Bachelor Project
Spring Semester 2025

Deep Learning-based Table Extraction from Pharmaceutical PDFs

Group members:
Vladyslav Horbatenko, vladyslav@ruc.dk, 76508
Salar Komeyshi, salar@ruc.dk, 76346

Supervisor:
Henning Christiansen, henning@ruc.dk

Roskilde University
Monday 2nd June, 2025

Abstract

In the pharmaceutical industry, critical laboratory data is often locked in unstructured PDF reports, making manual extraction laborious and error-prone. This project develops an automated framework for detecting and parsing tables from pharmaceutical PDFs using transformer-based models. First, DETR is employed to identify table regions on rendered PDF pages. Next, each detected table is processed by TATR to recognize its internal structure, including rows, columns, headers, and spanning cells. We curated and annotated a dataset of 1,500 pages reflecting diverse lab report formats from Novo Nordisk and fine-tuned both models on this domain-specific corpus. After fine-tuning, DETR achieved an average precision of 0.936 for table detection, and TATR reached a GriTSTop score of 0.802 for structure recognition. These results demonstrate improved accuracy over baseline and robustness to domain-specific formatting variations. The automated pipeline reduces manual processing time and enhances data consistency and fidelity. Ultimately, this work establishes a foundation for scalable extraction of structured data from unstructured documents, supporting clinical research and regulatory reporting.

1 Introduction

In the digital age, vast amounts of data are stored in various formats, with PDFs being one of the most widely used for document sharing and distribution. PDFs are highly versatile in terms of formatting, making them an excellent choice for professional and scientific documentation. However, this flexibility also introduces challenges when trying to automate the data extraction, especially when the data is embedded in tables. Extracting information from PDF tables is crucial for industries that rely on structured data, such as healthcare, finance, and research.

In recent years, deep learning and computer vision techniques have shown great promise in automating the extraction of information from images and documents. However, the ability to accurately detect, extract, and structure data from tables in PDFs remains a difficult challenge due to the variability in table formats and layouts across documents.

A common task for many industries is extracting and standardizing data from diverse sources. This is especially crucial when the data needs to be transferred into structured databases or applications. For companies in fields like pharmaceuticals, where accurate data extraction is essential for compliance, research, and analysis, the problem of dealing with unstructured document formats is even more pressing.

This project, in collaboration with Novo Nordisk, aims to address these challenges by creating a deep learning-based model that can automatically identify and extract tables from PDFs, with a specific focus on handling diverse and complex table structures. By leveraging advanced techniques like DETR (Detection Transformer) and TATR (Table Transformer), we aim to automate the extraction of tabular data from medical lab reports, significantly improving efficiency and accuracy in Novo Nordisk's workflows.

1.1 Motivation and Context

Novo Nordisk, a global healthcare company specializing in the development of medicines for chronic diseases, is currently facing a significant challenge in handling and processing medical lab results that come in the form of PDFs. These documents, which originate from various labs worldwide, often contain tabular data, a key source of information. However, the formatting of these tables varies greatly from one document to another. The lack of a standardized structure means that the data contained within these tables is not immediately accessible or usable for processing. This creates a major bottleneck in the workflow.

Currently, Novo Nordisk receives medical reports from labs worldwide. Although these reports generally contain the same types of data (test results, numerical values, tables), they are often formatted differently. This variation makes it difficult to automatically extract meaningful information from these PDFs. Currently, the process of transferring data from PDFs into Novo Nordisk's internal systems is highly manual. Employees must spend considerable time interpreting, extracting, and standardizing the data, especially when it comes to numerical values in tables that may be formatted inconsistently (varying decimal places, text representations of values). This not only leads to delays but also increases the risk of human error.

The core of the issue lies in the tables embedded in these documents. Tables are often used to present structured data in a human-readable format. However, because each lab or institution may have its own conventions for creating tables, the appearance, alignment, and even the semantic meaning of each table can differ. As a result, traditional methods of document parsing struggle to recognize tables, let alone interpret their contents in a meaningful way. This issue is made even more complicated by the lack of standardization. The lab results that are received by Novo Nordisk do not follow any standardization, and it is often an added task for its employees.

Given these challenges, the need for a robust, automated solution that can handle the extraction, recognition, and standardization of tabular data from diverse PDFs is clear. The main goal of this project is to develop a deep learning-based model that can extract tables from PDFs, recognize their internal structures (rows, columns, cells) according to Novo Nordisk's data domain.

