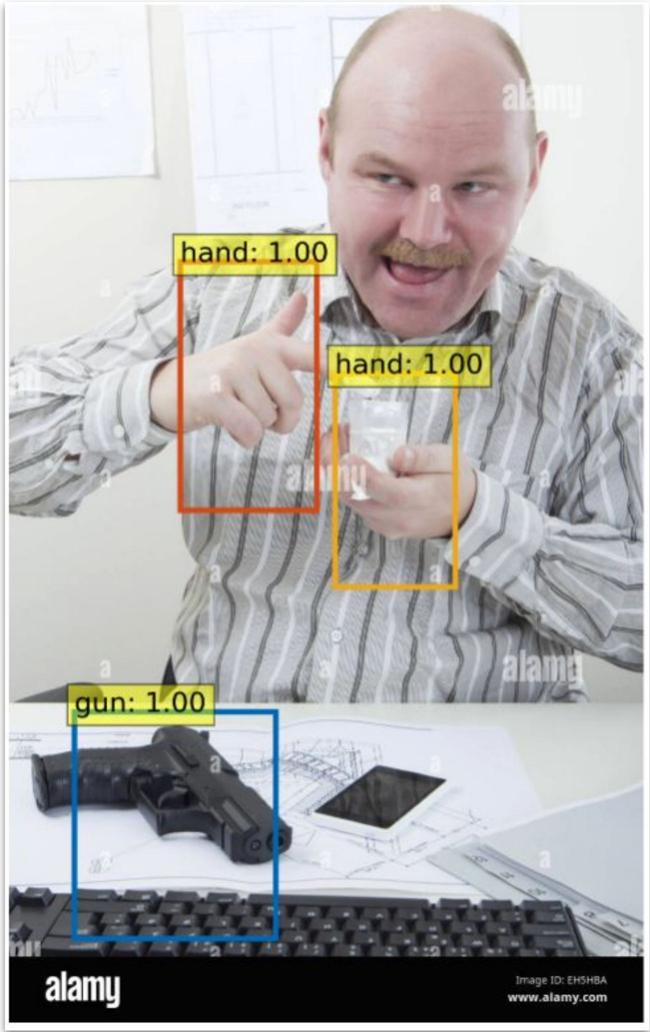
Detection Transformer
For Hands, Guns and Phones

Joey Palanca | Ali Ghazy Alsharif | Marco Lorenz

Overview

1. **Problem definition: Object Detection**
2. History - Object Detection Milestones
3. DETR
    a. Previous work: Transformer and Parallel Decoding
    b. Object Detection Set Prediction Loss
    c. Architecture
    d. HGP-Dataset
4. Results: Hyperparameter Study
5. Live Demo



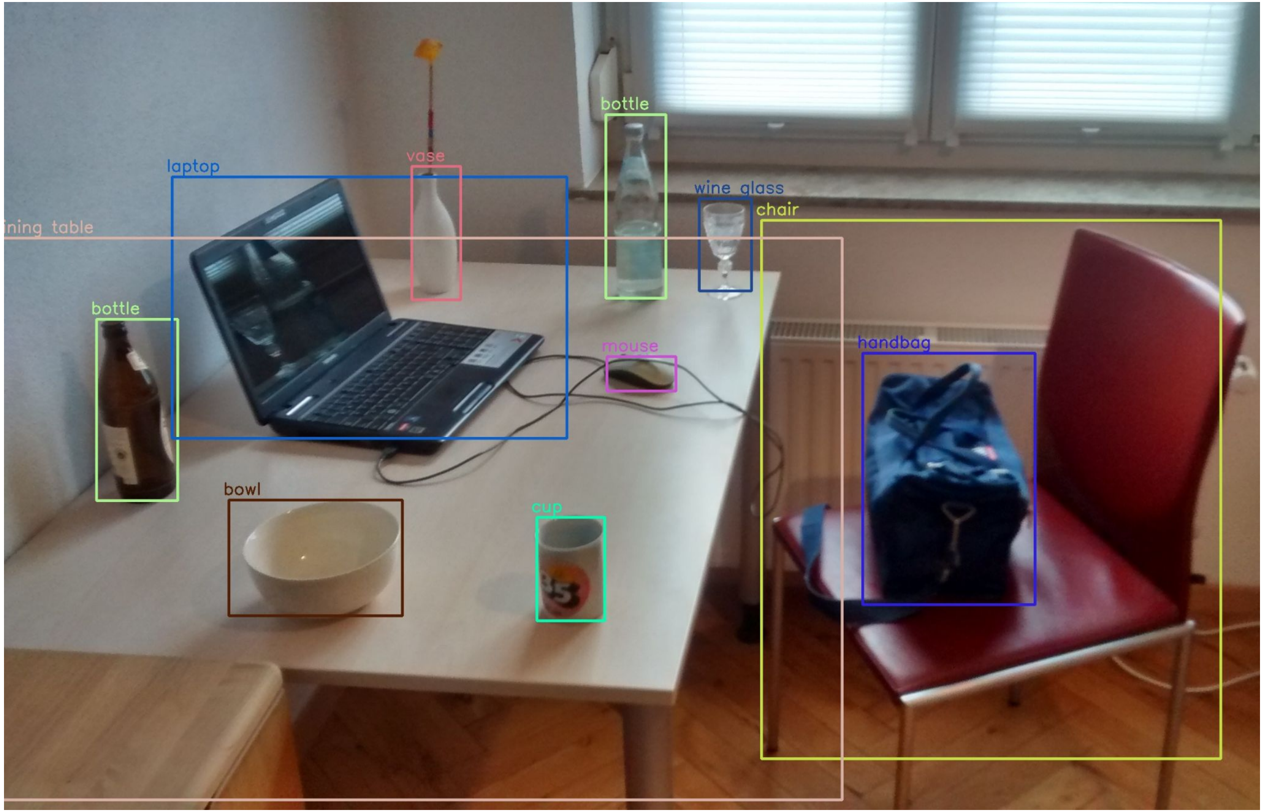
Object Detection

Detecting instances of **semantic objects** of a certain **class** (such as humans, cars, tools) in photos and videos by predicting

- **Object class**
- **Bounding box**

Applications:

- Autonomous Driving
- Robotics
- Image Retrieval
- Video Surveillance
- etc…



Source: https://en.wikipedia.org/wiki/Object_detection

Object Detection - History

Zou et al.: Object Detection in 20 Years: A Survey

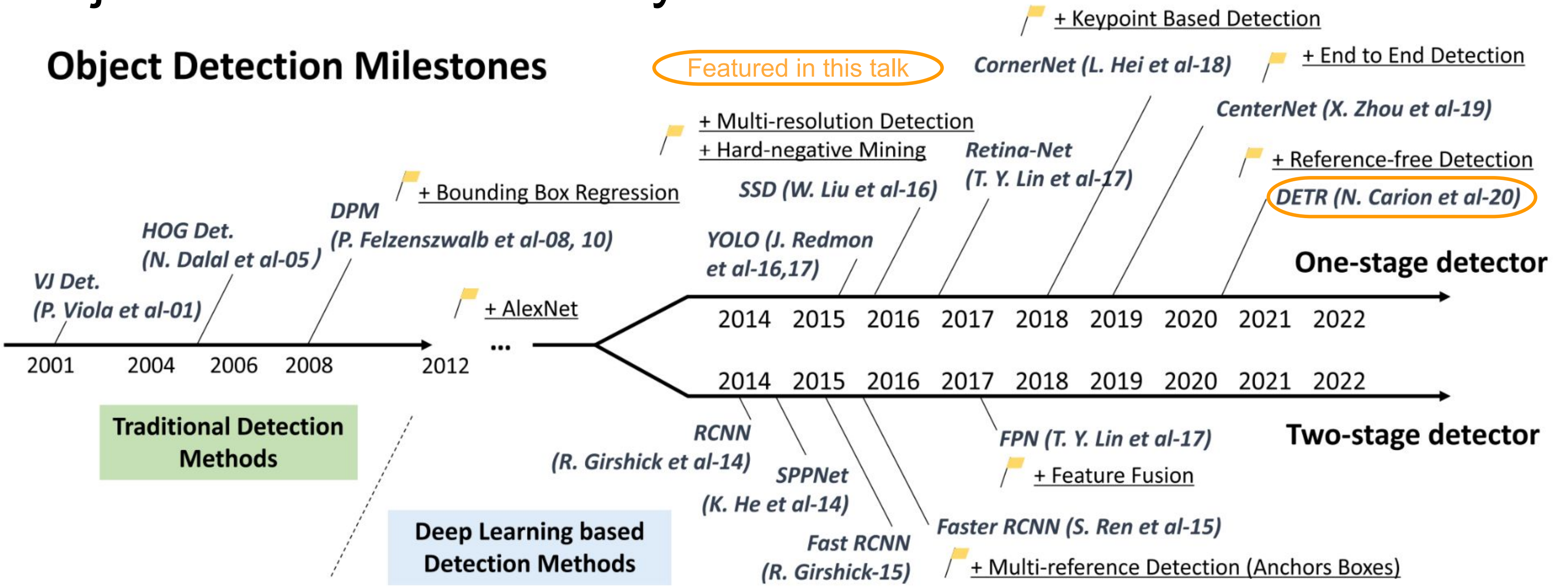


Fig. 2. ***Road map of object detection. Milestone detectors in this figure: VJ Det. [10], [11], HOG Det. [12], DPM [13], [14], [15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20], [21], [22], SSD [23], FPN [24], Retina-Net [25], CornerNet [26], CenterNet [27], and DETR [28].***

Image Source: Zhou et al., Object Detection in 20 Years: A Survey (2019)

DETR - Object Detection as Direct Set Prediction Problem

Main innovations:

Set-based global loss forcing unique predictions

AND

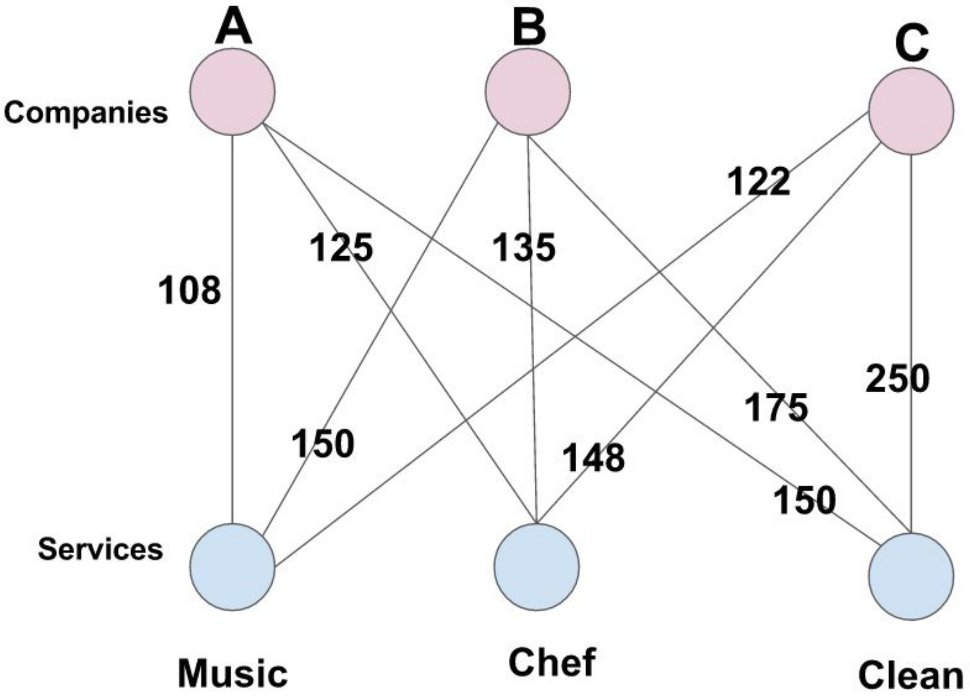
Transformer encoder-decoder architecture applied to Object Detection

→ **Cutting off** hand-designed components based on **prior knowledge** like non-maximum suppression or anchor generation

→ Accuracy and run-time performance on par with highly optimized Faster R-CNN, indicating future potential (Deformable DETR, Swin Transformer)

Source: Carion et al., End-to-End Object Detection with Transformers (2020)

Object detection set prediction loss - Introduction



In the context of DETR:

- Services: Set of predictions
- Companies: Set of ground truth labels and bounding boxes (padded with "no-object" labels)
- Weights: Custom cost function computed for each set of predictions

Image Source: https://brilliant.org/wiki/hungarian-matching/

Object detection set prediction loss - Overview

Two-Step Approach:

1. Compute **optimal assignment** with *Hungarian algorithm*

   → Criterion: Pairwise matching cost w.r.t. class AND bounding box

2. With optimal assignment, compute the *Hungarian loss function to optimize for*

Source: Carion et al., End-to-End Object Detection with Transformers (2020)
Image Source: Wikipedia.org

