Rethinking Table Recognition using Graph Neural Networks

Shah Rukh Qasim*, Hassan Mahmood*, and Faisal Shafait*,†

*School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST), Islamabad, Pakistan
†Deep Learning Laboratory, National Center of Artificial Intelligence (NCAI), Islamabad, Pakistan
Email: faisal.shafait@seecs.edu.pk

Abstract—Document structure analysis, such as zone segmentation and table recognition, is a complex problem in document processing and is an active area of research. The recent success of deep learning in solving various computer vision and machine learning problems has not been reflected in document structure analysis since conventional neural networks are not well suited to the input structure of the problem. In this paper, we propose an architecture based on graph networks as a better alternative to standard neural networks for table recognition. We argue that graph networks are a more natural choice for these problems, and explore two gradient-based graph neural networks. Our proposed architecture combines the benefits of convolutional neural networks for visual feature extraction and graph networks for dealing with the problem structure. We empirically demonstrate that our method outperforms the baseline by a significant margin. In addition, we identify the lack of large scale datasets as a major hindrance for deep learning research for structure analysis and present a new large scale synthetic dataset for the problem of table recognition. Finally, we open-source our implementation of dataset generation and the training framework of our graph networks to promote reproducible research in this direction[1].

Keywords-Table Recognition; Structure Analysis; Graph Neural Networks; Document Model; Graph Model; Dataset

I. INTRODUCTION

Structural analysis is one of the most important aspects of document processing. It incorporates both physical and logical layout analysis and also includes parsing or recognition of complex structured layouts including tables, receipts, and forms. While there has been a lot of research done in physical and logical layout analysis of documents, there is still ample room for contribution towards parsing of structured layouts, such as tables, within them. Tables provide an intuitive and natural way to present data in a format which could be readily interpreted by humans. Based on its significance and difficulty level, table structure analysis has attracted a large number of researchers to make contributions in this domain.

Table detection and recognition is an old problem with research starting from the late nineties. One of the initial work is by Kieninger et al. [1]. They used a bottom-up approach on words bounding boxes using a heuristics-based algorithm. Later on, many different hand-crafted features based methods were introduced including [2], [3], [4] and [5] which relied on custom-designed algorithms. Zanibbi et al. [6] present a

comprehensive survey of table detection and structure recognition algorithms at that time. An approach to recognize tables in spreadsheets was presented by [7] which classified every cell into either a header, a title, or a data cell. Significant work was done by Shafait et al. [8] where they introduced different performance metrics for the table detection problem. These approaches are not data driven and they make strong assumptions about tabular structures.

Chen et al. [9] used support vector machines and dynamic programming for table detection in handwritten documents. Kasar et al. [10] also used SVMs on ruling lines to detect tables. Hao et al. [11] used loose rules for extracting table regions and classified the regions using CNNs. They also used textual information from PDFs to improve the model results. Rashid et al. [12] used positional information in every word to classify it as either a table or a non-table using dense neural networks.

After 2016, the research trod towards using deep learning models to solve the challenge. In 2017, many papers were presented which used object detection or segmentation models for table detection and parsing. Gilani et al. [13] employed distance transform encoded information in an image and applied Faster RCNN [14] on these images. Schreiber et al. [15] also used Faster RCNN for table detection and extraction of rows and columns. For parsing, they applied object detection algorithm on vertically stretched document images. Leveraging the tables' property to empirically contain more numeric data than textual data, Arif et al. [16] proposed to color code the document image to distinguish the numeric text and applied faster RCNN to extract table regions. Similarly, Siddique et al. [17] presented an end-to-end Faster-RCNN pipeline for table detection task and used Deformable Convolutional Neural Network [18] as feature extractor for its capability to mold its receptive field based on the input. He et al. [19] segmented the document image into three classes: text, tables, and figures. They proposed to use Conditional Random Field (CRF) to improve results from Fully Convolutional Network (FCN) conditioned on the output from the contour edge detection network. Kavasidis et al. [20] employed CRFs on saliency maps extracted by Fully Convolutional Neural Network to detect tables, and different type of charts.

