Pattern Recognition Letters 167 (2023) 38–44

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Automatic generation of scientific papers for data augmentation in document layout analysis☆

Lorenzo Pisaneschi*, Andrea Gemelli, Simone Marinai

University of Florence, Department of Information Engineering, Via di Santa Marta 3, Firenze, 50139, Italy

ARTICLE INFO

Article history:
Received 13 August 2022
Revised 17 December 2022
Accepted 28 January 2023
Available online 31 January 2023

Edited by Maria De Marsico

Keywords:
Document layout analysis
Transformers
Automatic document generation
Deep learning
Document object detection
Generative models

ABSTRACT

Document layout analysis is an important task to extract information from scientific literature. Deep-learning solutions for document layout analysis require large collections of training data that are not always available. We generate a large number of synthetic pages to subsequently train a neural network to perform document object detection. The proposed pipeline allows users to deal with less common layouts for which it is not easy to find large annotated datasets. High-quality annotations for a small collection of papers are obtained through a semi-automatic approach. Then, a generative model, based on LayoutTransformer, is used to generate plausible layouts that are subsequently populated with random information to perform data augmentation. We evaluate the proposed method considering scientific articles with two different types of layouts: double and single columns. For double-column papers, we improve detection by 1% starting from 385 manually annotated scientific articles. For single-column papers, we improve detection by 49% starting from 218 articles.

1. Introduction

Document Layout Analysis (DLA) [1] is an important task in document image analysis aimed at understanding the document structure and semantics. Nowadays, DLA is a mature application area that, similarly to other research fields, evolved from well-established hand-designed techniques [2] into approaches based on deep-learning techniques [3] that often require large collections of labeled training data. Scientific documents are usually released as born-digital PDF files where the extraction of textual information is often not an issue. However, to understand the document semantics, the identification of regions in the pages is needed. Deep learning techniques are widely used for document layout analysis, mostly relying on algorithms originally designed for computer vision which have been demonstrated to be effective in recognizing document layout elements when trained with a huge amount of data.

Large annotated datasets are available to support layout analysis research (e.g. [4,5]), however, the papers in these collections do not cover all the possible layouts and publishing styles since they mostly come from the biomedical literature (e.g. PubMed) and self-archiving platforms such as arXiv. The most common strategy adopted to automatically annotate large collections of scientific papers requires the availability, for each PDF, of the corresponding source files. There are two main alternatives to get this information: 1) source files can be submitted to a sharing repository (e.g. arXiv[1]) together with the corresponding PDF by the authors; 2) PDF papers and their corresponding XML files can be distributed by PubMed Central (PMC[2] a free digital archive of biomedical and life sciences journals) and in this case the content is contributed by the journal publishers.

Once a PDF and the corresponding structured representation are available, it is possible to match the information and automatically obtain annotations for objects in the page. In both cases, the collections of scientific papers are large and freely available and are therefore ideal for building large annotated datasets that can be used to train deep-learning based models for DLA. As a matter of fact, available datasets, such as PubLayNet and DocBank [4,5], exhibit limited variability in content and layout: they collect papers from arXiv or PubMed Central and mostly include double column layouts. As a consequence, when trained models are applied to different layouts (e.g. single-page proceedings) the results can be worse than expected. To address these problems, we aim to

☆ Editor: Maria De Marsico.

* Corresponding author.
E-mail addresses: lorenzo.pisaneschi@unifi.it (L. Pisaneschi), andrea.gemelli@unifi.it (A. Gemelli), simone.marinai@unifi.it (S. Marinai).

[1] https://arxiv.org/
[2] https://www.ncbi.nlm.nih.gov/pmc/

https://doi.org/10.1016/j.patrec.2023.01.018

support the research on DLA by designing a synthetic data generator that can be easily customized to generate documents of a desired specific layout, to augment the training data and boost the performance of a pre-trained object detector model. There are two main contributions of this work:

- We propose a semi-automatic annotation approach to obtain high-quality annotations for a small set of scientific papers; annotation errors are manually fixed and 11 different regions (title, authors, abstract, keywords, subtitle, text, image, table, caption, formula, reference) are labeled; these annotations are used for data augmentation.
- We propose a data generation pipeline that starts from the annotated pages to train a generative model (based on LayoutTransformer [6]) and produce large collections of pages with variable layout whose regions are populated with synthetically generated content (e.g. text, images, tables). These pages can then be used to train DLA engines.

