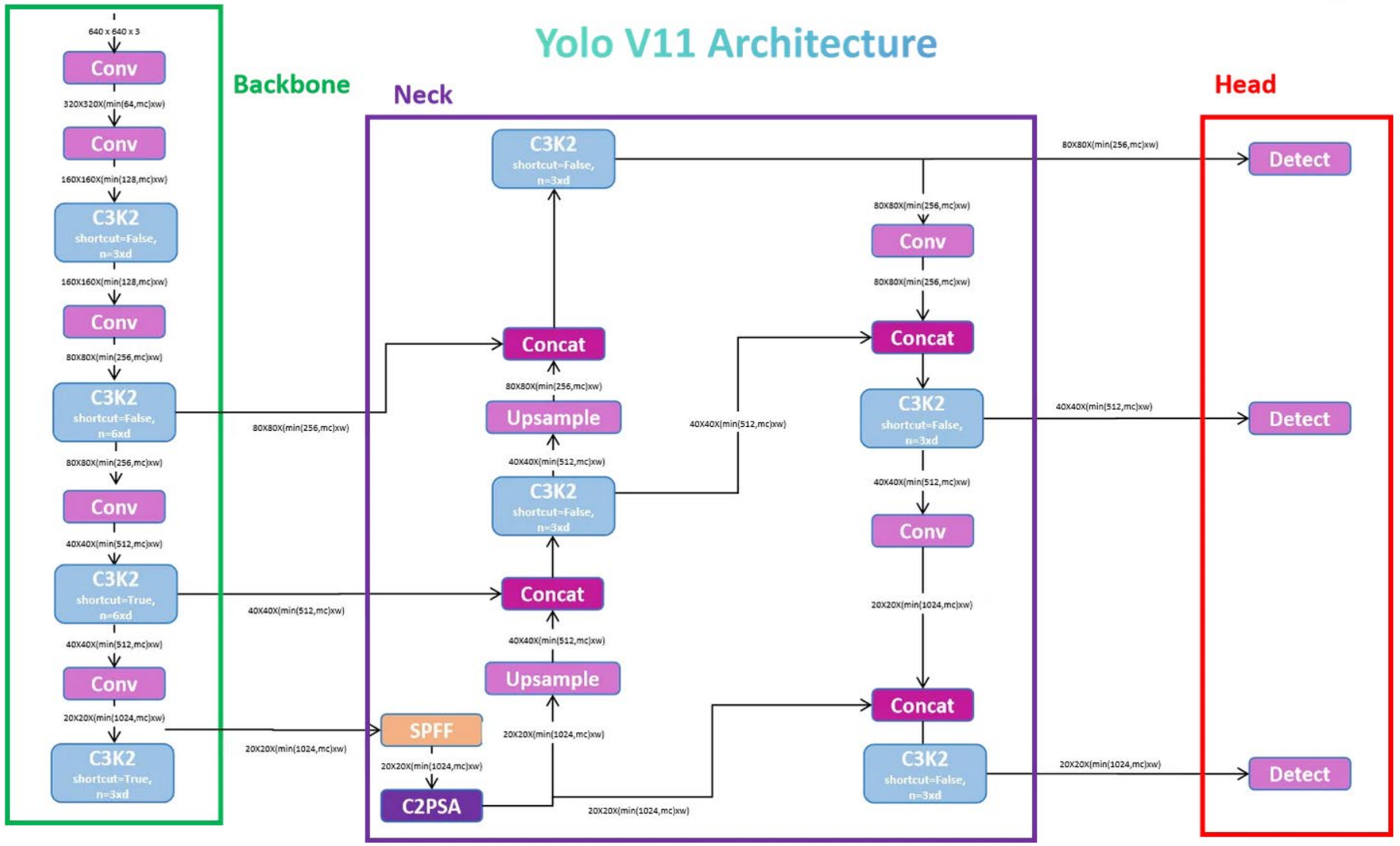
Machine Learning for Robotics: **Object Detection with Transformers**

