TOWARDS ONE-STAGE END-TO-END TABLE STRUCTURE RECOGNITION WITH PARALLEL REGRESSION FOR DIVERSE SCENARIOS

Anyi Xiao,
Nanchang Hangkong University
2304081200003@stu.nchu.edu.cn

Cihui Yang*
Nanchang Hangkong University
yangcihui@nchu.edu.cn

ABSTRACT

Table structure recognition aims to parse tables in unstructured data into machine-understandable formats. Recent methods address this problem through a two-stage process or optimized one-stage approaches. However, these methods either require multiple networks to be serially trained and perform more time-consuming sequential decoding, or rely on complex post-processing algorithms to parse the logical structure of tables. They struggle to balance cross-scenario adaptability, robustness, and computational efficiency. In this paper, we propose a one-stage end-to-end table structure parsing network called TableCenterNet. This network unifies the prediction of table spatial and logical structure into a parallel regression task for the first time, and implicitly learns the spatial-logical location mapping laws of cells through a synergistic architecture of shared feature extraction layers and task-specific decoding. Compared with two-stage methods, our method is easier to train and faster to infer. Experiments on benchmark datasets show that TableCenterNet can effectively parse table structures in diverse scenarios and achieve state-of-the-art performance on the TableGraph-24k dataset. Code is available at https://github.com/dreamy-xay/TableCenterNet.

***K*eywords** Table Structure Recognition · One-stage · End-to-End · Parallel Regression

1 Introduction

As the main carrier of structured data, tables are prevalent in diverse media such as financial statements, academic papers, and natural-scene images, carrying the organization and delivery of key information. With the deepening digital transformation, the demand for accurately parsing the machine-understandable formats (including the spatial and logical locations of cells) of tables from unstructured data is constantly growing. Although humans can easily understand tables of various styles and layouts, it remains challenging for automated systems.

Previously, most table structure recognition methods based on deep learning adopted a two-stage processing flow, such as [1, 2, 3] and other studies that first located the table region through object detection and subsequently conducted structural parsing on the cropped region, as illustrated in Figure 1a. Although these methods can handle some standard document tables, they are difficult to cope with the problem of geometric deformation in natural scenes due to shooting angle, occlusion or bending. In recent years, some works have attempted to optimize the two-stage paradigm. For instance, TGRNet[4] and LORE[5] sequentially predicted the spatial location and logical indices of cells through two networks, as shown in Figure 1b. While these methods perform well in different scenarios, their phased design requires independent training of multiple sub-modules to optimize performance, which significantly increases the training complexity. Furthermore, the serial inference process introduces additional computational overhead, making it difficult to meet the real-time requirements. Although the existing one-stage methods simplify the process, they face bottlenecks in scenario generalization. For example, TableNet[6] rely on the assumption of a clean and structured background and cannot handle blurring, uneven illumination, or geometric deformation in natural scenes. Although Cycle-CenterNet[7] and SCAN[8] support table parsing in natural scenes, it does not consider borderless tables and has insufficient accuracy

* **Corresponding author.**

in complex scenarios such as cross-row and cross-column merging. TRACE[9] is sensitive to geometric distortions such as bending and relies on complex post-processing algorithms. Overall, the existing methods have difficulty achieving an effective balance among cross-scenario adaptability, structural recognition robustness, and computational efficiency.

To address the above problems, we propose a one-stage end-to-end table structure parsing network, TableCenterNet, based on the CenterNet[10] framework. As illustrated in Figure 1c, this method for the first time unifies the spatial location detection and logical location prediction of cells into a parallel regression task, and realizes efficient joint learning through the synergistic architecture of a shared feature extraction layer and task-specific decoding. This design not only avoids the error accumulation problem of the two-stage approach, but also significantly improves the parsing accuracy of complex tables (e.g., merging cells across rows and columns) by implicitly learning the spatial-logical location mapping law. Experiments show that TableCenterNet is highly competitive and outperforms existing state-of-the-art methods for logical location prediction on the TableGraph-24k[4] dataset.

The main contributions of this paper are as follows:

- We propose an end-to-end table structure parsing network that simultaneously predicts the spatial and logical locations of cells through parallel regression.
- Experiments on three public benchmark datasets containing tabular images of diverse scenarios demonstrate the effectiveness of our proposed method.
- We provide a simple and effective one-stage TSR method that simplifies the training process and accelerates inference, and the code is available to support further research on TSR.

2 Related Work

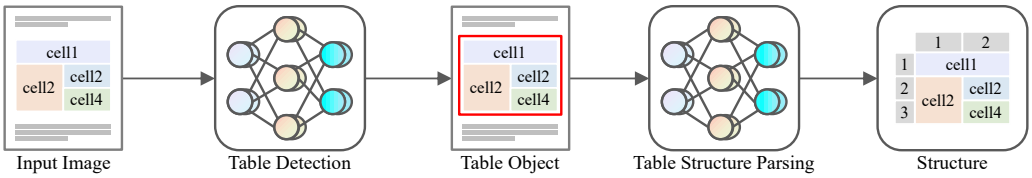
Early table structure recognition methods[11, 12, 13, 14, 15, 16] primarily rely on manually designed features and heuristic rules. For example, they parse tables using visual cues such as text arrangement, lines, and templates. Such methods are effective only for tables in specific formats (e.g., PDF) and struggle to handle complex tables. With the rise of statistical machine learning, approaches such as [17, 18, 19, 20] attempt to reduce the reliance on rules, but still require manual feature design, the stringent assumptions regarding table layouts restrict their practical applications. In recent years, deep learning-based methods have shown better performance than traditional methods[21]. Based on the descriptions of these methods, we can broadly categorize them into three groups: row/column division-based methods, markup sequence generation-based methods, and cell reconstruction-based methods.

2.1 Row/column division-based methods

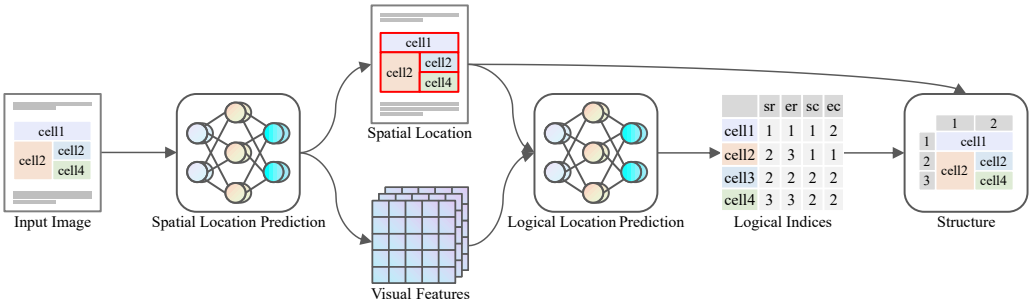
These methods mainly reconstruct the table structure by detecting or segmenting row and column regions[1, 6, 22, 23, 24, 25, 26, 3], or by directly predicting the separation lines for rows and columns[27, 28, 29, 30, 31, 32, 9]. For example, DeepTabStR[22] improves the object detection model using deformable convolutions to directly locate rows and columns; DeepdeSRT[1] and TableNet[6] utilize semantic segmentation models to predict row and column regions and generate cells through intersection operations. However, none of the above methods can handle the cases of row-spanning and column-spanning. Therefore, SPLERGE[27] further proposes a two-stage strategy of segmentation and merging, using a segmentation model to generate an initial cell grid and then processing cross-cells through a merging module. But these methods rely on the strict alignment assumption of rows and columns and have poor adaptability to curved tables or non-axis-aligned layouts. Although RobusTabNet[32] enhances robustness to distorted tables by introducing a spatial convolution module, the heuristic rules in its post-processing stage may still falter due to inadequate segmentation quality.

