Parsing Electronic Theses and Dissertations Using Object Detection

Aman Ahuja Alan Devera Edward A. Fox
Department of Computer Science
Virginia Tech, Blacksburg, VA
{aahuja, alandevera1, fox}@vt.edu

Abstract

Electronic theses and dissertations (ETDs) contain valuable knowledge that can be useful for a wide range of purposes. To effectively utilize the knowledge contained in ETDs for downstream tasks such as search and retrieval, question-answering, and summarization, the data first needs to be parsed and stored in a format such as XML. However, since most of the ETDs available on the web are PDF documents, parsing them to make their data useful for downstream tasks is a challenge. In this work, we propose a dataset and a framework to help with parsing long scholarly documents such as ETDs. We take the Object Detection approach for document parsing. We first introduce a set of *objects* that are important elements of an ETD, along with a new dataset ETD-OD that consists of over 25K page images originating from 200 ETDs with bounding boxes around each of the objects. We also propose a framework that utilizes this dataset for converting ETDs to XML, which can further be used for ETD-related downstream tasks. Our code and pre-trained models are available at: https://github.com/Opening-ETDs/ETD-OD.

1 Introduction

Long scholarly documents like Electronic Theses and Dissertations (ETDs) contain a vast amount of information which can be of immense value to the scholarly community. Millions of ETDs are now publicly available on the web, and can serve as a rich source of scholarly information. However, relative to the large amount of information in such documents, a significant portion remains untapped.

Part of the problem is that these documents are often long and filled with highly specialized details. This makes it difficult for many users to understand the information contained in ETDs. In recent years, advances have been been made in NLP-based techniques such as question-answering and text summarization, which might be incorporated to make

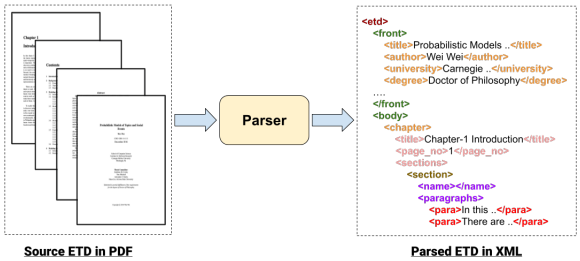


Figure 1: Illustration of the proposed framework. We take a source document in PDF as the input, and generate a parsed version in a structured format like XML.

ETDs more accessible. However, a majority of these documents exist as PDF files. While some tools can work with these files, the results we have observed have been poor; other tools require data in a structured format such as XML. This leads to the research question: *Is there a way to identify, parse, and extract the information from a PDF version of an ETD so that it is more accessible to a wider audience?*

Many research challenges arise when transforming ETDs from PDF to other formats. These scholarly documents do not have a standard layout. Different institutions have their own layouts and formats, making rule-based parsing methods difficult to apply. Moreover, the structure and organization of elements present in documents varies by domain and organization. For instance, documents from domains such as mathematics often contain equations, while documents from computer science frequently contain algorithms. Hence, there is a need to develop machine-learning based document-parsing methods that can generalize to documents with different layouts and across domains.

In recent years, with the advances in the field of computer vision, several methods have been proposed for extracting important elements from documents. Some of these approaches perform document layout analysis using object detection models (Girshick, 2015; Ren et al., 2015). However,

Proceedings of the 1st Workshop on Information Extraction from Scientific Publications, pages 121–130

many of these works and the associated datasets are focused on a very narrow set of scholarly document elements. For instance, TableBank (Li et al., 2020a) only contains bounding boxes for tables, while ScanBank (Kahu et al., 2021) focuses on figures (and some tables). More recently, there has been research that primarily focuses on layout analysis of scholarly documents (Zhong et al., 2019; Li et al., 2020b). However, most existing work in the domain of scholarly document understanding focuses on research papers, which differ in many ways from longer documents like theses and dissertations. First, research papers tend to be shorter in length and have a narrower scope. As such, many elements such as *chapters*, *committee*, and *university* that are important in an ETD cannot be found in datasets derived from research papers. Moreover, research papers significantly differ from ETDs in their structure and format. For instance, many research papers are in double-column format, while ETDs typically have a single column, and have bigger font size and spacing. Consequently, existing methods for document understanding for research papers are not easily adapted to ETDs.