The overall approach is layout-agnostic. Our experiments show that it can work for both double and single-column scientific papers and we demonstrate that it is possible to improve the results of layout analysis by using a small semi-automatically annotated dataset. The code of this paper is available at[3].

The paper is structured as follows: in Section 2 we revise the related work on syntetic document generation. In Sections 3 and 4 we explain how data are annotated and the method in use to generate new layouts. In Section 5, we show numerical and qualitative experimental results. Conclusions are drawn in Section 6.

2. Related work

The generation of synthetic data for boosting the recognition performance of trainable models has been extensively used in the last years and research on DLA is not an exception. Some methods require the user to carefully design rules that describe the distribution of regions in the page layout. For instance, in [7] a generator of semi-structured documents is proposed. This tool generates samples of administrative documents requiring the user to design by hand the general structure (defines the different types of information needed). the tool then modifies the position of items in the page by taking into account suitable random variables. Structured historical documents are generated, starting from a hand-designed general organization of the document in [8]. In [9] a generative process that treats every physical component of a document as a random variable and models their intrinsic dependencies using a Bayesian Network graph is proposed. The Bayesian Network defines the synthetic document generation process. The primitive units, style attributes, and layout elements are all treated as random variables and represented as nodes in a graph. Several sub-networks are designed to model different parts of a scientific paper (e.g. document, section, table, and figure). User-defined layout and font styles are considered as starting information in Document Domain Randomization (DDR [10]) to render simulated pages of scientific papers by modeling randomized textual and non-textual contents of interest that are downloaded from online repositories.

More recently, several generative models for document layout generation have been proposed. In [11], the authors try to learn layouts from a set of document images, leveraging GAN (Generative Adversarial Network). In [12] an architecture based on VAE (Variational Auto-Encoder) and RNN (Recurrent Neural Network) is described. It can convert training page layouts into an embedded compact representation, whose space is represented by a Gaussian distribution. New document layouts can be created by sampling novel values from the distribution. A GAN-based method to generate document layout given images, keywords and category of the document is described in [13]. One problem with this approach is that a large amount of training data is needed. Also transformers have been used to generate layouts. In [14] the authors model the distribution of objects in example layouts and a self-attention mechanism is used to detect high-level object relationships. LayoutTransformer, proposed in [6] is a framework that leverages self-attention to learn contextual relationships between layout elements and generates new layouts in the given domain. The authors exploit it to generate scientific papers, but only rely on PubLayNet for training.

Since transformers came into play [15] their usage has been spread among a wide range of machine learning communities [16]. Even if they are mostly used for NLP tasks, transformers have been applied also to tackle other document-related tasks, like document classification [17–20], table detection [17], key information extraction from receipts or forms [18,20,21], and Document Layout Analysis [1,5,22].

Motivated by the transformers' achievements in document analysis, we make use of the LayoutTransformer generative model [6] to perform data augmentation by first creating synthetic layouts starting from a small number of annotated pages and then populating the regions with techniques inspired by DDR [10].

3. Semi-automatic annotation

The pipeline of the proposed approach is summarized in Fig. 1. Given a few papers from the target publication (journal or conference proceedings) some preliminary annotations are obtained by integrating the output of two PDF parsing tools (Section 3.1). These annotations are then refined by users (Section 3.2) and clean data are used to train the generative model as described in Section 4.

3.1. Automatic annotation

Since PDF files faithfully represent the layout and typographic information of documents, but not their semantics, we parse the PDF papers with two tools: the GROBID [23] machine learning library, focused on extracting bibliographic information from scientific papers, and the PDFMiner library [24] that extracts layout elements in the pages. The GROBID output is a TEI XML file [25] that represents the metadata of the paper. In particular, the TEI tags describe the class type and the bounding box for titles, sub-titles, tables, and formulas. However, in the case of authors, abstract, and keywords the position information is missing. The PDF files are then processed by PDFMiner which extracts all the text blocks and images with their bounding box coordinates. The information extracted with the two tools is then merged and nine object classes are identified as follows.