Prof. Navid Dadkhah Tehrani

CNN based object detection such as YOLO have very complicated models. YOLO v11, for example has many handcrafted components in the architecture:



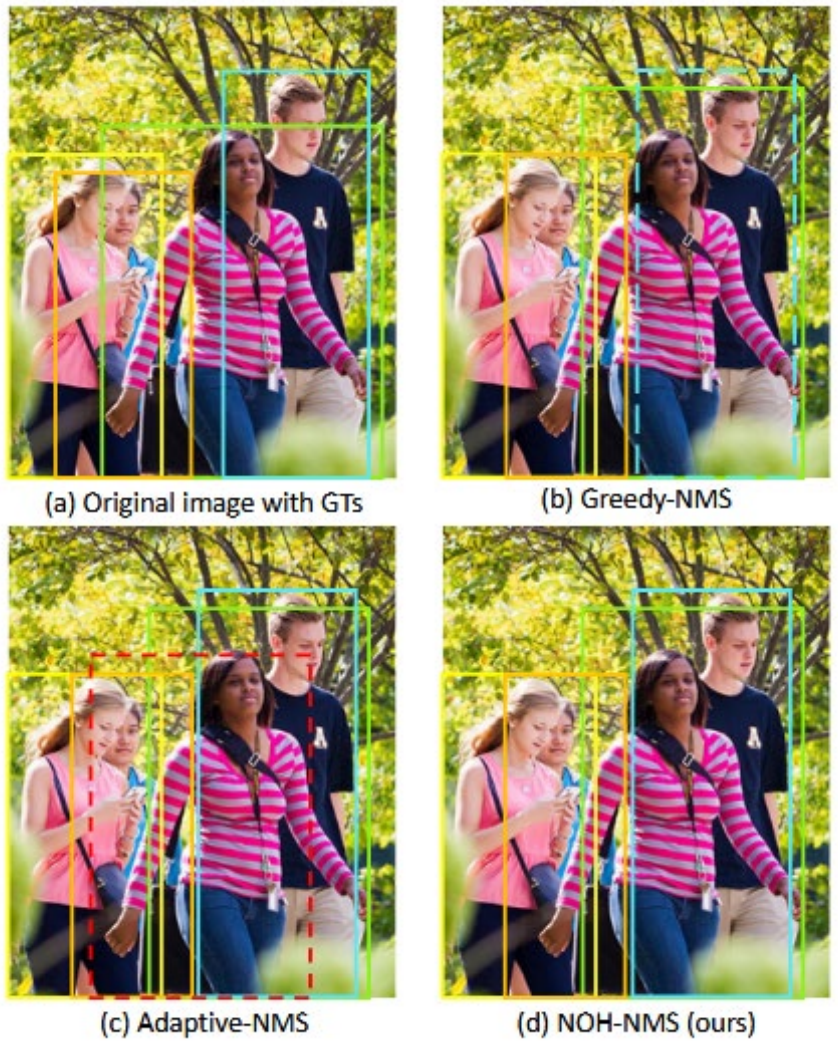
Ref: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71

- In this lecture we are going to look at object detection with transformers via DETR [1] architecture.
- In DETR, we don't have any handcraft component in the model, it's just a transformer model that we feed images and get bounding box and class predictions. This is a much simpler and cleaner architeture.
- In DETR architecture, the transformer does all the hard work via the attention mechanism.
- Although the DETR does not outperform the recent version of YOLO yet, the state of the art is moving toward transformer models.
- After the DETR paper came out in 2020, there has been many modifications to it that achieve state of the art performance in object detection [2].

