Neural Formatting for Spreadsheet Tables

Haoyu Dong, Jinyu Wang, Zhouyu Fu*, Shi Han, Dongmei Zhang
Microsoft Research
{hadong,wang.jinyu,shihan,dongmeiz}@microsoft.com,fu.zhouyu@gmail.com

ABSTRACT

Spreadsheets are popular and widely used for data presentation and management, where users create tables in various structures to organize and present data. Table formatting is an important yet tedious task for better exhibiting table structures and data relationships. However, without the aid of intelligent tools, manual formatting remains a tedious and time-consuming task. In this paper, we propose CellGAN, a neural formatting model for learning and recommending formats of spreadsheet tables. Based on a novel conditional generative adversarial network (cGAN) architecture, CellGAN learns table formatting from real-world spreadsheet tables in a self-supervised fashion without requiring human labeling. In CellGAN we devise two mechanisms, row/column-wise pooling and local refinement network, to address challenges from the spreadsheet domain. We evaluate the effectiveness of CellGAN against real-world datasets using both quantitative metrics and human perception studies. The results indicate remarkable performance gains over rule-based methods, graphical models or direct application of the state-of-the-art cGANs used in visual synthesis tasks. Neural Formatting is the first step towards auto-formatting for spreadsheet tables with promising results.

CCS CONCEPTS

• **Information systems** → **Semi-structured data**; • **Computing methodologies** → **Neural networks**.

KEYWORDS

document intelligence; automatic formatting; deep learning

ACM Reference Format:
Haoyu Dong, Jinyu Wang, Zhouyu Fu*, Shi Han, Dongmei Zhang. 2020. Neural Formatting for Spreadsheet Tables. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411943

1 INTRODUCTION

Spreadsheets are popular and widely used for data presentation and management, with tables playing a central role. Different from database tables, spreadsheet tables by themselves provide effective visualization for data presentation. Since tables typically have various structures and layouts on the cell grid, table formats are created to intuitively reflect data correspondence for easy look-up, or serve side-by-side comparison for higher-order knowledge exhibition. From this visual perspective, table formatting such as border, alignment, font, etc, significantly helps with table layout shaping or structure scoping. For example, the rich formats on the table in Figure 1(b) not only helps to shape the complex hierarchies in the top and left headers ("A1:G7" and "A8:A17"), but also helps to scope the data groups in "B8:G17". On the contrary, it is not intuitive to understand the raw data in Figure 1(a). Hence, table formatting is an important task in the spreadsheet domain.

Manual formatting of spreadsheet tables is tedious and time-consuming, especially for professional and complex tables in finance and government domains. For example, at least 79 mouse clicks with a correct order are required for an Excel professional to format Figure 1(a) to Figure 1(b) using Excel. The objective of this study is to explore intelligence towards automatic formatting of spreadsheet tables. Unfortunately, there is hardly any related study or existing spreadsheet tool available to address this task. Major challenges are concluded as follows. (1) Table structures and data layouts are often complex. Due to the flexibility of commodity spreadsheet tools and the diversity of human artifacts when organizing data, users create tables with various structures. It is challenging to

	A	B	C	D	E	F	G
1	CHARACTERISTIC	Employed wage and salary workers			Non employed usual weekly earnings		
2							
3		Total	Percent		Total	Number	
4							
5				Not			
6			Union members	union members		Represented by union	Not represented by union
7							
8	Total	129378	0.124	0.876	8960	4904	5056
9	Age						
10	16 to 34 years old	47981	0.157	0.843	3421	1342	2079
11	35 to 54 years old	59495	0.294	0.706	3427	1867	1560
12	55 years and over	21901	0.256	0.744	3112	1695	1417
13	Industry						
14	Private sector	108073	0.076	0.924	7429	3238	4191
15	Agriculture	1057	NA	NA	NA	NA	NA
16	Nonagriculture	107016	NA	NA	NA	NA	NA
17	Public sector	21305	0.0368	0.9632	1531	1666	865

(a) A table with default formats

	A	B	C	D	E	F	G
1		Employed wage and salary workers			Not employed usual weekly earnings		
2							
3			Percent			Number	
4	CHARACTERISTIC	Total			Total		
5							
6			Union members	Not union members		Represented by union	Not represented by union
7							
8	**Total**	**129,378**	**12.4%**	**87.6%**	**9,960**	**4,904**	**5,056**
9	**Age**						
10	16 to 34 years old	**47,981**	15.7%	84.3%	**3,421**	1,342	2,079
11	35 to 54 years old	**59,495**	29.4%	70.6%	**3,427**	1,867	1,560
12	55 years and over	**21,901**	25.6%	74.4%	**3,112**	1,695	1,417
13	**Industry**						
14	Private sector	**108,073**	7.6%	92.4%	**7,429**	3,238	4,191
15	Agriculture	**1,057**	NA	NA	**NA**	NA	NA
16	Nonagriculture	**107,016**	NA	NA	**NA**	NA	NA
17	Public sector	**21,305**	3.7%	96.3%	**2,531**	1,666	865

(b) A table with human-crafted formats

Figure 1: Comparison of an example spreadsheet table with deault formats and human-crafted formats.

* Work done while the author was full-time employee at Microsoft Research.

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411943

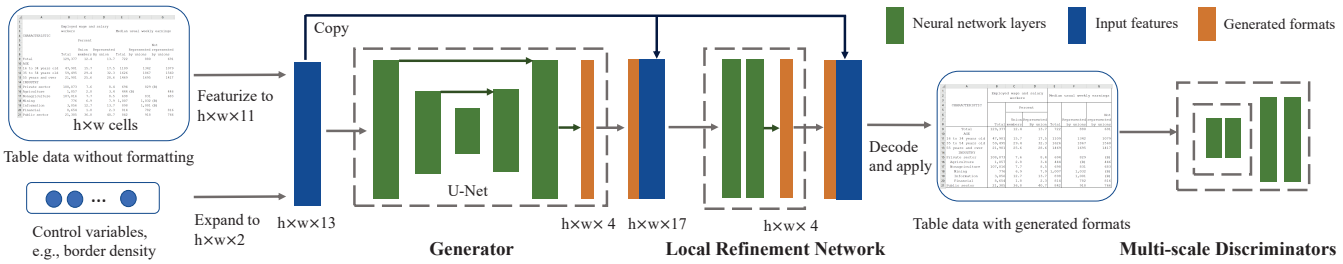


Figure 2: Architecture of CellGAN to map table data→formats. The generator learns to generate formats based on data, and the local refinement net is appended to enhance formats with better local consistency. The discriminators learn to distinguish generated formats from real formats.

systematically describe and automatically extract various table structures, e.g., the hierarchical headers and data groups. Several works [8, 10, 20] aim to analyze table structures, but only target web or PDF tables with simple table structures, such as single header row/column that can match simple templates, and are not applicable to complex table structure extraction. [3] focuses on spreadsheet table structure extraction in a user interactive way. (2) To the best of our knowledge, there is no existing metric to evaluate the quality of table formatting, while metrics are neccessary for guiding both rule-based methods and data-driven methods. (3) Since different cultures and domains may have different preferences, there exists various formatting styles in real-world tables (the same table can be formatted in different border densities, alignment preferences, color themes, etc.), and makes it challenging to summarize various implicit formatting styles.

