SciTexNet: dataset generation for document layout analysis based on LaTeX and PDF

Lorenzo Pisaneschi
Department of Information Engineering
University of Florence
Florence, Italy
lorenzo.pisaneschi1@stud.unifi.it

***Abstract*—Document Layout Analysis (DLA) is an important task which is widely used in several applications, such as document retrieval, document and text recognition and contents categorization; it is also useful for document digitalization purposes. The main goal of DLA is to identify all the homogeneous blocks inside any kind of digital document and determine theirs relationship. Recognizing the layout of unstructured digital documents is not easy in parsing documents, but deep neural networks developed for computer vision are proven to be effective using documents converted into images. However, these networks need a very large amount of examples in order to perform a good training and generate a good prediction model. In this paper SciTexNet project is presented. The main goal is to generate large annotated document dataset combining PDF contents and their LaTeX source files, in order to be able to annotate a large amount of scientific papers using their corresponding images. This work is presented by automatically matching the LaTeX representation and the content of PDF articles available on arXiv, a public archive of scientific articles. At first, a set of annotated documents is generated with an annotation algorithm; then, experiments and tests are done training and testing an FR-CNN network; finally, it is shown that a large amount of scientific papers can be processed, annotated and categorized thanks to deep learning neural networks.**

***Index Terms*—automatic annotation, document layout, deep learning, information retrieval, neural networks**

I. Introduction

Portable Document Format (PDF) is the most used format for human documents consumption all over the world. However, PDF files are not meant to be easily read and analyzed by automatic processing algorithms: their structure is really complex [1]. For document digitalization and parsing purposes, OCR (Optical Character Recognition) techniques were used [4] [6]; obviously, it is an obsolete method in exploiting this kind of operation. More recently, it has been proved that deep learning can be used for document layout analysis using corresponding images, for example, an image for each paper page. However, machine learning algorithms have to be fed using very exhaustive input examples, in order to obtain accurate prediction models to be used during the testing phases; on the other hand, we all know that many scientific article templates exist, so the number of examples to be given in input to a deep neural network grows exponentially with the number of templates we would like to automatically annotate [1] [5] [6]. Furthermore, another problem exists: all the documents involved in machine learning algorithms have to be manually annotated to set up a standard ground truth from which the learning process has to start. Manual annotation of thousands of documents is a very slow and expensive process, that discourages us to use deep learning methods in document layout analysis. On the other hand, if an automatic annotation mechanism could be used, deep learning algorithms power could be exploited.

In this paper it is proposed a method to automatically annotate the layout of scientific arXiv PDF articles [7]. The generated annotations then can be used to train a deep neural network, which will finally generate a model to use to build a large scientific papers dataset formed by many different templates. The final dataset generated by the neural network predictions will contains the most important layout analysis elements, like titles, lists, tables and figures. Such a dataset can be used in many downstream applications, which could be simplify. speeded up and improved thanks to this kind of annotation task.

II. Related work

As said before, manual annotated dataset exist for document layout analysis. Some of these dataset are available at ICDAR (International Conference of Document Analysis and Recognition) challenges [8] [9] [10]; The US NIH National Library of Medicine has provided the Medical Article Records Ground truth (MARG), which are obtained from scanned article pages. These dataset are too small to be used with deep learning algorithms; however, an interesting work, PubLayNet, shows a innovative way to do document layout analysis: the main idea is to use two different kind of file, PDF and its XML counterpart, for each single article, as input to an annotation algorithm which is able to generate a precise ground truth to be used in training scope with a neural network; then the mathematical model generated by the network is used to build an enormous dataset formed by documents images, after a prediction evaluation test [11]. PubLayNet is the dataset resulted working with the publicly available on PubMed Central, a free full-text archive of biomedical and life sciences journal literature at the U.S. National Institutes of Health's National Library of Medicine (NIH/NLM).

SciTexNet is inspired by PubLayNet: the aim is to use PDF files and source LaTeX files to create an annotated ground truth and test the prediction capability of computer

vision deep learning neural network. It has been demonstrated that FR-CNN (Faster Region Based Convolutional Neural Network) achieves the best results: for this reason it will be used in training and testing with the documents set provided from arXiv; on the other hand, SciTexNet has been used with 10200 scientific papers written relying on different templates: an important goal of this research is to investigate the possibility to automatically generate annotations for many different-style scientific articles. The number of papers used is not as large as the PubLayNet processed documents, but it is sufficient to perform further insights on this task following the right direction.

III. Automatic annotation algorithm

A. Dataset documents retrieval

All the documents data have been downloaded from arXiv; arXiv is a public archive of any kind of scholarly scientific articles. The main arXiv categories are Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance, Statistics, Electrical Engineering and Systems Science, Economics; they are in turn divided into more specific topics. The fact arXiv covers varied typologies of research fields papers implies that it has a lot of different-style structured paper, in form and items layout. This archive fills perfectly the needs mentioned above: to get a large amount of scientific document, different from each other in layout and covered topics.
The arXiv website provides articles in PDF format and its source files like DVI, PostScript, HTML; actually, the source files are often in LaTeX , a markdown language which is structured in such a way that it is possible to distinguish some of the main text items. Given these premises, besides the fact that in the arXiv archive each paper is uniquely represented by its emission date and a progressive ID updated within the current month (ex: 1908.07836 indicates that this paper was published in August 2019 and its ID is 7836) 10200 scientific articles, regarding the mentioned categories, was downloaded, in their PDF and source LaTeX form, ready to be analyzed. The arXiv download script, available in the source code of this project, takes into account the possibility that a paper source could not be LaTeX : in such a case, the paper is ignored. Finally, a deep learning algorithm should be more able to identify the instances belonging to such these categories during text classification, object recognition, contents understanding and many other identification problems.

