CONTENT EXTRACTION AND RECOGNITION IN SCIENTIFIC PUBLICATIONS

A Dissertation

by

ZELUN WANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jyh-Charn (Steve) Liu
Committee Members,	Frank M. Shipman
	Ruihong Huang
	Thomas K. Ferris
Head of Department,	Scott Schaefer

December 2020

Major Subject: Computer Science

ABSTRACT

In the era of digitization, the vast volume of scientific publications has become readily accessible to the readers. With the help of information retrieval technologies, a reader can conveniently locate an existing publication by typing in only a few keywords in a search engine. However, existing technologies cannot be directly applied on the contents of many scientific publications. This is due to the limitations of the PDF format, which is the *de facto* standard format for scientific publications nowadays. Being a layout-based graphical format, PDF unfortunately does not offer easy access to its fine-grained contents.

In this dissertation, we introduce a PDF content extraction and recognition system to bridge the gap. The system focuses on extracting crucial elements from scientific publications including text, math expressions, figures, and tables, which carry most of the technical substances. The proposed system investigated four specific problems. Firstly, we designed a set of algorithms to locate math expressions (ME) in PDF documents, which are often blended into the body text. These algorithms include calculating the ME likelihood of each PDF object based on the PDF font information, and reducing the fragmented detections using a bigram regularization model. In addition to the algorithm development, we also released a new dataset for the research community. Secondly, we proposed a deep neural network to recognize math expressions and produce their markup LaTeX. We used an encoder-decoder neural architecture, while the encoder takes images as inputs, and the decoder generates LaTeX tokens as outputs. We also designed a sequence-level objective function to train the neural network in an end-to-end fashion, which affectively enforced the grammar-level correctness of the predicted LaTeX sequences. Thirdly, we developed the PDF2LaTeX OCR system, which recognizes entire PDF pages of mixed text and MEs. In the backend, we implemented machine learning algorithms to segment and label the contents, and applied the neural translators to convert page images into their LaTeX sources. Finally, we integrated the PDF2LaTeX system with two existing figure and table extraction tools, which enables the system to process a much wider range of scientific documents. For demonstration, we developed a graphical user interface for readers to conveniently interact with the contents on PDF pages.

DEDICATION

To my family.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Liu for his support on this dissertation. His insights were crucial towards the completion of this work. Dr. Liu is not only an advisor on my Ph.D. career, but also a mentor in my life. He taught me how to face adversities in life and never give up easily. I will be forever grateful for his guidance and patience.

I would also like to thank Dr. Shipman, Dr. Huang, and Dr. Ferris, for agreeing to be a part of my Ph.D. committee and giving me suggestions and support throughout the course of this research.

I want to thank my colleagues Jason Lin, Xing Wang, Guoyu Fu, Donald Beyette, Junqi Yang, Colton Riedel, and Haidong Wang, for their help and efforts on my research. They made my time at the RTDS lab a great experience.

Thanks also go to my friends Guanlong Zhao, Dennis Silva, Avinash Parnandi, Jin Huang, Christopher Liberatore, Adam Hair, Jiayi Huang, Mengyuan Chao, Di Xiao, Jay Chou, Xilong Zhou, Yuhang Wei, Stephen Brownlee, Zichao Xie, Xuan Wu, Annette Thompson, Mike Thompson, and many others, for making my life at Texas A&M University such a wonderful memory.

Finally, thanks to my family for their unconditional support and to my girlfriend for her care and love.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Jyh-Charn Liu (advisor), Professor Frank M. Shipman, and Professor Ruihong Huang of the Department of Computer Science and Engineering and Professor Thomas K. Ferris of the Department of Industrial and Systems Engineering.

The MOP dataset analyzed for Chapter 3 was a collaboration work with my colleagues Donald Beyette and Jason Lin. The bigram regularization model in Chapter 3 was a collaboration work with my colleague Xing Wang.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

There are no funding sources related to this dissertation.

NOMENCLATURE

AST	Abstract Syntax Tree
BBox	Bounding Box
CMML	Content Mathematical Markup Language
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DME	Displayed Math Expression
FP	False Positive
FN	False Negative
GUI	Graphical User Interface
IME	Inline Math Expression
IR	Information Retrieval
LSTM	Long Short-Term Memory
MathML	Mathematical Markup Language
ME	Math Expression
MIP	Mixed Integer Programming
MIR	Mathematical Information Retrieval
MLE	Maximum Likelihood Estimation
MOP	Math Objects in PDF
MRF	Markov Random Field
NLP	Natural Language Processing

NME	Non- Mathematical Expression
OCR	Optical Character Recognition
PDF	Portable Document Format
PMML	Presentational Mathematical Markup Language
PoS	Part-of-Speech
PPC	Projective Profiling Cutting
Regex	Regular Expression
RL	Reinforcement Learning
STEM	Science, Technology, Engineering, and Mathematics
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
URL	Uniform Resource Locator
XML	eXtensible Markup Language

