International Journal on Document Analysis and Recognition (IJDAR) (2024) 27:127–145
https://doi.org/10.1007/s10032-023-00453-8

ORIGINAL PAPER

TableStrRec: framework for table structure recognition in data sheet images

Johan Fernandes[1] · Bin Xiao[1] · Murat Simsek[1] · Burak Kantarci[1] · Shahzad Khan[2] · Ala Abu Alkheir[3]

Received: 24 November 2021 / Revised: 11 August 2023 / Accepted: 16 August 2023 / Published online: 8 September 2023

Abstract

Billions of documents in data sheet format are shared between various organizations across the globe on a daily basis. The essential information in these documents is presented in tabular format. Extracting and assimilating this information can help organizations make data-driven decisions. Solutions for detecting tables in document images have been well explored. Thus, in this work, we propose TableStrRec, a deep learning-based approach to recognize the structure of such detected tables by detecting rows and columns. TableStrRec comprises two Cascade R-CNN architectures, each with a deformable backbone and Complete IOU loss to improve their detection performance. One architecture detects and classifies rows as regular rows (rows without a merged cell) and irregular rows (groups of regular rows that share a merged cell). The second architecture detects and classifies columns as regular columns (columns without a merged cell) and irregular columns (groups of regular columns that share a merged cell). Both architectures work in parallel to provide the results in a single inference. We show that utilizing TableStrRec to detect four classes of objects improves the table structure recognition performance on three public test sets. We achieve 90.5% and 89.6% weighted average F1 scores on the ICDAR2013 test set for rows and columns, respectively. On the TabStructDB test set, we achieve 72.7% and 78.5% weighted average F1 score for rows and columns, respectively. We also evaluate the proposed method under the FinTabNet dataset using the structure-only TEDS score, achieving 98.34%, which can outperform most state-of-the-art benchmark models.

Keywords Deep learning · Deformable convolutional neural networks · Image processing · Document analysis · Table structure recognition · Row and column detection · Page object detection

✉ Burak Kantarci
burak.kantarci@uottawa.ca

Johan Fernandes
jfern090@uottawa.ca

Bin Xiao
bxiao103@uottawa.ca

Murat Simsek
murat.simsek@uottawa.ca

Shahzad Khan
shahzad@gnowit.com

Ala Abu Alkheir
ala_abualkheir@lytica.com

1 School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, ON K1N 6N5, Canada

2 Gnowit, 383 Celtic Ridge Cres, Ottawa, ON K2W 0B5, Canada

3 Lytica, 555 Legget Dr, Ottawa, ON K2K 2X3, Canada

1 Introduction

Most organizations operate as part of a sophisticated global supply chain involving manufacturers, component distributors and product designers. These entities need to share information through various types of documents. The electronic industry in particular is reliant on accurate and up-to-date data sheets to develop reliable and compatible systems. These data sheets are mostly digitally born documents. They can be portable document format (PDF) based or scanned images of paper-based documents. They contain valuable information such as product availability, specifications and pricing in tabular format to provide an easy and concise way to analyse and make decisions involving large amounts of structured information [1]. Extracting and assembling the data from such tables for analysis can help organizations to develop a sophisticated knowledge management system. This system can be utilized to support efficient automation as well as evidence and data-driven decision-making [2]. We hypothesize that the task of extracting these data can be handled in three stages, as shown in our proposed table text extraction pipeline in Fig. 1.

Stage 1 comprises table detection and table image classification steps. Documents have pages that contain tables and pages that do not contain tables. Thus, we propose that this stage should provide the locations of the tables in the image (table detection task) and the full page image (table image classification task) as the final result of this stage. The detected tables can then be extracted and sent to the next stage to address table structure recognition (TSR). With increasing work done on the application of deep learning towards image processing tasks such as object detection, multiple solutions are proposed by [3–5] for addressing the table detection task in data sheet images. TableDet [6] is proposed as a deep learning-based solution to address both table detection and table image classification in a single inference. The TableDet solution achieves the highest F1 scores for table detection on three public datasets (ICDAR 13 [1], ICDAR 17 [7] and ICDAR 19 [8]). Furthermore, for table image classification, it also accurately identifies all images with tables and significantly reduces the number of images without tables from being processed in the proposed pipeline. Thus, TableDet addresses both steps of stage 1 of the table text extraction pipeline which is displayed in Fig. 1.

The table structure ascertainment (or mapping out) is a more complex task due to the many variations in the table layouts in documents published by different companies. In this work, we propose a solution to address the TSR task in stage 2 of the proposed pipeline, as shown in Fig. 1. This task can be decomposed into detecting rows and columns [9, 10] which can then be used to identify the locations of cells within the table. An alternative approach would be to directly detect the cells of the tables [4, 11, 12]. The tables can appear with or without line separators between the cells [4]. Furthermore, the presence of the merged cells can also indicate a hierarchically structured layout of information [11, 12].

To bridge this gap, we present TableStrRec, a deep learning-based approach to detect and classify rows and columns in table images. To develop the table images, we employ TableDet [6] which provides accurate locations of tables in full page images. Utilizing these locations, we can extract the tables from such images. In each table image, we propose detecting and classifying rows as regular rows (rows without a merged cell) and irregular rows (groups of regular rows with a shared merged cell). Similarly, we propose detecting and classifying columns as regular columns (columns without a merged cell) and irregular columns (groups of regular columns with a shared merged cell). In addition, we propose utilizing one Cascade R-CNN architecture to detect and classify the rows and another to detect and classify the columns in parallel. Thus, TableStrRec can detect four classes of objects in a table image. Each Cascade R-CNN architecture is equipped with a deformable convolution backbone [13] and utilizes Complete IOU loss [14] to improve its object detection performance.

Our main contributions in this work are as follows:

1. An end-to-end deep learning-based approach that solves the table structure recognition in data sheet images (referred to as TableStrRec).
2. Detection of irregular rows and columns in addition to detecting regular rows and columns to capture the hierarchical layout of tables with merged cells.
3. Exploring the impact of utilizing a deformable convolution backbone as a feature extractor and Complete IOU loss for bounding box regression to improve the detection performance of the proposed TableStrRec system.

As a result of these contributions, we show that TableStrRec can improve row and column detection by more than 7% regarding weighted average F1 scores on ICDAR2013 and TabStructDB test sets and can also outperform state-of-the-art methods regarding the structure-only TEDS score on the FinTabNet [15] dataset.

The remainder of this article is structured as follows: In Sect. 2, we describe the previous works that address table detection and table structure recognition. In Sect. 3, we present the steps to solve table structure recognition in data sheet images. We present the results of the proposed system in Sect. 4 and conclude this work in Sect. 5.

2 Related work and motivation

Document analysis has been well investigated from multiple perspectives, such as the type of document to be processed and the type of solution employed to analyse the document. The important information in these documents is presented in tabular format [16]. Identifying such tables and extracting the content for query answering can help organizations with automation as well as support data-driven analysis and decisions [2]. Recent works have shifted from PDF-based extraction techniques [17] to deep learning techniques on images [4, 11] to extract this tabular information. As aforementioned, many table detection approaches have achieved promising results using different types of object detection models. In this study, we simply use TableDet [6] to extract tables. Therefore, in this section, we mainly focus on the studies regarding TSR.

2.1 Table structure recognition

There have been many studies discussing table structure recognition (TSR) using different problem formulations. Based on the problem formulations of these studies, we

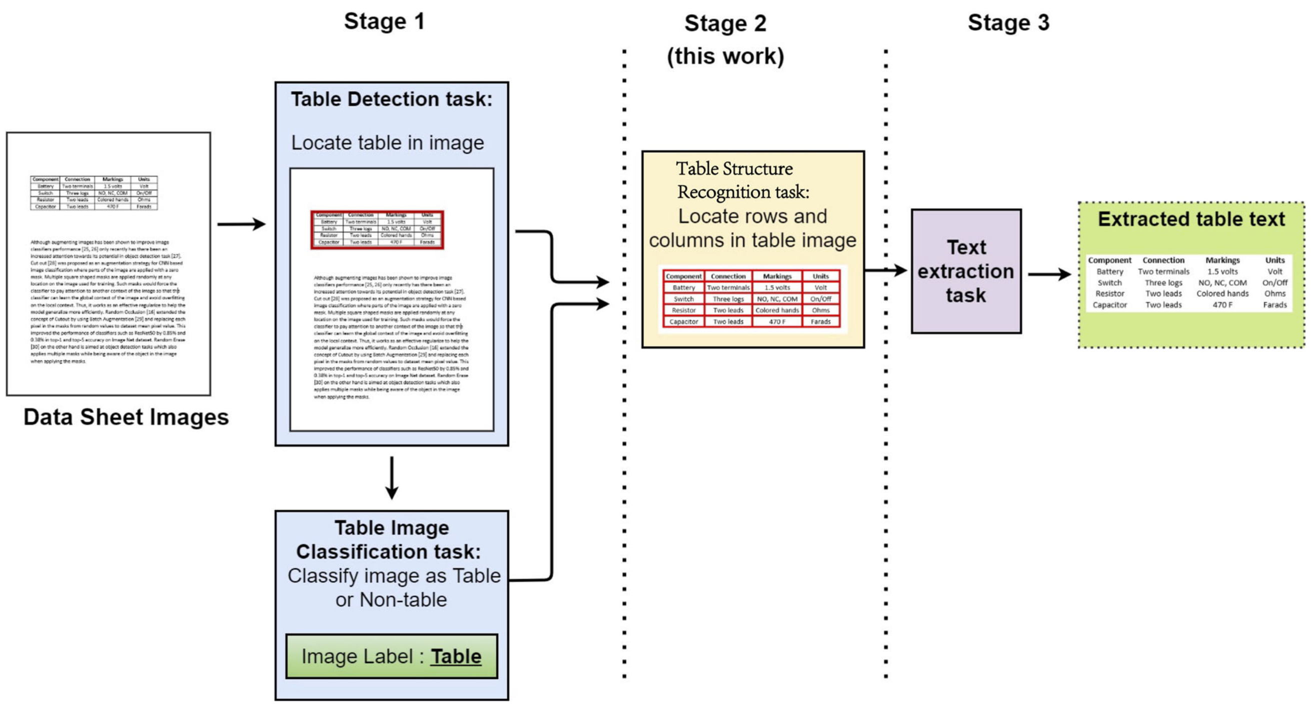


Fig. 1 Table text extraction pipeline. This work deals with stage 2 of the proposed pipeline

