VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Integration Project

Project Report

Graphical Object Detection From Document

Advisor(s): Tran Tuan Anh, HCMUT
Mai Xuan Toan, HCMUT
Tran Hong Tai, HCMUT

Student(s):	Nguyen Trinh Chau	2252091
	Trinh Tri Bao	2252077
	Van Duy Anh	2252045

HO CHI MINH CITY, JANUARY 2025

Contents

List of Figures

1 Introduction

1.1 Motivation

In today's digital era, an enormous amount of information is stored in document form, including scanned images, PDFs, and digital reports. These documents often contain a mix of textual and graphical elements such as tables, charts, diagrams, and images. Extracting meaningful information from these documents is a very important task in fields such as education, research, business, and legal compliance. However, detecting and analyzing graphical objects within these documents is such a challenge because of the following reasons:

- Diverse Document Formats: Documents come in various formats and layouts, making it difficult to develop a one-size-fits-all solution.
- Complex Structures: Graphical objects are often embedded alongside text, with varying sizes, resolutions, and orientations.
- Manual Processing Limitations: Relying on human effort to manually identify and extract graphical content is time consuming and inefficient, especially when dealing with large-scale document collections.

Given these challenges, there is a clear need for an automated system capable of accurately detecting and extracting graphical objects from documents, which will provide several benefits:

- Efficiency: Automation significantly reduces the time and effort required to process large volumes of documents.
- Accuracy: Object detection models are capable of achieving high precision, minimizing errors associated with manual extraction.
- Scalability: Automated solutions can handle a large amount of documents, making them suitable for enterprise-level applications.

To address these needs, this project utilizes the DEtection TRansformer (DETR) model. DETR is a new approach to object detection, it is the result of the Facebook Researcher Team. DETR leverages the power of transformers to directly predict bounding boxes and class labels for objects in an image, eliminating the need for traditional region proposal methods. Although DETR does not have a really impressive performance, it is

a promising approach and can be further developed to be state-of-the-art model with highest precision.

1.2 Detection Transformer(DETR)

Transformers are a deep learning architecture that has gained popularity in recent years. They rely on a simple yet powerful mechanism called attention, which enables AI models to selectively focus on certain parts of their input and thus reason more effectively. Transformers have been widely applied on problems with sequential data, in particular in natural language processing (NLP) tasks such as language modeling and machine translation, and have also been extended to tasks as diverse as speech recognition, symbolic mathematics, and reinforcement learning. But, perhaps surprisingly, computer vision has not yet been swept up by the Transformer revolution.

By proposing a direct set prediction approach, DETR bypass the surrogate tasks such as anchor proposal or window center of other detection models. This led to a simple structure and an end-to-end training process.

Differences between DETR and other object detection models

DETR and Faster-RCNN

- Advantages of DETR:
  - End-to-End Framework: Unlike Faster R-CNN, which uses a two-stage approach (region proposal followed by classification), DETR is fully end-to-end, simplifying the training and inference pipelines.
  - Global Context Understanding: DETR uses transformers, which excel at capturing global context in images, making it effective for complex scenes.
  - No Need for Non-Maximum Suppression (NMS): DETR eliminates the need for NMS, as it avoids redundant predictions by design.
- Disadvantages of DETR:
  - Training Time: DETR requires longer training times due to the complexity of transformers and the bipartite matching process.
  - Small Object Detection: DETR struggles with detecting small objects compared to Faster R-CNN, which excels in this area due to its region proposal network.

DETR and YOLO

- Advantages of DETR:
  - High Accuracy: DETR often achieves higher precision on complex datasets due to its transformer-based architecture.
  - Flexible Output: DETR's direct prediction of bounding boxes allows it to handle irregular object shapes better than YOLO
- Disadvantages of DETR:
  - Inference Speed: YOLO is significantly faster at inference, making it more suitable for real-time applications.
  - Hardware Requirements: DETR's transformer architecture demands more computational resources compared to the lightweight YOLO models.

1.3 Requirements

The project aims to use DETR model for detecting and localizing graphical objects. Then classifying them as tables, figure, graph, and equation from various document formats. The solution will face challenges in processing mixed-content documents and provide efficient and accurate information detection.