2.2 Markup sequence generation-based methods

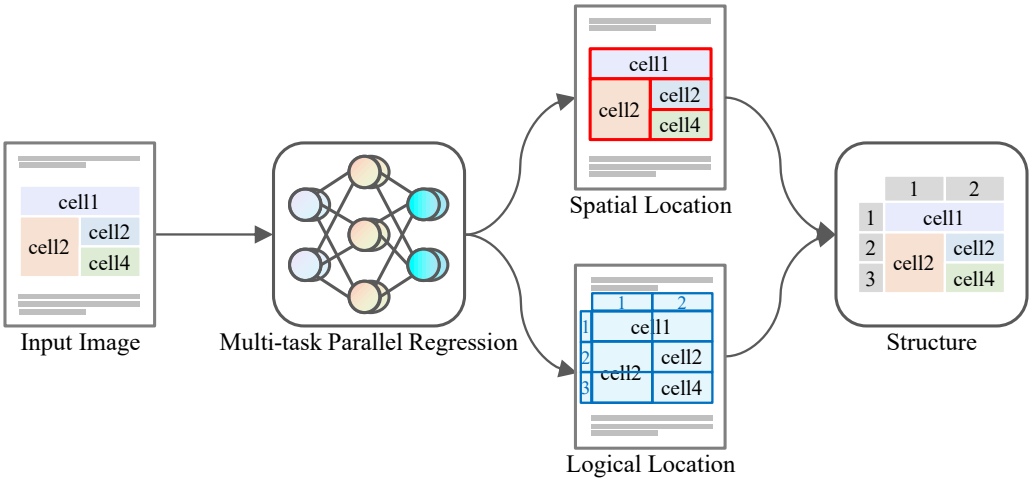
Such methods[33, 34, 35, 36, 37] treat table structure recognition as a generation task from images to tag sequences and directly output structured tags such as HTML or LaTeX. TableMaster[36] is built on the Transformer architecture, which extracts visual features through the encoder and generates HTML sequences via the decoder. TableFormer[37] not only decodes structural tags but also decodes cell coordinates. However, these methods usually rely on large-scale training data, and the length of the generated markup sequence increases sharply with greater table complexity, resulting in performance degradation. In addition, since the model uses a sequential decoding process, its inference is also more time-consuming.



(a) Previous two-stage approach based on object detection and table structure parsing



(b) Previous two-stage approach based on spatial location prediction and logical location regression



(c) Proposed one-stage approach

Figure 1: Comparison of our one-stage approach with previous two-stage approaches.

2.3 Cell reconstruction-based methods

Cell reconstruction-based methods[38, 39, 40, 4, 41, 7, 42, 43, 44, 8, 5] usually predicts table cells first and then utilize these cells as the basic units to construct the global table structure by analyzing the associative information between them. For example, GraphTSR[38] models the predicted cells as graph nodes and employs a graph attention network to predict horizontal, vertical, or irrelevant relationships. FLAG-Net[40] enhances the reasoning ability for cell associations through both dense and sparse context modeling. TGRNet[4] and LORE[5] perform logical location prediction on the detected cells using Graph Convolutional Networks and Cascade Transformers, respectively, and directly generate logical indices. Existing methods generally adopt a two-stage process of "detection-association". Our method, however, makes full use of the cell location distribution information in the feature map during the detection stage, performs parallel regression of spatial and logical location, and directly matches the physical coordinates with the logical location through interpolation maps to obtain the logical indices, thus simplifying the two-stage process into a one-stage end-to-end parsing.

3 Methodology

3.1 Overall Architecture

The overview of the proposed one-stage TableCenterNet is shown in Figure 2. It employs a CNN backbone to extract visual features of cells from the input image, and then jointly predicts the spatial locations $\{\hat{b}_1, \hat{b}_2, ..., \hat{b}_n\}$ and logical locations $\{\hat{l}_1, \hat{l}_2, ..., \hat{l}_n\}$ of cells in a multi-task manner through six parallel regression heads. n is the number of cells, and the definitions of $\hat{b}_i$ and $\hat{l}_i$ are as follows:

$$\hat{b}_i = \{(\hat{x}_{i,k}, \hat{y}_{i,k})\}_{k=1}^{4} \tag{1}$$

$$\hat{l}_i = \{\hat{r}_i^{st}, \hat{r}_i^{ed}, \hat{c}_i^{st}, \hat{c}_i^{ed}\} \tag{2}$$

Where $(\hat{x}_{i,k}, \hat{y}_{i,k})$ denotes the coordinates of the k-th corner point of the i-th cell. The four corner points are arranged clockwise starting from the upper-left corner. $\hat{r}_i^{st}$, $\hat{r}_i^{ed}$, $\hat{c}_i^{st}$, and $\hat{c}^{ed}$ correspond to the four logical indices of the i-th cell: start row, end row, start column, and end column, respectively.

Specifically, TableCenterNet comprises three main components: key point detection, spatial location regression, and logical location regression. The key point detection aims to streamline the joint prediction of spatial and logical location. Similar to CenterNet[10], we use two independent heads to regress the offsets and heatmaps of key points. The two channels of the key point heatmap correspond to the predictions of the cell center point and the corner coordinates, respectively. For the heatmap $\hat{Y} \in [0,1]^{\frac{H}{4} \times \frac{W}{4}}$ of each channel, when $\hat{Y}_{x,y} = 1$, it indicates the position of a key point, while $\hat{Y}_{x,y} = 0$ signifies the background area, where H and W denote the height and width of the input image.

To regress the spatial locations of cells, we again incorporate two regression heads: one to predict the vectors from the cell centers to the corner points, and the other to predict the vectors from the corner points back to the centers. The last two regression heads are used to predict the row and column span information of the cells and to generate the logical location interpolation map. All regression results will be parsed and processed sequentially, and finally the table structure information will be output.

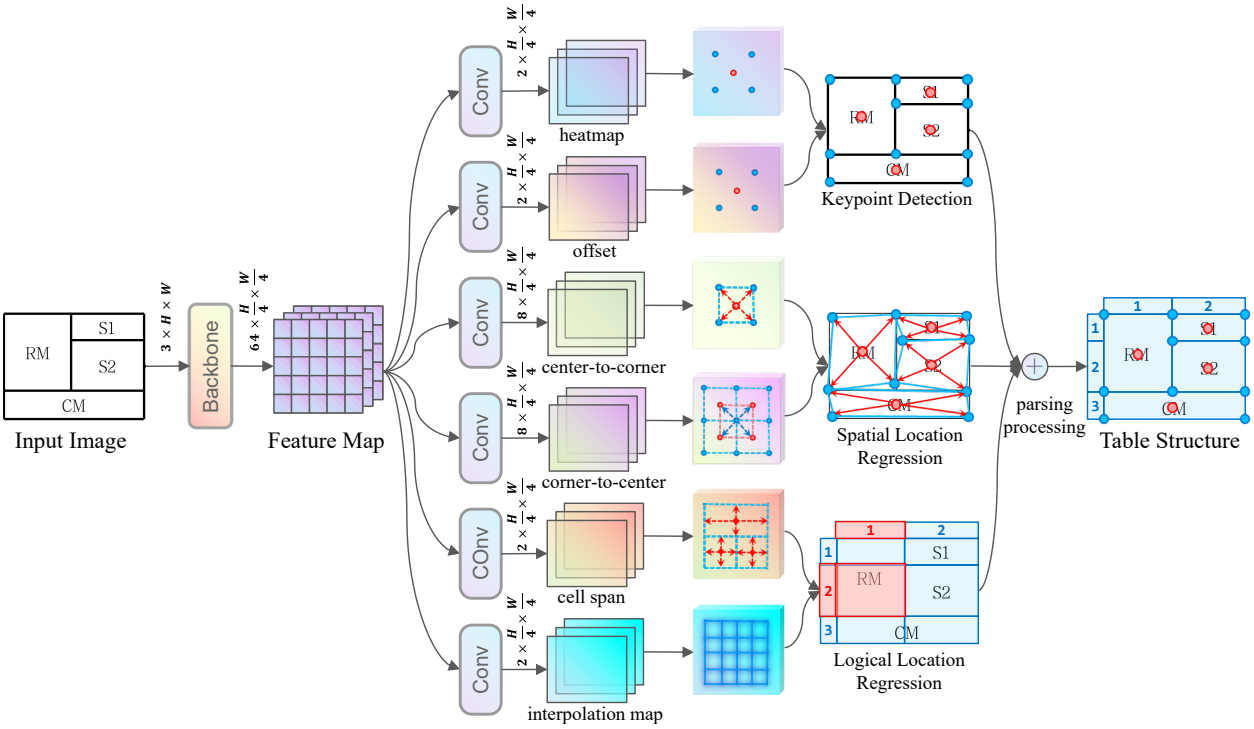


Figure 2: The architecture of our proposed method. Each "Conv" in the illustration represents a regression head, which comprises a 3×3 convolution followed by a 1×1 convolution.