B. Layout categories

Once the documents set, PDF and LaTeX source files, is obtained, it is possible to put attention to the elements we want to detect in pages layout. LaTeX is a structured language, strictly correlated to its compiled PDF; since this, it allows to identify the main items of any kind of text:

1) Titles: all LaTeX elements like section, subsection and subsubsection; abstracts are considered titles.
2) Figures: all the LaTeX figure elements
3) Tables: all the LaTeX table elements
4) Lists: all the LaTeX list elements, like itemize and enumerate
5) Text: all the text which is not included in the above categories

These text items categories have been chosen for the studies here presented because differences between them are clear at high and low levels of comprehension. Titles, figures, lists and tables are objects commonly found in all texts, from scientific articles to books and newspapers, so their identification is important for many applications for which recognize learnable patterns is crucial, considering many different papers.

TABLE I: Categories of document layout included in SciTexNet.

Document layout category	LaTeX category
Text	author, author affiliation; paper information; copyright information; abstract; paragraph in main text, footnote, and appendix; figure & table caption; table footnote
Title	article title, subsection, subsubsection, title[a]
List	list and their sub-lists[b]
Table	main body of table
Figure	main body of figure and sub-figures[b]

[a] It is verify during the automatic annotation algorithm if the first two examined article lines contain titles.
[c] Nested lists (i.e., child list under an item of parent list) are annotated separately, then they will be separated during the FR-CNN learning and testing processes;
[d] Each figure is annotated separately, though it is a sub-figures; like in the sub-lists case, figures and sub-figures will be separated during the FR-CNN elaborations.

C. Annotation algorithms: two engines for the same article

The automatic annotation algorithm matches the elements identified inside the LaTeX source files and the related PDF itself; to do this operation, the annotation algorithm has to be divided into two separated and consecutive sub-operation: a first phase, where the source file parsing is done and a second phase, where PDF file is processed; in particular, in this last mentioned PDF parsing operation, the matching between the detected source file instances and the detected PDF file instances is done and the results are stored in order to be used as ground truth later by the FR-CNN network. In the source files parsing operation, it is important to specify that not all the articles have the same LaTeX structure: some papers could have all their source in a unique file, others in many; for this reason, in the first phase of the automatic paper annotation, all the downloaded .tex files are taken in accounting.

1) First phase: LaTeX files parsing: Now is possible to describe more precisely the first annotation phase. Each .tex file which contributes to constitute the source files set for the current analyzed paper is processed independently by the others; on the other hand, obviously all the instances found from this source set will be considered belonging to the same first phase paper analysis result. Files which terminate with the .tex extension can be considered as simple text files; from this consideration, it become easy to implement

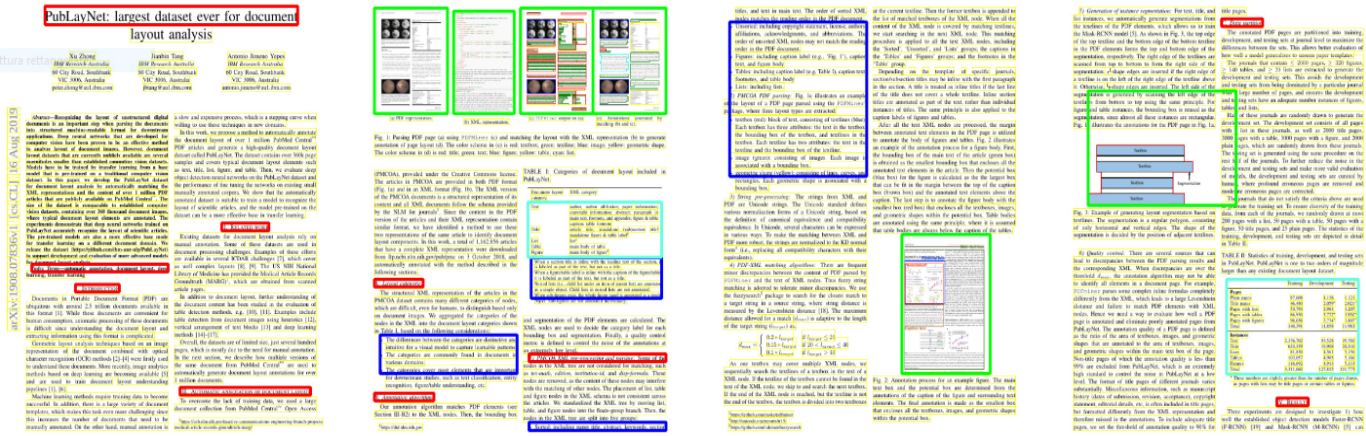

Fig. 1: Example of annotation algorithm result using the PubLayNet paper.

a line-by-line reader and store store the instances belonging to one of the above-mentioned five categories, exploiting the LaTeX keywords. When an instance is identified, its constituting words are refactored: we know that LaTeX is a very powerful markdown language and make available very deep customization directives which are present in analyzed lines during the first phase of automatic annotation algorithm work; but on the other side, during the second phase only words will be detected, without any kind of extra markdown "tags" or "directives". So, for a matching as correct as possible, all these directives belonging to the source files have to be cleaned-up, maintaining only the content, which will be effectively used during the matching task.
Each instance is identified by:

- A category ID: 1 for titles, 2 for figures, 3 for lists and 4 for tables
- The content of the found instance; in case of title, list and tables, is saved also the caption, if present, in order to allow a more precise matching during the second phase of the annotation.
- The position in the instance category list

At the end, a list with all titles, figures, lists, and tables source files identifications is returned and ready to support the PDF parsing phase. Text instances will be intended as all those article items which not belong to mentioned four categories; text items will be labelled during the second phase of annotation algorithm.
Summarizing, the first annotation phase in SciTexNet consists in a line-by-line reading of all the LaTeX files related to the PDF paper currently examined; the process will return a list consisting in all detected titles, lists, tables and figures with captions, an index and contents, that is words that make up the instance as clean as possible.

2) Second phase: PDF file parsing: In the second phase of automatic annotation process complexity increases: three steps are followed and many exceptional cases have to be handled. First of all, images generation: the very first step is the creation of the PNG files which will be used in training and testing. In fact, 90% of all papers will be used for training phase, the remaining 10% for testing. More precisely, the 60% of train images will effectively constitute the training set, the remaining 40% the validation set. Obviously, the annotator will process only the images generated for the training set chosen papers. Going back to the images creation, for each paper page will be generated a corresponding PNG; two Python module are used:

- PyPDF2 module, in order to collect information from PDF files, split them page by page, in brief simply read an article PDF file;
- pdf2image which is the converter. For the conversion, 72 dpi have been used generating in output a 612x792 image for each paper page.

During this process, some errors may occur: remember that PDF file are very complex in structure, so PyPDF2 could not be able to read some of them, or some pages could be very large in size due, for example, to the fact there are high resolution images. Because of these difficulties, papers which cause some errors during the PDF to PNG conversion are not take in consideration, simply because they could be not correctly and completely used and analyzed by any deep learning algorithm developed for computer vision.
Once paper images have been generated and the source files have been parsed, the very PDF parsing phase starts. For this purpose, another Python module is used, pdfminer, a powerful tool which allows layout analysis and text items information retrieval (such as categorization and spatial location) given a PDF file. During this step, pdfminer creates particular objects representing each page; these PDF page objects will be effectively used in layout analysis. As said before regarding the complexity and the PDF files reading performed by PyPDF2, also pdfminer could encounter borderline and difficult situations during the parsing process; for this reason, if an error is encountered, then the same "image conversion errors management policy will be used. In general, erroneous papers are not taken in accounting. It is necessary specify that in experiments it was found that less than the 1% of papers

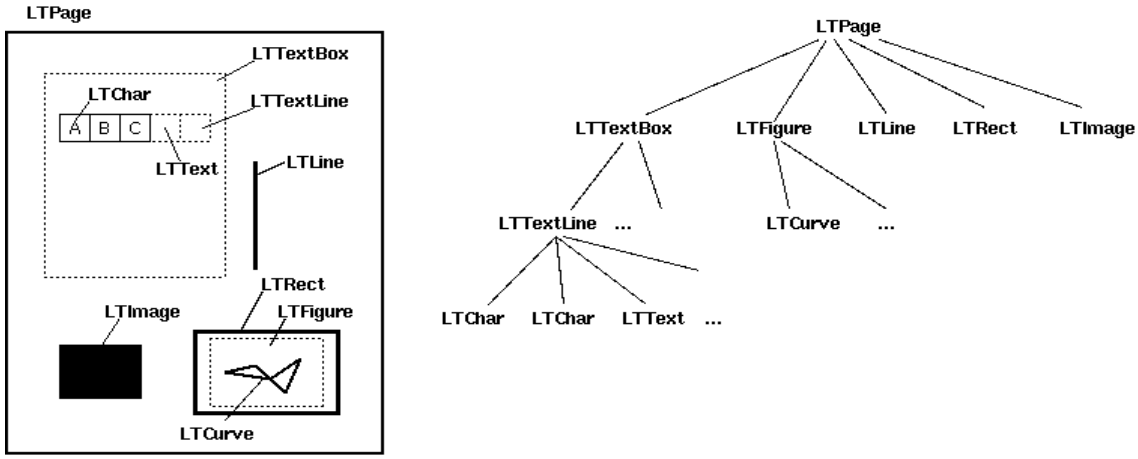

Fig. 2: pdfminer output with its hierarchic objects page disposition.

(on a total 10200 units) were erroneous; in conclusion, few papers are rejected because of parsing or conversion errors.
Pdfminer attempts to convert PDF file structure to text. A page is processed independently by the others by the pdfminer processor which returns a layout object, where all is categorized, stored and most of all properties are accessible and readable. The main pdfminer idea in this text creation is to use heuristics in order to detect similar object belonging to a specific bounding box; each bounding box is uniquely identifiable, and pdfminer specifies also the bounding box rectangle coordinates, using two points: the upper left and lower right ones. Summarizing, pdfminer first collects character into words and text-lines, and locates different-type objects inside the LTPage (the higher level pdfminer item, that represents a PDF page), then groups all the similar instances in bounding boxes and finally groups them hierarchically.