On the other hand, large volumes of expert-made formatted spreadsheets are available on the web and can be obtained with trivial efforts. **This motivates an end-to-end approach to learn automatic formatting directly from a large amount of formatted sheets in a self-supervised manner without explicitly modeling table structures**, since structure information is embedded in the underlying formats and implicitly captured by an end-to-end model. (1) A sheet can be viewed as a two-dimensional array of cells, and formats are well structured on sheets. This motivates us to use Convolutional Neural Networks (CNNs) [16] or graphical models [15] to capture spatial correlations between cells. (2) There lacks a loss function or an objective to score the overall quality of formats, and designing effective losses needs a lot of expert efforts. Since a table can have multiple reasonable formatting styles in real data, if we take a naive approach by asking a CNN to minimize the Euclidean distance between predicted and real-world formats, it may tend to produce "blurry" formats since the Euclidean distance is minimized by "averaging" all reasonable formatting styles in real data. Fortunately, Generative Adversarial Nets (GANs) [12] are proposed to learn an overall loss by adversarially training a network that "makes the output indistinguishable from reality". (3) While standard GANs aim to fit the natural data distribution, conditional GANs [19] are proposed to learn the probabilistic mapping from one domain to another domain in conditional settings [13, 23], and have been successful in various computer vision tasks including image style transfer [11] and multi-domain image translation [4]. This motivates us to learn a **one-to-many mapping from table data to table formatting** with a cGAN architecture.

Nevertheless, the cross-domain application is never straightforward due to domain-specific characteristics of spreadsheet data and the auto-formatting task. Directly applying cGAN-based models to spreadsheet data without incorporating domain-specific and task-specific cues would produce results with poor quality. In this study, we propose CellGAN, a CNN-based generative model with several key enhancements customized for table auto-formatting. Major contributions in this paper include:

- We provide the first formulation of the table auto-formatting problem in this paper. In addition, we propose ways to evaluate the generated table formatting including subjective evaluation with user perceptual studies and objective evaluation with quantitative methods.
- We propose a novel method, CellGAN, to learn table format generation with convolutional networks in a generative adversarial way.[1] CellGAN achieves superior performance in table format generation over all comparison methods according to both quantitative metrics and human studies.
- Based on the characteristics of spreadsheet data and formats, we devise a novel row/column-wise pooling layer and a local refinement network. Our ablation studies show that they can considerably improve the quality of generated formats.
- We also explore user-controllable auto-formatting such as customizing borderlines with varied densities, demonstrating the model's ability to navigate intrinsic formatting styles in the space of acceptable formats.

2 METHOD

2.1 Problem Formulation

Table formats in real-world spreadsheets depend on both implicit and explicit factors. While explicit factors such as cell values and types provide direct cues for table formatting (e.g., a border should be placed to separate data and header regions), implicit factors such as the culture and domain are also useful in capturing the variations of formatting styles (e.g., the best practices for border placement and alignment adjustment vary with different professions). This

[1] Available at https://github.com/haoyudong/NeuralFormatting

motivates a data-driven approach to learn table formats based on large volumes of professional spreadsheet tables crawled from diverse backgrounds. Moreover, with increasing amounts of training data, we can also suppress the undesirable effect of sporadic user factors over model learning, yielding models that concentrate on common and domain practices rather than individual preferences.

To this end, we formulate auto-formatting as a task to map a matrix of data in a table to their formats. Among different format types, **borders** are one of the most effective and widely used formats to visualize table structures, especially the hierarchical headers and data groups, and **alignments** are also commonly used to provide neat and clean data presentation. Hence in this paper, we select border and alignment as two representative formats to demonstrate the validity of our method. However, the proposed method is quite general and can be directly applied to learn other format types.

2.2 Cell Featurization

To enable end-to-end format learning, we devise a featurization scheme to represent input cells and their output formats. Each input cell is encoded by an 11-dimensional vector to capture cell value, formula, etc, whereas the output formats are encoded by a 4-dimensional vector, each representing the encoding for a specific format type along a specified direction. Table 1 provides more details on the encoding schemes. Suppose h and w are the height and width of the table, an $h\times w\times 11$ input tensor $\boldsymbol{d}$ and an $h\times w\times 4$ output tensor $\boldsymbol{f}$ can be extracted based on our proposed featurization.

2.3 Network Architecture

In general, a cGAN architecture consists of a generator G and a discriminator D. In this formatting task, the generator G is trained to generate output formats $\boldsymbol{f}$ conditioned on the input features $\boldsymbol{d}$ such that an adversarially trained discriminator cannot distinguish generated formats from real-world formats. The discriminator D is trained to classify whether the data and formats pair $(\boldsymbol{d}, \boldsymbol{f})$ is "real" or "fake". The objective function is then given by:

$$\begin{aligned}\mathcal{L}_{\text{cGAN}}(G, D) =& \mathbb{E}_{(\boldsymbol{d},\boldsymbol{f})}[\log D(\boldsymbol{d}, \boldsymbol{f})]+ \\ & \mathbb{E}_{\boldsymbol{d}}[\log(1 - D(\boldsymbol{d}, G(\boldsymbol{d})))],\end{aligned} \tag{1}$$

where G and D aim to model the conditional distribution of formats via the following minimax game:

$$\min_G \max_D \mathcal{L}_{\text{cGAN}}(G, D) \tag{2}$$

The architecture of our method is shown in Figure 2. We use convolutional neural networks to build the backbone of G and D to capture spatial correlations in the cell matrix. To effectively use cell-level spreadsheet features to generate high-quality formats, the "u-net" architecture [22] is adopted in the generator with shortcut connections between corresponding down-sampling and up-sampling layers. Multi-scale Markovian discriminators [7] are used during training in an effort to better capture information in widely varying scales of receptive fields in a coarse-to-fine fashion. We use 3 discriminators to differentiate real and generated formats in 3 different scales. The learning problem in Eq. (2) then becomes:

$$\min_G \max_{D_1, D_2, D_3} \sum_{k=1,2,3} \mathcal{L}_{\text{cGAN}}(G, D_k) \tag{3}$$

