DETR: End-to-End Object Detection with Transformers

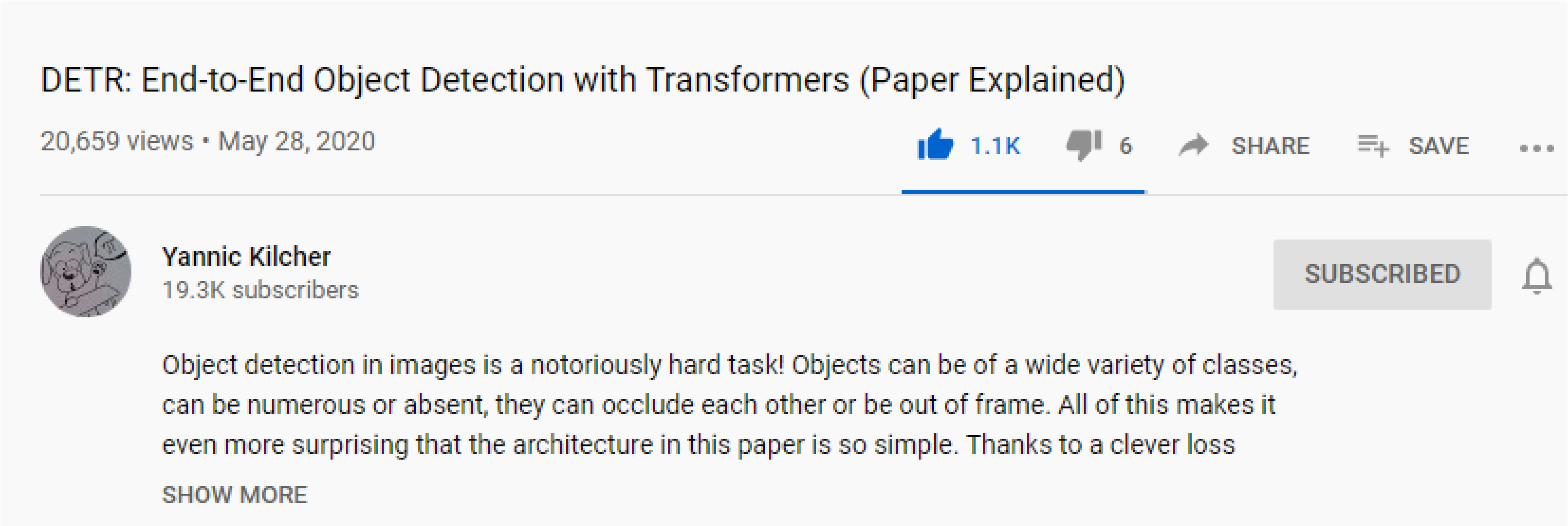
Li Shaohua

08/06/2020

DETR is hot

- Github:

- Youtube:

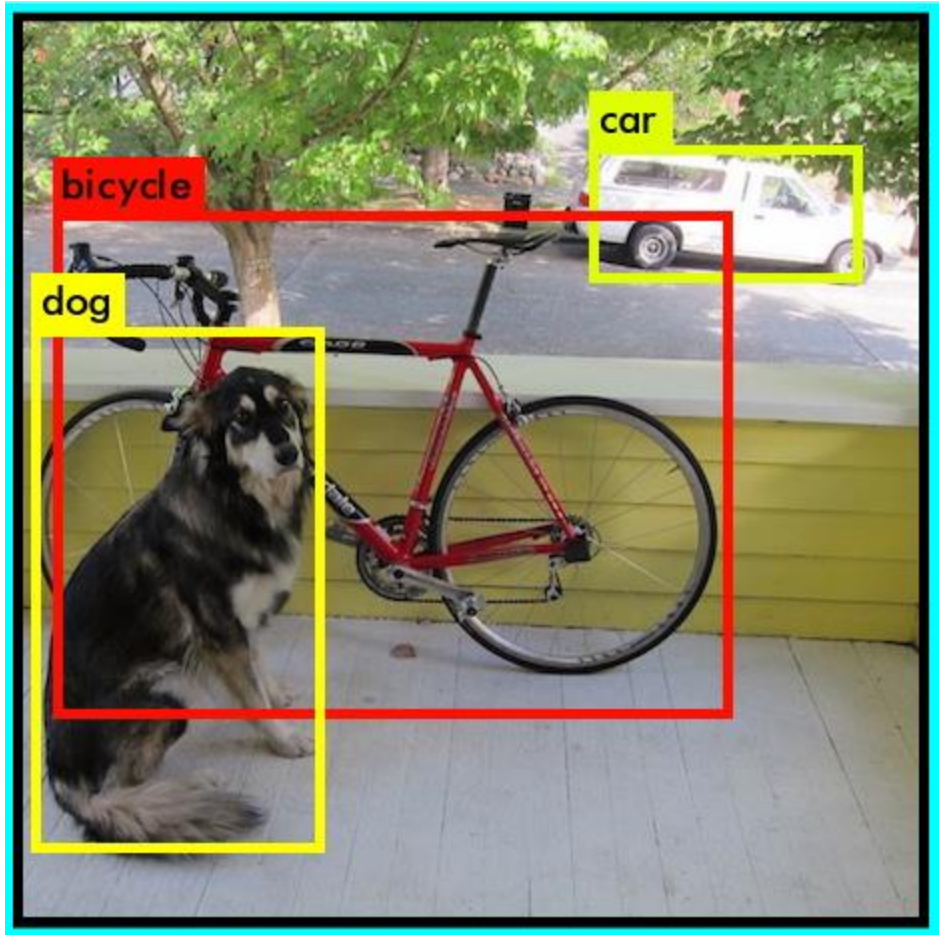


Outline

- Background
  - What is object detection
  - Problems with the traditional object detection paradigm
- DETR
  - What is transformer and self-attention
  - Intuitions
  - DETR architecture
  - Results & Visualizations
- Gained insights and possible future work
- References