Input

- Formats: Scanned images (JPEG, PNG). For futher improving, we try to detect from a PDF file, video, etc.
- Content: Mixed-content between text and graphical elements (tables, charts, diagrams, etc.)

Ouput: JSON file containing object coordinates and types, which can then be visualized as:

- Graphical objects detected and localized within the document.
- Class it belongs to and the score for the prediction.

  Labels specifying object types (e.g., table, chart, image).

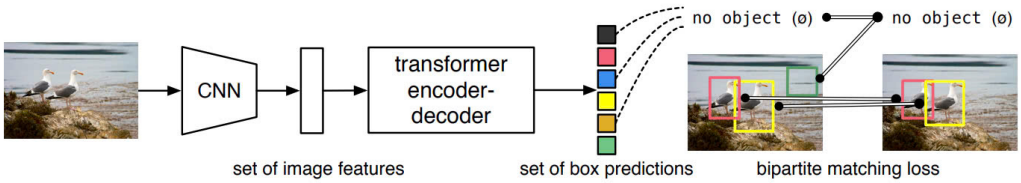


Figure 2.1: Image of the architecture

2 Detection Transformer (DETR) Model

2.1 DETR Architecture

Overall, first the image go through a CNN backbone, in this case ResNet50, to extract features. Then it is flatten and go through a transformer encoder-decoder, where all the magic comes from. The outputs of transformer are our predictions, the pairs of classification and bounding box. These pairs are then mapped to the ground truth bounding boxes using bipartile matching loss to calculate the loss. We will go through each module step by step.

2.2 Convolutional neural network (CNN)

As with RCNN or YOLO, DETR also leverages CNN to extract the image features. Because CNN works efficiently with image, it extracts the feature without losing the geometric shape of the image. In this model, we use ResNet-50, a popular architecture with well performance. In ResNet50, by using Residual Blocks that allow for the direct flow of information through the skip connections, it solved the vanishing gradient problem. ResNet50 also benefits from extensive pretraining on large datasets like ImageNet, providing a strong starting point for fine-tuning in specific tasks.

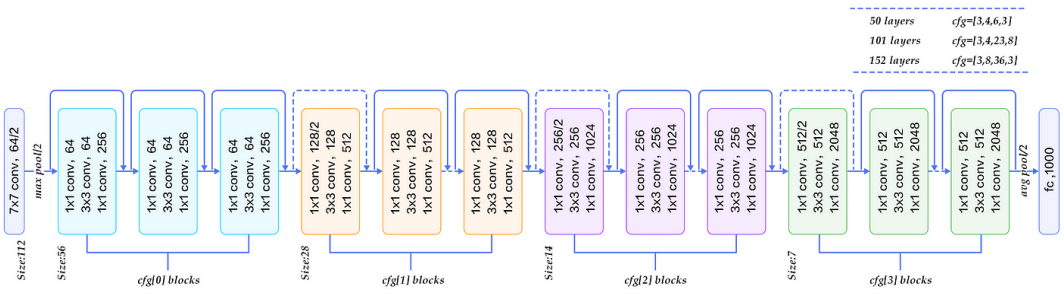


Figure 2.2: Architecture of ResNet50

The output of Backbone CNN will then be fed into Transformer Encoder Decoder.

2.3 Transformer Encoder-Decoder

2.3.1 Encoder

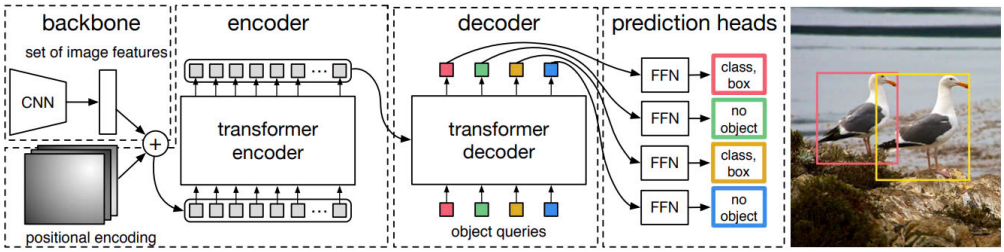


Figure 2.3: Architecture of DETR

After running through the backbone CNN, the image features are then flatten and combined with the positional encoding. Because when we flatten it out, all the information about spatial structure is lost, so positional encoding add information about position.