- *Titles, subtitles, tables* and *formulas* (arrow 1 in Fig. 1) are identified in the GROBID output using beautifulsoup4 [26] to parse the TEI XML files.
- GROBID also extracts *abstract, authors*, and *keywords* (arrow 2 in Fig. 1), in this case, but there is no bounding box information in the TEI tags. Since all the text-boxes are extracted by PDFMiner it is possible to find position information by using text matching between GROBID and PDFMiner results. The text matching is computed with *SequenceMatcher* (from Python difflib [27]) considering two strings as similar if the score is greater than a threshold set to 0.7 (the threshold is a balance between too much noise with low values and too many false negatives with high ones).

[3] https://github.com/pisalore/master_thesis.

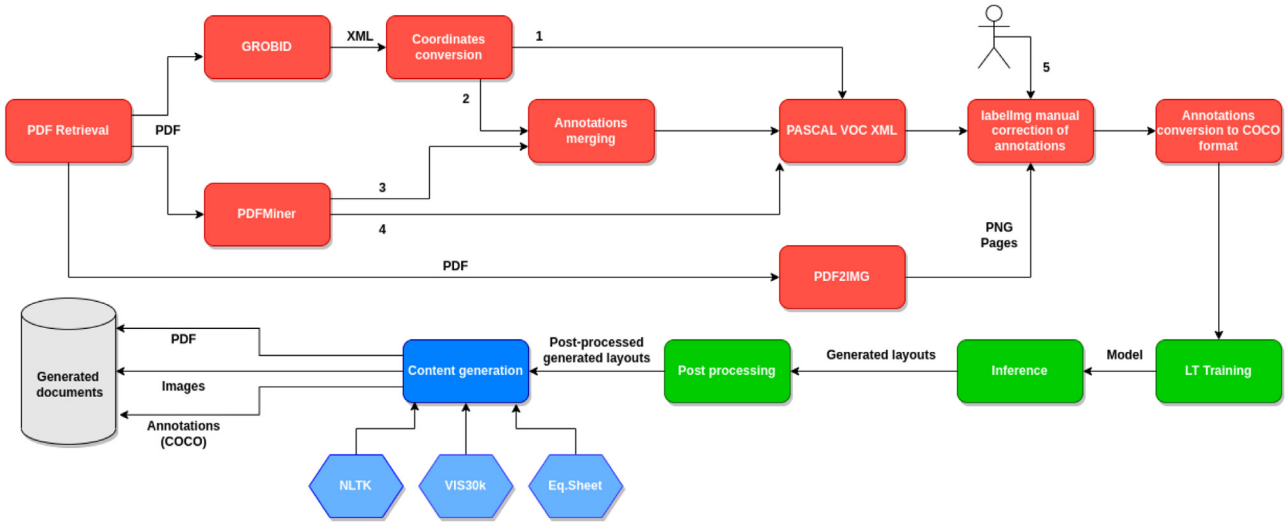

Fig. 1. The proposed pipeline: first, data are collected, annotated in a semi-automatic fashion, and finally manually corrected (red boxes); then they are used to train a LayoutTransformer generator model that in turn generates synthesized layouts (green boxes) that are filled with content (blue box). We obtain a dataset of documents belonging to the domain of the initial data. Labels in the diagram are as follows: 1: Formulas, titles, tables; 2: Abstract, authors, keywords; 3: Text; 4: Figures; LT: LayoutTransformer.