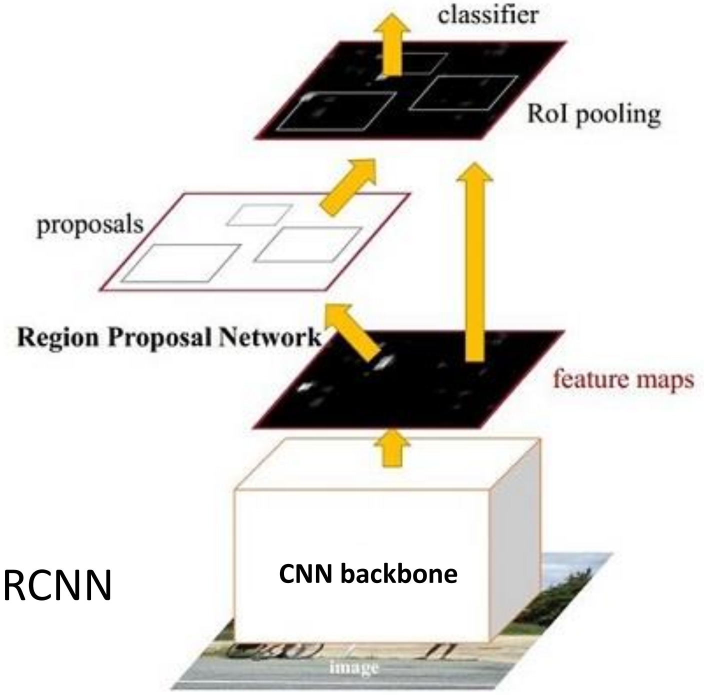
Object Detection

- Localization & classification at the same time
- Detection is an "entry-level" task for many tasks involving localization
  - Methods transfer to other tasks easily
- Challenges
  - Many candidate locations
  - Varying scales
  - Occlusions
    - object features "pollute" each other
  - Limited annotations
    - COCO: 300K images, 80 categories
    - ImageNet: 1.5M images, 1000 categories
    - Bounding boxes are laborious



Traditional object detection paradigm

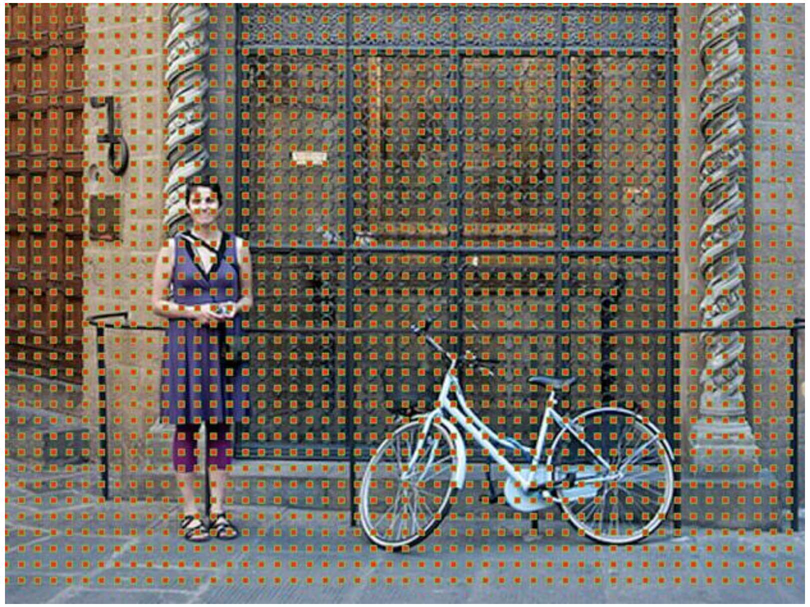
- A CNN backbone extracts feature maps from the input image
- A Region Proposal Network (RPN) enumerates all windows on the feature maps, outputs N candidate boxes
- A classifier classifies each box into a class



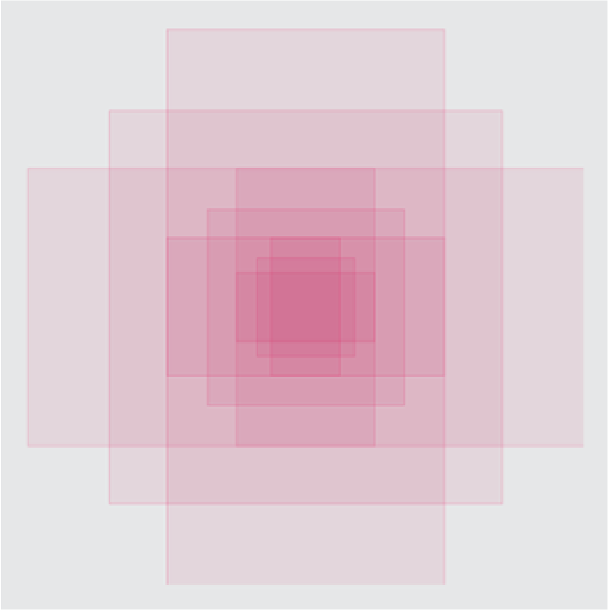
Architecture of Faster-RCNN

Problem 1: Enumerate candidate boxes in RPN

- Enumerate all pixels on the feature maps
- At each pixel, enumerate all predefined boxes
- Most candidate boxes are bad. Inefficient, slow