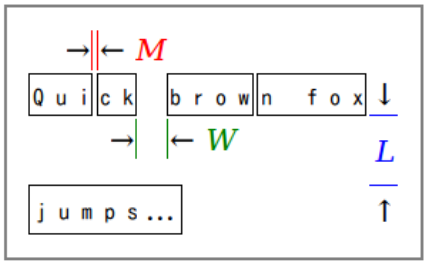

Fig. 3: An image which describes pdfminer parameters used in parsing PDF files. SciTexNet uses char margin (M) = 2.0, word margin (W) = 0.1, line margin = 0.4.

It is clear that pdfminer has to be guided to perform a correct characters, words and boxes grouping; some parameters, which describes the expected words disposition in the PDF document, has to be defined, in order to instruct pdfminer how to analyze all the document elements.
These parameters are the following:

- Char margin: it discriminates if a character belongs to a word or an other;
- Word margin: it indicates the expected distance between two words;
- Line margin: it discriminates if a words group belong to a box rather than another;

These parameters are useful to decide how to parse a document and to detect the correct document objects flow; obviously, the chosen parameters should perform better in a specific article rather than an other because of the different layout; SciTexNet elaborates many different styled documents, and the chosen pdfminer parameters work fine.

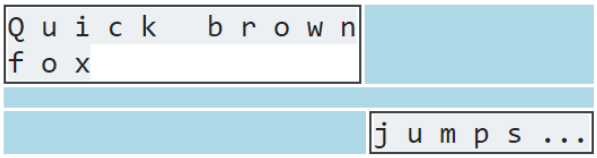

Fig. 4: The bounding box choose by pdfminer depends by the overlap between different detected boxes.

In the end, a box will consists in a list of lines, which themselves are formed by a list of words, formed by characters; pdfminer is strictly hierarchic also in text processing. Finally, text boxes are grouped in a meaningful way; this final step strictly depends by the boxes closeness all around the document: the closeness between boxes is determined by the area of the bounding box that surrounds both lines, minus the area of the bounding boxes of the individual lines, relating to the line margin parameter.
In a LTPage there are not only text instances: it is possible to find also LTFigures, which are paper images; LTLines, which are straight lines; LTCurves, which are different shape lines. This pdfminer classification is useful in searching matches between LaTeX files parsing and PDF file parsing outputs; in particular, with SciTexNet LTFigures and LTLines have been used, as it is explained below.
The algorithm proceeds paying attention to LTTextBoxHorizontal (instantiated as described below), LTLines and LTFigures. When a LTTextBoxHorizontal is

encountered, it is verified firstly if it is a title, analyzing the first two line of the box, since titles are always at the beginning; if the LTTextBoxHorizontal is not a title, then is verified if it is a list or a table. All these checks imply the comparison between the source file parsing output list, where all is stored; this fact demonstrates the importance of the LaTeX in SciTexNet annotation process. When there is a title match, the bounding box coordinates of the matched instance are saved; list match are more complicated: all the found list item are stored, obviously after a comparison with the first phase parsing output, but the bounding box list coordinates, like all the other kind of instances, are identified by two points, which are the upper left of the first item and the lower right of the last item which belong to the examined list. Regarding tables, they are not easy to find within a PDF files: tools, software modules and libraries exist, but for different reason they are not as precise as SciTexNet needs. For this reason, tables are matched using all the possible knowledge and information derived by the the two annotation phases: from the first, caption, if exists, and tables content; from the second, besides text matching, horizontal LTLines are exploited. In fact, tables are always characterized by horizontal lines, for this reason in tables annotation text matching (comparing source and PDF files) and the presence of LTLines objects in the LTPage are combined for tables recognition. In this case, as instance coordinates the upper left and the lower right points between lines and text are chosen. Finally, the pdfminer LTFigure objects are compared with the image items found by the annotation first phase for a stronger and safer correspondence. The similarity between source and PDF files instances is calculated using a sequence matcher, which has to compare different phases objects; if the matcher find a similarity greater than the 75%, the result is a match. The 25% error tolerance derives from the fact that the parsed LaTeX instances could not be completely cleaned by the LaTeX directives, with the result that false negatives matches could considerably increase. It is also to be considered that pdfminer and image Cartesian reference systems are different: pdfminer uses increasing y-coordinates, while an image reference system uses decreasing y-coordinates, since the images origin point is located upper left. In order to obtain consistent coordinates which will identify each instance in a paper page, a coordinates conversion is done.

Summarizing, during the second phase of automatic annotation algorithm, pdfminer is used in parsing PDF file; matching text instances are found using a sequence matcher, which compares source and PDF files annotations in order to identify a unique instance; tables searching is also supported by pdfminer LTLine objects, not only by LTTextBoxHorizontal, and source files images are validated by pdfminer LTFigures; each matched instance is characterized by a bounding box defined by two point coordinates, accurately converted from the document reference system to the reference system of the PNG page file which will be effectively used with the FR-CNN. The annotation algorithm presented in this section iterates over each page of each training set paper downloaded from arXiv; all the final found instances are saved in a txt files, which will be used as input to the FR-CNN; each instance is characterized by the path to the PNG page file to which it belongs, the bounding box coordinates (stored as two couple of x and y values) and the instance category.