can categorize them into three groups: cell-level methods, column–row-level methods and image-to-sequence models. Cell-level methods usually require extracting table cells first and then building the associations of the extracted cells. FLAG-Net [18] is a typical cell-level method that employs an Element Proposal Network (EPN) to detect table elements first and then uses a proposed Flexible Context Aggregation to build the associations of the detected table elements. More specifically, the EPN in the FLAG-Net is a variation of Mask-RCNN [19] without the mask branch, the Flexible Context Aggregation is implemented by a stack of transformer layers considering the dense and sparse context of the table elements, and the associations of different table elements are defined as “same row”, “same column” and “same cell”. TabStructNet [20] is another cell-level approach that detects table cells first and then builds the cell associations. TabStructNet also employs a variation of Mask-RCNN, which uses dilated convolutions in the RPN network, appends a pathway to the FPN network and adds an extra proposed loss function. After detecting the table cells, TabStructNet classifies the detected cells into “same column” and “same row” groups to build the associations of detected cells with a proposed Structured Recognition Network implemented by LSTM layers and two classifiers. Graphical models have also been applied in cell-level approaches. For example, study [21] introduces a Neural Collaborative Graph Machines which uses text segmentation bounding boxes as table elements and proposes Collaborative Blocks to extract and represent features as directed graphs and, at last, classifies paired table elements into “same column”, “same row” and “same cell” groups. Many other studies [22–27] can also be categorized into cell-level approaches, and most of those methods leverage either object detection or semantic segmentation methods to locate table elements, then use different strategies to extract features and classify paired table elements into “same column”, “same row” and “same cell” groups or use a graph model to represent complex table structures. It is worth mentioning that not only visual features but also text features have also been applied in some of these studies [21].

In contrast, column–row-level approaches [9, 10, 28–30] often detect or segment columns and rows directly or predict the separator lines of columns and rows. Study [28] formulates the TSR problem as an object detection problem and defines six types of table components, including table, column, row, column header, projected row header and table spanning cell in the proposed dataset. DeepTabStR [9] is another study formulating the TSR problem as detecting columns and rows. In DeepTabStR, deformable convolution layers are used to replace conventional convolution layers to build the baseline object detection models, including Faster-RCNN, RFCN and FPN [29]. Following the problem formulation of DeepTabStR, study [10] argues that training two separate models detecting columns and rows can perform better than a single model detecting both columns and rows simultaneously. Furthermore, study [10] proposes an anchor optimization method to further refine the anchor generation process of two-stage detection models by applying a K-means-based clustering method. Besides detecting columns and rows directly, some studies [31–34] also formulate the problem of predicting the separator lines of columns and rows. SPLERGE [32] proposes a solution containing two models: a Split Model and a Merger Model. The Split Model is used to predict the separator lines of the table without considering the spanning cells, and the Merger is responsible for predicting the table elements and spanning cells. Similarly, CornerNet+FRCN [35] also follows the pipeline of predicting the separator lines first and then merging cells to build the table structures. More specifically, CornerNet+FRCN uses a spatial CNN network containing two branches to predict columns separator lines and row separator lines, respectively, and employs relation network [33] to merge cells. To sum up, column–row-level approaches usually either predict columns and rows or predict separator lines of columns and rows first and then utilize post-process methods or models to merge cells. It is worth mentioning that our proposed method in this paper is also a columns–row-level approach which detects columns and rows firstly and uses a post-processing method to build the complex table structure.

Besides cell-level and column–row-level methods, image-to-sequence models are another popular formulation. Image-to-sequence models usually utilize a CNN backbone to extract the input features first and then follow the transformer encoder–decoder [36] architecture to output the sequence of HTML tags directly. TableMASTER [37] is a typical method in this group and contains two transformer decoder branches to predict the HTML sequence and bounding boxes, respectively. TableFormer [38] follows a similar architecture that consists of a CNN backbone network, a transformer encoder, a Structure Decoder to output HTML tags and a BBox Decoder to predict the bounding boxes. It is worth mentioning that for the evaluation of those image-to-sequence models, TEDS score [39] is the most popular metric because it can measure the distance between the prediction sequence and the ground sequence. In this study, even though our proposed method is a column–row-level method, we can transform the predictions into HTML tag sequences and evaluate our model with the TEDS score.

3 Methodology

In this section, we describe the steps taken to develop the TableStrRec system which detects rows and columns on table images. In addition, it also classifies the rows as regular or irregular rows and also classifies the columns as regular or irregular columns. Multiple deep learning-based solutions have addressed the table detection task in images [3–5, 40]. However, we propose utilizing the detected table region provided by the TableDet [6] system to develop table images. For this work, we determined that utilizing a table image which contains only the table region of a full page image and avoids the non-table sections of the image would help the TableStrRec system to focus on locating the rows and columns of the table, as shown in Ref. [5, 9, 11].

3.1 Task definition

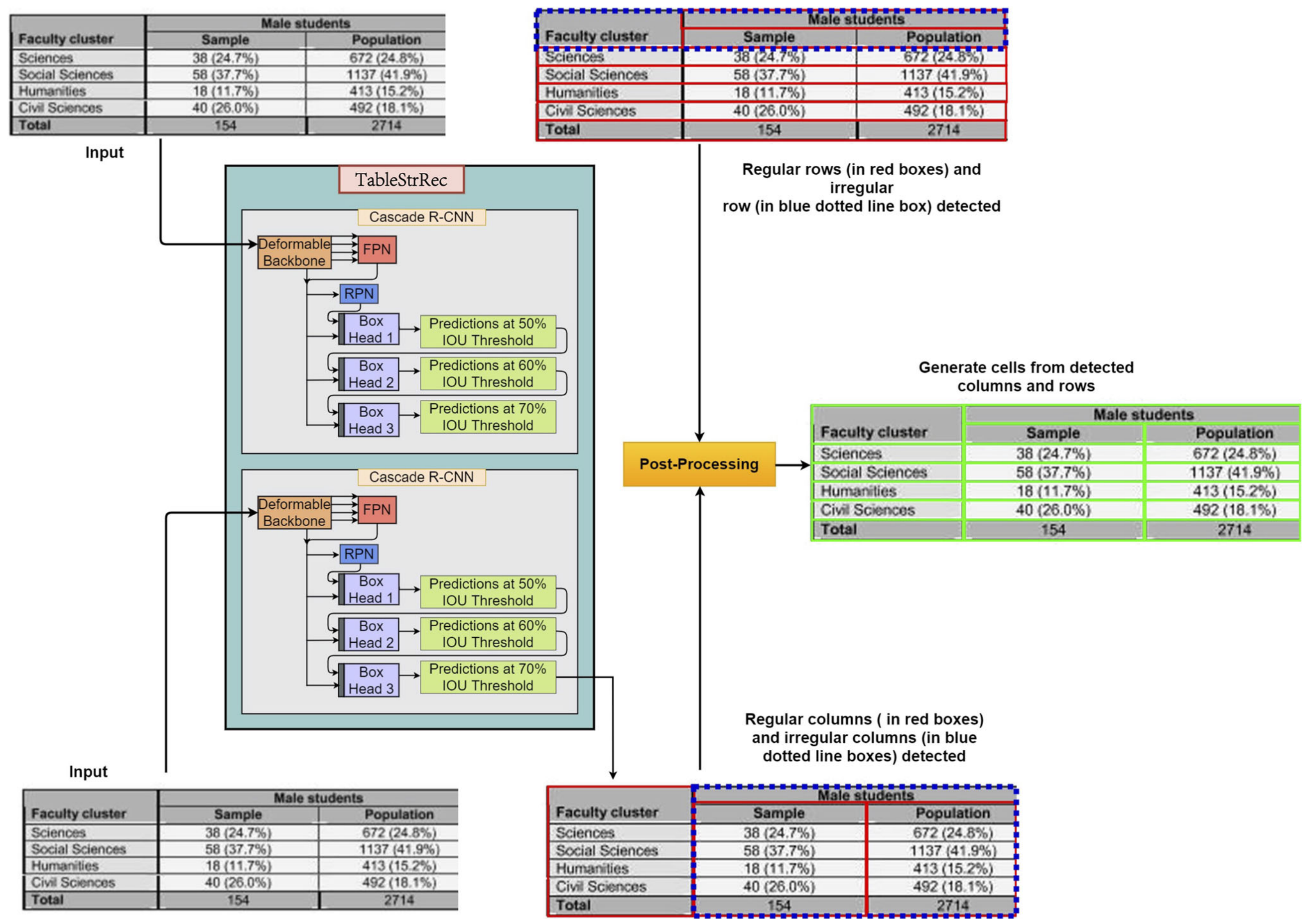
The task of TSR in table images involves identifying the components of the table which can be decomposed into two classes or components: rows and columns, as shown in Ref. [5, 9]. It can further be decomposed into a single class or component: cells, as shown in Ref. [4, 11]. Due to the large differences in the layouts of different tables, we propose to detect rows and columns first to understand the layout of the content within the table. The tables within different data sheet (document) images have different layouts, based on the need of the author to show different relationships between entries in the table, or a need to present the content of these tables in a concise format, or simply aesthetic preferences. This difference is brought by the presence or absence of merged cells and the presence or absence of line separators between rows and columns. We define a merged cell as multiple cells grouped as one cell that stretches across more than one row or column [12]. These merged cells indicate the common content between two or more rows or columns, as shown in the input table images in Fig. 2. These merged cells are added to reduce the repetition of content among the rows or columns. In addition to the existence or absence of merged cells, the tables may or may not contain line separators between rows and columns to indicate these merged cells.