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

Background

Researchers are experiencing an explosion of scientific publications. According to a research done by the University of Ottawa [1], by the year of 2009 researchers have published 50 million research papers in cumulative. This trend is fast growing. According to the 2019 statistics report from [2], on arXiv.org alone there are around 150,000 preprints published each year, mostly in the fields of physics, math, and computer science, as illustrated in Figure 1 (reprinted with permission from arXiv.org). Modern search engines have made it possible to retrieve a research paper with only a few keywords from large-scale databases. However, retrieving fine-grained contents from research papers in PDF format remains an unsolved problem. “PDF is evil”, as stated by the creator of PDFMiner [3], for good reasons. PDF is the *de facto* standard publishing format. Despite its popularity, PDF is essentially a collection of graphical representations. Even though modern PDF encodes text information into PDF fonts [4], this information can be missing or even be wrong [5]. In addition, PDF does not contain structural information or tags, making it difficult for machines to understand contents beyond text. For example, in the fields of science, technology, engineering, and mathematics (STEM), math expressions are heavily used and are blended into the main body text. Unfortunately, math expressions are not tagged and are often represented as graphics. This not only harms text-based information retrieval and knowledge mining, but also misses the opportunity to explore the rich technical information carried inside math expressions.

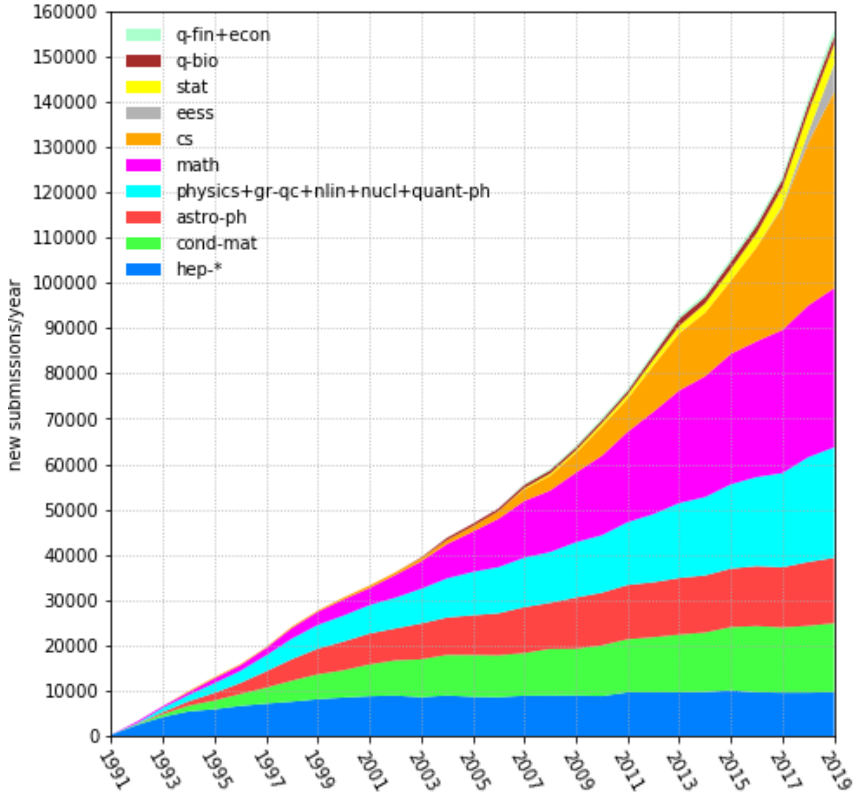


Figure 1 Trends of arXiv publications, reprinted from [2].

In contrast to the PDF format, LaTeX is much more informative on organizing document contents. It explicitly marks different types of document components including math expressions, figures, tables, etc. One existing effort to advocate LaTeX is the preprint publisher arXiv.org, which gives authors the option to upload LaTeX source files together with PDF files. Nevertheless, this is still a relatively small portion compared to the vast volume of existing PDF publications. Since manual annotation is apparently infeasible, advanced techniques are needed to process PDF contents.

Existing PDF parsers like *PDFMiner* [3] and *Apache PDFBox* [6] provide some nice functions to decode PDF contents and reconstruct text and basic page layout structures such as columns and paragraphs. However, these tools are far from ideal not only because they rely on PDF fonts (which may not be available), but also because contents beyond text cannot be processed. In addition, image-based PDF documents would make PDF parser-based solutions in vain. Optical character recognition (OCR)-based approaches can be used to overcome the limitations of PDF parsers. Modern OCR techniques can recognize English text at very high accuracy [7], but recognizing other components remains challenging. *InftyReader* [8] is the best-known commercial software to convert mathematical documents into LaTeX source files. Yet it is costly and the performance is still unsatisfying when being applied to scientific publications.