3.2 Spatial Location Regression

To enhance the accuracy of spatial location prediction, we adopted the Cycle-Pairing Module[7]. By regressing the vectors $\hat{u}_i = \{(\hat{x}^u_{i,k}, \hat{y}^u_{i,k})\}^4_{k=1}$ from the center of each cell to the four corner points, as well as the vectors $\hat{v}_i = \{(\hat{x}^v_{i,k}, \hat{y}^v_{i,k})\}^4_{k=1}$ from the four corner points back to the center, and combining the corresponding cell center coordinates $\hat{t}_i = \{\hat{x}^t_i, \hat{y}^t_i\}$, we can calculate the approximate spatial locations $\{\tilde{b}_1, \tilde{b}_2, ..., \tilde{b}_n\}$ of the cells. The specific calculation formula is as follows:

$$\tilde{b}_i = \{(\tilde{x}_{i,k}, \tilde{y}_{i,k})\}^4_{k=1}, \begin{cases} \tilde{x}_{i,k} = \hat{x}^t_{i,k} + \hat{x}^u_{i,k} \\ \tilde{y}_{i,k} = \hat{y}^t_{i,k} + \hat{y}^u_{i,k} \end{cases} \tag{3}$$

In addition, we also improved the PairLoss[7] to better combine it with logical location regression and to minimize the interference caused by invalid vectors from the corners to the center. The formula for the new spatial location regression loss is as follows:

$$\mathcal{L}_{spatial} = \frac{1}{8n} \sum_{i=1}^{n} \omega(i)(\lambda_u \mathcal{L}_1(u_i, \hat{u}_i) + \lambda_v \mathcal{L}_1(v_i, \hat{v}_i)) + \lambda_e \mathcal{L}_e \tag{4}$$

Among them, u_i and v_i represent the GT corresponding to the predicted vectors $\hat{u}_i$ and $\hat{v}_i$, respectively. $\mathcal{L}_1$ denotes the MAE loss function, while $\mathcal{L}_e$ represents the loss of the vector from the invalid corner point to the center. The hyperparameters $\lambda_u = 1.0$, $\lambda_v = 0.5$, and $\lambda_e = 0.2$ are used to adjust the relative importance of these loss items. Additionally, $\omega(i)$ dynamically weights the $\mathcal{L}_1$ loss based on the quality of the regression, it is defined as follows:

$$\omega(i) = sin(\frac{\pi}{2} \cdot min(\frac{|u_i - \hat{u}_i| + |v_i - \hat{v}_i|}{|u_i|}, 1.0)) \tag{5}$$

Taking corner points as reference points, each corner point may point to the centers of at most four different cells, i.e., $4m$ corner-to-center vectors can be obtained through regression, where m is the total number of corner points. Since $4m > 4n$, in addition to the vectors $\{\hat{v}_1, \hat{v}_2, ..., \hat{v}_n\}$ matched by cells, there are also many invalid corner-to-center vectors, which are defined as $\hat{e} = \{(\hat{x}^e_j, \hat{y}^e_j)\}^{4m-4n}_{j=1}$. The loss $\mathcal{L}_e$ is formulated as follows:

$$\mathcal{L}_e = \frac{1}{8m - 8n} \sum_{j=1}^{4m-4n} (|\hat{x}^e_j| + |\hat{y}^e_j|) \tag{6}$$

3.3 Logical Location Regression

Considering that the extracted feature maps already contain the location information of cells, we capture the row-column demarcation features of the table by regressing the corresponding logical location interpolation map, obtaining the row interpolation map $\hat{I}_r \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4}}$ and the column interpolation map $\hat{I}_c \in \mathbb{R}^{\frac{H}{4} \times \frac{W}{4}}$. Meanwhile, based on the cell center, we regress the row-column span $\hat{s}_i = \{\hat{s}^r_i, \hat{s}^c_i\}$ of each cell, with the purpose of synergistically supervise the regression of the interpolation map and simplify the parsing process.

Generation of interpolation maps. Before training, we generate corresponding interpolation maps for each table image. For the GT of each image, we pair the logical indices $l_i = \{r^{st}_i, r^{ed}_i, c^{st}_i, c^{ed}_i\}$ of each cell with the physical coordinates $b_i = \{(x_{i,k}, y_{i,k})\}^4_{k=1}$ to obtain two sets of polygons $\mathcal{P}_r = \{P^r_1, P^r_2, ..., P^r_n\}$ and $\mathcal{P}_c = \{P^c_1, P^c_2, ..., P^c_n\}$ for row and column interpolation. Where P^r_i and P^c_i can be defined as follows:

$$P^r_i = \{(x_{i,k}, y_{i,k}, o^r_{i,k})\}^4_{k=1}, o^r_{i,k} = \begin{cases} r^{st}_i & \text{if } k = 1 \text{ or } k = 2 \\ r^{ed}_i + 1 & \text{otherwise} \end{cases} \tag{7}$$

$$P^c_i = \{(x_{i,k}, y_{i,k}, o^c_{i,k})\}^4_{k=1}, o^c_{i,k} = \begin{cases} c^{st}_i & \text{if } k = 1 \text{ or } k = 4 \\ c^{ed}_i + 1 & \text{otherwise} \end{cases} \tag{8}$$

Then, we input $\mathcal{P}_r$ and $\mathcal{P}_c$ into Algorithm 1 to calculate the corresponding row interpolation map $I_r \in \mathbb{R}^{H \times W}$ and column interpolation map $I_c \in \mathbb{R}^{H \times W}$. Since the physical coordinates of the input cells are the same in the two calculations, the obtained mask image $M \in \mathbb{R}^{H \times W}$ is also the same. The visualization results are shown in Figure 3. During training, we only need to reduce I_r, I_c, and M by a factor of 4 to use them as the GT for interpolation map regression.

Supervision of cell boundaries and spans. When performing logical location regression, two supervision methods are proposed to better understand the correspondence between logical and spatial locations and the constraints between

Algorithm 1 Polygons Interpolation

Input: Polygons $P_1, \ldots, P_n$ where $P_k = \{(x_i, y_i, o_i)\}$, Image dimensions (H, W)
Output: Interpolated image $I \in \mathbb{R}^{H \times W}$, Mask matrix $M \in \{0, 1\}^{H \times W}$

$I \leftarrow [\mathbf{0}]_{H \times W}$ ▷ Initialize interpolated image
$M \leftarrow [\mathbf{0}]_{H \times W}$ ▷ Initialize mask matrix
$\{A_k\} \leftarrow \left[\frac{1}{2}\left|\sum_i (x_i y_{i+1} - x_{i+1} y_i)\right|\right]_{k=1}^{n}$ ▷ Compute polygon areas
$\{P_{(k)}\} \leftarrow \text{argsort}(\{A_k\}, \text{ascending})$ ▷ Sort polygons by area
for each polygon $P_{(k)}$ **do**
    $x_{\min} \leftarrow \lfloor \min\{x_i\} \rfloor, \quad y_{\min} \leftarrow \lfloor \min\{y_i\} \rfloor$
    $x_{\max} \leftarrow \lceil \max\{x_i\} \rceil, \quad y_{\max} \leftarrow \lceil \max\{y_i\} \rceil$
    $w \leftarrow x_{\max} - x_{\min} + 1, \quad h \leftarrow y_{\max} - y_{\min} + 1$
    $\mathbf{X} \leftarrow \{(x_i - x_{\min}, y_i - y_{\min}, i)\}$ ▷ Collect and convert to local coordinates
    $\mathbf{O} \leftarrow \{o_i\}$ ▷ Collect vertex coordinate values
    $\mathbf{G} \leftarrow \{(i, j) | 0 \le i < w, 0 \le j < h\}$ ▷ Generate grid
    $\mathbf{T} \leftarrow \text{Triangulate}(\mathbf{X})$ ▷ Delaunay triangulation
    $\mathbf{Q} \leftarrow \text{LinearInterpolation}(\mathbf{T}, \mathbf{O}, \mathbf{G})$ ▷ Return -1 for extrapolation
    for each $(i, j) \in \mathbf{G}$ with $q_{i,j} \in \mathbf{Q}$ **do**
        $(i_g, j_g) \leftarrow (i + x_{\min}, j + y_{\min})$ ▷ Restore to global coordinates
        if $q_{i,j} \ge 0$ **and** $M[j_g, i_g] = 0$ **then**
            $I[j_g, i_g] \leftarrow q_{i,j}$
            $M[j_g, i_g] \leftarrow 1$
        end if
    end for
end for
return I, M

