Array 15 (2022) 100220

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/locate/array

Tablext: A combined neural network and heuristic based table extractor

Zach Colter, Morteza Fayazi *, Zineb Benameur-El Youbi, Serafina Kamp, Shuyan Yu, Ronald Dreslinski

Department of EECS, University of Michigan, Ann Arbor, United States

ARTICLE INFO

Keywords:
Table extraction
CNN
Deep learning
Image processing
OCR

ABSTRACT

A significant portion of the data available today is found within tables. Therefore, it is necessary to use automated table extraction to obtain thorough results when data-mining. Today's popular state-of-the-art methods for table extraction struggle to adequately extract tables with machine-readable text and structural data. To make matters worse, many tables do not have machine-readable data, such as tables saved as images, making most extraction methods completely ineffective. In order to address these issues, a novel, open-source, general format table extractor tool, Tablext, is proposed. This tool uses a combination of computer vision techniques and machine learning methods to efficiently and effectively identify and extract data from tables. Tablext begins by using a custom Convolutional Neural Network (CNN) to identify and separate all potential tables. The identification process is optimized by combining the custom CNN with the YOLO object detection network. Then, the high-level structure of each table is identified with computer vision methods. This high-level, structural meta-data is used by another CNN to identify exact cell locations. As a final step, Optical Characters Recognition (OCR) is performed on every individual cell to extract their content without needing machine-readable text. This multi-stage algorithm allows for the neural networks to focus on completing complex tasks, while letting image processing methods efficiently complete the simpler ones. This leads to the proposed approach to be general-purpose enough to handle a large batch of tables regardless of their internal encodings or their layout complexity. Additionally, it becomes accurate enough to outperform competing state-of-the-art table extractors on the ICDAR 2013 table dataset.

1. Introduction

Tables are widely used in documents, articles, web-pages, etc. as they can concisely show complex information in a way that is suitable for human readers [1]. Autonomous table extraction enables translating this abundant amount of information to a machine-readable format which has broad applications in data-mining and information-retrieval [2,3]. The complexity of some tables' layouts and the various formats in which they may be found are current problems that prevent the existence of a universal, highly-accurate, automatic table extraction tool. The purpose of this paper is to introduce a highly accurate extractor that can extract previously unobtainable information.

Formats such as Hypertext Markup Language (HTML), Portable Document Format (PDF), and Portable Network Graphics (PNG) are common encodings that may contain tables. Because of the many different ways to represent a table, designing separate extraction methods for every encoding is time-consuming and prone to errors. Moreover, an environment may have multiple formats that are not encoded uniformly. For instance, a web-page may contain an image that is not encoded in HTML. Unifying internal encodings of files is the first step to overcome universality issues.

In this work, we present an open-source[1] table extractor tool, Tablext, which addresses all the above-mentioned problems. Tablext solves the universality issue by extracting data from images. There are several Java and Python open-source libraries to convert specific file formats to images such as pdf2image [4] (PDF to PNG), GrabzIt [5] (HTML to PNG), etc.

Tablext uses separate Convolutional Neural Networks (CNNs) [6,7] to identify and extract information from tables, but it does not solely rely on them. Before data extraction, Tablext uses common conventional computer vision techniques, such as line identification, to extract positional information from tables before another CNN is used to fix small mistakes. This approach can be both faster and more accurate than having a single neural network.

* Corresponding author.
E-mail addresses: zcolter@umich.edu (Z. Colter), fayazi@umich.edu (M. Fayazi), zinebbe@umich.edu (Z.B.-E. Youbi), serafibk@umich.edu (S. Kamp), shuyan@umich.edu (S. Yu), rdreslin@umich.edu (R. Dreslinski).

[1] Source code for the Tablext can be downloaded from https://github.com/idea-fasoc/datasheet-scrubber/tree/master/src/table_extraction.

https://doi.org/10.1016/j.array.2022.100220
Received 20 May 2022; Received in revised form 28 June 2022; Accepted 2 July 2022
Available online 7 July 2022

By first identifying important features within the tables, each neural network only has to solve a small sub-task. This means that the size required for each network is drastically reduced. The smaller size of the neural networks reduces the computation requirement as well as the risk of overfitting.

Tablext begins by using an efficient CNN to identify the exact location of the tables within the input images. Using an ensemble of this CNN and an object detection network, YOLO [8], allows for the extraction algorithm to be confined to relevant areas. By isolating the relatively expensive extraction algorithm to areas found by the identification CNN, Tablext is efficient in handling massive databases.

Tablext achieves a high extraction accuracy by both identifying the lines separating various cells, as well as looking at the positions of the text within the tables. Together, these methods can recognize complex table structures. It can generally be assumed that the identified lines correctly split the table into cells, but when no lines exists or they cannot be correctly identified, the positional data is used to separate the cells. After correctly segmenting the table into cells, the data is cleaned by a CNN before the open-source Optical Character Recognition (OCR) software, Tesseract [9], identifies the text in every individual cell.

The main contributions of this paper can be summarized as follows.

- A new method to identify cells that utilizes the positions of the data and lines in order to obtain high precision preliminary results at a low computational cost.
- A hybrid pipeline that allows the neural network to focus its attention on complex problems, resulting in a high accuracy.
- A novel, general format, open-source tool that identifies and extracts all complex layout tables with a high accuracy within various internal encodings files.

The rest of the paper is organized as follows. Section 2 briefly presents related works. Section 3 describes the proposed method for identifying the location of the tables. Section 4 explains Tablext's approaches for data extraction. Section 5 shows the final results and comparisons to other table extractors. Finally, the paper is concluded in Section 6. Throughout this paper, four different tables (Table A, Table B, Table C, Table D) are used to explain various concepts. For clarity, figures will declare which original table they are referencing.

2. Related work

Several methods have been studied table identification and extraction. However, some of them describe various approaches to identify tables from different types of documents and do not focus on data extraction [10–16].

Several systems have been proposed for the identification and extraction of tables within PDF files [2,17–20]. Oro and Ruffolo [2] find positional data for the text and they use an agglomerative hierarchical clustering algorithm [21] to combine words into lines. The main limitation of this approach is that it only supports unlocked, machine-readable PDFs. However, a large amount of tables are found in locked PDFs or images. Perez-Arriaga et al. [17] find columns and rows by comparing the locations of the text-boxes and combining text-boxes below a distance threshold. Although the method has a decent recall, its false-positive rate is high. Liu et al. [20] propose a method in which, tables are found by grouping text with similar positions and font size. This approach works with HTML format in addition to PDF files. However, it is only optimized for research papers and its extraction performance is poor especially on general documents. Similar to the other mentioned studies, this method cannot work with locked PDF files or images.