3.1.1 Regular and irregular class definitions

We formulate the TSR problem as an object detection task where the objects to be detected are the rows and columns of the table. We refer to the rows and columns that do not contain a merged cell (multiple cells grouped as one cell) as regular rows and regular columns, respectively. When a group of regular rows share a merged cell [12] that stretches across them we refer to this group of rows as an irregular row. Similarly, we refer to a group of regular columns that share a merged cell [12] which stretches across them as an irregular column. Thus, along with detecting regular rows and regular columns, we propose detecting irregular rows and irregular columns to provide more details on the layout of the table contents.

3.1.2 Detecting groups of components versus detecting cells

Instead of detecting only the merged cell in this study, we detect the entire group of rows and columns attached to the merged cell for two reasons. Firstly, by capturing the group of regular rows and columns that are part of irregular rows


Input
TableStrRec
Cascade R-CNN
Deformable Backbone
FPN
RPN
Box Head 1
Box Head 2
Box Head 3
Predictions at 50% IOU Threshold
Predictions at 60% IOU Threshold
Predictions at 70% IOU Threshold
Input
Regular rows (in red boxes) and irregular row (in blue dotted line box) detected
Post-Processing
Generate cells from detected columns and rows
Regular columns (in red boxes) and irregular columns (in blue dotted line boxes) detected

Faculty cluster	Male students	
	Sample	Population
Sciences	38 (24.7%)	672 (24.8%)
Social Sciences	58 (37.7%)	1137 (41.9%)
Humanities	18 (11.7%)	413 (15.2%)
Civil Sciences	40 (26.0%)	492 (18.1%)
Total	154	2714

Fig. 2 TableStrRec system comprises of two Cascade R-CNN architectures each with deformable backbones and CIOU loss at each box head to improve detection performance. One Cascade R-CNN architecture detects and classifies rows, while the other detects and classifies columns. As a post-processing step the intersecting lines of the detected rows and columns can be used to generate cells

and columns, respectively, it becomes easier to provide hierarchical details of which rows and columns share a common merged cell. Secondly, if we only detect the merged cells instead of the group of regular rows and columns, then the object detection architecture would try to detect cells along with rows and columns which would be redundant. Thus, we propose not only to detect rows and columns but also to classify them as regular rows, irregular rows, regular columns and irregular columns to address TSR in images.

3.2 TableStrRec architecture

State-of-the-art solutions for tasks such as image classification and object detection utilize deep learning-based techniques due to the generalization ability of the same architecture to detect different types of objects, such as natural objects [41] or objects in data sheet images [4]. In this study, we utilize a deep learning-based object detection approach to recognize the structure of a table in data sheet images. For this task, we determined that the structure of the table can be decomposed into four classes of objects. Instead of detecting only rows and columns, we propose detecting irregular and regular rows along with irregular and regular columns in order to capture the merged cells within the structure. By identifying these irregular rows and irregular columns, the object detection architecture would be able to provide more details about the hierarchical layout of the table.

Thus, we put forward TableStrRec, a deep learning-based approach that utilizes an object detection architecture to accurately identify objects of these four classes in data sheet images. We employ the Cascade-RCNN [41] object detection architecture to develop this TableStrRec system. The Cascade R-CNN architecture comprises four components as shown in Fig. 2: backbone, Feature Pyramid Network (FPN), Region Proposal Network (RPN) and the Region of Interest or box head. In this work, we utilize a ResNet [42] architecture as a backbone to act as the feature extractor. As rows and columns appear with different scales and orientations, we employ deformable convolutions [13] in this ResNet architecture to enhance the quality of the detected rows and columns. In addition to utilizing deformable convolutions, we also utilize an FPN to provide rich features to the box head and RPN of this architecture.

The Cascade R-CNN architecture uses three box heads to refine the bounding boxes for each image [41] at multiple Intersection Over Union (IOU) values [43]. The IOU value is measured as the Intersection Over Union of proposed bounding box $B = (x_1, y_1, x_2, y_2)$ generated by the architecture and ground truth $B^{\mathrm{gt}} = (x_1^{\mathrm{gt}}, y_1^{\mathrm{gt}}, x_2^{\mathrm{gt}}, y_2^{\mathrm{gt}})$ bounding box, as shown in (1). Furthermore, each box head receives multi-scale feature maps from the FPN and the final layer of the backbone along with proposal bounding boxes for refinement, as shown in Fig. 2. The first box head receives the proposal bounding boxes from the RPN and refines the bounding boxes which achieve an IOU value greater than 50%. These improved boxes are provided as proposals to the second box head where the bounding boxes greater than 60% IOU value are refined and are then sent to the third box head where the boxes above 70% IOU value are refined and provided as the final output. At each box head, the bounding boxes are refined by calculating the bounding box regression loss to measure the difference between the proposed and ground truth boxes. In addition, a cross-entropy loss is also calculated at each box head to measure the difference between the predicted and ground truth object class labels. Both losses are calculated together as a multi-task loss as formulated in (2).

$$\mathrm{IoU}(B, B^{\mathrm{gt}}) = \frac{B \cap B^{\mathrm{gt}}}{B \cup B^{\mathrm{gt}}} \tag{1}$$

$$\begin{aligned} L(p_i, t_i) = & \frac{1}{N_{\mathrm{cls}}} \sum_i L_{\mathrm{cls}}(p_i, p_i^*) \\ & + \lambda \frac{1}{N_{\mathrm{reg}}} \sum_i p_i^* L_{\mathrm{reg}}(t_i, t_i^*) \end{aligned} \tag{2}$$

3.2.1 IOU-based regression loss

The standard bounding box regression loss utilized by each box head [41] is the $l1$ loss. However, the bounding box regression performance of each box head is evaluated in terms of an IOU value. To bridge this gap between the metrics for loss calculation and evaluation, IOU loss [44] was introduced. Furthermore, rows and columns can appear with different scales and orientations in table images. Utilizing aspect ratio as a parameter in the regression loss calculation can be beneficial towards refining the bounding boxes for objects with multiple scales and orientations as shown in [14].

$$v = \frac{4}{\pi^2} \left(\arctan \frac{w^{\mathrm{gt}}}{h^{\mathrm{gt}}} - \arctan \frac{w}{h} \right)^2 \tag{3}$$

$$\alpha = \frac{v}{(1 - \mathrm{IoU} + v')} \tag{4}$$

$$L_{\mathrm{CIoU}} = 1 - \mathrm{IoU} + \frac{\rho^2(b, b^{\mathrm{gt}})}{c^2} + \alpha v \tag{5}$$

Thus, we propose utilizing Complete IOU (CIOU) [14] which operates with parameters such as distance between the centre points of the two boxes ($\rho(.)$), diagonal length of the box which is denoted as c and the aspect ratio of the bounding boxes to accurately detect the object. CIOU loss considers the consistency of the aspect ratio (v) and a positive trade-off parameter (α) as given in (3) and (4), respectively. To accurately detect and classify the rows and columns, we employ CIOU loss at all three box heads of the Cascade R-CNN architecture which is utilized to develop the TableStrRec system.

In this work, we detect regular and irregular rows along with regular and irregular columns to gain more information about the table structure. However, rows generally appear with a horizontal alignment, while columns appear with a vertical alignment [45] within the table. Thus, our proposed TableStrRec system consists of two Cascade R-CNN architectures, each detecting two classes of objects in the same table image as shown in Fig. 2. Each architecture utilizes a deformable backbone and CIOU loss at each box head to improve bounding box predictions. As shown in Fig. 2, we employ one Cascade R-CNN architecture to detect the rows (regular and irregular) and the second Cascade R-CNN architecture to detect the columns (regular and irregular) in the table images. Both architectures work together in parallel to provide these results. The final result of the TableStrRec system is the locations of regular and irregular rows along with regular and irregular columns. Hence, TableStrRec detects four classes of objects in a single inference.

We display the inference process of the TableStrRec in Fig. 2. Since there are two Cascade R-CNN architectures working in parallel within the TableStrRec system, we provide a two copies of input table images to the TableStrRec system. The input table image in Fig. 2 has objects of all four classes. Thus, the TableStrRec system provides the locations of each of the four object classes. We denote the detected irregular rows and columns with blue dotted boxes, while the detected regular rows and columns are denoted with red boxes to indicate the output of the TableStrRec system. Furthermore, as post-processing step we utilize the intersecting lines of the detected rows and columns to generate cell locations, as indicated in Fig. 2. The cell locations display not only the locations of regular cells (cells not stretching over multiple cells) but also the locations of merged cells. The irregular rows and irregular columns consider the merged cell and the regular rows and regular columns, respectively. Thus, the intersecting lines developed by these four detected object classes presents an easier pathway to locate the merged cells in the tables, as shown in the final result in Fig. 2. By locating the merged cells, the TableStrRec system can accurately identify the hierarchical layout of data within the table image.

3.3 Post-processing algorithm

In this section, we discuss the post-processing algorithm after the columns and rows are detected and classified. As aforementioned, we classify columns and rows with merged cells as "irregular", as shown in Fig. 4. Therefore, we can infer the merged cells by the regular and irregular results. Algorithm 1 shows an example of inferring merged cell through detected regular and irregular columns. It is worth mentioning that interoperation in Algorithm 1 means calculating the intersection area. Similarly, we can also use irregular and regular rows to infer merged cells. And the regular cells are the intersection areas of regular rows and regular columns. After regular columns, regular rows and merged cells (spanning cells) are obtained, we can further apply the algorithm in study [28], which can be implemented by its code base [46] to generate the HTML and CSV files.

