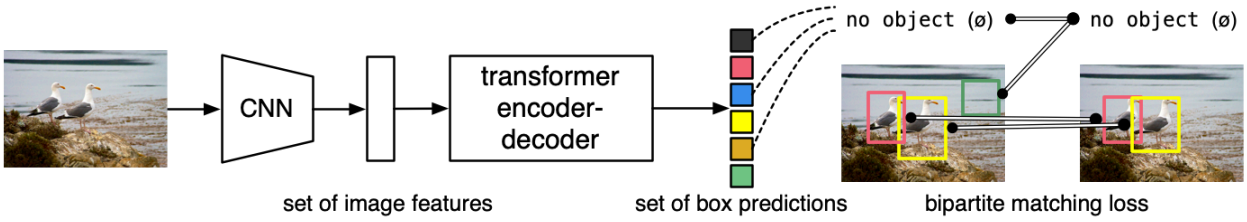
Detection Transformers (DETR)

Detection Transformer (DETR) is an object detection model developed by Facebook AI that introduces a novel approach by leveraging transformers, originally designed for natural language processing, to directly predict object locations and categories in images. Unlike traditional object detectors that rely on complex hand-crafted pipelines with region proposals and non-maximum suppression , DETR simplifies the process by treating object detection as a direct set prediction problem. It uses a convolutional backbone to extract image features, which are then passed to a transformer encoder-decoder architecture. The decoder outputs a fixed number of object queries, each responsible for predicting a single object or no object at all.



1. **Feature Extraction using Convolutional Backbone:** The DETR pipeline begins with a convolutional neural network backbone, which processes the input image and extracts dense feature maps. These feature maps capture semantic information about objects and textures in the image at a lower spatial resolution. To make this data suitable for the transformer architecture, the feature map is flattened into a sequence of feature vectors and supplemented with positional encodings. Since transformers lack inherent

knowledge of spatial structure, positional encodings are essential for allowing the model to understand the position of features in the original 2D image space.

2. **Transformer Encoder for Global Contextualization:** Once the feature vectors are prepared, they are passed through the transformer encoder. The encoder consists of multiple layers of self-attention and feed-forward networks, enabling the model to capture long-range dependencies across the entire image. Unlike traditional object detectors that rely on local features and region proposals, DETR's encoder can model global relationships between different parts of the image, allowing it to reason about object context holistically.

3. **Decoder with Object Queries for Detection:** Parallel to the encoder, DETR introduces a fixed-size set of learned object queries, which are vectors that serve as object detection slots. These queries are passed through the transformer decoder, where they interact with the encoder output via cross-attention. Each query learns to focus on different regions or aspects of the image and is responsible for predicting a single object. The output from the decoder is then passed through simple feed-forward layers to predict both the class label and the normalized bounding box coordinates for each query.

4. **Set-Based Prediction and Hungarian Matching:** DETR formulates object detection as a set prediction problem, treating detection as selecting a fixed set of object outputs, rather than generating variable numbers of detections through anchor boxes or proposals. During training, a bipartite matching algorithm (the Hungarian algorithm) is used to find the optimal one-to-one assignment between the model's predicted objects and the ground truth annotations. This matching ensures that each object is predicted only once and avoids the redundancy seen in traditional detectors.

Applications of DETR

- **Autonomous Driving:** DETR is used to detect vehicles, pedestrians, and traffic signs in real-time driving scenarios.

- **Remote Sensing:** In satellite and drone imagery, DETR detects buildings, roads, and vehicles with high accuracy.

- **Medical Imaging:** DETR can identify tumors, organs, or anomalies in X-rays and scans. Its end-to-end design enables precise detection without handcrafted features, useful in subtle or irregular patterns.

- **Robotics and Automation:** DETR helps robots detect tools, components, or defects on assembly lines. Its one-to-one prediction style ensures accurate object counting and placement.

Implementation Summary

1. **Setting Up the Environment and Dependencies**
   The implementation begins by installing required dependencies, primarily using the transformers and torch libraries. The transformers library provides prebuilt utilities to load DETR models, and the training utilities are configured using PyTorch and related tools.

2. **Loading the Pretrained Model and Processor**

   The pretrained DETR model and corresponding image processor are loaded using the Hugging Face Transformers library. The processor handles image transformations like resizing, normalization, and padding, ensuring input images match the expected format of the model. The model is moved to the GPU to take advantage of accelerated computation during both training and inference phases.

3. **Preparing the Custom Dataset for Training**

   A custom object detection dataset in COCO format is used for training. Images and corresponding annotation files are structured to follow the standard COCO format, including bounding boxes and class labels. The dataset is loaded and split into training and evaluation sets using PyTorch's datasets library. Each image is processed and augmented for model input using the preloaded image processor.

4. **Defining the Training Pipeline and Hyperparameters**

   A training loop is defined using the Trainer class from Hugging Face, which wraps the model, optimizer, data loaders, and training arguments. Hyperparameters such as learning rate, batch size, number of epochs, and evaluation intervals are configured to control the training process. The loss function combines classification and bounding box regression losses for end-to-end optimization.

5. **Training and Evaluating the Model**

   The model is trained using the prepared dataset, with periodic evaluation on the validation split. During evaluation, metrics such as mean Average Precision are computed to measure object detection performance. The model outputs bounding boxes and class labels for each object query, which are compared against ground truth annotations.

6. **Saving the Trained Model and Processor**

   After training, the fine-tuned DETR model and its image processor are saved locally. This allows for the model to be reused later for inference on new images or further fine-tuned on additional data without retraining from scratch.

Major Findings & Conclusion

- **DETR Delivers Fast and Accurate Object Detection**

  The pretrained DETR model achieves high detection accuracy while maintaining real-time performance, making it well-suited for time-sensitive applications such as robotics, surveillance, and autonomous systems

- **Custom Dataset Integration with Minimal Overhead**

  The model adapts effectively to custom datasets formatted in COCO style. With proper preprocessing and a well-defined configuration, DETR trains efficiently and produces precise bounding box predictions tailored to the target dataset.

- **Reusable for Inference and Further Fine-Tuning**

  The trained DETR model and processor can be saved and reused for future inference tasks or continued training. This enables scalable deployment and flexibility in adapting to new object detection scenarios.

References:

1. https://github.com/facebookresearch/detr/tree/main
2. https://www.geeksforgeeks.org/object-detection-with-detection-transformer-dert-by-facebook/
3. https://github.com/roboflow/notebooks/blob/main/notebooks/train-rt-detr-on-custom-dataset-with-transformers.ipynb