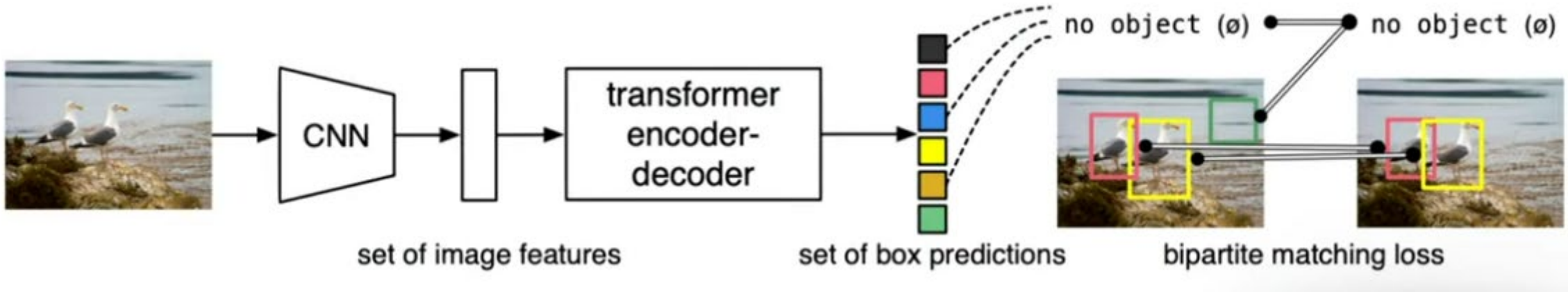
[1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, End-to-End Object Detection with Transformers, 2020
[2] Z. Zong, G. Song, Y. Liu , DETRs with Collaborative Hybrid Assignments Training, 2023

In YOLO there are many cells and each cell is predicting a bbox. i.e. the same object is predicted many times via different cells. Then the extra bboxes are removed via a process called
NMS (none-maximum supression)

If the image is crowded of objects, the NMS could fail to detect some objects.



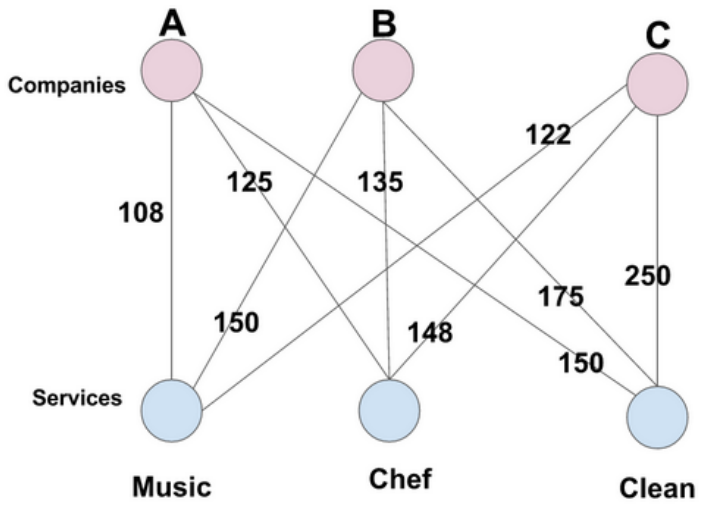
[1] X. Sun, X. Guo, F. Huang, NOH-NMS: Improving Pedestrian Detection by Nearby Objects Hallucination, 2021


CNN
transformer
encoder-
decoder
no object (ø)
no object (ø)
set of image features
set of box predictions
bipartite matching loss

Hungarian Matching Algorithm

Assume we're organizing a party and want to select musician, chef and cleaner from 3 different company. but each company can only do one job at a time. How can we select the minimum cost solution?

