TGRNet: A Table Graph Reconstruction Network for Table Structure Recognition

Wenyuan Xue[*], Baosheng Yu[†], Wen Wang[*], Dacheng Tao[‡†], Qingyong Li[*]

[*]Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, China
[†]The University of Sydney, Australia
[‡]JD Explore Academy, China

{wyxue17,wangwen,liqy}@bjtu.edu.cn, baosheng.yu@sydney.edu.au, dacheng.tao@gmail.com

Abstract

A table arranging data in rows and columns is a very effective data structure, which has been widely used in business and scientific research. Considering large-scale tabular data in online and offline documents, automatic table recognition has attracted increasing attention from the document analysis community. Though human can easily understand the structure of tables, it remains a challenge for machines to understand that, especially due to a variety of different table layouts and styles. Existing methods usually model a table as either the markup sequence or the adjacency matrix between different table cells, failing to address the importance of the logical location of table cells, e.g., a cell is located in the first row and the second column of the table. In this paper, we reformulate the problem of table structure recognition as the table graph reconstruction, and propose an end-to-end trainable table graph reconstruction network (TGRNet) for table structure recognition. Specifically, the proposed method has two main branches, a cell detection branch and a cell logical location branch, to jointly predict the spatial location and the logical location of different cells. Experimental results on three popular table recognition datasets and a new dataset with table graph annotations (TableGraph-350K) demonstrate the effectiveness of the proposed TGRNet for table structure recognition. Code and annotations will be made publicly available at https://github.com/xuewenyuan/TGRNet.

1. Introduction

Tabular data have been widely used to help people manage and extract important information in many real-world scenarios, including the analysis of financial documents, air pollution indices, and electronic medical records [19, 33]. Though human can easily understand tables with different

---

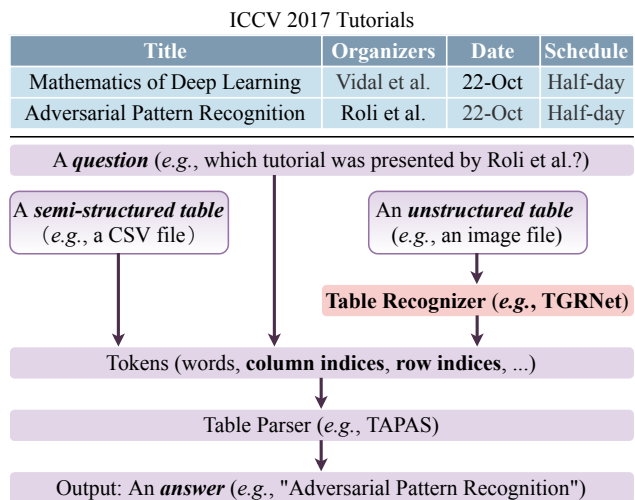*Qingyong Li and Wen Wang are the corresponding authors.

Figure 1. A table parsing example shows the main concern of this paper. Before applying a table parser (*e.g.*, TAPAS [7]) to answer a question over the above table, both the table and the question are represented as a sequence of tokens, in which the cell logical location (*i.e.*, the index of column/row) provides the table structure information. Though the cell logical location can be directly acquired from the semi-structured table (*e.g.*, a CSV file), a table structure recognizer is required to obtain such important information from the unstructured table (*e.g.*, an image file).

layouts and styles, it remains a great challenge for machines to automatically recognize the structure of various tables. Considering the massive amount of tabular data presented in unstructured formats (*e.g.*, image and PDF files) and that most table analysis methods focus on semi-structured tables (*e.g.*, CSV files) [7, 33, 19, 32], the community will significantly benefit from an automatic table recognition system, facilitating large-scale tabular data analysis such as table parsing [7, 32], patient treatment prediction [33, 34], and credit card fraud detection [19].

To understand the structure of different tables, both the cell spatial and logical locations are of great importance

in many applications. As a table parsing example shown in Fig. 1, before applying a table parser (*e.g.*, TAPAS [7]) to answer a question over a table, both the table and the question are tokenized, and the cell logical location (*i.e.*, the index of column/row) provides the table structure information. If the table is presented as an image instead of structured or semi-structured formats, a table structure recognizer is required to detect the cell spatial location and infer the cell logical location. Existing table structure recognition methods usually utilize rule-based or statistical techniques with hand-crafted features [18, 23, 26], working well in only constrained settings (*e.g.*, tables with fixed layouts). As shown in Fig. 2, with the success of deep learning, recent deep learning-based table structure recognition approaches can be divided into three categories: 1) identify cell bounding boxes through visual detection and segmentation methods [25, 27, 29, 28, 8]; 2) transform a table image into the markup sequence, such as LaTeX and HTML [10, 2]; and 3) explore the adjacency relation between different table cells [30, 22, 11]. Though the logical location of each cell can be inferred from the adjacency matrix of table cells, additional complex graph optimization algorithms are required. As a result, the importance of the logical location of table cells has been poorly investigated in existing table structure recognition methods.

To further explore the logical relation between different table cells, we introduce a more powerful graph-based table representation, which is referred to as *Table Graph*. Specifically, the structure of each table can be represented as a graph: each node indicates a table cell and the edge between two nodes reflects their logical relation on the row and column dimensions, which can be related to their row and column indices. With the proposed table graph, a table cell can be located in the image by the position of its pixels, and its relevant information can be retrieved along the row and column indices. As a result, if a model can reconstruct such a table graph from the given image, it then has a good understanding on the table structure. As the table parsing example shown in Fig. 1, the table structure is represented by the logical location of each cell, which can be inferred from the cell spatial location when the input table is presented as an image. To this end, both the cell spatial and logical locations are important for table structure recognition and further table understanding.