Some other studies focus on table extraction that are not related to PDFs [1,3,22]. Tengli et al. [1] propose a machine-learning-based technique for table extraction from HTML files. Nishida et al. [22] focus on HTML extraction too. They use a Recurrent Neural Network (RNN) model to extract data from HTML tags. These such approaches are not applicable to formats other than HTML and are therefore, limited in scope. Pinto et al. [3] propose a method to identify the horizontal lines within tables that are formatted using white space and fixed fonts to align columns in Web pages, however they do not concentrate on table extraction.

Koci et al. and Puha et al. [23,24] propose methods that can handle tables in more diverse formats. Koci et al. [23] propose a graph representation of tables so that they can use heuristics approaches in order to extract information. Their proposed methodology is optimized to extract data from various spreadsheets. Puha et al. [24] propose a heuristic method for general table extraction. They recognize tables in images by first using OCR to identify data within cells, then using the locations of the data.

Many state-of-the-art general table extraction methods solely use neural networks. Qasim et al. [25] use a Graph Neural Network (GNN) to identify cell locations within tables. DeepDeSRT [26] uses separate, specially made neural networks to identify tables and extract their data, while TableNet [27] uses a single network for both identification and extraction. They argue this is an efficient approach since these tasks are interdependent. However, this also means that the network is less optimized for both tasks. Tablext outperformed both of these methods when tested on a common dataset. A comparison between the related works has been summarized in Table 1.

By utilizing Machine Learning (ML), modern table extractors can handle more diverse tables than earlier works. However, identifying and extracting data is a difficult undertaking that requires many sub-tasks to be completed. Many of these tasks, such as identifying lines that can make up a table's structure, are good candidates for conventional computer vision methods. Instead of forcing a neural network to learn and solve every one of these tasks, Tablext introduces a new approach that combines both conventional computer vision and ML techniques. This allows the neural network to be smaller in size and focused on its particular task of identifying abnormal cells. This is especially important given the limited amount of labeled training data available for tables.

3. Table location identification

While the main focus of this paper is table extraction, a novel algorithm is created and used to identify tables before extraction. It is imperative to find the table locations before extraction because doing so increases both the performance and accuracy of the algorithm. The boost in performance comes from reduced search space for the complex information extraction. The increased accuracy is a result of the table identification network separating tables from one another and removing irrelevant text and lines that could potentially confuse the table extractor. To further increase the accuracy of the table identification, it is later combined with YOLO.

The identification algorithm needs to find both the minimum and maximum X and Y coordinates for each table that exists within each page. To efficiently calculate these coordinates, two relatively simple CNNs are used in succession. These two networks identify at which Y and X coordinates, respectively, a table exists.

All input images, after being converted from any file format, are first converted to gray-scale and resized. Preliminary tests show that a width of 800 pixels allows for accurate table extraction without a significant performance penalty. The height at this stage is scaled proportionally to avoid distortion. This resized image cannot directly be sent to a conventional CNN because its height is variable. Additionally, the large size of the input data would require a significant amount of memory for each batch, slowing down the training process. In order to reduce the input size and make the input regular, the image is sliced into horizontal strips, each with a height of 64 pixels. This number is chosen to reduce the training time without significantly degrading accuracy.

While cutting the image into strips reduces the input and output size of the network, this process introduces a new problem. The accuracy

Table 1
Comparison of related works.

Work	Table identification	Table extraction	File format	Approach
[10,12,16]	✓	✗	All	Image processing w/o ML
[11]	✓	✗	PDF	PdfExtra paradigm
[13]	✓	✗	All	DNN
[14]	✓	✗	PDF	CNN
[15]	✓	✗	All	SVM
[1]	✗	✓	HTML	ML
[22]	✗	✓	HTML	RNN
[23]	✗	✓	Spreadsheets	Graph representation
[24]	✗	✓	All	Image processing w/o ML
[25]	✗	✓	All	GNN
[2]	✓	✓	PDF	Agglomerative hierarchical clustering algorithm
[17]	✓	✓	PDF	Comparing the locations of the text-boxes
[20]	✓	✓	PDF & HTML	Grouping text
[26]	✓	✓	All	NN
[27]	✓	✓	All	NN
Tablext	✓	✓	All	Heuristic-based + CNN

of identifying a table at the top and the bottom of the slices is lower than identifying a table near the middle. Intuitively this is because the network only has information in one direction from these edge outputs. To combat this issue, each strip only detects the existence of a table within its innermost 32 rows of pixels. To cover the entire table, a moving window approach is used where each input strip partially overlaps its adjacent strips.

The neural network has 4 output nodes for every horizontal strip. Each node determines if there is a table at a particular location within the centermost 32 pixels. In the innermost 32 pixels, node $i, 0 \leq i \leq 3$, of the neural network is trained to identify if a table exists between any of the $[8 * i, 8 * i + 7]$ pixels. The outputs of all the slices are concatenated and stored in an array that identifies in which rows a table exists. With this approach, the top and bottom of each page cannot be searched for tables, because only the inner pixels are represented in the outputs. To fix this problem, the input image is extended by 16 pixels at the top and bottom with a white border.

Fig. 1 shows the concatenated output of all the slices from the neural network. In this figure, the thin horizontal boxes represent every row the network checked for a table. These boxes show the Y locations where the neural network predicts a table exists. While this particular example shows a table that takes up the entire width of the document, this network can also identify tables that are within one of several columns on the page. The next step is to clean the output by concatenating rows of positively identified tables into groups. If no groups are created, because none of the rows contain a table, Tablext outputs that no table is found in the image. Otherwise, the groups are prepared and sent to the second neural network in order to find the correct X coordinates. This process is simpler than having to find the correct Y coordinates as the amount of area to search is smaller. For this reason, a CNN is used without a sliding window approach.

Each group of rows is resized into a 400 × 400 pixel square to efficiently achieve high accuracy. This input size is obtained by preliminary testing. The image is sent into a CNN that identifies which columns contain a table. Fig. 2 shows this technique being applied to the bottom group of rows in Fig. 1. The actual result produced by the network looks distorted to a human because the table is scaled into a square. Instead a representative, recreated image is shown. Here the vertical boxes show where the network predicts that a table exists. If the table is only in a single column of a multi-column image, the lines would cover only that section. This network can also identify two separate adjacent tables. This is achieved by not identifying a table in between the two table groups. Each vertical group is separately sent to this second stage. As an example, Fig. 1 would send three separate images into the second network.

The columns are concatenated into groups, in a similar manner to the rows. The scaling that converted the group of rows into the square, shown in Fig. 2, is inverted and applied to the X coordinates. Additionally, the scaling used to convert the original image into the resized image, shown in Fig. 1, is inverted and applied to the Y coordinates. Doing this gives accurate coordinates of each table's start and end position regarding the original image. Fig. 3 shows a cutout of the first table, using the found X and Y coordinates.