1.2 Research Question

How can deep learning-based models, specifically DETR (Detection Transformer) and TATR (Table Transformer), be adapted and trained to accurately detect and extract tables from unstructured PDF documents at Novo Nordisk?

1. How can the DETR-based approach be used for table extraction?
2. How well does the TATR model perform on unseen Novo Nordisk data?
3. Can a rule-based approach achieve better results than DL specific models?
4. What are the ethical ramifications of adopting such AI model in Novo Nordisk?

2 Background & Related Work

2.1 PDF

The Portable Document Format (PDF) is a file format developed by Adobe Systems in the early 1990s to present documents consistently across different devices, software, and operating systems. According to Adobe's official documentation [1], the primary goal of PDF was to create a universal file format that preserved fonts, images, graphics, and layout of source documents, independent of the application or platform used to create them.

PDF files are fundamentally structured as a collection of objects, organized into four main components: a header, a body, a cross-reference table (xref), and a trailer. The header identifies the file as a PDF and specifies the version. The body contains all the content, such as text streams, images, fonts, and graphic objects, typically described in a page description language. The cross-reference table allows efficient random access to these objects, and the trailer provides information about the structure of the file, including references to the xref table and the document's metadata [1].

PDF files, while visually similar, can differ significantly in internal structure, encoding, and metadata depending on the software used to create them, such as LaTeX, Acrobat, Microsoft Word, or Google Docs. LaTeX-generated PDFs often rely on precise positioning, vector graphics, and custom font embedding [2], whereas Word and Google Docs focus on WYSIWYG layouts and may include hidden XML structures or accessibility tags. These differences affect the ease of information extraction, with LaTeX PDFs potentially lacking structured tags and Word/Docs PDFs introducing redundant or inconsistent formatting. Metadata, which aids in document management and reflects the platform of origin, also varies. LaTeX often includes minimal metadata unless added manually, while Word and Acrobat embed rich metadata by default [1], [2]. Understanding these structural and encoding differences is crucial for effective automated PDF processing.

Tables in PDF

In the PDF files, tables are not treated as a distinct object type. Instead, tables are visually constructed using a combination of low-level graphics and text elements. Typically, what appears to be a table to a human reader is composed of individually drawn lines (for rows and columns) and separately positioned text elements for the content of each cell. Because of this, tables in PDFs are fundamentally layout-based rather than semantically structured. There is no inherent "table" tag or object unless the file is specifically tagged for accessibility (as in PDF/UA standards), which is often not the case for general documents [1, p. 936].

When generating a PDF, a table is usually constructed by placing text at precise coordinates on a page. The horizontal and vertical alignments simulate the appearance of rows and columns. Some programs, such as LaTeX, define tables mathematically with exact spacing and rules, producing highly regular and precisely aligned structures [2]. In contrast, programs like Microsoft Word may introduce slight inconsistencies in cell alignment or introduce floating elements that affect parsing. Furthermore, line separators are often independent graphical objects rather than semantically tied to the table data. In

some cases, tables may even be represented as a single embedded image with no textual information preserved at all, making extraction much more complex, especially with rule-based approaches.

Encoding practices further complicate table recognition. For instance, text within a table cell may be split into multiple fragments if font changes, kerning adjustments, or hyphenation[1] occur. Additionally, spacing between words or numbers might be represented inconsistently depending on how the generating software interprets white space. Without explicit structural tags, table reconstruction relies heavily on heuristic or machine learning approaches to infer row and column boundaries based on spatial relationships rather than logical document structure. As a result, even visually identical tables may have radically different underlying encodings depending on the source platform.

2.2 Artificial Intelligence and Machine Learning

This project understands Artificial Intelligence (AI) as a technology and concept that broadly refers to computational systems designed to perform tasks that traditionally require human intelligence, such as decision-making, pattern recognition, and problem-solving. Within AI, machine learning (ML) has emerged as a dominant paradigm, focusing on algorithms that enable machines to learn patterns and make predictions by being exposed to data, rather than by relying solely on explicit, hand-coded rules [3].

In traditional machine learning, constructing systems capable of handling natural data, such as images, text, or audio, required substantial manual effort. Engineers would design "feature extractors" to convert raw input into internal representations suitable for classification or prediction. However, this process depended heavily on human expertise and domain-specific knowledge.