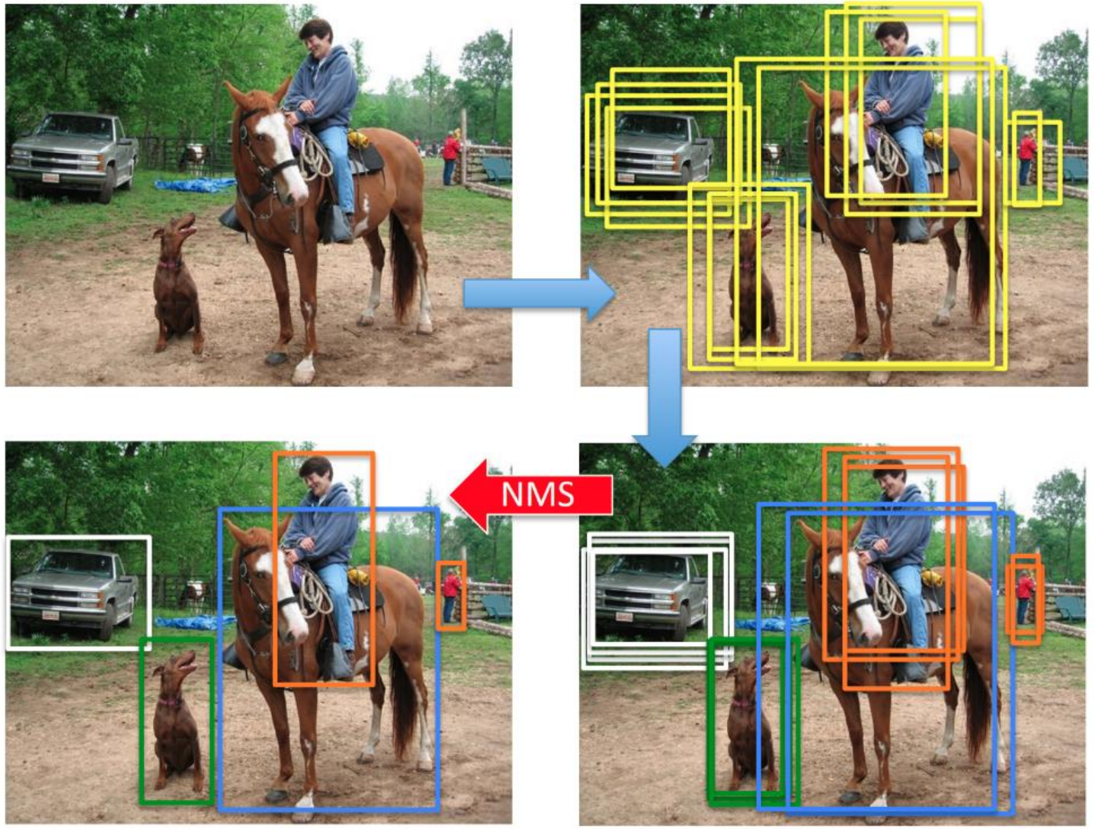
Enumerate all pixels (in feature maps)



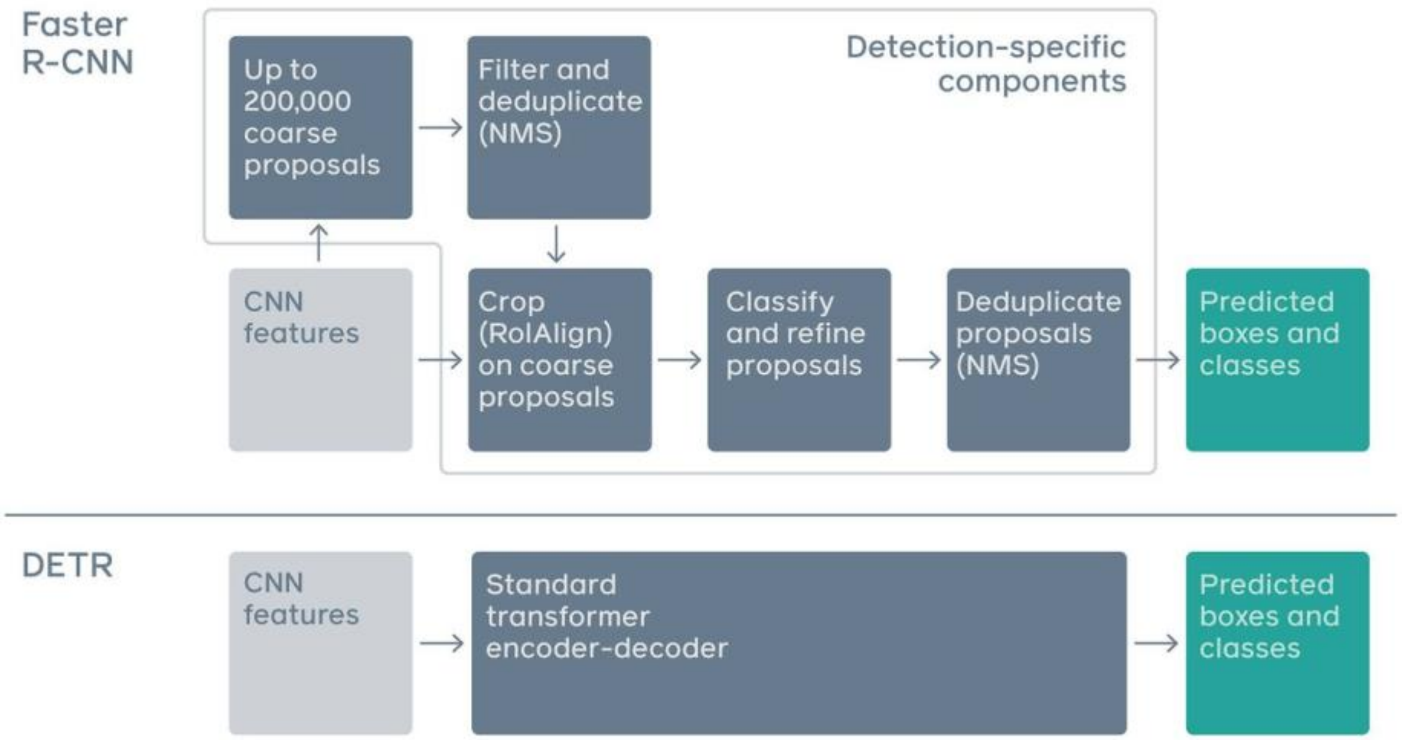
Predefined boxes

Problem 2: Redundant boxes and NMS

- RPN outputs many redundant boxes
- Non-maximum suppression (NMS) merges/removes redundant boxes
- These hand-designed components have a few hyperparameters
- Model tuning is complex