In this work, we propose ETD-OD, an object detection based framework to parse long PDF documents such as ETDs. Our approach works on the PDF version of an ETD by first identifying important elements such as figures, tables, captions, paragraphs, delimiters like chapter and section headers, and metadata such as title, author name, etc. This is done using object detection models such as Faster-RCNN (Ren et al., 2015) or YOLOv7 (Wang et al., 2022) on individual page images. For textual elements such as paragraphs and captions, the textual content is further extracted using PDF-based tools such as pymupdf, or optical character recognition (OCR). Finally, we put together all these elements in a structured XML format. We also introduce a new object detection dataset that contains over 25K page images originating from 200K ETDs, consisting of elements that commonly occur in ETDs, that can be important sources of information. An XML schema to support parsing with such objects is also introduced.

2 Related Work

2.1 Methods

Early works in the domain of document layout understanding used rule-based approaches (Lebourgeois et al., 1992; Ha et al., 1995). Other approaches, e.g., GROBID (Lopez and et al., 2008–2022) and CERMINE (Tkaczyk et al., 2015) designed for parsing scientific documents primarily focused on short documents such as research papers, and use an ensemble of sequence labeling methods for document parsing. With the advent of deep-learning based object detection methods such as Fast-RCNN (Girshick, 2015), Faster-RCNN (Ren et al., 2015), and YOLO (Redmon and Farhadi, 2018; Wang et al., 2022), document layout analysis based on object detection has been proposed. LayoutParser (Shen et al., 2021) uses object detection models that have been pre-trained on different object detection datasets to support layout understanding. However, since it primarily uses research-paper based datasets, it doesn't perform well on ETDs. Moreover, the number of object types it supports is very limited. More recently, layout-based language models (Xu et al., 2020, 2021; Huang et al., 2022) have been proposed. This line of work uses a multimodal architecture, i.e., a combination of visual and textual features, to pre-train the model on a large corpus of unlabeled data consisting of document images and their corresponding text. Although these models can then be fine-tuned on other downstream tasks such as object detection, they still require domain-specific annotated data for fine-tuning. Recently, to make the documents more accessible, services such as SciA11y (Wang et al., 2021) have been developed. However, their scope is limited to research papers, rather than long documents such as books and ETDs.

2.2 Datasets

With the growing interest in using object detection based methods for document layout analysis, several datasets have been introduced. Many of these datasets focus on specific object types. For instance, TableBank (Li et al., 2020a), ScanBank (Kahu et al., 2021), and MFD (Anitei et al., 2021) consist of tables, figures, and equations, respectively. Several datasets that consist of a diverse set of objects have also been introduced. HJDataset (Shen et al., 2020) consists of historical Japanese documents. PRImA (Antonacopoulos et al., 2009) consists of document images from magazines and research papers. PubLayNet (Zhong et al., 2019) is based on PDF articles from PubMed Central. The number of different objects, however, is limited in these datasets. DocBank (Li et al., 2020b) is a large

dataset that consists of a diverse set of objects from research papers. But given the differences between research papers and long documents such as ETDs, models trained on DocBank do not generalize well on ETDs.

3 ETD Elements

Historically, ETDs do not conform to a universally accepted format, since different colleges and universities have their specific standards and requirements for ETDs. In this section we discuss the elements that are typically found in ETDs and would be important to extract for further analysis and downstream tasks. This list was curated after extensive discussions with digital librarians and researchers. We broadly categorize the different elements of ETDs into the following categories.

3.1 Metadata

The metadata consists of elements that contain unique identifiable information about an ETD, including information found on the front page. Key metadata elements are:

- **Title:** The main title of the document.
- **Author:** Name of the document author.
- **Date:** Date (or month/year) when the document was published, or of the final research defense.
- **University:** University/institution of the author.
- **Committee:** Committee that approved the document, e.g., the student's graduate committee.
- **Degree:** Degree (e.g., Master of Science, Doctor of Philosophy) being earned.

3.2 Abstract

The abstract is an important element of an ETD, as it contains a summary of the work, typically about a page long. Its elements include:

- **Abstract Heading:** Since some ETDs contain multiple abstracts, such as a technical abstract and general audience abstract, or an abstract in English as well as the original language, extracting the abstract heading makes it easier to segment, and could be helpful in categorizing the abstract by audience type.
- **Abstract Text:** The actual text of the abstract.