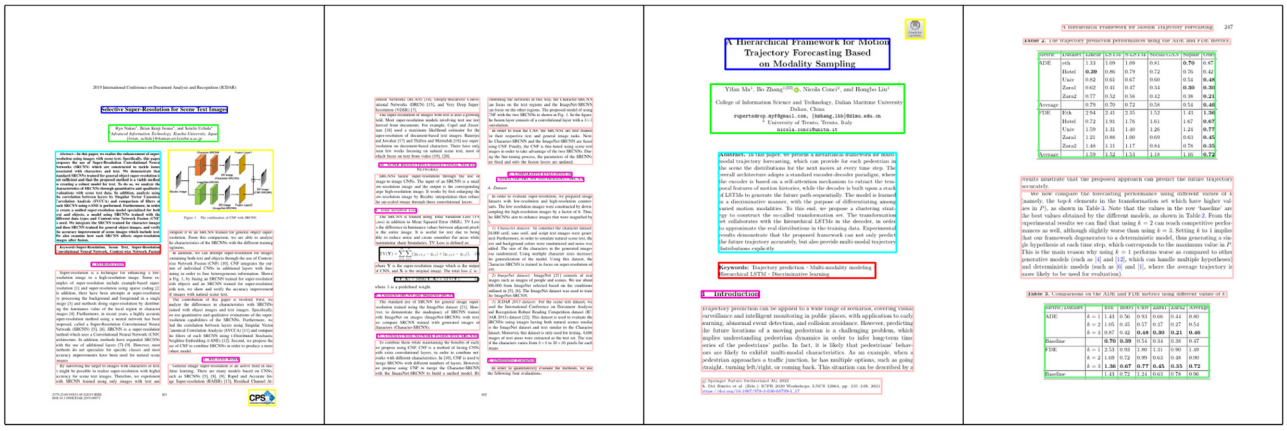

Fig. 2. Examples of automatically annotated pages. From GROBID and PDFMiner we extract nine categories: title (blue), authors (light green), abstract (cyan), keywords (red), subtitles (purple), text (pink), images (yellow), formulas (black) and tables (green). Captions and references are added by hand.

- The *text* regions extracted by PDFMiner (arrow 3 in Fig. 1) that are not matched with the output of GROBID are first filtered with some heuristics to exclude false positives and then saved as *text*. For instance, text regions smaller than the font size are removed. The other objects extracted by PDFMiner are *figures* (arrow 4 in Fig. 1).

In the subsequent manual correction, the labels of text regions corresponding to *captions* and *references* are fixed obtaining regions for the 11 classes that are used by the generative model. Annotation examples for double and single columns are shown in Fig. 2. After performing the previous steps the annotation information is converted to a format suitable for manual correction. For each processed paper we create page images (using the pdf2image python library [28]) and convert the annotations as XML files in PASCAL VOC format [29].

3.2. Annotation correction

Not surprisingly, there are some errors in the automatic annotation. We therefore manually check the annotations using labelImg [30] (arrow 5 in Fig. 1), an interactive tool that reads PASCAL VOC annotations, supports the user to fix errors, and saves the updated annotations.

Corrections performed include the addition of missing objects (e.g. tables, images, captions, and page numbers) and the separation of overlapping objects that we want to avoid.

Since the proposed content generation (Section 4.3) cannot easily handle overlapping regions, we deal with the overlaps by removing the shared area and reducing the overlapping regions accordingly. Given simple annotation guidelines, for each page, the annotators needed a time ranging from 30 s to two minutes depending on the complexity of the page (i.e. number of different regions in the page).

By correcting the annotations we expect to help the generative model to produce better training samples: considering that we started with very few training examples, it is crucial to work with high-quality annotations to produce plausible results. We will show how this step improves also the final results by a large margin, in the experiments presented in Section 5.3. The last step is to prepare the data to use them to train the generative model (LayoutTransformer) that accepts data in COCO format [31].

4. Document generation

After preparing the data, a generative model is used to increase the number of samples to train a computer vision model for DLA. The document generation follows three main steps (bottom part of

Fig. 3. Examples of training examples (top) and generated pages before manual correction (bottom). Double column layout in the left, single column layout on the right. Region colors are the same as those used in Fig. 2 apart from the orange box in the last example that corresponds to References.

Fig. 1). 1) the generative model is trained from annotated data; 2) layouts are generated using the model and then adjusted in post-processing; 3) regions in the synthetic layouts are populated with realistic content.

4.1. Layout transformer training

The goal is to generate many document pages with a fair amount of variance in content, starting from a small set of input data. To do this, we use the generative layout method LayoutTransformer [6], motivated by the results achieved on the PubLayNet dataset.

We summarize here some key information about the model. The reader can refer to [6] for additional details. The model uses self-attention [15] to learn contextual relationships between objects in the layout of a document page. A layout can be represented as a set of bounding boxes of regions of different types. The layout of one page can be defined as a graph $\mathcal{G}$ with n nodes, where each node $i \in \{1, \ldots, n\}$ is a region. The graph is fully connected and the goal of the attention network is to learn the relationships occurring between nodes. Each node can be represented as $(s_i, x_i, y_i, h_i, w_i)$, where s_i is the feature vector, (x_i, y_i) are the coordinates of the center, and (h_i, w_i) are the height and width of the node bounding box.