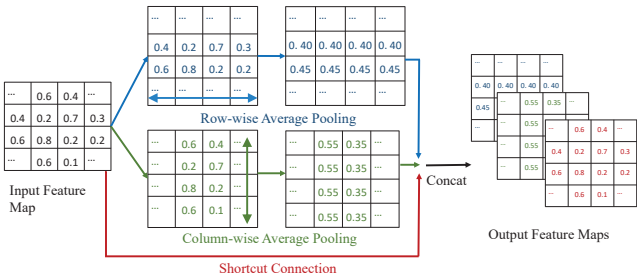


Figure 3: The architecture of row/column-wise average pooling. The input feature map is concatenated with their row and column average.

2.4 Row/column-wise Average Pooling

In a spreadsheet table, data are usually arranged row-wise or column-wise for easy look-up and query. Consequently, table formats also exhibit such row-level and column-level patterns. For example, in Figure 1(b), all the horizontal alignments of numbers in column G are right alignments. To capture these row-level and column-level patterns in a more effective way, we propose row/column-wise average pooling as shown in Figure 3.

Suppose $E \in \mathbb{R}^{h\times w}$ is the input feature map to the row/column-wise average pooling layer. For row-wise pooling, we first average the feature vector for each row, then fill each row of the pooling result $H \in \mathbb{R}^{h\times w}$ with the corresponding row-average value. And the same operations can be applied to column-wise pooling but in a different orientation to produce the pooling result $V \in \mathbb{R}^{h\times w}$. Precisely, the row/column-wise pooling results are defined by:

$$H = \frac{1}{w}(E \cdot \mathbf{1}^{w\times w}) \tag{4}$$

$$V = \frac{1}{h}(\mathbf{1}^{h\times h} \cdot E) \tag{5}$$

where $\mathbf{1}^{w\times w}$ and $\mathbf{1}^{h\times h}$ are $w\times w$ and $h\times h$ matrices of ones respectively. Finally, the output is produced by concatenating the input E with the pooling results H and V as shown in Figure 3.

By using row/column-wise pooling, the subsequent convolutional layers can directly integrate both row-level and column-level features, regardless of the kernel sizes used for convolution. Moreover, the shortcut connection between input and output also help preserve the original features. In our method, row/column-wise pooling is applied to all down-sampling layers of u-net as well as the first layer of the local refinement network as shown in Figure 2.

2.5 Local Refinement Network

Viewers are sensitive to the local patterns in table formats, especially in important table regions like hierarchical headers. Even a single cell with an inconsistent format from neighboring cells is obtrusive and can destroy the eventual effect of table formatting, such as a cell with an inconsistent border in the header region or a cell with a different alignment in the middle of a column. In order to prevent undesirable artifacts, we need to enforce stronger local consistency of the generated formats in model training. However,

Table 1: Featurization of table data and formats.

Name	Description	Value
Input features		
Log Length	Log length of the string. Set to 0 for blank string.	Float
Alpha Prop	Proportion of the letters in the string.	[0, 1]
Number Prop	Proportion of digits in the string	[0, 1]
Space Log Length	Log length of spaces at the starting of string.	Float
Merged With Top	If the cell is merged with top neighbor.	{0, 1}
Merged With Left	If the cell is merged with left neighbor.	{0, 1}
Merged With Bottom	If the cell is merged with bottom neighbor.	{0, 1}
Merged With Right	If the cell is merged with right neighbor.	{0, 1}
Is Number Type	If the cell value is displayed as number.	{0, 1}
Is Date Type	If the cell value is displayed as date.	{0, 1}
Has Formula	If the cell contains a formula.	{0, 1}
Output formats		
Horizontal Border	If horizontal border of the cell exists.	{0, 1}
Vertical Border	If vertical border of the cell exists.	{0, 1}
Horizontal Alignment	Horizontal text alignment (left, center, right).	{0, 1, 2}
Vertical Alignment	Vertical text alignment (top, middle, bottom).	{0, 1, 2}

due to the variety of table structures, it is difficult to employ explicit rules with high robustness. To this end, we propose a novel local refinement network customized for the spreadsheet domain to perform a coarse-to-fine refinement on the generated formats and diminish the local outliers.

The local refinement network takes the previously generated formats with the original spreadsheet data as inputs, and it outputs the refined formats over initial ones. The network employs a fully convolutional network [17] as its backbone, and incorporates our proposed row/column-wise pooling. We further employ a new data augmentation strategy that randomly adds some incorrect local formatting patterns to the input, and encourage this network to eliminate them by training with an additional loss term, $\mathcal{L}_{\text{denoising}}$:

$$\mathcal{L}_{\text{denoising}} = \frac{1}{\sum_{t\in T,i,j} \boldsymbol{e}_{t,i,j}}\Big(\sum_{t\in T,i,j} \boldsymbol{e}_{t,i,j}\left|\boldsymbol{f}_{t,i,j} - \boldsymbol{f}^*_{t,i,j}\right|\Big), \quad (6)$$

where $\boldsymbol{e}_{t,i,j}$ indicates if the specific input format of class t in the (i,j)-th cell is changed when adding incorrect local patterns, $\boldsymbol{f}^*$ is the input formats before adding incorrect patterns, $\boldsymbol{f}$ is the output formats, and T is the full set of format types. Moreover, to encourage continuous formats in row and column directions, we incorporate a novel discontinuity loss. Considering different format types may require continuity in different directions, we divide format types into two groups, T_{row} and T_{col}. T_{row} contains format types that require row-wise continuity (*e.g.*, horizontal border, vertical alignment), and T_{col} respects to column-wise continuity. The discontinuity loss $\mathcal{L}_{\text{discontinuity}}$ is then given by:

$$\mathcal{L}_{\text{discontinuity}} = \frac{1}{wh}\Big(\sum_{t\in T_{row},i,j}\left|\boldsymbol{f}_{t,i,j} - \boldsymbol{f}_{t,i,j+s}\right| + \sum_{t\in T_{col},i,j}\left|\boldsymbol{f}_{t,i,j} - \boldsymbol{f}_{t,i+s,j}\right|\Big), \quad (7)$$

where s is the step size that defines the neighborhood relations for the specific orientation, which is set to 1 in our local refinement network. The final objective combines both the adversarial loss and the local refinement loss:

$$\min_G\Big(\max_{D_1,D_2,D_3}\Big(\sum_{k=1,2,3}\mathcal{L}_{\text{cGAN}}(G,D_k)+\alpha\mathcal{L}_{\text{discontinuity}}+\beta\mathcal{L}_{\text{denoising}}\Big)\Big), \quad (8)$$

where α and β controls the importance of the two terms. Here we can also view the local refinement module as an implicit soft regularization for the overall loss $\mathcal{L}_{\text{cGAN}}$ to encourage formats with better local continuity and mitigate local defects.