3.3 List of Contents

The list of contents (also referred to as table of contents) of an ETD determines where different components are located based on their page numbers. This helps with accurately mapping the chapters and sections, as well as figures and tables, since they are generally included in the list of contents. This subcategory includes the following elements:

- **List of Contents Heading:** This helps identify the specific type of list (e.g., list of chapters/sections, list of figures, list of tables).
- **List of Contents Text:** This is the actual list of entries for this type of content.

3.4 Main Content

Chapters are one of the most important components of an ETD, as they contain detailed information about the research described in the document. This subcategory consists of elements that can typically be found in the chapters of an ETD.

- **Chapter Title:** The title of the chapter.
- **Section:** Quite often, chapters themselves can be long. It may be desirable to have further delimiters such as sectional headers. Hence, we include the section names which can be used for further splitting of the document.
- **Paragraph:** The main textual content of the ETD.
- **Figure:** This includes figures, charts, and other visual illustrations included in the document.
- **Figure Caption:** The text caption that describes the figure.
- **Table:** The table element category.
- **Table Caption:** The text caption that describes the table.
- **Equation:** Mathematical equation/formula.
- **Equation Number:** Quite often, equations are numbered, which can be helpful in linking them to the list of equations that may be included in the document.
- **Algorithm:** Algorithm description, e.g., as pseudo-code.
- **Footnote:** We separate footnotes from regular paragraphs, as they typically provide auxiliary information which might be undesirable in many downstream tasks, such as summary generation.
- **Page Number:** Page numbers, which could be helpful in cross-referencing pages and the objects contained therein, to the list of contents.

3.5 Bibliography

We also include bibliographic elements in the list of objects. They are described below:

- **Reference Heading:** The header that indicates start of the references list.
- **Reference Text:** The actual list of references cited in the document.

In our dataset, we regard appendices as chapters, since they contain many elements that are found in the main chapters. They can however, be easily differentiated from main chapters based on the title.

4 Dataset

4.1 Dataset Source

The ETD-OD dataset consists of 25K page images from 200 theses and dissertations. These documents were downloaded from publicly accessible institutional repositories, and were randomly sampled with regards to degree, domain, and institution. Since object detection requires images as the input data, the documents were split into page images using the `pdf2image`[1] Python library. These images were then used for annotation.

4.2 Annotation

We use Roboflow[2] for annotating the page images in our dataset. The annotation was done by a group of 6 undergraduate students (Zhu et al., 2022), each of whom was a computer science student from junior year or above. Each data sample was further validated for correctness by two graduate students.

4.3 Dataset Statistics

Table 1 shows the detailed statistics for different object categories in our dataset. The dataset consists of ∼25K page images and ∼100K bounding boxes spanning across different object categories. Owing to the variation in the frequency of occurrence of various object categories in documents, some categories have many more samples as compared to others. Elements such as paragraphs can be found on most pages, and hence, it is the dominant category in our dataset. 80% of the images and their corresponding objects were used for training, while the remaining 20% were used as the validation set.

5 Proposed Framework

We now introduce the proposed framework for transforming long PDF documents into structured XML format. The architecture of our framework is illustrated in Figure 2. The different modules shown can broadly be divided into the following three categories.

[1] https://pypi.org/project/pdf2image/
[2] https://roboflow.com/

Category	# Instances
Title	439
Author	404
Date	338
University	309
Committee	282
Degree	279
Abstract Heading	169
Abstract Text	183
List of Contents Heading	512
List of Contents Text	1059
Chapter Title	2211
Section	9337
Paragraph	30359
Figure	6359
Figure Caption	5722
Table	2654
Table Caption	2213
Equation	5092
Equation Number	3051
Algorithm	96
Footnote	5722
Page Number	24543
Reference Heading	313
Reference Text	2088
Total Objects	99859
Total Images	25073

Table 1: Distribution of different object categories in our dataset. *Note: Some of the documents were accompanied with front matter (metadata) pages that are sometimes generated by the digital libraries. We include annotations for such documents as well, and hence, the number of metadata elements does not exactly match the number of documents.*

5.1 Data and Preprocessing

Since our framework is primarily built for parsing long scholarly documents, it takes the PDF version of the document as input. The input file is converted to individual page images (.jpg format) using Python-based PDF libraries such as `pdf2image`. Next, the page images are individually fed to the Element Extraction module for further processing.