In this paper, we formulate the problem of table structure recognition as the table graph reconstruction, which requires the model to jointly predict the cell spatial location and the cell logical location. To address this problem, an end-to-end trainable table reconstruction network (TGRNet) is proposed. Specifically, the proposed method employs a segmentation-based module to detect the cell spatial location, and the cell logical location prediction is solved as an ordinal node classification problem. We evaluate the

(a) Cell bounding box detection.

(b) Table image to markup sequence.

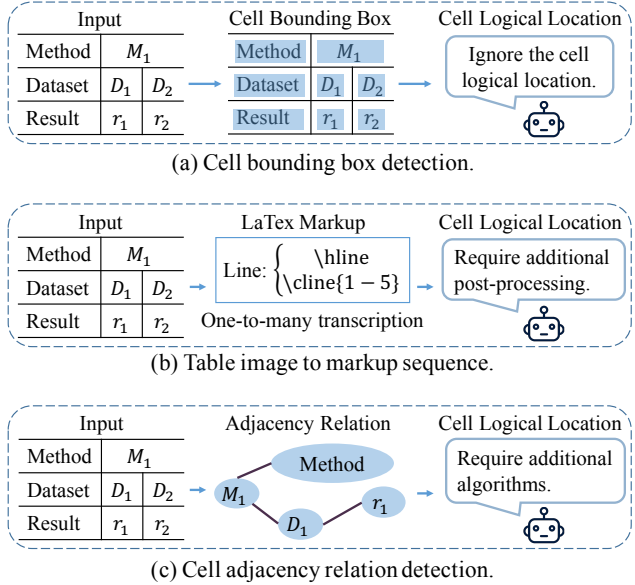(c) Cell adjacency relation detection.

Figure 2. Three types of existing methods for table structure recognition.

proposed method on four datasets and experimental results demonstrate the effectiveness of the proposed method for table structure recognition. Considering most table recognition datasets do not provide the cell logical location annotation, we provide the table graph annotations for 350K table images from the TABLE2LATEX-450K dataset [2] as a new benchmark, TableGraph-350K. The contributions of this paper are summarized as follows:

- We reformulate the problem of table structure recognition as the table graph reconstruction, and further propose a table graph reconstruction network to jointly predict the spatial and logical locations of table cells.

- We contribute a new benchmark generated from the TABLE2LATEX-450K dataset with 350K table graph annotations.

2. Related Work

2.1. Cell Detection and Segmentation

Inspired by recent work on semantic segmentation and object detection, some researchers utilized deep learning techniques to detect table cells. DeepDeSRT [25] is a 2-fold system that applies Faster RCNN [24] and FCN [16] for both table detection and row/column segmentation. Paliwal *et al.* [20] proposed an end-to-end deep model with one encoder and two decoders for both table and column segmentation. In [8, 28, 29], they classify an entire row or column into the cell or non-cell categories instead of the pixel-wise classification. Siddiqui *et al.* [27] treated the

row/column identification as an object detection problem. Prasad *et al.* [21] used a cascade architecture for both table detection and cell segmentation. These work well explore deep vision methods for the cell spatial location detection, while ignore the cell logical location.

2.2. Table to Markup Sequence

From the perspective of natural language processing, other researchers tried to convert a table image into the markup sequence (*e.g.*, LaTeX or HTML) [10, 2]. They usually applied an image-to-sequence model that includes an encoder to extract features and a decoder to produce the label sequence. Ideally, the structure of a table can be recognized by parsing the markup sequence. However, the markup sequence contains diversified commands for different styles, which make the table structure can be transcribed into different markup sequences. This one-to-many mapping brings a lot of noise to the ground truth and impedes the model training. Even so, they contribute several large datasets by collecting data from public arXiv articles.

2.3. Adjacency Relation Detection

As cell detection and segmentation approaches do not consider the cell logical location, some research begin to use the graph structure to explore the relation between different table cells. Generally, these methods can be divided into the edge classification [1, 22, 11, 30] and the node classification [15]. Edge classification methods identify whether two different candidate cells belong to the same cell, row, or column. Node classification methods try to predict the category (*e.g.*, "date" or "price") of a candidate cell on a specific domain. Xue *et al.* [30] combined these two kinds of methods. They utilized the neighbor relation among cells to infer the cell logical location. However, most of these methods only explore the cell adjacency relation. When searching information on a table, complex graph optimization algorithms are required to infer the global logical relation from the pair-wise cell adjacency relation.

