# CV / VLMs

## Unit 5: State-of-the-Art Object Detection Techniques

# 5.2.3

# Anchor-Free Object Detection

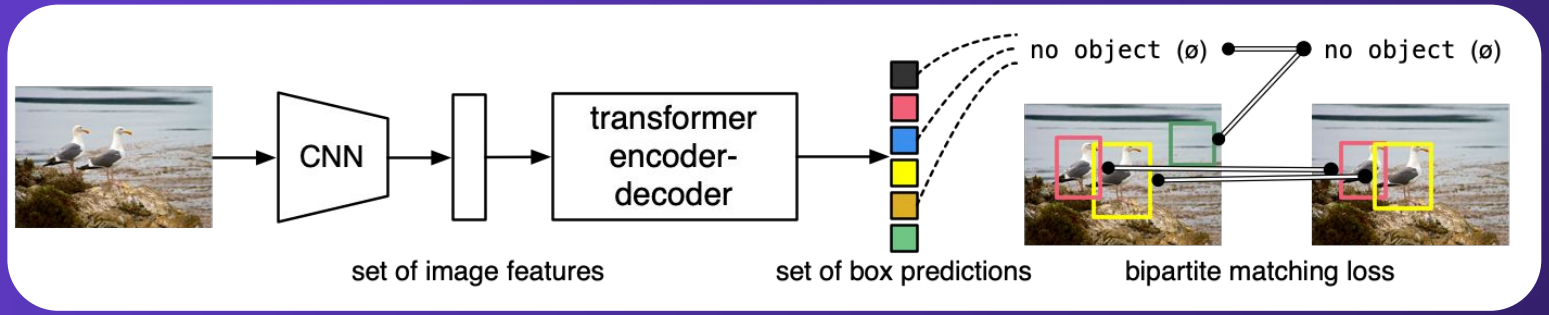
## Detection Transformer (DETR)

TIL-AI
TODAY I LEARNED AI

# DETR (DEtection TRansformer)

Developed by Facebook, they proposed an object detection pipeline with
- CNN - ResNet-50
- A Transformer (encoder/decoder) with positions

**What it is**
DETR approaches object detection as a direct set prediction problem. Given a fixed small set of learned object queries, DETR reasons about the relations of the objects and the global image context to directly output the final set of predictions in parallel.




facebookresearch/detr: End-to-End Object Detection with Transformers (github.com)

# DETR (DEtection TRansformer)

## DETR pseudo algorithm

```
Demo implementation of DETR in minimal number of lines, with the
following differences wrt DETR in the paper:
* learned positional encoding (instead of sine)
* positional encoding is passed at input (instead of attention)
* fc bbox predictor (instead of MLP)
The model achieves ~40 AP on COCO val5k and runs at ~28 FPS on Tesla V100.
Only batch size 1 supported.
"""
def __init__(self, num_classes, hidden_dim=256, nheads=8,
                         num_encoder_layers=6, num_decoder_layers=6):
        super().__init__()

        # create ResNet-50 backbone
        self.backbone = resnet50()
        del self.backbone.fc

        # create conversion layer
        self.conv = nn.Conv2d(2048, hidden_dim, 1)

        # create a default PyTorch transformer
        self.transformer = nn.Transformer(
                hidden_dim, nheads, num_encoder_layers, num_decoder_layers)

        # prediction heads, one extra class for predicting non-empty slots
        # note that in baseline DETR linear_bbox layer is 3-layer MLP
        self.linear_class = nn.Linear(hidden_dim, num_classes + 1)
        self.linear_bbox = nn.Linear(hidden_dim, 4)

        # output positional encodings (object queries)
        self.query_pos = nn.Parameter(torch.rand(100, hidden_dim))

        # spatial positional encodings
        # note that in baseline DETR we use sine positional encodings
        self.row_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
        self.col_embed = nn.Parameter(torch.rand(50, hidden_dim // 2))
```

Resnet50 Backbone

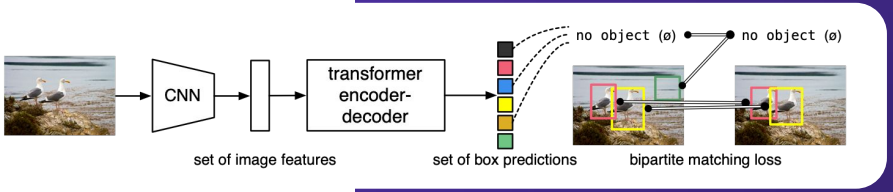
Convolution layers (typically Conv2d layers)

Transformers

Prediction for bounding box and class

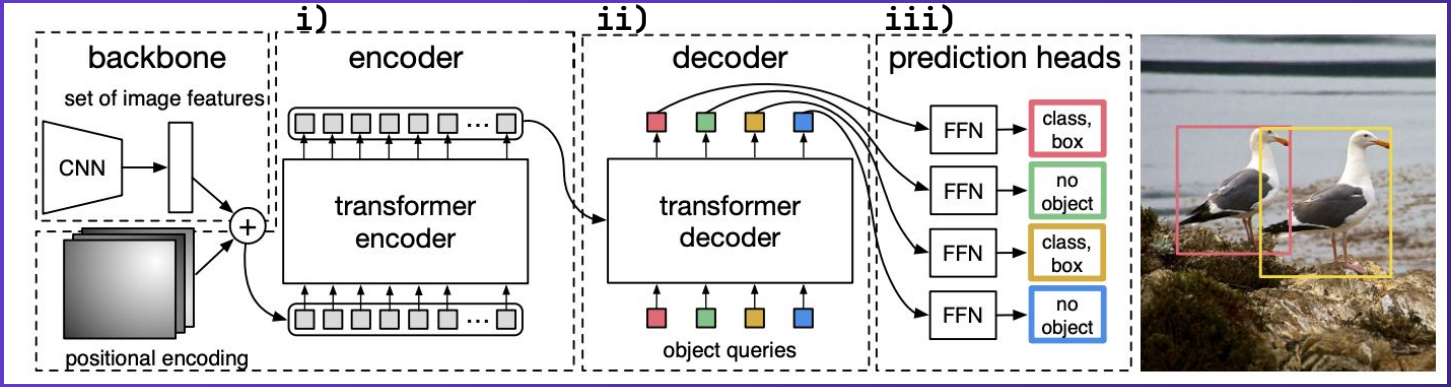
Output for positional encoding

Position encoding shift




Adapted from DETR Colab Notebook

# DETR (DEtection TRansformer)




(top)
i) DETR uses a conventional CNN backbone to learn 2D representation of an input image. The model flattens it and supplements it with positional encoding before passing it into a transformer encoder.

ii) A transformer decoder then takes as a input a small fixed number of learned positional embeddings, which we call object queries, and additionally attends to the encoder output.

iii) We pass each output embedding of the decoder to the shared feed forward network (FFN) that predicts either a detection (class and bounding) or a “no object” class.

TIL-AI
TODAY I LEARNED-AI