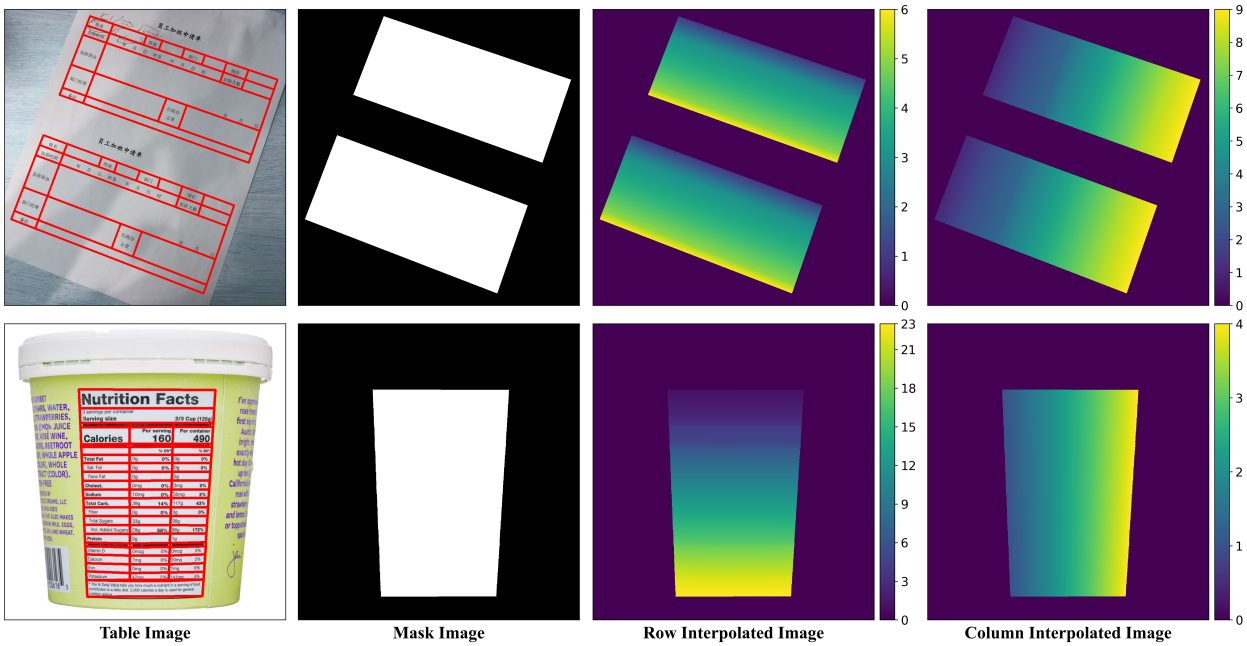


Figure 3: Visualization of row and column interpolation maps generated by the algorithm.

logical locations. Supervision of cell boundary requires that all logical boundaries of cells can effectively anchor the row-column demarcation of the table and ensure the mutual exclusivity of the logical locations of different cells. Specifically, the loss scheme for cell boundaries can be expressed as follows:

$$\mathcal{L}_{boundary} = \frac{\sum f(I_r)\mathcal{L}_1(I_r, \hat{I}_r) + \sum f(I_c)\mathcal{L}_1(I_c, \hat{I}_c)}{2\sum M} \tag{9}$$

Where $f(I_c)$ is a mapping function used to enhance the supervision of cell boundaries, which can be defined as follows:

$$f(I) = (1.0 - |I - round(I)|)^2 \tag{10}$$

Supervision of cell spans stipulates that the logical location of each cell should be consistent with its span. Formally, the loss for constraining cell spans can be defined as follows:

$$\mathcal{L}_{span} = \frac{1}{2n}\sum_{i=1}^{n}\mathcal{L}_s(i) + \frac{1}{4n}\sum_{i=1}^{n}\tilde{\mathcal{L}}_s(i) \tag{11}$$

Among them, the loss $\mathcal{L}_s(i)$ of spanning regression and the loss $\tilde{\mathcal{L}}_s(i)$ of spanning supervision for logical location of the i-th cell can be described by the following formulas:

$$\begin{cases}\mathcal{L}_s(i) = d_r(i)\mathcal{L}_1(s_i^r, \hat{s}_i^r) + d_c(i)\mathcal{L}_1(s_i^c, \hat{s}_i^c) \\ \tilde{\mathcal{L}}_s(i) = d_r(i)\mathcal{L}_1(s_i^r, \tilde{s}_i^r) + d_c(i)\mathcal{L}_1(s_i^c, \tilde{s}_i^c)\end{cases} \tag{12}$$

In formula (12), $s_i = \{s_i^r, s_i^c\}$ denotes the GT corresponding to $\hat{s}_i$, $d_r(i)$ and $d_c(i)$ are weight functions used to calculate the average distance between $\hat{s}_i^r$ and $\tilde{s}_i^r$, which helps to coordinate the regression of the interpolation map with the cell span and accelerates the convergence of the model. $d_r(i)$ and $d_c(i)$ can be formulated as follows:

$$\begin{cases}d_r(i) = sin(\frac{5\pi}{2} \cdot min(|\hat{s}_i^r - s_i^r| + |\tilde{s}_i^r - s_i^r|, 0.2)) \\ d_c(i) = sin(\frac{5\pi}{2} \cdot min(|\hat{s}_i^c - s_i^c| + |\tilde{s}_i^c - s_i^c|, 0.2))\end{cases} \tag{13}$$

Where the corresponding row span $\tilde{s}_i^r$ and column span $\tilde{s}_i^c$ of the i-th cell in the interpolation maps $\hat{I}_r$ and $\hat{I}_c$ can be calculated by the following formulas:

$$\begin{cases}\tilde{s}_i^r = (\hat{I}_r[y_{i,4}, x_{i,4}] - \hat{I}_r[y_{i,1}, x_{i,1}], \hat{I}_r[y_{i,3}, x_{i,3}] - \hat{I}_r[y_{i,2}, x_{i,2}]) \\ \tilde{s}_i^c = (\hat{I}_c[y_{i,2}, x_{i,2}] - \hat{I}_c[y_{i,1}, x_{i,1}], \hat{I}_c[y_{i,3}, x_{i,3}] - \hat{I}_c[y_{i,4}, x_{i,4}])\end{cases} \tag{14}$$

Then the regression loss of the logical location are as:

$$\mathcal{L}_{logical} = \mathcal{L}_{boundary} + \mathcal{L}_{span} \tag{15}$$

3.4 Parsing Processing

In Figure 4a, $\{\tilde{b}_1, \tilde{b}_2, ..., \tilde{b}_n\}$ obtained through regression are only approximations. The detected cell corner points (e.g., Figure 4b) need to be used for alignment processing[7] to obtain more accurate spatial locations $\{\check{b}_1, \check{b}_2, ..., \check{b}_n\}$, as shown in Figure 4c. Then, calculate the physical coordinates $\{\hat{b}_1, \hat{b}_2, ..., \hat{b}_n\}$ of all cells through 4x magnification.

(a) (b) (c)

Figure 4: Visualization of cell spatial location alignment. (a) is the result of cell spatial location regression, (b) is the result of cell corner point detection, and (c) is the result after cell spatial location alignment.

Subsequently, by directly matching through physical coordinates and the interpolation map of logical location, the logical indices $\{\hat{l}_1, \hat{l}_2, ..., \hat{l}_n\}$ of all cells can be obtained. The specific matching process for each cell is shown in the following formula:

$$\hat{l}_i = \{\hat{r}_i^{st}, \hat{r}_i^{ed}, \hat{c}_i^{st}, \hat{c}_i^{ed}\}, \begin{cases} \hat{r}_i^{st} = round(\hat{I}_r[\check{y}_{i,1}, \check{x}_{i,1}]) \\ \hat{r}_i^{ed} = round(\hat{I}_r[\check{y}_{i,1}, \check{x}_{i,1}]) + \lfloor \hat{s}_i^r \rfloor - 1 \\ \hat{c}_i^{st} = round(\hat{I}_c[\check{y}_{i,1}, \check{x}_{i,1}]) \\ \hat{c}_i^{ed} = round(\hat{I}_c[\check{y}_{i,1}, \check{x}_{i,1}]) + \lfloor \hat{s}_i^c \rfloor - 1 \end{cases} \tag{16}$$

Where $(\check{y}_{i,1}, \check{x}_{i,1})$ denotes the coordinates of the upper-left corner of the physical coordinates $\check{b}_i = (\check{y}_{i,k}, \check{x}_{i,k})_{k=1}^4$ of the i-th cell.