Algorithm 1 Finding the merging cells

```
    Input: The detected regular columns ($\mathcal{R}$) and irregular columns ($\mathcal{I}$)
    Output: Merged cells in the table
1:  Initialize an empty Dict $\mathcal{D}$
2:  while $\mathcal{I}$.size() $> 0$ do
3:      Visit irregular column $i_n$ from $\mathcal{I}$
4:      Initialize an empty List $\mathcal{L}$
5:      while $\mathcal{R}$.size() $> 0$ do
6:          Visit regular column $r_k$ from $\mathcal{R}$
7:          Calculate the score $s_k$=inter($i_n$, $r_k$) / $r_k$
8:          if $s_k >$ threshold then
9:              Add $r_k$ into $\mathcal{L}$
10:         end if
11:     end while
12:     Add $\{i_n : \mathcal{L}\}$ into $\mathcal{D}$
13: end while
14: Initialize an empty List $\mathcal{M}$
15: while $\mathcal{D}$.size() $> 0$ do
16:     Visit $key_j$ and value list $l_j$
17:     Calculate the union area $u_j$ of $l_j$
18:     Exclude the union area $u_j$ from $key_j$ to obtain merged cell $c_j$
19:     Add $c_j$ into $\mathcal{M}$
20: end while
21: RETURN $\mathcal{M}$
```

3.4 Dataset

We utilize three data sheet (document) image datasets in this work; ICDAR2013 [1], TabStructDB [9] and FinTabNet [15]. All datasets consist of only the table regions of each image. The ICDAR2013 dataset [1] contains 67 American and European Union documents. There are a total of 238 pages combining all documents of the dataset which are converted to images. This dataset provides a total of 156 tables. The TabStructDB was developed by [9] who utilized the ICDAR2017 POD competition dataset [7] to extract a total of 1081 tables. We maintain the same training and test split as [9] for both datasets. As shown in Table 1, 125 tables of ICDAR2013 dataset are used for training and 31 tables for testing. Similarly, 731 tables and 350 tables from TabStructDB are used for training and testing, respectively. We manually corrected the row and column annotations of each image to ensure that the rows and columns within the image are aligned with each other and they do not capture merged cells. We refer to these rows and columns as regular rows and regular columns, respectively. Moreover, we develop annotations for irregular rows and irregular columns as well. An irregular row is annotated by drawing a single bounding box over a group of regular rows and the merged cell that stretches over these regular rows. Thus, an irregular row annotation captures the regular rows as well as the merged cell shared by them. Similarly, we develop annotations for irregular columns which group together multiple regular columns and include the merged cell that stretches over these regular columns. FinTabNet is a popular dataset collected from the annual reports of S&P 500 companies. We first processed the annotations to align with the other two datasets. As a result, we obtained 78,537, 9650 and 9289 tables for the training, validation and testing, respectively.

Table 1 Datasets statistics

Dataset	Train	Test	Validation
ICDAR2013	125	31	–
TabStructDB	731	350	–
FinTabNet	78537	9289	9650

4 Performance study

In this work, we propose detecting regular rows, irregular rows, regular and irregular columns in table images to address the task of TSR. There are four object classes in a table image to accurately identify the layout of content in the table. We evaluate the performance of our presented TableStrRec system on ICDAR2013 [1], TabStructDB [9] and FinTabNet [15] datasets, which have annotations for all four object classes. To the best of our knowledge, the state-of-the-art solution for row and column detection on ICDAR2013 and TabStructDB datasets is the DeepTabStr [9] system. The DeepTabStr system only detects rows and columns and does not classify them into regular and irregular classes. Thus, we train the DeepTabStr system on the training sets of the

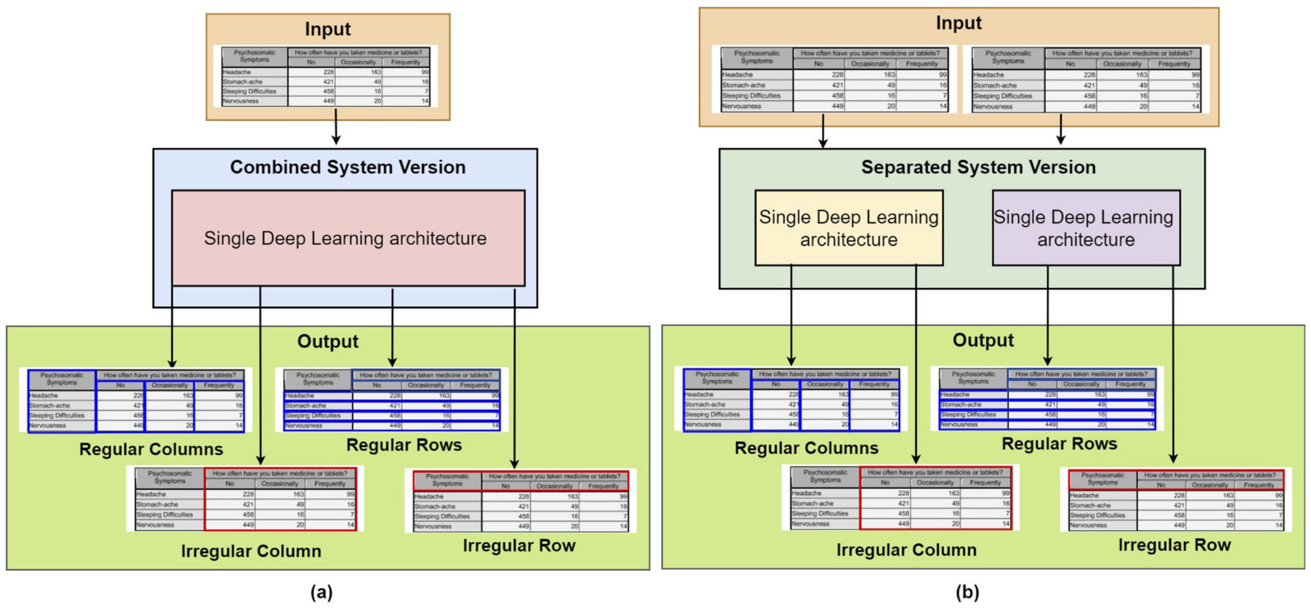


Fig. 3 Combined version **a** and separate versions **b** of the TSR systems

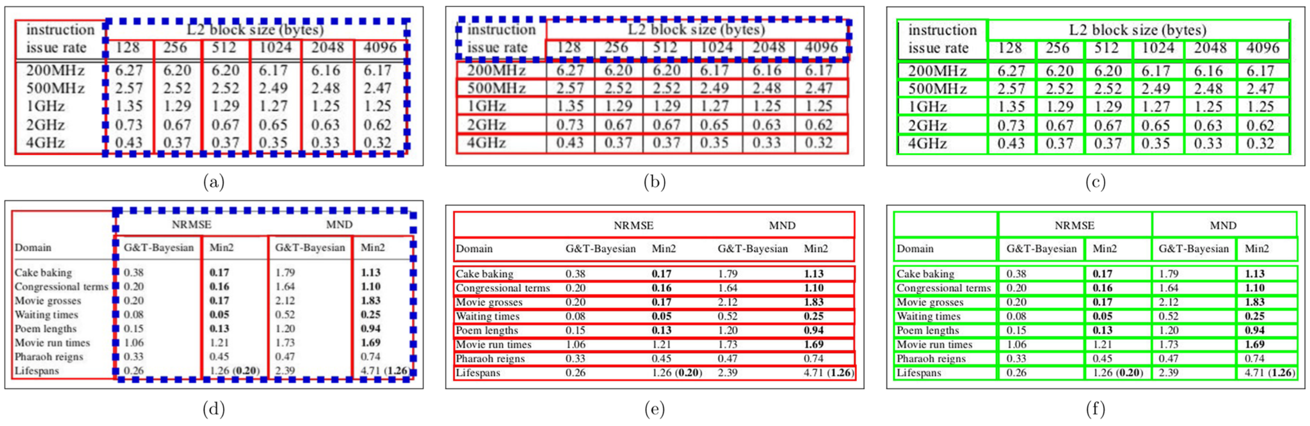
two available datasets and fairly evaluate the row and column detection performance of our presented system with the state-of-the-art solution. In addition, image-to-sequence methods have achieved promising results recently, but these methods usually require a large dataset. Therefore, we also use the FinTabNet dataset to train our model and further transform the outputs into HTML format with a post-processing step and compare the structure-only TEDS scores with state-of-the-art methods.

The TableStrRec system utilizes the Cascade R-CNN architecture with ResNet50 backbone, while the DeepTabStr system utilizes the deformable FPN architecture with a ResNet-101 backbone. We display the difference between the separate and combined versions of these two TSR systems in Fig. 3 and also analyse the impact of utilizing these version types in this section. The separate version uses two deep learning architectures where one architecture detects and classifies columns and another detects and classifies rows. As shown in Fig. 3a in the combined version, a single architecture detects all four objects. For instance, the separate versions of TableDetStr and DeepTabStr utilize two Cascade R-CNN and two deformable FPN architectures along with a ResNet-50 and ResNet-101 backbone, respectively. Similarly, the combined versions of TableDetStr and DeepTabStr utilize a single Cascade R-CNN and a single deformable FPN architecture along with a ResNet-50 and ResNet-101 backbone, respectively, to detect all four object classes. In our analysis, we determine that the separate version of TableStrRec is able to outperform DeepTabStr in detecting and classifying both rows and columns. The results of our approach can be seen in Fig. 4.