The CNN by itself tends to over propose table regions because the existence of lines on the page are very influential to the model. This causes the CNN to predict areas around these lines as tables whether these lines correspond to tables or not. This will cause the extraction portion of the algorithm to take more time as it has to run OCR on these extra regions to figure out they are not tables. To counteract this, a YOLO [8] network is trained on table images to see if this more popular object detection method could outperform the CNN. YOLO typically identifies all the right table regions, but the bounding boxes that it proposes do not cover the entire table. For this reason, an ensemble of the CNN and YOLO is used to create an optimal identification model. Both networks first independently propose table regions. When these models propose regions that overlap, the union of these two regions is taken. This ensures both that the final regions are likely to contain tables and that the regions will likely contain the entirety of the tables.

While this approach can perfectly identify tables in multiple columns, there is an extremely rare case where two adjacent tables in separate columns have vastly different heights. In this case, extra data could be identified as a table. This rare issue is rectified later because the proposed extraction method can tolerate some extra area identified as a table.

4. Table extraction

After all of the tables are identified, the extraction method starts independently on each table. In order to process high resolution images efficiently, a resized copy of each individual table is created with a width of 800 pixels and a proportional height. This copy is used for a majority of the cell identification steps, however the original image is kept because it is used later during OCR.

Several methods are used to identify and isolate the cells within each table. The table extraction begins by identifying the high-level, general structure of the table. It identifies the vertical and horizontal lines, both visible and implied, that correctly divide a majority of the cells. There are several ways to locate this information within the tables. One approach is to identify the visible lines separating various cells. Another method is to look at the positions of the text within the tables and extrapolate how the cells should be positioned in accordance to each other. Tablext uses both of these approaches in order to obtain the structure of the table. Afterwards, a neural network is used to clean up any cells that do not conform to the general structure. Fig. 4 shows a table that already went through table identification. This will be used as an example to explain the steps of the table extraction.

MCP3021

DC ELECTRICAL SPECIFICATIONS (CONTINUED)

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5.0V, V_{SS} = GND, R_{PU} = 2 kΩ, T_A = -40°C to +85°C, I²C Fast Mode Timing: f_{SCL} = 400 kHz (Note 3).

Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Input Leakage Current	I_{LI}	-1	—	+1	µA	V_{IN} = V_{SS} to V_{DD}
Output Leakage Current	I_{LO}	-1	—	+1	µA	V_{OUT} = V_{SS} to V_{DD}
Pin Capacitance (all inputs/outputs)	C_{IN}, C_{OUT}	—	—	10	pF	T_A = 25°C, f = 1 MHz; (Note 2)
Bus Capacitance	C_B	—	—	400	pF	SDA drive low, 0.4V
Power Requirements						
Operating Voltage	V_{DD}	2.7	—	5.5	V	
Conversion Current	I_{DD}	—	175	250	µA	
Standby Current	I_{DDS}	—	0.005	1	µA	SDA, SCL = V_{DD}
Active Bus Current	I_{DDA}	—	—	120	µA	Note 4
Conversion Rate						
Conversion Time	t_{CONV}	—	8.96	—	µs	Note 5
Analog Input Acquisition Time	t_{ACQ}	—	1.12	—	µs	Note 5
Sample Rate	f_{SAMP}	—	—	22.3	ksps	f_{SCL} = 400 kHz (Note 1)

Note 1: Sample time is the time between conversions after the address byte has been sent to the converter. Refer to Figure 5-6.
2: This parameter is periodically sampled and not 100% tested.
3: R_{PU} = Pull-up resistor on SDA and SCL.
4: SDA and SCL = V_{SS} to V_{DD} at 400 kHz.
5: t_{ACQ} and t_{CONV} are dependent on internal oscillator timing. See Figure 5-5 and Figure 5-6 in relation to SCL.

TEMPERATURE SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5.0V, V_{SS} = GND.

Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Temperature Ranges						
Operating Temperature Range	T_A	-40	—	+125	°C	
Extended Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, SOT-23	θ_{JA}	—	256	—	°C/W	

Fig. 1. (Table A) Each horizontal boxes represents a row where a table is identified. The bottom group is highlighted with a thick box (red) for clarification. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5.0V, V_{SS} = GND.

Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Temperature Ranges						
Operating Temperature Range	T_A	-40	—	+125	°C	
Extended Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, SOT-23	θ_{JA}	—	256	—	°C/W	

Fig. 2. (Table A) Vertical boxes represent every column where a table is identified.

Electrical Characteristics: Unless otherwi se noted, all parame ters apply at V_{DD} = 5.0V, V_{S} = GND.

Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Temperature Ranges						
Operating Temperature Range	T_A	-40	—	+125	°C	
Extended Temperature Range	T_A	-40	—	+125	°C	
Storage Temperature Range	T_A	-65	—	+150	°C	
Thermal Package Resistances						
Thermal Resistance, SOT-23	θ_{JA}	—	256	—	°C/W	

Fig. 3. (Table A) A cutout of the original table using the coordinates identified.

Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)

Cornerstone Investor	Investment Amount (US$ in million)#	Number of Offer Shares (rounded down to nearest whole board lot of 500 Shares)	Approximate % of total number of Offer Shares: Assuming the Over-allotment Option is not exercised	Approximate % of total number of Offer Shares: Assuming the Over-allotment Option is exercised in full	Approximate % of total Shares in issue immediately following the completion of Capitalization Issue and Global Offering: Assuming the Over-allotment Option is not exercised	Approximate % of total Shares in issue immediately following the completion of Capitalization Issue and Global Offering: Assuming the Over-allotment Option is exercised in full
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC Tetrad Ventures Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Fig. 4. (Table B) A table sample identified by the proposed table identification method.

Cornerstone Investor	Investment Amount (US$ in million)#	Number of Offer Shares (rounded down to nearest whole board lot of 500 Shares)	Approximat % of total number of Offer Shares: Assuming the Over-allotment Option is not exercised	Approximate % of total number of Offer Shares: Assuming the Over-allotment Option is exercised in full	Approximate % of total Shares in issue immed ately following the completion of Capitalization Issue and Gl bal Offering: Assuming the Over-allotment Option is not exercised	Approximate % of total Shares in issue immed ately following the completion of Capitalization Issue and Gl bal Offering: Assuming the Over-allotment Option is exercised in full
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC Tetrad Ventures Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Fig. 5. (Table B) Inferred lines shown with thick bright (yellow) lines and real lines shown with thin dark (blue) lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Real line identification

Vertical and horizontal lines are commonly used to separate and organize the contents of tables. To identify these lines, the second-derivative gradient field is approximated for the gray-scale image of the table. Assuming that $f(x, y)$ represents the brightness of a pixel at the location (x, y). The values of $\frac{\partial^2}{\partial x^2} f(x, y)$ and $\frac{\partial^2}{\partial y^2} f(x, y)$ are approximated for each pixel by looking at their adjacent pixels and the gradients of them. These gradients are then stored into separate matrices. Then a 3×3 max pooling is independently applied to these matrices, to account for noise in the image and imperfect lines. The max pooling has a stride of 1 in both X and Y directions to retain as much precision as possible when identifying the lines.