IV. FR-CNN: SciTexNet deep neural network

At this point, we have realized the annotated set, that will be used for training the neural network, and we have separated the set of images which will be used for testing. In this section the FR-CNN (Faster Region Convolutional Neural Network) is presented [12] [13]. It is important to know how SciTexNet tries to generate a huge documents dataset using layout analysis and computer vision deep learning algorithms. SciTexNet must do two different tasks:

- Detect objects (titles, lists, figures and tables) inside a paper page, that is individuate instances bounding boxes coordinates;
- Classify detected objects, that is labelled those instances with the predicted class name;

FR-CNN is right for these addressed problems; effectively, FR-CNN is formed by two separated layers, one for each main task: region proposal network (RPN) for generating region proposals and features, and another layer which detects objects from RPN output proposals. RPN allows to share results with all the other layers and obtain a general improvement in all the prediction process; moreover, RPN is one of the principal responsible of the higher performances (especially in time consumption) among other well known detection algorithms.

A. RPN

First of all, in SciTexNet FR-CNN training process, each image (is that, each page of each training set paper) is the input, which is resized and being fed into the backbone convolutional neural network, a ResNet50, 50 layers deep. The output features map is used by the network to learn whether an object is present in the input image; to do that, a set of "anchors" (boxes) are placed on the input image for each location on the output features map; usually, nine anchors are used, with different sizes, shapes, scales and aspect ratios. Now, the network moves over the features map, checks if there are objects and redefines the anchors in order to obtain the object proposals. Obviously, foreground and background is a very first classification analyzing images using FR-CNN. During training task, anchors are selected as "positives" if they results assimilable to a ground truth bounding box page, using IoU (intersection over union) metric; many ground truth boxes could assign a positive result to many different predicted anchors. The training loss function is given by:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{1}$$

In the above equation, *i* represents the index of an anchor in

a page mini-batch of anchors. $L_{cls}(p_i, p_i^*)$ is the log loss over object-not object classes; p_i^* is the ground truth label, while p_i is the classification score for the i-th anchor; the $L_{reg}(t_i, t_i^*)$ is considered for the i-th anchor which contains objects (i.e, $p_i = 1$). The t_i is the prediction score of the regression layer, consisting in four variables which defines detected box width and coordinates; t_i^* is the regression target.

B. Object detection and classification

RPN output is simply a set of anchors with different features. Next, all these proposals passes through the ROI pooling layer, which allows to process proposals which are different in size; in fact, in many pooling algorithms (such as max pooling) there is the fixed size bound for each proposal. Furthermore, ROI pooling produces the fixed-size feature maps from non-uniform inputs by doing max-pooling on the inputs. In substance, ROI pooling layer accepts two inputs (the feature map and ROIs) and does max pooling, a sample-based discretization process, in order to generate fixed feature output maps. Once the features have been extracted, they are passed into classification and regression branches; the classification branch consists in a softmax layer which will output the probability of a proposal belonging to each class, while the regression outputs error coefficient is used to make objects predicted bounding box more precise.

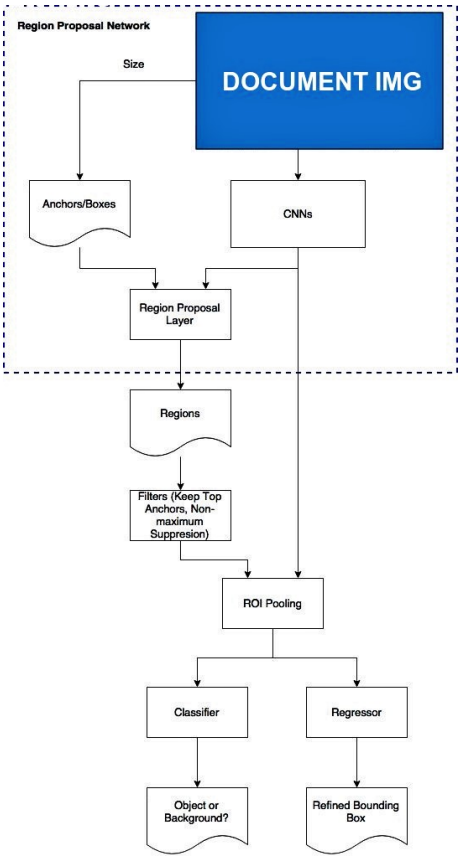

Fig. 5: Faster Region Convolutional Neural Network architecture.

The training process will generate a mathematical model that contains all the weights obtained by the training set analysis. This model will be used during the testing phase with the test set images; the testing results will be evaluated and the obtained evaluation will indicates how good detection and classification have been done with FR-CNN and the SciTexNet annotated papers.

V. Experiments and Results

In this section experiments and their results are presented. The experiments intend to evaluate the accuracy of SciTexNet annotations on a images test set, using the prediction model obtained by the relative training phase and investigate if further optimizations are possible, following the main idea presented in this work: the generation of an huge amount of annotated papers combining the information retrieved from a PDF file and its source LaTeX files in order to train an FR-CNN. Two different training with the same training set have been made: the first training is to simply use the images generated in the steps above mentioned; the second type of training is to use the images that are used also in the training type one, but with data augmentation, which provides a rotation of 90° for each image in order to enrich the information contained in the dataset. Therefore, two different models have been generated, and two different test have been ran, one for each model, with the aim of verify if data augmentation could improves the predictions.