Research Objectives

In this dissertation I developed a PDF content analysis system for scientific publications. The system can extract and recognize different paper components including plaintext, math expressions, figures, and tables. The recognition results can be used to reconstruct the LaTeX source files of the target PDF documents. The system provides semantic-level understanding of the PDF format, which brings at least the following benefits to the research community:

- Recognizing the correct text encodings can improve the performance of text-based information retrieval and knowledge mining on PDF documents.
- Math expressions in LaTeX format can be used for math information retrieval (MIR) [9]. MIR is almost inaccessible nowadays except for a few applications with

very limited databases, such as NIST Digital Library of Mathematical Functions [10] and Wolfram Functions Site [11].

- Math expressions in LaTeX format can be easily converted to other formats like MathML [12], which can be used on web browsers, and Braille code [13], which can be used for blind people.
- Figures and tables can be indexed and retrieved to help researchers quickly grasp the gist of other research works.
- For historical PDF documents based on images, conversion to LaTeX is useful for data compression and indexing purposes.

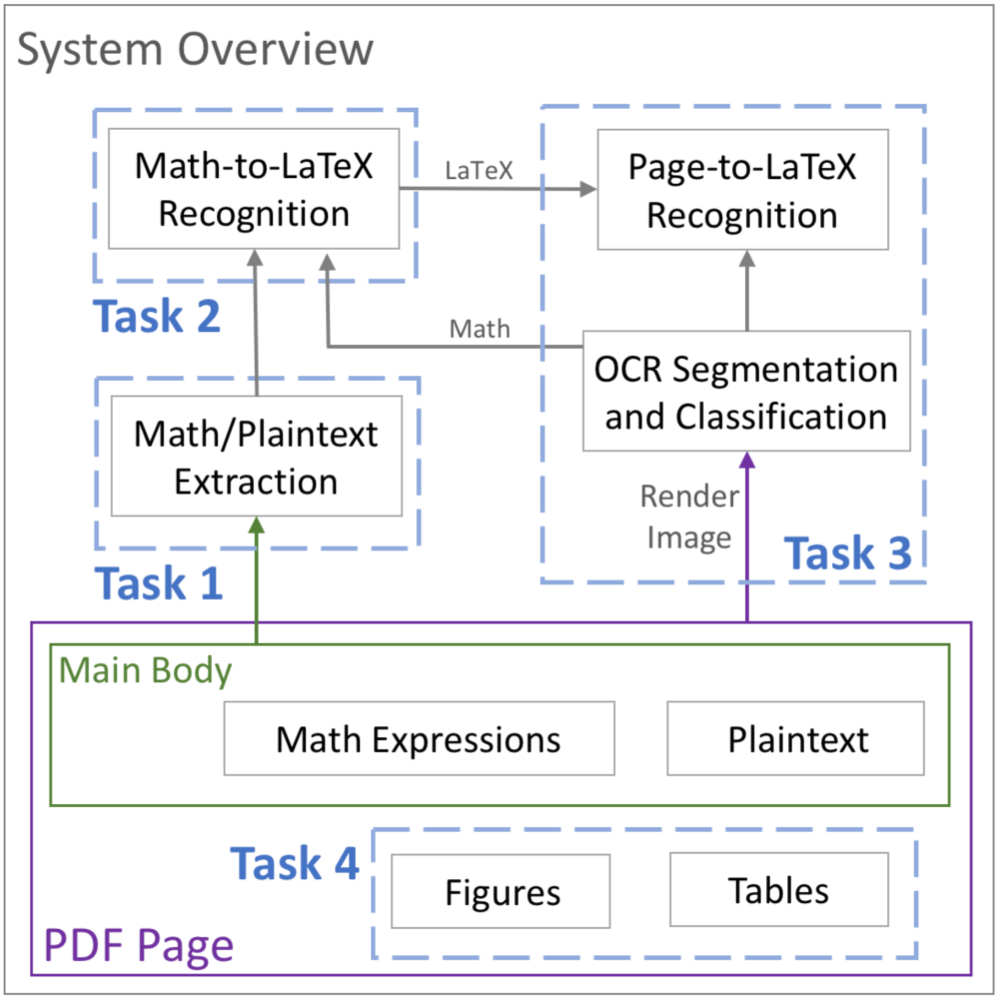


Figure 2 System overview and research tasks.