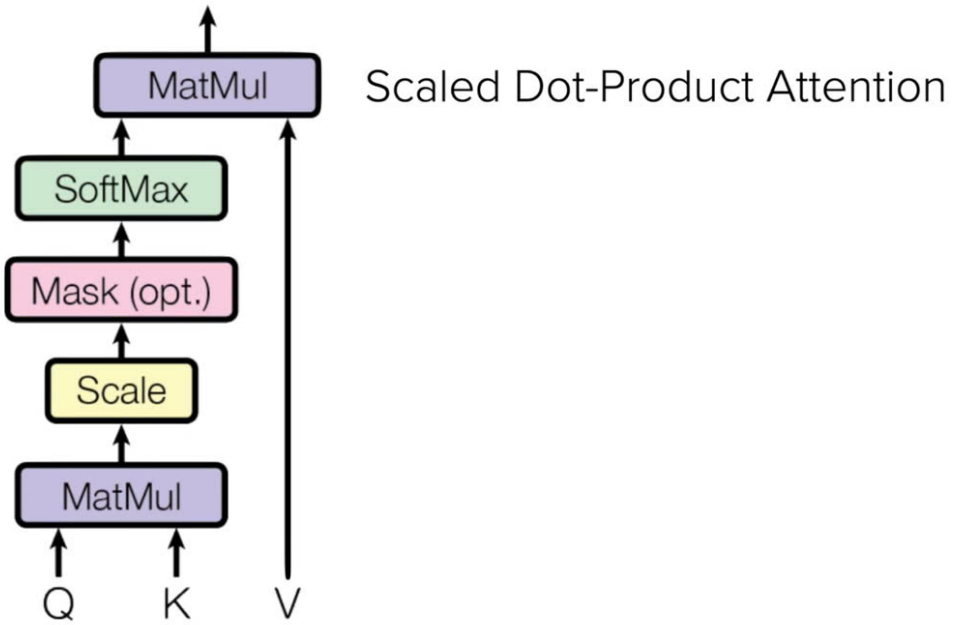


Figure 2.4: Encoder demonstration

First we apply dot product on the Query weight Wq and the image to get Q. We do similar thing to get K. Then we calculate the attention mask to get the relationship between regions in the image. After that, we put it through softmax and add it with the value V to represent the affection of different parts of the image to each other.

