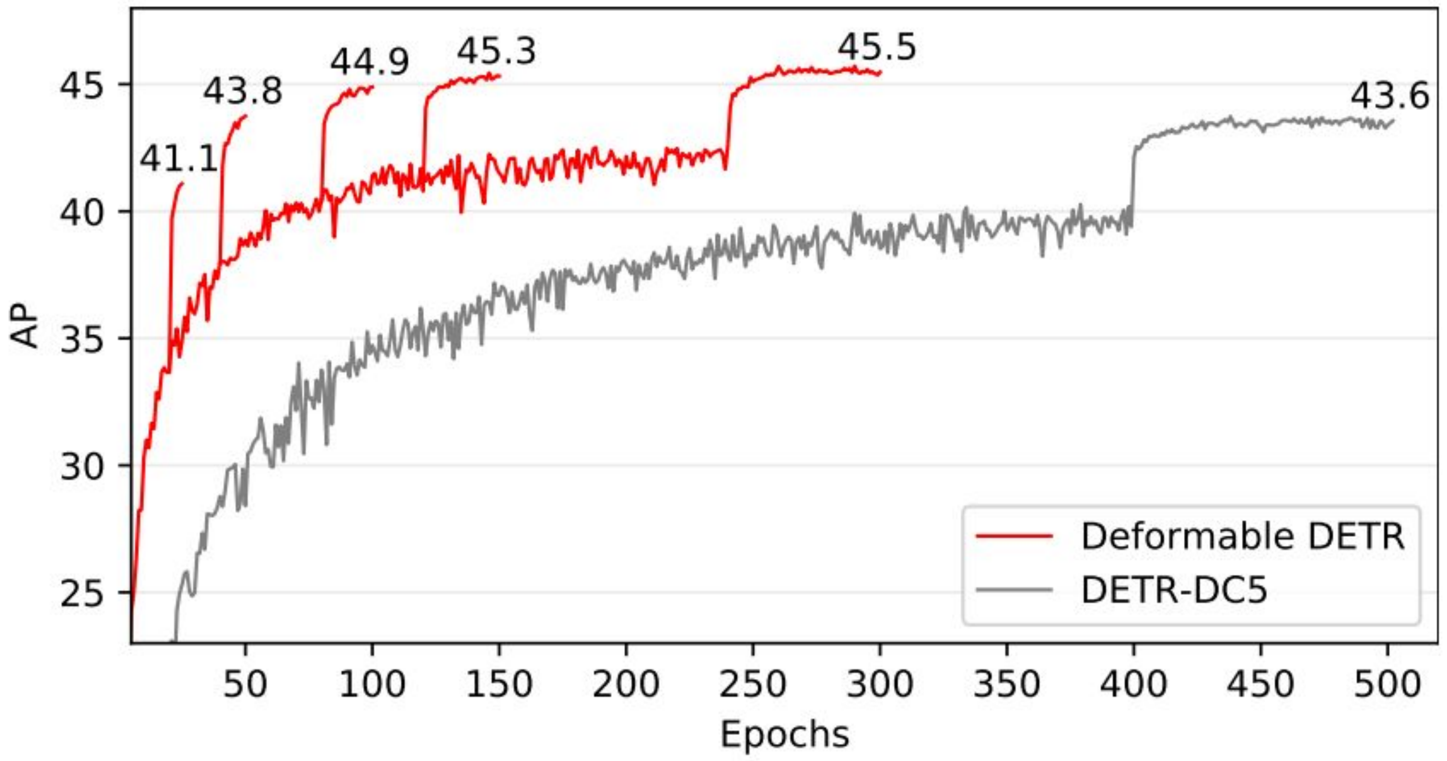
single-scale feature map → multi-scale feature maps

$K$ sampling points → $L \times K$ sampling points (i.e., $K$ sampling points per feature level)

