Deep Learning for Computer Vision

Transformers for Detection

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Recall: Transformers to Vision Transformers

- Transformers revolutionized NLP by offering more efficient and effective ways to model sequential data

Recall: Transformers to Vision Transformers

- Transformers revolutionized NLP by offering more efficient and effective ways to model sequential data
- ViT extended transformer architecture to vision tasks, replacing CNNs with self-attention mechanisms and showed promise in image classification benchmarks like ImageNet

Recall: Transformers to Vision Transformers

- Transformers revolutionized NLP by offering more efficient and effective ways to model sequential data
- ViT extended transformer architecture to vision tasks, replacing CNNs with self-attention mechanisms and showed promise in image classification benchmarks like ImageNet

Onwards to Object Detection

Object detection often involves processing large volumes of visual data and making fine-grained decisions.

Problem: Traditional object detectors rely on hand-crafted components like anchor boxes and non-maximum suppression, adding complexity and inefficiency

Recall: Transformers to Vision Transformers

- Transformers revolutionized NLP by offering more efficient and effective ways to model sequential data
- ViT extended transformer architecture to vision tasks, replacing CNNs with self-attention mechanisms and showed promise in image classification benchmarks like ImageNet

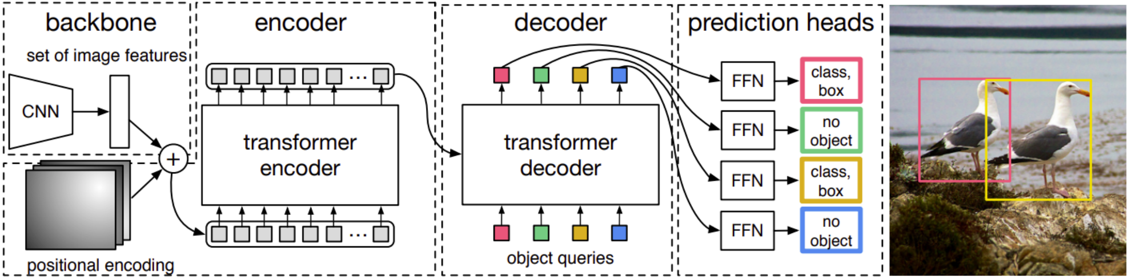
Onwards to Object Detection

Object detection often involves processing large volumes of visual data and making fine-grained decisions.

Problem: Traditional object detectors rely on hand-crafted components like anchor boxes and non-maximum suppression, adding complexity and inefficiency

Solution: DETR [3] was introduced as an end-to-end object detection framework using the transformer architecture, eliminating the need for hand-crafted components

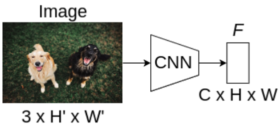
DEtection TRansformer (DETR)[1]



- DETR views object detection as a direct set prediction problem
- The main components of DETR are:
  - A set-based global loss that forces unique predictions via bipartite matching
  - A transformer encoder-decoder architecture

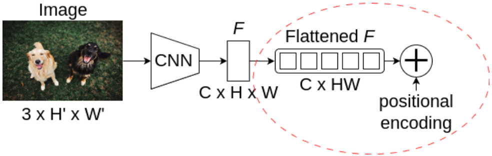
[1] Carion et al, End-to-End Object Detection with Transformers, ECCV 2020

DEtection TRansformer (DETR)



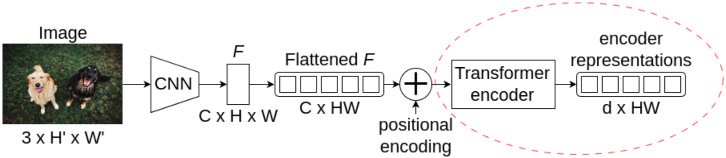
- The input image is passed through a Convolutional Neural Network (CNN) to extract a set of feature maps
- These feature maps capture the visual information of the image at different spatial scales

DEtection TRansformer (DETR)



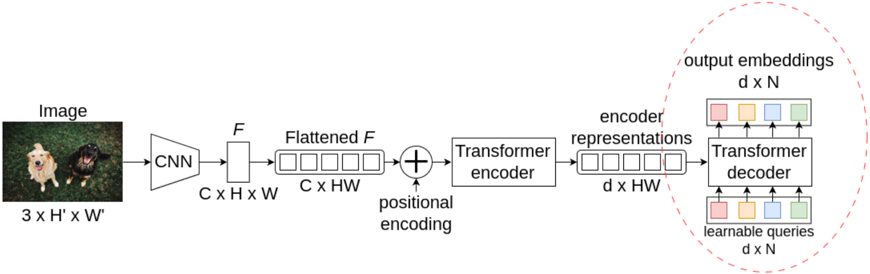
- Positional encoding is added to the flattened feature maps to provide spatial information about objects in the image
- This allows the model to understand relative positions of different parts of the image