5.2 Element Extraction using Object Detection

This module forms the backbone of our system. It takes the individual page images as input, and uses an object detection model such as Faster-RCNN or YOLO for object detection. These models are first

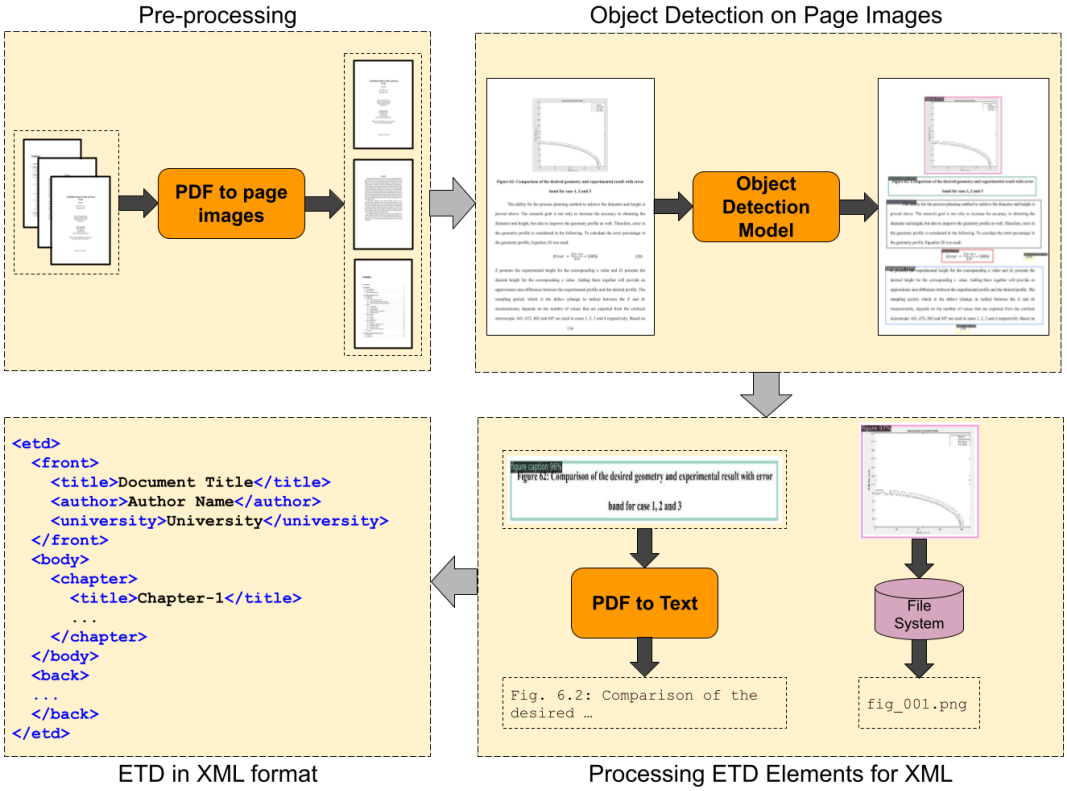


Figure 2: Architecture of the proposed PDF to XML parsing framework.

pre-trained on the dataset described in Section 4. The specific details about training object detection models are included in later sections of this paper. While using the object detection models as a part of this module, only inference is performed, and no updates are made to the model parameters. The output of object detection will be a list of elements, where each element contains information about the bounding boxes such as the coordinates, along with the category labels. This process is repeated for all of the pages in the document, and finally, a list of pages accompanied by their respective elements is populated.

In some instances, the object detected by the model is classified as one belonging to a different, yet similar category. In such cases, we use certain post-processing rules to correct the predictions. For example, *abstract heading* being mis-classified as chapter heading is one of the common errors, since both of these elements are often found in bigger font size at the beginning of a page. This can, however, be corrected by enforcing a constraint such as: a chapter heading in the first 10 pages with matching keyword “abstract" will be the abstract heading. We use a set of such rules for different object types to correct mis-classifications before the objects are sent to the XML module.