4.1 Training details

In this work, we propose detecting and classifying rows as well as columns to address the task of TSR in table images. We train separate and combined versions of the DeepTabStr and our proposed TableStrRec systems to analyse the impact of each version. The separate versions comprise of one architecture trained to detect regular and irregular rows, while the other architecture is trained to detect regular and irregular columns. The combined versions comprise one architecture that is trained to detect all four object classes. The separate versions are individually trained for 60 epochs, whereas the combined versions are trained for 70 epochs. Both versions of each system are trained on the training set of the ICDAR2013 and TableStructDB datasets, as shown in Table 1.

The final inference of both systems irrespective of their versions is the locations and classes of the four object types. TableStrRec has been developed with the Detectron2 library [47]. Furthermore, Group Normalization [48] enhances the detection performance of object detection architectures such as Cascade R-CNN when the training batch size is small. We train the separate and combined versions of TableStrRec with a batch size of 2 images. Thus, we utilize Group Normalization in the architecture of both versions instead of the standard Batch Normalization [49] to improve their detection performance. We utilize a learning rate of 0.002 with an SGD optimizer to train the systems. All versions of TableStrRec and DeepTabStr have been trained on a single NVIDIA Tesla V100 GPU with 32GB RAM. It is worth mentioning that to compare with different types of state-of-the-art TSR methods, we trained the Table-Transformer [28] and TableMaster [50] using FinTab-



instruction issue rate	L2 block size (bytes)					
	128	256	512	1024	2048	4096
200MHz	6.27	6.20	6.20	6.17	6.16	6.17
500MHz	2.57	2.52	2.52	2.49	2.48	2.47
1GHz	1.35	1.29	1.29	1.27	1.25	1.25
2GHz	0.73	0.67	0.67	0.65	0.63	0.62
4GHz	0.43	0.37	0.37	0.35	0.33	0.32

	NRMSE		MND	
Domain	G&T-Bayesian	Min2	G&T-Bayesian	Min2
Cake baking	0.38	**0.17**	1.79	**1.13**
Congressional terms	0.20	**0.16**	1.64	**1.10**
Movie grosses	0.20	**0.17**	2.12	**1.83**
Waiting times	0.08	**0.05**	0.52	**0.25**
Poem lengths	0.15	**0.13**	1.20	**0.94**
Movie run times	1.06	1.21	1.73	**1.69**
Pharaoh reigns	0.33	0.45	0.47	0.74
Lifespans	0.26	1.26 (**0.20**)	2.39	4.71 (**1.26**)

(a) (b) (c) (d) (e) (f)

Fig. 4 Results of TableStrRec on table image. Detected irregular and regular columns in image (**a**) and (**d**) and irregular and regular rows in image (**b**) and (**e**) are shown with irregular type in blue colour dotted line box and regular type in red colour line box, respectively. Result of post-processing step where the intersections of columns and rows from are used to generate cell locations for images (**c**) and (**f**), respectively

Net dataset with their official code bases [46, 51] and obtained their structure-only TEDS scores.

4.2 Results

In this section, we present the improvements in TSR on table images brought on by our proposed TableStrRec system. The system approaches the task of TSR as an object detection task. The system detects four classes of objects; regular rows, irregular rows, regular columns and irregular columns. It detects all four object classes in a single inference to provide the locations of content in a table image, as shown in Fig. 2. As discussed in Sect. 2.1, there are different formulations for the TSR problem, including cell-level, column–row-level and image-to-sequence formulations. Cell-level formulations often use F1 scores on the cell level to evaluate the performance, while column–row-level and image-to-sequence formulations usually use detection metrics and TEDS scores, respectively. Therefore, we utilize the ICDAR2019 Table Detection and Recognition competition [8] evaluation metric to evaluate the TSR performances that are achieved by TableStrRec and the state-of-the-art DeepTabStr method on the two ICDAR2013 and TabStructDB datasets. As per this metric the F1 scores at four IOU values (60, 70, 80, 90) are calculated. Following upon this, the corresponding IOU values are used as the weights to each F1 scores. The product of the weights and the F1 scores are then added to develop the weighted average F1 scores (W. Avg F1).

Besides, to compare with state-of-the-art image-to-sequence methods, we choose structure-only Tree-Edit-Distance-Based Similarity (TEDS) [39] for the FinTabNet dataset, which can overcome the drawbacks of adjacency relation metrics [52], and can be defined as Eq. 6.

$$\mathrm{TEDS}(T_a, T_b) = 1 - \frac{\mathrm{EditDist}(T_a, T_b)}{\max(\mid T_a \mid, \mid T_b \mid)} \tag{6}$$

where EditDist is the tree-edit distance, and T is the number of nodes in the tree. Notably, we focus on the TSR problem without considering the impact of OCR tools that extract text content from images. Therefore, we consider the structure-only TEDS, which means that we only transform the model outputs into HTML tags without extracting the tag contents from the images.

We evaluate the impact of utilizing a deformable convolution [13] backbone instead of a conventional convolution backbone. Furthermore, we examine the detection performance improvement brought on by using separate versions of the systems over combined versions as well. This analysis is conducted on the ICDAR2013 and TabStructDB datasets, as shown in Tables 2 and 3, respectively. All systems are trained to detect regular rows, regular columns, irregular rows and irregular columns. While the systems detect four classes of objects, we measure their detection performance on how well they detect and classify rows and columns as shown in Tables 2 and 3. We infer that the TableStrRec system should utilize two Cascade R-CNN architectures, each with deformable backbones and CIOU loss to accurately detect and classify rows and columns separately. Furthermore, we measure how efficient the separate version of TableStrRec is over DeepTabStr on detecting objects of each of the four classes in Table 4. We also measure the generalization capability of both solutions, as shown in Table 5. To demonstrate the effectiveness of the deformable convolution backbone, we compare it with conventional convolution backbone using the separated version of the proposed method. The experimental results are shown in Table 7.

Table 2 TSR results comparison with state-of-the-art systems on ICDAR2013 test set

Training set	System	Version	Row (regular and irregular)					Column (regular and irregular)				
			F1 score at each IoU				W.Avg	F1 score at each IoU				W.Avg
			60	70	80	90	F1	60	70	80	90	F1
ICDAR2013 (training set)	DeepTabStR [9]	Combined	83.7	74.5	59.5	13.8	54.1	91.7	87.4	80.8	38.9	71.9
		Separated	91.5	88.1	72.5	19.6	64.1	91.3	89.7	87.1	67.1	82.6
	Ours (TableStrRec)	Combined	89.5	87.4	82.5	64.4	79.6	91.1	90.5	89.2	82.3	87.8
		Separated	98.2	96.9	93.9	77.4	90.5	92.1	91.7	89.6	85.2	89.6

Table 3 TSR results comparison with state-of-the-art systems on TabStrDB test set

Training set	System	Version	Row (regular and irregular)					Column (regular and irregular)				
			F1 score at each IoU				W.Avg	F1 score at each IoU				W.Avg
			60	70	80	90	F1	60	70	80	90	F1
TabStructDB (training set)	DeepTabStR [9]	Combined	64.2	56.7	39.7	13.6	40.8	88.4	82.4	68.7	40.1	67.2
		Separated	81.9	76.5	65.8	30.5	60.9	85.9	82.0	72.1	46.2	69.4
	Ours (TableStrRec)	Combined	75.3	73.2	62.2	34.2	59.0	89.0	85.9	78.3	58.0	76.1
		Separated	84.4	82.6	75.3	54.8	72.7	91.2	89.0	81.1	59.4	78.5

Table 4 TSR results comparison for irregular and regular classes of rows and columns on ICDAR2013 and TabStrDB test sets

Training Set	Test Set	System	Row						Column					
			Class	F1 score at each IoU				W.Avg	Class	F1 score at each IoU				W.Avg
				60%	70%	80%	90%	F1 %		60%	70%	80%	90	F1 %
ICDAR 13 (Train set)	ICDAR 13 (Test set)	DeepTabStr[9]	Regular	97.8	92.7	79.0	33.0	72.1	Regular	99.2	96.1	92.0	74.2	89.0
		Ours (TableStrRec)		96.9	94.5	88.4	54.6	81.4		99.2	98.4	94.3	85.4	93.6
		DeepTabStr[9]	Irregular	88.0	88.0	88.0	61.4	80.0	Irregular	85.0	85.0	85.0	75.3	82.1
		Ours (TableStrRec)		99.8	99.8	99.8	99.8	99.8		85.2	85.2	85.2	85.2	85.2
TabStructDB (Train set)	TabStructDB (Test set)	DeepTabStr[9]	Regular	92.2	87.3	73.1	30.1	67.4	Regular	95.8	91.9	78.6	49.1	76.3
		Ours (TableStrRec)		93.5	90.6	78.3	41.2	73.1		97.6	94.0	82.3	54.5	79.7
		DeepTabStr[9]	Irregular	75.4	71.1	68.9	45.3	63.7	Irregular	80.6	76.5	71.9	51.9	68.7
		Ours (TableStrRec)		82.0	81.1	79.1	74.3	78.7		87.4	86.7	82.5	66.3	79.6

Table 5 TSR results comparison for irregular and regular classes of rows and columns on ICDAR2013 and TabStrDB test sets in the cross dataset settings