3.5 Training Strategy

Design of overall loss function. The proposed TableCenterNet is trained in an end-to-end manner for three optimization tasks: keypoint detection, spatial location regression and logical location regression. The global optimization can be defined as:

$$\mathcal{L}_{overall} = \mathcal{L}_{keypoint} + \mathcal{L}_{spatical} + \mathcal{L}_{logical} \tag{17}$$

Where $\mathcal{L}_{keypoint}$ is the keypoint detection loss, which mainly consists of the heat map loss $\mathcal{L}_k$ and offset loss $\mathcal{L}_{off}$ of keypoints, consistent with CenterNet[10].

Selective gridding interpolation. For datasets with unbiased prior knowledge of physical and logical structures and no significant deformation, there is no need to generate interpolation maps based on cells before training. As shown in Figure 5, the cells can be converted into a logical grid structure, i.e., decomposing the cross-row and cross-column cells into multiple logical grid cells. This effectively unifies the interpolation style of each cell and improves the regression accuracy of the interpolation map.

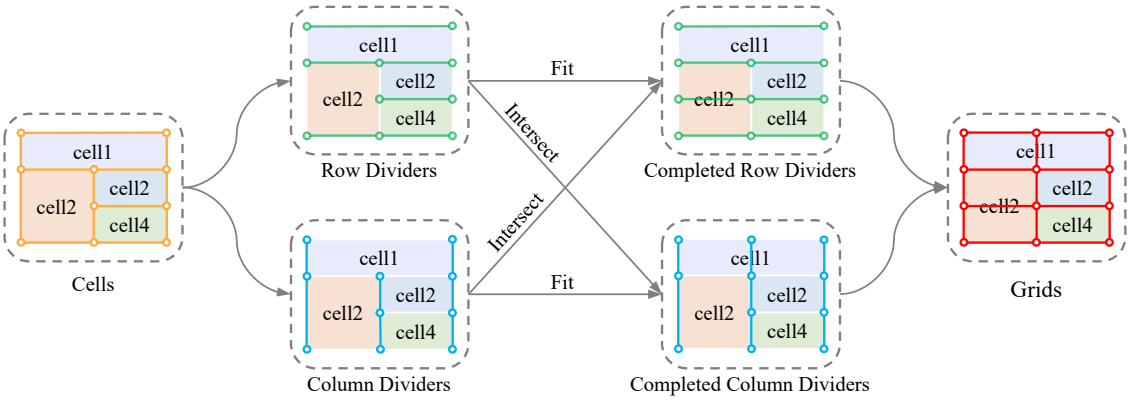


Figure 5: Flow of converting cells into grids. First, based on the physical coordinates and logical indexes of cell corners, row and column dividers are grouped. Then, the two types of dividers are completed by fitting and intersecting with each other. Finally, the completed row and column dividers are fused by corners to generate logical grids.

4 Experiments

4.1 Benchmark Datasets

We conducted experiments on three popular public benchmarks, including the ICDAR2013 Table Competition dataset, the Wired Table in the Wild (WTW) dataset, and the TableGraph-24k dataset to validate the effectiveness of TableCenterNet.

ICDAR-2013[45] consists of 156 tables in PDF format from EU/US government websites, which contain cross-cells and various other styles. Because this paper focuses on reconstructing table from cells, an modified version of ICDAR-2013, the ICDAR-2013.c[46], is used, which provides detailed cell bounding boxes rather than word region boxes. It should be noted that the original dataset does not specify the training and test sets, we use 80% of the table images for training and others for test following the setting in [39, 42, 40, 5, 47].

WTW[7] consists of digital document images, scanned images, and images taken in the field, and contains a total of 10,970 training images and 3,611 test images collected from wild complex scenes. This dataset focuses on bordered tabular objects only and covers a wide range of challenging situations such as inclined tables, curved tables, occluded and blurred tables, extreme aspect ratio tables, overlaid tables, irregular tables, and multicolored tables. The labeling of the dataset includes both the physical and logical coordinates of the cells, with the physical coordinates represented by quadrilaterals.

TableGraph-24k[4] is a subset of TableGraph-350K, containing 20,000 training images, 2,000 validation images, and 2,000 test images. All images in this dataset are sourced from scholarly articles on the arXiv platform and include wired tables, wireless tables, and partial border tables.

4.2 Evaluation Metrics

Similar to previous studies[39, 4], we first evaluate the performance of our model in predicting the spatial locations of cells (physical coordinates) using precision, recall, and F1 scores under the IoU threshold of 0.5. For the logical locations of cells (row/column information), we employed the accuracy of logical location[48], the F1 score of adjacency relationships between cells[49, 45], and Tree-Edit-Distance-based Similarity (TEDS)[35] as evaluation metrics. The accuracy of logical location and TEDS directly reflect the correctness of the predicted structure, while the adjacency relationships only measure the quality of the intermediate results of the structure[5].

4.3 Implementation Details

All experiments were conducted in a PyTorch environment and executed on a workstation equipped with two GeForce RTX 3090 GPUs. In these experiments, we respectively adopted DLA-34[50] and StartNet-s3[51] as the backbone networks, and their weights were all initialized based on pre-trained models for the ImageNet[52] classification task. Within the CenterNet[10] framework's regression network, we configured the number of hidden channels for each convolutional head to 256. Model optimization was performed using the Adam algorithm with a batch size of 22. For both backbone networks, the models were trained for 200 epochs, with the initial learning rate set to 1.25×10^{-4}. The learning rate was decayed to 1.25×10^{-5} and 1.25×10^{-6} in the 140th and 180th epochs, respectively. To ensure scale invariance, we uniformly resized the input table images to a fixed size of 1024×1024 pixels, and 768×768 pixels for the TableGraph-24k dataset, while maintaining the aspect ratio of the images. After resizing, we also applied data augmentation and normalization to the input images to accelerate the convergence of the network.

Due to the limited size of the ICDAR-2013 dataset, which contains only 258 table images after preprocessing, we opted to perform 100 rounds of fine-tuning on the model trained on WTW using the segmented ICDAR-2013 training set. During the fine-tuning phase, we expanded the original image by adding 100 black pixels to each of the four directions: top, bottom, left, and right, before scaling the image. The initial learning rate decays to 1.25×10^{-5} and 1.25×10^{-6} in the 70th and 90th epochs, respectively, while ensuring that the other training configurations remain unchanged. In the testing phase, we similarly expand the perimeter of the test image by 100 black pixels before performing inference.

4.4 Results

To evaluate TableCenterNet's ability to recognize table structures in different scenarios, we tested it on three benchmark datasets with different scenarios. The qualitative results in Figure 6 show that our method is robust to complex structures, diverse colors, geometric transformations, and distorted or deformed tables in both photographic and scanning scenes. Furthermore, our method also demonstrates generalization to digital native scenarios, effectively handling tables across cells, with bounded cells, and without bounded cells.

Results on ICDAR-2013. As shown in Table 1, our proposed TableCenterNet method outperforms most previous methods in recognizing both the physical and logical structures of tables. Compared to the robust baseline model LORE[5], TableCenterNet-D enhances the physical coordinate F1 scores and the accuracy of the logical location by 0.6% and 4.3%, respectively. Similarly, TableCenterNet-S improves these two metrics by 0.3% and 1.3%, respectively. For comparison with Cycle-CenterNet[7], we used a model trained on the WTW dataset to test the wired tables in ICDAR-2013[46]. The results indicate that our method exceeds Cycle-CenterNet[7] and SCAN[8] in terms of precision, recall, and F1 score metrics for adjacency. This shows that our method achieves competitive results in both wired and wireless scenarios.