The model takes as input a permutation of nodes and their d-dimensional vector representation. It is important to note that by doing this, the attention module can assign weights explicitly to each layout element. The attention module is similar to transformers' decoder: it is formed by L attention layers each of which is composed of masked multi-head attention and a fully connected feed-forward layer, residual connections and Layer Normalization. Each d-dimensional representation attends to all the input latent vectors as well as previously predicted latent vectors. *Teacher forcing* is used at training and validation time, so ground truth samples are used instead of the previous step output, to train the model efficiently. Regarding the training loss, the KL-Divergence is minimized between softmax prediction by softmax layer, outputting a one-hot distribution with label smoothing, a regularization technique which prevents over-fitting and also the model being over-confident.

In this work we use the following model and training hyperparameters: $d = 512$ as the embedding dimension for each element in the layout, $L = 6$ as the layers number and $n_{heads} = 8$ as the number of multi-attention heads in Transformer architecture. Concerning the layout parameters, *max length* $= 128$ indicates the maximum number of elements to be generated for the layout (most real pages have fewer regions). Regarding training, Adam optimizer is used, with $\beta_1 = 0.9$ and $\beta_2 = 0.99$, $lr = 10^{-4}$. The network is also set up to synthesize 612 x 792 and 440 x 667 layout pages, respectively for double and single columns. In the top part of Fig. 3 we show some examples of training layouts.

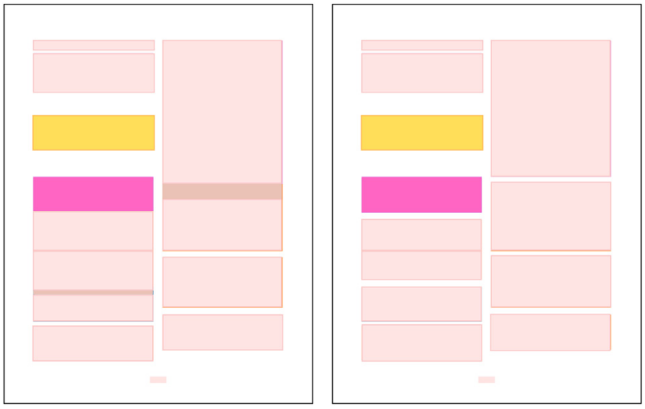

Fig. 4. Example of post-processing layout corrections: overlapping bounding boxes in the left are correctly separated in the right.

4.2. Layout generation

Exploiting LayoutTransformer, trained from scratch with few input samples, it is possible to generate an arbitrary number of synthetic pages. In the second row of Fig. 3 we show some examples of generated layouts. it is possible to observe the strengths and weaknesses of the method: the ability to recognize and then generate the high-level page layout, but also overlapping regions that are not realistic. To remove overlapping regions, post-processing is performed with a set of operations: (i) group bounding boxes by category; (ii) identify overlapping objects; (iii) merge overlapping boxes; (iv) create extra annotations; (v) remove small noisy annotations. In Fig. 4 we show the input and the output of the post-processing on a sample page where some regions overlap.

4.3. Content generation

Once the synthetic layout is generated, it is possible to fill the generated bounding boxes with suitable content. Textual objects in

Table 1
Statistics of training and test data.

Layout	# Samples			Train	Test
	PDF	Pages	Regions	Pages	Pages
ICDAR (2C)	385	2088	15,478	1044	1044
with aug.	-	12,079	147,960	12,079	1044
ICPR (1C)	218	2088	8242	1044	1044
with aug.	-	12,582	115,170	12,582	1044

the generated PDF files are obtained by using a simple language model based on n-grams that assigns a probability to sequences of words. Our goal is to make the text as realistic as possible to train a DLA model. However, we do not aim at generating "fake" papers with semantically meaningful content. We generate the text exploiting NLTK (Natural Language Tool Kit) [32]. We use different language models for different types of regions and therefore there are separate language models for titles, authors, abstracts, subtitles, texts, and references. Given textual instances for each type of region, we use them to train the different language models and then to generate fake text to be used in document generation.

In addition to text, we also fill the generated layouts with images, tables, and formulas gathered from the Internet. The different sources for the contents of interest are as follows:

- *Text categories*: the bounding boxes labeled as titles, authors, subtitles, keywords, text, abstract, caption, and references are filled with synthetically generated text using specific 3-grams models.
- *Images categories*: figure and tables. The bounding boxes labeled as figures and tables are filled with images retrieved from VISImageNavigator[4] a collection of more than 30K figures and tables from proceedings of conferences.
- *Mathematical objects*: the bounding boxes corresponding to formulas or equations are populated with formulas samples from the EquationSheet dataset[5] then typewritten in the PDF using *pylatex*.