2.6 Formatting with User-Controllable Attributes

As discussed previously, it is desirable for the auto-formatting model to generate a diversity of acceptable formats for input tables. Current cGANs achieve this by providing a random variable z as an additional input to the generator, allowing multiple outputs to be produced in the same condition. However, the use of random variable z to control diversity is not intuitive or meaningful.

Hence in our work, we try to explore intuitive and meaningful control variables so as to enable flexible control over the output formats by manipulating these variables. Taking border format as an example, we find the border density to be an intuitive variable to control the effect of generated borders for a table. Here the horizontal (vertical) border density is defined as the proportion of the cells with horizontal (vertical) borders in a table and can be easily calculated. In the training phase, we augment the input features $\boldsymbol{d}$ of a table with two new channels by filling the true horizontal and vertical densities to all of its cells and feed the augmented features to the generator. Then the generator $G(\boldsymbol{d})$ in Eq. (1) generates formats conditioned on both input features and border densities. In the testing phase, the trained generator can then be used to recommend diversified border formats over the same table by feeding different target density values to the generator. Note that this methodology is quite general and can also be applied

Table 2: Characteristics comparison between datasets.

Dataset	SAUS	NCES	NSF
Number of tables	1,109	5,237	1,195
Average row count	54	60	45
Average column count	16	20	14
Tables with horizontal borders	100.0%	99.8%	100.0%
Tables with vertical borders	100.0%	62.7%	28.4%

to other format types that can be controlled via a ratio-like variable, such as portion of highlighted cells by font or color.

3 EXPERIMENTS

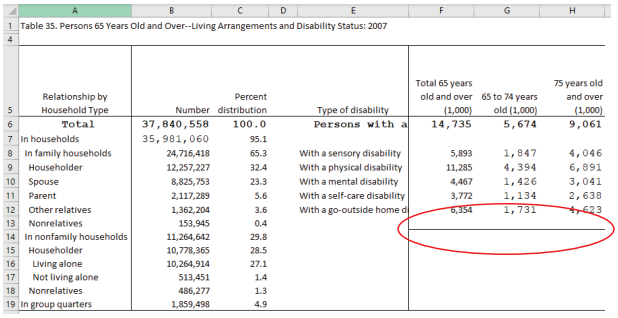
Datasets We conduct evaluations on three web-crawled datasets, namely SAUS[2], NCSE[3], NSF[4], each containing rich amounts of spreadsheets created by professionals in different domains with high formatting quality. For each dataset, we randomly select 80% of tables for training and the remaining 20% for testing. Table 2 shows some statistics. NSF prefers horizontal borders to shape table layouts, while SAUS uses more vertical borders. We use the ClosedXML[5] library to parse Excel spreadsheets and a recent CNN-based approach, TableSense[5], to detect tables in spreadsheets.

Baselines Given the absence of prior work on this task, we first investigate a rule-based method by consolidating effective heuristics. Then we adapt a graphical model to this task. Considering the analogy between an image as a 2D matrix of pixels and a sheet as a 2D matrix of cells, state-of-the-art methods for image generation can also be strong baselines. Moreover, we evaluate two variants of CellGAN for ablation studies. These methods are listed as follows:

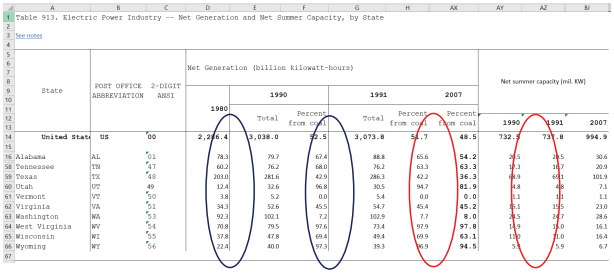
- Rule-based method. First, we detect and parse the hierarchical headers based on merged cells in the top and indent levels on the left. Based on the parsed header trees, we consolidate heuristics to generate table formats, e.g., add borders between the respective regions of two neighboring tree levels.
- Undirected graphical model [15] that learns the joint distribution of $(\boldsymbol{d}, \boldsymbol{f})$ with the node potential to capture features of a single cell and the edge potential to capture relations of neighboring cells in pairwise.
- CRN [2], a supervised image synthesis model using a feed-forward convolutional network.
- Pix2pixHD [24], a state-of-the-art high resolution image generation algorithm based on cGANs.
- CellGAN (w/o LRN). Based on CellGAN (full), the Local Refinement Network (LRN) is removed.
- CellGAN (w/o RCP). Based on CellGAN (full), the Row/Column-wise Pooling (RCP) is removed.

To ensure unbiased comparison, all methods use the same featurization scheme introduced in Section 2.2.

[2]Downloaded the 2010 Statistical Abstract of US from the Census Bureau website.
[3]Crawled from the National Center for Education Statistic website.
[4]Crawled from the National Science Foundation website.
[5]https://github.com/ClosedXML/ClosedXML



(a)



(b)

Figure 4: Bad cases of CellGAN's generation in SAUS.

3.1 Implementation Details

Generator in CellGAN We use `Ca×b-k` to denote a convolutional layer with k a×b filters and stride 1. Similarly, `Da×b-k` denotes a convolutional layer down-sampled by a factor of 2, while `Ua×b-k` denotes a convolutional layer up-sampled by a factor of 2. `RRCP-k` denotes a residual block with k 5×3 filters. The proposed row/column-wise pooling layer is inserted before each residual block and downsampling layer. The generator adopts a u-net architecture that consists of:

- Encoder: C1×1-64, 3×C7×3-64, D8×2-128, D8×3-256, D8×3-512, D8×2-1024, D8×3-1024, D8×2-1024, 6×RRCP-1024.
- Decoder: U8×2-1024, U8×3-1024, U8×2-512, U8×3-256, U8×2-128, U8×2-5

Discriminator in CellGAN The discriminator adopts multi-scale PatchGAN [24] that consists of: C1×1-64, 2×C5×3-64, D7×3-128, D5×3-128, C3×1-512, D7×3-512, C5×3-512, C5×3-1.