The overall design of the proposed system is shown in Figure 2. In this system, a PDF page is modeled as the main body, figures, and tables, where the main body is composed of a mixture of plaintext and math expressions. The system processes tables and figures separately, because they are usually placed at dedicated page areas. The bounding boxes of inline and displayed math expressions are first extracted based on a customized PDF parser and a feed-forward multi-stage algorithm. Next, a deep neural network with CNN-LSTM architecture is trained to translate the images of math expressions into their markup LaTeX. The system can then use OCR approaches combined with a series of machine learning algorithms to convert entire PDF pages into their LaTeX sources. The system is composed of four research tasks listed below:

Task 1: Math expression extraction with feed-forward multi-stage algorithms

PDF is meant for page layout design, where the structural information and semantic tags of the document contents are not available. As a result, extracting math expressions from PDF is not a straightforward process. In this task, I will propose a model that can extract math expressions based on the PDF typesetting. The model is based on the observation that math expressions follow different layout patterns as compared to plaintext. This makes it possible to predict the tags and positions of math expressions based on the font information given by PDF parsers. A PDF page is formulated as a collection of *symbols* and *tokens*. Each symbol is associated with glyph name, value v_c, font f_c, and bounding boxes. The likelihood of a token being a plaintext word or a segment of a math expression is first calculated using the statistics of the font size. To reduce the split detection errors, a bigram regularization model is proposed to increase the stability and the smoothness of the label prediction by considering the labels of neighbors. The complete math expressions can be constructed by merging neighboring tokens with math tags. The contents and bounding boxes of math expressions will be extracted and evaluated, and will be further analyzed in the following research tasks. I will also introduce a new dataset, which was generated semi-automatically for evaluating the proposed model and related research tasks.

Task 2: Math2LaTeX translation with an encoder-decoder deep neural network

Math expressions carry the most significant technical substances in STEM papers, but they cannot be easily understood by machines because they are represented as graphical elements in PDF documents. In this task I will propose a model that recognizes math expressions based on image inputs. The recognition results will be saved as LaTeX format. This not only involves segmenting and recognizing individual symbols, but also the size and layout relationship between different symbols, and a language model to guide correct LaTeX grammar. This problem is an intersection of image processing and sequence prediction. As a result, I propose to use a deep neural network with encoder-decoder (CNN-LSTM) architecture, where the encoder is used to process the input images, and the decoder is used to generated the LaTeX sequences that mark up the input MEs. Relative position is important for analyzing the structure and relationship between different math symbols. To preserve this information, I propose to tailor the sinusoidal positional encoding method proposed in the Transformer model [9] into 2D to preserve the spatial locality information of math expressions. I will also introduce a sequence-level training objective function, which can enforce the correctness of the entire LaTeX sequence during training and improve the performance of the model. I will also introduce the policy gradient algorithm that made sequence-level training possible.

Task 3: PDF2LaTeX conversion with OCR and machine learning algorithms

The task of PDF2LaTeX conversion aims to reconstruct the LaTeX source code for entire PDF pages, which are composed of a mixture of text and math expressions. This is different from task 2 in that previously the model only translates individual math expressions that are assumed to be extracted already. This task processes entire mathematical documents. This can be done by simply combining task 1 and task 2, but it will not overcome the limitations of PDF parsers such as missing fonts. Instead, I will introduce a new system based on OCR, which extracts math expressions and text in both postscript and image-based PDF files and translates them into markup LaTeX. The input becomes a grayscale image of a page. Assuming that tables and figures are already detected and removed from PDF pages, the proposed system need to first recognize the page layout and segment it into plaintext words and math expressions, and then translate them to LaTeX individually. The boundary between plaintext words and math expressions will also need to be determined. This is done by first using the profile projection cutting (PPC) to split the images of pages into column, lines, and tokens. To determine if a token is a plaintext word or a math segment, we used a CNN as a binary classifier that captures visual features and classify the labels of each token. To determine the boundary of math and text, I propose to post-process the CNN classification results with a conditional random field (CRF). After math classification, complete math expressions can be constructed by merging neighboring math segments. Their contents are finally recognized using the CNN-LSTM neural network.

Task 4: Integration of figure and table extraction modules

The presence of figures and tables in a page can disrupt the parsing process of the PDF2LaTeX system. In addition, figures and tables themselves contain important information. As a result, it becomes necessary to add figure and table extraction modules into the proposed system. Figure images can be very diverse and difficult to recognize, while their caption text can be detected and utilized. For tables, one can extract not only their caption text, but also their rich cell data because table structure can be reconstructed relatively easily. Commercial software and research tools that extract figures and tables are readily available, but a comprehensive system to parse various components in scientific papers still does not exist. In this task, I will integrate figure and table extraction modules into the PDF2LaTeX system as a further step to the semantical understanding of paper contents. The positions of figures and tables are detected using the PDFFigures [10] software. The extraction results are given as the bounding boxes of these objects and the text of the captions. In addition, I will use the Camelot software [11] to extract the cell data from tables. Finally, figures and tables are masked out from the page by overlapping the detected area with white pixels. The rest of the contents are converted to LaTeX using the existing PDF2LaTeX system. The final system is able to extract text, math expressions, figures, tables, and their captions all together and index the recognized contents for information retrieval purposes.

Summary of Findings

In summary, the proposed PDF content extraction and recognition system can semantically parse different components in scientific publications, including plaintext, math expressions, figures, and tables. The main body including plaintext and math expressions can be recognized and converted to LaTeX format. Figures and tables can be extracted and saved together with their captions for indexing purposes. The system serves as the foundation to understand semantical contents and retrieve information from scientific publications in PDF format. The technical novelties in the proposed work are summarized below:

Table 1 A summary of technical novelties in this dissertation.