The rise of deep learning introduced a shift. Instead of manually crafting features, machines could learn useful representations directly from raw data. Deep learning builds upon this idea by employing multi-layered architectures that automatically learn hierarchies of increasingly abstract features.

[1] Kerning adjustments refer to the adjusting of spacing between specific pairs of characters to achieve visually better letter spacing, while hyphenation is the process of breaking words at syllable boundaries with a hyphen at the end of a line to improve text and avoid large gaps.

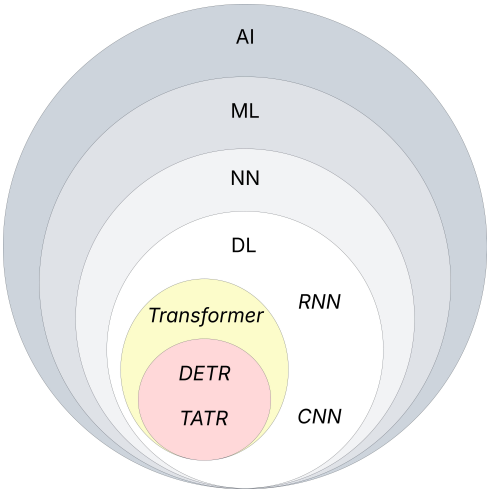

Figure 1: A diagram showcasing the project's understanding of the relationship between AI and its different architectures

2.3 Deep Learning and Neural Networks

The project will define deep learning or DL as computational models that consist of multiple processing layers, allowing them to learn and represent data at various levels of abstraction. This hierarchical structure allows the models to progressively extract and understand increasingly complex features from the input data [3].

In other words, DL employs deep neural networks (DNN), where layers of simple non-linear modules transform representations progressively from raw input toward a desired output. Early layers typically capture basic structures, such as edges in images, while deeper layers compose these into more complex patterns, such as object parts, and eventually complete objects [3].

At the core of these networks are artificial neurons. Each neuron computes a weighted sum of its inputs, followed by a non-linear activation function, such as a Rectified Linear Unit (ReLU). The training of a network involves adjusting the weights via backpropagation, an application of the chain rule of calculus that computes gradients of an objective (loss) function [3].

Neural networks can be broadly divided into several architectures depending on the data and the task [3]–[5]:

- **Feedforward Neural Networks (FNNs):** Data flows only forward, from input to output. Common in classification tasks.
- **Recurrent Neural Networks (RNNs):** Incorporate loops to allow information persistence, ideal for sequential data like text or speech.
- **Convolutional Neural Networks (CNNs):** Specialized for spatial data like images, exploiting local connections and shared weights to detect spatial hierarchies.
- **Transformers**: Recent architectures relying on self-attention mechanisms to model relationships within data, excelling in both sequential and spatial domains.

Each architecture offers different strengths, depending on the task and data complexity.

Convolutional Neural Networks

CNNs have been particularly revolutionary in field of computer vision (CV). A CNN processes inputs with multiple 2D arrays (RGB channels of an image) through convolutional layers and pooling layers [3].

Convolutional layers apply learned filters, also known as kernels, to local regions of the input, helping to detect features such as edges, textures, and shapes. Pooling layers, on the other hand, reduce the spatial dimensions of the input, summarizing the detected features and making the representations more robust to small spatial variations.

These operations build hierarchies of features [4]: Edges $\rightarrow$ Motifs $\rightarrow$ Parts $\rightarrow$ Full objects.

CNNs are inherently translation-invariant due to their local receptive fields and weight sharing. As a result, they are highly effective in recognizing patterns across different parts of an image.

CNN-based architectures stack many convolutional and pooling layers, followed by fully connected layers for classification or regression tasks. Such networks have achieved state-of-the-art results across a wide range of applications, including object recognition, facial recognition, and medical imaging.

2.4 Object Detection

Object detection is a critical task in CV that involves identifying objects within an image and localizing them via bounding boxes. Early deep learning-based object detectors were built upon CNNs, leveraging their ability to extract spatial features from images [6].