Results on WTW. Considering that the previous methods used mainly two IoU thresholds, 0.5 and 0.9, to evaluate the prediction performance of physical coordinates, we conducted independent evaluations under these two distinct thresholds. The experimental results are presented in Table 2. The physical coordinate F1 scores of TableCenterNet-D and TableCenterNet-S are both improved by 0.9% compared to the optimal LORE[5] at an IoU threshold of 0.5,

and the physical coordinate F1 scores at an IoU threshold of 0.9 demonstrated improvements of 13% and 12.6% over Cycle-CenterNet[7], respectively. For the evaluation of cell logical coordinates, TableCenterNet-D outperforms LORE[5] in both adjacency and the accuracy of logical location. In addition, TableCenterNet-D and TableCenterNet-S improve by 1% and 0.4% in TEDS compared to the optimal method [8], indicating that our approach also offers parsing advantages for complex tables in the wild.

Results on TableGraph-24k. To our knowledge, only two methods have been trained and tested on this data set for the precision of cell physical and logical coordinates: TGRNet[4] and LORE[5], respectively. In Table 3, we observe that TableCenterNet significantly outperforms previous methods. When the F1 scores for the physical coordinates of the cells are quite similar compared to the optimal method [5], TableCenterNet-D and TableCenterNet-S have improved the accuracy of logical location by 7.2% and 6.4%, respectively, achieving state-of-the-art performance. This demonstrates the substantial advantage of our methods in the digital native table image recognition scenario for scientific articles.

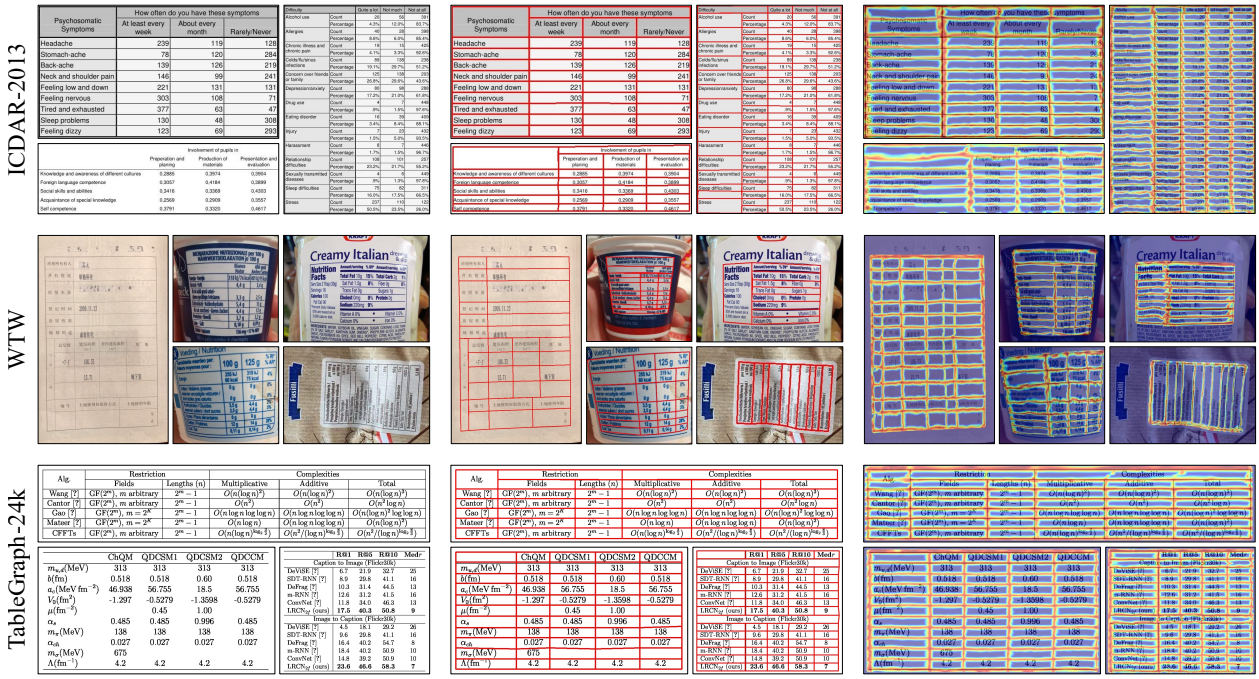


Figure 6: Qualitative results of our approach. The second column displays the spatial location of each cell. The third column visualizes the logical separation of rows and columns in the table using a heat map, which intensifies to red as the predicted interpolated map values approach positive integers.

4.5 Ablation Study

In this subsection, we isolate the contributions of the key components in TableCenterNet and conduct ablation experiments on the WTW dataset. In all the experiments presented in Table 4, the differences in spatial location accuracy are minimal, so we focus on the effect of key components on logical location accuracy.

Effectiveness of cell boundaries and spans supervisions. Based on the results of Exp.1a and Exp.1b, we observe that the cell-span regression yields a performance improvement of +4.4% Acc. Comparing Exp.1a and Exp.1c, the cell span supervised loss L_{span} leads to an even more substantial enhancement, with a performance improvement of +6.5% Acc. This is due to the fact that L_{span} provides anchors for interpolation plot regression and strengthens the constraints on the range of interpolation between cells. From Exp.1d and Exp.1f, the cell boundary supervised loss $L_{boundary}$ also improves by +0.6%Acc due to the fact that it reduces the interpolation interference. According to the properties of interpolation map generation, the logical indices of cell boundaries are closer to the true values, while the logical indices of in-cell interpolation are approximate and less reliable. In Figure 7, we visualize the accuracy of each logical location by heat map. From the three heat maps, we can find that cell spanning regression improves the accuracy of small logical indices. When the two supervision mechanisms are combined, the accuracy of large logical indices is also significantly improved.

Table 1: Comparison of the accuracy of logical location on ICDAR-2013. The precision, recall and F1 score are evaluated on physical coordinates-based and adjacency relation-based metrics. Here, * denotes for the result on wired tables only, whereas † means that pre-trained data are used.

Model	Backbone	Training Datasets	Physical Coordinates			Adjacency Relation			Acc
			P	R	F1	P	R	F1	
ReS2TIM[48]	-	ICDAR2013†	-	-	-	73.4	74.7	74.0	17.4
TGRNet[4]	ResNet-50	ICDAR2013†	68.2	65.2	66.7	-	-	-	27.5
Cycle-CenterNet[7]	DLA-34	WTW+ICDAR2019	-	-	-	95.5	88.3	91.7	-
TabStrNet[39]	ResNet-101	SciTSR	-	-	-	93.0	90.8	91.9	-
LGPMA[25]	ResNet-50	ICDAR2013†	-	-	-	96.7	99.1	97.9	-
Cycle-CenterNet*[7]	DLA-34	WTW	-	-	-	97.5	98.4	98.0	-
TOD[44]	-	FinTabNet	-	-	-	98.0	97.0	98.0	-
SCAN*[8]	ResNet-50	WTW	-	-	-	98.2	98.1	98.2	-
FLAGNet[40]	ResNet-50	ICDAR2013†	-	-	-	97.9	99.3	98.6	-
NCGM[42]	ResNet-18	ICDAR2013†	-	-	-	98.4	99.3	98.8	-
LORE[5]	DLA-34	ICDAR2013†	-	-	97.2	99.2	98.6	98.9	86.8
TableCenterNet-D	DLA-34	ICDAR2013†	98.7	96.9	97.8	96.1	99.1	97.6	91.1
TableCenterNet-D*	DLA-34	WTW	97.1	97.9	97.5	98.4	98.6	98.5	94.6
TableCenterNet-S	StartNet-s3	ICDAR2013†	98.2	96.8	97.5	93.7	97.5	95.6	88.1
TableCenterNet-S*	StartNet-s3	WTW	97.1	98.2	97.6	98.4	98.6	98.5	94.1

Table 2: Comparison of the accuracy of logical location and TEDS on WTW. The precision, recall and F1 score are evaluated on physical coordinates-based and adjacency relation-based metrics. Here * means using an IoU threshold of 0.9 to evaluate the performance of physical coordinates. Underlines denote the best.