5.3 Structuring Objects into XML

After extracting all of the elements for all of the pages in the document, we generate the XML representation of the document. We regard the objects as broadly belonging to two types. The first type includes **image-based** objects such as figures, tables, algorithms, and equations, that need to be stored on the file system as an image. We regard tables as image-based objects even though they might contain text, since further extraction of information in structured format from tables is beyond the scope of this work. The second type of object includes **text-based** elements such as paragraphs, titles, etc., which need further processing to be converted to plain text. We regard all object categories excluding the image-based ones as textual elements.

For converting text-based objects to plain text, we use off-the-shelf tools and libraries. Some PDF documents are born-digital, where the text can be easily extracted using Python libraries such as `pymupdf`[3] based on page ID and bounding box coordinates. For scanned documents we use op-

[3] https://pymupdf.readthedocs.io/en/latest/

tical character recognition (OCR) tools such as `pytesseract`[4].

```
<etd>
 <front>
  <title>Document Title</title>
  <author>Author Name</author>
  <university>University</university>
  <degree>Degree Type</degree>
  <committee>Committee</committee>
  <date>Date or Month/Year</date>
  <abs_heading>Abstract</abs_heading>
  <abs_text>In this..</abs_text>
  <loc_heading>Table of..</loc_heading>
  <loc_text>1. Intro ...</loc_text>
 </front>
 <body>
  <chapter>
   <title>Chapter-1..</title>
   <page_no>1</page_no>
   <sections>
    <section>
     <name>1.1..</name>
     <paragraphs>
      <para>In this...</para>
      <para>Next, we...</para>
     </paragraphs>
     <figures>
      <figure>
       <path>fig_001.png</path>
       <caption>Fig.1...</caption>
      </figure>
     </figures>
     <tables>
      <table>
       <path>tab_001.png</path>
       <caption>Table.1.. </caption>
      </table>
     </tables>
     <equations>
      <equation>
       <path>eqn_001.png</path>
       <eq_no>1</eq_no>
      </equation>
     </equations>
     <algorithms>
      <algorithm>
       <path>alg_001.png</path>
      </algorithm>
     </algorithms>
     <footnotes>
      <footnote>...</footnote>
     </footnotes>
    </section>
   </sections>
  </chapter>
 </body>
 <back>
  <ref_heading>Ref..</ref_heading>
  <ref_text>..</ref_text>
 </back>
</etd>
```

Schema 1: XML Schema for Representing ETDs in Structured Format.

For image-based elements, we include the relative path of the image that is cropped based on the coordinates. Figures and tables are mapped to their respective captions based on proximity. For any figure/table element, the caption object closest to them based on Euclidean distance w.r.t. bounding box coordinates is assumed to be the caption. A similar method is followed to map equations with their equation numbers, with an added constraint that the y-coordinate of the center of the equation number should fall between min and max y-coordinates of the equation object. Finally, all the element values are put into the XML file under their corresponding tags. The detailed XML schema is shown in Schema 1.

6 Object Detection Training

We use the ETD-OD dataset introduced in this paper for training object detection models for our framework. The models currently supported are:

- **Faster-RCNN** (Ren et al., 2015): Faster-RCNN is an object detection model that has two stages. A region proposal network generates regions of interest, which are fed to another network for final detection. We use the version of Faster-RCNN that uses ResNeXt-101 (Xie et al., 2017) as the backbone model.
- **Faster-RCNN pre-trained on DocBank** (Li et al., 2020b): Faster-RCNN (with ResNeXt-101 backbone) pre-trained on DocBank (from the DocBank model zoo) is fine-tuned on ETD-OD. Although DocBank does not include all of the elements found in ETDs, we hypothesize that the scholarly nature of documents used in pre-training should help improve the performance over the vanilla version of the model.
- **YOLOv5** (Jocher et al., 2022): YOLO is a family of single stage object detection models that perform the processes of localization and detection using a single end-to-end network. This improves the speed without any significant drop in performance. These models have shown impressive performance on various datasets.
- **YOLOv7** (Wang et al., 2022): This is the most recent version of YOLO, which has been shown to outperform many object detection models.

Both of the Faster-RCNN models were trained on our dataset for 60K iterations with an inference score threshold of 0.7. The models were based on the implementation included in the open-source detectron2 (Wu et al., 2019) framework. For the DocBank-pretrained version of the model, we used the original set of weights and configurations open-sourced by the authors. Both of the versions of YOLO were based on the open-source implementations, and were trained for 150 epochs.