Company	Cost for Musician	Cost for Chef	Cost for Cleaners
Company A	$108	$125	$150
Company B	$150	$135	$175
Company C	$122	$148	$250



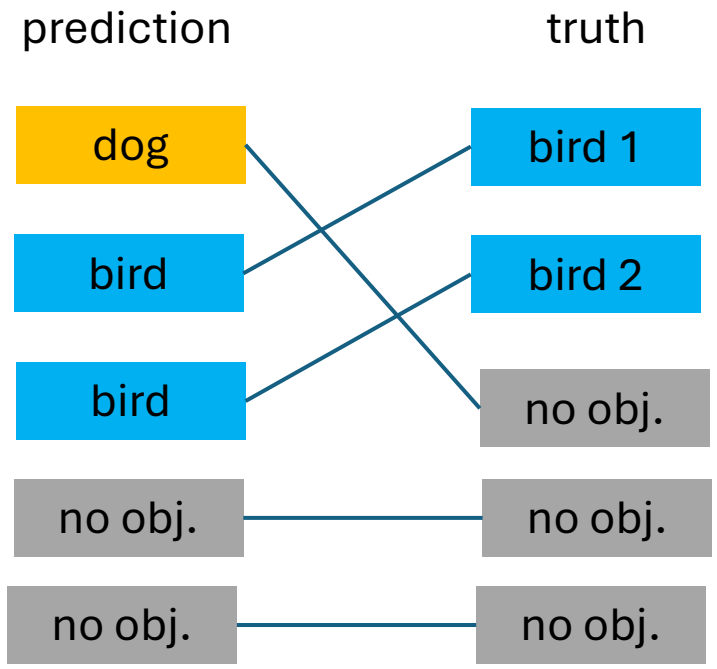
Ref: https://brilliant.org/wiki/hungarian-matching/

Bipartite matching loss

- We first use Hungarian matching to find optimal matches between predicted and ground truth (left and right nodes).
- The loss has 3 parts:
  - Classification loss
  - Bbox L1 loss
  - Bbox GIoU loss

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

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

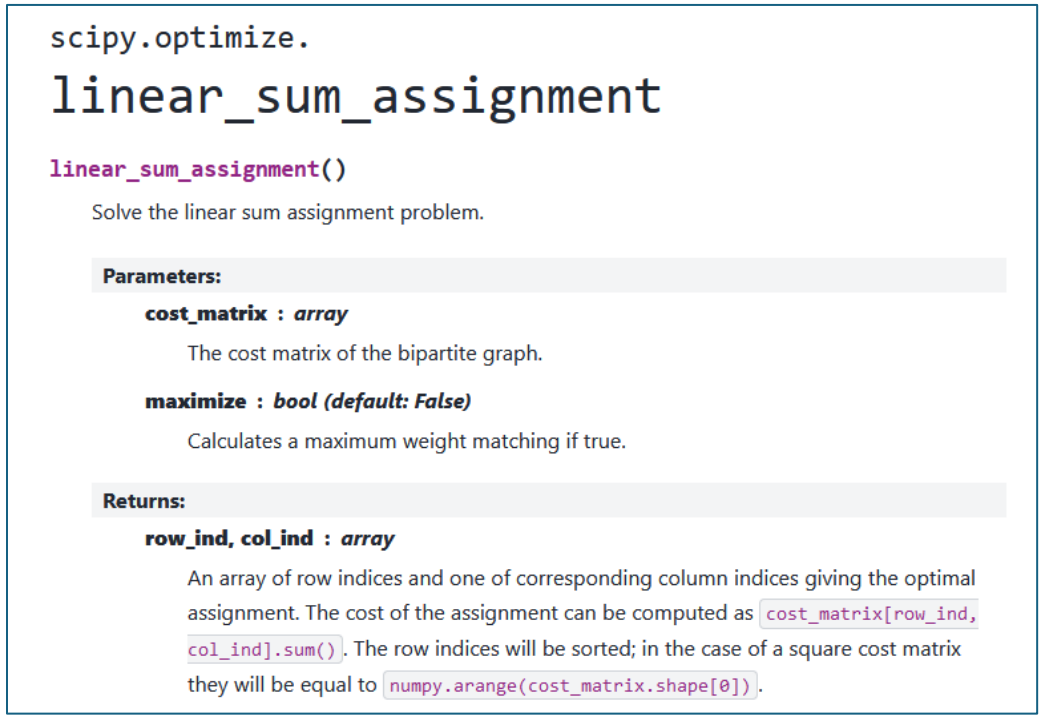


- One of the difference here compared to yolo is that we have a “no-object” class as part of classification.