Initial approaches like R-CNN introduced a two-stage pipeline: first proposing candidate regions, then classifying each region using CNN features. This was followed by improvements such as Fast R-CNN and Faster R-CNN [4], which integrated region proposal mechanisms and feature extraction into a more unified framework, significantly boosting speed and accuracy. Meanwhile, YOLO (You Only Look Once) family approached detection as a single regression problem, directly predicting bounding boxes and class probabilities from an image in one pass [7].

These models all rely heavily on CNNs to process the image, extract high-level features, and predict object classes and locations. The convolutional layers act as powerful feature encoders, transforming raw pixel data into abstract representations that are useful for detection tasks. Although RNNs and related architectures have made significant contributions in domains like language modeling and sequence prediction, they are generally not used for object detection, where spatial feature extraction remains paramount.

Transformers in Vision Tasks

While CNNs have long been the dominant architecture for vision tasks, recent advances have demonstrated the effectiveness of Transformer architectures [5], initially developed for natural language processing, in handling image data.

Transformers introduce a self-attention mechanism that allows models to weigh the importance of different input elements relative to each other. In the context of images, this means that a transformer can relate one part of the image to any other part, capturing global dependencies without being limited to the local receptive fields inherent to CNNs. This ability is particularly powerful for object detection tasks, where objects might occupy different spatial scales, and contextual relationships across the entire image can aid in accurate localization and classification [8].

Unlike convolutional layers, which process local neighborhoods and rely on stacking many layers to increase the receptive field, transformers can immediately model relationships between distant parts of the input. This results in a richer, more flexible feature representation that can better handle complex scenes and varied object layouts.

In practice, modern vision architectures often combine CNN backbones (for initial low-level extraction of a 2D feature representation from the input image) with transformer encoders and decoders (for modeling global relationships and performing detection), achieving strong performance across various benchmarks [8].

The Need for New Object Detection Approaches

While CNN-based detectors like Faster R-CNN rely on generating region proposals and applying classification on top of these proposals, this multi-step process is complex and often requires heuristic components like non-maximum suppression (NMS) to remove duplicate detections. These hand-designed elements and multi-stage pipelines can add complexity and training challenges. This inherent complexity and reliance on heuristics highlighted a significant need for more streamlined and end-to-end trainable approaches in object detection.

Transformer-Based Object Detection

While addressing this need, the emergence of transformer-based architectures led to a fundamental rethinking of how we approach object detection. Researchers began exploring the idea of set prediction, where the task is to directly predict a fixed set of objects in an image, without relying on proposals, anchors, or NMS. This shift in approach led to the development of DETR, the DEtection TRansformer [9], a model that treats object detection as a direct set prediction problem, fundamentally changing the object detection pipeline by simplifying it into a single, end-to-end trainable architecture with the help of transformers [5].

DETR: A Streamlined Approach to Object Detection

DETR (DEtection TRansformer) introduces an end-to-end approach to object detection by eliminating many of the hand-crafted components used in traditional pipelines. Instead of relying on region proposal networks or post-processing steps like NMS, DETR re-frames object detection as a direct set prediction problem.

The process begins with data preprocessing. Input images, such as rendered PDF pages, are resized to a fixed maximum dimension (1000 pixels on the longer side) while maintaining their aspect ratio [10]. Bounding box annotations are scaled accordingly to match the

resized image dimensions. For tasks like Table Structure Recognition (TSR) and Functional Analysis (FA), table bounding boxes are often padded by a fixed margin (commonly 30 pixels) before cropping. This padded cropping enables more robust model learning through data augmentation techniques such as random cropping and resizing [10]. Although a 30-pixel padding is standard, some research has explored reducing this padding [11]. Details on the PubTables-1M dataset and its use in training will be provided later.

The preprocessed image is then passed through a CNN backbone, typically a pre-trained ResNet, which extracts a 2D feature map. This feature map is flattened and fed into a Transformer encoder, which applies multi-head self-attention to capture global context and long-range dependencies across the image [9].

A Transformer decoder then processes a fixed set of learned positional embeddings, known as object queries. These queries interact with the encoder output via cross-attention, enabling the model to reason about potential objects and their relationships. Each decoder output is passed through a feed-forward network (FFN) to predict a class label and corresponding bounding box coordinates. Because the number of object queries is fixed, the model includes a special "no object" class (∅) to account for predictions that do not correspond to any ground truth object.