Training Details We train the entire model end-to-end using an Adam optimizer [14] by simultaneously optimizing the objectives for borders and alignments. We train our models on 16 NVIDIA Tesla V100 GPUs for around 900,000 iterations.

3.2 Case Study

To facilitate an easier understanding of the subsequent evaluations, we first show two typical cases to help illustrate key concepts intuitively. In Figure 4, Figure 5 and Figure 6, the two cases are shown on the left and right columns, while results obtained by different techniques are placed on different rows for comparison.

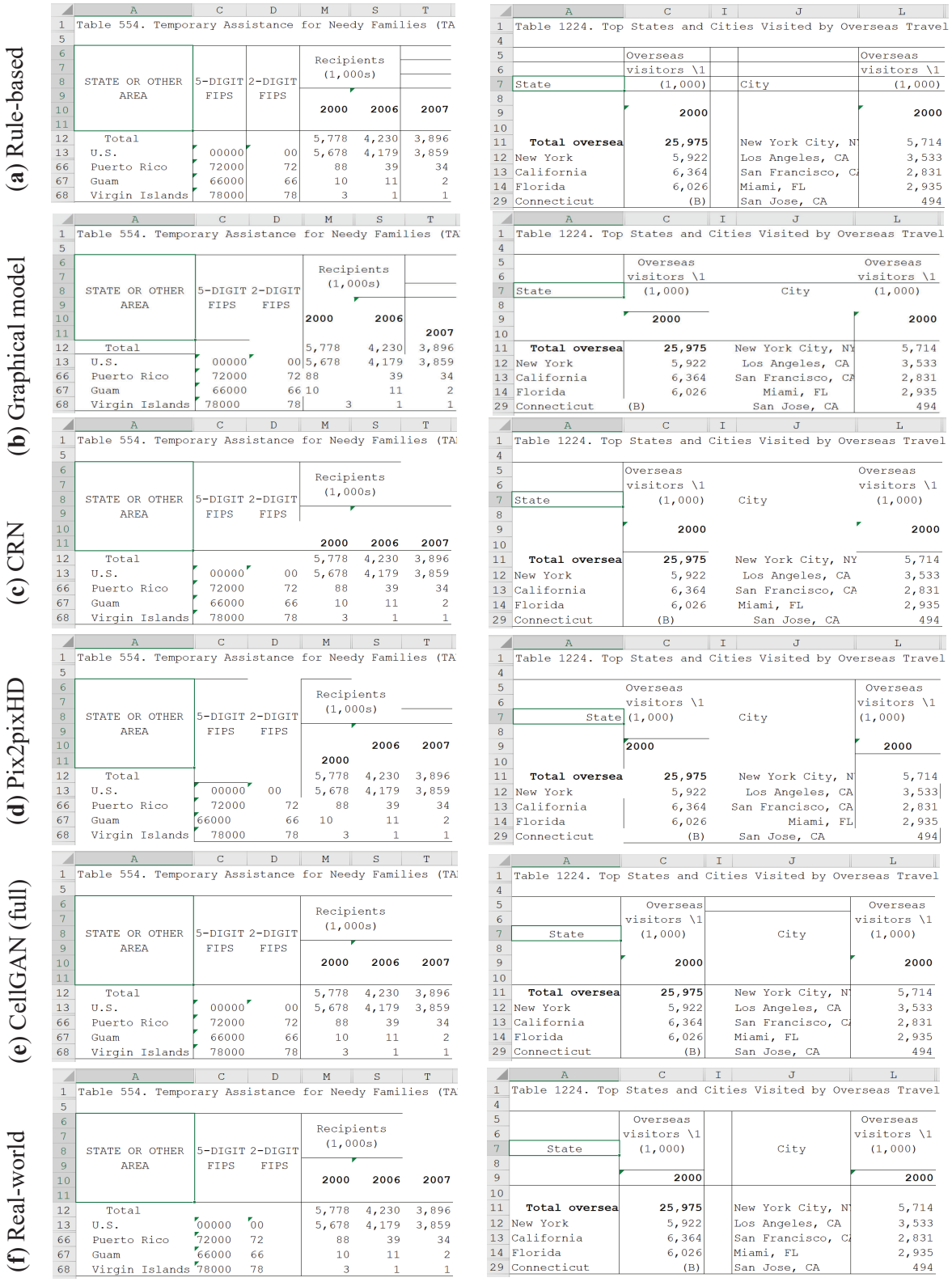


Figure 5: Case study of various format generation approaches. Two test cases in SAUS are shown on the left and right columns, while generated cell borders and alignments by different techniques are placed on different rows for comparison. To display all these results on this constrained space, some rows and columns are hidden.

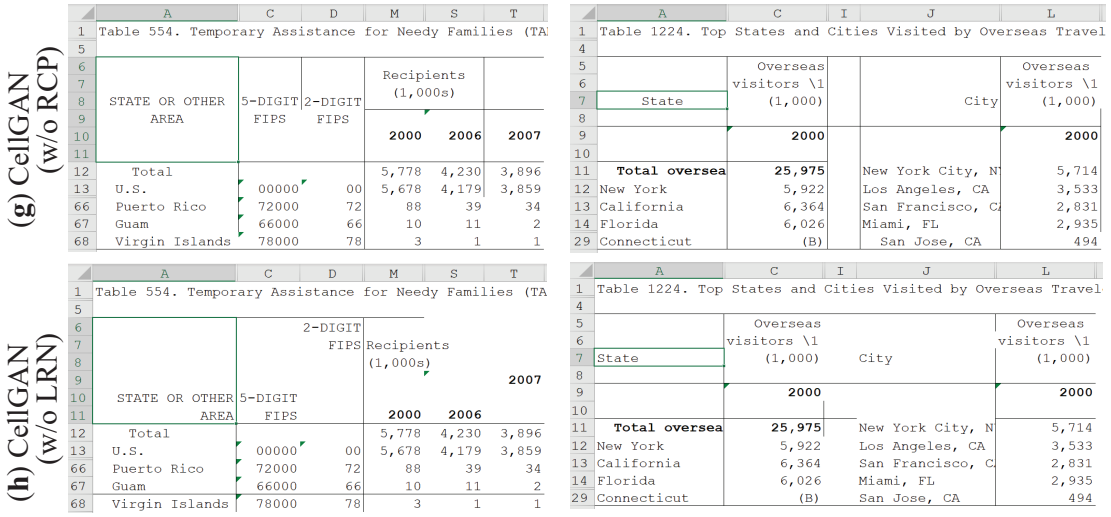


Figure 6: Case study of two CellGAN variants.

