

VISION TRANSFORMERS AND OBJECT DETECTION

Sergey Nikolenko

Harbour Space University

June 22, 2026

Random facts:

- on June 22, 1633, the Roman Inquisition forced the 69-year-old Galileo Galilei to kneel and formally recant his claim that the Earth moves around the Sun; an unconfirmed legend has it that as he rose he muttered "E pur si muove" ("and yet it moves")
- on June 22, 1986, in a single World Cup quarter-final against England, Diego Maradona scored both the "Hand of God" goal (illegal goal the referee missed) and, four minutes later, the "Goal of the Century"
- on June 22, 1969, the Cuyahoga River in Cleveland was so clogged with oil and industrial waste that it caught fire; the blaze lasted barely half an hour, but the public outrage it caused helped bring about the U.S. Environmental Protection Agency
- on June 22, 1978, astronomer James Christy discovered Charon, Pluto's largest moon, after noticing a strange recurring bump on photographs of Pluto; Charon is so large relative to Pluto that the two bodies actually orbit a point in the empty space between them
- on June 22, 1948, the HMT Empire Windrush docked near London carrying some 800 passengers from the Caribbean, a voyage now remembered as the symbolic beginning of modern multicultural Britain



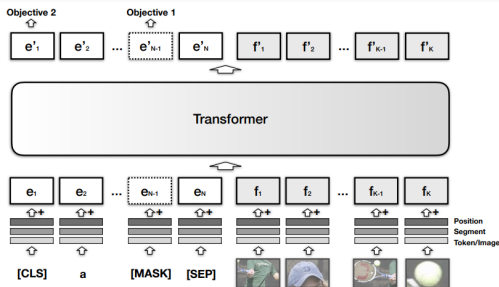
VISION TRANSFORMERS

VISUAL BERT

- Images are not sequences, right? Well...
- Visual BERT: let's model captions and images jointly
- Use a fixed pretrained object detection system such as Faster R-CNN, cut the objects out, embed them into vectors via CNNs and special positional embeddings, and feed into a single Transformer with the caption

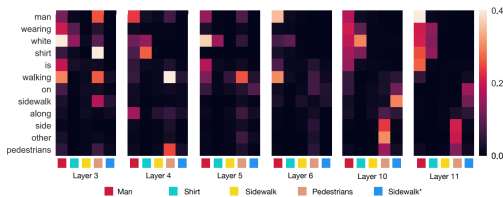
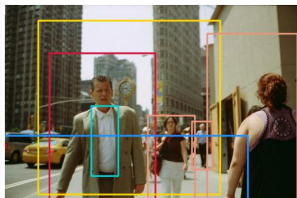


A person hits a ball with a tennis racket

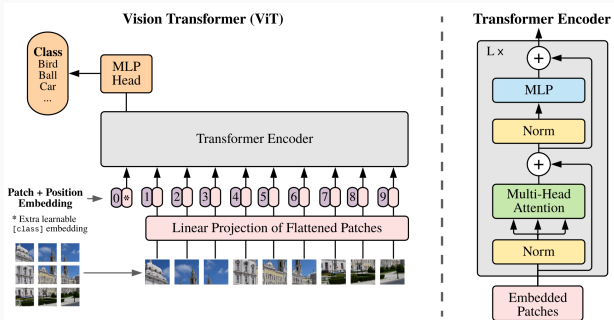


VISUAL BERT

- Pretraining: masked language modeling + sentence-image prediction (distinguish whether a given caption matches the image)
- Sample attention heads show how words from the caption actually do attend to the corresponding objects

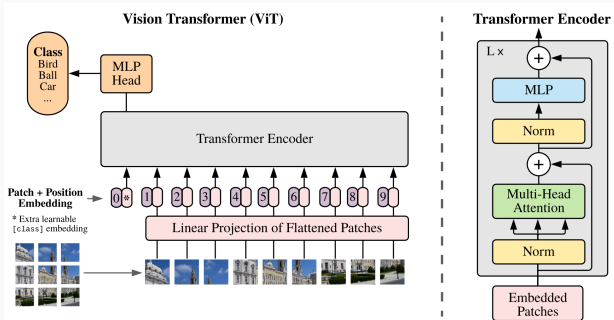


VISUAL TRANSFORMER (ViT)



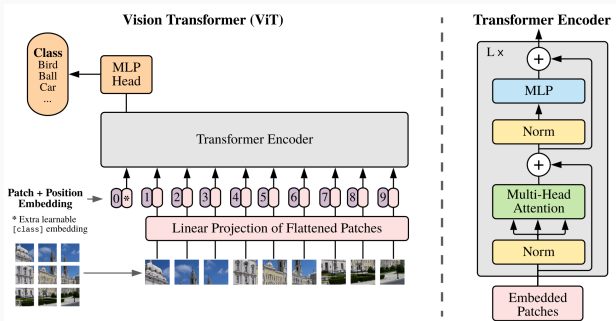
- The most successful Transformer-based architecture for images: *Vision Transformer* (ViT; Dosovitsky et al., 2020)
- ViT is again a straightforward modification of BERT, but now there is no text at the input, only image-based tokens

VISUAL TRANSFORMER (ViT)



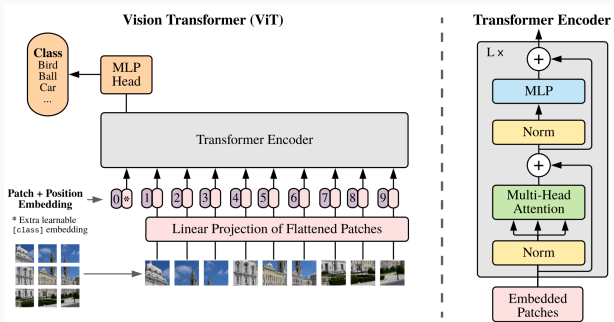
- The input image is cut into small patches: an $H \times W$ image with C channels $\mathbf{x} \in \mathbb{R}^{H \times W \times C}$ becomes a sequence of patches $\mathbf{x}_p \in \mathbb{R}^{N \times P^2 \times C}$, where $N = HW/P^2$ is the number of $P \times P$ patches that fit

VISUAL TRANSFORMER (ViT)



- Patches are turned into embeddings via a simple linear projection, and then the sequence is fed into a Transformer encoder just like BERT
- Pretraining: masked patch modeling just like BERT, replacing half of the input embeddings with the same learnable [mask] embedding

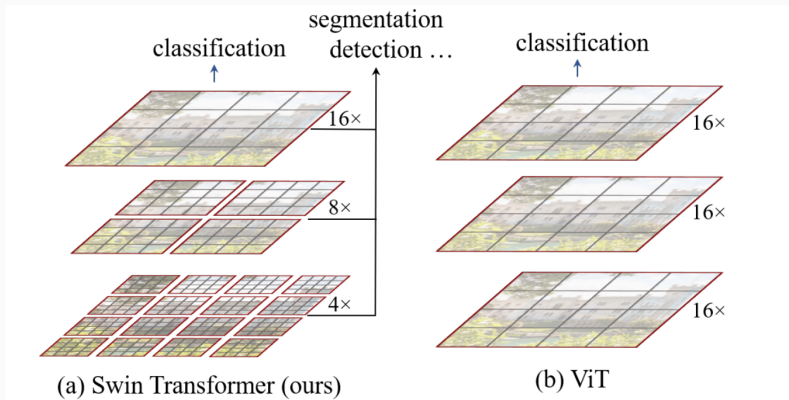
VISUAL TRANSFORMER (ViT)