DEtection TRansformer (DETR)



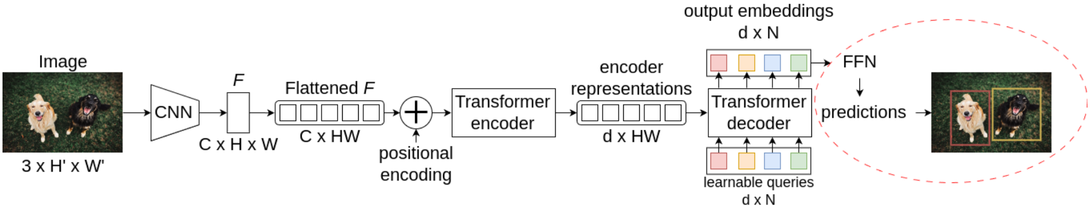
- Feature maps with positional encoding are passed through an encoder
- Encoder consists of multiple transformer encoder layers, each comprising self-attention mechanisms and feed-forward networks (FFNs)
- Self-attention mechanisms allow the model to capture global context and relationships between different parts of the image

DEtection TRansformer (DETR)



- Encoder output is then passed to the decoder along with a set of learnable queries representing possible object locations and classes
- Decoder also consists of transformer layers with self-attention and FFNs
- Decoder attends to both encoded image features and object queries to make predictions about the presence, location, and class of objects

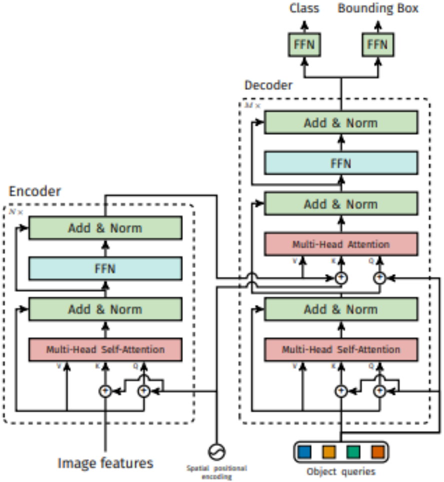
DEtection TRansformer (DETR)



- Decoder produces output embeddings for each object query
- These embeddings include class probabilities, bounding box coordinates, and a special "no object" class indicating the absence of an object
- The model predicts these outputs in parallel for all object queries, allowing end-to-end training without additional post-processing steps

DEtection TRansformer (DETR)

Transformer Architecture:



DEtection TRansformer (DETR)

Bipartite matching:

- DETR predicts N objects in one decoder pass, with N larger than typical object counts

DEtection TRansformer (DETR)

Bipartite matching:

- DETR predicts N objects in one decoder pass, with N larger than typical object counts
- Unlike traditional object detection methods (Faster-RCNN, FPNs), DETR does not rely on heuristic assignment rules for matching proposals or anchors to ground truth

DEtection TRansformer (DETR)

Bipartite matching:

- DETR predicts N objects in one decoder pass, with N larger than typical object counts
- Unlike traditional object detection methods (Faster-RCNN, FPNs), DETR does not rely on heuristic assignment rules for matching proposals or anchors to ground truth
- Instead, DETR utilizes a loss function to achieve an optimal bipartite matching

DEtection TRansformer (DETR)

Bipartite matching:

- DETR predicts N objects in one decoder pass, with N larger than typical object counts
- Unlike traditional object detection methods (Faster-RCNN, FPNs), DETR does not rely on heuristic assignment rules for matching proposals or anchors to ground truth
- Instead, DETR utilizes a loss function to achieve an optimal bipartite matching

Let y denote the set of ground truth objects padded with $\emptyset$ (no object) to match the size of $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$, the set of N predictions. A bipartite matching between these sets is found by searching for a permutation $\sigma \in \mathcal{S}_N$ with the lowest cost:

$$\hat{\sigma} = \underset{\sigma \in \mathcal{S}_N}{\operatorname{argmin}} \sum_{i}^{N} \mathcal{L}_{match} \left(y_i, \hat{y}_{\sigma(i)} \right) \tag{1}$$

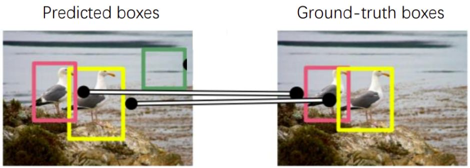
DEtection TRansformer (DETR)