Hungarian loss - Putting it all together

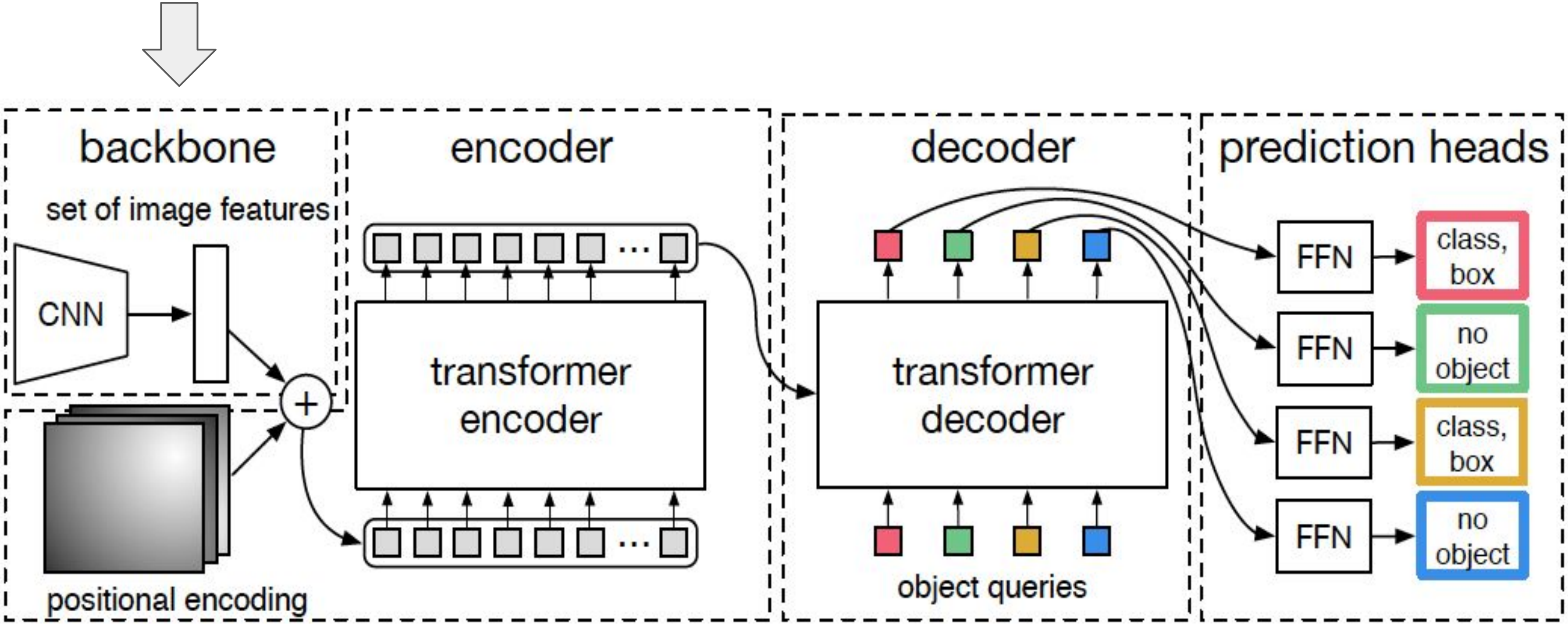
Fixed size N of prediction set

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

Optimal assignment computed in step 1

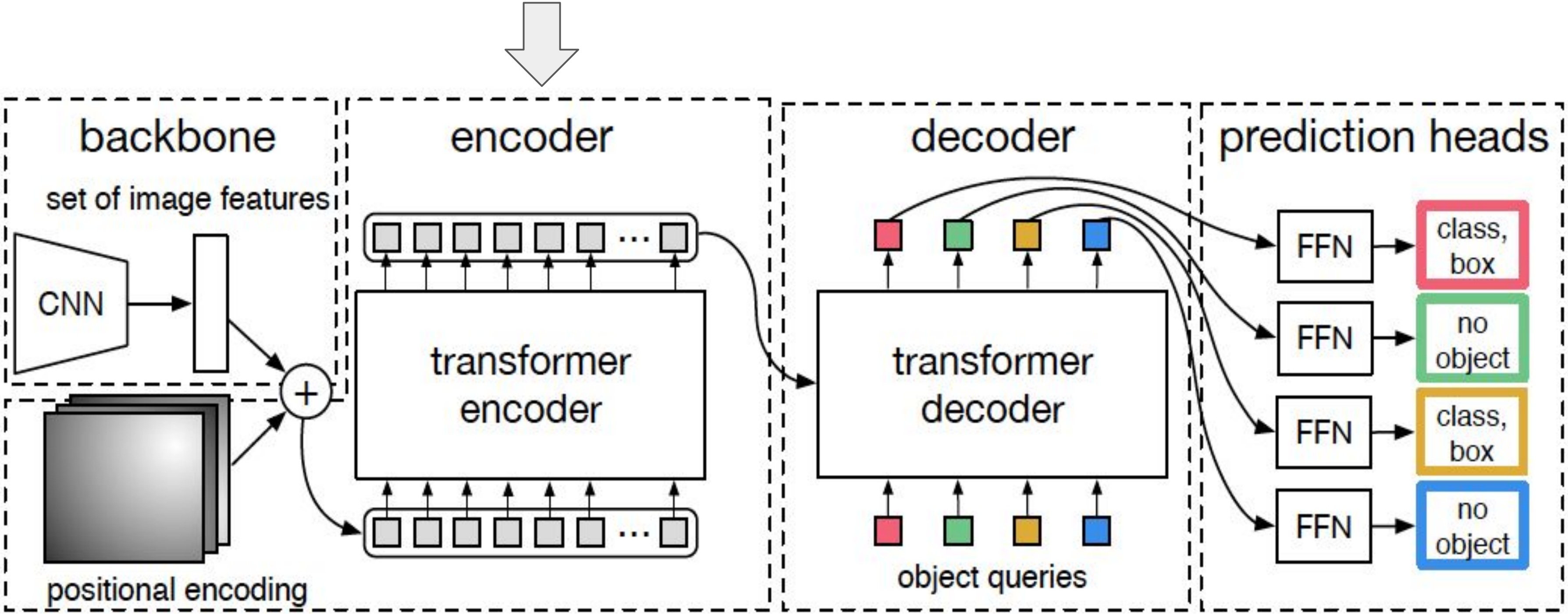
Source: Carion et al., End-to-End Object Detection with Transformers (2020)

DETR Architecture - Overview



Source: Carion et al., End-to-End Object Detection with Transformers (2020)