Model	Backbone	Physical Coordinates			Adjacency Relation			Acc	TEDS
		P	R	F1	P	R	F1		
CascadeTabNet[53]	CascadeNet	-	-	-	16.4	3.6	5.9	-	11.4
Split+Heuristic*[27]	-	3.2	3.6	3.4	25.7	29.9	27.6	-	26.0
TGRNet[4]	ResNet-50	-	-	64.7	-	-	-	24.3	-
FLAGNet[40]	-	-	-	-	91.6	89.5	90.5	-	-
Cycle-CenterNet*[7]	DLA-34	78.0	78.5	78.3	93.3	91.5	92.4	-	83.3
TSRFormer[31]	ResNet-18	-	-	-	93.7	93.2	93.4	-	-
NCGM[42]	ResNet-18	-	-	-	93.7	94.6	94.1	-	-
SCAN[8]	ResNet-50	-	-	-	-	-	-	-	90.7
TRACE*[9]	ResNet-50	63.8	65.7	64.8	93.5	95.5	94.5	-	-
LORE[5]	DLA-34	-	-	96.4	94.5	95.9	95.1	82.9	-
TableCenterNet-D	DLA-34	**98.1**	**96.5**	**97.3**	**94.8**	**97.2**	**96.0**	**83.0**	**91.7**
TableCenterNet-D*	DLA-34	91.9	90.8	91.3	-	-	-	-	-
TableCenterNet-S	StartNet-s3	**98.1**	**96.5**	**97.3**	94.3	96.9	95.6	81.5	91.1
TableCenterNet-S*	StartNet-s3	91.4	90.4	90.9	-	-	-	-	-

Effect of different logical location regression methods. In Exp.2a and Exp.2b, we replace interpolation map regression with upper-left corner point regression and four corner points regression, respectively, and both of them use the optimal loss computation scheme, and the network structure remains consistent with TableCenterNet. Compared with Exp.2f, the performance of these two logical location regression methods decreases by 3.2%Acc and 1.2%Acc, respectively, indicating that the interpolation map regression can better model the dependencies and constraints between logical locations of different cells. When there is an offset in the spatial locations regressed by the methods in Exp. 2a and Exp. 2b, the logical locations are prone to errors, whereas interpolation maps can ensure the accuracy of logical indices as much as possible by rounding the interpolation results.

Table 3: Comparison of the accuracy of logical location on TableGraph-24k. Acc_{rowSt}, Acc_{rowEd}, Acc_{colSt}, and Acc_{colEd} refer to the accuracy of the four logical indices of the start row, start column, end row, and end column, respectively. The precision, recall and F1 score are evaluated on physical coordinates-based metrics. Underlines denote the best.

Model	Backbone	Physical Coordinates			Logical Location				
		P	R	F1	Acc_{rowSt}	Acc_{rowEd}	Acc_{colSt}	Acc_{colEd}	Acc
TGRNet[4]	ResNet-50	91.6	89.5	90.6	91.7	91.6	91.9	92.3	83.2
LORE[5]	DLA-34	-	-	96.1	-	-	-	-	87.9
TableCenterNet-D	DLA-34	95.4	**96.6**	96.0	**97.9**	**97.9**	**97.3**	**97.1**	**95.1**
TableCenterNet-S	StarNet-s3	**95.7**	**96.6**	**96.2**	97.8	97.7	96.7	96.4	94.3

Table 4: Ablation study of TableCenterNet. The precision, recall and F1 score are evaluated on physical coordinates-based metrics. A-c, A-r and Acc refer to the accuracy of column indices, row indices and all logical indices. All these models are trained from scratch according to the 'Implementation Details' section.

Exp.	Regression		Objectives		Spatial Location			Logical Location		
	Logical Location	Cell Span	L_{span}	$L_{boundary}$	P	R	F1	A-c	A-r	Acc
1a	Interpolation map	✗	✗	✗	98.1	96.4	97.3	86.4	84.8	75.1
1b	Interpolation map	✓	✗	✗	98.1	96.4	97.2	89.0	87.4	79.5
1c	Interpolation map	✗	✓	✗	98.2	96.5	97.4	90.7	88.3	81.6
1d	Interpolation map	✓	✓	✗	98.1	96.5	97.3	91.0	88.8	82.4
1f	Interpolation map	✓	✓	✓	98.1	96.5	97.3	91.4	89.1	83.0
2a	Upper-left corner point	✓	-	-	97.8	96.4	97.1	89.6	87.0	79.8
2b	Four corner points	✓	✓	-	97.8	96.5	97.1	90.9	88.2	81.8

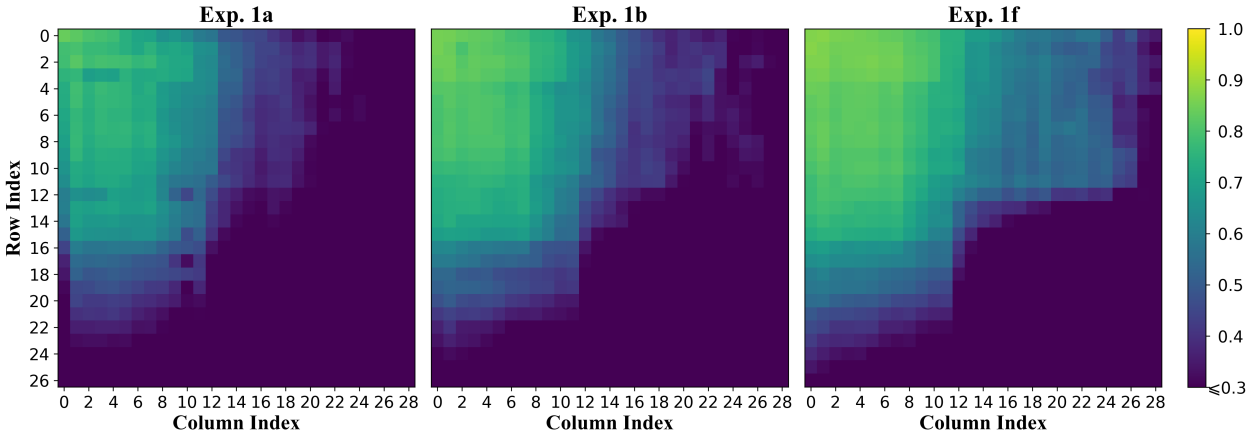


Figure 7: Visualization of the accuracy on each logical location in ablation study. Only the parts with a relatively large proportion in the logical indices are displayed.

4.6 Computational Analysis

We compare three measures of the computational efficiency of the models in Table 5, provided that the comprehensive performance is comparable. The input image size is 1024×1024, and both the comprehensive performance and the average inference time are computed based on the WTW[7] test set, where the comprehensive performance is referenced from [4], based on a combination of the F1 of the physical coordinates and the accuracy of logical locations. With the same backbone, TableCenterNet-D reduces the number of parameters by 27.3% compared to LORE, but increases the inference speed by 15.7 times. TableCenterNet-S further improves the computational efficiency by using the StarNet-s3[51] backbone. Compared with TableCenterNet-D, the number of model parameters and FLOPs of TableCenterNet-S are reduced by 62.7% and 62.2%, respectively, which makes it possible to deploy on edge devices.

Table 5: Comparison between TableCenterNet and LORE in terms of the number of parameters, FLOPs, and average inference time when the comprehensive performance $F_{\beta=0.5}$ is comparable. The units are million for the number of parameters, giga for the FLOPs, and milliseconds for the average inference time.

Model	Backbone	$F_{\beta=0.5}\uparrow$	Parameters↓	FLOPs↓	Inference Time↓
LORE	DLA-34	0.934	27.86	142.3	3788.4
TableCenterNet-D	DLA-34	0.941	16.82	133.1	227.4
TableCenterNet-S	StarNet-s3	0.937	6.27	50.3	214.8

5 Conclusion

In this paper, we present a one-stage end-to-end table structure parsing method, TableCenterNet, which uses a single model to simultaneously regress both the spatial and logical structures of tables. Unlike previous methods[4, 5] that focus on logical location regression, TableCenterNet directly regresses the distribution of the logical structure of each table at various locations within the input image only through convolution along with supervision of cell boundaries and spans, simplifying the model training process and accelerating model inference. Experimental results demonstrate that our approach achieves competitive outcomes across diverse scenarios, including the challenging table scenario, and achieves state-of-the-art performance on the TableGraph-24k dataset. In future work, we plan to collect challenging wireless tabular data from diverse scenarios to further train our models and provide a reference for improvement.

References

[1] 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 *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 1162–1167, 2017.

[2] Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 697–706, 2021.

[3] Johan Fernandes, Bin Xiao, Murat Simsek, Burak Kantarci, Shahzad Khan, and Ala Abu Alkheir. Tablestrrec: framework for table structure recognition in data sheet images. *International Journal on Document Analysis and Recognition (IJDAR)*, 27(2):127–145, Jun 2024.