Matching loss:

$$\mathcal{L}_{match}\left(y_i, \hat{y}_{\sigma(i)}\right) = -\mathbb{1}_{\{c_i \neq \emptyset\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \emptyset\}} \mathcal{L}_{box}\left(b_i, \hat{b}_{\sigma(i)}\right)$$

Every ground truth element y_i can be seen as $y_i = (c_i, b_i)$ where c_i is the target class label (which may be $\emptyset$) and $b_i \in [0,1]^4$ is a vector that defines ground truth box coordinates

For the prediction with index $\sigma(i)$, $\hat{p}_{\sigma(i)}(c_i)$ is the probability of class c_i and $\hat{b}_{\sigma(i)}$ is the predicted box



DEtection TRansformer (DETR): Loss Terms

- Following the matching loss (see Eq. 1, Slide 9), DETR optimizes object-specific losses for training. The optimal assignment $\hat{\sigma}$ is computed using the well-known Hungarian matching algorithm[2] [1]; the loss hence is termed **Hungarian loss**:

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

[2] Kuhn, The Hungarian Method for the Assignment Problem, Naval Research Logistics Quarterly, 1955

DEtection TRansformer (DETR): Loss Terms

- Following the matching loss (see Eq. 1, Slide 9), DETR optimizes object-specific losses for training. The optimal assignment $\hat{\sigma}$ is computed using the well-known Hungarian matching algorithm[2] [1]; the loss hence is termed **Hungarian loss**:

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

- Commonly used L_1 loss has different scales for small and large boxes even if their relative errors are similar. To mitigate this issue, **bounding box loss** is defined as linear combination of L_1 loss and generalized IOU loss [2], which is scale-invariant:

$$\mathcal{L}_{box}(b_i, \hat{b}_{\hat{\sigma}}(i)) = \lambda_i \mathcal{L}_{iou}(b_i, \hat{b}_{\hat{\sigma}}(i)) + \lambda_{L1} ||b_i - \hat{b}_{\hat{\sigma}}(i)||_1$$

[2] Kuhn, The Hungarian Method for the Assignment Problem, Naval Research Logistics Quarterly, 1955

DETR: Performance

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**

DETR: Limitations

Computational cost: DETR demands significant computational resources due to the quadratic complexity of the self-attention mechanism, making it impractical to handle high-resolution feature maps.

DETR: Limitations

Computational cost: DETR demands significant computational resources due to the quadratic complexity of the self-attention mechanism, making it impractical to handle high-resolution feature maps.

Long training schedules and instability: Initially, almost equal attention weights of $\frac{1}{N_k}$ are set to all pixels, where N_k is number of key elements. In the image domain, where the key elements are usually image pixels, N_k can be very large and convergence is tedious

DETR: Limitations

Computational cost: DETR demands significant computational resources due to the quadratic complexity of the self-attention mechanism, making it impractical to handle high-resolution feature maps.

Long training schedules and instability: Initially, almost equal attention weights of $\frac{1}{N_k}$ are set to all pixels, where N_k is number of key elements. In the image domain, where the key elements are usually image pixels, N_k can be very large and convergence is tedious

Limited performance on small objects: Self-attention mechanism in DETR may not effectively capture intricate details of small objects, resulting in insufficient focus on relevant image regions during detection process

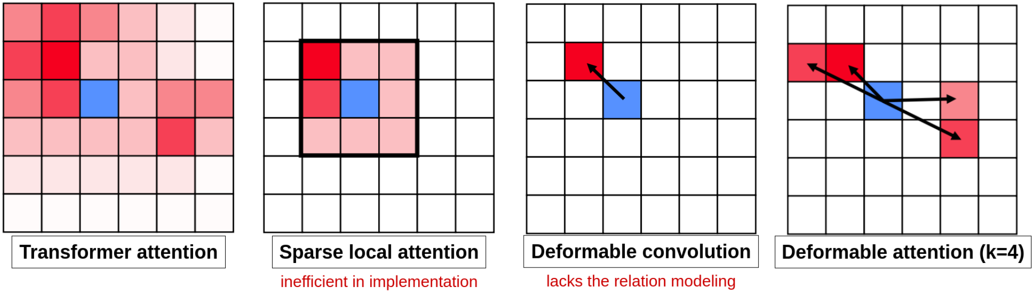
Recent DETR-like models

Question: How can we reduce the quadratic computational complexity of self-attention?