DETR Architecture - Overview

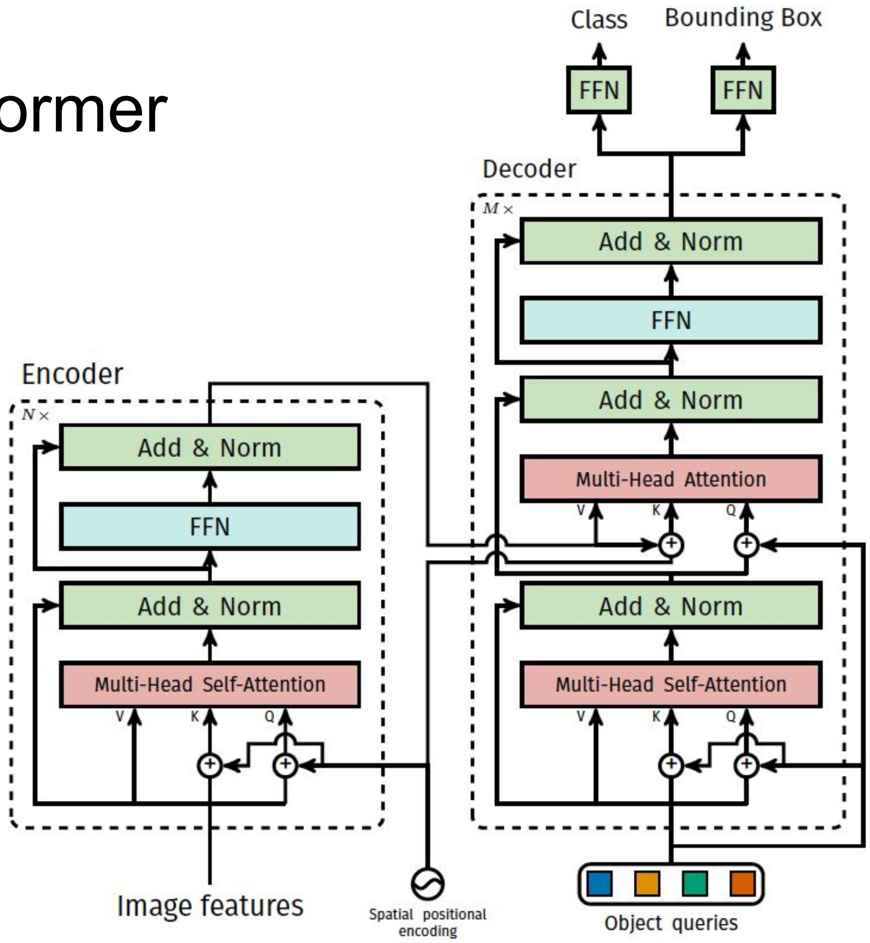


Source: Carion et al., End-to-End Object Detection with Transformers (2020)

DETR Architecture - Transformer Encoder-Decoder

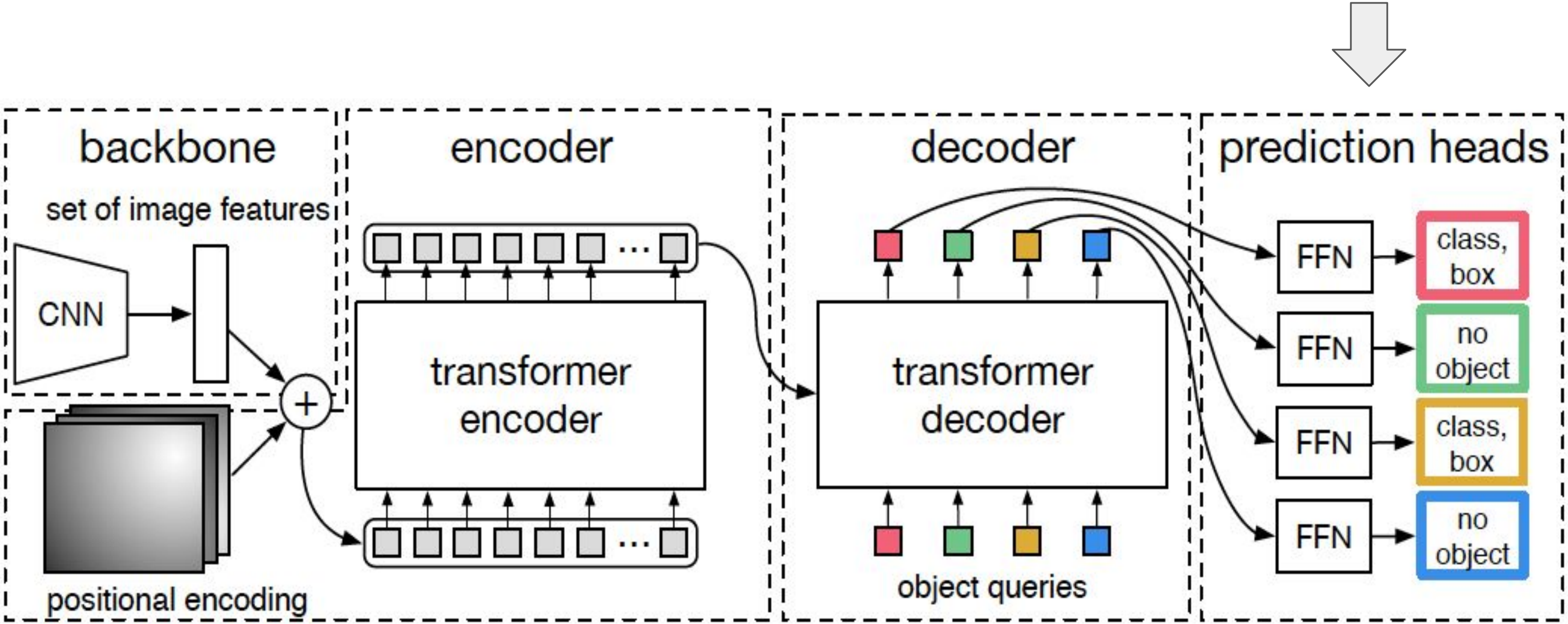
Main differences to original transformer (Vaswani et al., 2017):

- Input: Convolutional feature map instead of word-embeddings
- **Spatial** positional encodings
- **Fixed size** set of Object Queries
- **Parallel decoding** of N object queries as opposed to auto-regressive one-by-one prediction of variable-length output sequence



Source: Carion et al., End-to-End Object Detection with Transformers (2020)

DETR Architecture - Overview



Source: Carion et al., End-to-End Object Detection with Transformers (2020)

DETR Architecture - PyTorch

```python
class DETR(nn.Module):
    def forward(self, samples: NestedTensor):
        """The forward expects a NestedTensor, which consists of:…
        if isinstance(samples, (list, torch.Tensor)):
            samples = nested_tensor_from_tensor_list(samples)
        with annotate("forward_backbone"): # Added by Marco Lorenz on April 8th, 2024
            features, pos = self.backbone(samples)

        src, mask = features[-1].decompose()
        assert mask is not None
        with annotate("forward_transformer"): # Added by Marco Lorenz on April 8th, 2024
            hs = self.transformer(self.input_proj(src), mask, self.query_embed.weight, pos[-1])[0]

        with annotate("forward_output_classes"): # Added by Marco Lorenz on April 8th, 2024
            outputs_class = self.class_embed(hs)
        with annotate("forward_output_boxes"): # Added by Marco Lorenz on April 8th, 2024
            outputs_coord = self.bbox_embed(hs).sigmoid()
        out = {'pred_logits': outputs_class[-1], 'pred_boxes': outputs_coord[-1]}
        if self.aux_loss:
            out['aux_outputs'] = self._set_aux_loss(outputs_class, outputs_coord)
        return out
```

Source: Carion et al., End-to-End Object Detection with Transformers (2020)

COCO vs HGP

- COCO (Common Objects in Context): Huge Dataset, rich API
  - Large-scale image recognition dataset
  - For object detection, segmentation, and captioning tasks
  - **330,000 images**, each annotated with **80 object categories** and 5 captions describing the scene
  - Sponsored by Microsoft, Meta, etc
- HGP (Hands, Guns and Phones) Dataset: Small Dataset, not so rich API
  - **1199 images** (1989 for training and 210 for testing): about 1:10
  - People using guns or phones in real-world scenarios (people making phones reviews, shooting drills, or making calls)
  - Labeled with the bounding boxes of Hands, Phones and Guns
  - Collected from Youtube videos, with different sizes.