Baseline: The left and right cases on Row(a) in Figure 5 show that the rule-based method is brittle. It generates inappropriate borders when cells in the top are not merged, and misses the top header's last row (row 9) in the right case. Row (b) shows that the graphical model fails to fully capture the correlations of formats on the two-dimensonal cell grid, with lots of intermittent borderlines. Rows (c) and (d) in Figure 5 show that without spreadsheet domain-specific techniques, even state-of-the-art models for image generation will produce low-quality results.

Human mistake: For the left case, the results on Rows (e) and (f) in Figure 5 indicate that humans may make mistakes resulting in incomplete borders. CellGAN avoids such mistakes via learning from data on a large scale.

Non-exclusive styles: The right case in Rows (e) and (f) in Figure 5 shows that both two different formats can be reasonable. Top borderlines are added in different ways on rows 9, 10, and 11.

RCP & LRN: Results on Rows (g) and (h) in Figure 6 show that discontinuous, incomplete or redundant borders are generated when RCP or LRN is absent. RCP helps better preserve row/column-wise consistency of formats, while LRN helps prevent local outliers of output formats. Both of them enhance the quality of output formats.

Bad cases of CellGAN: There are also bad cases of CellGAN. Case (a) in Figure 4 shows inappropriate generated borderlines in the red circle. It seems that CellGAN is fooled by the blank region of F14:H19 and tries to generate a bottom borderline for the region F5:H13. Case (b) in Figure 4 shows that the generated borderlines are inconsistent among peer data groups, as highlighed by four circles with different colors. A main cause for this bad case is that the top header of this table is complex with 9 rows of hierachical cells. It indicates that although CellGAN can learn common table structures from large data in an end-to-end way, it can still be further improved for fine-grained table structure extraction.

3.3 Human Perceptual Study

The key purpose of table formatting is to help users better understand the table structure and data correspondence. It is thus important to evaluate the formatting quality based on the subjective perception of humans. We conduct a perceptual study with 16 Excel data professionals from a professional data service supplier[6]. We randomly select 200 tables from each dataset for human evaluation. For each table, the generated formatting and its real-world formatting form a pair for comparison and are presented in random ordering with anonymized labels. A professional has unlimited time to visually inspect the generated formatting against the real-world formatting for each table, and marks the one with **better** quality or a tie for **comparable** qualities according to:

- **Integrity:** Whether the formatting looks complete without missing/redundant local pieces.
- **Effectiveness:** Whether the formatting reflects table structure and data correspondence for easy look-up.
- **Harmony:** Whether the formatting is visually appealing and matches the table in harmony.

As shown in Figure 7, both borders and alignments generated by CellGAN are rated much higher than those produced by alternative approaches. **69.67**% of the generated borders and **80.50**% of the generated alignments obtained by CellGAN are rated better than or comparable to the real-world formats on average. And even for 10.5% of tables in SAUS, the generated borders by CellGAN are considered better than the real-world borders. Results also show that both the row/column-wise pooling and the local refinement network considerably enhance the quality of generated formats. To measure the agreement among human evaluators, we also evaluate Fleiss' kappa, which achieves 0.615, indicating substantial agreement according to [25].

[6] http://en.speechocean.com/

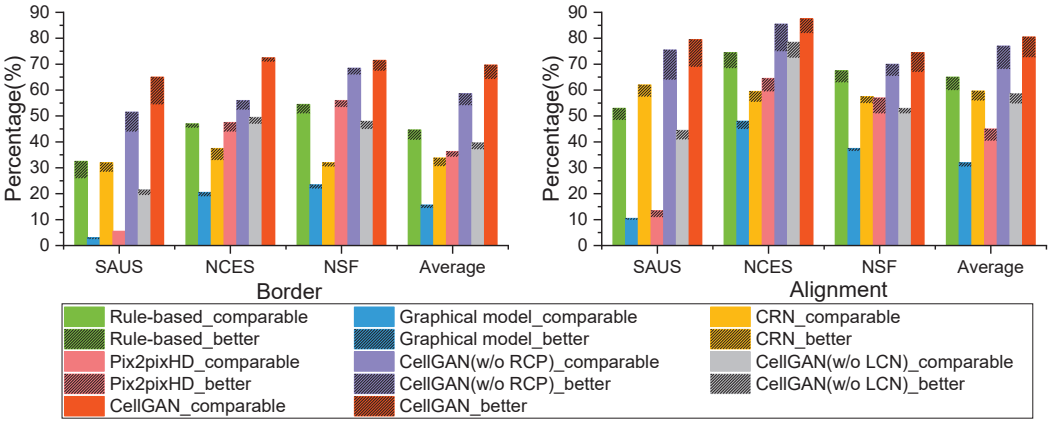


Figure 7: Human evaluation results.

3.4 Quantitative Evaluation

Quantitative evaluation of generative models is known to be challenging, and to the best of our knowledge, there are no existing quantitative metrics for table formatting evaluation. To address this challenge, we define the following two metrics to reflect the quality of generated formats from global and local perspectives.

3.4.1 Table-Level Accuracy (TLA). If a formatting algorithm can generate formats for a given table that exactly matches the real-world formats of the table, the generated formats are considered as high quality. To this end, we employ the Table-Level Accuracy (TLA) as a global metric, which is defined as the proportion of exact matchings among all tables in a test dataset:

$$\text{TLA} = \sum_{n=1}^{N} |f^n_{\text{gen}} == f^n_{\text{real}}|/N, \tag{9}$$

where "==" returns 1 if the generated formats f^n_{gen} of table n and the corresponding real formats f^n_{real} are exactly the same for all cells, and N is the number of test tables.

As Table 3 shows, for border generation, the average TLA of CellGAN achieves 38.93%, which significantly outperforms Pix2pixHD of 17.84% and CRN of 3.95%. Due to the variety of formatting styles, some generated formats which are not "Exactly Matching" the real data may be also reasonable. By comparing Table 3 and Figure 7, we find that results of TLAs are lower than results in human perceptual studies. In particular for the rule-based method, TLA of border generation only achieves 11.9%, much lower than 40.83% in human studies, showing that the rule-based method fails to adapt to various implicit formatting styles in real-world datasets.