With this information, the vertical lines are found by searching for vertical segments where all the pixels in the segment have high $\frac{\partial^2}{\partial x^2} f(x, y)$ values relative to the pixels in rest of the image and similar $\frac{\partial^2}{\partial y^2} f(x, y)$ values to one another. Similarly, horizontal lines are found by searching for horizontal segments where all the pixels in the segment have high $\frac{\partial^2}{\partial y^2} f(x, y)$ values relative to the pixels in rest of the image and similar $\frac{\partial^2}{\partial x^2} f(x, y)$ values to one another.

The segment length has to be a small fraction of the table's height or width for the vertical and horizontal lines respectively. However, once a line is found, that line is extended to cover the entire width or height of the table. Identifying lines in this way helps define the structure of the table, but some cells that do not conform to the general format might be split by lines that do not cover the entire width or height of the table. This issue is solved later with a neural network that stitches together improperly cut cells. Fig. 5 depicts a debug image, created by the real line identification algorithm, that shows the few real lines in dark blue. The rest of the lines in this figure are created with another process that will be explained in the following section.

4.2. Inferred line identification

Real line identification alone is sufficient for regular tables that have all of their data properly split into cells with defined borders. However, as Fig. 4 shows, many tables only have proper borders for some of the cells or do not have any well defined lines at all. To correctly identify cells that real line identification alone could not, Tablext uses the positions of the data. Finding the locations of the data is trivial. All of the high contrast pixels, not already identified as a real line, belong to the data. With the locations of the data known, Tablext attempts

Table 2. Digital Interface Timing

Parameter	Symbol	Min	Typ	Max	Unit
CONVERSION TIME—CNV RISING EDGE TO DATA AVAILABLE	t_{CONV}	270	290	320	ns
ACQUISITION PHASE[1]	t_{ACQ}				
AD4002		290			ns
AD4006		790			ns
AD4010		1790			ns
TIME BETWEEN CONVERSIONS	t_{CYC}				
AD4002		500			ns
AD4006		1000			ns
AD4010		2000			ns
CNV PULSE WIDTH ($\overline{CS}$ MODE)[2]	t_{CNVH}	10			ns
SCK PERIOD ($\overline{CS}$ MODE)[3]	t_{SCK}				
VIO > 2.7 V		9.8			ns

Fig. 6. (Table C) Initial sparse table before inferred vertical lines. A red box has been placed around the, difficult to handle, referenced sparse column for clarity. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

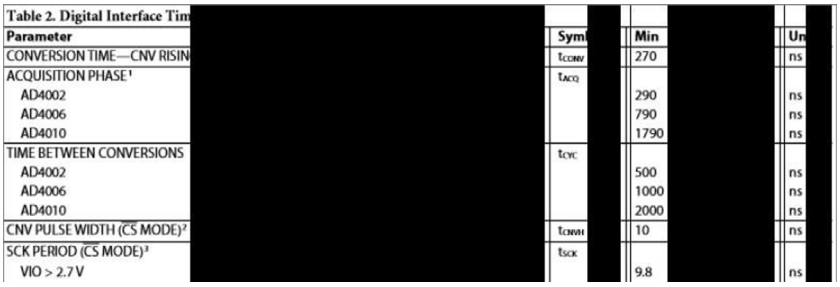


Fig. 7. (Table C) Sparse table with initialized inferred vertical lines.

to segment the pieces of data into columns and rows. To achieve this, Tablext creates straight lines that do not intersect the data. These such lines will be referred to "inferred lines" as the table requires these non-existent lines to properly convey its information. Fig. 5 shows when all inferred and real lines of Fig. 4 are drawn.

The task of finding inferred lines is made harder by the existence of cells that do not conform to the general table's structure. In Fig. 4, multiple cells span two columns. If inferred lines are defined as lines that do not overlap any pieces of data, the columns underneath the cells that span multiple columns would not be separated. In order to solve this issue, Tablext starts by defining the term "threshold distance".

For vertical lines, the threshold distance is the proportion of the table's height an inferred line must not touch any data in order to be considered valid. If the table is dense, meaning that most cells contain data, a small, static threshold distance would produce accurate results. However if the table is sparse, meaning that many cells do not contain data, this approach would not work. A small, static threshold distance would combine adjacent empty cells, into wide inferred lines. Figs. 6 and 7 show the inferred vertical lines initialization. Fig. 7 illustrates that a few pieces of data within these columns or rows could be overwritten by the inferred lines.

To accommodate both sparse and dense tables, the threshold distance is adaptable. Algorithm 1 describes a simplified procedure of acquiring the final inferred lines with a variable threshold distance. In the algorithm, possible inferred lines refers to inferred lines found at the given threshold distance, Δ is a small learned constant, and number of groups refers to the amount of separable groups of inferred lines that exist within the possible inferred lines.

For a visual representation of this algorithm, images of this process are produced by the code. Fig. 8 shows the final output after applying this algorithm to the sparse table in Fig. 7. At the start, the vertical inferred lines, represented as black columns, completely cover the column beginning with "Typ". If the threshold distance was a low static number, this column would be merged with one of its neighboring columns. However, by using Algorithm 1, the lines, over several iterations, split around the "Typ" column as shown by Fig. 8.

During this process, every inferred line is assigned a "quality score". The quality score is the percentage of the maximum possible line length

Algorithm 1: Setting the threshold distance

```
Variables: num_group1, num_group2, threshold_distance, temp_lines
Output: final_lines
Start:
    threshold distance = 0.6; num_group1 = 0; final_lines = [];

  Loop:
    temp_lines = possible inferred lines;
    num_group2 = number of groups;
    if num_group2 ≥ num_group1:
      num_group1 = num_group2;
      final_lines = temp_lines;
      threshold_distance += Δ;
      go to Loop;
    return final_lines;
```

that a particular inferred line covers without intersecting a piece of data. These quality scores are used later for combining the various lines.

Inferred horizontal lines are created in a similar manner as inferred vertical lines, except the threshold distance is a percentage of the table's width instead of the table's height. To better show this complete process, both vertical and horizontal inferred line identification are used on Fig. 4. The resulting inferred lines are used in conjunction with previously calculated real lines to create the debug image shown in Fig. 5.