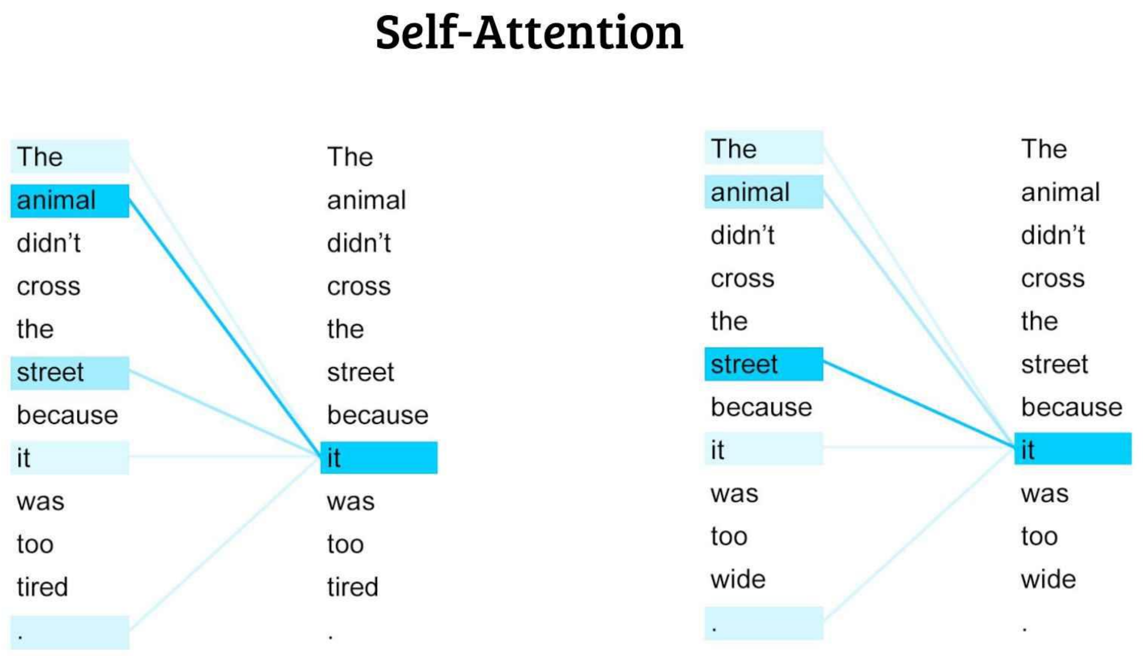
Simplicity of DETR



Self-attention: core of Transformer

1. *“The* *animal* *didn’t cross the street because* *it* *was too tired”*

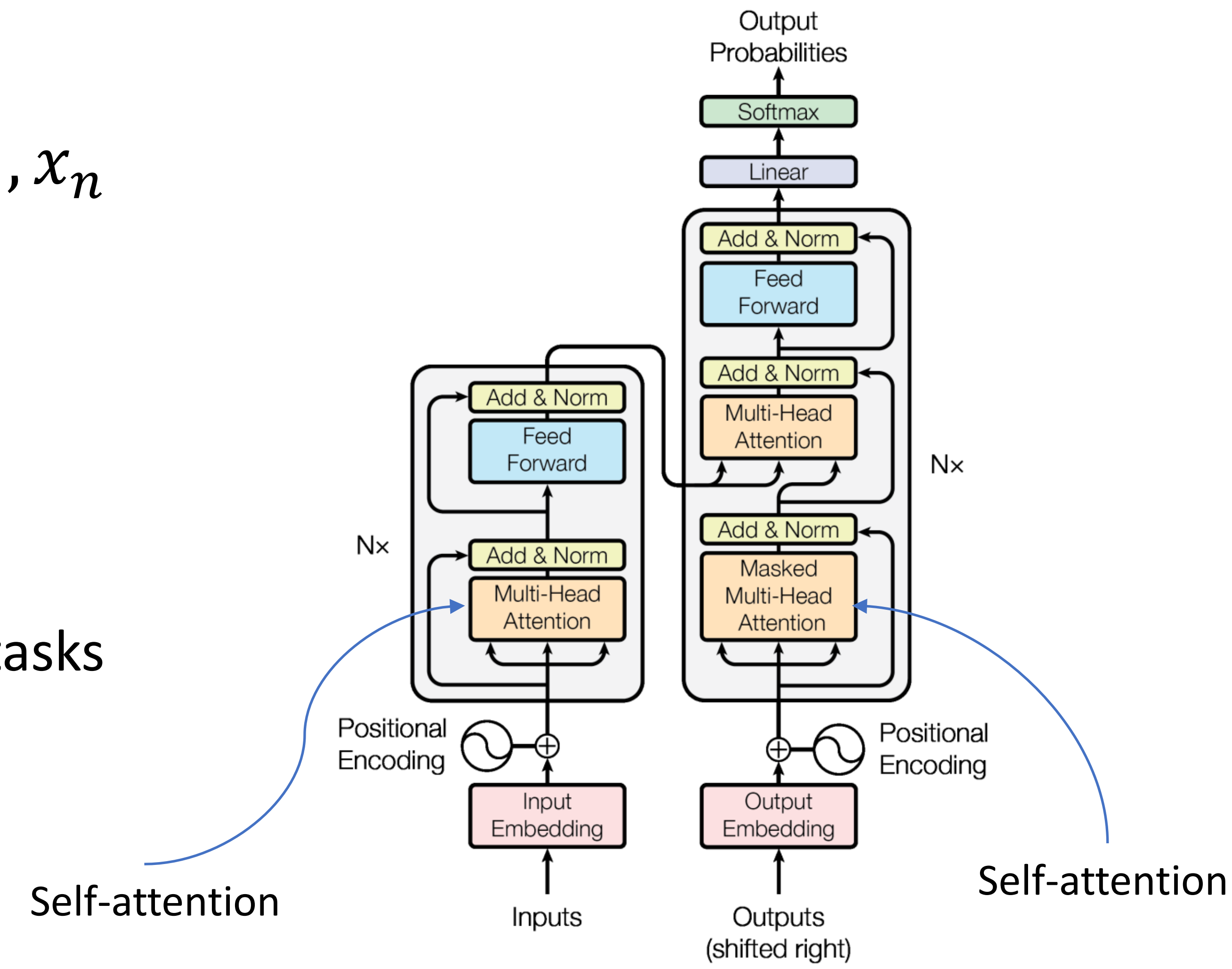
2. *“The animal didn’t cross the* *street* *because* *it* *was too wide”*



- All tokens can attend to each other equally well, no matter how far they are from each other
- Transformer captures long-range semantic dependencies
  - Creates global context into the output embeddings