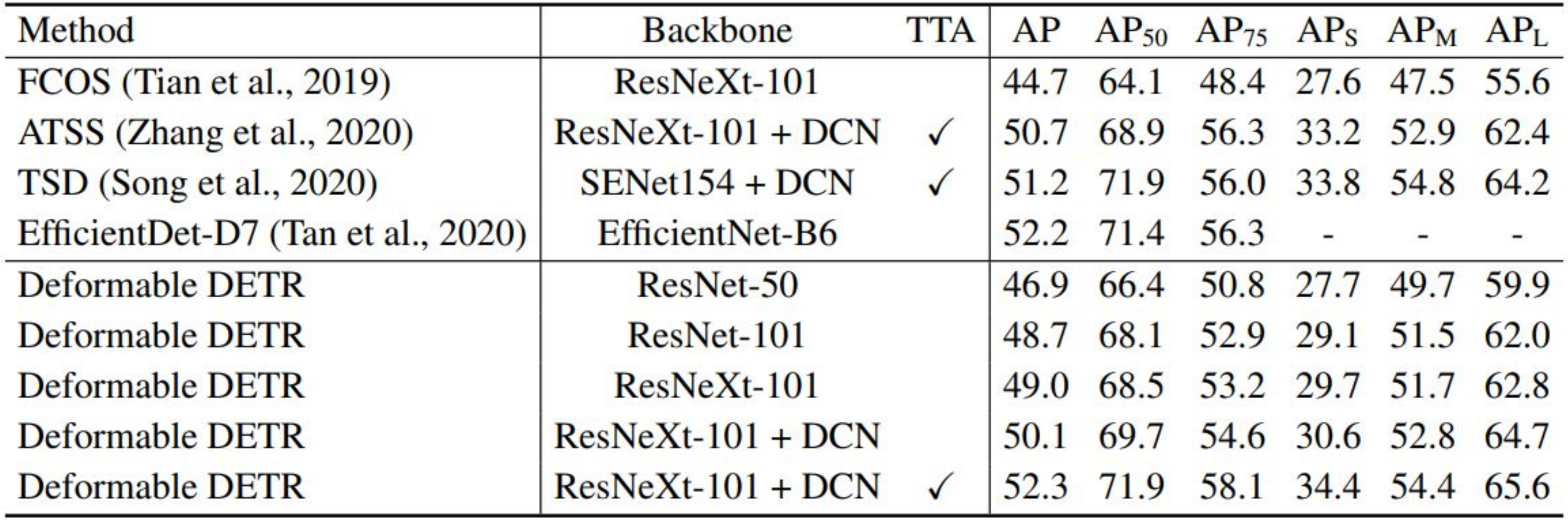
# Architecture -


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
Encoder
Decoder
Image
Object Queries