Even though many researchers have shown that using object detection based approaches work well for table detection

---

[1] github.com/shahrukhqasim/TIES-2.0

P.O. B	KT	the	racks Rock	rock CPX	Serial and
before	Other Cal	(w/me)	performance	24.9	3,000
the	available compacted	j13	in	8.3.1.3.4	-56.0
Good Robertson	December	$	Alger ash-flow,	25	1987
Electronic	LIABILITIES MERCURY	h1	Selling	3.34	16.1
Systems	kg	167e1	I capable	1980	5.3.1.1
Ni SHARE	Japanese Dec	(radioactive/mca)	Springs	106	661.4
of	PROCEDURE L	3508005	EQUITY	73	1018
major	not	32500+800	Exploration	74.0	66,866
doubtful	b	533°	so NY	102335	5
Requires	isolation-related	california;	Based Effect	50	7
Hamilton taxes	working	$	Balance	10,187	1.0

(a) Category 1

NV ash-flow,	CD	EP	NO	axis
atmospheric muR	4,103	700	package heermannii	BGB tons
Glass	73	150	administrative	Increase operating
body,	40.4	0.07	NRC	and
Value S	1640	1.648-010	Revenues tah	section
Cs	390.00	59	N.A.	of AD
from late	118,311	30	surface .Current	Number Pressure
data,	48.88	150	Radial	of
to	480	1.59	expense	three
Fair	1,700	82,100	plagioclase, TH	Annual States

(b) Category 2

Items	clay				and Costs		Franklin Homi,
	NO Corp.	fp Assumptions	July Interest	roots	Bergen, U.S.	Snap-Up x	
shares YEARS	tax	(lat	19	Dodge Stock	NW	1.02	290
D Jun.	Se Total	$	265	Headed	subsystems	4	719,424
current	moderately and	20ny1996	2.00	for Data	of Income	673	20
Ended	loss THE		10-85	Ground weeks	NOTV	47	43
provided No.	air	367°	313	bore DUMP	drawdown and	0.04	480
BP	pluvial.		2.0000-01	Service	Field Others	2,017	05-31-81

(c) Category 3

Containerboard	noncurrent	Death	and	PRNTAR	ft Back
	loading emplacement,	Common	2.5	100	psi Ridge,
	plant	same Imaging	400	0.00	SRL BD
	for	well III	1	1111	charge
15-10	Yes	Vegas	0	1964	Three Smith
	Spent of	CA	68.0	63,020	FIGURE
	extensive Kentucky	triplex etc.	3,000	5.089-002	than
	Waste improvements	oz.	29.7	10	the and
2.1462	waste Taxes	paid TAUSCAT	21	18	dune Net

(d) Category 4

Fig. 1: Images with different difficulty categories. Category 1 images are plain images with no merging and with ruling lines. Category 2 adds different border types including occasional absence of ruling lines. Category 3 is the hardest one which introduces cell and column merging. Category 4 models camera captured images by linear perspective transform.
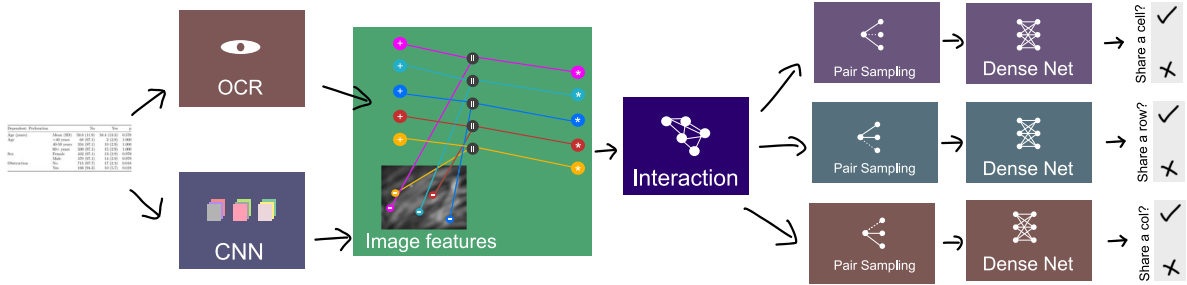

Fig. 2: A pictorial representation of our architecture. For a document image, a feature map is extracted using the CNN model and words' positions are extracted using an OCR engine or an oracle (+). Image features, corresponding to the words' positions, are gathered (-) and concatenated (||) with positional features to form the input features (*). An interaction network is applied to the input features to get representative features. For each vertex (word), sampling is done individually for cells, rows and columns classification. The representative features for every sample pair are concatenated again and used as the input for the dense nets. These dense nets are different for rows, columns and cells sharing. Sampling is only done during the training. During the inference, every vertex pair is classified.

and recognition, defining the parsing problem in the form of object-detection problem is hard, especially if the documents are camera captured and contain perspective distortions. Approaches like [15] partially solve the issue but it is still not a natural approach. It also makes it harder to use further features which could be extracted independently, for instance, the language features which could possibly hint towards the existence of a table.