Sources: https://paperswithcode.com/dataset/hgp: https://cocodataset.org/

HGP Dataset - Implementation of VisionDataset (PyTorch)

```python
class HGPDetection(VisionDataset):
    def __init__(self, img_folder: str, lab_folder: str, ann_file: str, image_set: str, transforms):...

    def _load_image(self, id: int) -> Image.Image:...

    def _load_target(self, id: int) -> List[Any]:...

    def __getitem__(self, index: int) -> Tuple[Any, Any]:
        image_id = self.ids[index]
        image = self._load_image(index)
        target = self._load_target(index)

        target = {'image_id': image_id, 'annotations': target}
        image, target = self.prepare(image, target)
        if self._transforms is not None:
            image, target = self._transforms(image, target)

        return image, target

    def __len__(self) -> int:
        return len(self.ids)
```

Precision Metric - Intersection over Union

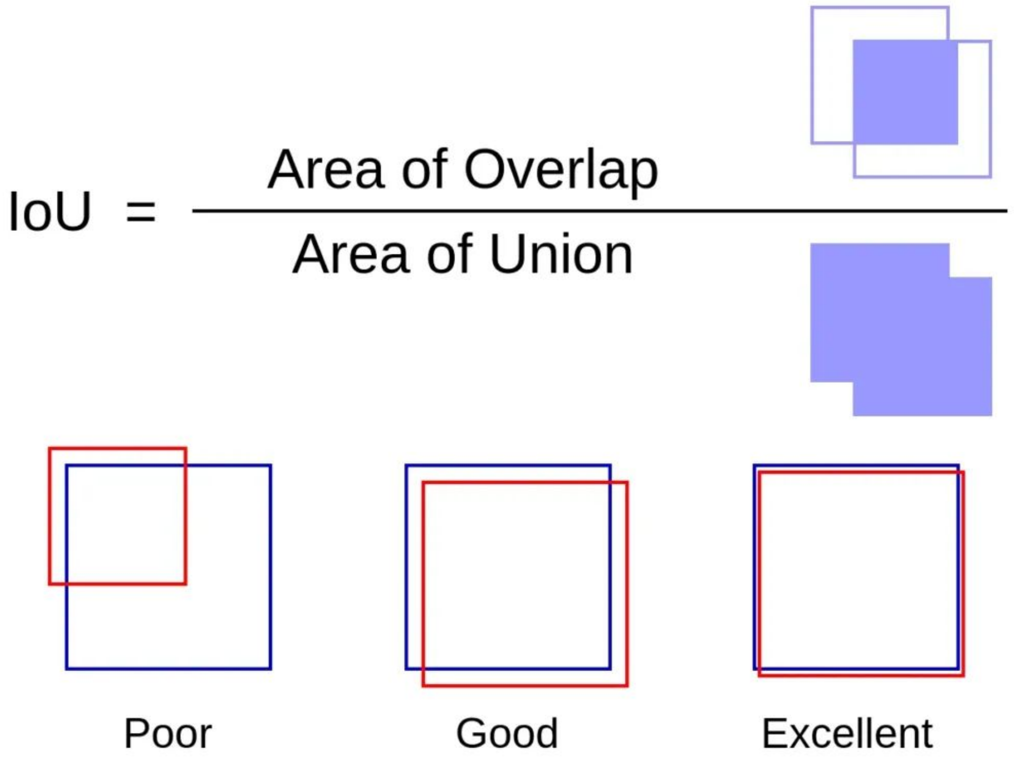
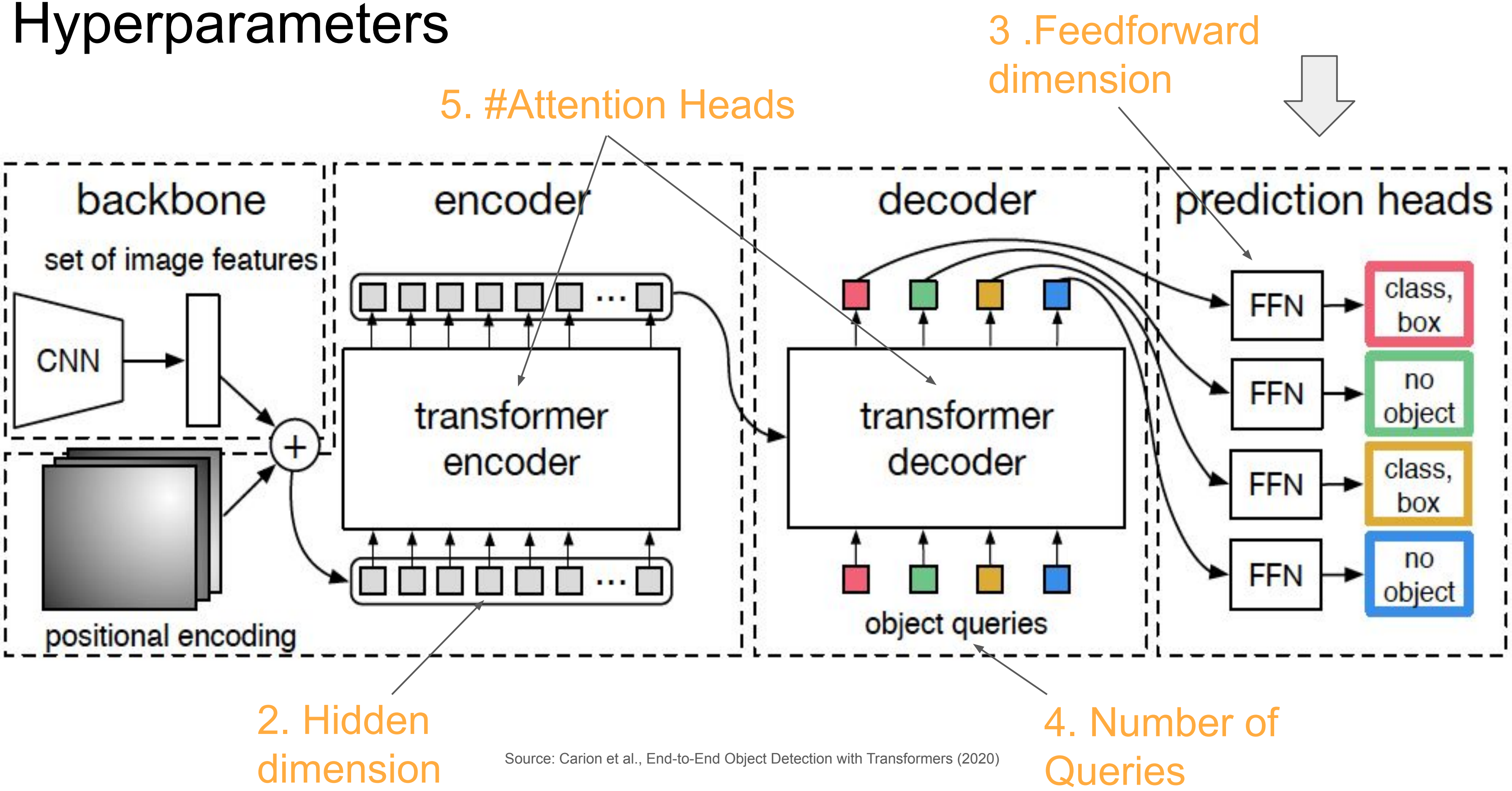