A key component of DETR is its set-based global loss, which uses bipartite matching via the Hungarian algorithm to optimally assign predicted objects to ground truth boxes in a one-to-one fashion. This assignment eliminates duplicate predictions and renders post-processing steps like NMS unnecessary. Unmatched predictions are trained to predict the ∅ class.

The loss function penalizes discrepancies in both classification and localization, typically combining a cross-entropy loss for the class label and a mix of L1 loss and generalized IoU (Intersection over Union) loss for bounding box regression [9], [12]. This results in a streamlined and fully differentiable object detection pipeline. A visual representation of the DETR model can be seen in Figure 2.

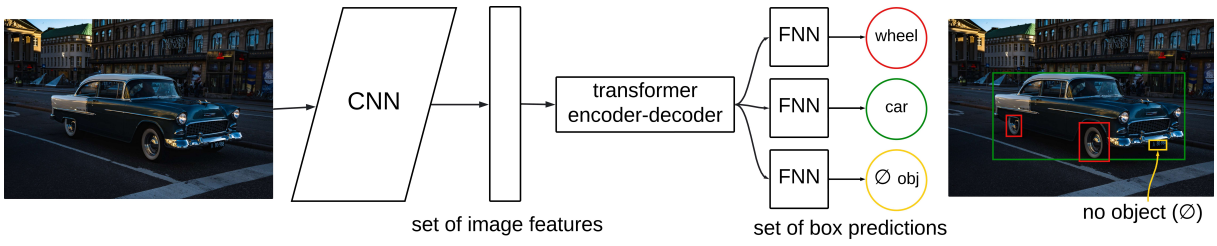

Figure 2: DETR set prediction flow

2.5 Table Detection

Table detection is a fundamental task in document image analysis, crucial for locating and identifying tables within various document types, including scanned images and PDFs. This process is complicated by the vast diversity in table layouts, structures, and visual appearances. Accurate detection is a long-lasting problem in the analysis of tabular in-

formation called table extraction (TE) [10].

While some tables are easy to detect even with simple rule-based approaches, it gets complicated once we enter real-world scenarios, as we mentioned before (2.1). Accurately detecting the table is only the first step in the TE process; correct table detection (TD) provides an essential foundation for subsequent table structure recognition (TSR), which involves identifying the rows, columns, and individual cells within the detected table, and finally functional analysis (FA) that help us understand values form tables [10], [13].

Various approaches exist for TD and TSR, broadly falling into rule-based methods and data-driven techniques. While traditional methods like OCR can extract text and apply heuristics, data-driven approaches primarily utilize deep learning or CNN-based models. These models have shown greater robustness to diverse table layouts and unseen examples because they are trained on specialized datasets with annotated tables. This reliance on annotated data for training is a key characteristic of these methods and leads us to consider the importance of such datasets and benchmarks for advancing TD and TSR [10], [13], [14].

TATR

The Table Transformer (TATR) model brings the power of Transformer-based object detection to the table extraction problem, unifying table localization and detailed structure parsing in a single, end-to-end framework [10], [14]. Rather than relying on separate rule-based steps, TATR treats every table element, whole tables, rows, columns, headers, and spanning cells as objects to be detected and classified, simplifying the pipeline and improving robustness to varied layouts.

TATR operates in two complementary phases. In the first phase, a convolutional backbone (ResNet-18) extracts image features which are then processed by a Transformer encoder–decoder pair. A small set of learnable "table" queries attends to the global feature map and directly predicts the bounding boxes of all tables on the page. This stage replaces heuristic region proposals with a single, end-to-end trainable model that learns to spot tables from visual cues alone [14].

In the second phase, the same backbone and Transformer layers are repurposed to focus on the internal structure of each detected table. A larger set of queries is used to detect rows, columns, header cells, and spanning cells simultaneously. The spatial arrangement of these detected elements implicitly defines the grid of cells, eliminating the need for post-processing rules. Both phases are trained on the richly annotated PubTables-1M dataset [10], which comprises nearly one million tables from scientific articles with detailed structure labels, using a matching-based loss to pair predictions with ground truth annotations [14].

At inference time, pages pass through the table detector to generate crops that isolate individual tables, which are then fed into the structure recognizer. The final outputs are complete HTML or CSV representations, and are produced by combining the detected object layouts with OCR-extracted text. By modeling tables as sets of interrelated objects, TATR captures both local cell detail and global table context within a unified

attention-based network, providing the way for more accurate and flexible table extraction for real-world documents [10], [14].

