Object Detection
(Beyond Faster R-CNN)

Earl Wong

Advances Since Faster R-CNN with FPN

- In the Detectron release from Facebook [2018], small object detection was improved by adding a feature pyramid network to Faster R-CNN.

- In the Detectron2 release from Facebook [2019], an algorithm employing a cascade of Faster R-CNN detectors was introduced.

- This further elevated the performance of Faster R-CNN.

- More recently [2020], a transformer based detector [DETR-Detection Transformer] was introduced into the object detection space.

Cascaded Faster R-CNN

- Cascade Faster R-CNN was motivated by the following fact: It was unreasonble to expect a single detector employing a fixed IoU (intersection over union) threshold to perform uniformly well over a range of IoU thresholds.

- This was because input objects could occur in all shapes and sizes, under a myriad of conditions.

- In general, when the detector threshold was decreased, the number of detected objects increased, resulting in many “noisy” outputs.

- In general, when the detector threshold was increased, the number of detected objects decreased (exponentially), resulting in subsequent model overfitting due to the lack of positive training examples.

Cascaded Faster R-CNN

- In an attempt to improve the detector performance, an ensemble of Faster R-CNN networks (with different thresholds) was created.

- Unfortunately, because the detectors with large thresholds (still) overfit, these “high threshold” detectors added little to no value to the ensemble.

- In another attempt, a cacade of Faster R-CNN detectors with a fixed threshold was created.

- Unfortunately, this architecture only improved the detector performance for the specific threshold selected.

Cascaded Faster R-CNN

- The correct solution employed a cascade of Faster R-CNN detectors with increasing thresholds (0.5, 0.6, 0.7, etc.)

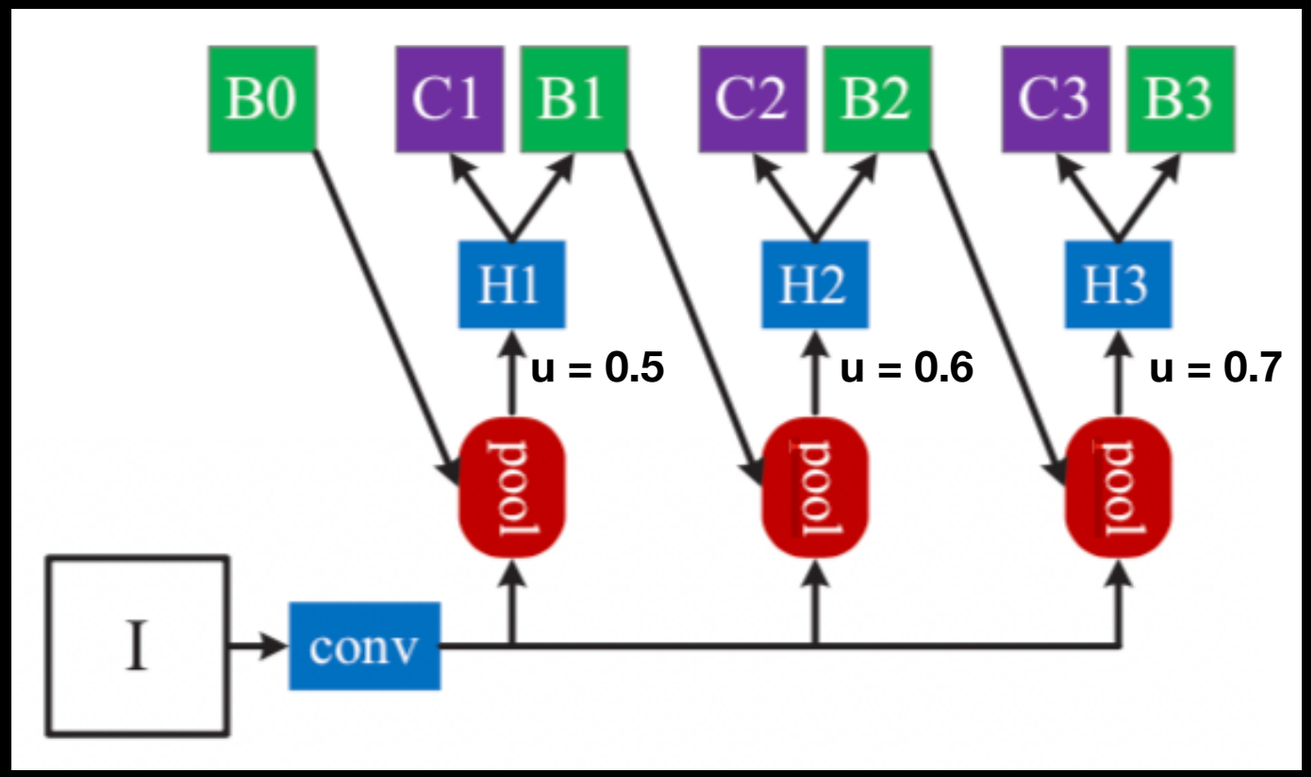
- In addition, each detector now needed to be trained stage by stage.