- ViT uses positional encodings to add information about the sequence; interestingly, it is the same positional encoding as in the original Transformer even though the geometry is 2D now
- Dosovitsky et al. experimented with 2D encodings, and it did not make any difference

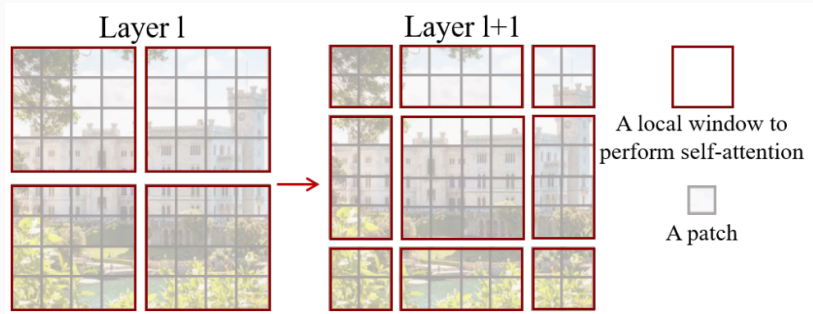
SWIN TRANSFORMERS

- *Swin Transformer* (Liu et al., 2021; **shifted windows**): similar to ViT but in a hierarchical fashion



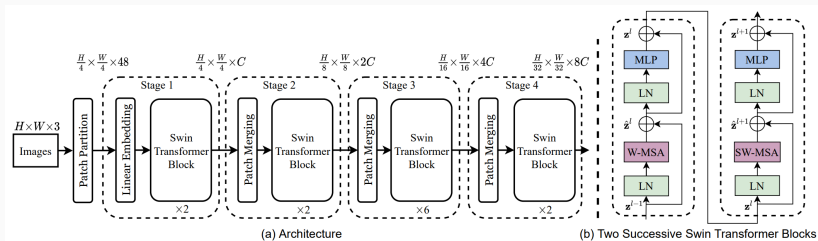
SWIN TRANSFORMERS

- It processes image patches on several scales, computing self-attention across patches in a convolutional-like architecture
- The windows shift between self-attention layers, providing connections between parts of the image



SWIN TRANSFORMERS

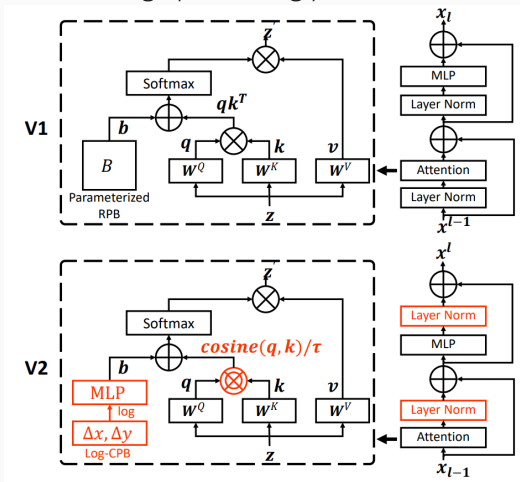
- The architecture itself goes on reducing the geometry, like classical CNN backbones:



- Swin can scale up to larger input resolutions and can be used for dense recognition tasks such as image segmentation

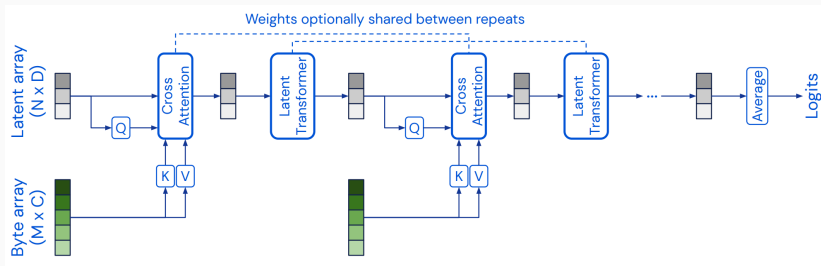
SWIN TRANSFORMERS

- A later iteration, *Swin Transformer v2* (Liu et al., 2022), scaled the Swin Transformer up to 3 billion parameters and allowed for training with images up to 1536×1536 pixels, further improving state of the art in image processing problems across the board



PERCEIVER

- *DeepMind's Perceiver* (Jaegle et al., 2021a) is a general-purpose architecture that can process numerous different modalities



- The main idea is to avoid the quadratic bottleneck of self-attention by using lower-dimensional latent units
- It's quadratic in the number of queries, so we use a small vector of latents for queries and add large byte arrays for K and V

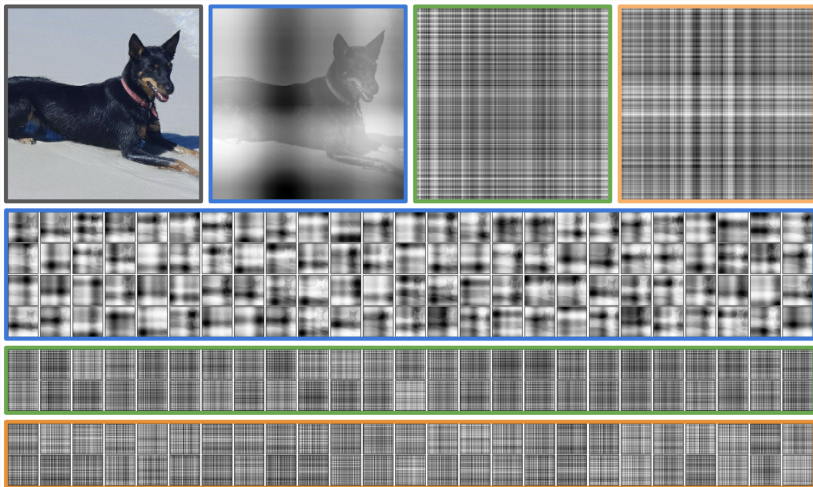
- Trained on images, point clouds, audio, and video, with no modality-specific encoders and no change to the architecture!



ResNet-50 (He et al., 2016)	77.6
ViT-B-16 (Dosovitskiy et al., 2021)	77.9
ResNet-50 (FF)	73.5
ViT-B-16 (FF)	76.7
Transformer (64x64, FF)	57.0
Perceiver (FF)	78.0

	Raw	Perm.	Input RF
ResNet-50 (FF)	73.5	39.4	49
ViT-B-16 (FF)	76.7	61.7	256
Transformer (64x64) (FF)	57.0	57.0	4,096
Perceiver:			
(FF)	78.0	78.0	50,176
(Learned pos.)	70.9	70.9	50,176

- Attention maps go down to individual pixels:

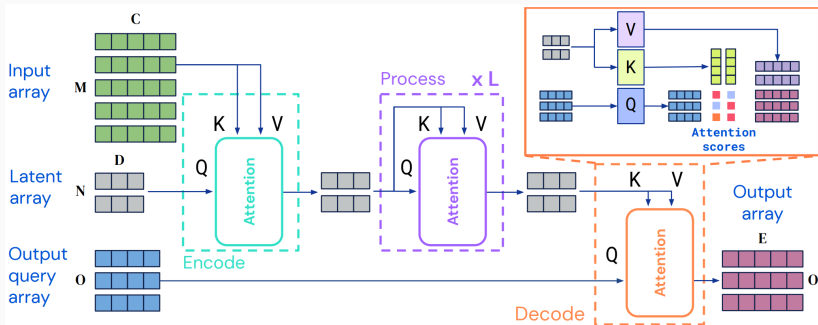


- *Perceiver IO* (Jaegle et al., 2021b), the next version that concentrates on high-dimensional *outputs* as well
- Can process even more various structured data: multi-task language understanding, dense visual tasks like optical flow, hybrid dense/sparse multimodal tasks such as video+audio+class autoencoding, and tasks with symbolic outputs like StarCraft II