2.3.2 Decoder

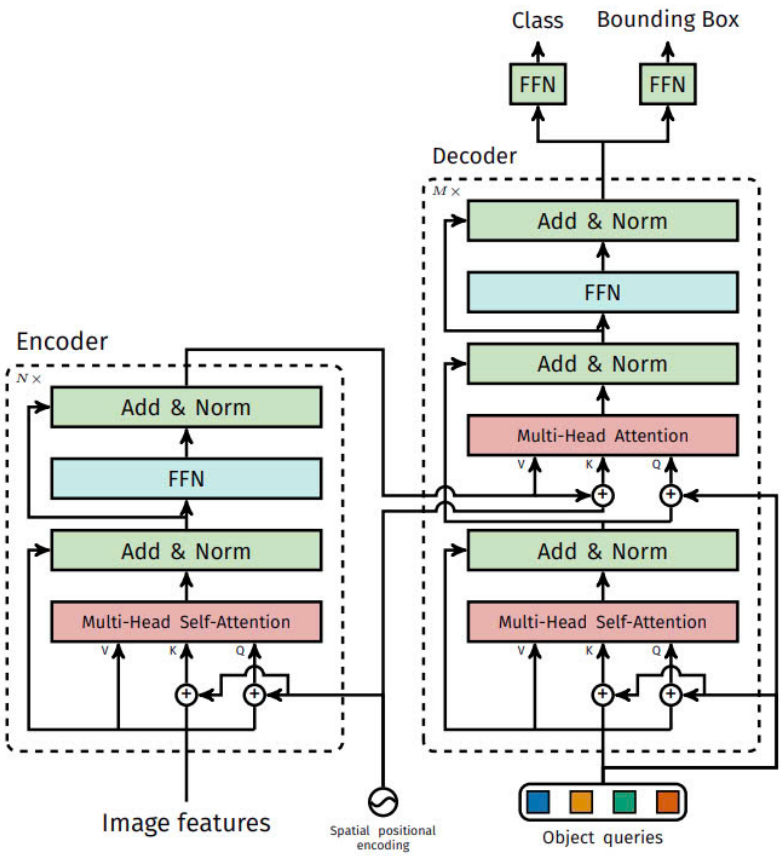


Figure 2.5: Decoder demonstration

The output of the encoder then go as a side input of decoder. The decoder in DETR is responsible for generating object predictions (bounding boxes and class labels) based on the features processed by the encoder. The decoder is fed with a fixed number of learnable object queries. Each query is a positional embedding initialized randomly and represents a potential object. In DETR, the number of object queries is 100. The decoder applies cross-attention between the object queries and the encoder's output features. The output of decoder is then go through a Feed-Foward Network (FFN) to generate the class label and bounding box coordinate.

2.4 Loss Function

In order to calculate the loss, DETR uses the bipartite matching loss with Hungarian matching algorithm. DETR employs the Hungarian algorithm to find the optimal one-to-

one matching between predictions and ground truth objects.

First, because the prediction is a set, we need find a way to map the prediction with the ground truth. If we don't, the order of the prediction will affect the loss. To solve that, we use Hungarian matching algorithm. The overall goal of this is to find the bipartite mapping with min loss.

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

The matching cost takes into account both the class prediction and the similarity of predicted and ground-truth boxes. Each element i of the ground truth set can be seen as a yi = (ci, bi) where ci is the target class label bi is a vector that defines ground truth box.

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

The loss of classification can be calculated using negative log-likelihood. And for bounding box loss, we use a linear combination of the '1 loss and the generalized IoU loss.

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

Figure 2.6: Hungatian Loss

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

Figure 2.7: Bounding box loss

This loss provide all the model needs:

- Set prediction: the prediction is a set and the order does not affect the overall model
- Penalize phantom objects: As we can only map the prediction to ground truth objects or no object class, it is capable for penalizing wrong-predicted objected.

- Penalize duplicate object: as the number of ground-truth object to be mapped is predefined. Duplicate predictions can not be mapped to the same object and will be penalized. So we do not need non-maxima.
- Singe-stage Prediction: as the output of the model is the class and bounding box, we do not need any post processing or hand-crafted module. We simply feed in an image and get what we want.