TATR's unique, attention-based design is particularly valuable for extracting tables from pharmaceutical lab report PDFs at Novo Nordisk. It can handle the wide variety of table layouts and font styles found in real-world medical documents, which reduces the need for manual correction. TATR directly addresses our goal of automating the conversion of non-standardized PDF tables into a consistent format suitable for analysis and compliance workflows by producing end-to-end structured outputs.

2.6 Data

This section describes the two main datasets used in our project, PubTables-1M and a set of internal lab reports from Novo Nordisk. PubTables-1M provides a large, well-annotated benchmark for training and testing table extraction models in a controlled setting. The Novo Nordisk dataset, by contrast, reflects real-world complexity, unstructured PDFs with varied table styles and formatting. Together, these datasets allow us to train robust models and test them under practical conditions.

PubTables-1M

PubTables-1M is a large-scale, richly annotated dataset of nearly one million tables extracted from scientific publications derived from the PubMed Central Open Access (PMCOA) dataset [15], introduced by Smock *et al* at Microsoft Research [10]. PubTables-1M was selected for this project due to its extensive scale, comprehensive structural annotations, and its approach to addressing annotation inconsistencies prevalent in other datasets, such as oversegmentation[2], through a novel canonicalization[3] procedure. These characteristics make it particularly well-suited for training robust deep learning models for complex tables with medical data.

Each table in the corpus is accompanied by precise bounding-box annotations for table elements (entire table, rows, columns, header cells, and spanning cells), enabling both coarse and fine-grained evaluation of TD, TSR, and FA.

To ensure annotation consistency, PubTables-1M employs a canonicalization procedure: HTML-derived cell coordinates are aligned with PDF renderings, and oversegmented regions (split spanning cells) are merged into single ground-truth objects. This process resolves ambiguities inherent in crowd-sourced and heuristic-based label sets, yielding unambiguous, one-to-one correspondence between ground truth and predicted detections. The authors also provide evaluation scripts implementing both standard object detection metrics (AP, AP50, AP75) and the GriTS family of metrics (topology, content, location) for holistic TSR assessment [14].

[2]Oversegmentation occurs when a spanning cell in a table header is split into multiple grid cells, typically because hidden borders are not represented explicitly in markup, thus resulting in ambiguous or inconsistent ground-truth annotations (Smock *et al.* [10]).

[3]Canonicalization is the procedure used to merge such oversegmented cells back into their intended single spanning cells, thereby producing a unique, unambiguous structural interpretation of the table (Smock *et al.* [10]).

issue involved tiny superscript and units, such as those found in pharmaceutical reports, such as µg/L, which often extended beyond visible cell borders. To preserve OCR [6] accuracy and context, annotators were instructed to include these elements within the cell box boundaries.
Finally, annotator fatigue posed a significant concern, as precision tended to decline during longer sessions. To mitigate this, annotation sessions were limited to 90 minutes, a threshold confirmed by QA logs as the point beyond which accuracy began to noticeably degrade.

3.2 Evaluation Metrics

We evaluate both table detection and table structure recognition tasks using a mix of standard object-detection metrics and specialized TSR metrics developed by the authors of TATR [10], [17]. These metrics are basic to our project since they are widely used in TD and TSR benchmarks, and including GriTS, that used by TATR, lets us compare our models directly against theirs.

Intersection over Union (IoU) measures how much a predicted box $\hat{B}$ overlaps a ground-truth box B. For example, if B the first covers pixels from $(10, 10)$ to $(110, 60)$ and $\hat{B}$ covers $(20, 20)$ to $(120, 70)$, the overlapping rectangle might be $(20, 20)$ to $(110, 60)$. If that overlap area is $90\times40 = 3600$ pixels and the union area (total area covered by both boxes) is 5000 pixels, then

$$\text{IoU}(B, \hat{B}) = \frac{3600}{5000} = 0.72.$$

A detection is counted as correct (true positive) only if $\text{IoU}(B, \hat{B}) \geq \tau$. For instance, if $\tau = 0.5$ (as in AP_{50}), an IoU of 0.72 is sufficient, but if $\tau = 0.75$it is not.