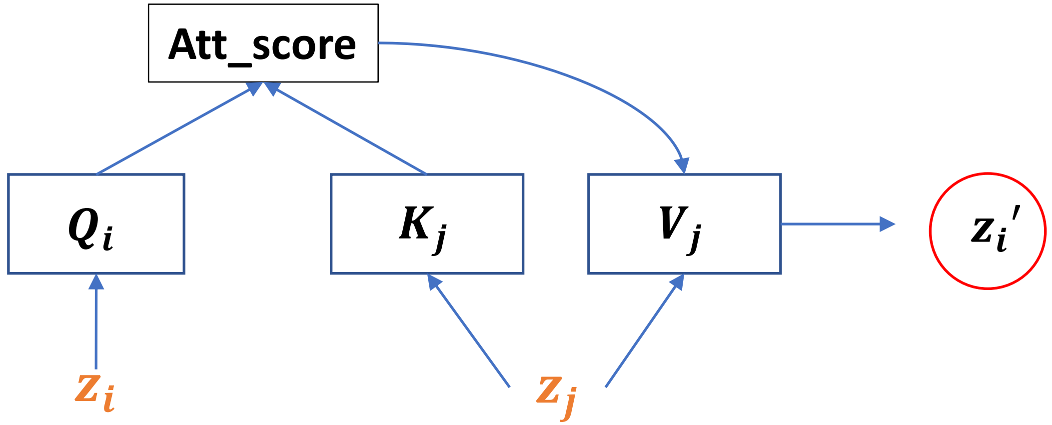
Transformer

- Take a sequence of tokens $x_1, \cdots, x_n$ as input
- Convert to intermediate embeddings $z_1, \cdots, z_n$
  - $z_1, \cdots, z_n$ have combined the semantics (embeddings) among $x_1, \cdots, x_n$
  - $z_1, \cdots, z_n$ can be used for various tasks
- Semantics = Embeddings
  - Manipulating semantics = Manipulating embeddings



Implementation of Self-attention

- $z_1, \cdots, z_n \rightarrow \{Q, K, V\}_{1,\cdots n}$
  - Mapped to Query, Key, Value subspaces



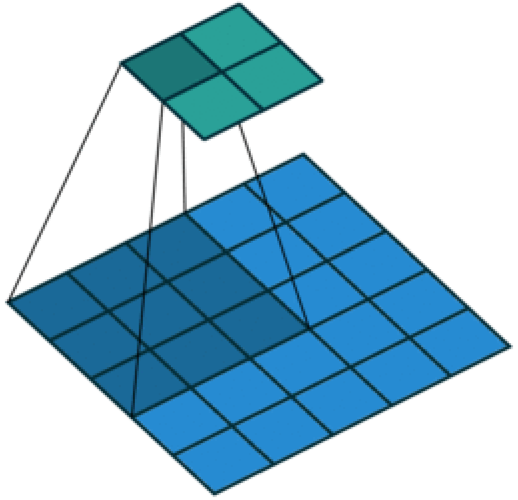
- Attention score between z_i and z_j:

$$\text{Att\_score}(z_i, z_j) = Q_i \cdot K^T$$

- z_i is transformed by self-attention:

$$z_i' = \text{Attention}(Q_i, K, V) = \sum_j \text{softmax}\left(\frac{Q_i K_j^T}{\sqrt{d}}\right) V_j$$

Self-attention vs. Convolution in Vision



- Each pixel is an input "token"
- Traditionally, convolution is used to integrate pixels
  - Recognize patterns within a small window of pixels
  - Difficult to integrate non-local pixels
  - Have to make network very deep to "see the big picture"
  - Detection & Segmentation 
- Self-attention (transformer) also integrates multiple pixels
  - Works when the correlated pixels are non-local
  - Trunk, tail, legs of an elephant to a whole elephant
  - Detection & Segmentation 

Intuition of Self-attention for Detection

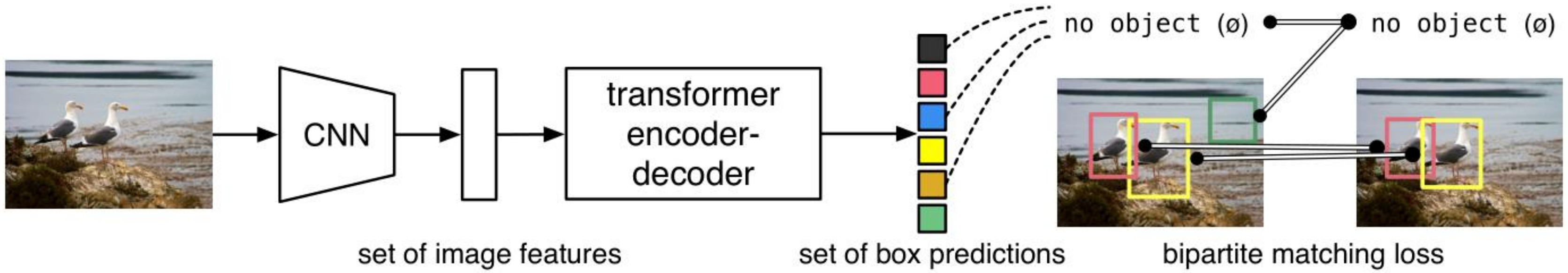
- First layer

Second Layer

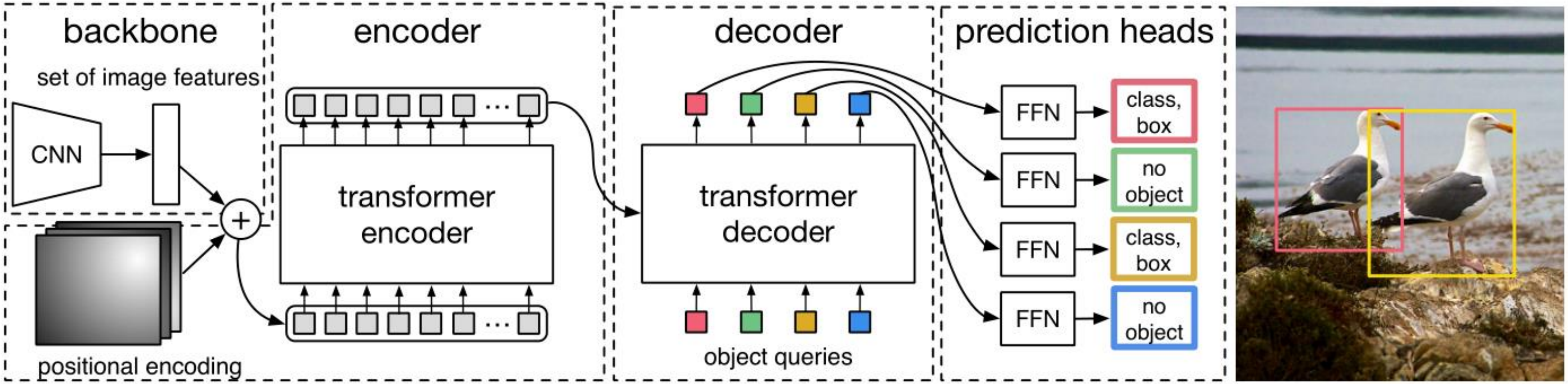
. . . . . .