Recent DETR-like models

Question: How can we reduce the quadratic computational complexity of self-attention?

Deformable DETR [5]: For a given pixel, only attend to a few key points around it instead of attending to every other pixel in the image.



Why go beyond DETR?

Recall: Multi-Head Attention in Transformers: Let $q \in \Omega_q$ denote a query element with feature $z_q \in \mathbb{R}^C$, $k \in \Omega_k$ represent a key element with feature $x_k \in \mathbb{R}^C$, and m denote the attention head. Here, C is the feature dimension, and Ω_q and Ω_k specify the sets of query and key elements. The multi-head attention feature is then computed as:

$$\text{MultiHeadAttn}(z_q, x) = \sum_{m=1}^{M} W_m \left[\sum_{k \in \Omega_k} A_{mqk} . W'_m x_k \right] \tag{2}$$

Why go beyond DETR?

Recall: Multi-Head Attention in Transformers: Let $q \in \Omega_q$ denote a query element with feature $z_q \in \mathbb{R}^C$, $k \in \Omega_k$ represent a key element with feature $x_k \in \mathbb{R}^C$, and m denote the attention head. Here, C is the feature dimension, and Ω_q and Ω_k specify the sets of query and key elements. The multi-head attention feature is then computed as:

$$\text{MultiHeadAttn}(z_q, x) = \sum_{m=1}^{M} W_m \left[\sum_{k \in \Omega_k} A_{mqk} . W'_m x_k \right] \tag{2}$$

Two main problems:

- Initialization of attention weights A_{mqk} close to $\frac{1}{N_k}$ causes ambiguous gradients, necessitating lengthy training schedules for weights to focus on specific keys
- Computational complexity of Eq. 2 is $O(N_q C^2 + N_k C^2 + N_q N_k C)$, which is quadratic in feature map size

Deformable DETR[3]

- Inspired by deformable convolution, deformable attention module focuses on a small set of key sampling points near a reference point, irrespective of feature map size

[3] Xizhou Zhu et al, Deformable DETR: Deformable Transformers for End-to-End Object Detection, ICLR 2021

Deformable DETR[3]

- Inspired by deformable convolution, deformable attention module focuses on a small set of key sampling points near a reference point, irrespective of feature map size
- Given input feature map $x \in \mathbb{R}^{C \times H \times W}$, let q index a query element with content feature z_q and a 2D reference point p_q, then **deformable attention** feature is computed as:

$$\text{DeformAttn}(z_q, p_q, x) = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mqk} . W'_m x(p_q + \Delta p_{mqk}) \right] \quad (3)$$

where m indexes attention heads, k indexes sampled keys $(K << HW)$, Δp_{mqk}, A_{mqk} represents sampling offset and attention weight for the k^{th} sampling point in the m^{th} head

[3]Xizhou Zhu et al, Deformable DETR: Deformable Transformers for End-to-End Object Detection, ICLR 2021

Deformable DETR[3]

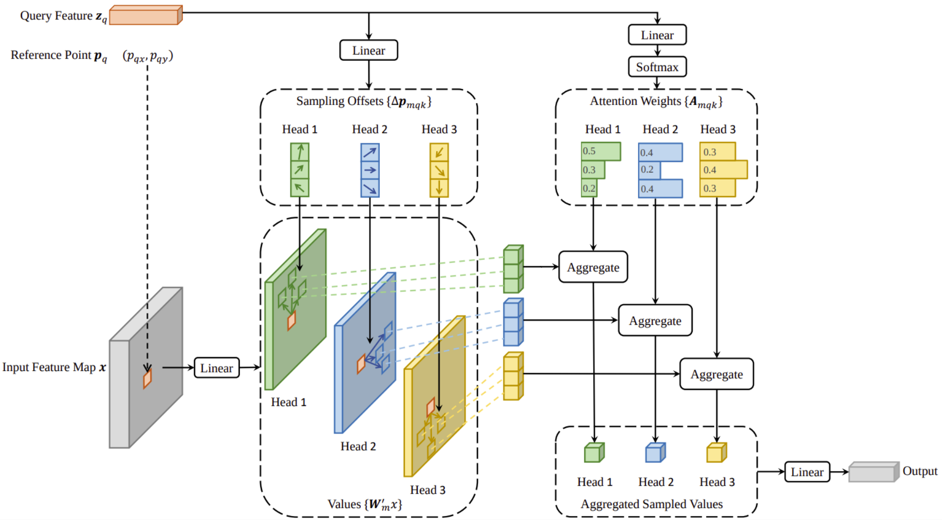
- Inspired by deformable convolution, deformable attention module focuses on a small set of key sampling points near a reference point, irrespective of feature map size
- Given input feature map $x \in \mathbb{R}^{C \times H \times W}$, let q index a query element with content feature z_q and a 2D reference point p_q, then **deformable attention** feature is computed as:

$$\text{DeformAttn}(z_q, p_q, x) = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mqk} . W'_m x(p_q + \Delta p_{mqk}) \right] \quad (3)$$

  where m indexes attention heads, k indexes sampled keys $(K << HW)$, Δp_{mqk}, A_{mqk} represents sampling offset and attention weight for the k^{th} sampling point in the m^{th} head
- Let N_q be number of query elements; complexity of deformable attention module is $O(2N_qC^2 + \min(HWC^2, N_qKC^2))$. For encoder, it becomes $O(HWC^2)$ which is linear to feature map size, and for decoder, it is $O(NKC^2)$, which is independent of spatial size

[3] Xizhou Zhu et al, Deformable DETR: Deformable Transformers for End-to-End Object Detection, ICLR 2021

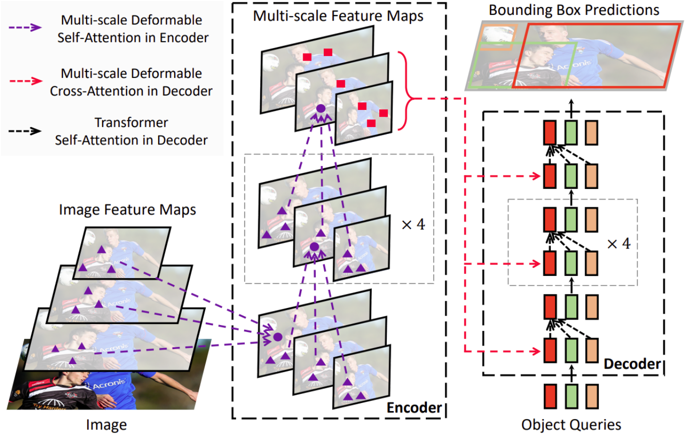
Deformable DETR



Deformable DETR: Multi-Scale

Use multi-scale input features $\{x_l\}_{l=1}^{L}$

MSDeformAttn$(z_q, \hat{p}_q, \{x^l\}_{l=1}^{L})$:
$\sum_{m=1}^{M} W_m [\sum_{l=1}^{L} \sum_{k=1}^{K} A_{mlqk} \cdot W'_m x^l(\phi_l(\hat{p}_q) + \Delta p_{mlqk})]$



Deformable DETR: Other Improvements

Iterative Bounding Box Refinement: Deformable DETR established a simple and effective iterative bounding box refinement mechanism to improve detection performance. Here, each decoder layer refines bounding boxes based on predictions from previous layer

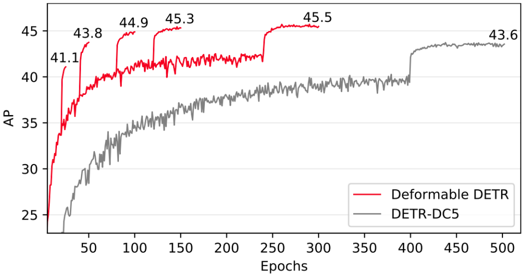
Deformable DETR: Other Improvements

Iterative Bounding Box Refinement: Deformable DETR established a simple and effective iterative bounding box refinement mechanism to improve detection performance. Here, each decoder layer refines bounding boxes based on predictions from previous layer

Two-Stage Deformable DETR:

- Inspired by two-stage object detectors, Deformable DETR explored a variant for generating region proposals as the first stage
- Generated region proposals are fed into the decoder as object queries for refinement, forming a two-stage Deformable DETR

Deformable DETR: Performance



Method	Epochs	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	params	FLOPs	Training GPU hours	Inference FPS
Faster R-CNN + FPN	109	42.0	62.1	45.5	26.6	45.4	53.4	42M	180G	380	26
DETR	500	42.0	62.4	44.2	20.5	45.8	61.1	41M	86G	2000	28
DETR-DC5	500	43.3	63.1	45.9	22.5	47.3	61.1	41M	187G	7000	12
DETR-DC5	50	35.3	55.7	36.8	15.2	37.5	53.6	41M	187G	700	12
DETR-DC5$^+$	50	36.2	57.0	37.4	16.3	39.2	53.9	41M	187G	700	12
Deformable DETR	50	43.8	62.6	47.7	26.4	47.1	58.0	40M	173G	325	19
+ iterative bounding box refinement	50	45.4	64.7	49.0	26.8	48.3	61.7	40M	173G	325	19
++ two-stage Deformable DETR	50	46.2	65.2	50.0	28.8	49.2	61.7	40M	173G	340	19