- GOIU is a differentiable version of IOU.
- Hungarian matching is only used during training.

- In addition in DETR, the prediction is done in single stage w/o NMS.

- The DETR implementation, the matching was done via the following function from SciPy:



scipy.optimize.

linear_sum_assignment

linear_sum_assignment()

Solve the linear sum assignment problem.

Parameters:

cost_matrix : *array*

The cost matrix of the bipartite graph.

maximize : *bool (default: False)*

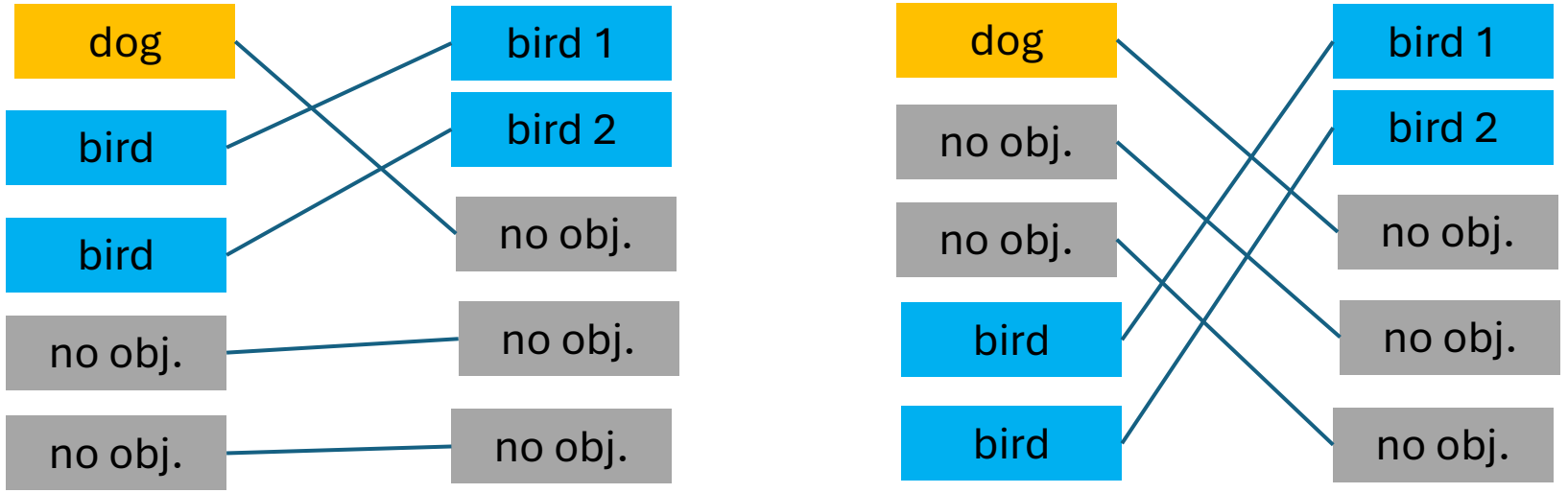
Calculates a maximum weight matching if true.