PERCEIVER

- Perceiver IO architecture:



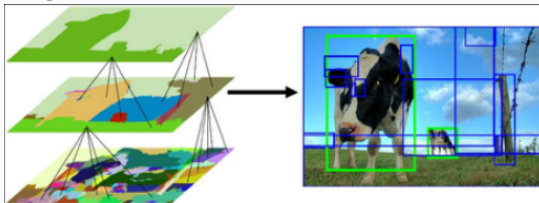
- The actual output queries are constructed automatically by combining a set of vectors that describe properties of the current output such as position coordinates

OBJECT DETECTION

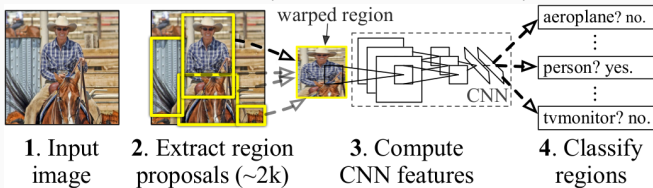
- We have already discussed basic classification architectures.
- But for computer vision, classification is not enough; we need to recognize objects on photographs.
- How do we use CNNs to understand *where* the objects are?
- This is exactly the object detection problem.

R-CNN

- R-CNN: Region-based ConvNet (Girshick et al., 2014).
- Basic idea: look for bounding boxes with some external algorithm (e.g., selective search).



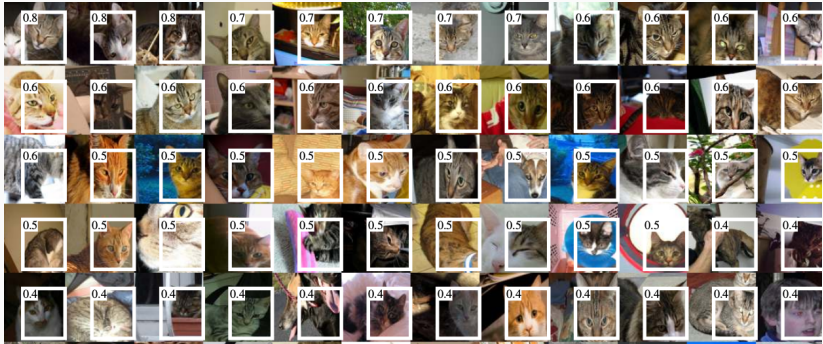
- Then extract CNN features (from a CNN trained on ImageNet and fine-tuned on the necessary dataset) and classify.



- Visualizing the result: the image shows regions of activation for a neuron from a high layer.

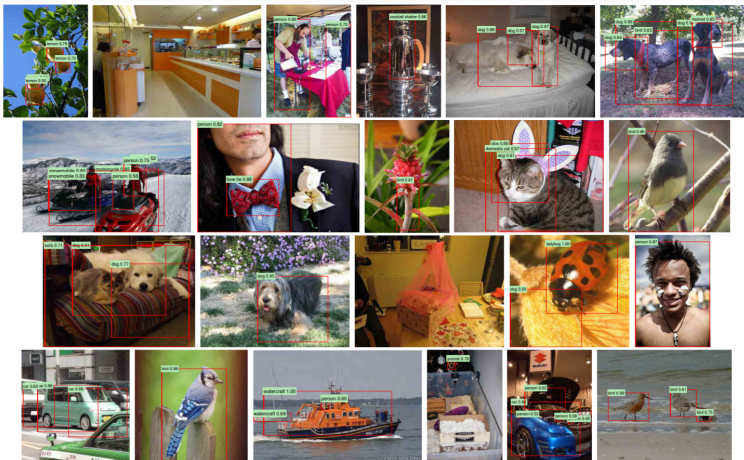


- Visualizing the result: the image shows regions of activation for a neuron from a high layer.



R-CNN

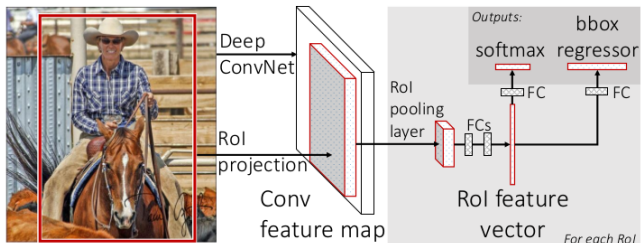
- Plus a bounding box regression: we localize with a linear regression on the features from the last pooling layer.
- Excellent for its time, R-CNN won ILSVRC2013 (200 classes).



- R-CNN was still the best at its time.
- But R-CNN has many problems:
 - we have to train it in several steps (first CNN, then SVM on CNN features, then bounding box regressors)
 - it is very long to train, for technical reasons (we need to extract features from each box in the training set and write it to disk to train SVM and regressors);
 - recognition is very slow (47s per image even on a GPU).
- The main reason is that we need to make a pass through the CNN for every region.

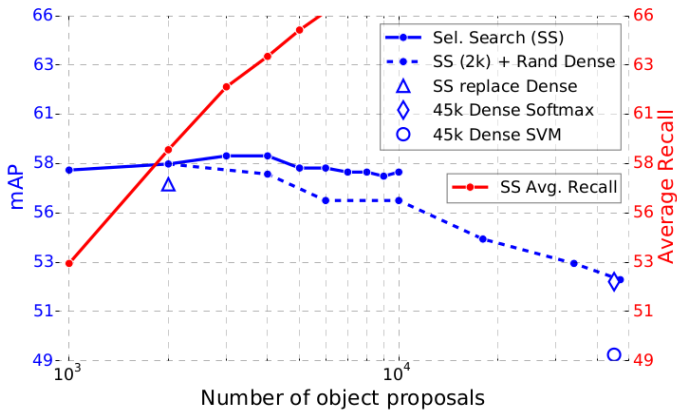
FAST R-CNN

- Hence, *Fast R-CNN* (Girshick, 2014).
- RoI (region of interest) projection collects features from a region.
- One pass of the main CNN for the whole image.
- Loss = classification error + bounding box regression error.
- Two orders of magnitude faster, no loss of quality.



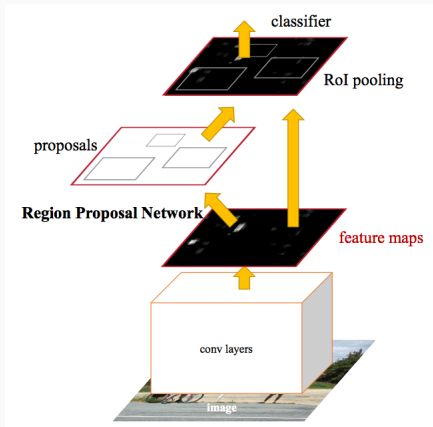
FAST R-CNN

- The candidates themselves still come from selective search.
- It turned out that it does not help much to add more candidates.