- Using this approach, a significant number of positive training examples were now available in the input data distribution for the subsequent detector.

- i.e. Each stage now performed a “rebalancing process” by rejecting / accepting a certain subset of it’s inputs. (i.e. resampling)

- This resulted in a 4 point improvement in object detector precision.

Cascaded Faster R-CNN



The cascaded Faster R-CNN network was trained stage by stage, with increasing threshold values.

However, after four stages, the authors observed that the performance began to degrade.

Observations of this nature are a re-occurring problem / concern in much of deep learning.

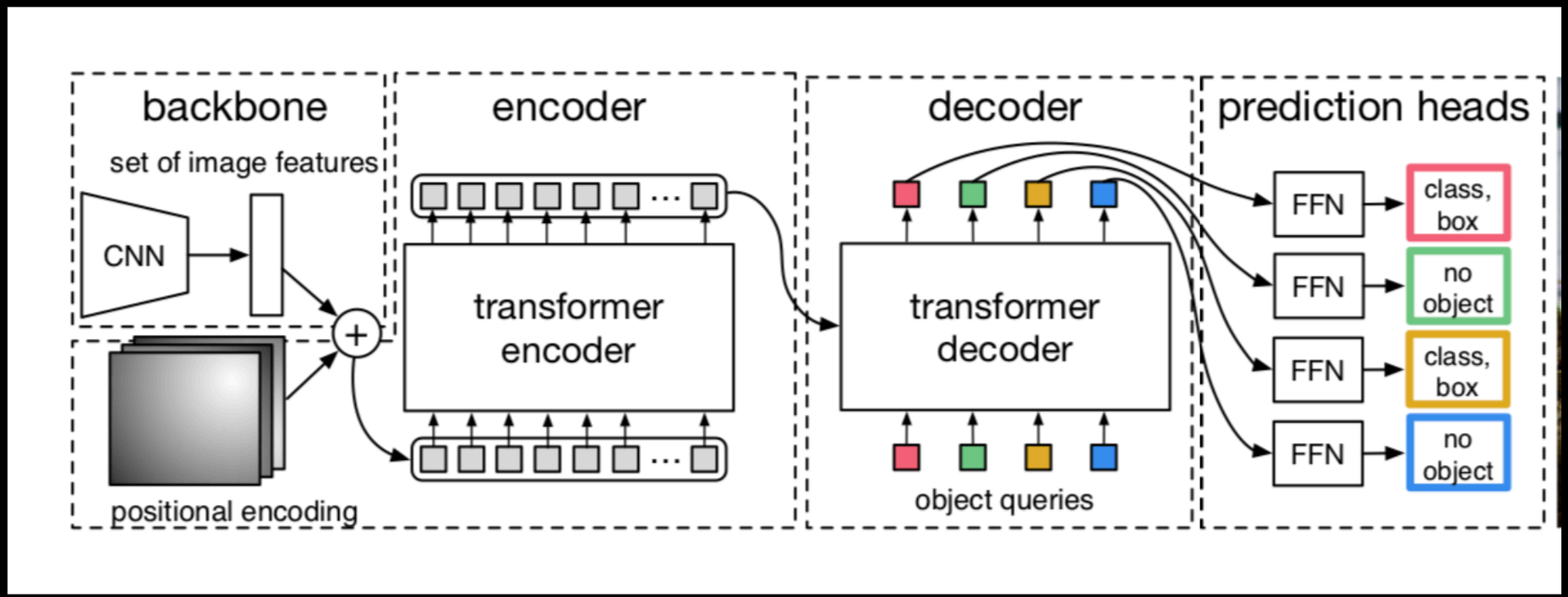
i.e. Architectures are tuned for an optimal “midpoint” configuration, whereas limiting arguments conclude that best result should plateau, or occur, at “infinity”.