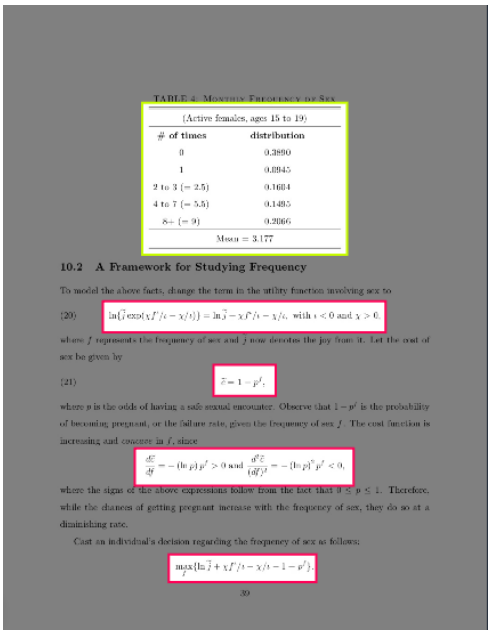
3 Implementation

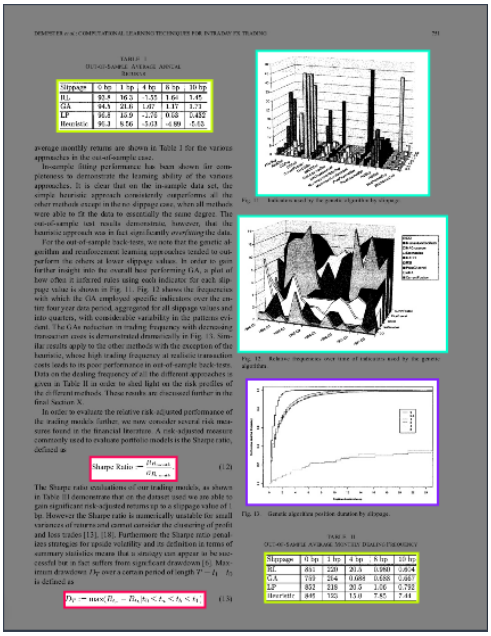
3.1 Data Labeling

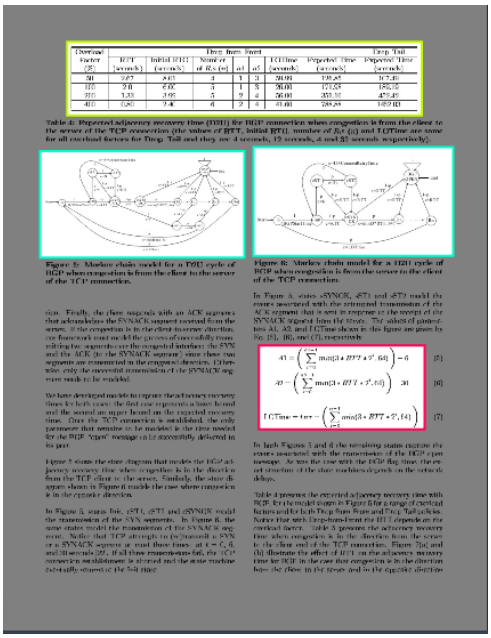
In this project, we use ICDAR-POD2017 to train and test, because this dataset includes a large variety in both layout and object styles like single-column, double-column, multi-column and various kinds of equations, tables and figures. Moreover, there are many projects with the same topics that used this dataset to train and test so we have lots of data to compare with our results to show how effective our project is. We consider to use around 800 images in this dataset (with about 550 images for training, 150 images for validation and the 100 images for testing).

We choose a tool which is Roboflow to label our dataset due to its user-friendly interface, powerful features, and seamless integration with machine learning workflows. As a cloud-based platform, it eliminates the need for local installations and simplifies the process of uploading, organizing, and annotating datasets. The most critical feature is that roboflow allows us to export dataset in COCO format, which we need to use to implement the training model for our dataset. Additionally, its API enables seamless integration into training pipelines and allows us to directly pull our dataset into our model.

In our model, we detect 4 main styles of objects: table, graph, equation, figure and we label our dataset by drawing bounding boxes of those graphical objects on all document images







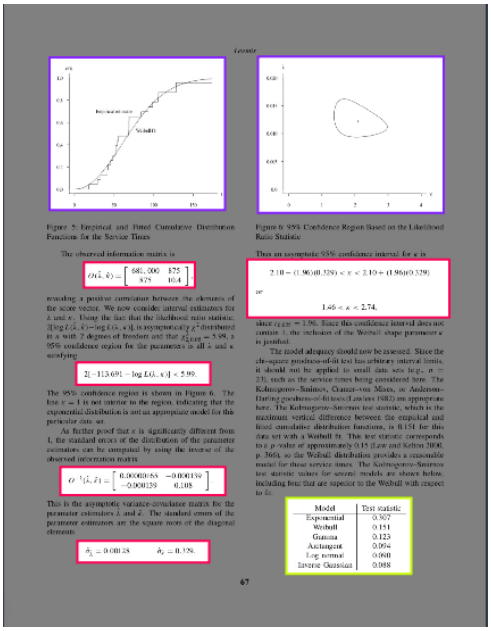


Figure 3.2: Data Labeling

3.2 Training model

Training an Object Detection Transformer (DETR) model involves several key steps and concepts, each build up a combination of modern transformer-based techniques and traditional object detection methods. At its core, DETR is a groundbreaking approach that combines the power of transformers with object detection tasks, utilizing the principles of attention mechanisms to handle the spatial and contextual relationships of objects within images. The processes of training DETR involve data preparation, setting up the model architecture, optimizing the learning process, and evaluating its performance on real-world datasets. The first crucial step in training DETR is preparing the dataset including labeled images in COCO-style annotations format, which we already prepared using roboflow as described in the labeling section. This annotated dataset is then loaded into the training pipeline for processing and model input.