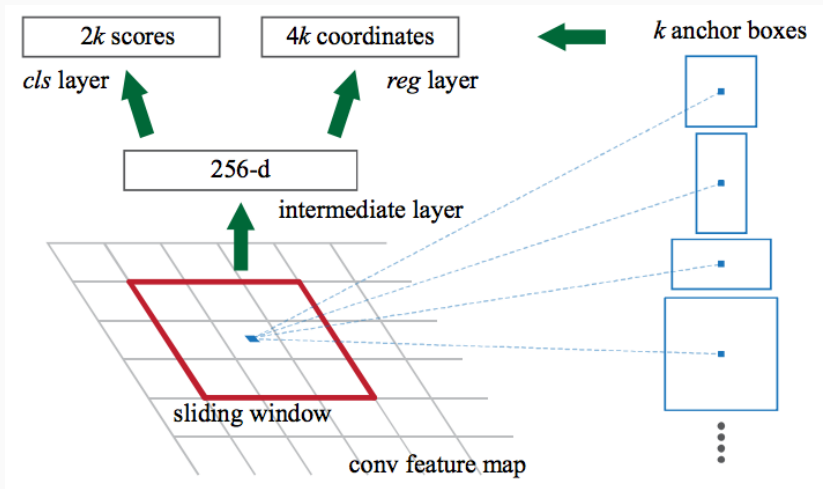
FASTER R-CNN

- One more bottleneck left: selective search to choose bounding boxes.
- It turns out that this can also be embedded into the network!
- *Faster R-CNN* (Ren et al., 2015) adds a separate *Region Proposal Network*:



FASTER R-CNN

- It evaluates each individual possibility from a set of predefined anchor boxes:



FASTER R-CNN

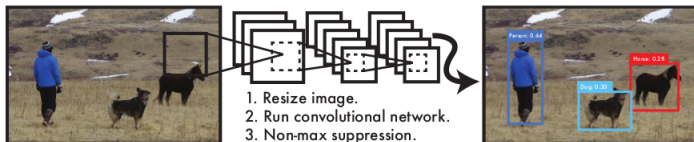
- A very good detector, best at the time for all evaluation metrics and still relevant.
- Pascal VOC 2007:

method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9 [†]	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	198
RPN+VGG, shared	300	07+12	73.2	198

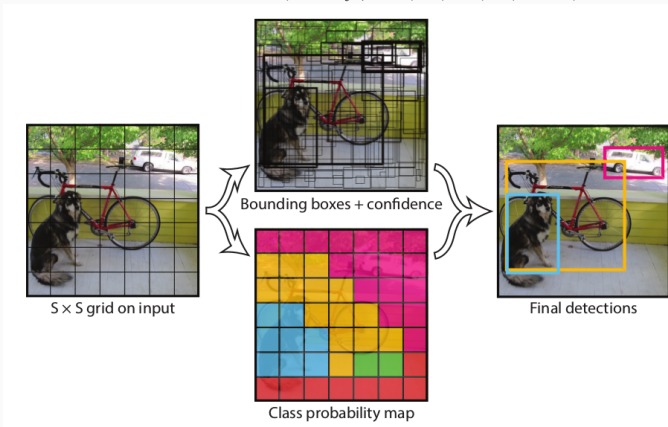
- Pascal VOC 2012:

method	# proposals	data	mAP (%)
SS	2k	12	65.7
SS	2k	07++12	68.4
RPN+VGG, shared [†]	300	12	67.0
RPN+VGG, shared [‡]	300	07++12	70.4

- You only look once (YOLO; Redmon et al., 2016).
- Look for both bounding boxes and the objects themselves in a single pass.

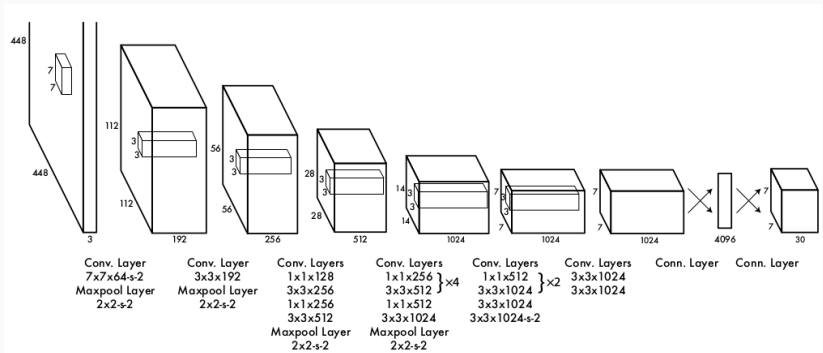


- Idea: split the image into an $S \times S$ grid.
- In each cell, predict both bounding boxes and class probabilities; then simply $p(\text{class}_i \mid \text{obj})p(\text{obj})p(\text{bbox})$.



- The network's output is a tensor of size $S \times S \times (5B + C)$, where each bbox consists of (x, y, w, h) and certainty.

- YOLO architecture:



- And all this is trained end-to-end, with a single loss function:

$$\begin{aligned}
 \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} & \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
 + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} & \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
 + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} & (C_i - \hat{C}_i)^2 \\
 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} & (C_i - \hat{C}_i)^2 \\
 + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} & (p_i(c) - \hat{p}_i(c))^2
 \end{aligned}$$

where $\mathbb{1}_{ij}^{\text{obj}} = 1$ iff the j th bbox predictor in cell i is responsible for predicting an object which is really there.

- We get slightly worse results than Faster R-CNN but much faster and out of the box:

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	155
YOLO	2007+2012	63.4	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

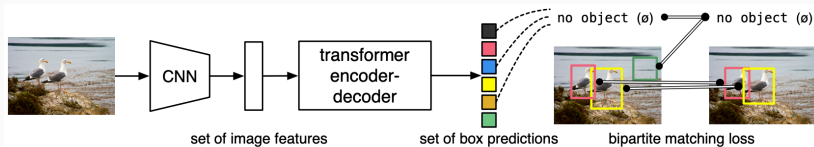
VOC 2012 test	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
MR_CNN_MORE_DATA [11]	73.9	85.5	82.9	76.6	57.8	62.7	79.4	77.2	86.6	55.0	79.1	62.2	87.0	83.4	84.7	78.9	45.3	73.4	65.8	80.3	74.0
HyperNet_VGG	71.4	84.2	78.5	73.6	55.6	53.7	78.7	79.8	87.7	49.6	74.9	52.1	86.0	81.7	83.3	81.8	48.6	73.5	59.4	79.9	65.7
HyperNet_SP	71.3	84.1	78.3	73.3	55.5	53.6	78.6	79.6	87.5	49.5	74.9	52.1	85.6	81.6	83.2	81.6	48.4	73.2	59.3	79.7	65.6
Fast R-CNN + YOLO	70.7	83.4	78.5	73.5	55.8	43.4	79.1	73.1	89.4	49.4	75.5	57.0	87.5	80.9	81.0	74.7	41.8	71.5	68.5	82.1	67.2
MR_CNN_S_CNN [11]	70.7	85.0	79.6	71.5	55.3	57.7	76.0	73.9	84.6	50.5	74.3	61.7	85.5	79.9	81.7	76.4	41.0	69.0	61.2	77.7	72.1
Faster R-CNN [28]	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
DEEP_ENS_COCO	70.1	84.0	79.4	71.6	51.9	51.1	74.1	72.1	88.6	48.3	73.4	57.8	86.1	80.0	80.7	70.4	46.6	69.6	68.8	75.9	71.4
NoC [29]	68.8	82.8	79.0	71.6	52.3	53.7	74.1	69.0	84.9	46.9	74.3	53.1	85.0	81.3	79.5	72.2	38.9	72.4	59.5	76.7	68.1
Fast R-CNN [14]	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	87.5	80.5	80.8	72.0	35.1	68.3	65.7	80.4	64.2
UMICH_FGS_STRUCT	66.4	82.9	76.1	64.1	44.6	49.4	70.3	71.2	84.6	42.7	68.6	55.8	82.7	77.1	79.9	68.7	41.4	69.0	60.0	72.0	66.2
NUS_NIN_C2000 [7]	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
BabyLearning [7]	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN	62.4	77.9	73.1	62.6	39.5	43.3	69.1	66.4	78.9	39.1	68.1	50.0	77.2	71.3	76.1	64.7	38.4	66.9	56.2	66.9	62.7
R-CNN VGG BB [13]	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
R-CNN VGG [13]	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
YOLO	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
Feature Edit [33]	56.3	74.6	69.1	54.4	39.1	33.1	65.2	62.7	69.7	30.8	56.0	44.6	70.0	64.4	71.1	60.2	33.3	61.3	46.4	61.7	57.8
R-CNN BB [13]	53.3	71.8	65.8	52.0	34.1	32.6	59.6	60.0	69.8	27.6	52.0	41.7	69.6	61.3	68.3	57.8	29.6	57.8	40.9	59.3	54.1
SDS [16]	50.7	69.7	58.4	48.5	28.3	28.8	61.3	57.5	70.8	24.1	50.7	35.9	64.9	59.1	65.8	57.1	26.0	58.8	38.6	58.9	50.7
R-CNN [13]	49.6	68.1	63.8	46.1	29.4	27.9	56.6	57.0	65.9	26.5	48.7	39.5	66.2	57.3	65.4	53.2	26.2	54.5	38.1	50.6	51.6