	Existing Works	This Work
Math Extraction from PDF	Rule-based and global training-based methods at lower accuracy, lack of training and test data	Font-based algorithms for math extraction, bigram regularization to incorporate neighboring information, MOP dataset for evaluation
Math-to-LaTeX Translation	Empirical math structural analysis, token-level training objective function, lack of position information	Positional encoding for math layout structure, elimination of exposure bias problem, policy gradient for sequence-level objective function
PDF-to-LaTeX Conversion	Open-source plaintext recognition tools, commercial math document analysis tools at lower accuracy and high cost	PPC for PDF image segmentation, CNN and CRF for math/plaintext labeling, deep neural networks for LaTeX translation
Figure/Table Extraction	Rule-based and machine learning-based algorithms that only detect figures/tables	An integrated system that extracts plaintext, math expressions, figures, and tables simultaneously

Dissertation Outline

The rest of the dissertation is organized as follows. In Chapter II we discuss the related works, including extracting MEs from PDF documents, recognition of MEs, conversion of MEs from images to LaTeX, conversion from PDF documents to LaTeX, and figure and extraction from PDF documents. In Chapter III, we introduce a feed-forward multi-stage algorithm which is used to locate the bounding boxes of MEs in PDF documents, and a bigram regularization model that can stabilize the predicted labels and reduce split detection errors. We also introduce the MOP dataset in this chapter. In Chapter

III, we introduce an encoder-decoder neural network which is used to take images of math formulas as inputs, and converts the recognized contents into their markup LaTeX strings in an end-to-end fashion. We also introduce the 2-dimensional positional encoding and a sequence-level objective function used in this model. In Chapter IV, we introduce a comprehensive OCR system called PDF2LaTeX, which can convert entire mathematical documents in PDF format into their LaTeX source files. We will discuss the details of the machine learning algorithms in the system including deep neural networks and a conditional random field. In Chapter V, we enhance the PDF2LaTeX system with figure and table extraction modules. We will briefly discuss the mechanisms of two external tools and explain how they are embedded into the existing system. Finally, in Chapter VI we conclude this dissertation and discuss future works.

CHAPTER II

LITERATURE REVIEW

Extracting Math Expressions from PDF documents

PDF is a layout-based format designed for printing and exchanging documents [4]. Modern works on ME extraction from PDF documents are mainly based on OCR or PDF parser [12-14]. In the OCR-based approaches, a PDF document is first rendered into an image, and MEs are detected based on shape analysis. For example, in [15], the OCR technique was used to recognize MEs from Japanese documents, where non-Japanese characters are classified as ME characters. A follow up work [8] further utilized the position and size information to improve the performance. An image segmentation method based on fuzzy logic was proposed in [16] to isolate MEs area from plaintext area. A deep learning technique based on the combination of convolutional neural network and recurrent neural network was employed to detect MEs based on image analysis in conjunction with PDF metadata analysis [13].

Since PDF does not contain tagged information about its contents, external processing tools are required to extract and understand its elements. Some popular text-based open source PDF parsers include Apache PDFBox [6], Apache Tika [17], Poppler [18], PDFMiner [3], etc. These tools can not only extract text from PDF documents, but also the metadata associated with the text, such as font, glyph name, Unicode, bounding box, etc. As a result, PDF parser provides richer and more accurate information over OCR-based PDF processing methods [13, 19, 20].

ME extraction based on PDF parsers has been extensively studied recently [12, 21-23]. The general features used to differentiate ME from Non-ME (NME) include the following aspects: math elements, fonts, linguistics, and spatial layouts. Math elements include operations, relations, Greek symbols, delimiters, functions, integrals, fractions, and squares. In [24] the authors also used the special font name to extract MEs. Linguistic features include the purity of words [12], letters ratio [22], and matching with plaintext word [21]. Spatial layout features include line height, above/below space, left/right indent [24], line centeredness, variation of line width [23], sparsity of characters, variance of baseline, variance of bounding box size [12]. There is a trend of using adaptive features [21, 22] besides the general features. For example, to accommodate the writing habits of each user, [21, 22] proposed to use the local features based on the identified displayed mathematical expression. Past methods mostly model the ME extraction problem as a classification problem and train a discriminant model. [22] is the only work that systematically models the neighboring information for the decision-making processes. The work proposed to use the Conditional Random Field (CRF) as a sequential model for the inline ME detection.

Different attributes of PDF objects have been used as features in machine learning models. In [12], the support vector machine (SVM) was used to identify inline MEs based on geometric layout and content features. Nine different machine learning algorithms, combined with heuristic rules were used to extract both inline and displayed MEs [23]. In a recent work [21], a weakly-supervised Font Setting based Bayesian model (FSB) was proposed for ME extraction. Without using any ground truth data for training, the algorithm first employed heuristic rules for displayed ME detection, and then used a Bayesian predictor based on the font and glyph name value of the displayed ME characters to detect inline MEs.