In this paper, we define the problem using graph theory and apply graph neural networks to it. One of the initial research in graph neural networks is done by Scarselli et al. [21] where they formulated a comprehensive graph model based on contraction maps. In recent years, they have gained a lot of traction due to the increase in compute power and with the introduction of newer methods. Many notable works include [22], [23], [24]. Battaglia et al. [25] argued that relational inductive biases are the key to achieving human like-intelligence and showed how graph neural networks are essential for it.

The use of graphs in document processing is not new. There have been many papers published which employ graph-based models for a wide range of problems. Liang et al. [26] introduced a hierarchical tree-like structure for document parsing. Work presented by Wang [27] gained a lot of popularity and was used by many researchers afterward. Hu et al. [28] introduced a comprehensive graph model involving a Directed Acyclic Graph (DAG) with detailed definitions of various elements of a table. Recently, Koci et al. [29] presented an approach where they encoded information in the form of a graph. Afterward, they used a newly-proposed rule-based remove-and-conquer algorithm. Bunke et al. [30] provides a detailed analysis of different graph-based techniques employed in the context of document analysis. These methods make strong assumptions about the underlying structure which contradicts

with the philosophy of deep learning. Even though we are not the first ones to use graphs for document processing, to the best of our knowledge, we are the first ones to apply graph neural networks to our problem. We have done our experiments on the table recognition problem, however, this new problem definition applies to various other problems in document structural analysis. There are two advantages to our approach. For one, it is more generic since it doesn't make any strong assumptions about the structure and it is close to how humans interpret tables, i.e. by matching data cells to their headers. Secondly, it allows us to exploit graph neural networks towards which there has been a lot of push lately.

In particular, we make the following contributions:

1) Formulate table recognition problem as a graph problem which is compatible with graph neural networks
2) Design a novel differentiable architecture which reaps the benefits of both convolutional neural networks for image feature extraction and graph neural networks for efficient interaction between the vertices
3) Introduce a novel Monte Carlo based technique to reduce memory requirements of training
4) Fill the gap of large scale dataset by introducing a synthetic dataset
5) Run tests on two state-of-the-art graph based methods and emperically show that they perform better than a baseline network

II. DATASET

There are a few datasets for table detection and structure recognition published by the research community, including UW3, UNLV [31] and ICDAR 2013 table competition dataset [32]. However, the size of all of these datasets is limited. It risks overfitting in deep neural networks and hence, poor generalization. Many people have tried techniques such as transfer learning but these techniques cannot completely offset the utility of a large scale dataset.

We present a large synthetically generated dataset of 0.5 Million tables divided into four categories, which are visualized in Figure 1. To generate the dataset, we have employed Firefox and Selenium to render synthetically generated HTML. We note that synthetic dataset generation is not new and similar work [2] has been done before. Even though it will be hard to generalize algorithms from our dataset to the real world, the dataset provides a standard benchmark for studying different algorithms until a large scale real-world dataset is created. We have also published our code to generate further data, if required.

III. THE GRAPH MODEL

Considering the problem of table recognition, the ground truth is defined as three graphs wherein every word is a vertex. There are three adjacency matrices representing each of the graphs, namely cell-, row-, and column-sharing matrices. So if two vertices share a row i.e. both words belong to the same row, these vertices are taken to be adjacent to each other (likewise for cell and column sharing).

The prediction of a deep model is also done in the form of the three adjacency matrices. After getting adjacency matrices, complete cells, rows and columns can be reconstructed by solving the problem of maximal cliques [33] for rows and columns and connected components for cells. It is pictorially shown in Figure 3.