COCO format organizes annotations in a structured JSON file, making it easy to use and highly compatible with popular machine learning frameworks like TensorFlow and PyTorch. The JSON file typically contains several key sections, including metadata about the dataset under the "info" field, information about the images under the "images" field, annotation details in the "annotations" field, and the list of categories or labels in the "categories" field. For object detection tasks, the COCO format uses bounding boxes defined as [x, y, width, height], where (x, y) represents the top-left corner of the box. For segmentation tasks, it supports both polygonal masks and Run-Length Encoding (RLE) masks, which are efficient representations of pixel-level object boundaries. The annotations also include the area of the object, typically in square pixels, and an "iscrowd" flag to indicate whether the annotation represents a single object or a group of objects.

The next step involves setting up the DETR model architecture, which is fundamentally different from traditional convolutional neural network (CNN)-based object detection models. DETR's core innovation is its use of transformers, which were initially designed for natural language processing tasks but have shown exceptional performance in handling sequential data. In DETR, a transformer model is used to capture the relationships between different regions of an image. The transformer decoder attends to the spatial features extracted by the backbone network (usually a CNN like ResNet or Vision Transformer) and generates object predictions based on these features. Unlike traditional object detection models that rely on anchor boxes and post-processing techniques like Non-Maximum Suppression (NMS), DETR directly predicts the final bounding boxes and class labels through a set of learned object queries, making the model more flexible and easier to train on diverse datasets.

Once the model architecture is in place, the training process begins. The goal of training a DETR model is to minimize the prediction loss, which is a combination of classification loss (for correctly identifying object classes) and bounding box loss (for accurately locating objects within the image). DETR's unique approach involves the use of a Hungarian algorithm to match predicted bounding boxes to ground truth objects during the training process. This matching ensures that the network learns to associate predicted objects with their true counterparts in the image, which is essential for accurate object detection. The Hungarian algorithm works by computing a cost matrix that measures the discrepancy between the predicted boxes and the ground truth boxes, and it then assigns the best matching predictions based on the lowest cost. This algorithm eliminates the need for complex anchor box generation or post-processing steps like NMS, which are common in other object detection models.

During training, the model is fed batches of images, and the output of the transformer model consists of a set of predictions, including bounding box coordinates and class labels for each object. These predictions are compared to the ground truth annotations, and the difference between the predicted and true values is used to compute the loss. The loss is back propagated through the network, and the weights of the transformer model are optimized. Because of the limitation of Google Colab, we only train the model in 100 epochs. However, the result is acceptable.

An important aspect of training DETR is fine-tuning. Given that DETR is a large and complex model, training from scratch on a small dataset might be inefficient. Therefore, we decided to fine-tune a pre-trained DETR model on our dataset. We load a pre-trained model of Facebook Researcher, which has already learned generic object detection features, and then train it further on the new dataset to adapt the model to the particular object classes and characteristics present in the new data. Fine-tuning speeds up convergence and leads to better performance, especially when the dataset is relatively small or has limited variations from the model's pre-trained data.

As the model trains, it is essential to periodically assess its performance using a validation set that is separate from the training set. This helps determine whether the model is overfitting to the training data or if it is learning meaningful patterns that generalize to unseen examples. In addition to evaluating the loss function, performance metrics like mean Average Precision (mAP) are often used to quantify the model's ability to detect objects accurately. mAP measures the precision and recall of object detection predictions across different class labels and threshold values, providing an efficient evaluation of the model's detection capabilities.

After training, the model's performance is evaluated on a test set, which consists of images that were not part of the training or validation sets. This step is crucial to ensure that the model is capable of generalizing to new, unseen data. During testing, the model performs inference by making predictions on each image in the test set, generating bounding boxes and class labels for detected objects. These predictions are then compared to the ground truth annotations to calculate the final evaluation metrics, such as mAP, precision, recall, and F1-score.

3.3 Result and Evaluation

3.3.1 Result

After training, the model was able to detect and localize graphical objects pretty well. We feed some random image to the model and it successfully detect the graphical objects.