Math Expression Recognition

Automatic recognition of math formulas in digital publications has long been recognized as a challenging task [25]. The task first requires to locate math formulas in digital documents, then analyze the structure of math formulas, and finally translate them into math markup languages. In [20], Garain *et al*. proposed to use a commercial OCR tool as a text classifier, where patterns that cannot be recognized by the OCR were further analyzed to detect math formulas. In [5], *Wang et al.* developed a PDF parser to detect math formulas based on the font statistics with a feed-forward algorithm. In [26], they further proposed a bigram label regularization method to reduce the over-segmentation problem during formula detections. In [13], Gao *et al*. proposed to combine the PDF font information with vision features, and manually labeled a large dataset to train a deep neural network for math formula detection. Once math formulas are detected, the next step is to analyze their 2-dimensional layout structure. Twaaliyondo *el al.* in [27] proposed a method that first divided the formulas into subexpressions based on larger symbols and blank spaces in a recursive manner, and then represented the structure of the formulas as a tree. In [8], Suziki *et al.* used a similar approach as [20] to first locate the math formulas, and then represented the structure of math formulas as trees, and used a minimum-cost spanning-tree algorithm for the structure analysis. This proposed work was made into the commercial software -- InftyReader.

Recently, convolutional neural networks have achieved new performance levels for OCR tasks [28], which gives new solutions to translate math formulas from images in a data-driven manner, yet requiring to resolve the following additional problems: 1) the input image is not segmented, 2) the output is a sequence of tokens of arbitrary length, and 3) structural information needs to be understood. Techniques such as Connectionist Temporal Classification (CTC) [29] models the inter-label dependencies implicitly, making it possible to train a neural network directly with unsegmented data. Existing solutions to predict sequence from image inputs can be found in text recognition and image captioning tasks [29-34], which usually combines CNN with a sequential model to construct an encoder-decoder (seq2seq) architecture. Jaderberg *et al.* in [30] showed that combining CNN with NLP techniques like Conditional Random Field (CRF) was very effective in recognizing text in images. Another common approach is to use RNN as the sequence predictor. This was referred to as a CRNN model in [34], which was end-to-end trainable for image-based sequence recognition tasks. The attention mechanism [35] has been proposed to emulate the human vision system, which allows the model to attend the salient parts of an image while generating the target sequence. Xu *et al*. in [32] combined the attention mechanism with the CRNN model which achieved further performance gain in image captioning task. With minor modifications, this architecture can be tailored to translate images of math formulas into their LaTeX markup sequences.

In [36], Zhang *et al.* proposed a gated recurrent unit (GRU) based encoder-decoder model combined with attention mechanism to translate handwritten math to LaTeX. The model takes the stroke information as inputs, and shows capability to recognize both symbols and their structures simultaneously. In [37], they replaced the GRU encoder with a CNN encoder, enabling the model to take images as inputs instead of strokes. In [38], Deng *et al.* proposed another seq2seq model that targets on machine-rendered real-world math formula images. The model is composed of a CNN and a multi-row RNN as the encoder, and an attention-based LSTM as the decoder. The model was tested on the IM2LATEX-100K dataset and outperformed the INFTY system [8]. The model was found to achieve good performance for recognizing handwritten math formulas as well [39]. In [40], Wang *et al.* improved the model in [38] by replacing the CNN encoder with a DenseNet [41], and enhanced the attention mechanism with a joint attention mechanism [42], which combines the channel-wise attention with spatial-wise attention. In [43], Zhang *et al.* increased the source image size by two times and applied double-attention mechanism, and improved the performance over [38]. All the above-mentioned works used the token-level maximum likelihood estimation as the training objective.

PDF Recognition

Similar to ME extraction, techniques for PDF document analysis can also be categorized into two types: PDF parser-based and OCR-based techniques. PDF parsers are tools used to decode PDF source files [4] into font objects which contain information such as text, bounding boxes, sizes, etc. On the other hand, OCR-based approaches process PDF documents as images. In either type, it is crucial to recover the structure of the pages and differentiate between math and text.

In PDF parser-based approaches, statistics of font information can be used to analyze the structure of PDF pages and detect math expressions. For example, in [21]

Wang et al. developed a PDF parser based on PDFBox to extract font objects from PDF documents, and used statistics of font positions and sizes to separate math expressions from plaintext. In [26], they further developed a bigram regularization model to incorporate the neighboring information to enhance the boundary detection between math and plaintext. Similarly, Iwatsuki et al. in [22] trained a conditional random field (CRF) which incorporated neighboring information and linguistic features, and achieved performance gain on detecting inline math expressions. Once math expressions are located, additional techniques are required to recognize the contents. For example, Baker et al. in [24] used a linear grammar approach to parse math expressions in PDF documents, and went further in [14] to interpret the semantical meaning of math expressions using spacing and font information. A major limitation of this type of technique is that they rely on the text information in fonts, which are not always available.