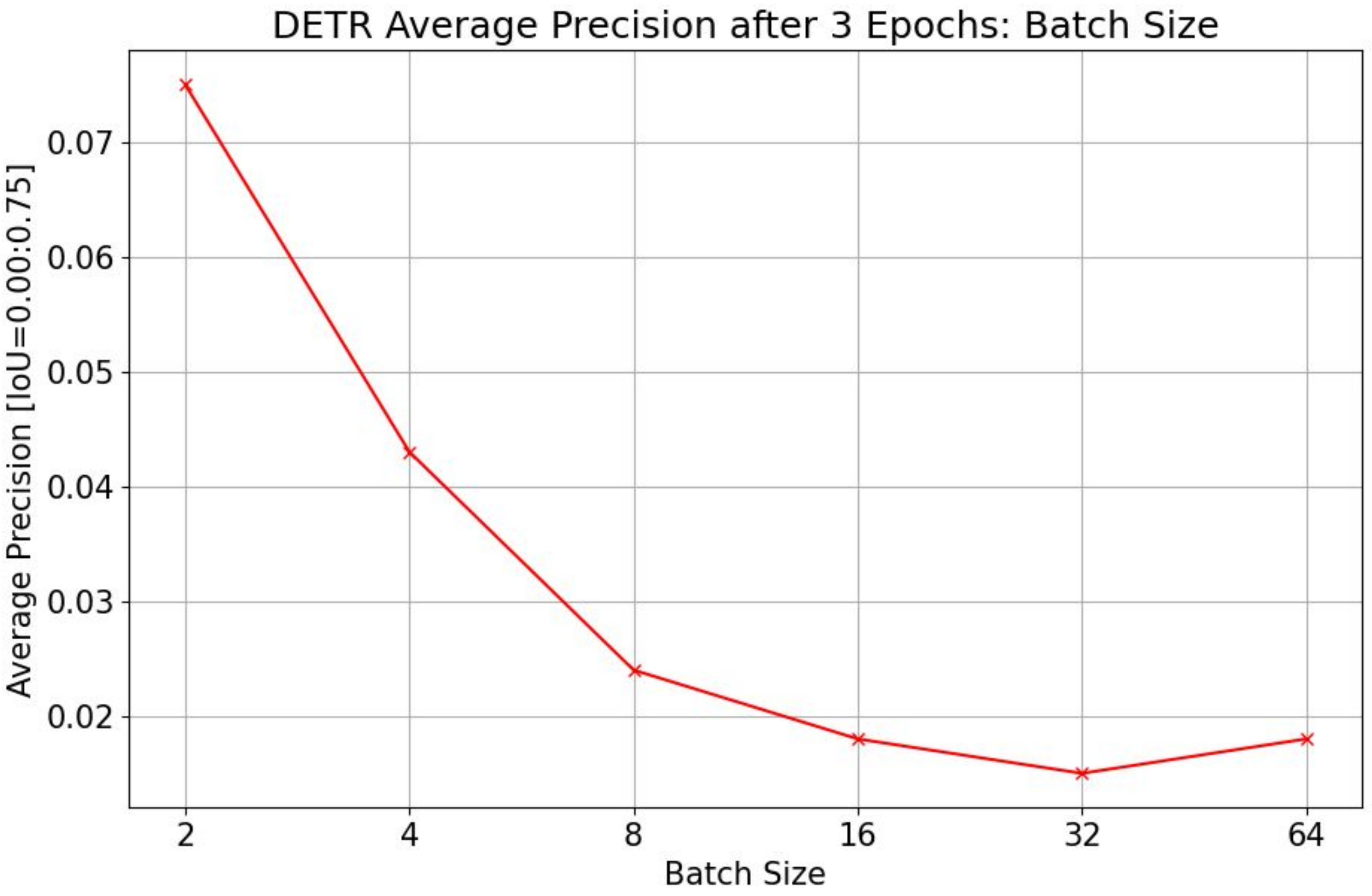
Image Source: https://idiotdeveloper.com/