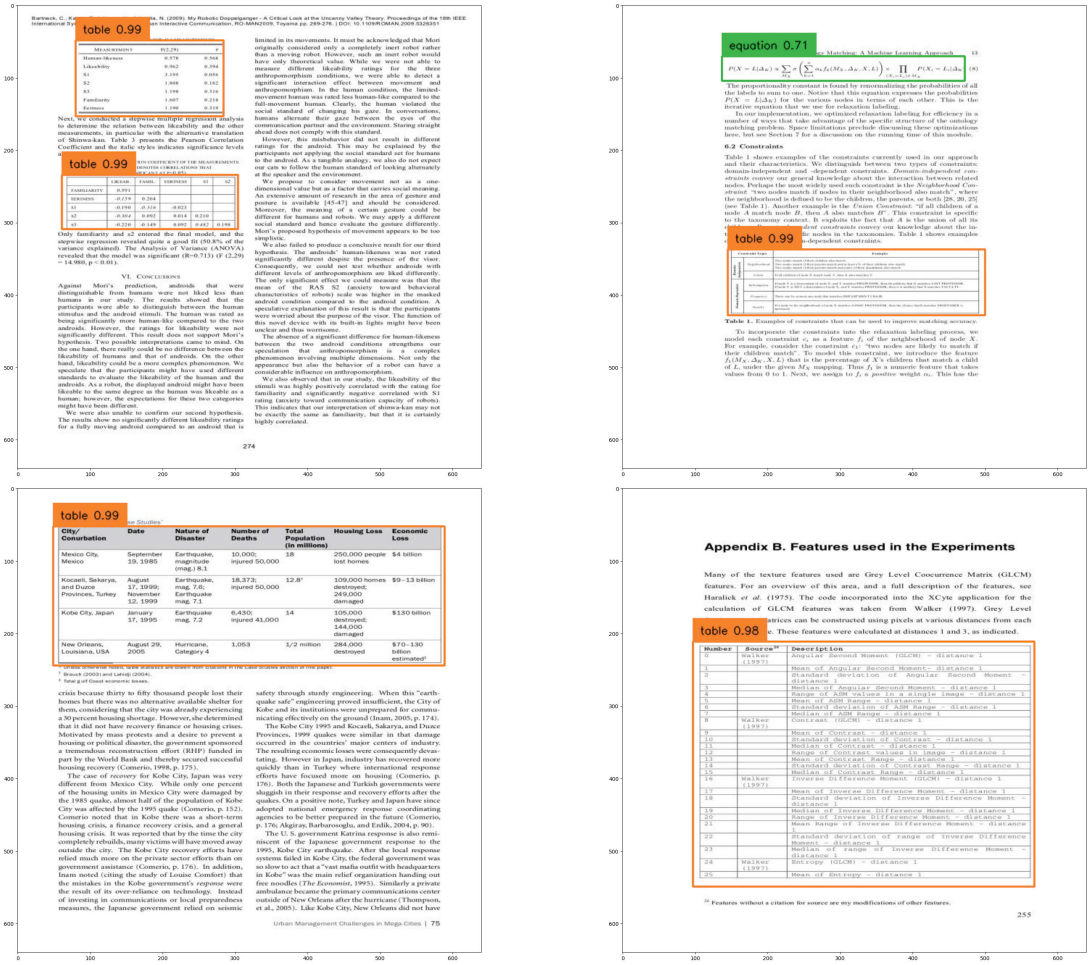


Figure 3.3: Result

3.3.2 Evaluation

After training, we run the evaluation on the test set, which give the result. The Average