This model is valid not only for table recognition problem but can also be used for document segmentation. In that scenario, if two vertices (could be words again) share the same zone, they are adjacent. The resultant zones can also be reconstructed using the maximal clique problem.
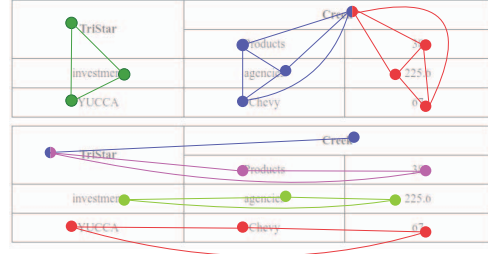

Fig. 3: Reconstructing the resultant column and row segments using maximal cliques. The upper figure shows column cliques while the bottom one shows row cliques. Note the merged vertex in both of these figures which belongs to multiple cliques.

IV. METHODOLOGY

All of the tested models follow the same parent pattern, shown in Figure 2, divided into three parts: the convolutional neural network for extraction of image features, the interaction network for communication between the vertices, and the classification part to label every paired vertices as being adjacent or not adjacent (class 0 or class 1) in each of the three graphs.

The algorithm for the forward pass is also given in Algorithm 1. It takes the image ($I \in \mathbb{R}^{h \times w \times c}$ — where h, w and c represent height, width and number of channels in the input image respectively), positional features ($F_p \in \mathbb{R}^{v \times 4}$ — where v represents number of vertices), and other features $F_o \in \mathbb{R}^{v \times o}$. In addition to this, it also takes the number of samples per vertex($s \in \mathbb{R}$) and three adjacency matrices ($A_{cells} \in \{0,1\}^{v \times v}$, $A_{rows} \in \{0,1\}^{v \times v}$ and $A_{cols} \in \{0,1\}^{v \times v}$) as the input during the training process. All the parametric functions are denoted by f and non-parametric functions by g. If all the parametric functions are differentiable, the complete architecture will be differentiable as well and hence, compatible with backpropagation.

The positional features include the coordinates of the upper left and the bottom right corner of each vertex. Other features consist only of the length of the word in our case. However, in a real-world dataset, natural language features [34] could also be appended which may provide additional information.

1) Convolutional neural network: A convolutional neural network (f_{conv}) takes an image (I) as its input and as the output, it generates the respective convolutional features ($I_f \in \mathbb{R}^{h' \times w' \times q}$ — w', h' and q being the width, height and number of channels of the convolutional feature map respectively). To keep parameter count low, we have designed a shallow CNN; however, any standard architecture can be used in its place. At the output of CNN, a gather operation (g_{gather}) is performed to collect convolutional features for each word corresponding to its spatial position in the image and form gathered features ($F_{im} \in \mathbb{R}^{v \times q}$). Since convolutional neural networks are translation equivariant, this operation works well. If the spatial dimensions of the output features are not the same as the input image (for instance, in our case, they were scaled down), the collect positions are linearly scaled down depending on the ratio between the input and output dimensions. The convolutional features are extended to the rest of the vertex features (g_{ext}).

2) Interaction: After gathering all the vertex features, they are passed as input to the interaction model (f_{int}). We have tested two graph neural networks to use as the interaction part which are the modified versions of [35] and [36] respectively. These modified networks are referred to as DGCNN* and GravNet* hereafter. In addition to these two, we have also tested with a baseline dense net (dubbed FCNN for Fully Connected Neural Network) with approximately the same number of parameters to show that the graph-based models perform better. For these three models, we have limited the total parameter count to $1M$ for a fair comparison. This parameter count also includes parameters of preceding CNN and the succeeding classification dense network. As the output, we get representative features ($F_{rep} \in \mathbb{R}^{v \times r}$ — r being the number of representative features) of each of the vertex which are used for classification.

3) Runtime pair sampling: Classifying every word pair is a memory intensive operation with memory complexity of $O(N^2)$. Since it would then scale linearly with the batch size, the memory requirements increase even further. To cater to this, we employed a Monte Carlo based sampling. The index sampling function is denoted by g_{sample}. this function would generate a fixed number of samples (s) for each of the vertex for each of the three problems (cell sharing, row sharing and column sharing).