7 Experiments

In this section, we discuss the results obtained in the experimental analysis of our work.

[4] https://pypi.org/project/pytesseract/

7.1 Evaluation Metrics

For the quantitative evaluation of object detection models, the commonly used metrics are average precision (AP) and mean average precision (mAP). AP is defined as the area under the precision-recall curve for a specific class. mAP is the average of AP values for all object classes. Both of these metrics have different versions based on the overlap threshold (also referred to as *Intersection over Union or IoU*) used for comparing the predicted object against ground truth. For example, in *mAP@0.5*, all of the objects with an intersection of 50% or more with the ground truth will be regarded as correct predictions. Another commonly used version of mAP is *mAP@0.5-0.95*, which is the average mAP over different thresholds, from 0.5 to 0.95 with step 0.05.

7.2 Analysis of Various Object Detection Models trained on ETD-OD

Model	mAP@0.5	mAP@0.5-0.95
Faster-RCNN	39.1	19.6
Faster-RCNN*	76.2	44.0
YOLOv5	83.4	52.1
YOLOv7	<u>85.3</u>	<u>52.7</u>

Table 2: mAP comparison for object detection models on ETD-OD. Faster-RCNN* represents the model pre-trained on DocBank and fine-tuned on ETD-OD. Underlined values indicate best performing models.

Table 2 shows performance of different object detection models on the validation set of our dataset. The following observations can be made from the mAP values shown:

- **Pre-training on scholarly documents improves model performance:** The basic version of Faster-RCNN without any pre-training on scholarly documents has the lowest performance among all the models. The same model, after pre-training on DocBank, and then fine-tuned on the ETD dataset, gives much better performance. Since DocBank also consists of scholarly documents, albeit of different type, the pre-training process exposes the model to a diverse dataset, which eventually results in better generalization and predictive performance.
- **YOLO outperforms Faster-RCNN on ETD dataset:** YOLO models belong to the class of single stage detectors, which are designed with an emphasis on speed. YOLO typically performs worse than Faster-RCNN in scenarios where the objects are smaller or multiple objects are close to each other. However, in case of documents, most objects are typically of large size and have minimal overlap with each other due to white spaces and line breaks around objects (such as between a header and paragraph). Hence, it outperforms Faster-RCNN on the ETD dataset.

7.3 Analysis of Detection Performance on Different Object Categories

Category	AP@0.5	Category	AP@0.5
Title	92.5	Paragraph	97.4
Author	89.5	Figure	98.4
Date	68.3	Fig. Caption	95.4
University	91.1	Table	94.7
Committee	96.5	Tab. Caption	89.8
Degree	68.3	Equation	72.6
Abs. Heading	94.2	Eqn. Number	55.0
Abs. Text	86.7	Algorithm	66.6
LOC Heading	75.5	Footnote	98.9
LOC Text	99.3	Page Number	51.3
Chapter Title	88.8	Ref. Heading	80.7
Section	90.9	Ref. Text	99.3

Table 3: AP@0.5 values for different object categories for YOLOv7 (Abs. = Abstract, LOC = List of Contents).

In Table 3, we show the performance of the best performing model (YOLOv7) on various object categories in our dataset. The lower performance of certain categories can generally be attributed to two reasons:

- **Limited Number of Training Samples:** Elements such as degree, date, and algorithm have very few instances in our dataset. As such, the performance on these classes is lower.
- **Smaller Object Sizes:** Elements such as page number and equation number tend to be of smaller size as compared to other elements. Since object detection models tend to struggle with localization of smaller objects, performance of such classes is impacted.

7.4 Comparison against Other Layout Detection Datasets

To evaluate how the performance of similar models varies across different datasets from the document layout analysis domain on layout analysis of ETDs, we compare the per class AP values for object categories supported by the DocBank dataset.