Improving DETR

Several improvements to DETR, beyond Deformable DETR, proposed to enhance its performance:

- DAB-DETR [7] proposed to formulate DETR queries as dynamic anchor boxes (DAB), which bridges gap between classical anchor-based detectors and DETR-like ones

Improving DETR

Several improvements to DETR, beyond Deformable DETR, proposed to enhance its performance:

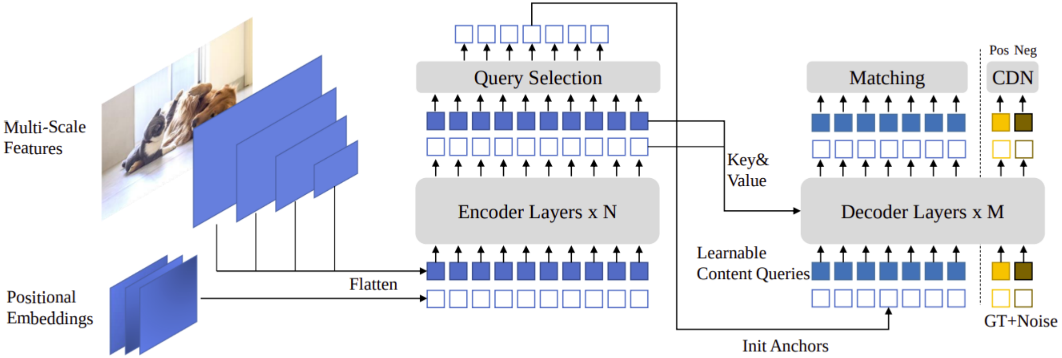
- DAB-DETR [7] proposed to formulate DETR queries as dynamic anchor boxes (DAB), which bridges gap between classical anchor-based detectors and DETR-like ones
- DN-DETR [6] addressed instability of bipartite matching by introducing a denoising (DN) technique

Improving DETR

Several improvements to DETR, beyond Deformable DETR, proposed to enhance its performance:

- DAB-DETR [7] proposed to formulate DETR queries as dynamic anchor boxes (DAB), which bridges gap between classical anchor-based detectors and DETR-like ones
- DN-DETR [6] addressed instability of bipartite matching by introducing a denoising (DN) technique
- More recent methods include Cascade DETR [11], DINO [12] and Grounding DINO [10]

DINO: DETR with Improved Denoising Anchor Boxes[4]



DINO incorporates elements from DN-DETR (denoising training) [6], a "look forward twice" scheme for better box prediction, and a mixed query selection method for anchor initialization

[4] Hao Zhang et al, DINO: DETR with Improved DeNoising Anchor Boxes for End-to-End Object Detection, ICLR 2023

DINO: DETR with Improved Denoising Anchor Boxes

Adopted from previous work

- Following DN-DETR [6], DINO adds ground truth labels and boxes with noise into the Transformer decoder layers to help stabilize bipartite matching during training
- Following Deformable DETR, DINO adopts deformable attention for its computational efficiency
- Following DAB-DETR [7], DINO formulates queries as dynamic anchor boxes and refines them step-by-step across decoder layers

DINO: DETR with Improved Denoising Anchor Boxes

Adopted from previous work

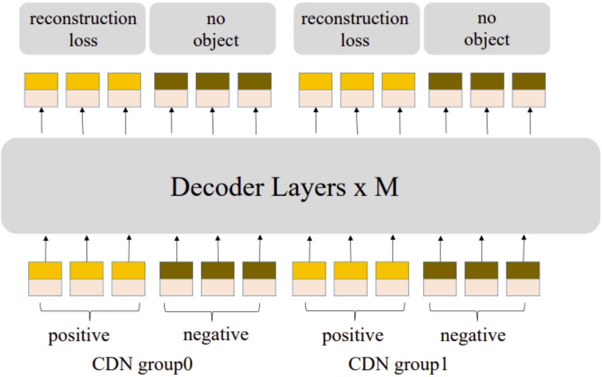
- Following DN-DETR [6], DINO adds ground truth labels and boxes with noise into the Transformer decoder layers to help stabilize bipartite matching during training
- Following Deformable DETR, DINO adopts deformable attention for its computational efficiency
- Following DAB-DETR [7], DINO formulates queries as dynamic anchor boxes and refines them step-by-step across decoder layers

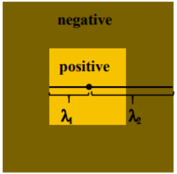
New ideas proposed

- **Contrastive denoising training** to improve one-to-one matching
- **Mixed query selection** that helps better initialize queries
- "**Look forward twice**" scheme to leverage refined box information from later decoder layers and to help optimize parameters of adjacent early layers

DINO: Contrastive DeNoising Training

DN-DETR leverages DN queries to learn predictions based on nearby ground truth boxes. However, it cannot predict "no object" for anchors without nearby objects. DINO hence proposes Contrastive DeNoising Training (CDN) to solve this problem.