Uniform sampling is highly biased towards class 0. Since we can't use a large batch size due to the memory constraints, the statistics are not sufficient to differentiate between the two classes. To deal with this issue, we changed the sampling distribution ($P_{samples}$) to sample, on average, an equal number of elements of class 0 and class 1 for each of the vertex. It can be easily done in a vectorized fashion as shown in Algorithm 1. Note that J in the algorithm denotes an all-one matrix. Different sets of samples are collected for each of the three classes for each of the vertex ($S_{cells} \in \mathbb{Z}^{v \times t}$, $S_{rows} \in \mathbb{Z}^{v \times t}$, $S_{cols} \in \mathbb{Z}^{v \times t}$). The values in these matrices represent the index of the paired samples for each of the vertex. For inference, however, we do not need to sample since we

don't need to use the mini-batch approach. Hence, we simply do it for every vertex pair. So, during training, $t = s$ and during inference, $t = v$.

Algorithm 1 Forward Pass

> **Input** $I, F_p, F_o, (A_{cells}, A_{rows}, A_{cols}, s)^2$
> **Output** $L_{cells}, L_{rows}, L_{cols}$

1: **function** PAIR_SAMPLING(A, s)
2:      $P_{samples} \leftarrow 0.5 * (1 - A) \cdot ((1 - A) \times J_{v,v})^{\circ -1}$
3:      $P_{samples} \leftarrow P_{samples} + 0.5 * A \cdot (A \times J_{v,v})^{\circ -1}$
4:      **return** $g_{sample}(P_{samples}, s)$
5: $I_f \leftarrow f_{conv}(I)$
6: $F_{im} \leftarrow g_{gather}(I_f, F_p)$
7: $F_{cat} \leftarrow g_{concat}(F_{im}, F_o, F_p)$
8: $F_{int} \leftarrow f_{int}(F_{cat})$
9: **if** training **then**
10:      $S_{cells} \leftarrow$ PAIR_SAMPLING(A_{cells}, s)
11:      $S_{rows} \leftarrow$ PAIR_SAMPLING(A_{rows}, s)
12:      $S_{cols} \leftarrow$ PAIR_SAMPLING(A_{cols}, s)
13: **else**
14:      $S_{cells} \leftarrow S_{rows} \leftarrow S_{cols} \leftarrow [x_{i,j}]_{v,v} \mid x_{i,j} = j$
15: $L_{cells} \leftarrow f_{cells}(S_{cells}, F_{int})$
16: $L_{rows} \leftarrow f_{rows}(S_{rows}, F_{int})$
17: $L_{cols} \leftarrow f_{cols}(S_{cols}, F_{int})$

---

4) Classification: After sampling, the elements from the output feature vector (F_{int}) of the interaction model and the elements from the sampling matrices are concatenated (g_{cat}) with each other in (f_{cells}, f_{rows}, and f_{cols}). These functions are parametric neural networks. As the output, we get three sets of logits $L_{cells} \in \mathbb{R}^{v \times t \times 2}$, $L_{rows} \in \mathbb{R}^{v \times t \times 2}$ and $L_{cols} \in \mathbb{R}^{v \times t \times 2}$. They can be used either to compute the loss and backpropagate through the function, or to predict the classes and form the resultant adjacency matrices.

V. RESULTS

Shahab et al [37] defined a set of metrics for detailed evaluation of the results of table parsing and detection. They defined criteria for correct and partial detection and defined a heuristic for labeling elements as under-segmented, over-segmented and missed. Among their criterion, two are the most relevant to our case, i.e. the percentage of the ground truth elements that are detected correctly (true positive rate) and the number of the predicted elements which do not have a match in the ground truth (false positive rate). In our case, as argued in Section III, the elements are cliques. So true positive rate and false positive rate is computed on all three graphs (cells, rows and columns) individually. This rate is averaged over the whole test set. These results are shown in Table I and Table II.

In addition to this, we also introduce another measure, i.e. perfect matching as shown in Table III. If all of the three predicted adjacency matrices are perfectly matched with the

---

[2]Only needed for training

TABLE I: True positive rate: the percentage of the GT cells/rows and the column cliques in category i which have a match in the predictions.

Method	$cells_1$	$rows_1$	$cols_1$	$cells_2$	$rows_2$	$cols_2$	$cells_3$	$rows_3$	$cols_3$	$cells_4$	$rows_4$	$cols_4$
FCNN	**99.9**	99.9	99.6	**99.9**	99.6	99.4	99.8	96.8	87.6	**99.9**	97.7	90.0
GravNet*	99.8	**100**	99.7	99.8	**99.9**	99.5	99.2	95.7	86.2	99.6	96.8	90.5
DGCNN*	99.8	99.9	**100**	**99.9**	**99.9**	**99.8**	**99.9**	**98.1**	**94.1**	99.8	**99.1**	**94.3**

TABLE II: False positive rate: the percentage of the detected cells/rows and the column cliques in category i which do not have a match in the ground truth.