3. Problem Formulation

In this section, we introduce the formulation of the proposed table graph representation. A table graph is the structured representation of a table in an image, which can be defined as $G = (V, A)$, where each node in V indicates a table cell and A is the adjacency matrix. Generally, each element in A represents the relation of two different nodes. However, for a table, the logical relation among cells can be represented by their logical indices. Therefore, for each cell or node v_i, we denote b_i and l_i as its two attributes that indicate its spatial and logical locations, respectively,

$$b_i = (b_i^x, b_i^y, b_i^w, b_i^h), \qquad (1)$$

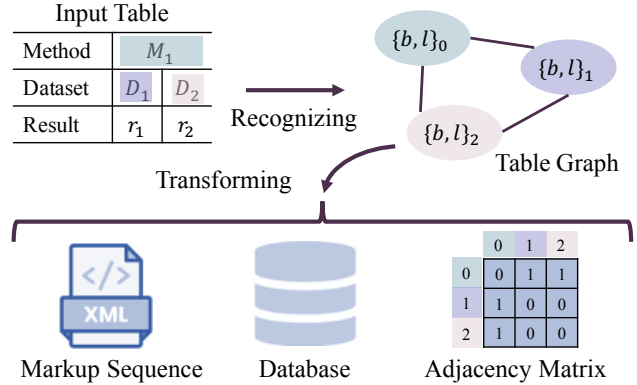$$l_i = (row_i^{start}, row_i^{end}, col_i^{start}, col_i^{end}), \qquad (2)$$

Figure 3. Data transformation from a table graph to other data formats. Three table cells are visualized for simplicity. The adjacency matrix indicates whether two cells belong to the same column.

where (b_i^x, b_i^y), b_i^w, and b_i^h indicate the centre point, width, and height of the bounding box of v_i, respectively, and the logical location l_i is composed of its four logical indices, *i.e.*, the start-row, end-row, start-column, and end-column. Because table cells arrange in a two-dimensional space, the adjacency matrix A can be represented based on the Euclidean distance between two different nodes. In this way, we can employ the popular graph convolutional network to learn the graph representation and predict the cell logical location through a node classification way.

Given a table image, the purpose of table structure recognition then is to reconstruct such a table graph, on which we can locate a cell by its spatial location and retrieve the relevant information according to its logical location, just like that we edit formulae in Microsoft Excel. Different from most existing methods, the table graph can be seen as the metadata of a table structure. Combined with other techniques, the table graph can be transformed into multiple representations in different scenarios. As shown in Fig. 3, with the logical location, we can directly build an adjacency matrix to represent the neighbor relation or the same-row and same-column relation between two cells. Followed by an Optical Character Recognition (OCR) engine, the table graph can be transcribed into an XML file or even a database format. On the contrary, though the pair-wise relation between cells or the markup sequence may be suitable in some scenarios, it usually fails to generalize because a complex algorithm is required to infer the global structure on a table.

4. Method

In this section, we first describe the main framework of the proposed TGRNet for table structure recognition. We then introduce two main components of TGRNet: cell spatial location detection and cell logical location prediction.
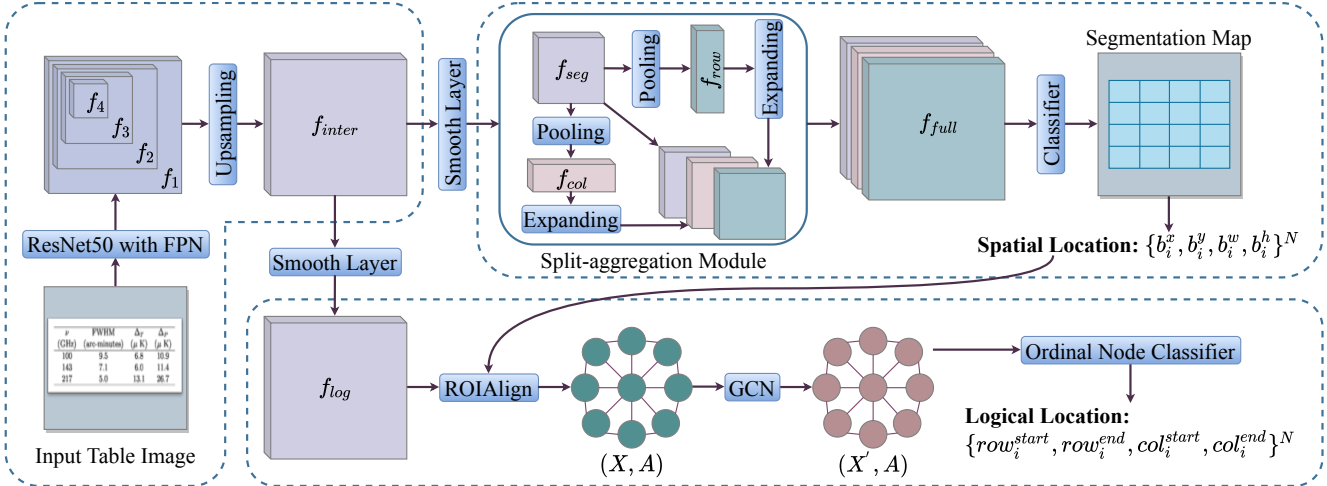
Figure 4. The main framework of the proposed table graph reconstruction network or TGRNet.

In order to achieve the purpose of reconstructing the table graph from a table image, we devise a table graph reconstruction network (TGRNet) and the main framework is shown in Fig. 4. Specifically, we first employ a backbone network, *e.g.*, ResNet-50 [6] with FPN [12], to extract multi-scale feature representations from the input table image. We then jointly perform cell spatial location detection and logical location prediction via two separate head branches in a multi-task manner. For cell spatial location, we first utilize the widely used segmentation-based method to generate a cell segmentation map, from which cells are then detected by computing bounding boxes of connected components. For cell logical location, we apply a graph convolutional network (GCN) [9] to learning the table graph representation and solve it as an ordinal node classification problem. In addition, we combine a typical ordinal regression loss with the focal loss [13] as the objective function to address the imbalance problem in cell logical location prediction. We introduce details of cell spatial and logical locations in Section 4.1 and 4.2, respectively.