DINO: Contrastive DeNoising Training

Implementation:

- Unlike DN-DETR, DINO uses two hyper-parameters λ_1 and λ_2, where $\lambda_1 < \lambda_2$, to define positive and negative queries
- Positive queries: noise $< \lambda_1$; expected to reconstruct corresponding ground truth boxes
- Negative queries: $\lambda_1 <$ noise $< \lambda_2$; expected to predict "no object"

DINO: Contrastive DeNoising Training

Implementation:

- Unlike DN-DETR, DINO uses two hyper-parameters λ_1 and λ_2, where $\lambda_1 < \lambda_2$, to define positive and negative queries
- Positive queries: noise $< \lambda_1$; expected to reconstruct corresponding ground truth boxes
- Negative queries: $\lambda_1 <$ noise $< \lambda_2$; expected to predict "no object"

Analysis:

- When multiple anchors are close to an object, the model may get confused on which anchor to choose
- This can lead to two problemss: *Duplicate predictions* and *Incorrect anchor selection*
- With CDN queries, it is shown that DINO can distinguish minor differences between anchors to avoid duplicate predictions and reject farther anchors

DINO: Mixed Query Selection

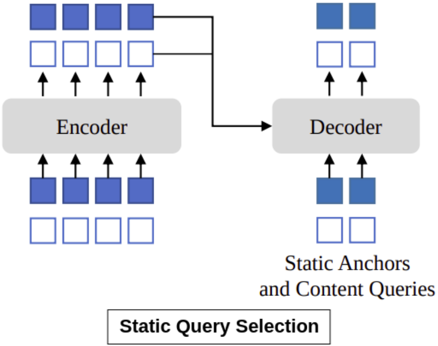
Queries in DETR are formed by two parts: a *content* part and a *positional* part

DINO: Mixed Query Selection

Queries in DETR are formed by two parts: a *content* part and a *positional* part

In DETR and DN-DETR, decoder queries are static embeddings, i.e. encoder features from given image are not part of the queries

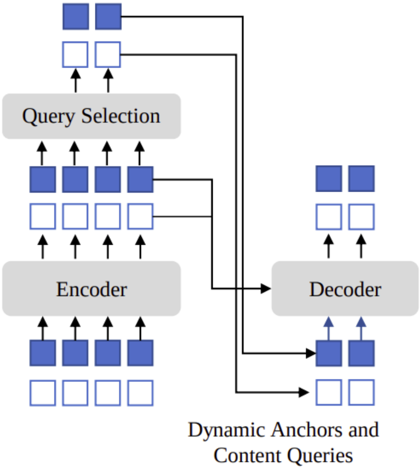
They learn anchors or positional queries from training data directly and set content queries as all zero vectors



DINO: Mixed Query Selection

Deformable DETR learns both positional and content queries generated by a linear transform of selected features

It has a query selection module (in its "two-stage" variant), which selects top-K encoder features from last encoder layer as priors to enhance decoder queries

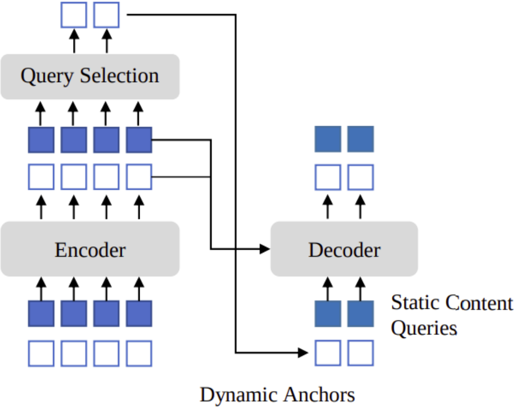


Pure Query Selection

DINO: Mixed Query Selection

In Deformable DETR, selected encoder features are preliminary content features without refinement; this can be ambiguous and misleading to the decoder

To mitigate this, DINO only initializes anchor boxes using the position information associated with the selected top-K features, but initializes the learnable content queries independently ("static" in figure means that they are kept the same for different images at inference)