4.3. Final structural line identification

Now that all of the real and inferred lines have been located, Tablext begins to group all of this information so that it has a high level understanding of the particular table's layout. Inferred lines are combined with their neighbors, if they are within close proximity, to create different groups of inferred lines. If a real line is within a group of inferred lines, or it is only several pixels away from a group, only the real line is kept. The real lines are considered to be the ground truth,

Table 2. Digital Interface Timing					
Parameter	**Symb**	**Min**	**Typ**	**Max**	**Un**
CONVERSION TIME—CNV RISING EDGE TO DATA AVAILABLE	t_{CONV}	270	290	320	ns
ACQUISITION PHASE[1]	t_{ACQ}				
AD4002		290			ns
AD4006		790			ns
AD4010		1790			ns
TIME BETWEEN CONVERSIONS	t_{CYC}				
AD4002		500			ns
AD4006		1000			ns
AD4010		2000			ns
CNV PULSE WIDTH ($\overline{CS}$ MODE)[2]	t_{CNVH}	10			ns
SCK PERIOD ($\overline{CS}$ MODE)[3]	t_{SCK}				
VIO > 2.7 V		9.8			ns

Fig. 8. (Table C) Sparse table with incremented inferred vertical lines.

so nearby inferred line groups are redundant. If there are no real lines near an inferred line group, the quality scores of the inferred lines are used to determine which line in the group is the best line to split the cells. A Simple Moving Average (SMA) of the quality scores, looking both forwards and backwards two pixels, is calculated for each group. Lines that do not have a valid inferred line are given a zero quality score. Using an SMA helps reduce the impact of noise in the image, while also fixing corner cases where a single pixel thick inferred line might sneak between characters in a piece of data.

The index that contains the maximum SMA value is taken for each inferred group which is not near a real line. Then, the locations of the real lines and the maximum SMA indexes are concatenated together. These combined lines are the final lines for the high level table identification. This process is completed independently for both the vertical and horizontal lines. Fig. 9 is created by applying this process on the lines found in Fig. 5.

4.4. Neural network correction

Since the final lines describe the regular structure of the table, cells that do not conform to that structure might be sliced into two or more pieces. Fig. 9 shows several cells that are cut in half by the final lines. In order to fix this issue, a CNN is used to recombine improperly sliced cells. A simpler solution would be to check if any high contrast pixels exist underneath an inferred line. However, this method has several problems. One issue is if characters in a cell are spaced out significantly, a vertical inferred line could exist in between two characters and the cell's contents would be split into two cells.

While these concerns can be partially alleviated by looking a certain distance to the left and right for high contrast pixels, this introduces a new risk of merging cells unnecessarily. Additionally, merging with that method can be triggered by noise in the tables. For these reasons, a neural network has been created to handle the possible merging of cells.

The CNN takes in two adjacent cells as inputs and resizes them to be 100×100 pixels each. Testing shows that this size allows the OCR to accurately extract data. The cells are resized independently so that the line that divides them is always in the center of the two cells. If instead, the cells are merged and then resized, the CNN would need to predict which data belongs to which cell or it would require the location of the dividing line as an additional input. These approaches result in significantly lower accuracy and higher complexity, respectively.

The proposed CNN outputs the chance that these two cells are merged and if one or both of the cells are empty. By using a neural network to compute this information, the effect of imperfections in the image can be greatly reduced. It is important to know which cells are empty to avoid running OCR on those cells. This increases performance, as the OCR is the most time consuming step. The following sections talk about the implementation details of the CNN, the training data, and the post CNN concatenation.

		Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)				
					Approximate % of total Shares in issue immediately following the completion of Capitalization Issue and Global Offering	
		Number of Offer Shares (rounded down to nearest whole board lot of 500 Shares)	Approximate % of total number of Offer Shares Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full	Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full
Cornerstone Investor	Investment Amount (US$ in million)#					
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC Tetrad Ventures Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Fig. 9. (Table B) The merged lines (thin green) created by the final line identification. Vertically split cells highlighted with thick boxes (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

		Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)				
					Approximate % of total Shares in issue immediately following the completion of Capitalization Issue and Global Offering	
		Number of Offer Shares (rounded down to nearest whole board lot of 500 Shares)	Approximate % of total number of Offer Shares Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full	Assuming the Over-allotment Option is not exercised	Assuming the Over-allotment Option is exercised in full
Cornerstone Investor	Investment Amount (US$ in million)#					
Indus Funds	15	10,603,500	5.69%	4.95%	1.08%	1.05%
GIC Tetrad Ventures Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Fig. 10. (Table B) The table after the CNN merged problematic cells. Rows kept are separated for accurate OCR.

4.4.1. CNN construction

The two cells that have been sent into the CNN are merged and rotated so that the potential improper line separating these two be always a vertical line in the center of the merged image. The merged image, as the CNN sees it, always has a dimension of 200×100.

The CNN has three branches that are concatenated into a dense network. The first branch has three 2D convolution layers with 16 filters with a size of 3×3, each layer is directly followed by a max pooling layer of size 2×2. This branch's main purpose is to give low fidelity

	Based on the Offer Price of HK$11.10 (being the Minimum Offer Price)	< EXTEND	< EXTEND	< EXTEND	< EXTEND	
					Approximate % of total Shares	< EXTEND
					in issue immediately following	< EXTEND
		Number of	Approximate % of total	< EXTEND	the completion of Capitalization	< EXTEND
		Offer Shares	number of Offer Shares	< EXTEND	Issue and Global Offering	< EXTEND
		(rounded		Assuming the		Assuming the
		down to	Assuming the	Over-allotment	Assuming the	Over-allotment
		nearest whole	Over-allotment	Option is	Over-allotment	Option is
Cornerstone	Investment	board lot of	Option is not	exercised	Option is not	exercised
Investor	Amount	500 Shares)	exercised	in full	exercised	in full
	(US$ in					
	million)*					
Indus Funds	15	10,603,500	5.69%.	4.95%	1.08%	1.05%
GIC						
Tetrad Ventures						
Pte Ltd	30	21,207,500	11.38%	9.89%	2.16%	2.10%
GIC Private						
Limited	10	7,069,000	3.79%	3.30%	0.72%	0.70%

Fig. 11. (Table B) Tablext output of the table in Fig. 4, in CSV format. Thick boxes (red) show the horizontally merged cells. Cells with multiple rows are kept separated for readability. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

information around the possibly improper line that splits the cells, but it also helps identify which cells contain data.

The second branch begins with an asymmetric average pooling layer of size 1×100. The output of this layer has the dimension of 200×1. This layer is followed by five 1D convolution layers with four filters each with a size of 3. The average pooling gives a good noise resistant indication if any data exists within the particular columns. This information helps identify the existence of data within one or both of the cells, but also helps confirm the presence of an improper split of the two cells.

The third and final branch does not take in the full merged image like the first two, instead it takes in the centermost 20×100 pixels. The exclusive purpose of this branch is to identify an improper splitting of the cells. Intuitively, the pixels where the cells are merged and the surrounding pixels contain the most crucial information regarding whether the cells should be merged. More complex neural network layers are used in this branch due to the importance of the information and the limited amount of inputs. Five convolutions layers, each with 64 filters of size 3×3, are stacked upon one another.

All of these branches are individually passed through a dropout layer with a 20% dropout rate before they are flattened and concatenated. The concatenated data is sent into a dense layer of size 256, which is followed by another dropout layer, with a dropout rate of 50% and another dense layer of equal size. This second dense layer is then connected to the final output which contains 3 nodes. The first two nodes signify if the two cells contain data and the third node is high when two cells should be merged. The first dropout layers help guide the network away from relying purely on a single branch. Additionally, all of the dropout layers help avoid overfitting that would otherwise be present due to the limited amount of training data.