- Classical detectors – R-CNN, Faster R-CNN, YOLO, SSD – all rely on hand-designed components: a dense grid of *anchor boxes*, region proposals, and *non-maximum suppression* (NMS) to remove duplicate predictions.
- None of these pieces are learned end-to-end: anchors and NMS need careful manual tuning, and NMS is a non-differentiable post-processing step bolted on after the network.
- Can we instead treat detection as a direct *set prediction* problem – let a Transformer predict the whole set of objects at once, with no anchors and no NMS?
- This is exactly the idea behind DETR, and it opens the door to the transformer-based detectors that dominate the field today.

OBJECT DETECTION WITH TRANSFORMERS

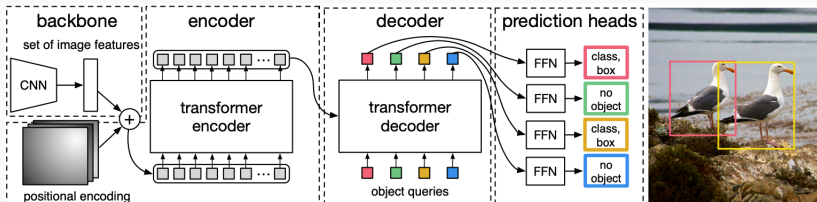
DETR AND ITS EXTENSIONS

- DETR (DEtection TRansformer; Carion et al., 2020) — a direct adaptation of transformers
- Object detection is a set prediction task; DETR adds a set-based global loss function that gives unique predictions through bipartite matching, and an encoder-decoder architecture based on a transformer



DETR AND ITS EXTENSIONS

- DETR uses a traditional convolutional backbone (ResNet in Carion et al., but you can use anything)
- Then DETR flattens the features, adds positional encoding, and mixes the features using an encoder
- The decoder takes as input a small number of *object queries*, learnable positional embeddings, with attention to the encoder output
- The output passes through a small feedforward network that predicts the class and bbox (or "no object")



- The loss function comes from bipartite matching:
 - for a set of predictions $\hat{\mathbf{y}} = \{\hat{y}_i\}_{i=1}^N$ and ground truth objects $\mathbf{y} = \{y_i\}_{i=1}^N$ (plus $\emptyset$, since N must be large), we find a permutation $\sigma_* = \arg \min_{\sigma} \sum_{i=1}^N l_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ using the Hungarian algorithm, where l_{match} is the matching cost;
 - l_{match} combines prediction accuracy for $y_i = (c_i, b_i)$, where c_i is the class label (possibly $\emptyset$), and $b_i \in \mathbb{R}^4$ is the bbox:

$$l_{\text{match}}(y_i, \hat{y}_{\sigma(i)}) = -[c_i \neq \emptyset] \hat{p}_{\sigma(i)}(c_i) + [c_i \neq \emptyset] l_{\text{box}}(b_i, \hat{b}_{\sigma(i)});$$

here l_{box} is a combination of IoU and L_1 loss functions;

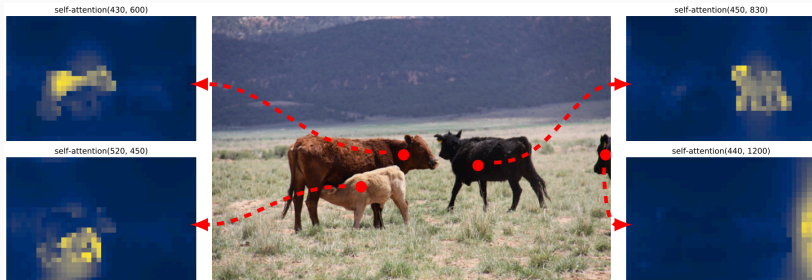
- after matching, the actual loss function is

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

now we also predict $c_i = \emptyset$, but in practice they get lower weight.

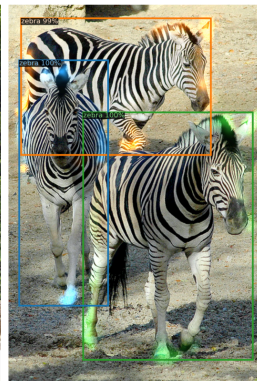
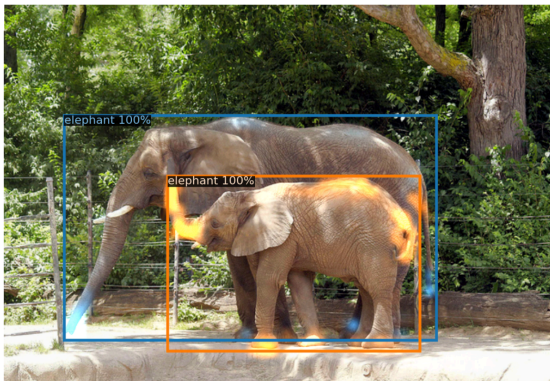
DETR AND ITS EXTENSIONS

- Good results, and DETR is able to distribute object instances across attention heads:



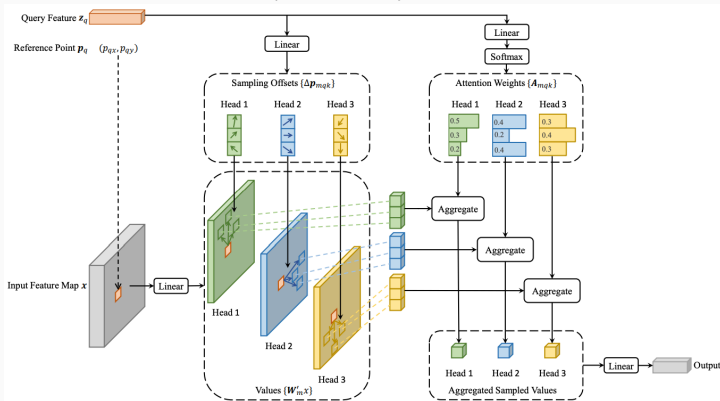
DETR AND ITS EXTENSIONS

- Decoder attention for each object is localized on boundaries
- Most likely, DETR separates instances with global attention in the encoder, and the decoder only needs to figure out classes and object boundaries