[4] Wenyuan Xue, Baosheng Yu, Wen Wang, Dacheng Tao, and Qingyong Li. Tgrnet: A table graph reconstruction network for table structure recognition. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1275–1284, 2021.

[5] Hangdi Xing, Feiyu Gao, Rujiao Long, Jiajun Bu, Qi Zheng, Liangcheng Li, Cong Yao, and Zhi Yu. Lore: logical location regression network for table structure recognition. AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

[6] Shubham Singh Paliwal, Vishwanath D, 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 *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 128–133, 2019.

[7] Long Rujiao, Wang Wen, Xue Nan, Gao Feiyu, Yang Zhibo, Wang Yongpan, and Xia Gui-Song. Parsing table structures in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021.

[8] Hongyi Wang, Yang Xue, Jiaxin Zhang, and Lianwen Jin. Scene table structure recognition with segmentation collaboration and alignment. *Pattern Recognition Letters*, 165:146–153, 2023.

[9] Youngmin Baek, Daehyun Nam, Jaeheung Surh, Seung Shin, and Seonghyeon Kim. Trace: Table reconstruction aligned to corner and edges. In *Document Analysis and Recognition - ICDAR 2023*, pages 472–489. Springer Nature Switzerland, 2023.

[10] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[11] A. Laurentini and P. Viada. Identifying and understanding tabular material in compound documents. In *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, pages 405–409, 1992.

[12] K. Itonori. Table structure recognition based on textblock arrangement and ruled line position. In *Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR '93)*, pages 765–768, 1993.

[13] Thomas G. Kieninger. Table structure recognition based on robust block segmentation. In *Document Recognition V*, volume 3305, pages 22 – 32. International Society for Optics and Photonics, SPIE, 1998.

[14] Thomas Kieninger and Andreas Dengel. The t-recs table recognition and analysis system. In *Document Analysis Systems: Theory and Practice*, pages 255–270. Springer Berlin Heidelberg, 1999.

[15] Alexey Shigarov, Andrey Mikhailov, and Andrey Altaev. Configurable table structure recognition in untagged pdf documents. In *Proceedings of the 2016 ACM Symposium on Document Engineering*, DocEng '16, page 119–122. Association for Computing Machinery, 2016.

[16] Roya Rastan, Hye-Young Paik, and John Shepherd. Texus: A unified framework for extracting and understanding tables in pdf documents. *Information Processing & Management*, 56(3):895–918, 2019.

[17] Hwee Tou Ng, Chung Yong Lim, and Jessica Li Teng Koo. Learning to recognize tables in free text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, page 443–450. Association for Computational Linguistics, 1999.

[18] Yalin Wang, Ihsin T. Phillips, and Robert M. Haralick. Table structure understanding and its performance evaluation. *Pattern Recognition*, 37(7):1479–1497, 2004.

[19] Ying Liu, Prasenjit Mitra, and C. Lee Giles. Identifying table boundaries in digital documents via sparse line detection. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 1311–1320. Association for Computing Machinery, 2008.

[20] Ying Liu, Kun Bai, Prasenjit Mitra, and C. Lee Giles. Improving the table boundary detection in pdfs by fixing the sequence error of the sparse lines. In *2009 10th International Conference on Document Analysis and Recognition*, pages 1006–1010, 2009.

[21] Khurram Azeem Hashmi, Marcus Liwicki, Didier Stricker, Muhammad Adnan Afzal, Muhammad Ahtsham Afzal, and Muhammad Zeshan Afzal. Current status and performance analysis of table recognition in document images with deep neural networks. *IEEE Access*, 9:87663–87685, 2021.

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

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

[24] Khurram Azeem Hashmi, Didier Stricker, Marcus Liwicki, Muhammad Noman Afzal, and Muhammad Zeshan Afzal. Guided table structure recognition through anchor optimization. *IEEE Access*, 9:113521–113534, 2021.

[25] Liang Qiao, Zaisheng Li, Zhanzhan Cheng, Peng Zhang, Shiliang Pu, Yi Niu, Wenqi Ren, Wenming Tan, and Fei Wu. Lgpma: Complicated table structure recognition with local and global pyramid mask alignment. In *Document Analysis and Recognition – ICDAR 2021*, pages 99–114. Springer International Publishing, 2021.

[26] Brandon Smock, Rohith Pesala, and Robin Abraham. Pubtables-1m: Towards comprehensive table extraction from unstructured documents. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4624–4632, 2022.

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

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

[29] Yajun Zou and Jinwen Ma. A deep semantic segmentation model for image-based table structure recognition. In *2020 15th IEEE International Conference on Signal Processing (ICSP)*, volume 1, pages 274–280, 2020.

[30] Zhenrong Zhang, Jianshu Zhang, Jun Du, and Fengren Wang. Split, embed and merge: An accurate table structure recognizer. *Pattern Recognition*, 126:108565, 2022.

[31] Weihong Lin, Zheng Sun, Chixiang Ma, Mingze Li, Jiawei Wang, Lei Sun, and Qiang Huo. Tsrformer: Table structure recognition with transformers. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 6473–6482. Association for Computing Machinery, 2022.

[32] Chixiang Ma, Weihong Lin, Lei Sun, and Qiang Huo. Robust table detection and structure recognition from heterogeneous document images. *Pattern Recogn.*, 133(C), January 2023.

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

[34] Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. TableBank: Table benchmark for image-based table detection and recognition. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1918–1925. European Language Resources Association, May 2020.

[35] Xu Zhong, Elaheh ShafieiBavani, and Antonio Jimeno Yepes. Image-based table recognition: data, model, and evaluation. In *European Conference on Computer Vision*, pages 564–580. Springer, 2020.

[36] Jiaquan Ye, Xianbiao Qi, Yelin He, Yihao Chen, Dengyi Gu, Peng Gao, and Rong Xiao. Pingan-vcgroup's solution for icdar 2021 competition on scientific literature parsing task b: Table recognition to html. *arXiv preprint arXiv:2105.01848*, 2021.

[37] Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. Tableformer: Table structure understanding with transformers. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4604–4613, 2022.

[38] 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.

[39] Sachin Raja, Ajoy Mondal, and C. V. Jawahar. Table structure recognition using top-down and bottom-up cues. In *In European Conference on Computer Vision*, pages 70–86. Springer, 2020.

[40] Hao Liu, Xin Li, Bing Liu, Deqiang Jiang, Yinsong Liu, Bo Ren, and Rongrong Ji. Show, read and reason: Table structure recognition with flexible context aggregator. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, page 1084–1092. Association for Computing Machinery, 2021.

[41] Koji Ichikawa. Image-based relation classification approach for table structure recognition. In *Document Analysis and Recognition – ICDAR 2021*, pages 632–647. Springer International Publishing, 2021.

[42] Hao Liu, Xin Li, Bing Liu, Deqiang Jiang, Yinsong Liu, and Bo Ren. Neural collaborative graph machines for table structure recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4523–4532, 2022.

[43] Bin Xiao, Murat Simsek, Burak Kantarci, and Ala Abu Alkheir. Table structure recognition with conditional attention. *arXiv preprint arXiv:2203.03819*, 2022.

[44] Sachin Raja, Ajoy Mondal, and Jawahar C V. Visual understanding of complex table structures from document images. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2543–2552, 2022.

[45] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. Icdar 2013 table competition. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1449–1453, 2013.

[46] Brandon Smock, Rohith Pesala, and Robin Abraham. Aligning benchmark datasets for table structure recognition. pages 371–386, 2023.

[47] Rujiao Long, Hangdi Xing, Zhibo Yang, Qi Zheng, Zhi Yu, Fei Huang, and Cong Yao. Lore++: Logical location regression network for table structure recognition with pre-training. *Pattern Recognition*, 157:110816, 2025.

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

[49] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. A methodology for evaluating algorithms for table understanding in pdf documents. In *Proceedings of the 2012 ACM Symposium on Document Engineering*, DocEng '12, page 45–48. Association for Computing Machinery, 2012.

[50] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, 2018.

[51] Xu Ma, Xiyang Dai, Yue Bai, Yizhou Wang, and Yun Fu. Rewrite the stars. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5694–5703, 2024.

[52] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[53] 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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2439–2447, 2020.