In practice, the empirical observations are rarely “root caused”. (Exceptions include: ResNet)

DETR

- Under DETR, object detection was now cast as a set prediction problem.
- Here, set prediction was performed using a bi-partite matching loss function and an encoder - decoder transformer architecture.
- Unlike previous transformer architectures, DETR now returned a set of object predictions simultaneously / parallel decoding.
- DETR currently outperforms Faster R-CNN on large objects due to the global self attention mechanism present in the transformer, but underperforms on small objects.
- In addition, when the number of objects in a scene exceeds half the number of DETR outputs N (where N is an architectural design parameter), DETR performance begins to degrade.

DETR-High Level Block Diagram



The C channels / feature maps from the Nth layer of a CNN were used as inputs to the transformer.

First, the C channels were converted to d channels using 1x1 convolutions.

Next, each channel was converted to a 1D vector, before being sent to transformer as a“word embedding” (token).

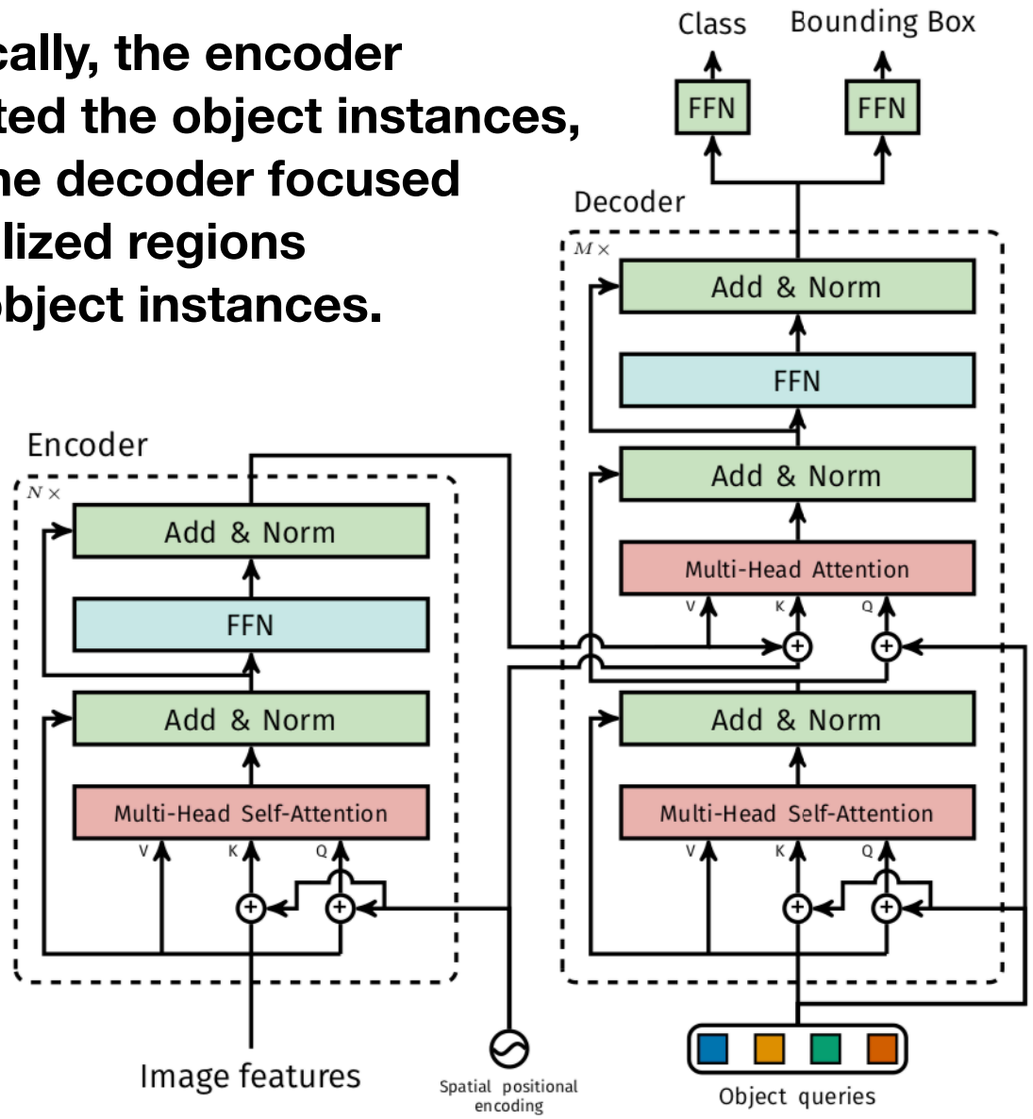
Positional information was added to each word embedding, before being fed into the transformer.