A. Data partition and experiments set-up

Experiments were performed using 10.200 downloaded papers from arXiv; 9180 papers belong to training set, 1.020 to the testing set. 204.658 is the number of downloaded source files, used in the first part of this project, the ground truth generation; 187.485 PNG files were generated (one for each papers page) and 466.452 were annotated in the ground truth: 171.186 titles, 217.493 figures, 40.083 tables, 37.690 lists. The PNG directory size is 25 gigabyte: 23 gigabyte for training images, 2 gigabyte for testing mages. All the procedures were ran on a Linux machine, with a Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz processor, a GeForce GTX 970 GPU, 14 GB RAM. The computational cost, especially in time, was very expensive: 28 hours are needed for ground truth generation; 1 week for training, 24 hours for testing. In the training phase, 150 epochs were performed, with 1000 epoch length.

TABLE II: Instances overview in training and testing sets.

	Training	Testing annotations	Testing prediction
Title	171,186	18.214	40.803
Figures	217.493	15.337	15.722
Tables	40.083	6.845	6.845
Lists	37.690	3.741	1
Total	466.452	33.551	63.371

In table II it is clear that prediction instances are more than their counterpart in ground truth; regardless the fact that in evaluation testing results overlapping instances of same class

are considered as one, this is because training needs more examples to process; the contribution given to lists is surely insufficient: this will weigh on the final evaluation, but we can concentrate on the other categories. Lists have not been predicted precisely for two main reasons: once again, the small number of paper used and probably their similarity with simple text. However, in the following we will examine deeper what performance level has been reached by FR-CNN.

B. Results

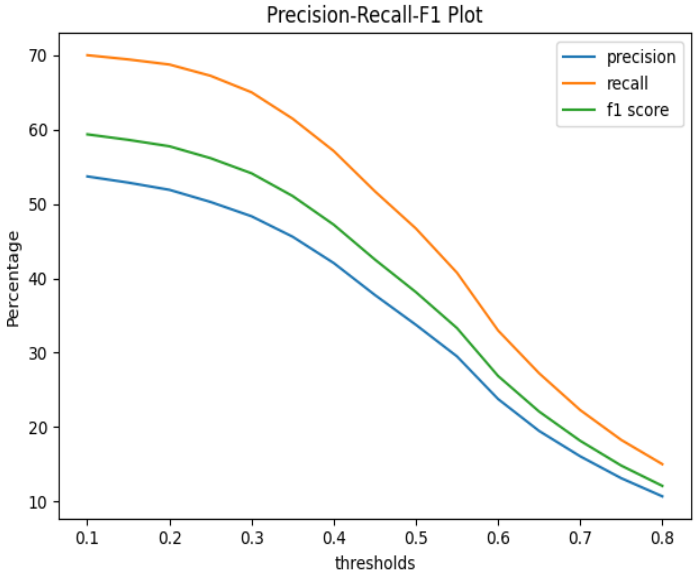

Fig. 6: Precision-Recall-F1 score plot for the first training (without data augmentation).

The testing phase produces some analysis, which shows how the network worked. The used metric is the Intersection Over Union (or Jaccard index), which measures the accuracy of an object detector. The idea is to calculate how good is a detected instance. In testing phase, all the detected instances and their information are saved in a txt file which is comparable with another txt file where all the respective ground truth annotations are saved. The evaluation on the detected instances is hierarchical: the IoU score is computed for each matching instance; each score at instance level contributes to the overall analysis on how a page has been processed by the FR-CNN, and in turn each page score contributes finally to the overall paper score. Obviously, IoU scores are used to calculate more accurate and general metrics: precision, recall and consequently the F1-score, varying a given threshold which indicates what kind of IoU error is tolerated in order to classify a predicted instance as a true positive or a false positives; regarding this, clearly not predicted instances are considered false negatives, on the other hand the instances which have been wrongly predicted (i.e. those who are not annotated in the ground truth but have been labelled by FR-CNN) are considered false positives.

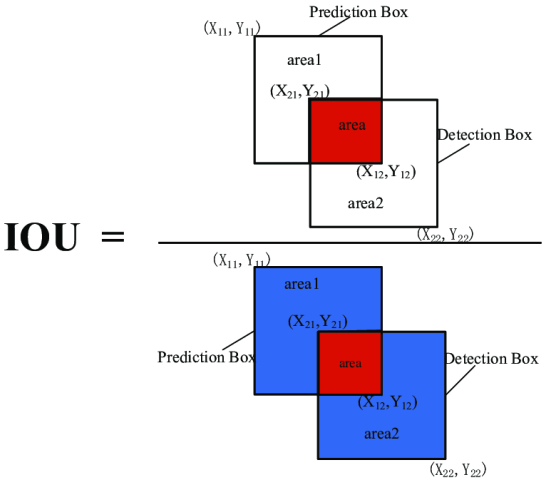

Fig. 7: Intersection Over Union metric: a graphical idea

Remembering that:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (2)$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (3)$$

$$F1score = 2 * \frac{Precision * recall}{Precision + Recall} \quad (4)$$

we can conclude that Precision indicates how accurate is a model, calculating how many predicted positive instances are effectively positives, given a greater weight to false positives; furthermore, Recall tells us how many of actual positives was identified correctly, given more importance to false negatives. F1-score is a more accurate metric, because it is simply the harmonic mean of precision and recall. In Fig. 6 it is represented a summarizing of SciTexNet behaviour using the dataset described in this article, varying the threshold tolerance from 10% to 80%, after a ”traditional” training (without data augmentation). Fig. 6 offers a SciTexNet overview, without any class distinction; however, it is possible to view a deeper inside SciTexNet performances, evaluating how the classification computed for each single class instances.