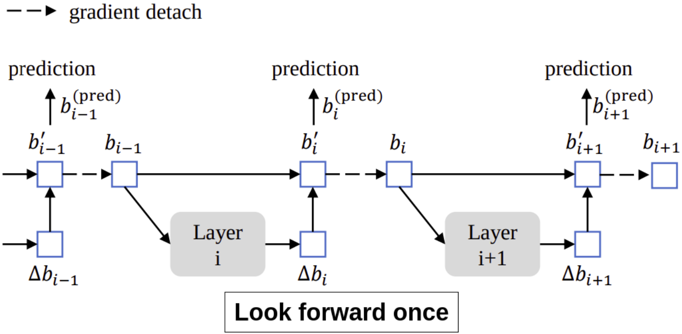


Mixed Query Selection

DINO: Look Forward Twice

Iterative box refinement in Deformable DETR [5] blocks gradient backpropagation to stabilize training

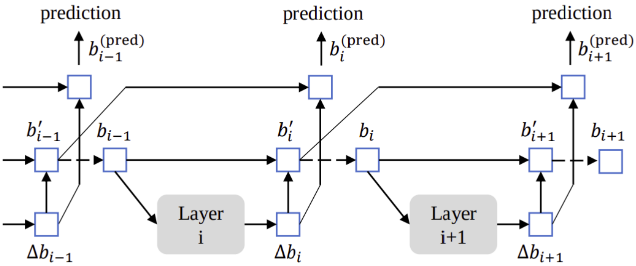
DINO terms this method as "look forward once" since parameters of layer i are updated based on auxiliary loss of boxes b_i only



DINO: Look Forward Twice

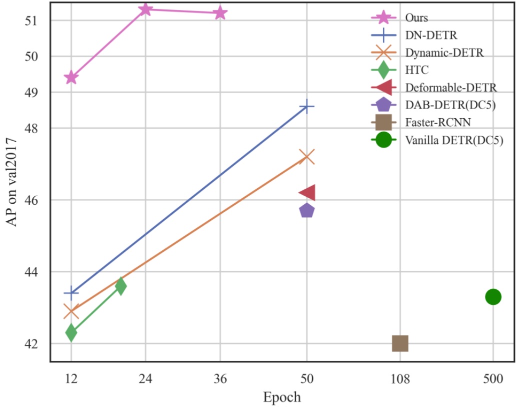
DINO conjectures that improved box information from a later layer could be more helpful in correcting the box prediction in its adjacent early layer

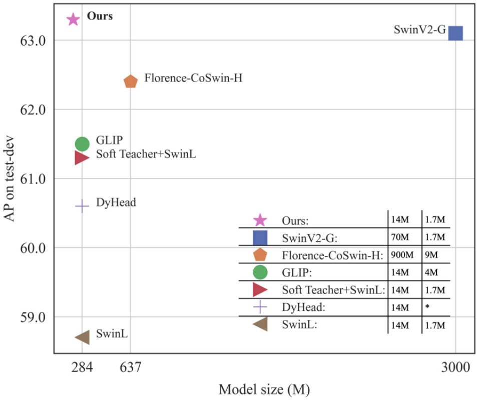
It hence proposes "look forward twice" to perform box update, where parameters of layer i are influenced by losses of both layer i and layer $i+1$



Look forward twice

DINO: Performance





More Information

Resources

- HuggingFace Link for Object Detection
- Object Detection with Transformers: A Review

References

[1] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* (1955).

[2] Hamid Rezatofighi et al. "Generalized intersection over union: A metric and a loss for bounding box regression". In: *CVPR*. 2019.

[3] Nicolas Carion et al. "End-to-end object detection with transformers". In: *ECCV*. 2020.

[4] Depu Meng et al. "Conditional DETR for fast training convergence". In: *ICCV*. 2021.

[5] Xizhou Zhu et al. "Deformable DETR: Deformable Transformers for End-to-End Object Detection". In: *ICLR*. 2021.

[6] Feng Li et al. "DN-DETR: Accelerate DETR training by introducing query denoising". In: *CVPR*. 2022, pp. 13619–13627.

[7] Shilong Liu et al. "DAB-DETR: Dynamic Anchor Boxes are Better Queries for DETR". In: *ICLR*. 2022.

[8] Byungseok Roh et al. "Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity". In: *ICLR*. 2022.

[9] Yingming Wang et al. "Anchor DETR: Query design for transformer-based detector". In: *AAAI*. 2022.

[10] Shilong Liu et al. "Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection". In: *arXiv preprint arXiv:2303.05499* (2023).