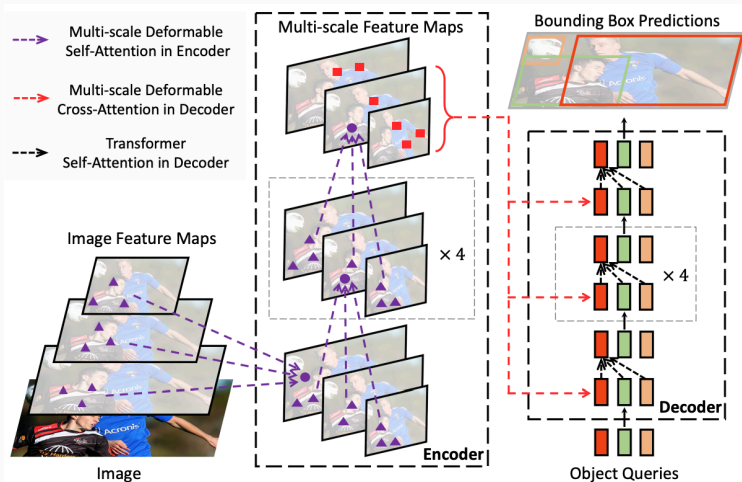
DETR AND ITS EXTENSIONS

- DETR works well, but not very well on small objects, and quadratic complexity makes multi-scale detection a very difficult task
- Deformable DETR (Zhu et al., 2021): let's use deformable attention and attend only to a few pixels



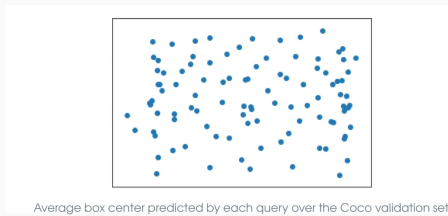
DETR AND ITS EXTENSIONS

- General architecture:



DETR AND ITS EXTENSIONS

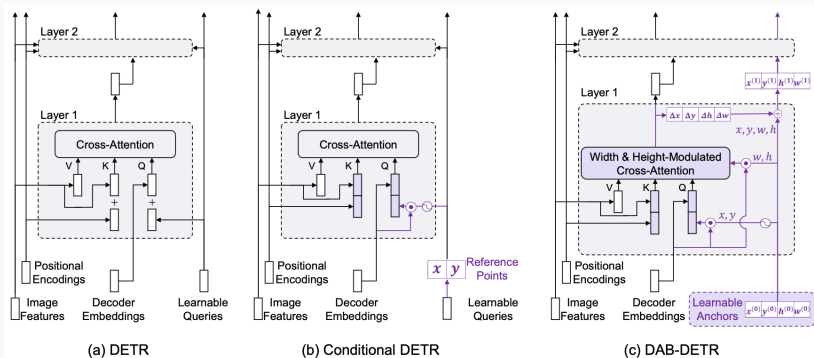
- DETR's positional queries are essentially flexible learnable anchors for bounding boxes; they are trained globally and do not depend on the current image



- Learning spatial queries from data (without any fixed anchor boxes) is an important part of DETR, but DETR trains very, very slowly, and spatial queries are a big factor in this
- How can we improve this?

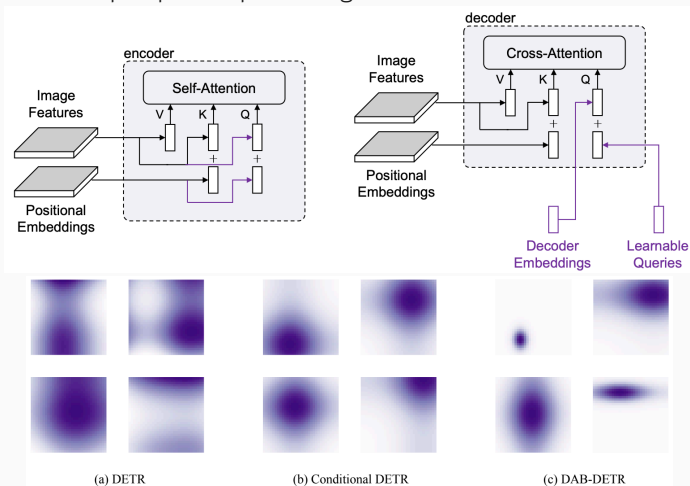
DETR AND ITS EXTENSIONS

- DAB-DETR (Dynamic Anchor Box DETR; Liu et al., 2022): let's return to the idea of anchor boxes
- Learnable anchors are gradually refined from layer to layer in the decoder, and now they have specific semantics:



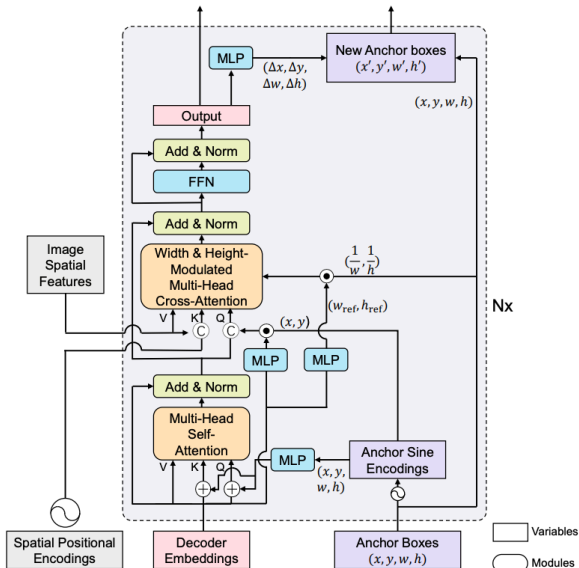
DETR AND ITS EXTENSIONS

- The width and height of the anchor box are used to modulate cross-attention, so attention becomes much better localized, and this helps speed up training



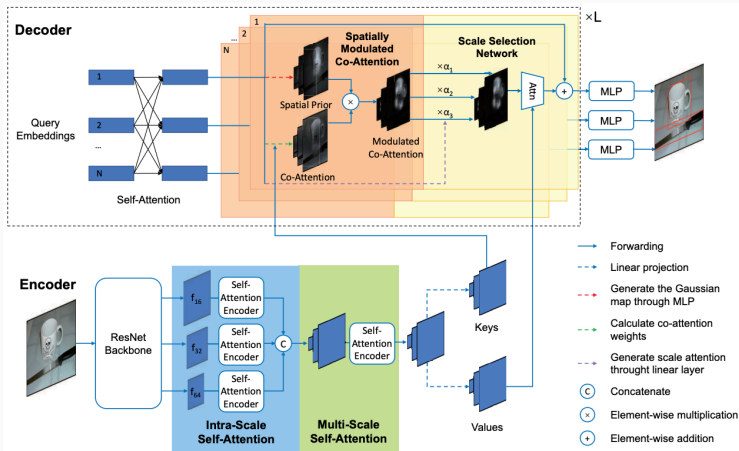
DETR AND ITS EXTENSIONS

- Overview:



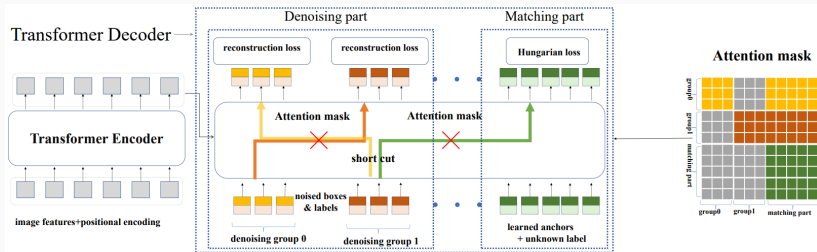
DETR AND ITS EXTENSIONS

- SMCA DETR (Spatially Modulated Co-Attention DETR; Gao et al., 2021) modulates attention weights through spatial priors; what's nearby is more important: spatial weights based on predicted offsets relative to reference points



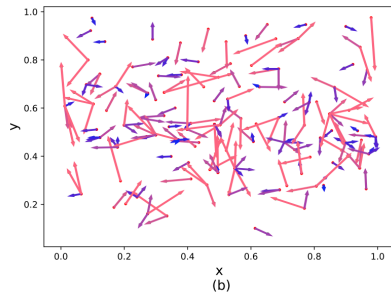
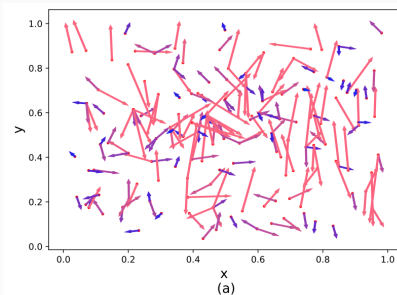
DETR AND ITS EXTENSIONS

- Another problem comes from optimal matching: a small change in weights can lead to a completely different result, predictions are unstable, especially at the beginning of training
- DN-DETR (Li et al., 2022): let's add another training objective — reconstruction of noisy ground truth boxes that are fed with additional queries



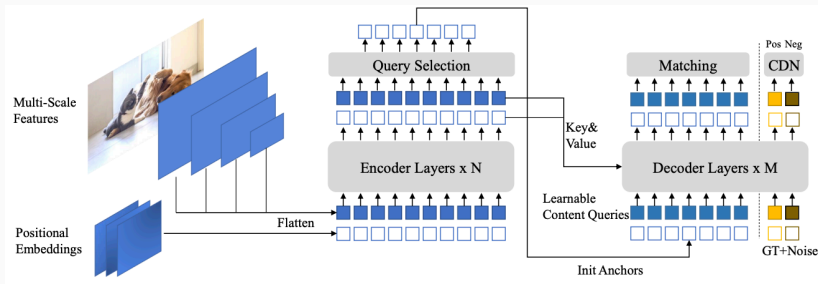
DETR AND ITS EXTENSIONS

- This improves offset prediction: each query has to predict very different offsets due to matching instability, and denoising queries serve as "good anchors" for offset prediction
- As a result, anchors are located closer to their targets (DAB-DETR on the left, DN-DETR on the right):



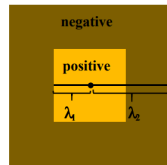
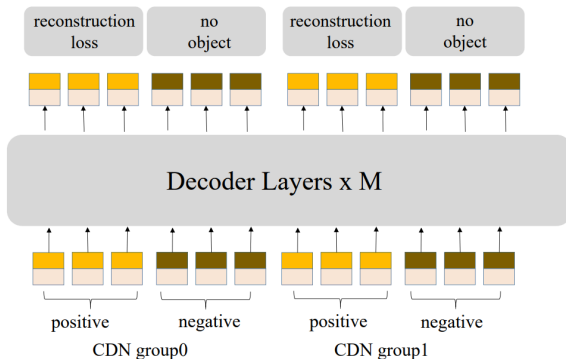
DETR AND ITS EXTENSIONS

- DINO (DETR with Improved DeNoising Anchor Boxes; Zhang et al., 2022): a modification of DETR based on the DN-DETR idea
- The basic structure is more or less the same:



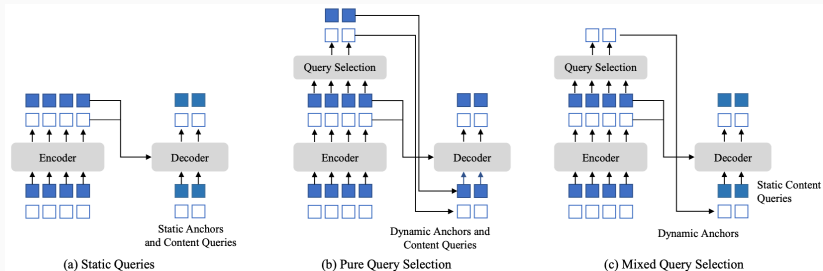
DETR AND ITS EXTENSIONS

- DN-DETR needs good anchors to improve offset learning; but what about bad anchors that have no nearby ground truth objects?
- DINO adds negative examples to mitigate this; part of the inputs are anchors with large noise that should be rejected



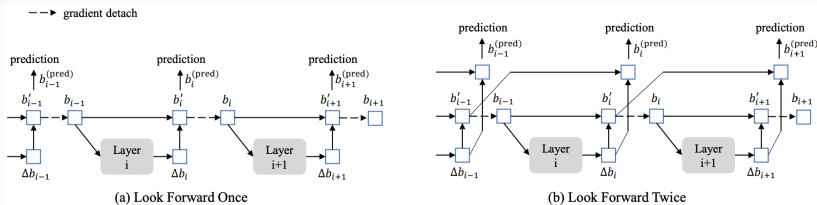
DETR AND ITS EXTENSIONS

- Mixed query selection: positional queries are initialized from encoder proposals, while content queries remain independent



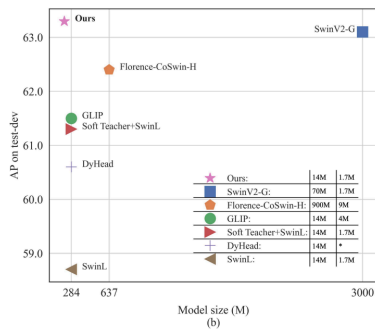
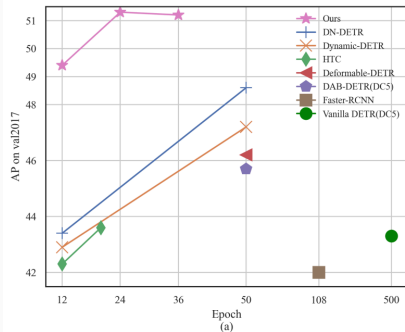
DETR AND ITS EXTENSIONS

- Two-step lookahead: bounding box quality depends on the current anchor (positional query) and the current offset prediction
- Deformable DETR blocked gradient backpropagation to stabilize training: layer i parameters are updated based on the auxiliary loss from boxes b_i
- DINO blocks gradients one more step further, so layer i parameters are updated based on the auxiliary loss from boxes b_i and b_{i+1}



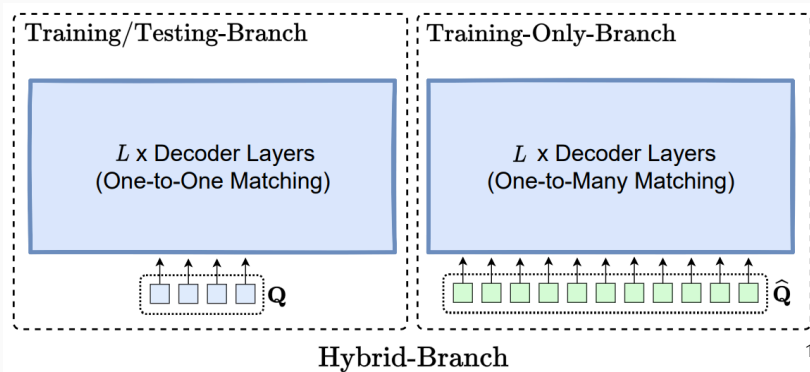
DETR AND ITS EXTENSIONS

- Very good results, fast training (left with ResNet-50 backbone)