3.4.2 Local Patch Metrics (LPM). TLA is quite a strict metric that requires 100% matching between real-world formats and generated formats at table level. However, direct cell-level comparison between formats is also problematic since multiple satisfactory formats can be generated from the same input table conforming to different formatting styles but aligning poorly at cell level. By noting that two generated format results that do not match the real-world formats at table level can have quite different quality in local details, we define LPM to evaluate the local patch-level quality for the generated formats. The general idea is to consider patches from real-world high-quality datasets as "natural" patches, and statistically measure the naturalness of generated formats by matching their patches against "natural" patches.

Similar to n-grams in documents [1], we collect $k \times k$ local format patches from spreadsheet tables with a sliding window. We denote the collection of local format patches as bag-of-format-patches, and build the bag-of-format-patches for both the real-world formats and the generated formats. We then define LPM precision as the percentage of generated patches covered by patches in the training set, LPM recall as the percentage of training patches covered by generated patches, and LPM as their F_1 score in the following:

$$\text{LPM}_{\text{prec.}} = |\{f^{k\times k}_{\text{gen}} \in P_{\text{real}}\}|/|P_{\text{gen}}|, \tag{10}$$

$$\text{LPM}_{\text{recall.}} = |\{f^{k\times k}_{\text{real}} \in P_{\text{gen}}\}/|P_{\text{real}}|, \tag{11}$$

where $P_{\text{real}} := \{f^{k\times k}_{\text{real}}\}$ and $P_{\text{gen}} := \{f^{k\times k}_{\text{gen}}\}$ denote the collections of all real-world patches and generated patches respectively, and $|\cdot|$ denotes the cardinality of a set.

As shown in Table 4, CellGAN achieves much higher average LPM scores than both CRN and Pix2pixHD. This means that the format patches generated by CellGAN are more "natural" with significant overlap with the true format patches in real-world datasets. Results also show that the local refinement network achieves about 4%~12% gains in LPM for borders and alignments.

3.5 Border Density for User Manipulation

We now show how the generated formats can be manipulated by users using border densities as an example. In practice, not all border formats are equally likely to be recommended for a given input table. The generator usually generates a small number of "authentic"-looking candidate formats with discrete border density levels, and is not sensitive to minor changes in the continuous control variable value. Therefore, we discretize the $[0, 1]$ density range into equal intervals with size 0.05 for both training and inference. For training, the generator takes horizontal and vertical

Table 3: Comparison results with the TLA metric.

%	Border				Alignment			
	SAUS	NCES	NSF	AVG	SAUS	NCES	NSF	AVG
Rule-based	7.62	12.18	15.91	11.90	18.46	23.27	23.19	21.64
Graphical model	1.37	7.63	12.48	6.70	1.12	18.31	16.22	11.88
CRN	4.11	3.38	4.37	3.95	**48.40**	27.32	36.68	37.47
Pix2pixHD	0.00	23.17	29.26	17.48	2.28	39.09	32.75	24.71
CellGAN(w/o RCP)	16.89	14.86	41.05	24.27	40.64	41.12	**44.54**	42.10
CellGAN(w/o LRN)	2.74	17.66	21.83	14.08	33.33	48.36	35.81	39.17
CellGAN(full)	**30.14**	**38.61**	**48.03**	**38.93**	**48.40**	**61.68**	44.10	**51.39**

Table 4: Comparison results with the LPM metric.

%		Border				Alignment			
Patch		SAUS	NCES	NSF	AVG	SAUS	NCES	NSF	AVG
4×4	Rule-based	82.22	90.51	84.18	85.64	83.32	75.28	62.45	73.68
	Graphical model	63.24	81.34	77.01	73.86	60.14	73.64	61.90	65.23
	CRN	85.94	87.85	**90.63**	88.14	86.95	58.49	66.86	70.77
	Pix2pixHD	72.67	92.12	88.04	84.28	70.51	76.33	68.57	71.80
	CellGAN(w/o RCP)	92.97	95.07	89.81	92.62	91.28	83.08	**77.23**	83.86
	CellGAN(w/o LRN)	88.83	95.23	88.11	90.72	83.92	81.36	64.44	76.57
	CellGAN(full)	**96.66**	**98.12**	88.86	**94.55**	**92.26**	**89.44**	74.33	**85.34**
6×6	Rule-based	77.04	83.51	78.36	79.64	76.39	60.81	57.96	64.96
	Graphical model	42.17	73.82	72.39	62.79	43.65	50.74	46.70	47.03
	CRN	77.01	79.15	**87.16**	81.11	79.26	44.06	55.66	59.66
	Pix2pixHD	49.99	85.39	82.18	72.52	52.19	61.33	57.25	56.93
	CellGAN(w/o RCP)	86.18	89.94	83.70	86.61	84.85	72.61	**66.44**	74.63
	CellGAN(w/o LRN)	79.22	90.83	82.26	84.11	74.63	66.74	50.45	63.94
	CellGAN(full)	**92.61**	**95.78**	82.48	**90.29**	**86.05**	**80.71**	62.39	**76.39**

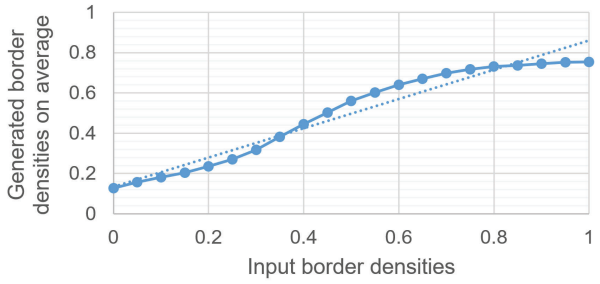


Figure 8: Generated border densities in SAUS.

border densities as conditions. For inference, users can specify the density attributes to generate formats with desired density levels.

We feed different border densities to the generator on test sheets and examine the variations of density values for the generated formats. Figure 8 plots the average output density values on the test dataset over different input user-specified density values to the generator. It can be clearly seen that the generated density values are proportional to the input density values with an approximately linear relationship highlighted by the dotted line. This demonstrates the effectiveness of user manipulation with control variables. We also show an intuitive example in Figure 9. By grouping the same results for different input densities, we find three major patterns with different generated borders. All look realistic by reflecting different levels of data groups, showing the ability of CellGAN to navigate intrinsic styles in the space of acceptable formats.