Final examples with generated layout and content can be evaluated qualitatively in Fig. 5.

5. Experiments

We evaluate the quality of generated documents using a ResNeXt [33] model to perform DLA on proceedings of ICDAR 2019 (double columns layout) and of the ICPR 2021 (single column layout) workshops. Starting from the same amount of annotated pages, we generate synthetic pages and compare the final results with and without data augmentation.

5.1. Training and test data

The pipeline described in Sections 3 and 4 is layout-agnostic, meaning that can work both for single and double-column layouts. To show this behavior, we used two different sources of scientific papers. We collect 2088 pages for each conference after the semi-automatic annotation process, then we augment their number by nearly 10k more samples using the generative model. In Table 1 we summarize the statistics of the training and test data: 50% of the original pages are retained as test set, while the rest are for training the LayoutTransformer. The number of pages and regions after data augmentation is also reported.

[4] https://visimagenavigator.github.io/.
[5] https://www.equationsheet.com/.

Table 2
Summary of experimental results: comparison of AP measures for different ResNeXt models trained on different data.

Layout	Models	Metrics (↑)		
		$AP_{0.50:.95}$	AP_{50}	AP_{75}
ICDAR 2 Columns	Baseline	0.735	0.917	0.809
	Augmented	0.681	0.913	0.776
	Augmented-refined	**0.745**	**0.949**	**0.834**
ICPR-W 1 Column	Baseline	0.107	0.256	0.077
	Augmented	0.079	0.124	0.018
	Augmented-refined	**0.597**	**0.823**	**0.764**

5.2. Object detector for DLA

We use an object detector to validate the quality of the generation process, performing layout analysis on the original and augmented data. To tackle DLA we make use of ResNeXt [34], taken from the detectron2 [35] model zoo. The model is initialized with DocBank weights [5]. Regarding training details, we use a learning rate warm-up (from 0.2 to 0.6) for the first 1.500 steps. We iterated for 10.000 steps for about 5 h. Concerning ResNeXt we left cardinality to 32, bottleneck width to 8 and depth to 101. The batch size is set to 8.

5.3. Results

Depending on the use of the data augmentation or not, we define three different models, trained on different data:

1. *Baseline*: only trained with corrected annotated pages.
2. *Augmented*: trained adding generated pages obtained from uncorrected annotations.
3. *Augmented-refined*: trained adding generated pages obtained from corrected annotations.

In the single-column case, due to the high unbalance of classes, we train three different models for data augmentation using three different training sub-sets: 1) one model consisting of first pages (those with title and authors); 2) one containing only pages with at least a "reference" instance 3) the last one including all the remaining pages. In Table 2 experimental results are presented: the metric in use is the mAP proposed by COCO, with different thresholds (0.5, 0.75 and 0.50:.05:.95). From the results we can notice:

- Data augmentation improves all the metric scores: dealing with a small dataset well-annotated (baseline) does not allow a deep model such as ResNeXt to generalize well, but increasing the size of the training set to improve results.
- In particular, we improve the $AP_{.50:.95}$ scores by 1% for double column and a considerable margin of 49% for single column. The worst results (in particular on the baseline) for single-column documents, in comparison with double-columns, are due to the initialization of the model with weights learned from the DocBank dataset, which is mostly composed by double columns papers. Thanks to the proposed approach, it is possible to do domain adaptation from double-column to single-column layouts.
- The removal of noise produced by the automatic annotation process, through manual correction of annotations, shows that the combination of semi-automatic labeling of a small amount of data along with a generative model, can improve the quality of generated documents. We obtain an average improvement for all the metrics of 5.2% on the double column and 65% on single column layouts, comparing augmented and augmented-refined models.

Fig. 5. Examples of generated documents: double-column on the upper side, single column on the bottom one.