4.1. Cell Spatial Location Detection

Recently, segmentation-based methods have been popular in table cell detection due to its statistical significance along table rows and columns [25, 27, 28]. Therefore, we use a segmentation-based module for the cell spatial location branch to detect the bounding boxes of table cells as follows.

Let $I \in \mathbb{R}^{3 \times H \times W}$ denote an input table image, where H and W indicate the height and width of the input image, respectively. We use four feature maps, f_1, f_2, f_3, and f_4, from the ResNet-50 with the stride $s = 4, 8, 16, 32$ to construct the feature pyramid [12]:

$$f_{inter} = \mathcal{U}_{\times 4}(\mathcal{C}(f_1, \mathcal{U}_{\times 2}(f_2), \mathcal{U}_{\times 4}(f_3), \mathcal{U}_{\times 8}(f_4))), \quad (3)$$

where the μ in $\mathcal{U}_{\times \mu}(\cdot)$ is the upsampling scale, $\mathcal{C}(\cdot)$ indicates the channel-wise concatenation operation, all feature maps f_1, f_2, f_3, and f_4 are converted to 256 channels via smooth layers. We then have the multi-scale feature representation $f_{inter} \in \mathcal{R}^{1024 \times H \times W}$ as the input of the cell spatial location branch.

To reduce the computational complexity, the cell spatial location branch first reduces the input channels from 1024 to 256 using a 1×1 convolutional layer, *i.e.*, $f_{seg} \in \mathbb{R}^{256 \times H \times W}$. Considering that tabular data are arranged in rows and columns, we further introduce a split-aggregation module to utilize the statistical information of row- and column-wise representations. Specifically, the row- and column-wise features, $f_{row} \in \mathbb{R}^{256 \times H}$ and $f_{col} \in \mathbb{R}^{256 \times W}$, are obtained by using the $1 \times W$ and $H \times 1$ average pooling layers, respectively. The row- and column-wise feature representations f_{row} and f_{col} are then expanded to concatenate with the pixel-wise features, *i.e.*,

$$f_{full} = \mathcal{C}(f_{row}, f_{col}, f_{seg}), \quad (4)$$

which is then used to obtain the segmentation map $\hat{y}_{full} \in \mathbb{R}^{K \times H \times W}$, where K indicates the number of classes (*i.e.*, "background", "cell", and "boundary"). During the training stage, besides $\hat{y}_{full}$, we also use f_{row} and f_{col} to predict the row- and column-wise segmentation maps, $\hat{y}_{row} \in \mathbb{R}^{K \times H}$ and $\hat{y}_{col} \in \mathbb{R}^{K \times W}$, which can be seen as a kind of statistical regularization. During the testing stage, only $\hat{y}_{full}$ is used to acquire the cell spatial locations by computing the minimum rectangular bounding box for each connected component on the segmentation map.

4.2. Cell Logical Location Prediction

We choose candidate table cells from the detected cell bounding boxes to initialize the table graph G. A graph

convolutional network is then used to learn effective graph representations. Considering that the prediction of logical indices can be seen as a classification with ranking problem, we thus formulate the problem of logical location prediction as an ordinal node classification problem.

We introduce the construction of the table graph G in detail as follows. During training, the node set V consists of all candidate table cells that have the intersection-over-union (IoU) overlaps with the ground truth table cells larger than 0.5. The feature of the i-th node x_i contains two parts: 1) the spatial feature $x_i^s \in \mathbb{R}^{256}$ extracted from the spatial location branch; and 2) $x_i^v \in \mathbb{R}^{1024}$ obtained by the RoIAlign operation [5] with the 2×2 output size from the smoothed multi-scale feature representation $f_{log} \in \mathbb{R}^{256 \times H \times W}$ according to the bounding box b_i of the i-th table cell. We then have the node feature $x_i = \mathcal{C}(x_i^v, x_i^s)$. For the adjacency matrix A, each element $a_{i,j}$ indicates the undirected edge between the i-th and j-th nodes ($i \neq j$). In order to further explore the spatial relation of cells from the row and column dimensions, $a_{i,j}$ is defined as a pair $\{a_{i,j}^{row}, a_{i,j}^{col}\}$, which is calculated based on the Euclidean distance between two nodes:

$$\begin{cases} a_{i,j}^{row} &= \exp\{-(\frac{b_i^y - b_j^y}{H} \cdot \alpha)^2\}, \\ a_{i,j}^{col} &= \exp\{-(\frac{b_i^x - b_j^x}{W} \cdot \alpha)^2\}, \end{cases} \quad (5)$$

where the adjustment factor α should increase with the growing number of rows or columns, which assigns a large weight to the edges of neighbor cells. After the initialization of the table graph, we then apply a GCN for message passing,

$$X' = \text{ReLU}(\text{GCN}(X, A)), \quad (6)$$

where X and X' indicate the input and output node feature matrixes, respectively. Because the GCN architecture does not support multi-dimensional representations for edges, in our implementation, we use a pair of parallel GCNs to update the node feature matrix for row and column indices prediction, respectively.