4 RELATED WORK

Spreadsheet Table Formatting To the best of our knowledge, this is the first study on learning spreadsheet formats. Hardly any related study or spreadsheet tool is available to address this task. In other related domains, we find some work on designing or assessing presentations, webpages, and visualizations. [26] targets to visual-textual presentation layout, e.g., digital magazine cover, poster, and Power Point slides, by formulating the typography as an energy optimization problem given pre-defined aesthetic principles and topic-dependent templates. [9] proposes a deep learning method for visualization assessment, and [6, 18] introduce ways to assess aesthetics of websites. But due to the gap between these domains on data structures and task formulations, techniques for these domains are not quite applicable to spreadsheet formatting.

CGANs CGANs have been typically used to learn a mapping from one domain to another in conditional settings, and have achieved

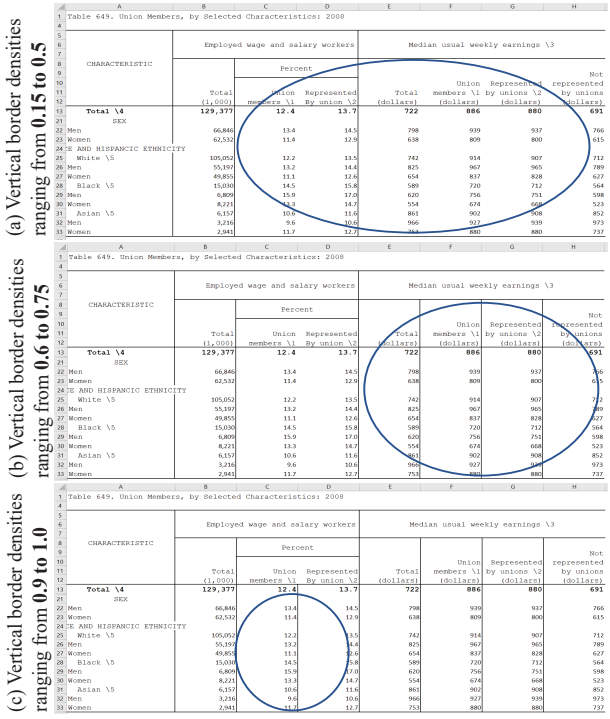


Figure 9: An example for border density manipulation in SAUS. CellGAN generates borders with different densities.

impressive results on inpainting [21], style transfer [11], and image manipulation guided by user constraints [27]. CellGAN is the first research effort leveraging cGANs for solving problems in the spreadsheet domain and remarkably outperforms state-of-the-art cGAN models that are specialized for other domains [24].

5 DISCUSSION AND CONCLUSION

In this paper, we propose a new problem, auto-formatting for spreadsheet tables. First, we provide the first formulation of the table auto-formatting problem. Second, as a new task, auto-formatting lacks practical metrics for evaluation. we propose both quantitative evaluations and human perception studies, which help to compare the effectiveness of different methods in practice. Third, different from domains such as computer vision and natural language processing, where models such as CNNs and LSTMs are well studied and proved highly effective in a wide range of tasks, there lacks a widely adopted model in spreadsheet domain, especially in generation tasks. CellGAN is the first deep model tailored for the spreadsheet domain by learning the latent mapping from table data to table formats in a generative adversarial way. We also explored user-controllable manipulation on border generation for interactive formatting. In the future, we plan to investigate fine-grained table structure extraction to enable high-quality formatting generation for complex tables.

REFERENCES

[1] Cavnar, W. B., Trenkle, J. M., et al. N-gram-based text categorization. *Ann arbor mi 48113, 2* (1994), 161–175.

[2] Chen, Q., and Koltun, V. Photographic image synthesis with cascaded refinement networks. In *IEEE International Conference on Computer Vision (ICCV)* (2017), vol. 1, p. 3.

[3] Chen, Z., and Cafarella, M. Integrating spreadsheet data via accurate and low-effort extraction. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014), pp. 1126–1135.

[4] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 8789–8797.

[5] Dong, H., Liu, S., Han, S., Fu, Z., and Zhang, D. Tablesense: Spreadsheet table detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 69–76.

[6] Dou, Q., Zheng, X. S., Sun, T., and Heng, P.-A. Webthetics: quantifying webpage aesthetics with deep learning. *International Journal of Human-Computer Studies 124* (2019), 56–66.

[7] Durugkar, I., Gemp, I., and Mahadevan, S. Generative multi-adversarial networks. *arXiv preprint arXiv:1611.01673* (2016).

[8] Fang, J., Mitra, P., Tang, Z., and Giles, C. L. Table header detection and classification. In *AAAI* (2012), pp. 599–605.

[9] Fu, X., Wang, Y., Dong, H., Cui, W., and Zhang, H. Visualization assessment: A machine learning approach. In *2019 IEEE Visualization Conference (VIS)* (2019), IEEE, pp. 126–130.

[10] Gatterbauer, W., and Bohunsky, P. Table extraction using spatial reasoning on the css2 visual box model. In *Proceedings of the National Conference on Artificial Intelligence* (2006), vol. 21, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, p. 1313.

[11] Gatys, L. A., Ecker, A. S., and Bethge, M. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2414–2423.

[12] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[13] Kim, T., Cha, M., Kim, H., Lee, J. K., and Kim, J. Learning to discover cross-domain relations with generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70* (2017), JMLR. org, pp. 1857–1865.

[14] Kingma, D. P., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[15] Koller, D., and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[16] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105.

[17] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3431–3440.

[18] Miniukovich, A., and Marchese, M. Relationship between visual complexity and aesthetics of webpages. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (2020), pp. 1–13.

[19] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[20] Nishida, K., Sadamitsu, K., Higashinaka, R., and Matsuo, Y. Understanding the semantic structures of tables with a hybrid deep neural network architecture. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[21] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2536–2544.

[22] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[23] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. In *Computer Vision and Pattern Recognition (CVPR)* (2017), vol. 1, p. 4.

[24] Wang, T.-C., Liu, M.-Y., Zhu, J.-Y., Tao, A., Kautz, J., and Catanzaro, B. High-resolution image synthesis and semantic manipulation with conditional gans. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), vol. 1, p. 5.

[25] Wikipedia contributors. Fleiss' kappa — Wikipedia, the free encyclopedia, 2019. [Online; accessed 3-January-2020].

[26] Yang, X., Mei, T., Xu, Y.-Q., Rui, Y., and Li, S. Automatic generation of visual-textual presentation layout. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 12, 2* (2016), 1–22.

[27] Zhu, J.-Y., Krähenbühl, P., Shechtman, E., and Efros, A. A. Generative visual manipulation on the natural image manifold. In *European Conference on Computer Vision* (2016), Springer, pp. 597–613.