6. Conclusions

In this paper, we propose a method to synthesize an arbitrarily large number of pages of scientific articles from few annotated pages. We implemented a semi-automatic pipeline to enhance the quality of layout annotations of scientific papers, exploiting information contained in the PDF. On top of a small amount of data, we work with a LayoutTransformer to generate new document layouts that are then filled with realistic content to create high-resolution synthetic data. Finally, we evaluate the effectiveness of the data augmentation proposed using a fine-tuned object detector model, obtaining an improvement in the mAP score for single and double-

column layouts. We have shown that a good quality of annotations with the support of a generative model can be efficient compared to the automatic annotation of large-scale data. Moreover, the generation of new data can help the pre-trained models to generalize also to different layouts. In future work, we aim to extend this pattern to different document layouts, such as legal documents, invoices, and medical records.

An important aspect we will explore more deeply is the semantic content of generated scientific paper pages, that relates to the possibility to develop a model that is able to generate documents in different domains. It might be also interesting to refine the generation of text and other contents to be meaningful and mutually correlated. In particular, content-aware of its relative (e.g., in which section) and absolute (e.g., in which page) positioning could allow the generation of complete papers. In addition, creating better synthetic data could permit the application of layout analysis methods that take into consideration also text information.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

A link to the github code is present into the paper

References

[1] G.M. Binmakhashen, S.A. Mahmoud, Document layout analysis: a comprehensive survey, ACM Comput. Surv. 52 (6) (2019), doi:10.1145/3355610.

[2] K. Kise, Page segmentation techniques in document analysis, in: D.S. Doermann, K. Tombre (Eds.), Handbook of Document Image Processing and Recognition, Springer, Berlin, Heidelberg, 2014, pp. 135–175, doi:10.1007/978-0-85729-859-1_5.

[3] L. Cui, Y. Xu, T. Lv, F. Wei, Document AI: benchmarks, models and applications, 2021, 10.48550/ARXIV.2111.08609 10.48550/ARXIV.2111.08609.

[4] X. Zhong, J. Tang, A.J. Yepes, PubLayNet: largest dataset ever for document layout analysis, in: 2019 International Conference on Document Analysis and Recognition (ICDAR), IEEE, 2019, pp. 1015–1022, doi:10.1109/ICDAR.2019.00166.

[5] M. Li, Y. Xu, L. Cui, S. Huang, F. Wei, Z. Li, M. Zhou, DocBank: a benchmark dataset for document layout analysis, in: D. Scott, N. Bel, C. Zong (Eds.), Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8–13, 2020, International Committee on Computational Linguistics, 2020, pp. 949–960, doi:10.18653/v1/2020.coling-main.82.

[6] K. Gupta, J. Lazarow, A. Achille, L.S. Davis, V. Mahadevan, A. Shrivastava, LayoutTransformer: layout generation and completion with self-attention, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 1004–1014.

[7] D. Belhadj, Y. Belaïd, A. Belaïd, Automatic generation of semi-structured documents, in: E.H.B. Smith, U. Pal (Eds.), Document Analysis and Recognition, ICDAR 2021 Workshops, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II, Lecture Notes in Computer Science, Vol. 12917, Springer, 2021, pp. 191–205, doi:10.1007/978-3-030-86159-9_13.

[8] S. Capobianco, S. Marinai, DocEmul: a toolkit to generate structured historical documents, in: 14th IAPR International Conference on Document Analysis and Recognition, ICDAR 2017, Kyoto, Japan, November 9–15, 2017, IEEE, 2017, pp. 1186–1191, doi:10.1109/ICDAR.2017.196.

[9] N. Raman, S. Shah, M. Veloso, Synthetic document generator for annotation-free layout recognition, Pattern Recognit. 128 (2022) 108660, doi:10.1016/j.patcog.2022.108660.

[10] M. Ling, J. Chen, T. Möller, P. Isenberg, T. Isenberg, M. Sedlmair, R.S. Laramee, H.-W. Shen, J. Wu, C.L. Giles, Document domain randomization for deep learning document layout extraction, in: Document Analysis and Recognition – ICDAR 2021, Springer International Publishing, Berlin, 2021, pp. 497–513, doi:10.1007/978-3-030-86549-8_32.

[11] S. Biswas, P. Riba, J. Lladós, U. Pal, DocSynth: a layout guided approach for controllable document image synthesis, in: International Conference on Document Analysis and Recognition, Springer, 2021, pp. 555–568.