OCR-based approaches are not subject to the missing font problem but can involve more in-depth techniques. This is because information available in PDF parsers will instead need to be inferred from pixel values. Fortunately, modern OCR techniques, especially deep learning techniques, have made it possible to effectively extract such information from images. For example, Gao et al. demonstrated using CNN to detect math expressions [13] from PDF documents, and showed that combining pixel information with PDF parser output can further enhance the recognition accuracy. Deng et al. [38, 39] demonstrated that by using the CNN-LSTM neural network architecture with attention mechanism, it is possible to translate an image of a math formula into its markup LaTeX source in an end-to-end fashion. In [44], the performance is further enhanced with positional encoding and sequence-level training.

We have reviewed the literature on math extraction and recognition. Still, recognizing an entire PDF page is a more complicated task and there are very few tools available for this. Tesseract OCR [45], originally developed in Hewlett-Packard lab and now maintained by Google, is an excellent open-source OCR tool that recognizes not only text in different languages at high accuracy, but also the basic structure of pages. However, it does not handle math expressions thus cannot be used to process mathematical documents. InftyReader [8] is the only known system by far that recognizes PDF pages with math expressions. It is a commercial software based on the Infty system [8] developed by Suzuki et al, which can not only convert math expressions to LaTeX and MathML, but can also process entire PDF pages and recover the markup LaTeX sources.

Extracting Figures and Tables from PDF Documents

In the 2017 international conference on document analysis and recognition (ICDAR), a research competition [46] was held on page object detection algorithms for document images. The target objects include figures, tables, and displayed math formulas. Almost all submitted works used deep learning models to detect page objects. Among these works, Faster-RCNN [47] is the most popular choice because it is the state-of-the-art object detection model that has been widely used on natural scene images. Saha et al. in [48] used this model to detect page objects. Due to the lack of training data, they used transfer learning based on a pre-trained model trained on ImageNet [49]. The model achieves excellent accuracy on table detection, but on figure and math expression detection the accuracy becomes lower. A similar approach was proposed by Schreiber et al. earlier in [50]. In this work, the authors also used Faster-RCNN but focused on table detection and went further to use another neural network to detect table rows and columns. A common shortcoming of deep learning-based approaches is the lack of data and the drifting of the predicted bounding boxes, which are not effectively captured by the evaluation criterion--intersection over union (IOU) rate of 80%. In contrast, PDF parser-based methods can behave more robust since text information can be utilized. In addition, captions can be detected easily based on string matching and position heuristics. In [51], Perez-Arriaga et al. developed a TAble Organization (TAO) system based on PDFMiner [3]. TAO generates table candidates by applying heuristic rules based on structural alignments on the XML output of PDFMiner. It then extracts the text contents of each cell in the candidate tables and saves them into JSON format. This system is purely based on structural information without using text information, thus does not detect captions. In [52], Clark et al. developed a figure and table detection system called PDFFigures. The system uses a unified framework to detect both figures and tables based on the observation that a region with no body text that is adjacent to a caption must contain either tables or figures. The first step is to extract the text with a PDF parser, and then detect captions based on keywords matching and a few heuristic rules. Body text and figure text (text inside figures/tables) are differentiated by page margins. Once the captions are located, their adjacent regions of space are scored based on their size, number of figure text, etc. Finally, the captions are assigned to the proposed regions with the highest scores. Figures and tables are differentiated by the caption keywords. In [10], the authors added more heuristic rules to the software and released PDFFigures 2.0, which is applicable to publications in a wider range of topics.

CHAPTER III

MATH EXPRESSIONS EXTRACTION FROM PDF DOCUMENTS BASED ON MODELING OF FONTS[*]

This chapter proposes a multi-stage architecture to extract math expressions (ME) from PDF documents based on font analysis. The feed-forward algorithm starts from symbol-level analysis based on metadata of PDF objects, including font size, font name, and glyph name. Two subsequent stages utilize a group of spatial and semantic heuristics to merge multiple ME symbols into both inline ME and displayed ME. For inline ME, they are blended into plaintext sentences in scientific papers. Detecting inline MEs is a non-trivial problem due to the unrestricted usage of font styles and blurred boundaries with plaintext in scientific publications. For instance, many inline MEs detected by existing algorithms are split into multiple parts incorrectly due to the misidentification of a few characters. As such, we propose a bigram regularization model to resolve the split detection problem in inline ME detection. The model incorporates neighboring constraints