Returns:

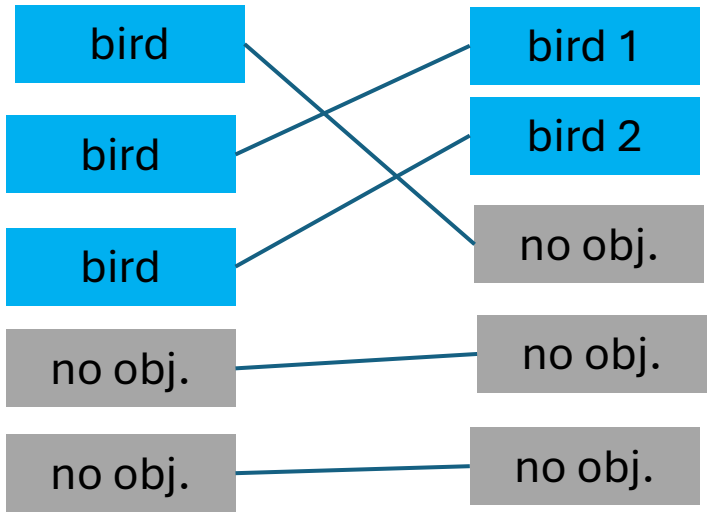
row_ind, col_ind : *array*

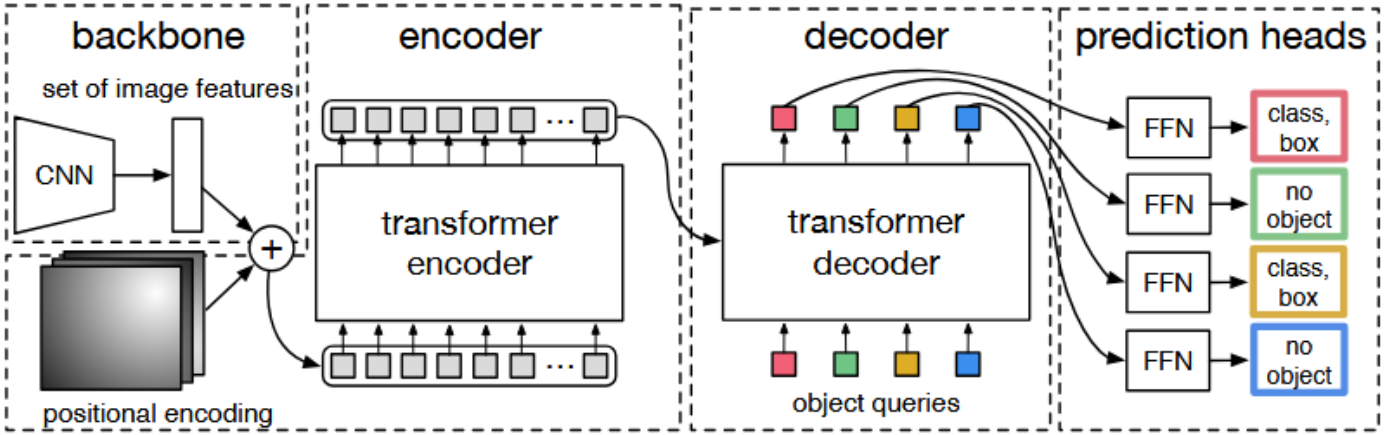
An array of row indices and one of corresponding column indices giving the optimal assignment. The cost of the assignment can be computed as `cost_matrix[row_ind, col_ind].sum()`. The row indices will be sorted; in the case of a square cost matrix they will be equal to `numpy.arange(cost_matrix.shape[0])`.

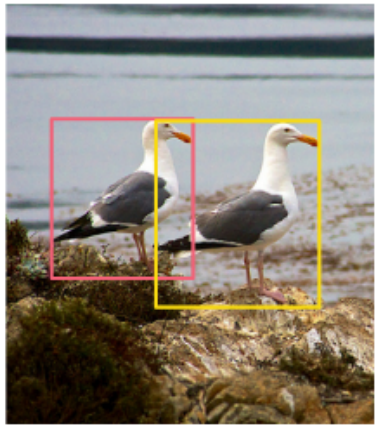
The loss does not care about the order of the prediction. Because the Hungarian matching still matches
The left nodes to the right nodes even if the order is different.