Training set	Test set	System	Row					Column				
			F1 score at each IoU				W.Avg	F1 score at each IoU				W.Avg
			60	70	80	90	F1	60	70	80	90	F1
ICDAR2013 (training set)	TabStruct	DeepTabStr [9]	49.3	41.4	29.8	11.3	30.9	76.3	68.9	54.0	27.2	53.9
	(test set)	Ours (TableStrRec)	57.3	47.5	34.7	20.9	38.1	80.5	75.3	61.7	39.7	62.0
	TabStruct	DeepTabStr [9]	55.6	46.3	30.2	10.7	33.2	78.3	68.5	56	33.9	56.7
	(complete set)	Ours (TableStrRec)	63.1	55.2	39.3	20.2	42.1	83.6	77.1	66.4	48.0	66.8
TabStructDB (training set)	ICDAR2013	DeepTabStr [9]	82.9	72.3	60.3	12.3	53.2	81.5	76.7	72.2	56.8	70.5
	(test set)	Ours (TableStrRec)	90.8	81.0	69.6	40.6	67.8	85.8	82.7	77.7	69.3	78.0
	ICDAR2013	DeepTabStr [9]	85.0	78.6	60.4	23.5	58.5	86.0	78.0	62.1	43.7	65.1
	(complete set)	Ours (TableStrRec)	89.1	84.5	73.9	40.9	69.5	86.0	80.4	72.3	53.8	71.3

Table 6 Experimental results on FinTabNet dataset with structure-only TEDS score

Model	TEDS-struc.(%)		
	Sim.	Com.	All
EDD [39]	88.40	92.08	90.60
GTE [15]	–	–	87.14
GTE$^{(FT)}$ [15]	–	–	91.02
TableFormer [38]	97.50	96.00	96.80
TableMaster	98.36	98.28	98.32
Table-Transformer*	97.89	97.07	97.47
TableStrRec (Ours)	98.41	98.27	98.34

4.2.1 ICDAR2013 dataset

As shown in Table 2, we evaluate the performance of DeepTabStr [9] and TableStrRec on the ICDAR2013 test set [1]. Both systems are trained on the training set of the ICDAR2013 to detect four classes of objects. We evaluate the impact of utilizing the separate and combined versions of both TableDetStr and DeepTabStr. The separate versions outperform the combined versions, especially on the task of detecting and classifying rows. The separate version of TableStrRec outperforms the separate version of DeepTabStr with a 26.4% and 7% increase in performance in terms of weighted average F1 scores for rows and columns, respectively. At each IOU value, the separate version of TableStrRec achieves a higher F1 scores than the separate version of DeepTabStr in detecting and classifying rows and columns. In Fig. 5a, we display the consistent higher F1 scores that TableStrRec achieves in detecting and classifying rows. In Fig. 5b, we show that TableStrRec can achieve higher F1 scores at all four IOU values, especially at 90% IOU value.

The separate version of the TableStrRec achieves the highest F1 scores as compared to the other systems at all IOU values for both rows and columns. Furthermore, the separate versions of all systems displayed in Table 2 achieve higher F1 scores and weighted average F1 scores for detecting and classifying rows and columns. Thus, we infer that the separate versions of the systems offer a more efficient solution for this problem. In addition we also display the training improvements provided by utilizing CIOU loss as a bounding box regression loss in Fig. 6a. We noted the average regression loss achieved by the separate version of TableStrRec when $l1$ loss or CIOU loss is used in the architecture.

It is worth noting that the architectures which are trained to detect and classify rows and columns with CIOU loss converged faster than the architectures which are trained with $l1$ loss. As shown in Fig. 6a and Table 2, utilizing CIOU loss for bounding box regression in TableStrRec improves its row and column detection performance. We further evaluate how well the separate versions of the TableStrRec and DeepTabStr can detect each of the four object classes in Table 4. TableStrRec is able to achieve the highest weighted average F1 scores for all four object classes, as shown in Table 4. It achieves a 99.8% and 85.2% weighted average F1 scores for irregular rows and columns, respectively. Thus, it can be concluded that TableStrRec can improve the detection performance of not only regular rows and regular columns but also irregular rows and irregular columns.

4.2.2 TabStuctDB dataset

We perform a similar analysis on the TabStructDB dataset to evaluate the performance improvement achieved by our proposed approach. All systems are trained on the TabStructDB training set for this analysis. The separate version of TableStrRec achieves the highest weighted average F1 scores for both detecting and classifying both rows and columns, as shown in Table 3. It outperforms the separate version of DeepTabStr by 11.8% and 9.1% on rows and columns, respectively, in terms of weighted average F1 scores. It also achieves the highest F1 scores on all four IOU values for both detecting and classifying rows and columns as shown in Table 3. We display the results that the separate versions of TableStrRec and DeepTabStr achieve in Fig. 7. At each IOU value, TableStrRec achieves a higher F1 scores in detecting and classifying rows and columns, as shown in Fig. 7a and b.

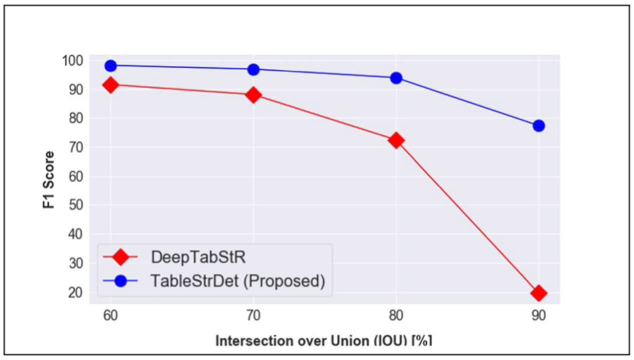
In addition, we also display the quicker convergence enhancement provided by utilizing CIOU loss in the TableStrRec architecture in Fig. 6b. Employing CIOU loss for bounding box regression not only helps TableStrRec converge faster but also improves the detection capability of the system to detect and classify rows and columns, as indicated in Table 3. Similar to our analysis on the ICDAR2013 test set, we conclude that the separate versions of the systems are more efficient solutions for this task on the TabStructDB test set, as shown in Table 3. The separate version of TableStrRec also achieves the highest weighted average F1 scores for not only regular and irregular rows but also regular and irregular columns, as shown in Table 4. Furthermore, while the combined version of TableStrRec provides results in 2 s, the separate version takes just an additional 0.2 s to provide more accurate results.

4.2.3 FinTabNet

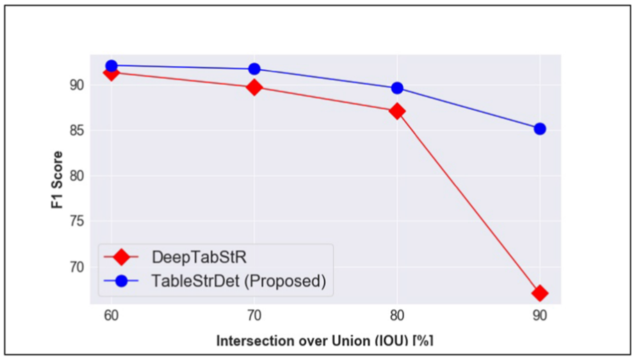
Table 6 shows the experimental results on the FinTabNet dataset, which uses structure-only TEDS scores as the evaluation metric. It is worth mentioning that we re-trained TableMaster [50] and Table-Transformer* [28] based on their official code bases. For the implementation Table-Transformer*, we use Deformable-DETR [53] as the base model, which can achieve better performance and faster divergence. We applied the post-processing method described in Algorithm 1

Table 7 TSR results comparison for conventional CNN backbone and deformable CNN backbone on ICDAR2013 and TabStrDB datasets

Data set	Component	Row					Column				
		F1 score at each IoU				W.Avg	F1 score at each IoU				W.Avg
		60	70	80	90	F1	60	70	80	90	F1
ICDAR2013	Conventional CNN + l1	98.3	94.2	91.3	72.8	87.8	93.8	90.6	88.5	81.1	87.8
	Deformable CNN + l1	97.9	94.5	90.7	67.8	86.2	90.9	90.4	87.9	80.4	86.8
	Conventional CNN + CIoU	98.9	95.8	92.1	77.6	89.9	93.1	91.7	90.7	82.6	89.0
	Deformable CNN + CIoU	98.2	96.9	93.9	77.4	90.5	92.1	91.7	89.6	85.2	89.6
TableStructDB	Conventional CNN + l1	84.3	79.9	71.9	44.8	68.1	87.1	83.3	77.6	56.3	74.4
	Deformable CNN + l1	82.4	80.5	71.6	47.9	68.7	87.5	84.4	77.0	56.9	74.8
	Conventional CNN +CIoU	83.5	81.3	74.1	47.9	69.8	90.0	87.7	80.2	58.3	77.4
	Deformable CNN + CIoU	84.4	82.6	75.3	54.8	72.7	91.2	89.0	81.1	59.4	78.5

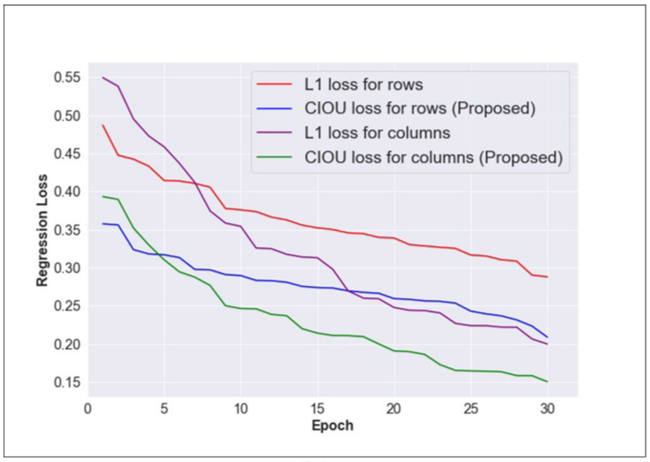


(a) Results for detecting Regular and irregular rows

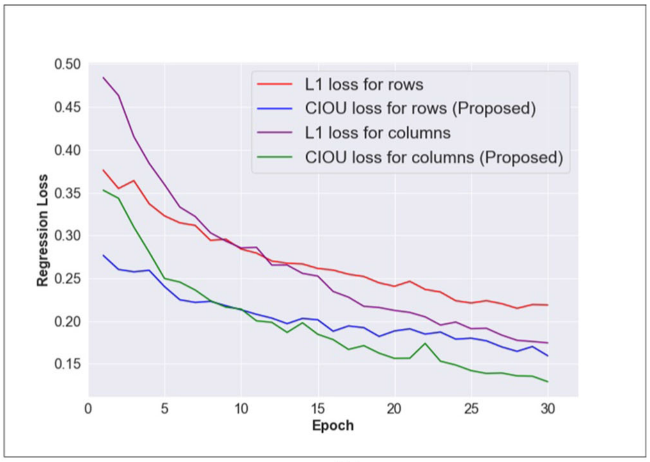


(b) Results for detecting regular and irregular columns

Fig. 5 Comparing detection performances of separate versions of TableStrRec and DeepTabStR on ICDAR2013 test set