4.4.2. CNN training

To train this CNN, a dataset that includes over 1000 tables from various domains is procured and manually annotated. The ground truth of every table's location, along with its respective cells' locations, is stored within Extensible Markup Language (XML) files. To translate this dataset into useful data for training purposes, Tablext's line identification techniques are used to split up the cells. All possible pairs of adjacent cells are then merged and these merged cells are used as the training data.

To create correct labels for training, the cells' locations are used. If a cell that contains data, according to the XML data, exists on either half of the merged cell, the respective half is considered to have data. When the XML data signifies that a single cell covers some part of both halves of the merged cell, the label for concatenation is high.

4.4.3. Post CNN concatenation

The outputs of the CNN fill in matrices that record which cells should be merged and which cells contain data. Multiple cells in both the vertical and horizontal directions can be combined, if necessary, into a single large merged cell. Fig. 10 is a debug image that shows this result. Cells with multiple independent horizontal lines are not combined at this stage, so that the OCR is fed a single line of data for having a higher accuracy. These horizontal lines can be merged later to allow for a more concise table, or left separated to better represent a table's proportions. With the cells properly defined, the data is almost ready for OCR. However, the pixel resolution used by the table extractor, 800 pixels, is generally not sufficient to obtain accurate results from OCR. To address this problem, the lines defining the cell boundaries are scaled and used to cut cells into the original image.

4.5. OCR

All the non-empty, individual cells are sent to Tesseract, an open-source OCR tool. The returned text is then stored in a Column Separated Values (CSV) format with the location provided by Tablext. The output is shown in Fig. 11. All the data within an extended cell is placed within a single cell in the CSV. To allow for accurate data interpretation, all of the merged cells point back to that original data by using the keyword "EXTEND" and a directional arrow. These cells are emphasized with boxes in Fig. 11. For clarity, cells that take up multiple horizontal lines are not merged in the CSV.

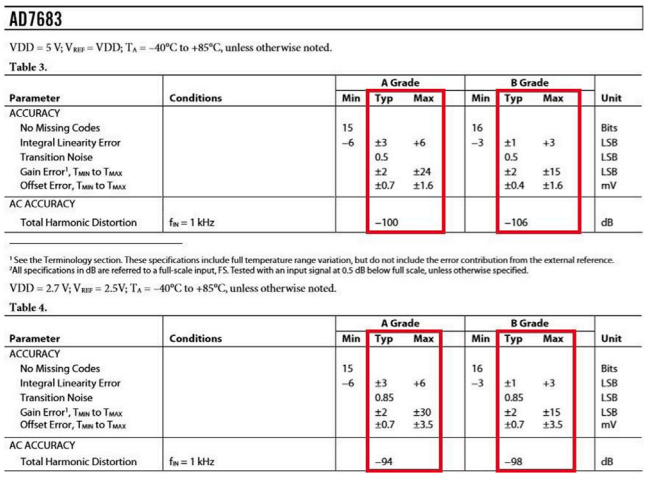
5. Evaluation

Unlike Tablext, most state-of-the art table extraction papers do not have their code available for download. However, both DeepDeSRT [26] and TableNet [27] have tested their design on the ICDAR 2013 table competition dataset [28]. To compare Tablext quantitatively to these works, Tablext is also tested against this dataset. Additionally, Tablext's extraction capability is compared against the popular, open-source, PDF extractor Tabula [29] on a manually procured diverse dataset with over 100 tables. Although table identification is not the focus of this paper, the proposed method produces high quality results, while only using a small fraction of the total runtime.

5.1. Table identification results

The table identifier is tested on a dataset with over 400 tables from various domains. This dataset includes many abstract tables without any lines that are much harder to identify than most tables.

Let AP be the area predicted in an image and AL be the true occupied area. Precision and recall, for an image, can be calculated by precision $= \frac{AP \cap AL}{AP}$ and recall $= \frac{AP \cap AL}{AL}$. The results of the custom CNN, the YOLO model, and a combination of the custom CNN and



AD7683

VDD = 5 V; V_{REF} = VDD; T_A = −40°C to +85°C, unless otherwise noted.

Table 3.

Parameter	Conditions	A Grade Min	A Grade Typ	A Grade Max	B Grade Min	B Grade Typ	B Grade Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		−6	±3	+6	−3	±1	+3	LSB
Transition Noise			0.5			0.5		LSB
Gain Error[1], T_{MIN} to T_{MAX}			±2	±24		±2	±15	LSB
Offset Error, T_{MIN} to T_{MAX}			±0.7	±1.6		±0.4	±1.6	mV
AC ACCURACY								
Total Harmonic Distortion	f_{IN} = 1 kHz		−100			−106		dB

[1] See the Terminology section. These specifications include full temperature range variation, but do not include the error contribution from the external reference.
[2] All specifications in dB are referred to a full-scale input, FS. Tested with an input signal at 0.5 dB below full scale, unless otherwise specified.

VDD = 2.7 V; V_{REF} = 2.5V; T_A = −40°C to +85°C, unless otherwise noted.

Table 4.

Parameter	Conditions	A Grade Min	A Grade Typ	A Grade Max	B Grade Min	B Grade Typ	B Grade Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		−6	±3	+6	−3	±1	+3	LSB
Transition Noise			0.85			0.85		LSB
Gain Error[1], T_{MIN} to T_{MAX}			±2	±30		±2	±15	LSB
Offset Error, T_{MIN} to T_{MAX}			±0.7	±3.5		±0.7	±3.5	mV
AC ACCURACY								
Total Harmonic Distortion	f_{IN} = 1 kHz		−94			−98		dB

Fig. 12. (Table D) A text-based PDF page in our testbench with potentially problematic columns highlighted with boxes (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 2
Table identification YOLO comparison.

	Recall	Precision	F1-Score
Custom CNN only	0.5093	0.3802	0.4354
YOLO only	0.8654	0.8989	0.8818
Custom CNN + YOLO	0.8716	0.8663	0.8689

YOLO model have been summarized in Table 2. After averaging the precision and recall across the dataset, with each image weighted equally, final results of 86% and 87% are achieved for precision and recall respectively for the combined model.

Although the YOLO only model achieves higher precision, the higher recall of the combination of the custom CNN and YOLO is preferable since it is better to over propose regions rather than to miss any tables. This is because regions that are not tables can be discarded later in the extraction process while completely missed tables cannot be recovered.

5.2. Table extraction result comparison

5.2.1. ICDAR comparison

Both DeepDeSRT and TableNet use 34 randomly selected images from the ICDAR dataset for their testing set. Then, they calculate precision and recall in the specific way mandated by ICDAR. For every cell within a table, "adjacency relations" defined by ICDAR, are found with its nearest horizontal and vertical neighbors. In order to get precision and recall, the adjacency relations are compared with the ground truth. Precision and recall are computed for each document then the average is taken across all the documents.

