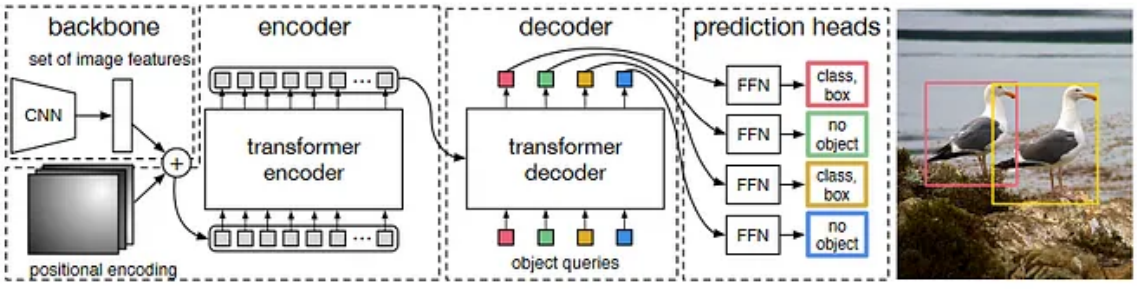
REPORT: END-TO-END OBJECT DETECTION WITH TRANSFORMERS

- Atanda Abdullahi Adewale, M2 ViBot 2023

INTRODUCTION

The video explains the use of DETR, a framework for end-to-end object detection with Transformers. DETR utilizes a transformer encoder-decoder architecture and a set-based global loss that enforces unique predictions. The matching process between the output and ground truth is achieved through bipartite matching using the Hungarian algorithm. The research paper focuses on training the detection pipeline without heuristics like non-max suppression. Experimental results show that DETR outperforms the Faster R-CNN baseline on large objects but underperforms on smaller objects. The importance of attention mechanisms and the potential for improving performance on smaller objects are highlighted. Additionally, the video discusses the extension of DETR for the panoptic segmentation task by adding a segmentation head, leading to significant improvements in segmenting objects.

HOW IT WORKS:



1. Data Preprocessing or Preparation:

Preprocess the COCO dataset, resize images, normalize pixels, and handle annotations (). Split the dataset into training and validation sets.

Forward Pass the dataset batches (images) through the ResNet101 backbone to extract visual features.

Flatten or reshape the features to create a compact representation.

Feed the compact feature representation through the transformer encoder-decoder architecture.

2. Training:

Implement bipartite matching to associate predicted object queries with ground truth objects. Calculate the Hungarian loss based on the matched pairs, including terms for bounding box regression and classification accuracy.

How?

DETR infers a fixed-size set of N predictions, in a single pass through the decoder, where N is set to be significantly larger than the typical number of objects in an image.

DETR needs to find one-to-one matching for direct set prediction without duplicates.

Firstly,

Let y be the ground truth set of objects, and ^y = {^yi}, where i is from 1 to N, be the predicted set. Thus, y can be an empty set for no object.

So we find a **bipartite matching** between these two sets using a matching function across a permutation of N elements with the lowest cost as follows:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}),$$

and

Lmatch(*yi*, ^y(*i*)) is:

$$-\mathbb{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$$

- where Lmatch(yi, ^y(i)) is a pair-wise matching cost between ground truth yi and a prediction with index σ(i).

- The matching cost takes into account both the class prediction and the similarity of predicted and ground truth boxes: yi={bi, ci}.
- bi is a vector that defines ground truth box center coordinates and its height and width relative to the image size, i.e. range of 0 to 1.
- ci is the class probability.

Secondly,

Compute the Hungarian loss, which is a linear combination of a negative log-likelihood for class prediction and a box loss defined later:

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

where **^σ** is the **optimal assignment** computed in the first step. The log-probability term is down-weighted by a factor of 10 when *ci* is empty set to tackle class imbalance issue.

It does a negative log-likelihood between all N permutations of predictions and ground truth to penalize the extra and incorrect boxes predicted and ground truth boxes: yi={bi, ci}.

Box Loss Function (L_box)

The bounding box loss is a linear combination of the L1 loss and the generalized IoU (GIoU) loss:

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

The Hungarian loss considers both the accuracy of predicting object categories and the precision of predicting object locations.