With the table graph representation X', we then use an ordinal classifier to predict the cell logical location as follows. Considering that the logical indices of rows and columns in $l_i = (row_i^{start}, row_i^{end}, col_i^{start}, col_i^{end})$ are the same problem, for simplicity, we do not distinguish them during the description of the ordinal node classification. Let $r_i \in \{0, 1, ..., T-1\}$ denote the logical index label of the i-th node, where T is the total number of rows (or columns). We first convert r_i into a binary label vector $q_i \in \mathbb{R}^{T-1}$ as follows,

$$q_i^t = \begin{cases} 1, & \text{if}(t < r_i), \\ 0, & \text{otherwise}. \end{cases} \quad (7)$$

In this way, the logical index prediction is transformed into $T-1$ binary classification sub-problems. The loss function
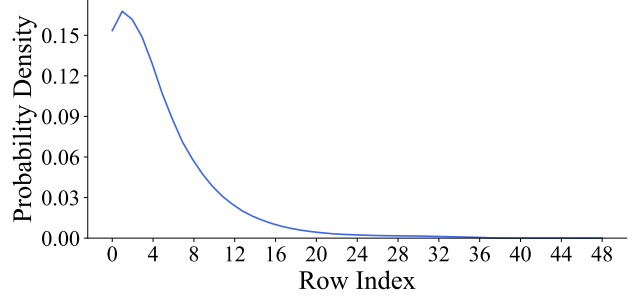

Figure 5. The probability distribution of row indices in TableGraph-350K dataset.

of all nodes can be defined as:

$$\mathcal{L}(X', \Theta) = -\frac{1}{N} \sum_{i=1}^{N} \Psi(x_i', \Theta), \quad (8)$$

$$\Psi(x_i', \Theta) = \sum_{t=0}^{r_i-1} \log(p_i^t) + \sum_{t=r_i}^{T-2} \log(1 - p_i^t), \quad (9)$$

where N is the number of nodes and p_i^t indicates the predicted probability that r_i is larger than t.

However, the distribution of the cell logical location is usually long-tailed. As shown in Fig. 5, the number of small indices is much larger than that of large indices. Inspired by the focal loss [13, 14], we address the problem of long-tailed cell logical locations in a similar way,

$$\begin{aligned} \Psi(x_i', \Theta) = &\sum_{t=0}^{r_i-1} (1 - p_i^t)^{\gamma_t} \log(p_i^t) + \\ &\sum_{t=r_i}^{T-2} (1 - p_i^t)^{\gamma_t} \log(1 - p_i^t), \end{aligned} \quad (10)$$

$$\gamma_t = \min(2, -(1 - \lambda_t)^2 \log(\lambda_t) + 1), \quad (11)$$

where λ_t is the statistical probability of the logical index t on the training set. The predicted logical index of the i-th node is the sum of all $T-1$ binary classification results without explicitly ensuring the consistency among the different classifiers [17].

5. TableGraph-350K

Existing table recognition datasets usually have very limited tabular data or lack of the cell logical location labels. Therefore, to build a large-scale benchmark for the table graph reconstruction task, we collect tabular data from the TABLE2LATEX-450K dataset [2] and provide table graph annotations to generate a new dataset with more than 350K tables, which is referred to as TableGraph-350K. For evaluation, we use the same train/val/test split provided by the original dataset. Finally, the new dataset contains 358,767

tables, including 343,988 tables for training, 7,420 tables for validation, and 7,359 tables for testing. The maximum indices of rows and columns are 48 and 27, respectively.

6. Experiments

In this section, we first introduce the datasets, evaluation metrics, and baseline methods. We then present the overall performance of TGRNet for table structure recognition. To demonstrate the robustness of the proposed method and further analyze the limitation of the adjacency relation-based evaluation, we also conduct experiments on incomplete table graphs and challenging historical documents, respectively. Lastly, we perform ablation studies to demonstrate the effectiveness of the main components in TGRNet.

6.1. Datasets

- **TableGraph-24K**. Considering the computational complexity for training a variety of models, we also randomly choose a subset of TableGraph-350K for the academic community, which is referred to as TableGraph-24K. Specifically, this subset contains 20,000 tables for training, 2,000 tables for validation, and 2,000 tables for testing. The maximum indices of rows and columns are 37 and 21, respectively.

- **CMDD** [31]. This is a medical laboratory report dataset including 476 tables (372 for training and 104 for test). The maximum indices of rows and columns are 24 and 5, respectively. There is no spanning cell and empty cells without texts are not annotated.

- **ICDAR13-Table** [4]. This dataset consists of 156 tables with spanning cells and other various styles. Empty cells without texts are not annotated. The maximum indices of rows and columns are 57 and 12, respectively. Because the original dataset does not specify the training and test sets, we use half tables for training and others for test following the setting in [30].

- **ICDAR19-cTDaR (TrackB1)** [3]. This dataset contains 750 pages from archival historical documents, from which 881 tables are extracted (679 for training and 202 for test). The maximum indices of rows and columns are 87 and 43, respectively. The largest table in this dataset includes more than 2,000 cells.

6.2. Evaluation Metrics

For cell spatial location detection, we use the same evaluation metrics with recent methods [25, 27, 29, 28, 20]. The predicted cell boxes are evaluated by using the *Precision (P)*, *Recall (R)*, and *Hmean (H)* with the IoU threshold 0.5.