## Convergence Curve on COCO val -



## Results -

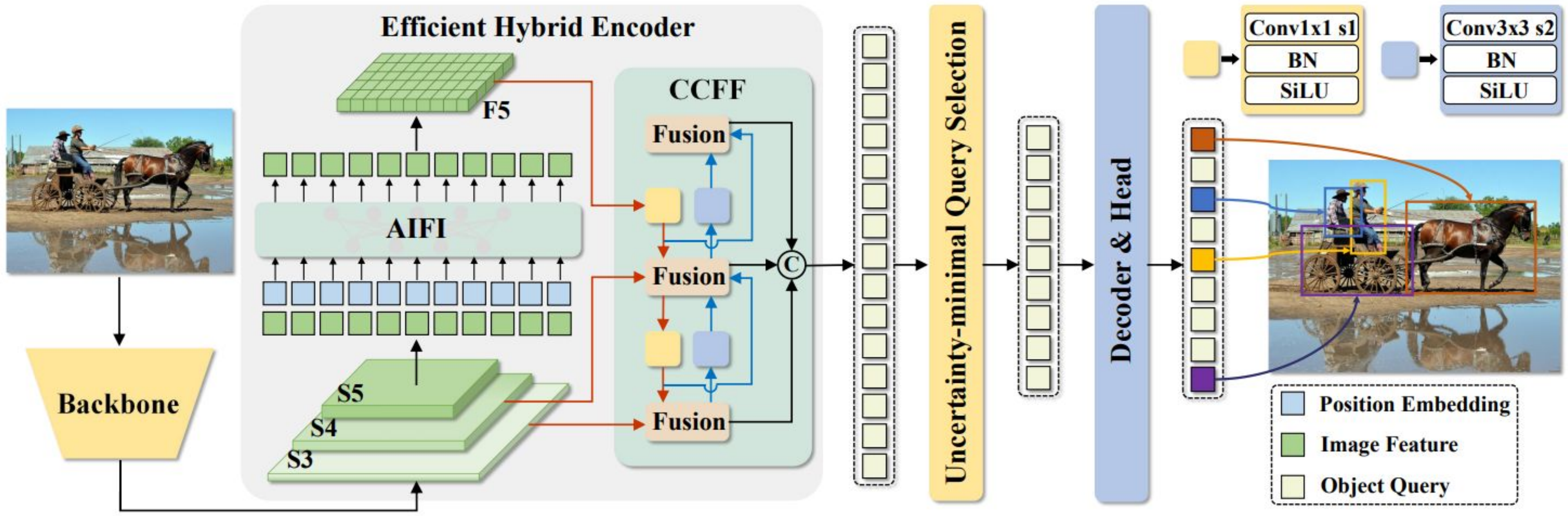


| Method | Backbone | TTA | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|
| FCOS (Tian et al., 2019) | ResNeXt-101 | | 44.7 | 64.1 | 48.4 | 27.6 | 47.5 | 55.6 |
| ATSS (Zhang et al., 2020) | ResNeXt-101 + DCN | ✓ | 50.7 | 68.9 | 56.3 | 33.2 | 52.9 | 62.4 |
| TSD (Song et al., 2020) | SENet154 + DCN | ✓ | 51.2 | 71.9 | 56.0 | 33.8 | 54.8 | 64.2 |
| EfficientDet-D7 (Tan et al., 2020) | EfficientNet-B6 | | 52.2 | 71.4 | 56.3 | - | - | - |
| Deformable DETR | ResNet-50 | | 46.9 | 66.4 | 50.8 | 27.7 | 49.7 | 59.9 |
| Deformable DETR | ResNet-101 | | 48.7 | 68.1 | 52.9 | 29.1 | 51.5 | 62.0 |
| Deformable DETR | ResNeXt-101 | | 49.0 | 68.5 | 53.2 | 29.7 | 51.7 | 62.8 |
| Deformable DETR | ResNeXt-101 + DCN | | 50.1 | 69.7 | 54.6 | 30.6 | 52.8 | 64.7 |
| Deformable DETR | ResNeXt-101 + DCN | ✓ | 52.3 | 71.9 | 58.1 | 34.4 | 54.4 | 65.6 |

## 4. Real Time DETR

Yian Zhao and Wenyu Lv and Shangliang Xu and Jinman Wei and Guanzhong Wang and Qingqing Dang and Yi Liu and Jie Chen. DETRs Beat YOLOs on Real-time Object Detection. [paper]

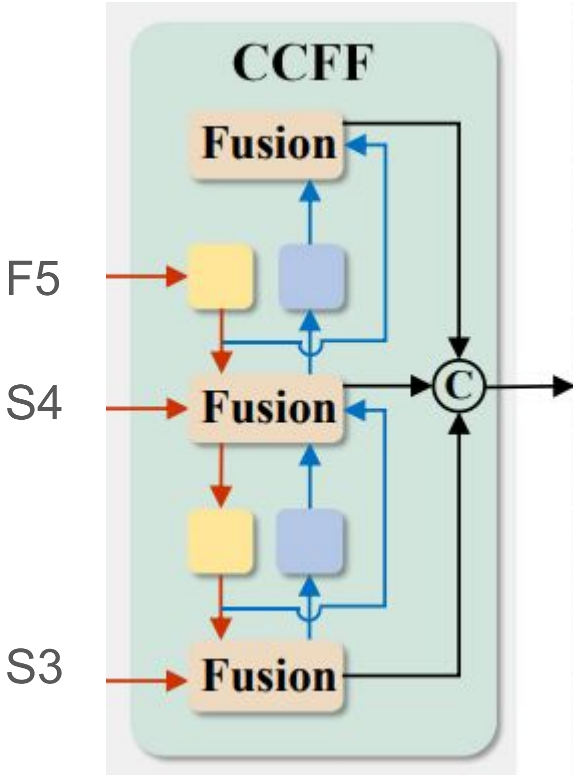
# Architecture -

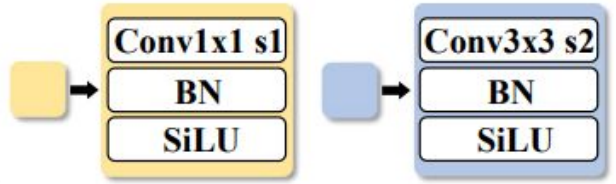


## Intra-scale Feature Interaction (AIFI) -

- AIFI Block performs intra-scale feature interaction only on S5 with transformer encoder.
- Multi-Head Self-Attention operation on high level feature map separates (relevant) objects, which helps in the localization and recognition of objects by subsequent modules.
- However, the intra-scale interactions of lower-level features are unnecessary due to the risk of duplication and confusion with high-level feature interactions.