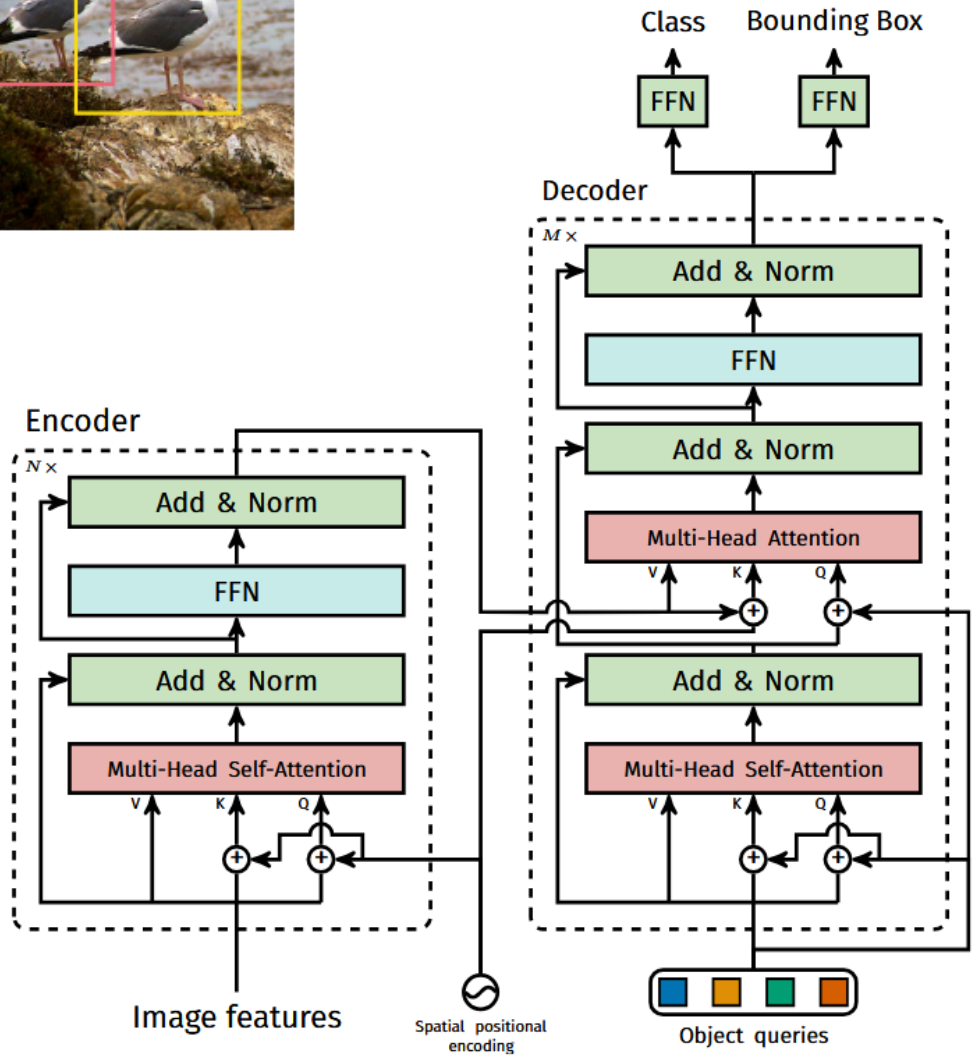


The loss penalizes if a wrong object is predicted: by Hungarian matching assigning of wrong object to the no-object.
The loss penalizes is duplicate object predicted: by Hungarian matching assigning of duplicate object to the no-object.









- Each object query is initialized as a random vector (of size say 256).
- Assuming the model is trained to detect 100 objects in the image, then there are 100 query vectors .
- The object queries, cross-attend to the encoder's key and value.
- Assuming the there are 80 object classes,
  Then there are two outputs after the feedforward network (FFN):
  - B*100*81 → 80+1 for all the class probs+"noobject"
  - B*100*4→ for the 4 coordinate of the bboxes.