TABLE III: Percentage accuracy for each class instances relating to a given threshold, using the model obtained from a ”simple” training.

	Titles	Figures	Tables	Lists
10%	82.5	50.0	20.8	0.02
20%	81.7	46.6	20.2	0.02
30%	75.9	44.9	19.6	0.02
40%	62.7	43.3	18.9	0.02
50%	45.8	41.4	18.4	0.00
60%	23.0	39.6	17.3	0.00
70%	7.2	35.8	15.2	0.00
80%	1.8	26.7	10.0	0.00

From tables III it is evident that titles is the best matched class; then in order figures, tables and lists; actually, lists are not classified.

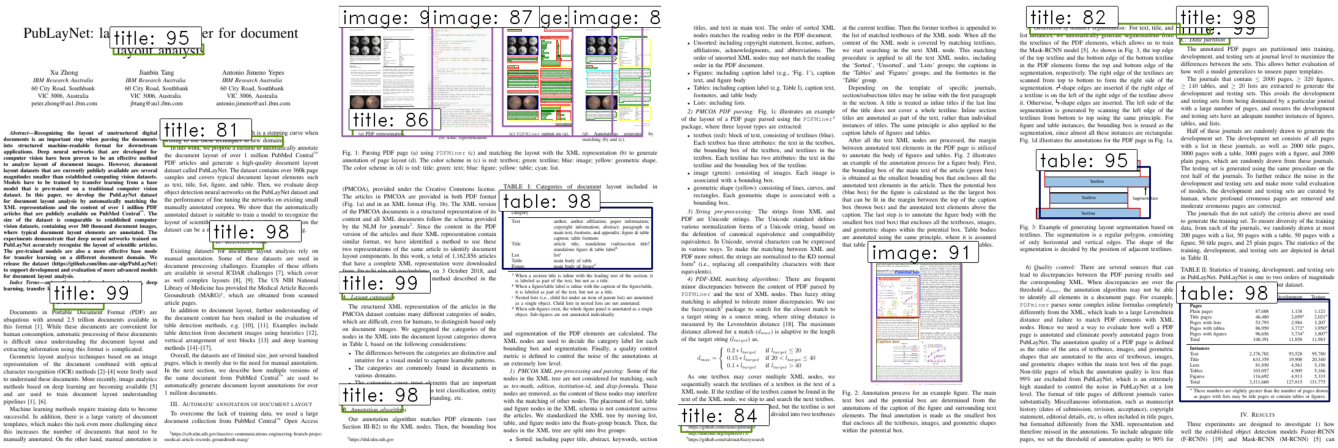

Fig. 8: Example of images automatically annotated by SciTexNet; they are the output of the FR-CNN testing phase.

TABLE IV: Instances overview in training and testing sets with data augmentation.

	Training	Testing annotations	Testing (d. aug.)
Title	171,186	18.214	65.785
Figures	217.493	15.337	162.273
Tables	40.083	6.845	1226
Lists	37.690	3.741	0
Total	466.452	33.551	229.284

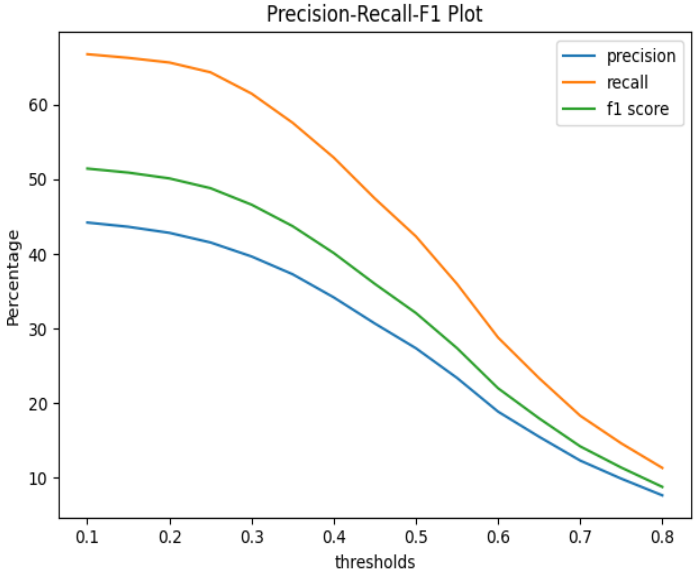

Fig. 9: Precision-Recall-F1 score plot for data augmentation training.

Now it is possible to compare the results from different models, obtained by different training phases: the results of the simple training (without data augmentation), which are indicated below, and the results of the data augmentation training, which are shown in the following. Remember that data augmentation provided in this section make a rotation of 90° for each image belonging to the testing set.

In Fig. 9 it is clear that the training without data augmentation has generated a more accurate model than the one obtained from the data augmented training; to confirm this, in Table IV we see that the data augmented model has identified many more objects that the simple model, resulting in more false positives, which collapse the model precision. On the other hand, the recall is enough in line with the simple trained test result.

TABLE V: Percentage accuracy for each class instances relating to a given threshold, using the model obtained from a data augmentation training.