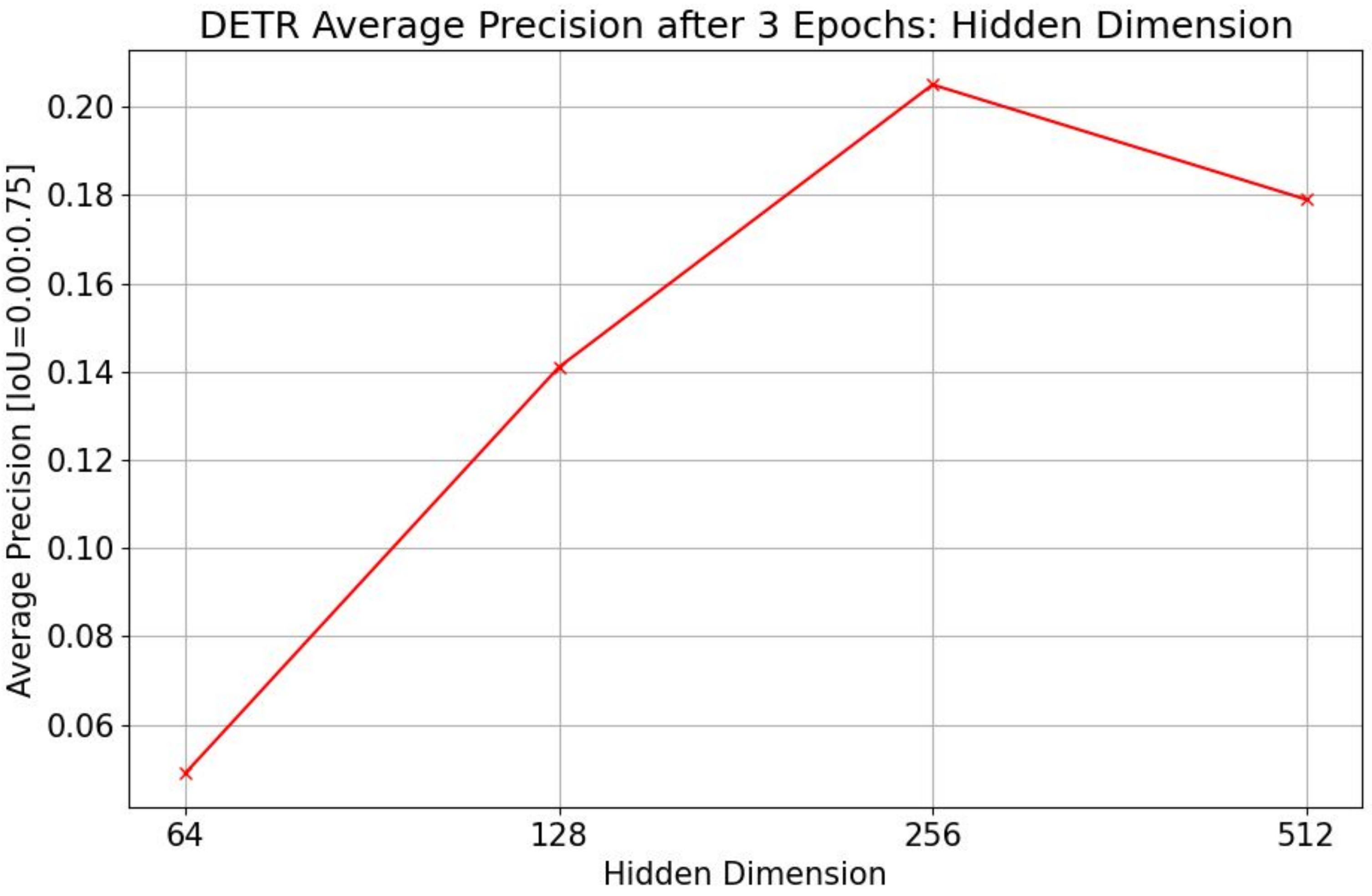
Hyperparameters

1. Batch Size

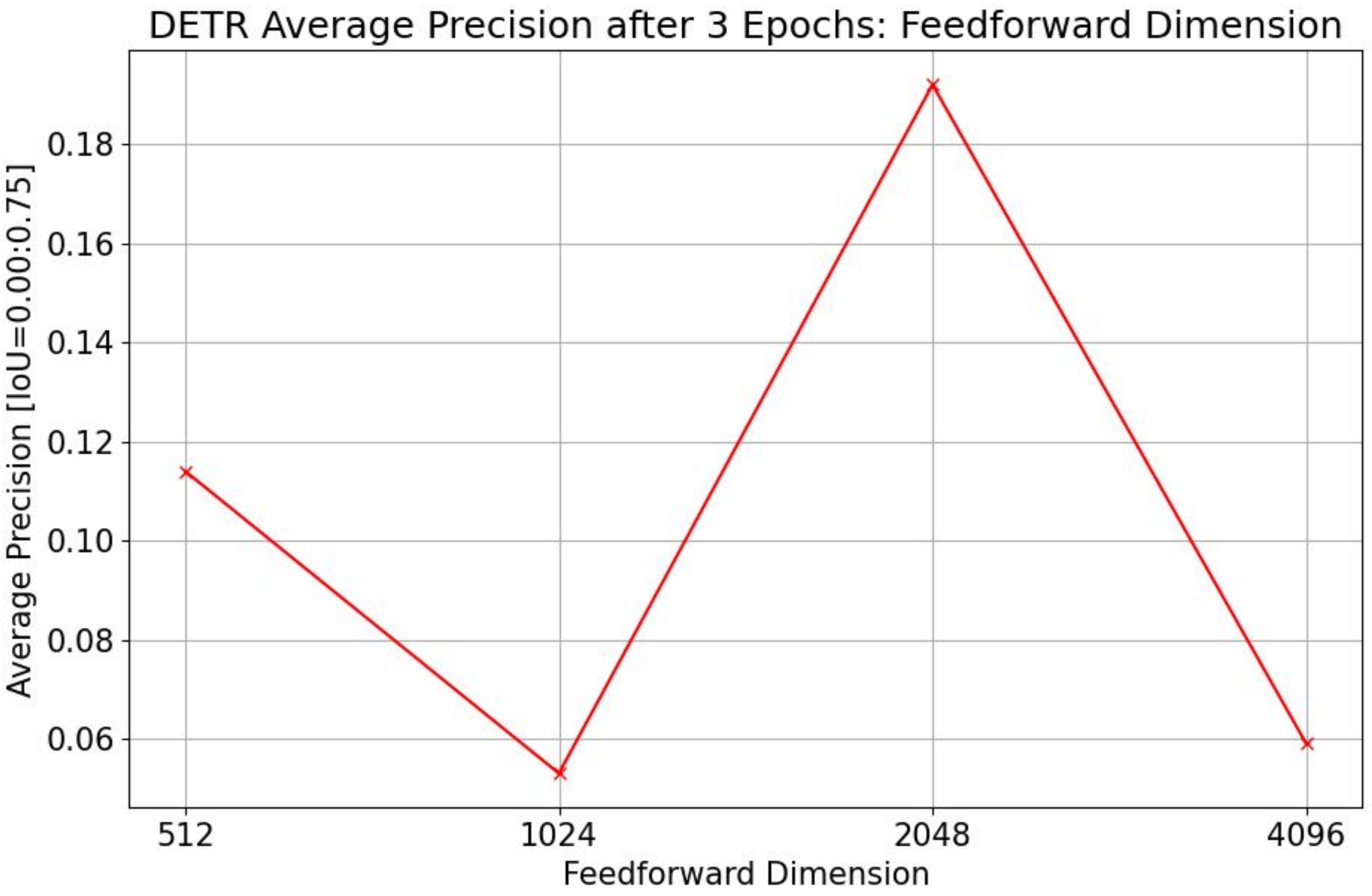


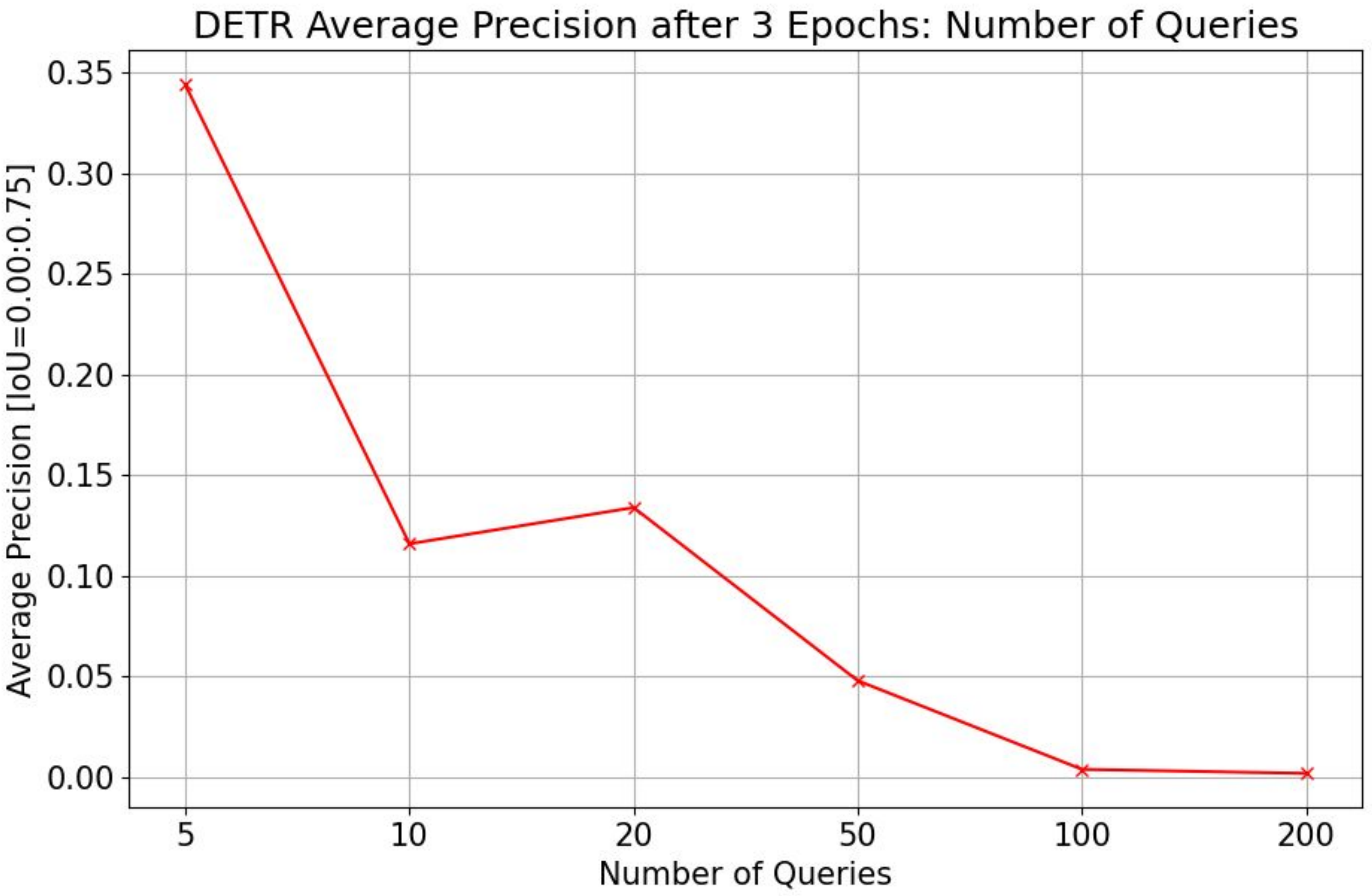
Source: Carion et al., End-to-End Object Detection with Transformers (2020)