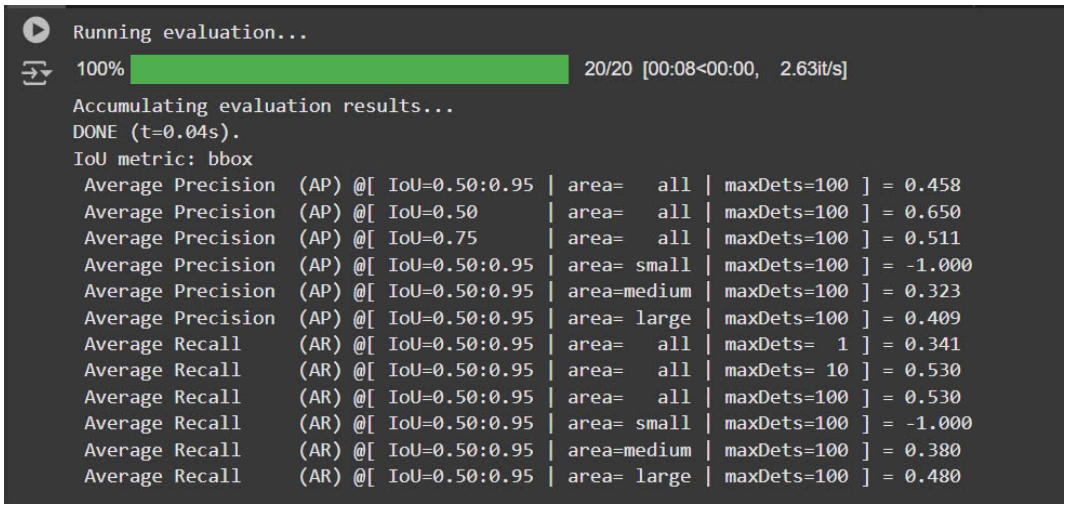


Figure 3.4: Evaluation

Precision (AP) for IoU thresholds ranging from 0.50 to 0.95 is 0.458, indicating moderate performance in detecting objects across all sizes and categories. The precision increases to 0.650 for IoU = 0.50, showing better performance for loose IoU thresholds.

Moreover, the model performs better with large object than medium and small object. In the dataset, there is no small object so the evaluation of this category is unknown.

For Average Recall, the Average Recall (AR) across all IoU thresholds is 0.341 for a single detection and improves to 0.530 for a maximum of 10 detections. For large objects, the AR is 0.480, whereas medium and small objects show poor recall.

3.4 Comparison

In compare with other model, our model does not perform well enough. In another model, Graphical Object Detection (GOD) which use Faster-RCNN. However, this model

IoU	Test Performance: AP			mAP	Test Performance: F1			Ave F1
	Eqn.	Table	Figure		Eqn.	Table	Figure	
0.5	0.922	0.980	0.881	0.928	0.922	0.977	0.851	0.917
0.6	0.903	0.962	0.851	0.905	0.889	0.959	0.878	0.909
0.7	0.810	0.937	0.815	0.854	0.832	0.952	0.833	0.872
0.8	0.742	0.916	0.786	0.814	0.772	0.927	0.833	0.844

Figure 3.5: Faster-RCNN Performance

is carefully fine-tuned on a much larger dataset. Due to the lack of time and resources, we can not do the same thing on our model, which causes our model to not have a competitive performance with GOD model.

4 Further improvement

For further improvement, we plan to do the following things, with the priority order:

- Fine-Tune with Larger Dataset: we want to label and fine-tune our model on a larger or more diverse dataset (e.g ICDAR-2013, Marmot table recognition, UNLV data set) to improve the model's performance. We also plan to do Data Augmentation on our existing data such as rotation, scaling, cropping, or color adjustments to simulate real-world scenarios.
- More training: Because of the limitation of Google Colab, we can only train our model for 100 epochs or less. We try to find another platform to train the model more efficiently.
- Model Architecture Enhancement: We want to use a hybrid approach like Co-DETR, which is the state-of-the-art model for accurate object detection.
- User interface: develop an interactive interface where users can use our model to make predictions, and to feedback for model retraining.

5 Conclusion

This is the end of our report. In this project, we used DETR model to detect graphical object from document. First we did some research on the potential model to solve the problem. Then after choosing Detection Transformer, we prepare our dataset, train the model and evaluating it. Thank you for reading.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *European Conference on Computer Vision (ECCV)*, 2020.

[2] Alexey Bochkovskiy Chien-Yao Wang and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of the-art for real-time object detectors. 2022.

[3] Benenson R. Schiele B Hosang, J.H. Learning non-maximum suppression. *CVPR*, 2017.

[4] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once: Unified, real-time object detection. 2016.

[5] Ajoy Mondal Ranajit Saha and C V Jawahar. Graphical object detection in document images. *ICDAR 2019*, 2020.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.