Method	$cells_1$	$rows_1$	$cols_1$	$cells_2$	$rows_2$	$cols_2$	$cells_3$	$rows_3$	$cols_3$	$cells_4$	$rows_4$	$cols_4$
FCNN	**0.01**	0.5	16.7	**0.06**	3.05	14.1	0.12	10.4	32.1	**0.04**	6.31	26.4
GravNet*	0.15	**0.18**	6.56	0.17	0.79	9.01	0.58	10.2	33.6	0.28	7.81	25.28
DGCNN*	0.07	0.46	**0.79**	0.08	**0.22**	**1.09**	**0.06**	**4.8**	**14.6**	0.07	**4.39**	**10.6**

TABLE III: End-to-end accuracy/perfect matching.

Method	Category 1	Category 2	Category 3	Category 4
FCNN	42.4	54.6	10.9	31.9
GravNet*	65.6	58.6	13.1	31.5
DGCNN*	**96.9**	**94.7**	**52.9**	**68.5**

corresponding matrices in the ground truth, the parsed table is labeled as being end-to-end accurate. This is a strict metric but it shows how misleading can the statistics computed on the basis of individual table elements be due to the large class imbalance problem.

As expected, since there is no interaction between vertices in the FCNN, it performs worse than the graph models. Note however, that the vertices in this network are not completely segregated. They can still communicate in the convolutional neural network part. This builds the case to introduce graph

(a) Adjacent in Cells

(b) Adjacent in Rows

(c) Adjacent in Columns

Fig. 5: Figures show DGCNN* model's predictions (cells, rows and columns neighbors) for a randomly selected vertex shown in orange. 5(a) shows the cells predictions. All the vertices adjacent to the random vertex in cell-sharing graph are shown in green and those not adjacent are shown in blue. Likewise 5(b) and 5(c) show the adjacent neighbors in row- and column sharing graphs respectively.

(a) Cells Predictions

(b) Rows Predictions

(c) Columns Predictions

Fig. 4: Figures show DGCNN* model's predictions (cells, rows and columns). 4(a) shows the cells predictions. All the words in a single cell are assigned the same color. 4(b) shows the rows predictions. The words belonging to the same row are assigned the same color and the words spanned across multiple rows (cliques) are stripe-colored. Similarly, 4(c) shows the column predictions.

neural networks further into document analysis.

Category 3 tables show relatively poor results as compared

to category 4 tables. This is because category 4 images also contain those from category 1 and 2 to study the effect of perspective distortion on simpler images. We conclude that while the graph networks struggle with merged row and columns, they gracefully handle the perspective distortions.

VI. CONCLUSION AND FUTURE WORK

In this work, we redefined the structural analysis problem using the graph model. We demonstrated our results on the problem of table recognition and we also argued how several other document analysis problems can be defined using this model. Convolutional neural networks are the most suited at finding representative image features and graph networks are the most suited at fast message passing between vertices. We have shown how we can combine these two abilities using the gather operation. So far, we only used positional features for the vertices, but for a real-world dataset, natural language processing features like GloVe can also be used. In conclusion, graph neural networks work well for structural analysis problems and we expect to see more research in this direction in the near future.

REFERENCES

[1] T. Kieninger and A. Dengel, "The t-recs table recognition and analysis system," vol. 1655, 11 1998, pp. 255–269.

[2] Y. Wangt, I. Phillipst, and R. Haralick, "Automatic table ground truth generation and a background-analysis-based table structure extraction method," in *Proceedings of Sixth International Conference on Document Analysis and Recognition*. IEEE, 2001, pp. 528–532.

[3] J. Hu, R. S. Kashi, D. P. Lopresti, and G. Wilfong, "Medium-independent table detection," in *Document Recognition and Retrieval VII*, vol. 3967. International Society for Optics and Photonics, 1999, pp. 291–303.