- ●, ●: initial queried positions
- Gradually attend to relevant parts of the same object

DETR architecture



DETR Pipeline

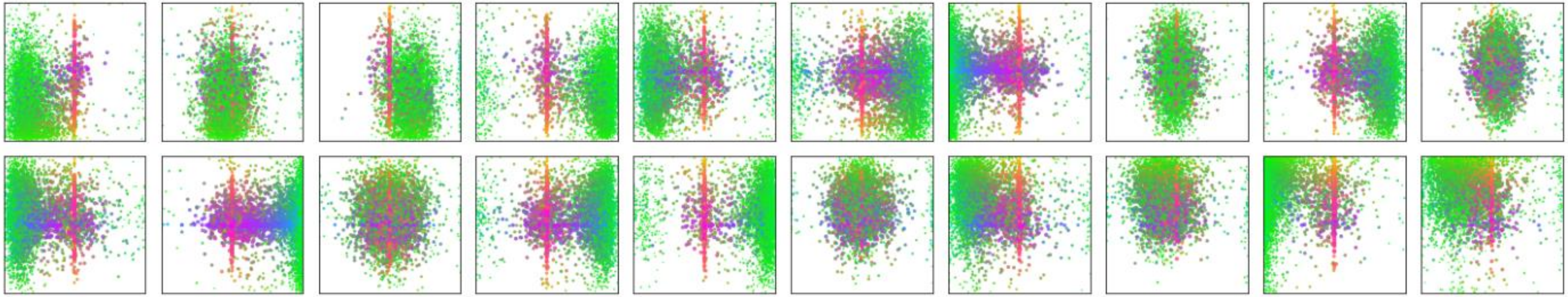


DETR Architecture

Positional encoding

- 16*16 feature maps are flattened to a 256-token sequence
  - Each pixel is a visual feature vector $v_{x,y}$ at (x, y), 2048-dimensional
  - $v_{x,y}$ has no information about coordinates (x, y)
- How to allow transformer to utilize location information?
  - Add a positional embedding to each pixel's embedding
  - $(x, y) \rightarrow p_{x,y}$, also 2048-dimensional vector
  - Transformer input: $v_{x,y} + p_{x,y}$ (combine visual and positional features)
- Can we concatenate $(v_{x,y}, p_{x,y})$?
  - Number of Parameters $\times$ 2 $\rightarrow$ overfitting
- How DETR disentangles visual and positional features?
  - We have no idea…
  - Model learns to use it in whatever ways, as long as the loss is minimized

Visualization of object queries



All box positions in COCO images predicted by each object query

- Object queries:
  - N vectors, each 2048-dimensional vector (similar to positional encoding)
  - Randomly initialized
  - Trained along with transformer parameters
- Different queries focus on different areas of the image
  - Left or middle or right…

Avoid redundant detections

- Hungarian matching loss
  - Find the best matching pairs
  - Redundant predictions match to ϕ, incur loss
- Minimize Hungarian loss
  ➡ Decoder learns to avoid redundant detections
  - No need for NMS

Predictions

Apple,	(50, 30, 120, 90)
Dog,	(150, 200, 550, 280)
Person,	(320, 140, 400, 640)
Person,	(315, 140, 360, 620)

Groundtruth

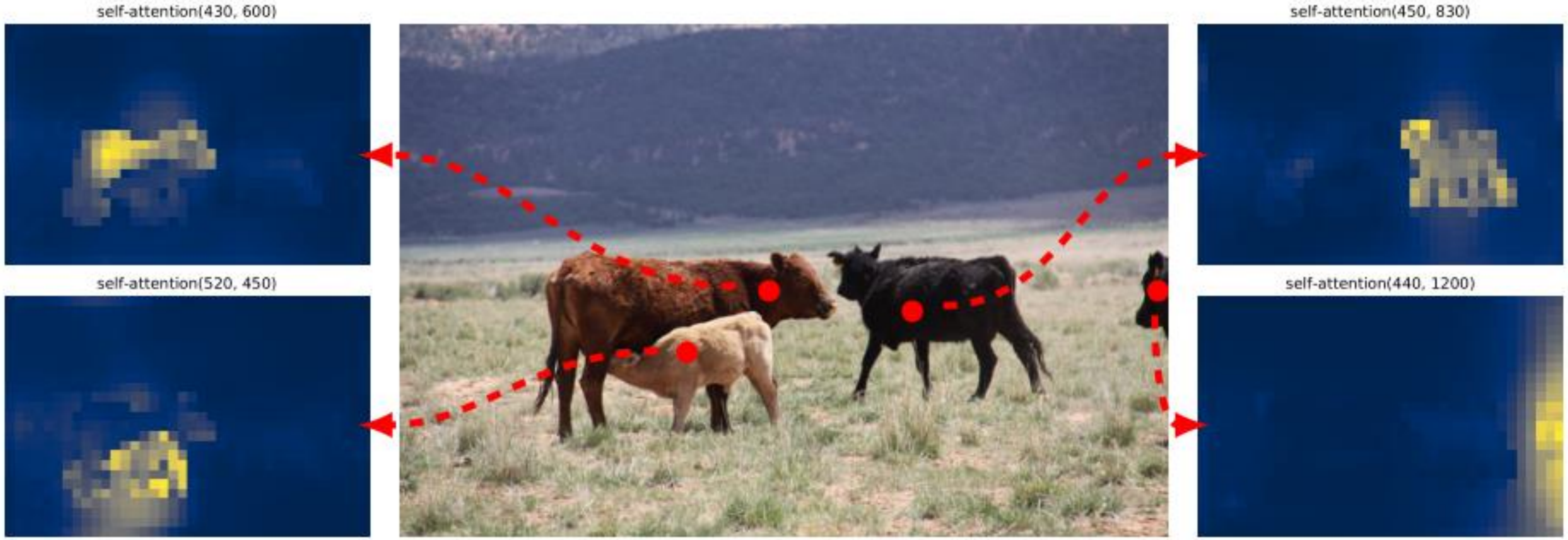
Dog,	(145, 210, 540, 290)
Person,	(325, 145, 398, 633)
Pear,	(52, 33, 117, 88)
ϕ,	(-1, -1, -1, -1)