	Titles	Figures	Tables	Lists
10%	83.6	44.9	11.8	0.0
80%	83.1	41.2	11.4	0.0
20%	76.9	39.2	11.2	0.0
30%	62.1	37.3	10.6	0.0
40%	44.7	35.1	10.3	0.0
50%	21.9	33.4	9.4	0.0
60%	6.9	29.8	8.5	0.00
70%	1.7	20.6	5.3	0.00

In Table V it is possible to see more precisely where the used data augmentation is weak: as we expect, figures and tables are poorly annotated, since the predicted figures and tables instances obtained with data augmentation are many more than we can see in Table III, where the numbers of instances labelled using simple training model are reported, resulting in a ore accurate model; lists are not classified (the used data augmentation is not useful in this aspect); titles prediction seems to be not affected by data augmentation.

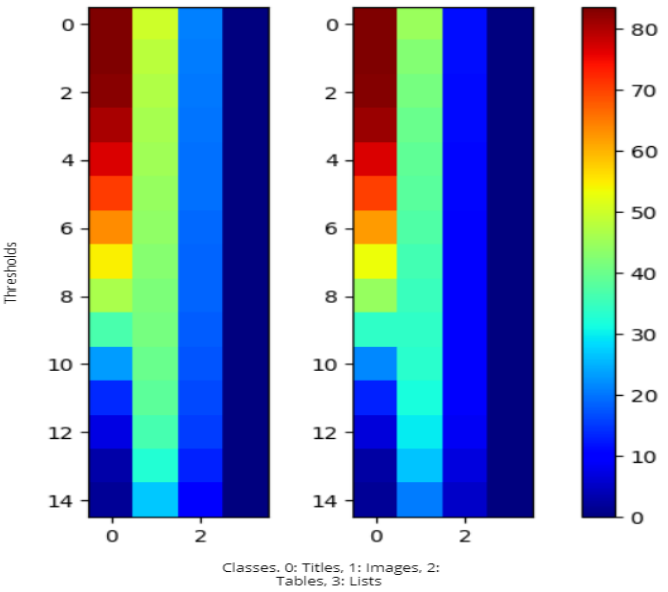

Fig. 10: Heatmaps which show the accuracy for each class varying thresholds considering the mentioned two different prediction models.

VI. Conclusion

SciTexNet can generate an huge annotated documents dataset using the information retrieved from PDF files an their LaTeX source files. In this work all the process has been presented: 10200 papers, which have been divided in training an testing sets, were downloaded from arXiv. All the paper pages have been converted in PNG files, in order to use a FR-CNN, which is a computer vision deep learning algorithm, to automatically annotate the testing set paper pages. The training set has been obtained thanks to an automatic annotation algorithm, which in a first phase, parses all the source files of a scientific article; in a second phase, it parses the related PDF and combines the two phases results in final label-annotations. These annotations are used in training; the training generates a prediction model, finally used to test the test set images. Some considerations have to be done: the data augmented model resulted in a worst model respect to the simple one; this because probably another type of data augmentation has to be processed to obtain more accurate predictions. Furthermore, concentrating on simple model predictions, the results are encouraging. It has to be clarified that automatic annotation algorithm produces some errors, and it can be improved; on the other hand, the FR-CNN in some cases produces good annotations which result in poor annotations because of the erroneous relative ground truth annotated instances. Furthermore, for hardware reasons, the training has been probably too brief: the training process could be improved increasing the epochs number and the epoch length or even better making also some fine tuning. Finally, it is also to be noticed that the used papers are different in a number of ways; this fact it is important on the one hand, because we want to be able to treat with different objects, on the other hand in this moment it results in worsening results. In the future, it is desirable to start with an homogeneous dataset in document layout structure, to verify if SciTexNet works fine; then, the aim is to improved the automatic annotation algorithm, extend the training phase and do more experiments to obtaine a better prediction model.

References

[1] P. W. J. Staar, M. Dolfi, C. Auer, and C. Bekas, "Corpus conversion service: A machine learning platform to ingest documents at scale," in Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery; Data Mining, ser. KDD '18. New York, NY, USA: ACM, 2018, pp. 774–782. [Online]. Available:http://doi.acm.org/10.1145/3219819.3219834

[2] R. Cattoni, T. Coianiz, S. Messelodi, and C. M. Modena, "Geometric layout analysis techniques for document image understanding: a review," ITC-irst Technical Report, vol. 9703, no. 09, 1998

[3] T. M. Breuel, "Two geometric algorithms for layout analysis," in International workshop on document analysis systems. Springer, 2002, pp. 188–199..

[4] ——, "High performance document layout analysis," in *Proceedings of the Symposium on Document Image Understanding Technology*, 2003, pp.209–218.

[5] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[6] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, vol. 1. IEEE, 2017, pp. 1162–1167.

[7] [Online]. Available: https://arxiv.org/

[8] Antonacopoulos, D. Bridson, C. Papadopoulos, and S. Pletschacher, "A realistic dataset for performance evaluation of document layout analysis," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 296–300.

[9] Clausner, C. Papadopoulos, S. Pletschacher, and A. Antonacopoulos, "The enp image and ground truth dataset of historical newspapers," in *Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*. IEEE, 2015, pp. 931–935.

[10] Clausner, A. Antonacopoulos, and S. Pletschacher, "Icdar2017 competition on recognition of documents with complex layouts-rdcl2017," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, vol. 1. IEEE, 2017, pp. 1404–1410.

[11] Zhong Xu, Tang Jianbin, Jimeno-Yepes Antonio. (2019). PubLayNet: largest dataset ever for document layout analysis.

[12] [Online]. Available: https://github.com/kbardool/keras-frcnn

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.