Lastly :

Perform backpropagation using the calculated Hungarian loss and stochastic gradient descent to update the model's parameters., Iterate through multiple epochs, adjusting the model to minimize the loss.

3. Validation:

For inference, pass new images through the trained model. Decode the predictions from the model to obtain bounding box coordinates and class labels for detected objects.

Evaluation and Fine-Tuning is done using metrics like precision, recall, and mAP (mean Average Precision).

DETR not using Non-max Suppression (NMS):

Non-maximum suppression is a technique used in object detection to filter out redundant bounding box predictions. After obtaining bounding boxes and confidence scores from a model, NMS involves sorting the boxes by confidence scores and iteratively selecting the highest-scoring box. For each subsequent box, it compares and removes overlapping boxes with lower scores based on a predefined threshold (typically using intersection over union). This process continues until all boxes have been considered, resulting in a final set of non-overlapping, high-confidence bounding box predictions. It helps improve the precision of object detection by retaining only the most reliable detections.

THE ARCHITECTURE:

DETR architecture relies CNN backbone to extract a compact feature representation, an encoder-decoder transformer, and a simple feed-forward network (FFN) that makes the final detection prediction

CNN Backbone:

- Use a CNN as the backbone to process the input image and extract features. The CNN captures hierarchical visual features at different spatial scales.
- These features serve as a rich representation of the image and are typically obtained from intermediate layers of the CNN.

Transformer Encoder:

- Pass the flattened features through a transformer encoder. The encoder processes the sequence of tokens, capturing the global context and relationships between different parts of the image.
- The self-attention mechanism in the transformer allows the model to consider long-range dependencies.

Transformer Decoder:

- Pass the output of the transformer encoder and the object queries through a transformer decoder. The decoder refines the representation and generates object-specific queries.
- During decoding, the model predicts bounding box coordinates, class labels, and other attributes for each object.

FFN: In the DETR architecture, the transformer components (encoder and decoder) process the visual information and generate object queries. The FFN layers play a crucial role in refining the representation of these queries and capturing intricate features. The FFN serves as a key element in the transformer's ability to capture both local and global contextual information in the image features.

KEYWORDS

BIPARTITE MATCHING:

Bipartite matching is simply determining the optimal pairing between the predicted bounding boxes and the actual objects in the image. Cost Matrix computed for predicted bounding box and each ground truth bounding box. This cost can represent spatial misalignment (e.g., distance or IoU) and class label discrepancy.

Sets of matched pairs are obtained where each predicted bounding box is associated with a ground truth bounding box. The pairs are chosen in a way that minimizes the total cost.

Then Hungarian Algorithm, the goal is to minimize the total cost of these pairings.

THE HUNGARIAN MATCHING LOSS:

Starts with computing Bounding Box Regression Loss i.e.: For each matched pair of predicted object queries label and ground truth objects label, compute the difference between the predicted bounding box coordinates (e.g., x, y, width, height) and the corresponding ground truth bounding box coordinates.

For predicting inaccuracies in the spatial location and size of objects, apply a loss function, often a regression loss like Smooth L1 or Mean Squared Error, to quantify the disparity between the predicted and ground truth bounding box information.

Then use a classification loss, such as Cross-Entropy Loss, to quantify the difference between the predicted class probabilities and the true class probabilities.

The overall Hungarian loss is computed by summing the individual losses for all matched pairs, during training, the model aims to minimize this total loss, which guides the adjustments of its parameters through backpropagation.

CONCLUSION

So, DETR is a fresh design for object detection systems based on transformers and bipartite matching loss for direct set prediction.

After testing out the DETR object detection model against the Faster R-CNN baseline. It turns out that the DETR model did better than Faster R-CNN when it came to spotting big objects, likely thanks to the processing of global information performed by self-attention. But, interestingly, it kind of lagged when it came to smaller objects compared to other object detectors of the same magnitude.

Some objects dominate a large part of the image. DETR's transformer, with its global attention across all layers, excels at capturing these large objects. In contrast, Faster R-CNN, relying on a CNN approach, requires going through multiple layers to understand such sizable objects. In essence, DETR's global attention proves crucial for effectively handling significant elements in the image.

Note: It takes long training hours and is not real-time. The transformer architecture leads to significant overhead in training/inference.