Performance compared with Faster-RCNN

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**

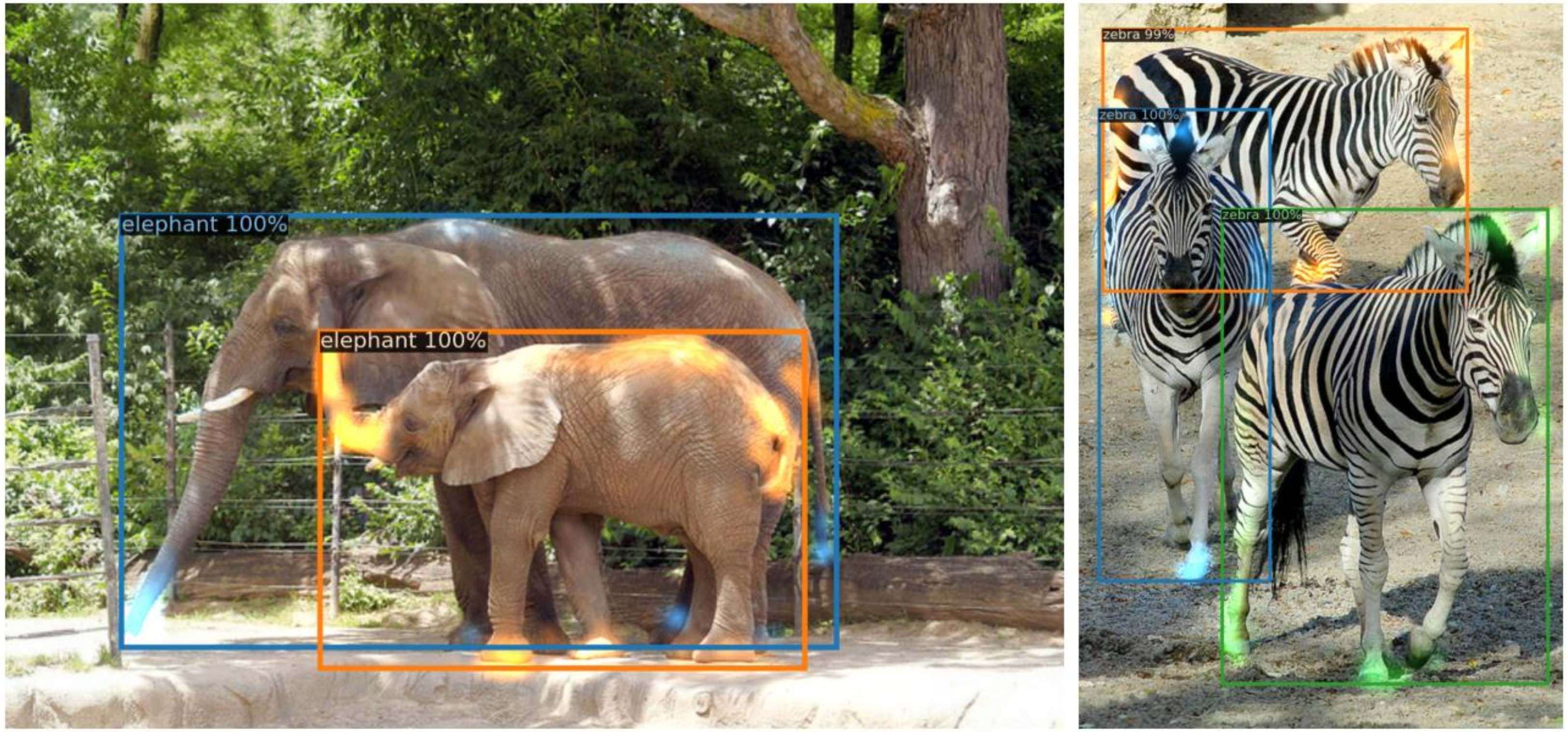
- Still performs worse than Faster-RCNN on small objects
- Because DETR only uses high-level feature maps
  - Low spatial resolution, cannot see small objects clearly
  - Faster-RCNN uses FPN to combine high- and low-level feature maps

Visualization of encoder self-attention



Visualization of decoder self-attention

- Decoder attends to object extremities, such as legs and heads
- Parts of different objects are separately attended (color of highlighted areas agree with identity)