For cell logical location prediction, we follow the metrics in [30] to calculate the accuracy of four logical in-

dices (*i.e.*, the start-row, end-row, start-column, and end-column) based on the detected table cells, which are denoted as A_{rowSt}, A_{rowEd}, A_{colSt}, and A_{colEd}, respectively. We also evaluate the overall accuracy A_{all} that all four logical indices are predicted correctly for each detected table cell. For using F-Score, β is set to be 0.5 to encourage more candidate cell boxes in practice:

$$F_{\beta=0.5} = \frac{(1 + 0.5^2) \cdot H \cdot A_{all}}{0.5^2 \cdot H + A_{all}}. \tag{12}$$

In Section 6.6, we also report experimental results using the cell adjacency relation-based metric, *i.e.*, the weighted average F-Score (*WAF*)[1] [3]. For each table cell, the adjacency relations are generated with its nearest neighbours at four directions, *e.g.*, "up", "down", "left", and "right". *WAF* calculates the weighted F-Score with $\beta = 1$ for the cell adjacency relation under different IoU thresholds:

$$WAF = \frac{\sum_{i=1}^{4} IoU_i \cdot F_{\beta=1}@IoU_i}{\sum_{i=1}^{4} IoU_i}, \tag{13}$$

where $IoU = \{0.6, 0.7, 0.8, 0.9\}$.

6.3. Baseline Methods

For cell logical location prediction, we compare the proposed method with ReS2TIM [30]. To the best of our knowledge, in previous methods, ReS2TIM is the only one that aims to predict the cell logical location and reports the evaluation results. Different from TGRNet, ReS2TIM does not contain a cell spatial location module.

For cell spatial location detection, many methods have emerged recently as introduced in Section 2.1. However, most of them were evaluated under different experimental settings due to the lack of standard benchmarks. For example, the DeepDeSRT [25] model was evaluated on a random subset of ICDAR13-Table. Tensmeyer *et al.* [29] trained the SPLERGE model on a private dataset and evaluated it on ICDAR13-Table by randomly choosing a subset as well. In the literatures [28, 27], methods were evaluated on ICDAR13-Table with table cell boxes instead of the original text-level bounding box annotations. It should be noticed that a cell box is larger than the bounding box of the text within that cell. Therefore, we do not present a comparable experiment with these methods for cell spatial location detection because it is hard to do a fair comparison.

6.4. Overall Performance

As an end-to-end solution, we present the overall performance of TGRNet in this subsection. ICDAR13-Table and TableGraph-24K are employed for evaluation. The adjustment factor α in Eq. (5) is set to be 3. Each input image

---

[1]We denote the weighted average F-Score as *WAF* for short instead of *WAvg. F1* in [3].

Table 1. The overall performance of TGRNet for end-to-end table graph reconstruction.

Dataset	Cell Spatial Location			Cell Logical Location					$F_{\beta=0.5}$
	P	R	H	A_{rowSt}	A_{rowEd}	A_{colSt}	A_{colEd}	A_{all}	
ICDAR13-Table	0.682	0.652	0.667	0.445	0.445	0.700	0.692	0.275	0.519
TableGraph-24K	0.916	0.895	0.906	0.917	0.916	0.919	0.923	0.832	0.890

Table 2. Experimental results with the metric A_{all} for robustness analysis.

Method	CMDD			ICDAR13-Table		
	100% cells	90% cells	80% cells	100% cells	90% cells	80% cells
ReS2TIM [30]	0.999	0.941	0.705	0.174	0.137	0.124
TGRNet	0.995	0.955	0.857	0.334	0.314	0.314

is resized into 480×480 pixels. Because the cell logical location branch requires the cell bounding boxes from the cell spatial location branch to extract the corresponding features, we adopt a pre-training strategy to accelerate the training process. Specifically, before training the whole model on TableGraph-24K, we pre-train TGRNet for 50 epochs while freezing the cell logical location branch. When training on ICDAR13-Table, the model is initialized with the parameters trained on TableGraph-24K.

Experimental results are presented in Table 1. The proposed model achieves 0.519 and 0.890 for $F_{\beta=0.5}$ on ICDAR13-Table and TableGraph-24K, respectively. Although the model has been pre-trained on TableGraph-24K, for ICDAR13-Table, TGRNet does not perform as well as on TableGraph-24K. The possible reasons may be the insufficient data of ICDAR13-Table (only 78 tables for training and the rest 78 tables for test) and the distributional difference between ICDAR13-Table and TableGraph-24K.

6.5. Robustness Analysis

Graph-based methods involve message passing between adjacent nodes. If some key nodes are missing, the inference on the graph will be affected. For the end-to-end table graph reconstruction, the proposed model first detects cell boxes from the input image and then treats them as nodes to predict their logical locations. However, achieving a perfect cell detection result is not easy. In order to evaluate the robustness of TGRNet on incomplete table graphs, we conduct experiments by randomly removing some nodes from the table graph. CMDD and ICDAR13-Table are used for evaluation. Both two datasets do not contain the annotation of empty cells, which means the ground truth may be an incomplete table graph. Tables in CMDD have two kinds of layouts, and TGRNet can achieve 0.991 *Hmean* and 0.992 A_{all} for the end-to-end table graph reconstruction on this dataset. Because ReS2TIM [30] does not contain a cell spatial location module, for a fair comparison, we use the ground truth of cell spatial locations instead.

As shown in Table 2, "100%", "90%", and "80%" indicate the percentage of the reserved ground truth table cells or nodes in the table graph, and the rest of nodes are ran-

domly removed. The A_{all} results demonstrate that the proposed model can keep a relatively stable performance on incomplete table graphs and outperform ReS2TIM.