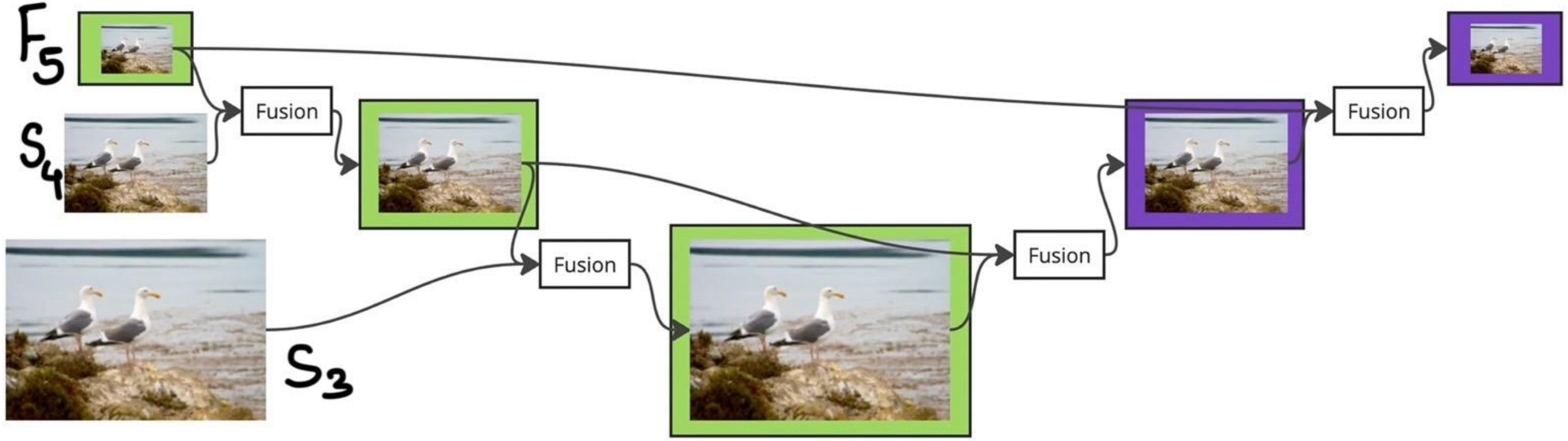
# Cross-scale Feature Fusion (CCFF) -



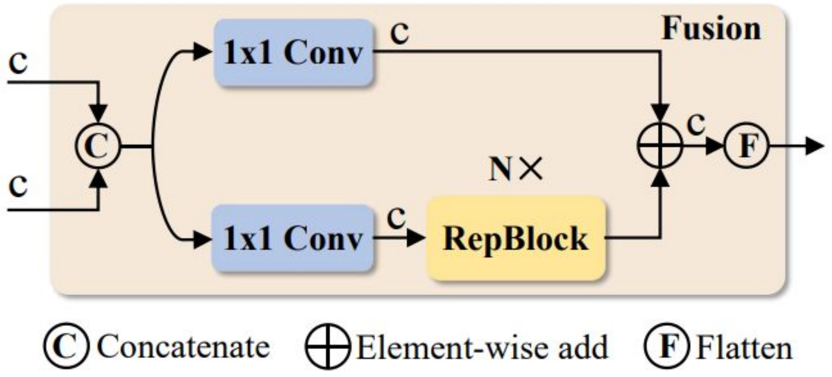


Downward Arrow = Upsample 2x
Upward Arrow = Downsample 2x

## Cross-scale Feature Fusion (CCFF) -



## Fusion Block -



## Decoder -

- Multi-Head Self-Attention for object queries.
- Multi-Scale Deformable Attention for cross attention between object queries and Hybrid encoder outputs.
- Also employs Denoising queries for efficient training. More on this later.

## Prediction Head(s) -

- Three layer MLP for bounding box predictions.
- Linear Projection for class logits predictions.
- Encoder block and every Decoder layer has its own (different) prediction heads.
- Prediction heads' outputs from intermediate decoder layers are treated as auxiliary outputs for that layer.
- Decoder layer output is the input for next decoder layer, as usual.
- Bounding box predictions gives the reference point for next decoder layer's multi-scale deformable cross attention.
- In our use case, we have 7 labels per object (license plate) and hence prediction head has 7 linear projections for label logits predictions.