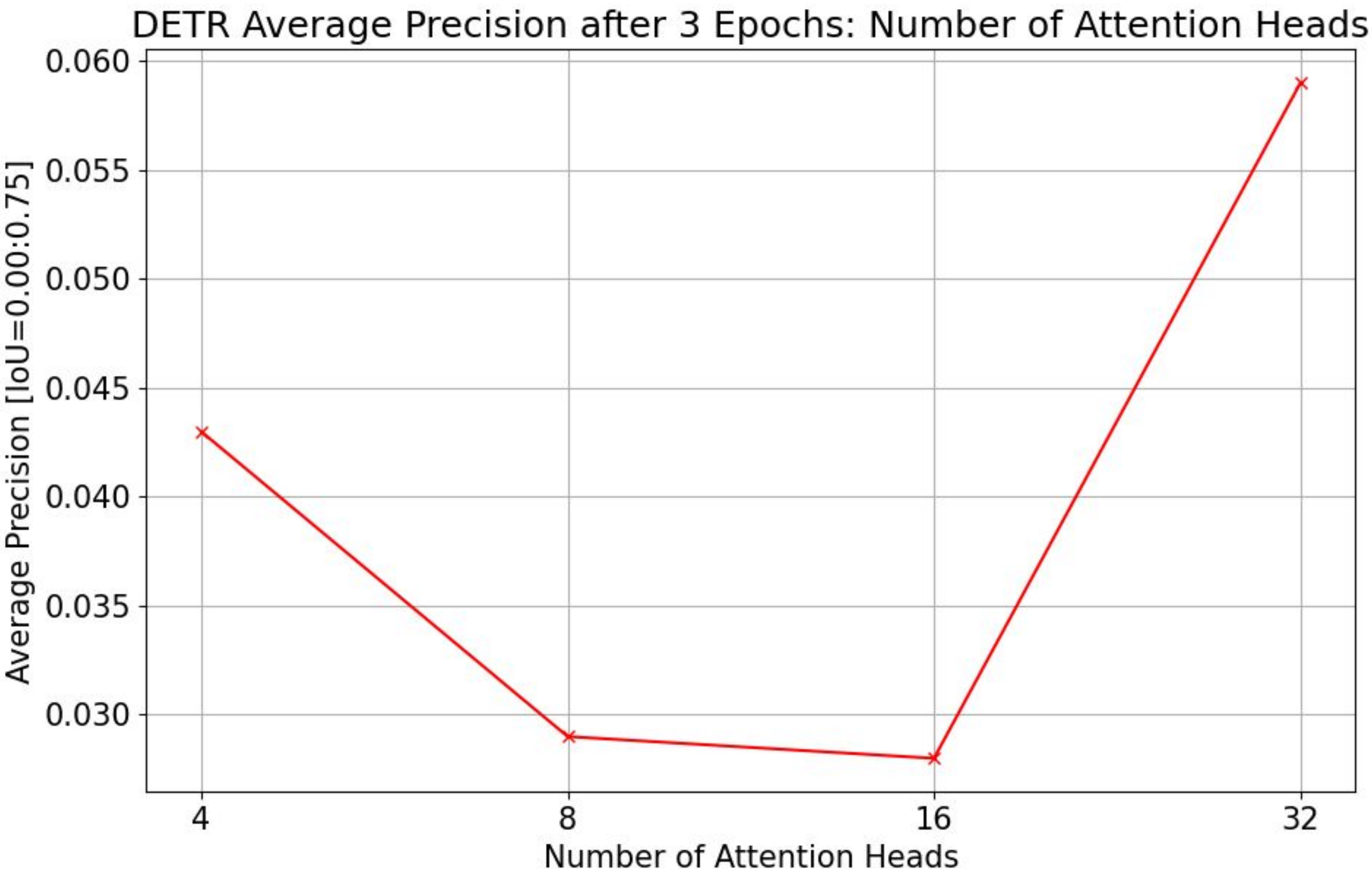

DETR Average Precision after 3 Epochs: Batch Size
Average Precision [IoU=0.00:0.75]
0.02
0.03
0.04
0.05
0.06
0.07
2
4
8
16
32
64
Batch Size



DETR Average Precision after 3 Epochs: Hidden Dimension
Average Precision [IoU=0.00:0.75]
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.06
64
128
256
512
Hidden Dimension


DETR Average Precision after 3 Epochs: Feedforward Dimension
Average Precision [IoU=0.00:0.75]
0.06
0.08
0.10
0.12
0.14
0.16
0.18
512
1024
2048
4096
Feedforward Dimension


DETR Average Precision after 3 Epochs: Number of Queries
Average Precision [IoU=0.00:0.75]
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
5
10
20
50
100
200
Number of Queries


DETR Average Precision after 3 Epochs: Number of Attention Heads
Average Precision [IoU=0.00:0.75]
Number of Attention Heads
0.060
0.055
0.050
0.045
0.040
0.035
0.030
4
8
16
32

Live-Demo

References

[1] Z. Zou, K. Chen, Z. Shi, Y. Guo, and J. Ye. Object detection in 20 years: A survey, 2023.

[2] N. Carion, F. Massa, G. Synnaeve, N. Usunier, and A. K. anc Sergey Zagoruyko. End-to-end object detection with transformers. https://doi.org/10.48550/arXiv.2005.12872, May 2020.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023.

[4] D.-V. et al. Hgp (hands guns and phones dataset). https://paperswithcode.com/dataset/hgp, November 2021.

Websites:
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/
https://en.wikipedia.org/wiki/Object_detection