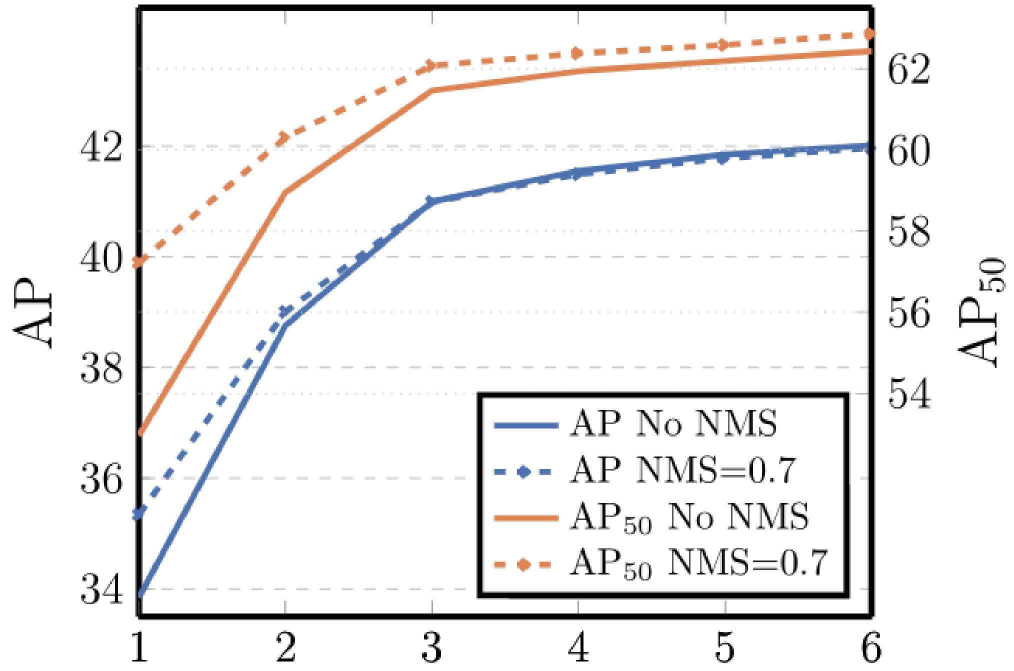


Impact of Encoder and Decoder

Table 2: Effect of encoder size. Each row corresponds to a model with varied number of encoder layers and fixed number of decoder layers. Performance gradually improves with more 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 is much more important than encoder!
- Decoder has implicit "anchors", essential for detection
- Encoder only helps aggregate pixels of the same object
    - Reduces the burden of decoder



Number of decoder layers

DETR for segmentation

1. Detect boxes first
2. Segment each box
3. Vote for the class of each pixel

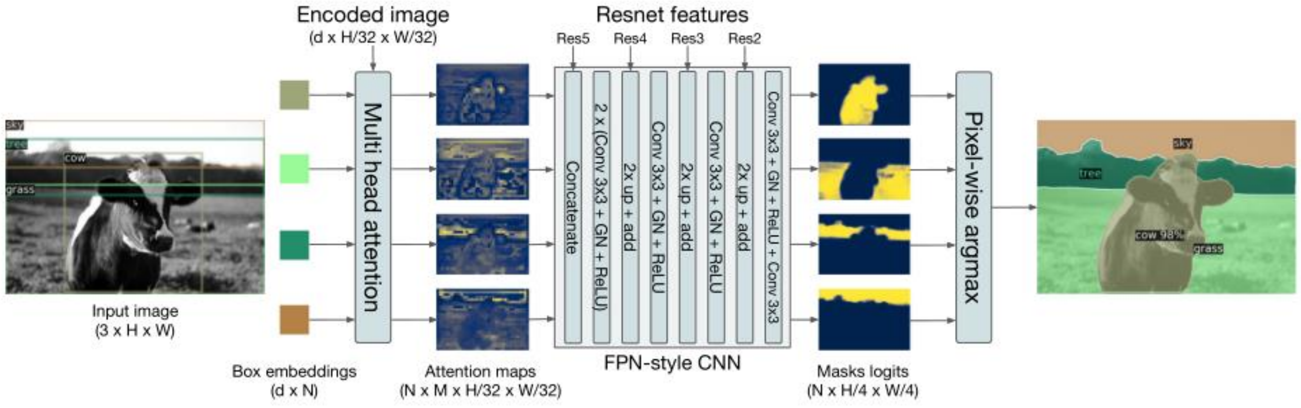


Fig. 8: Illustration of the panoptic head. A binary mask is generated in parallel for each detected object, then the masks are merged using pixel-wise argmax.

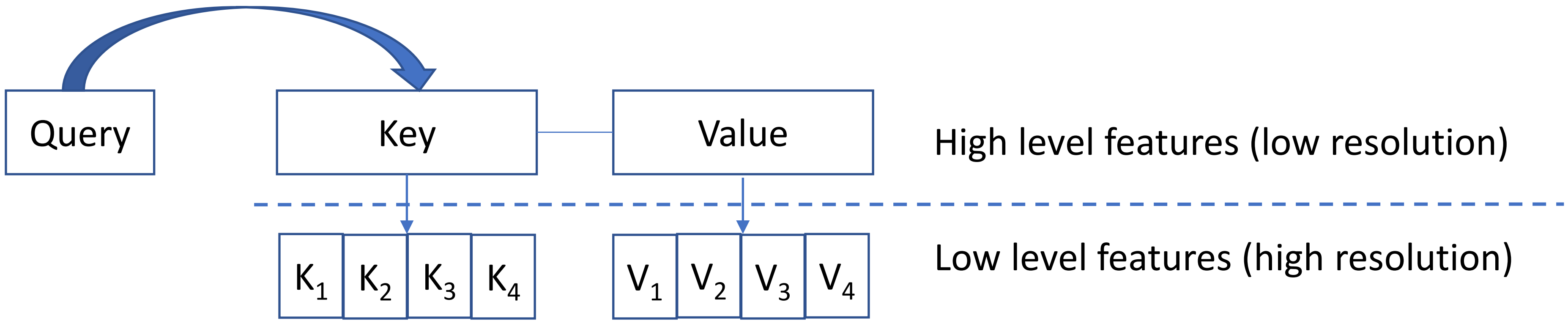
Fig. 9: Qualitative results for panoptic segmentation generated by DETR-R101. DETR produces aligned mask predictions in a unified manner for things and stuff.

Insights gained

- Transformers are an effective new paradigm for computer vision
  - Esp. for tasks that require integrating information from non-local pixels
  - Pure data-driven, replace hand-designed components and hard-coded procedures
  - Easily integrate more constraints (by adding new loss functions)
- RAM consumption
  - Training: 16 V100 * 3 days (a bit high)
    - Supervision signal is indirect → Slow training
    - How to speed up training?
  - Inference: Similar to Faster-RCNN
    - Model size (R101): 232MB

Easy extensions to DETR

- Improve DETR for segmentation
  - End-to-end segmentation, without training detector first?
- Improve performance on small objects: hierarchical transformers
  - Integrate different granularities of features
  - Keep computation and RAM low



Challenging future work

- Apply transformers to video analysis
  - Capture temporal correlations across video frames?
  - Computation and RAM consumption is huge
  - Efficient representation is the key. Detection first?
- DETR is still supervised
  - How to design self-supervised learning, like in NLP?
  - Exploit correlations/consistency across video frames?

References for further study

- Attention is All you Need. NIPS 2017
- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019
- LXMERT: Learning Cross-Modality Encoder Representations from Transformers. EMNLP 2019
- DETR: End-to-End Object Detection with Transformers. ECCV 2020
- https://www.youtube.com/watch?v=TQQlZhbC5ps (tutorial on transformers)
- https://www.youtube.com/watch?v=T35ba_VXkMY (tutorial on DETR)