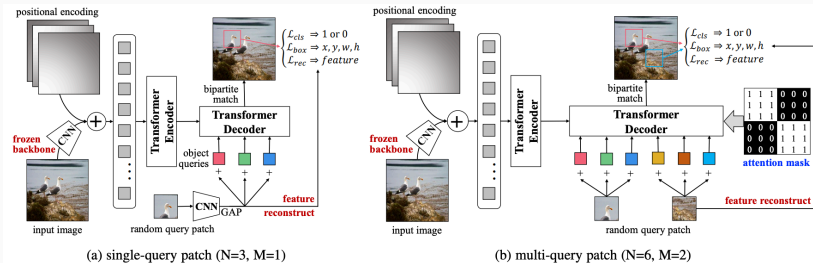
DETR AND ITS EXTENSIONS

- H-Deformable-DETR (Jia et al., 2023) — a very simple and interesting idea
- We have very few positive examples, so let's copy them!
- During training we add additional queries that should predict the same ground truth boxes, with attention masks that separate query groups



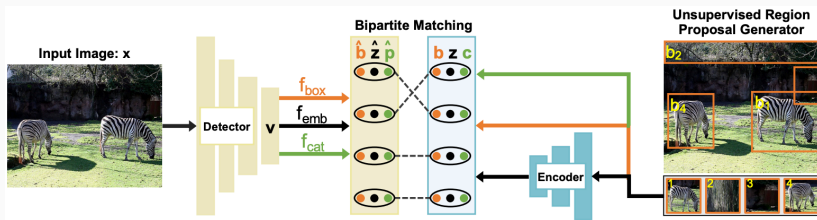
DETR AND ITS EXTENSIONS

- How else can we fix DETR's long training time?
- UP-DETR (Dai et al., 2021) — let's add unsupervised, self-supervised pretraining; we take a random patch and ask DETR to predict its position and reconstruct its features (this is needed because we don't have classification labels, but we need to preserve semantics):



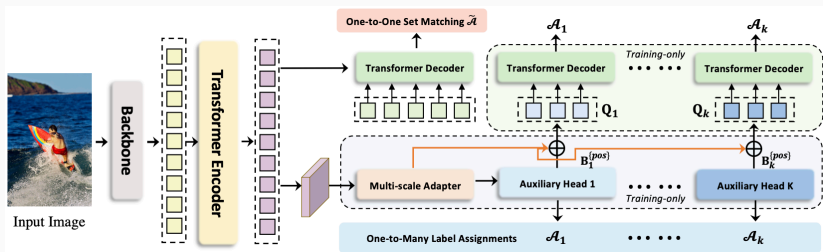
DETR AND ITS EXTENSIONS

- DETReg (Bar et al., 2022) — instead of random crops let's use objects or regions found by other algorithms, possibly even selective search
- We can also predict embeddings for these pseudo-objects



DETR AND ITS EXTENSIONS

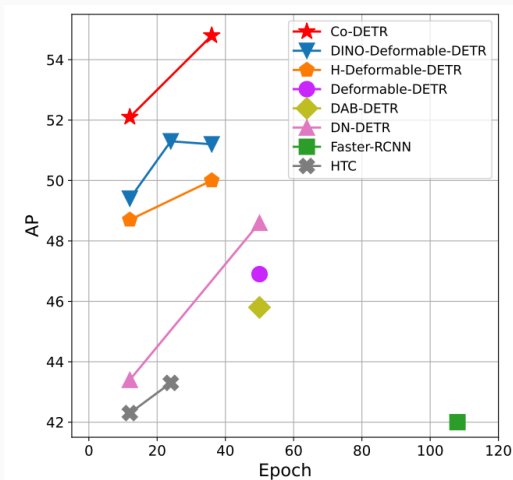
- Co-DETR (Zong et al., 2023) adds several auxiliary heads from other object detection models; new positive examples through matching



Head i	Loss $\mathcal{L}_i$	Assignment $\mathcal{A}_i$		
		$\{pos\}, \{neg\}$ Generation	$\mathbf{P}_i$ Generation	$\mathbf{B}_i^{\{pos\}}$ Generation
Faster-RCNN [27]	cls: CE loss, reg: GloU loss	$\{pos\}$: IoU(proposal, gt)>0.5 $\{neg\}$: IoU(proposal, gt)<0.5	$\{pos\}$: gt labels, offset(proposal, gt) $\{neg\}$: gt labels	positive proposals (x_1, y_1, x_2, y_2)
ATSS [40]	cls: Focal loss reg: GloU, BCE loss	$\{pos\}$: IoU(anchor, gt)>(mean+std) $\{neg\}$: IoU(anchor, gt)<(mean+std)	$\{pos\}$: gt labels, offset(anchor, gt), centerness $\{neg\}$: gt labels	positive anchors (x_1, y_1, x_2, y_2)
RetinaNet [21]	cls: Focal loss reg: GloU Loss	$\{pos\}$: IoU(anchor, gt)>0.5 $\{neg\}$: IoU(anchor, gt)<0.4	$\{pos\}$: gt labels, offset(anchor, gt) $\{neg\}$: gt labels	positive anchors (x_1, y_1, x_2, y_2)
FCOS [31]	cls: Focal Loss reg: GloU, BCE loss	$\{pos\}$: points inside gt center area $\{neg\}$: points outside gt center area	$\{pos\}$: gt labels, ltrb distance, centerness $\{neg\}$: gt labels	FCOS point (cx, cy) $w = h = 8 \times 2^{2-\frac{1}{4}}$

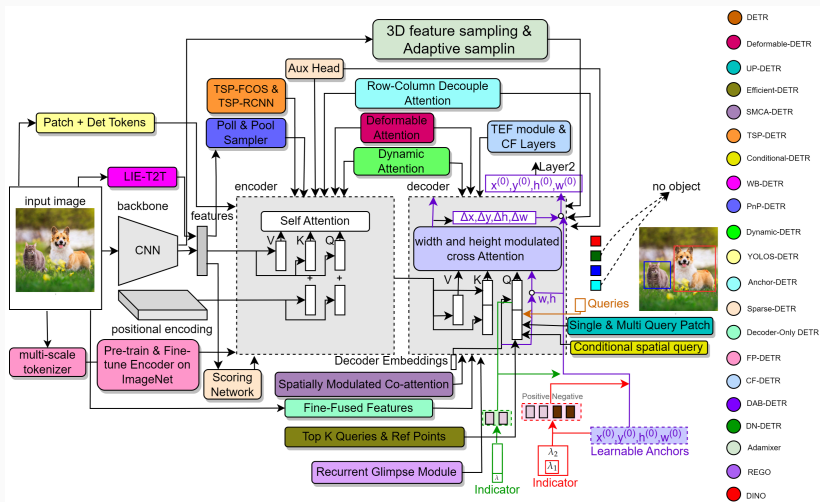
DETR AND ITS EXTENSIONS

- Graph for ResNet-50 backbone:



DETR AND ITS EXTENSIONS

- Overview from an (old) survey (Shehzadi et al., 2023):



THANK YOU!

Thank you for your attention!