Categories	DocBank only	ETD-OD only	DocBank ETD-OD
Abstract	2.29	0.0	67.42
Author	5.8	19.27	73.27
Caption	42.72	55.04 / 18.27	97.46 / 89.03
Date	0.0	0.0	76.28
Equation	8.13	62.28	76.19
Figure	72.44	78.21	95.01
Footer	69.38	85.03	97.64
List	NA	NA	NA
Paragraph	5.01	80.64	94.34
Reference	2.94	75.43	97.92
Section	19.88	66.99	77.63
Table	33.25	49.04	89.7
Title	1.1	11.3	73.85

Table 4: AP@0.5 values for categories supported by DocBank using Faster-RCNN trained on different datasets and evaluated on validation set of ETD-OD. For *Caption*, we list the Figure Caption / Table Caption values for models trained on ETD-OD.

These results are shown in Table 4. The **DocBank only** is the version of Faster-RCNN pre-trained on DocBank, that was evaluated on the ETD dataset without any fine-tuning. The **ETD only** model has been trained only on the ETD dataset without pre-training on any other scholarly dataset. The **DocBank ETD-OD** was pre-trained on DocBank and then fine-tuned on ETD-OD.

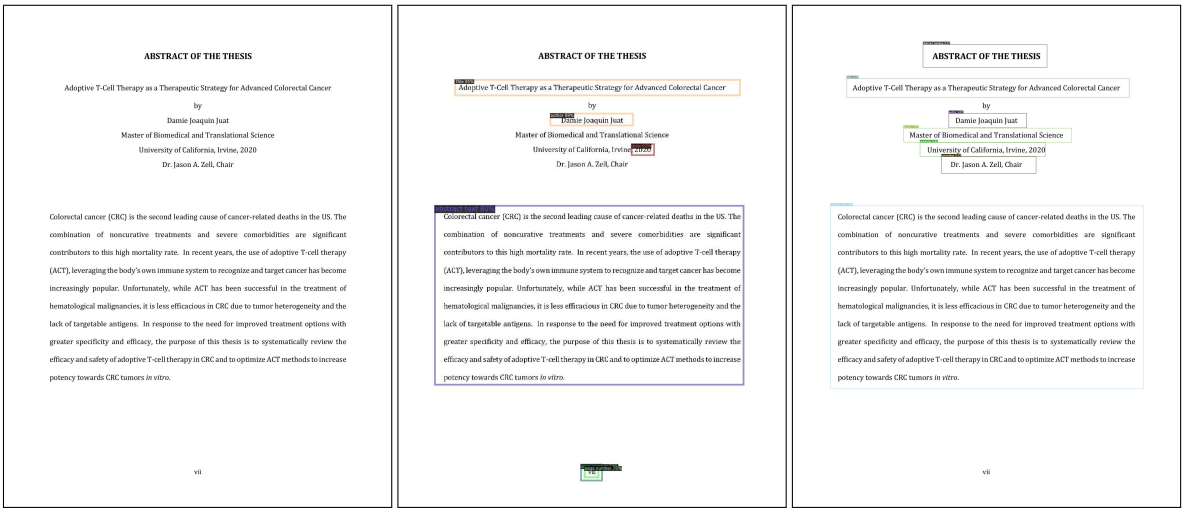
We can see that both of the models that were trained on the ETD dataset perform better than the model that was just trained on the DocBank dataset. This may be due to the fact that DocBank consists of images of research papers, which have different layouts as compared to long documents such as ETDs. On the other hand, since research papers do tend to have some similarities with ETDs, pre-training on DocBank followed by fine-tuning on ETD-OD gives the best results among all three.

7.5 Qualitative Analysis

In Fig. 3, we show example outputs generated by the best performing versions of Faster-RCNN (pre-trained on DocBank, fine-tuned on ETD-OD) and YOLO (v7) models. Faster-RCNN fails to detect many of the metadata elements, which is also reflected by its low mAP values. YOLOv7 is able to detect most of the elements on the page, with the exception of page number. We conclude that YOLOv7 is the best performing model on the ETD dataset.

8 Conclusion and Future Work

In this work, we presented a new dataset and a framework for parsing long scholarly documents such as ETDs from PDF to structured formats such as XML. We also presented a schema to represent ETDs in XML format, along with extensive experimental evaluation of multiple state-of-the-art models on the newly introduced ETD-OD dataset. In the future, we plan to extend this work to other types of documents, such as old archival documents which typically contain a great amount of noise, and make further improvements to the performance of minority categories.



(a) Original Image (b) Faster-RCNN (DocBank, ETD-OD) (c) YOLOv7

Figure 3: Examples of outputs generated by the Faster-RCNN and YOLOv7 models.