[12] A.G. Patil, O. Ben-Eliezer, O. Perel, H. Averbuch-Elor, READ: recursive autoencoders for document layout generation, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14–19, 2020, Computer Vision Foundation/IEEE, 2020, pp. 2316–2325, doi:10.1109/CVPRW50498.2020.00280. https://openaccess.thecvf.com/content_CVPRW_2020/html/w34/Patil_READ_Recursive_Autoencoders_for_Document_Layout_Generation_CVPRW_2020_paper.html.

[13] X. Zheng, X. Qiao, Y. Cao, R. Lau, Content-aware generative modeling of graphic design layouts, ACM Trans. Graph. 38 (2019) 1–15, doi:10.1145/3306346.3322971.

[14] D.M. Arroyo, J. Postels, F. Tombari, Variational transformer networks for layout generation, in: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19–25, 2021, Computer Vision Foundation/IEEE, 2021, pp. 13642–13652, doi:10.1109/CVPR46437.2021.01343.

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U. von Luxburg, S. Bengio, H.M. Wallach, R. Fergus, S.V.N. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA, 2017, pp. 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[16] T. Lin, Y. Wang, X. Liu, X. Qiu, A Survey of Transformers, CoRR (2021) abs/2106.04554.

[17] J. Li, Y. Xu, T. Lv, L. Cui, C. Zhang, F. Wei, DiT: self-supervised pre-training for document image transformer, in: J. Magalhães, A.D. Bimbo, S. Satoh, N. Sebe, X. Alameda-Pineda, Q. Jin, V. Oria, L. Toni (Eds.), MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10–14, 2022, ACM, 2022, pp. 3530–3539, doi:10.1145/3503161.3547911.

[18] J. Wang, L. Jin, K. Ding, LiLT: a simple yet effective language-independent layout transformer for structured document understanding, in: S. Muresan, P. Nakov, A. Villavicencio (Eds.), Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22–27, 2022, Association for Computational Linguistics, 2022, pp. 7747–7757, doi:10.18653/v1/2022.acl-long.534.

[19] S. Appalaraju, B. Jasani, B.U. Kota, Y. Xie, R. Manmatha, DocFormer: end-to-end transformer for document understanding, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10–17, 2021, IEEE, 2021, pp. 973–983, doi:10.1109/ICCV48922.2021.00103.

[20] Y. Huang, T. Lv, L. Cui, Y. Lu, F. Wei, LayoutLMv3: pre-training for document AI with unified text and image masking, in: J. Magalhães, A.D. Bimbo, S. Satoh, N. Sebe, X. Alameda-Pineda, Q. Jin, V. Oria, L. Toni (Eds.), MM '22: The 30th ACM International Conference on Multimedia, Lisboa, Portugal, October 10–14, 2022, ACM, 2022, pp. 4083–4091, doi:10.1145/3503161.3548112.

[21] T. Hong, D. Kim, M. Ji, W. Hwang, D. Nam, S. Park, BROS: a pre-trained language model for understanding texts in document (2020).

[22] P. Zhang, C. Li, L. Qiao, Z. Cheng, S. Pu, Y. Niu, F. Wu, VSR: a unified framework for document layout analysis combining vision, semantics and relations, in: International Conference on Document Analysis and Recognition, Springer, 2021, pp. 115–130.

[23] Grobid, 2008–2021, (https://www.github.com/kermitt2/grobid), 1:dir:dab86b296e3c3216e2241968f0d63b68e8209d3c.

[24] Pdfminer, 2019, (https://pypi.org/project/pdfminer/).

[25] T. Consortium, TEI P5: guidelines for electronic text encoding and interchange, 2021, 10.5281/zenodo.4609855.

[26] beautifulsoup4, 2016, (https://pypi.org/project/beautifulsoup4/).

[27] difflib, 2022, (https://www,docs.python.org/3/library/difflib.html).

[28] pdf2image, 2021, (https://pypi.org/project/pdf2image/).

[29] M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2010) 303–338.

[30] labelImg, 2022, (https://www.github.com/tzutalin/labelImg).

[31] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C.L. Zitnick, P. Dollár, Microsoft COCO: common objects in context, 2014, http://www.arxiv.org/abs/1405.0312.

[32] E.L. Bird Steven, E. Klein, Natural Language Processing with Python, O'Reilly Media Inc., 2009.

[33] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, AAAI, 2017.

[34] S. Xie, R.B. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, CoRR (2016) abs/1611.05431.

[35] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, R. Girshick, Detectron2, 2019, (https://www.github.com/facebookresearch/detectron2).