To compare Tablext to the other papers, the same method to calculate precision and recall is used. It should be noted that both DeepDeSRT and TableNet fine tune their networks by training on the remaining ICDAR dataset. In order to prove its ability to extract data from tables in general formats, Tablext does not use any ICDAR data to train with. The results comparing the different methods can be found in Table 3. Tablext has both the highest precision and overall F1-Score out of the methods.

5.2.2. Diverse dataset comparison

Without the source code for DeepDeSRT or TableNet, Tablext cannot compare to them qualitatively or quantitatively on a large diverse dataset. Therefore, a comparison is made with the popular PDF extractor Tabula. Tabula can only extract non-scanned, text-based PDFs.

Table 3
ICDAR data extraction results comparison. Both DeepDeSRT and TableNet are trained on a subset of ICDAR dataset while Tablext does not use any ICDAR data to train with which proves Tablext's ability to extract data from tables in general formats.

Work	Recall	Precision	F1-Score
Tablext	0.9091	0.9221	0.9156
DeepDeSRT	0.8736	0.9593	0.9144
TableNet	0.9001	0.9307	0.9151

Table 4
Diverse data extraction results comparison.

Work	Recall	Precision	F1-Score
Tablext	0.9192	0.9437	0.9313
Tabula	0.7110	0.734	0.7223

Table 3.

Parameter	Conditions	Min	A Grade Typ	A Grade < EXTEND Max	Min	B Grade Typ	B Grade < EXTEND Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		−6	3	6	-3	1	3	LSB
Transition Noise			0.5			0.5		LSB
Gain Error', Twin to Tmax			2	24		2	15	LSB
Offset Error, Twin to Tmax			0.7	1.6		0.4	1.6	mV
AC ACCURACY								
Total Harmonic Distortion	fin = 1 kHz		−100			−106		dB

Table 4.

Parameter	Conditions	Min	A Grade Typ	A Grade < EXTEND Max	Min	B Grade Typ	B Grade < EXTEND Max	Unit
ACCURACY								
No Missing Codes		15			16			Bits
Integral Linearity Error		−6	3	6	-3	1	3	LSB
Transition Noise			0.85			0.85		LSB
Gain Error', Twin to Tmax			2	30		2	15	LSB
Offset Error, Twin to Tmax			0.7	3.5		0.7	3.5	mV
AC ACCURACY								
Total Harmonic Distortion	fin = 1 kHz		-94			−98		dB

Fig. 13. (Table D) Tablext's final output in CSV format. Boxes (red) are showing the correct handling of problematic cells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Parameter	Conditions	Min	A Grade Typ Max	Min	B Grade Typ Max	Unit
ACCURACY						
No Missing Codes		15		16		Bits
Integral Linearity Error		?6	±3 +6	?3	±1 +3	LSB
Transition Noise			0.5		0.5	LSB
Gain Error1, TMIN to TMAX			±2 ±24		±2 ±15	LSB
Offset Error1, TMIN to TMAX			±0.7 ±1.6		±0.4 ±1.6	mV
AC ACCURACY						
Total Harmonic Distortion	fIN = 1 kHz		?100		?106	dB
			A Grade		B Grade	
Parameter	Conditions	Min	Typ Max	Min	Typ Max	Unit
ACCURACY						
No Missing Codes		15		16		Bits
Integral Linearity Error		?6	±3 +6	?3	±1 +3	LSB
Transition Noise			0.85		0.85	LSB
Gain Error1, TMIN to TMAX			±2 ±30		±2 ±15	LSB
Offset Error1, TMIN to TMAX			±0.7 ±3.5		±0.7 ±3.5	mV
AC ACCURACY						
Total Harmonic Distortion	fIN = 1 kHz		?94		?98	dB

Fig. 14. (Table D) Tabula's final output in CSV format. Boxes (red) are showing the incorrect handling of problematic cells. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

So a testbench, that exclusively contains text-based PDFs, is created by randomly selecting over 100 tables within PDFs from various domains. Tabula is given the PDF pages with the text and meta-information, while Tablext is given a scanned image of each PDF page. Fig. 12 is an example of one such page within our testbench.

Fig. 12 is extracted by both Tablext and Tabula. With this input, Figs. 13 and 14 respectively show Tablext's and Tabula's final CSV

output. Both have a high accuracy when reproducing the cells' contents, though Tabula is simply reading text provided to it, while Tablext is using OCR. One issue with the Tablext output is that Tesseract appears to not recognize the plus–minus character. Tabula also did not perfectly print the data within the cells, despite reading straight from the PDF. The minus character could not be recognized by Tabula. This problem appears because this PDF is using an abnormal character instead of the standard ASCII symbol.

Several severe structural errors appear in Tabula's output and these errors have been highlighted in Fig. 14. Looking back at the original image in Fig. 12, it becomes clear what caused these errors. The cells "A Grade" and "B Grade" overlap both the columns beginning with "Typ" and "Max". Errors like this are common for conventional text-based extractors [30]. Tablext meanwhile, with its dynamic threshold ratio, easily separates these two columns then merges the split "A Grade" and "B Grade" cells back into a single cell.

The highlighted cells in Fig. 14 emphasize the importance of accurate cell location identification. Even with clever post-processing, it would be impossible to tell which column the merged cells with a single piece of data belong to. For instance, looking at the row beginning with "Transition noise", there is no way to know if the data belongs to the "Typ" or "Max" column.

The results comparing the two methods can be found in Table 4. Tablext's ability to handle complex tables greatly surpasses Tabula's, despite all of the extra meta-information that Tabula has access to.

6. Conclusion

This paper introduces a novel, general approach for table extraction. By utilizing both deep learning and computer vision techniques, the neural network can focus its attention on complex problems, while allowing more conventional methods to handle the simpler tasks. This focus allows Tablext to beat competing state-of-the-art neural-network-based table extraction methods as well as a popular open-source tool that requires table meta-data.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Morteza Fayazi reports financial support was provided by Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under agreement number FA8650-18-2-7844.