Acknowledgements

This project was made possible in part by the Institute of Museum and Library Services [LG-37-19-0078-19], PI William A. Ingram. The authors are grateful to the University Libraries at Virginia Tech for their generous support of this research. We also thank Kecheng Zhu, Jiangyue Li, You Peng, Zach Gager, and Shelby Neal for their help in dataset curation.

References

Dan Anitei, Joan Andreu Sánchez, José Manuel Fuentes, Roberto Paredes, and José Miguel Benedí. 2021. ICDAR 2021 Competition on Mathematical Formula Detection. In *International Conference on Document Analysis and Recognition*, pages 783–795. Springer.

Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. 2009. A realistic dataset for performance evaluation of document layout analysis. In *2009 10th International Conference on Document Analysis and Recognition*, pages 296–300. IEEE.

Ross Girshick. 2015. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448.

Jaekyu Ha, R.M. Haralick, and I.T. Phillips. 1995. Recursive X-Y cut using bounding boxes of connected components. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 2, pages 952–955 vol.2.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. LayoutLMv3: Pre-Training for Document AI with Unified Text and Image Masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091, New York, NY, USA.

Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Kalen Michael, Jiacong Fang, Imyhxy, Lorna, Colin Wong, Zeng Yifu, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Tkianai, YxNONG, Piotr Skalski, Adam Hogan, Max Strobel, Mrinal Jain, Lorenzo Mammana, and Xylieong. 2022. ultralytics/yolov5: v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations. https://zenodo.org/record/7002879.Y1nceezMIiw.

Sampanna Yashwant Kahu, William A. Ingram, Edward A. Fox, and Jian Wu. 2021. ScanBank: A Benchmark Dataset for Figure Extraction from Scanned Electronic Theses and Dissertations. In *2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 180–191. IEEE Computer Society.

Frank Lebourgeois, Zbigniew Bublinski, and Hubert Emptoz. 1992. A fast and efficient method for extracting text paragraphs and graphics from unconstrained documents. In *11th IAPR International Conference on Pattern Recognition. Vol. II. Conference B: Pattern Recognition Methodology and Systems*, volume 1, pages 272–273. IEEE Computer Society.

Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, Ming Zhou, and Zhoujun Li. 2020a. TableBank: Table Benchmark for Image-Based Table Detection and Recognition. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1918–1925.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. 2020b. DocBank: A Benchmark Dataset for Document Layout Analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 949–960.

Patrice Lopez and et al. 2008–2022. GROBID. `https://github.com/kermitt2/grobid`.

Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Zejiang Shen, Kaixuan Zhang, and Melissa Dell. 2020. A large dataset of historical Japanese documents with complex layouts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 548–549.

Zejiang Shen, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson, and Weining Li. 2021. LayoutParser: A unified toolkit for deep learning based document image analysis. In *International Conference on Document Analysis and Recognition*, pages 131–146. Springer.

Dominika Tkaczyk, Paweł Szostek, Mateusz Fedoryszak, Piotr Jan Dendek, and Łukasz Bolikowski. 2015. CERMINE: automatic extraction of structured metadata from scientific literature. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(4):317–335.

Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*.

Lucy Lu Wang, Isabel Cachola, Jonathan Bragg, Evie Yu-Yen Cheng, Chelsea Haupt, Matt Latzke, Bailey Kuehl, Madeleine N van Zuylen, Linda Wagner, and Daniel Weld. 2021. SciA11y: Converting Scientific Papers to Accessible HTML. In *The 23rd International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–4.

Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2021. LayoutLMv2: Multi-modal Pre-training for Visually-rich Document Understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591. DOI:10.18653/v1/2021.acl-long.201.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. LayoutLM: Pre-training of Text and Layout for Document Image Understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200. https://www.kdd.org/kdd2020/accepted-papers/view/layoutlm-pre-training-of-text-and-layout-for-document-image-understanding.

Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. 2019. PubLayNet: Largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE. DOI:10.1109/ICDAR.2019.00166.

Kecheng Zhu, Zachary Gager, Shelby Neal, Jiangyue Li, and You Peng. 2022. Object Detection. Virginia Tech CS4624 team term project, http://hdl.handle.net/10919/109979.