[4] B. Gatos, D. Danatsas, I. Pratikakis, and S. J. Perantonis, "Automatic table detection in document images," in *International Conference on Pattern Recognition and Image Analysis*. Springer, 2005, pp. 609–618.

[5] S. Tupaj, Z. Shi, C. H. Chang, and H. Alam, "Extracting tabular information from text files," *EECS Department, Tufts University, Medford, USA*, 1996.

[6] R. Zanibbi, D. Blostein, and J. R. Cordy, "A survey of table recognition," *Document Analysis and Recognition*, vol. 7, no. 1, pp. 1–16, 2004.

[7] I. A. Doush and E. Pontelli, "Detecting and recognizing tables in spreadsheets," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. ACM, 2010, pp. 471–478.

[8] F. Shafait and R. Smith, "Table detection in heterogeneous documents," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. ACM, 2010, pp. 65–72.

[9] J. Chen and D. Lopresti, "Table detection in noisy off-line handwritten documents," 09 2011, pp. 399–403.

[10] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, "Learning to detect tables in scanned document images using line information," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1185–1189.

[11] L. Hao, L. Gao, X. Yi, and Z. Tang, "A table detection method for pdf documents based on convolutional neural networks," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. IEEE, 2016, pp. 287–292.

[12] S. F. Rashid, A. Akmal, M. Adnan, A. Adnan Aslam, and A. Dengel, "Table recognition in heterogeneous documents using machine learning," 11 2017, pp. 777–782.

[13] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, "Table detection using deep learning," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 771–776.

[14] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[15] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. 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)*, vol. 1. IEEE, 2017, pp. 1162–1167.

[16] S. Arif and F. Shafait, "Table detection in document images using foreground and background features," 12 2018, pp. 1–8.

[17] S. A. Siddiqui, M. I. Malik, S. Agne, A. Dengel, and S. Ahmed, "Decnt: Deep deformable cnn for table detection," *IEEE Access*, vol. 6, pp. 74 151–74 161, 2018.

[18] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 764–773.

[19] D. He, S. Cohen, B. Price, D. Kifer, and C. Lee Giles, "Multi-scale multi-task fcn for semantic page segmentation and table detection," 11 2017, pp. 254–261.

[20] I. Kavasidis, S. Palazzo, C. Spampinato, C. Pino, D. Giordano, D. Giuffrida, and P. Messina, "A saliency-based convolutional neural network for table and chart detection in digitized documents," 04 2018.

[21] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[22] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR.org, 2017, pp. 1263–1272.

[23] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[24] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[25] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[26] J. Liang, I. T. Phillips, and R. M. Haralick, "An optimization methodology for document structure extraction on latin character documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 7, pp. 719–734, 2001.

[27] X. Wang, "Tabular abstraction, editing, and formatting," 1996.

[28] J. Hu, R. S. Kashi, D. P. Lopresti, and G. Wilfong, "Table structure recognition and its evaluation," in *Document Recognition and Retrieval VIII*, vol. 4307. International Society for Optics and Photonics, 2000, pp. 44–56.

[29] E. Koci, M. Thiele, W. Lehner, and O. Romero, "Table recognition in spreadsheets via a graph representation," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 139–144.

[30] H. Bunke and K. Riesen, "Recent advances in graph-based pattern recognition with applications in document analysis," *Pattern Recognition*, vol. 44, pp. 1057–1067, 05 2011.

[31] "Table ground truth for the uw3 and unlv datasets (dfki-tgt-2010)," http://tc11.cvc.uab.es/datasets/DFKI-TGT-2010_1.

[32] M. Göbel, T. Hassan, E. Oro, and G. Orsi, "Icdar 2013 table competition," in *2013 12th International Conference on Document Analysis and Recognition*. IEEE, 2013, pp. 1449–1453.

[33] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *Communications of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.

[34] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[35] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *arXiv preprint arXiv:1801.07829*, 2018.

[36] S. R. Qasim, J. Kieseler, Y. Iiyama, and M. Pierini, "Learning representations of irregular particle-detector geometry with distance-weighted graph networks," 2019. [Online]. Available: https://arxiv.org/abs/1902.07987

[37] A. Shahab, F. Shafait, T. Kieninger, and A. Dengel, "An open approach towards the benchmarking of table structure recognition systems," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. ACM, 2010, pp. 113–120.