Acknowledgments

This material is based on research sponsored by Air Force Research Laboratory (AFRL), United States and Defense Advanced Research Projects Agency (DARPA), United States under agreement number FA8650-18-2-7844. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

References

[1] Tengli A, Yang Y, Ma NL. Learning table extraction from examples. In: Proceedings of the 20th international conference on computational linguistics. Association for Computational Linguistics; 2004, p. 987.
[2] Oro E, Ruffolo M. Trex: An approach for recognizing and extracting tables from pdf documents. In: 2009 10th international conference on document analysis and recognition. IEEE; 2009, p. 906–10.
[3] Pinto D, McCallum A, Wei X, Croft WB. Table extraction using conditional random fields. In: Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval; 2003, p. 235–42.
[4] Belval E. Pdf2image: Open source scientific tools for python. 2019, https://github.com/Belval/pdf2image, last accessed 2019-07-28.
[5] Grabzit: Open source scientific tools for python. 2019, https://github.com/GrabzIt/grabzit, last accessed 2019-07-29.
[6] LeCun Y, Haffner P, Bottou L, Bengio Y. Object recognition with gradient-based learning. In: Shape, contour and grouping in computer vision. Springer; 1999, p. 319–45.
[7] Fayazi M, Colter Z, Afshari E, Dreslinski R. Applications of artificial intelligence on the modeling and optimization for analog and mixed-signal circuits: A review. IEEE Trans Circuits Syst I Regul Pap 2021.
[8] Redmon J, Farhadi A. YOLOv3: An incremental improvement. 2018, ArXiv.
[9] Smith R, et al. Tesseract OCR: Open source scientific tools for python. 2019, https://github.com/UB-Mannheim/tesseract, last accessed 2019-07-29.
[10] Cesarini F, Marinai S, Sarti L, Soda G. Trainable table location in document images. In: Object recognition supported by user interaction for service robots, Vol. 3. IEEE; 2002, p. 236–40.
[11] Fan M, Kim DS. Table region detection on large-scale PDF files without labeled data. 2015, CoRR, abs/1506.08891.
[12] Gatos B, Danatsas D, Pratikakis I, Perantonis SJ. Automatic table detection in document images. In: International conference on pattern recognition and image analysis. Springer; 2005, p. 609–18.
[13] Gilani A, Qasim SR, Malik I, Shafait F. Table detection using deep learning. In: 2017 14th IAPR international conference on document analysis and recognition (ICDAR), Vol. 1. IEEE; 2017, p. 771–6.
[14] Hao L, Gao L, Yi X, Tang Z. A table detection method for pdf documents based on convolutional neural networks. In: 2016 12th IAPR workshop on document analysis systems (DAS). IEEE; 2016, p. 287–92.
[15] Kasar T, Barlas P, Adam S, Chatelain C, Paquet T. Learning to detect tables in scanned document images using line information. In: 2013 12th international conference on document analysis and recognition. IEEE; 2013, p. 1185–9.
[16] Shafait F, Smith R. Table detection in heterogeneous documents. In: Proceedings of the 9th IAPR international workshop on document analysis systems; 2010, p. 65–72.
[17] Perez-Arriaga MO, Estrada T, Abad-Mota S. TAO: system for table detection and extraction from PDF documents. In: The twenty-ninth international flairs conference. 2016.
[18] Hassan T, Baumgartner R. Table recognition and understanding from pdf files. In: Ninth international conference on document analysis and recognition (ICDAR 2007), Vol. 2. IEEE; 2007, p. 1143–7.
[19] Yildiz B, Kaiser K, Miksch S. Pdf2table: A method to extract table information from pdf files. In: IICAI. 2005, p. 1773–85.
[20] Liu Y, Bai K, Mitra P, Giles C. Searching for tables in digital documents. In: Ninth international conference on document analysis and recognition (ICDAR 2007), Vol. 2. IEEE; 2007, p. 934–8.
[21] Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Comput Surv 1999;31(3):264–323.
[22] Nishida K, Sadamitsu K, Higashinaka R, Matsuo Y. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In: Thirty-first AAAI conference on artificial intelligence. 2017.
[23] Koci E, Thiele M, Lehner W, Romero O. Table recognition in spreadsheets via a graph representation. In: 2018 13th IAPR international workshop on document analysis systems (DAS). 2018, p. 139–44.
[24] Puha A, Rincioq O, Posea V. Enhancing open data knowledge by extracting tabular data from text images. In: DATA. 2018.
[25] Qasim SR, Mahmood H, Shafait F. Rethinking table recognition using graph neural networks. In: 2019 international conference on document analysis and recognition (ICDAR). IEEE; 2019, p. 142–7.
[26] Schreiber S, Agne S, Wolf I, Dengel A, Ahmed S. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In: 2017 14th IAPR international conference on document analysis and recognition (ICDAR), Vol. 1. IEEE; 2017, p. 1162–7.
[27] Paliwal SS, Vishwanath D, Rahul R, Sharma M, Vig L. TableNet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images. In: 2019 international conference on document analysis and recognition (ICDAR). IEEE; 2019, p. 128–33.
[28] Göbel M, Hassan T, Oro E, Orsi G. Icdar 2013 table competition. In: 2013 12th international conference on document analysis and recognition. 2013, p. 1449–53.
[29] Aristarán M, Mike T, Merrill JB, Das J, Frackman D, Swicegood T. Tabula: Open source scientific tools for python. 2019, https://github.com/tabulapdf/tabula, last accessed 2019-07-29.
[30] Dreslinski R, et al. Fully-autonomous SoC synthesis using customizable cell-based synthesizable analog circuits. Tech. rep., University of Michigan Ann Arbor United States; 2019.

Zach Colter was born in Michigan, United States in 1997. He recieved both the B.Sc. degree, in computer engineering, and M.S. degree, in electrical computer engineering, from the University of Michigan. The degrees were acquired in 2019 and 2020 respectively.

Morteza Fayazi was born in Tehran, Iran, in 1994. He received the B.Sc. major degree in electrical engineering and minor degree in computer science from Sharif University of

Technology (SUT), Tehran, Iran, in 2017. He Also received the M.S. degree in electrical engineering and computer science in 2020 from University of Michigan, Ann Arbor, MI. He is currently working toward the Ph.D. degree at University of Michigan, Ann Arbor, MI. His research interests include circuit design automation, machine learning, software development and computer architecture.

Zineb Benameur-El Youbi received the B.Sc. and M.S degree in Computer Networks and Multimedia Communication from the University of Grenoble, France, in 2009. She Also received the M.S. degree in Computer Science and Engineering in 2020 from University of Michigan, Ann Arbor, MI. Her research interests include reliable software development and Machine Learning.

Serafina Kamp was born in Michigan, US. She received her BSE in Computer Science Engineering from the University of Michigan in 2022. She is currently working toward her Ph.D. degree at the University of Michigan in Computer Science Engineering. Her research interests include machine learning, algorithm design, fairness of machine learning algorithms, and statistical decision making.

Shuyan Yu was born in Shenzhen, China, in 1999. She received the B.Sc. major degree in computer science and in mathematics from University of Michigan, Ann Arbor, MI, in 2020. She is currently working toward the M.S. degree in computer and information sciences at University of Pennsylvania, PA. Her research interests include data mining, natural language processing, information extraction and machine learning.

Ronald Dreslinski received the BSE degrees both in electrical engineering and computer engineering, and the MSE and PhD degrees in computer science and engineering from the University of Michigan, Ann Arbor, Michigan. He is currently an assistant professor of computer science and engineering with the University of Michigan. His research focuses on architectures that enable emerging low-power circuit techniques.