Precision and Recall summarize the quality of those detections:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

Here, TP, FP, and FN stand for true positives, false positives, and false negatives, respectively. To illustrate, imagine a page with exactly five tables (ground truth). Our model outputs six candidate boxes labeled as tables. Among those six, four align with real tables (so TP = 4), while two do not match any true table (FP = 2). Because there were five actual tables and only four were detected, there is one table that the model missed entirely (FN = 1). In this scenario,

$$\text{Precision} = \frac{4}{4+2} = 0.67, \quad \text{Recall} = \frac{4}{4+1} = 0.80.$$

Precision of 0.67 means that out of all detections the model made, 67% were correct. Recall of 0.80 means that it found 80% of the actual tables present.

Average Precision (AP) is the area under the precision–recall curve at a given IoU threshold. We compute AP for each chosen threshold τ by ranking all predictions by confidence and, at each confidence cutoff, recalculating TP, FP, and FN using that same

[6] Optical Character Recognition

τ. Plotting the resulting precision and recall values as the confidence threshold moves from highest to lowest yields a curve; the area under that curve is AP τ. We report:

$$\mathrm{AP}_{[.5:.95]} = \frac{1}{10}\sum_{k=0}^{9} \mathrm{AP}\big(\tau = 0.5 + 0.05k\big),$$

as well as the special cases AP_{50} and AP_{75}. For example, suppose the model's predictions sorted by confidence produce the following precision–recall pairs at certain score cutoffs: $(0.25, 1.00)$, $(0.50, 1.00)$, $(0.50, 0.67)$, $(0.75, 0.75)$, and $(1.00, 0.80)$. At the highest confidence cutoff (0.95) only one true table is counted (recall = 0.25, precision = 1.00). Lowering the cutoff to 0.90 yields two true detections (recall = 0.50, precision = 1.00). Further lowering to 0.85 introduces one false positive (recall = 0.50, precision $\approx$ 0.67), and so on until all predictions are included (recall = 1.00, precision = 0.80). Connecting those points and measuring the area under them gives AP at that τ. If we repeat this for $\tau = 0.50, 0.55, \ldots, 0.95$ an average, we obtain $\mathrm{AP}_{[.5:.95]}$.

Average Recall (AR) is recall averaged over multiple IoU thresholds or over a fixed number of top detections per image. For instance, consider three IoU thresholds: 0.50, 0.75, and 0.95. If at IoU = 0.50 the model detects 3 out of 4 true tables (recall = 0.75), at IoU = 0.75 it detects 2 out of 4 (recall = 0.50), and at IoU = 0.95 it detects 1 out of 4 (recall = 0.25), then AR over those three thresholds is

$$\frac{0.75 + 0.50 + 0.25}{3} = 0.50.$$

In practice, we average recall over the ten IoU thresholds from 0.50 to 0.95 in steps of 0.05, using a fixed maximum number of detections per page (top 100 highest-confidence boxes). AR thus measures how robustly the model can recover true tables as the required overlap becomes more stringent.

Table Content Exact Match Accuracy $\mathrm{Acc}_{\mathrm{Cont}}$ is the fraction of tables whose every cell, both its text and its row/column spans, matches exactly. A single mismatch gives a score of zero, so it reflects end-to-end exactness in TSR outputs.

Grid Table Similarity (GriTS) treats each table as a $m \times n$ matrix of grid cell entries and computes an F-score–like similarity with output value from 0 to 1.

Three variants are used for our evaluation. We will not go deeply into their explanation, but will give an overall idea of what they measure. Those same metrics were used for the initial TATR model, and will serve as a baseline score in experiments.

- $\mathrm{GriTS}_{\mathrm{Top}}$: f is IoU between each cell's *grid-box* span $[0, 0, 1, 1]$ normalized by row/column spans, so it scores the table's layout and spanning structure.
- $\mathrm{GriTS}_{\mathrm{Con}}$: f is the normalized longest-common-subsequence similarity of the two cell text strings, capturing partial text matches.
- $\mathrm{GriTS}_{\mathrm{Loc}}$: f is IoU of each cell's absolute bounding box in document coordinates, measuring how accurately positions and sizes are recovered.

By combining AP/AR (IoU-driven detection quality) with $\mathrm{Acc}_{\mathrm{Cont}}$ (exactness) and GriTS (partial correctness in topology, content, and location), we obtain a comprehensive, unified evaluation set of metrics for DETR and TATR models across both TD and TSR tasks.