6.6. Logical Location vs Adjacency Relation

As introduced in Section 2.3, some existing methods model the structure of a table as an adjacency matrix between different cells. We argue the limitation of this representation by conducting experiments on ICDAR19-cTDaR (TrackB1). Tables in this dataset are from archival historical documents, which are more challenging than tables in other datasets due to the deformation caused by shooting angles and the overlapping between strokes and table lines. Moreover, some tables are larger than those in other datasets. The means of image heights and widths are 3,298 and 3,149 pixels, respectively. The maximum number of cells within a table is 2,267. So, it will be hard to distinguish cells when resizing a table image into a small scale before feeding it into the model. Due to these challenges, we make some different settings from previous experiments:

- Considering the limited GPU memory, TGRNet is trained separately for cell spatial and logical locations prediction. In the meantime, the input table image is resized into 800×800 pixels.

- During cell spatial location detection, TGRNet applies the morphological open operation with a 3×3 kernel on the segmentation map to separate squeezed cells.

- When initializing the table graph for cell logical location prediction, only $8 \times N$ edges with higher weights are reserved. The adjustment factor α is set to be 10.

- For ReS2TIM, the proportion of positive and negative neighbor relations is kept to $1 : 4$ by sampling cells around each target cell.

We present experimental results in Table 3. TGRNet achieves 0.828 *Hmean* for cell spatial location detection and 0.267 A_{all} for cell logical location prediction. Because ReS2TIM does not contain a cell detection module, it takes the detected cells from TGRNet as input and results in 0.138

Table 3. Comparable results for cell logical location and adjacency relation on ICDAR19-cTDaR (TrackB1).

Method	Cell Spatial Location			Cell Logical Location					WAF
	P	R	H	A_{rowSt}	A_{rowEd}	A_{colSt}	A_{colEd}	A_{all}	
ReS2TIM [30]	-	-	-	0.230	0.223	0.562	0.492	0.138	0.481
TGRNet	0.860	0.798	0.828	0.551	0.546	0.542	0.534	0.267	0.283

Table 4. Ablation studies for logical location prediction.

Exp.	GCN	Ord-Reg	Focal	A_{rowSt}	A_{rowEd}	A_{colSt}	A_{colEd}	A_{all}	H	$F_{\beta=0.5}$
1				0.884	0.884	0.779	0.775	0.697	0.899	0.850
2	✓			0.886	0.889	0.892	0.888	0.788	0.901	0.876
3	✓	✓		0.903	0.909	0.919	0.922	0.824	0.898	0.882
4	✓	✓	✓	0.917	0.916	0.919	0.923	0.832	0.906	0.890

A_{all}, which is lower than the result of TGRNet. We then compare these two methods based on the cell adjacency relation metric, *i.e.*, *WAF*. Benefiting from the carefully designed pair relation-based loss, ReS2TIM achieves a good result to predict the cell adjacency relation. However, it fails to predict the cell logical location. This reveals the limitation of the cell adjacency relation when understanding the global table structure. Specifically, it is usually required complex graph optimization algorithms to infer the cell logical location from pair-wise relations. However, the cell adjacency relation can be directly gotten from cell logical indices. Moreover, the close results of TGRNet for A_{all} and *WAF* demonstrate its reliability under different evaluation metrics for table structure recognition.

6.7. Ablation Studies

In this subsection, we isolate the contributions of the main components in TGRNet and present ablation studies on TableGraph-24K. For all experiments shown in Table 4, the backbone network and the cell spatial location branch are unchanged. The baseline of the logical location branch consists of a linear layer with the cross entropy loss, which is denoted as Exp. 1. Then, we add GCN, the ordinal regression loss, and the focal loss to the logical location branch step by step in Exp. 2-4. According to the experimental results, the *Hmean* for cell spatial location detection is around 0.900. GCN and the ordinal regression loss bring significant improvements to the performance, which lead A_{all} from 0.697 to 0.788 and 0.824, respectively. In Fig. 6, we visualize the accuracy of each logical location on a heatmap. From the four heatmaps, we can find the baseline model fits well for small logical indices and border indices. When GCN and the ordinal regression loss join in the model, the accuracy of large logical indices increases significantly. The focal loss further improves the performance.

7. Conclusion

In this paper, we formulate the problem of table structure recognition as the table graph reconstruction, which requires the model to jointly predict the cell spatial location
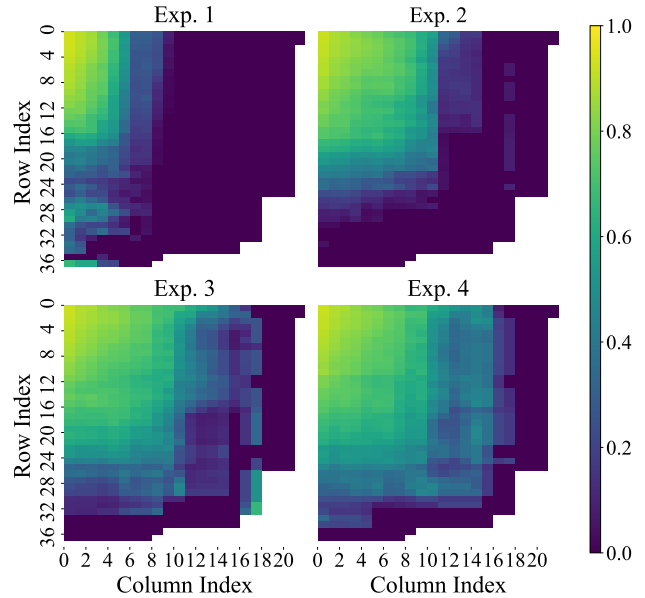

Figure 6. Visualization of the accuracy on each logical location in ablation studies. Locations masked in white mean no cell starts or ends there.

and the cell logical location. TGRNet is proposed to solve such a problem, which uses a segmentation-based module to detect the cell spatial location and solves the cell logical location prediction as an ordinal node classification problem. Experiments on four datasets demonstrate the effectiveness and robustness of TGRNet. Moreover, we contribute a new benchmark generated from TABLE2LATEX-450K dataset with 350K table graph annotations.