At the transformer output, each output embedding was fed into a feed forward neural network (FFN).

Each FFN produced a class output and it’s corresponding bounding box estimates.

DETR-Inside the Transformer

Empirically, the encoder separated the object instances, while the decoder focused on localized regions in the object instances.



The transformer consisted of an encoder and a decoder.

The main detection elements in the transformer were: the self attention layers in the encoder, the FFN, the multiple decoder layers and the positional encodings.

According to the authors, the FFN blocks inside the encoder and decoder used shared learned weights, unlike the FFN blocks outside the transformer.

Unclear (from the diagram), but stated by the authors, was the fact that positional encodings were added to the inputs of every multihead attention block in the encoder-decoder network.

Finally, object query inputs were learned through training.

DETR

The loss function used by DETR was computed in the following manner:

First, the permutation associated with the minimum match loss was determined:

$$\hat{\sigma} = argmin_{\sigma \in \wp_N} \sum_{i}^{N} L_{match}(y_i,\ \hat{y}_{\sigma(i)})$$

$$L_{match}(y_i,\ \hat{y}_{\sigma(i)}) = -1_{\{c_i \neq \phi\}} \hat{p}_{\sigma(i)}(c_i) + 1_{\{c_i \neq \phi\}} L_{box}(b_i,\ \hat{b}_{\hat{\sigma}(i)})$$

Next, the loss was computed using this specific permutation, using all N output results:

$$L_{Hungarian}(y_i,\ \hat{y}) = \sum_{i=1}^{N} [-log\hat{p}_{\sigma(i)}(c_i) + 1_{\{c_i \neq \phi\}} L_{box}(b_i,\ \hat{b}_{\hat{\sigma}(i)})$$

where the box loss was comprised of an L1 loss term and an IoU loss t**erm:**

$$L_{box}(b_i,\ \hat{b}_{\hat{\sigma}(i)}) = \lambda_{IoU}(b_i,\ \hat{b}_{\hat{\sigma}(i)}) + \lambda_{L1}|b_i - \hat{b}_{\sigma(i)}|_1$$

$$\mathcal{L}_{\text{iou}}(b_{\sigma(i)}, \hat{b}_i) = 1 - \left(\frac{|b_{\sigma(i)} \cap \hat{b}_i|}{|b_{\sigma(i)} \cup \hat{b}_i|} - \frac{|B(b_{\sigma(i)}, \hat{b}_i) \setminus b_{\sigma(i)} \cup \hat{b}_i|}{|B(b_{\sigma(i)}, \hat{b}_i)|} \right)$$

$|.|$ means "area"

and b consisted of 4 values: x and y center coordinates and the box height and width **(**relative to the image size)

References

- Cascade R-CNN: Delving Into High Quality Object Detection, Cai and Vasconcelos, CVPR 2018

- End to End Object Detectors with Transformers, Carion, et. al., arXiv May 2020