## Uncertainty-minimal Query Selection -

- We select top num_queries encoder outputs for which the corresponding class predictions are most confidently some class (excluding no object class).
- The corresponding bounding box predictions gives the initial reference point for multi-scale deformable cross attention.
- In our use case, we get 7 label logits predictions so we add these individual confidence scores from each label predictions to get a consolidated score.
- We select top num_queries encoder outputs for which the corresponding consolidated score is maximum.

## Denoising Queries -

- We collect all ground truth objects in an image and add noise to their bounding boxes and labels. This collection is called a group.
- We have multiple groups (noised version of all ground truth objects) to maximize the effect of denoising learning.
- Noised bounding box generation includes scaling the width and height of the bounding box; and shifting the center of the bounding box. We make sure the shifted center is still in the original bounding box.
- Ground truth labels are flipped randomly to generate noised labels. This forces the model to learn strong box-label relationships.
- Denoising is only considered in training, during inference the denoising part is removed, leaving only the matching part.

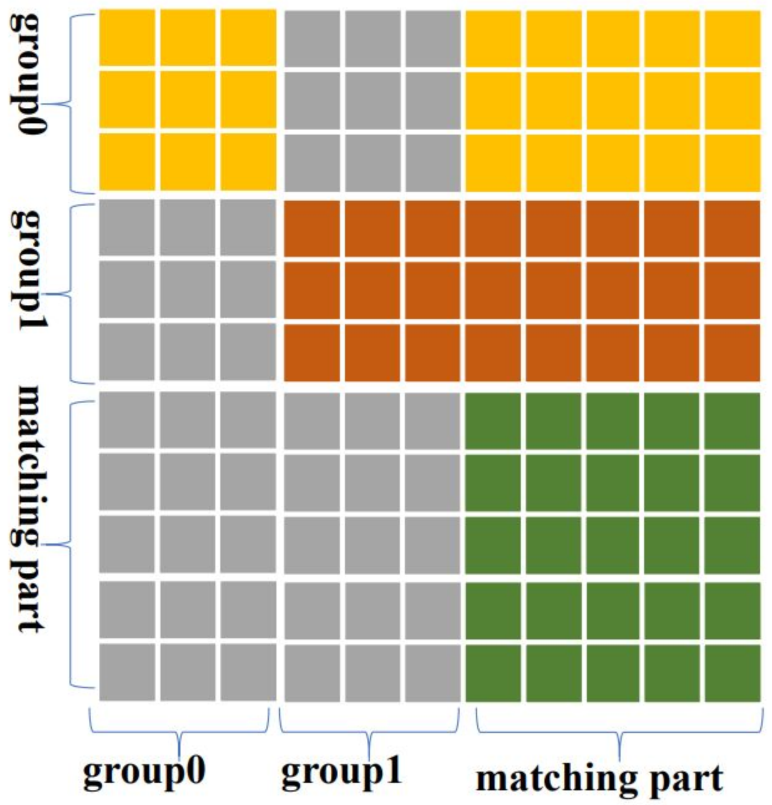
## Denoising Queries -

- The noised labels gives us the queries and the noised boxes gives us the corresponding initial reference points.
- The noised labels is an integer and the query is supposed to be a vector of dimension hidden_dim.
- Thus, we use learnable label embeddings of dimension hidden_dim. And label embeddings corresponding to the noised labels are used as queries.
- In our use case, we have 7 labels per object or 7 label embeddings per object. We add these 7 label embeddings to get queries. This is a common practice in NLP.
- Alternatively, we could have used learnable embeddings of dimension hidden_dim / 7 and concatenated the 7 embeddings to get the query.

## Denoising Queries : Attention Mask -

- We don’t want information leakage between different groups and matching part (real object queries).
- Thus, we initialize attention mask such that real object queries can only attend to each other; and not attend to any denoising queries.
- Similarly, the denoising queries from a group can only attend to other denoising queries from the same group; and not attend to any denoising queries from other groups.

## Attention Mask Example -



- The group0 (group1) queries can attend to group0 (group1) queries and matching queries.
- The matching queries can attend to matching queries only.