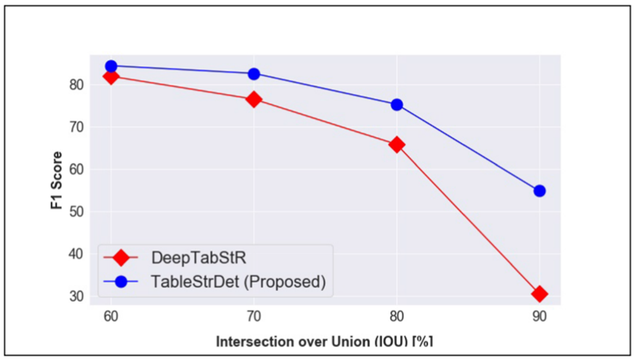


(a)

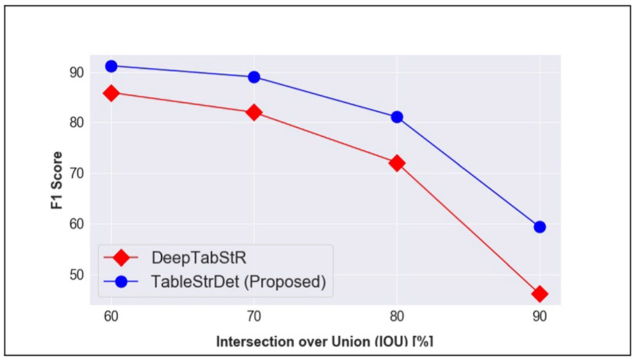


(b)

Fig. 6 L1 loss and CIOU loss comparison at each training epoch achieved by separate versions of TableStrRec for detecting and classifying rows and columns on ICDAR2013 test set (**a**) and TabStructDB test set (**b**)

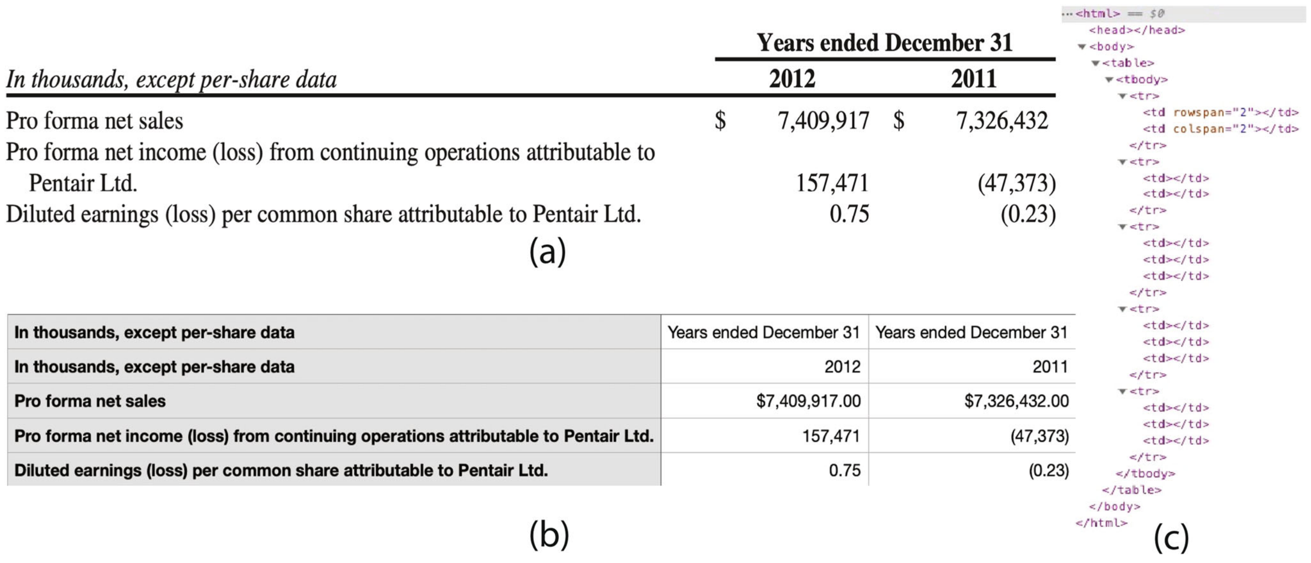


(a) Results for detecting Regular and irregular rows



(b) Results for detecting regular and irregular columns

Fig. 7 Detection performances of separate versions of TableStrRec and DeepTabStR on TabStructDB test set



In thousands, except per-share data	Years ended December 31	
	2012	2011
Pro forma net sales	$ 7,409,917	$ 7,326,432
Pro forma net income (loss) from continuing operations attributable to Pentair Ltd.	157,471	(47,373)
Diluted earnings (loss) per common share attributable to Pentair Ltd.	0.75	(0.23)

(a)

In thousands, except per-share data	Years ended December 31	Years ended December 31
In thousands, except per-share data	2012	2011
Pro forma net sales	$7,409,917.00	$7,326,432.00
Pro forma net income (loss) from continuing operations attributable to Pentair Ltd.	157,471	(47,373)
Diluted earnings (loss) per common share attributable to Pentair Ltd.	0.75	(0.23)

(b)

```
…<html> == $0
   <head></head>
  ▼<body>
    ▼<table>
      ▼<tbody>
        ▼<tr>
            <td rowspan="2"></td>
            <td colspan="2"></td>
          </tr>
        ▼<tr>
            <td></td>
            <td></td>
          </tr>
        ▼<tr>
            <td></td>
            <td></td>
            <td></td>
          </tr>
        ▼<tr>
            <td></td>
            <td></td>
            <td></td>
          </tr>
        ▼<tr>
            <td></td>
            <td></td>
            <td></td>
          </tr>
        </tbody>
      </table>
    </body>
  </html>
```

(c)

Fig. 8 Examples of generated HTML and CSV files. **a** is the original table, **b** is the generated CSV file, and **c** is the generated HTML file. Notably, we filled the cell content to the CSV file to show the structure of the generated table

to obtain table columns, table rows and table spanning cells. Further, we used the post-processing algorithm described in study [28, 46] to generate the CSV and HTML files. Figure 8 shows a generated CSV file and an HTML file whose input image is from the test set of FinTabNet. The experimental results show that our proposed method can outperform column–row-level models and is as competitive as state-of-the-art image-to-sequence models.

4.2.4 Cross dataset

As shown in Table 2, 3 and Figs. 5, 7, utilizing the separate version of TableStrRec provides high-quality rows and columns detection results. To measure the generalization capability of our proposed approach, we conduct the experiment presented by Siddiqui et al. [9], as shown in Table 5. We utilize the separate version of TableStrRec and the separate version of DeepTabStr to measure the generalization potential of both solutions. As we have conducted experiments on ICDAR2013 and TabStructDB datasets in this work, the systems are trained on one dataset and evaluated on the test set of the other dataset to evaluate how well the systems generalize to the test dataset. For instance, we train the systems on ICDAR2013 training set and test their rows and columns detection performance on the TabStructDB test set and TabStructDB complete set. Similarly, as shown in Table 5, we train the systems on TabStructDB training set and test their detection performance on the ICDAR2013 test set and ICDAR2013 complete set. The systems are trained to detect and classify each object in the image into one of the four object classes; regular rows, irregular rows, regular columns and irregular columns.

Furthermore, the table images in both datasets appear in different types of documents. Thus, we conduct this test to evaluate which method would perform better when it is exposed to unseen table images such as tables from data sheets. Due to the lower number of training images and the difference in structures of tables between ICDAR2013 and TabStructDB datasets, the systems provide lower results, especially in the case of detecting rows. However, in the case of detecting columns the systems achieve greater than 50% weighted average F1 scores. In the second case, the systems are trained on the TabStructDB training set and evaluated on the test set of ICDAR2013 and complete set of ICDAR2013. TableStrRec is able to achieve greater than 67% weighted average F1 scores for detecting both rows and columns, on these test sets. At all IOU values, TableStrRec achieves higher F1 scores as compared to DeepTabStr for all four test sets except at 60% IOU value for ICDAR2013 complete set where the two methods share the same F1 scores. Thus, we infer that the separate version of TableStrRec system with two Cascade R-CNN architectures, each with a deformable backbone and CIOU loss not only can improve row and column detection on two public test sets but also can have a higher generalization potential than DeepTabStr (Table 7).

5 Conclusion

A large volume of data is shared between global organizations in tabular format on a daily basis to streamline activities and ensure the efficient functioning of these organizations. Extracting the content of these tables and assembling them for analysis can help such organizations with automation and enables them to take data-driven decisions. The first step to extracting such content is to detect the tables, and the second step is to identify the rows and columns to provide context about the layout of information in the table.

In this work, we have proposed TableStrRec, a deep learning-based approach to detect and classify rows and columns, in table images. TableStrRec comprises one Cascade R-CNN architecture to detect and classify rows as regular and irregular rows and another Cascade R-CNN architecture to detect and classify columns as regular and irregular columns to solve the TSR task. Each Cascade R-CNN architecture employs a deformable convolution backbone and utilizes Complete IOU loss to improve detection performance. With these enhancements, TableStrRec has been shown to achieve 90.5% and 89.6% weighted average F1 scores under the ICDAR2013 test set for rows and columns, respectively. Under the TabStructDB test set, it achieves 72.7% and 78.5% weighted average F1 scores for rows and columns, respectively.