Acknowledgments

References

[1] Zewen Chi, Heyan Huang, Heng-Da Xu, Houjin Yu, Wanxuan Yin, and Xian-Ling Mao. Complicated table structure recognition. *arXiv preprint arXiv: 1908.04729*, 2019.

[2] Yuntian Deng, David Rosenberg, and Gideon Mann. Challenges in end-to-end neural scientific table recognition. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 894–901, 2019.

[3] Liangcai Gao, Yilun Huang, Hervé Déjean, Jean-Luc Meunier, Qinqin Yan, Yu Fang, Florian Kleber, and Eva Lang. Icdar 2019 competition on table detection and recognition (ctdar). In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1510–1515, 2019.

[4] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. Icdar 2013 table competition. In *Proceedings of 2013 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1449–1453, 2013.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4320–4333, 2020.

[8] Saqib Ali Khan, Syed Muhammad Daniyal Khalid, Muhammad Ali Shahzad, and Faisal Shafait. Table structure extraction with bi-directional gated recurrent unit networks. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1366–1371, 2019.

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[10] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. Tablebank: Table benchmark for image-based table detection and recognition. *arXiv preprint arXiv: 1903.01949*, 2019.

[11] Yiren Li, Zheng Huang, Junchi Yan, Yi Zhou, Fan Ye, and Xianhui Liu. Gfte: Graph-based financial table extraction. *arXiv preprint arXiv: 2003.07560*, 2020.

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017.

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (CVPR)*, pages 2980–2988, 2017.

[14] Xin Lin, Changxing Ding, Jinquan Zeng, and Dacheng Tao. Gps-net: Graph property sensing network for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3746–3753, 2020.

[15] Devashish Lohani, Abdel Belaïd, and Yolande Belaïd. An invoice reading system using a graph convolutional network. In *Proceedings of Asian Conference on Computer Vision (ACCV)*, pages 144–158, 2018.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[17] Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. Ordinal regression with multiple output cnn for age estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4920–4928, 2016.

[18] Ermelinda Oro and Massimo Ruffolo. Trex: An approach for recognizing and extracting tables from pdf documents. In *Proceedings of 2009 International Conference on Document Analysis and Recognition (ICDAR)*, pages 906–910, 2009.

[19] Inkit Padhi, Yair Schiff, Igor Melnyk, Mattia Rigotti, Youssef Mroueh, Pierre Dognin, Jerret Ross, Ravi Nair, and Erik Altman. Tabular transformers for modeling multivariate time series. *arXiv preprint arXiv: 2011.01843*, 2020.

[20] Shubham Singh Paliwal, D Vishwanath, Rohit Rahul, Monika Sharma, and Lovekesh Vig. Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 128–133, 2019.

[21] Devashish Prasad, Ayan Gadpal, Kshitij Kapadni, Manish Visave, and Kavita Sultanpure. 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 Workshops*, pages 572–573, 2020.

[22] Shah Rukh Qasim, Hassan Mahmood, and Faisal Shafait. Rethinking table recognition using graph neural networks. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 142–147, 2019.

[23] Roya Rastan, Hye-Young Paik, and John Shepherd. Texus: A task-based approach for table extraction and understanding. In *Proceedings of the 2015 ACM Symposium on Document Engineering*, pages 25–34, 2015.

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[25] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *Proceedings of 2017 International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1162–1167, 2017.

[26] Alexey Shigarov, Andrey Mikhailov, and Andrey Altaev. Configurable table structure recognition in untagged pdf doc-

uments. In *Proceedings of the 2016 ACM Symposium on Document Engineering*, pages 119–122, 2016.

[27] Shoaib Ahmed Siddiqui, Imran Ali Fateh, Syed Tahseen Raza Rizvi, Andreas Dengel, and Sheraz Ahmed. Deeptabstr: Deep learning based table structure recognition. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1403–1409, 2019.

[28] Shoaib Ahmed Siddiqui, Pervaiz Iqbal Khan, Andreas Dengel, and Sheraz Ahmed. Rethinking semantic segmentation for table structure recognition in documents. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1397–1402, 2019.

[29] Chris Tensmeyer, Vlad I Morariu, Brian Price, Scott Cohen, and Tony Martinez. Deep splitting and merging for table structure decomposition. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 114–121, 2019.

[30] Wenyuan Xue, Qingyong Li, and Dacheng Tao. Res2tim: Reconstruct syntactic structures from table images. In *Proceedings of 2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 749–755, 2019.

[31] Wenyuan Xue, Qingyong Li, Zhen Zhang, Yulei Zhao, and Hao Wang. Table analysis and information extraction for medical laboratory reports. In *Proceedings of the IEEE 4th International Conference on Cyber Science and Technology*, pages 193–199, 2018.

[32] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 8413–8426, 2020.

[33] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems (NeurlPS)*, 33, 2020.

[34] Jing Zhang and Dacheng Tao. Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet of Things Journal*, 8(10):7789–7817, 2020.