Over the state-of-the-art solution, it achieves greater than 5.7% and 3.1% improvement in detecting and classifying rows and columns, respectively, on both test sets. We have also shown that while detecting four classes of objects, and with some post-processing, TableStrRec can generate accurate cell locations that also capture the hierarchical layouts of tables with merged cells in a single inference. Furthermore, we show that TableStrRec has a higher generalization potential than the state-of-the-art solution and can be adopted to different types of table images. Open issues and opportunities included in the future agenda include exploring the possibility of avoiding empty cells from being generated.

References

1. Göbel, M., Hassan, T., Oro, E., Orsi, G.: Icdar 2013 table competition. In: 12th International Conference on Document Analysis and Recognition, pp. 1449–1453 (2013)
2. Brynjolfsson, E., McElheran, K.: Data in action: data-driven decision making and predictive analytics in U.S. manufacturing. Entrepreneurship & Economics eJournal (2019)
3. Siddiqui, S.A., Malik, M.I., Agne, S., Dengel, A., Ahmed, S.: Decnt: deep deformable cnn for table detection. IEEE Access **6**, 74151–74161 (2018)
4. Prasad, D., Gadpal, A., Kapadni, K., Visave, M., Sultanpure, K.: Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2020)
5. Kara, E., Traquair, M., Simsek, M., Kantarci, B., Khan, S.: Holistic design for deep learning-based discovery of tabular structures in datasheet images. Eng. Appl. Artif. Intell. **90**, 103–551 (2020)
6. Fernandes, J., Simsek, M., Kantarci, B., Khan, S.: Tabledet: an end-to-end deep learning approach for table detection and table image classification in data sheet images. Neurocomputing **468**, 317–334 (2022)
7. Gao, L., Yi, X., Jiang, Z., Hao, L., Tang, Z.: Icdar2017 competition on page object detection. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) **01**, pp. 1417–1422 (2017)
8. Gao, L., Huang, Y., Déjean, H., Meunier, J.L., Yan, Q., Fang, Y., Kleber, F., Lang, E.: Icdar 2019 competition on table detection and recognition (ctdar). In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 1510–1515 (2019). https://doi.org/10.1109/ICDAR.2019.00243
9. Siddiqui, S.A., Fateh, I.A., Rizvi, S.T.R., Dengel, A., Ahmed, S.: Deeptabstr: deep learning based table structure recognition. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 1403–1409 (2019)
10. Hashmi, K.A., Stricker, D., Liwicki, M., Afzal, M.N., Afzal, M.Z.: Guided table structure recognition through anchor optimization. IEEE Access **9**, 113,521-113,534 (2021)
11. Jiang, J., Simsek, M., Kantarci, B., Khan, S.: Tabcellnet: deep learning-based tabular cell structure detection. Neurocomputing **440**, 12–23 (2021)
12. Chi, Z., Huang, H., Xu, H., Yu, H., Yin, W., Mao, X.: Complicated table structure recognition. CoRR arXiv:1908.04729 (2019)
13. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: IEEE International Conference on Computer Vision (ICCV) pp. 764–773 (2017)
14. Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., Ren, D.: Distance-IoU loss: Faster and better learning for bounding box regression. In: AAAI (2020)
15. Zheng, X., Burdick, D., Popa, L., Zhong, P., Wang, N.X.R.: Global table extractor (gte): a framework for joint table identification and cell structure recognition using visual context. In: Winter Conference for Applications in Computer Vision (WACV) (2021)

16. Zanibbi, R., Blostein, D., Cordy, J.: A survey of table recognition. IJDAR **7**, 1–16 (2004). https://doi.org/10.1007/s10032-004-0120-9
17. Liu, Y., Bai, K., Mitra, P., Giles, C.L.: Tableseer: automatic table metadata extraction and searching in digital libraries. In: In Technical Report, pp. 91–100 (2007)
18. Liu, H., Li, X., Liu, B., Jiang, D., Liu, Y., Ren, B., Ji, R.: Show, read and reason: table structure recognition with flexible context aggregator. In: Proceedings of the 29th ACM International Conference on Multimedia, pp. 1084–1092 (2021)
19. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask r-cnn. IEEE International Conference on Computer Vision pp. 2980–2988 (2017)
20. Raja, S., Mondal, A., Jawahar, C.: Table structure recognition using top-down and bottom-up cues. In: European Conference on Computer Vision, Springer, pp. 70–86 (2020)
21. Liu, H., Li, X., Liu, B., Jiang, D., Liu, Y., Ren, B.: Neural collaborative graph machines for table structure recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4533–4542 (2022)
22. Chi, Z., Huang, H., Xu, H.D., Yu, H., Yin, W., Mao, X.L.: Complicated table structure recognition. arXiv preprint arXiv:1908.04729 (2019)
23. Xue, W., Yu, B., Wang, W., Tao, D., Li, Q.: Tgrnet: A table graph reconstruction network for table structure recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1295–1304 (2021)
24. Xiao, B., Simsek, M., Kantarci, B., Alkheir, A.A.: Table structure recognition with conditional attention. arXiv preprint arXiv:2203.03819 (2022)
25. Raja, S., Mondal, A., Jawahar, C.: Visual understanding of complex table structures from document images. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2299–2308 (2022)
26. Ichikawa, K.: Image-based relation classification approach for table structure recognition. In: International Conference on Document Analysis and Recognition, Springer, pp. 632–647 (2021)
27. Long, R., Wang, W., Xue, N., Gao, F., Yang, Z., Wang, Y., Xia, G.S.: Parsing table structures in the wild. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 944–952 (2021)
28. Smock, B., Pesala, R., Abraham, R.: Pubtables-1m: Towards comprehensive table extraction from unstructured documents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4634–4642 (2022)
29. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision, pp. 764–773 (2017)
30. Qiao, L., Li, Z., Cheng, Z., Zhang, P., Pu, S., Niu, Y., Ren, W., Tan, W., Wu, F.: Lgpma: complicated table structure recognition with local and global pyramid mask alignment. In: International Conference on Document Analysis and Recognition, Springer, pp. 99–114 (2021)
31. Zhang, Z., Zhang, J., Du, J., Wang, F.: Split, embed and merge: an accurate table structure recognizer. Pattern Recognit. **126**, 108–565 (2022)
32. Tensmeyer, C., Morariu, V.I., Price, B., Cohen, S., Martinez, T.: Deep splitting and merging for table structure decomposition. In: 2019 International Conference on Document Analysis and Recognition (ICDAR) (IEEE), pp. 114–121 (2019)
33. Zhang, J., Elhoseiny, M., Cohen, S., Chang, W., Elgammal, A.: Relationship proposal networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5678–5686 (2017)
34. Lin, W., Sun, Z., Ma, C., Li, M., Wang, J., Sun, L., Huo, Q.: Tsrformer: table structure recognition with transformers. In: Proceedings of the 30th ACM International Conference on Multimedia, pp. 6473–6482 (2022)
35. Ma, C., Lin, W., Sun, L., Huo, Q.: Robust table detection and structure recognition from heterogeneous document images. Pattern Recognit. **133**, 109,006 (2023)
36. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems **30**, (2017)
37. He, Y., Qi, X., Ye, J., Gao, P., Chen, Y., Li, B., Tang, X., Xiao, R.: Pingan-vcgroup's solution for icdar 2021 competition on scientific table image recognition to latex. arXiv preprint arXiv:2105.01846 (2021)
38. Nassar, A., Livathinos, N., Lysak, M., Staar, P.: Tableformer: table structure understanding with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4614–4623 (2022)
39. Zhong, X., ShafieiBavani, E., Jimeno Yepes, A.: Image-based table recognition: data, model, and evaluation. In: European Conference on Computer Vision, Springer, pp. 564–580 (2020)
40. Schreiber, S., Agne, S., Wolf, I., Dengel, A., Ahmed, S.: Deepdesrt: deep learning for detection and structure recognition of tables in document images. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR) **01**, pp. 1162–1167 (2017)
41. Cai, Z., Vasconcelos, N.: Cascade R-CNN: delving into high quality object detection. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 6154–6162 (2018)
42. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR arXiv:1512.03385 (2015)
43. Lin, T., Maire, M., Belongie, S.J., Bourdev, L.D., Girshick, R.B., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. CoRR arXiv:1405.0312 (2014)
44. Yu, J., Jiang, Y., Wang, Z., Cao, Z., Huang, T.S.: Unitbox: an advanced object detection network. CoRR arXiv:1608.01471 (2016)
45. Paliwal, S., Vishwanath, D., Rahul, R., Sharma, M., Vig, L.: Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In: 2019 International Conference on Document Analysis and Recognition (ICDAR) pp. 128–133 (2019)
46. Smock, B., Pesala, R.: Table Transformer. https://github.com/microsoft/table-transformer (2021)
47. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019)
48. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV), pp. 3–19 (2018)
49. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning (PMLR), pp. 448–456 (2015)
50. Ye, J., Qi, X., He, Y., Chen, Y., Gu, D., Gao, P., Xiao, R.: Pingan-vcgroup's solution for icdar 2021 competition on scientific literature parsing task b: table recognition to html. arXiv preprint arXiv:2105.01848 (2021)
51. He, Y., Qi, X., Ye, J., Gao, P., Chen, Y., Li, B., Tang, X., Xiao, R.: TableMASTER-mmocr https://github.com/JiaquanYe/TableMASTER-mmocr (2021)
52. Hurst, M.: A constraint-based approach to table structure derivation. In: Seventh International Conference on Document Analysis and Recognition, 2003. vol. 3, IEEE Computer Society, pp. 911–911 (2003)
53. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)