[*]Reprinted with permission from “Extraction of Math Expressions from PDF Documents based on Unsupervised Modeling of Fonts” by Wang, Z., Beyette, D., Lin, J., & Liu, J. C. in *2019 International Conference on Document Analysis and Recognition (ICDAR)* (pp. 381-386), Copyright 2019 IEEE, “Bigram Label Regularization to Reduce Over-Segmentation on Inline Math Expression Detection” by Wang, X., Wang, Z., & Liu, J. C. in *2019 International Conference on Document Analysis and Recognition (ICDAR)* (pp. 387-392), Copyright 2019 IEEE, and “Semi-Automatic LaTeX-Based Labeling of Mathematical Objects in PDF Documents: MOP Data Set” by Beyette, D., Wang, Z., Lin, J., & Liu, J. C. in the *Proceedings of the ACM Symposium on Document Engineering 2019* (DocEng’19) (pp. 1-4), Copyright 2019 ACM.

during labeling of ME vs. plaintext. The algorithm is tested on the Marmot dataset (amended with missing cases). For displayed ME, the proposed method achieved 93.6% precision, 99.4% recall, and 96.4% F1-score. For inline ME, the method achieved 92.2% precision, 91.9% recall, and 92.1% F1-score. In addition, the algorithm only takes an average of 1.09s to process a page, which is faster than other existing methods. Finally, we will introduce the MOP (Mathematical Objects in PDF Documents) dataset and the semi-automatic ME labeling system used to generate the dataset. A total of 1,802 PDF pages from arXiv high energy physics (hep-th) were labelled, and a benchmark was generated with the proposed algorithm.

Overview

Natural language processing (NLP) techniques have been widely used to extract knowledge from scientific publications and facilitate the process for information retrieval [53]. However, NLP techniques are specifically designed for standard natural languages, which makes it difficult to be directly used in lots of scientific publications because a sentence is often mixed with non-textual elements such as math expressions (ME). Thus, identifying the non-textual parts of a sentence becomes a critical issue. This is especially important for Portable Document Format (PDF) documents, because the PDF format does not contain tagged information about its content. PDF is the *de facto* standard format for modern electronic publications. Although the PDF format is a versatile format for document sharing and printing, it is difficult to retrieve non-plaintext information from PDF documents. While research has been done on recognizing basic components from PDF documents, such as headings, tables, paragraphs, etc. [54-56], extraction of mathematical expressions remains an unsolved problem. Human readers can effortlessly distinguish MEs from natural language words, based on reserved words, names, and symbols, it can be highly challenging for a computer to achieve perfect accuracy in ME extraction, especially when plaintexts are used for ME representations. In addition, complex math notations/symbols may be composed of smaller pieces of graphic primitives. For instance, a square root “$\sqrt{\ }$” could be composed of graphic elements “√” and “−” in PDF. MEs can be loosely classified as displayed MEs and inline MEs. Displayed MEs are isolated from plaintext and occupy one or more lines. They are relatively easier to detect because of their special spatial patterns. Inline MEs are blended into natural language sentences and are generally harder to detect.

Being able to detect MEs automatically would greatly benefit the sentence understanding by machines such as Part-Of-Speech Tagging [57]. In addition, the MEs themselves contain important information because they are concise representations of scientific contents in publications, especially in the Science, Technology, Engineering, and Mathematics (STEM) fields. As such, being able to extract MEs automatically is the foundation to scientific contents analysis. Different approaches have been proposed to extract ME from PDF documents. One category of existing works used the optical character recognition (OCR) techniques by first converting the PDF documents into images. However, special math symbols and spatial relationship could be incorrectly recognized by OCR tools. OCR techniques do not utilize the rich information encoded in PDF format. Another category of solutions used PDF parsers to extract PDF objects first, and then apply rule-based methods or machine learning algorithms to predict MEs. The key is to utilize the encoded features with discriminant power in order to achieve good extraction performance at low computation cost.

In this chapter, we propose a multi-stage algorithm to extract ME, based on the hierarchical relationship between math elements. In the first stage, we introduce a likelihood ratio test model based on font size variation feature, and matching of font name and glyph name to detect individual ME symbols. In the following stages we propose a group of heuristic rules to merge ME symbols into inline MEs and displayed MEs. Finally, we will introduce a bigram regularization model that utilizes the neighboring information in bigram tokens to fix the misidentified ME labels and reduce the split ratio. The model penalizes the label change thus increases the stability of the prediction. While our model is simple, fast, and accurate, it does not recover the semantics (the math symbolic notation) of ME symbols when they are encoded by multiple PDF objects. Thus, we will use "symbol" and "PDF object" interchangeably in the following sections.

The Multi-Stage Algorithm

In our model, a PDF document is formulated as a set of encoded PDF objects. The metadata of an object describes its attributes, such as Unicode, bounding boxes, font, glyphs (shape in vector graphics), glyph names, etc. A math symbol can be represented by a set of PDF objects. A set of closely spaced math symbols forms a math token which represents a multi-variable/operator math notation. An ME may be composed of multiple tokens. The overall hierarchy, starting from PDF objects, math symbols, math tokens, to inline and displayed MEs is illustrated in Figure 3.

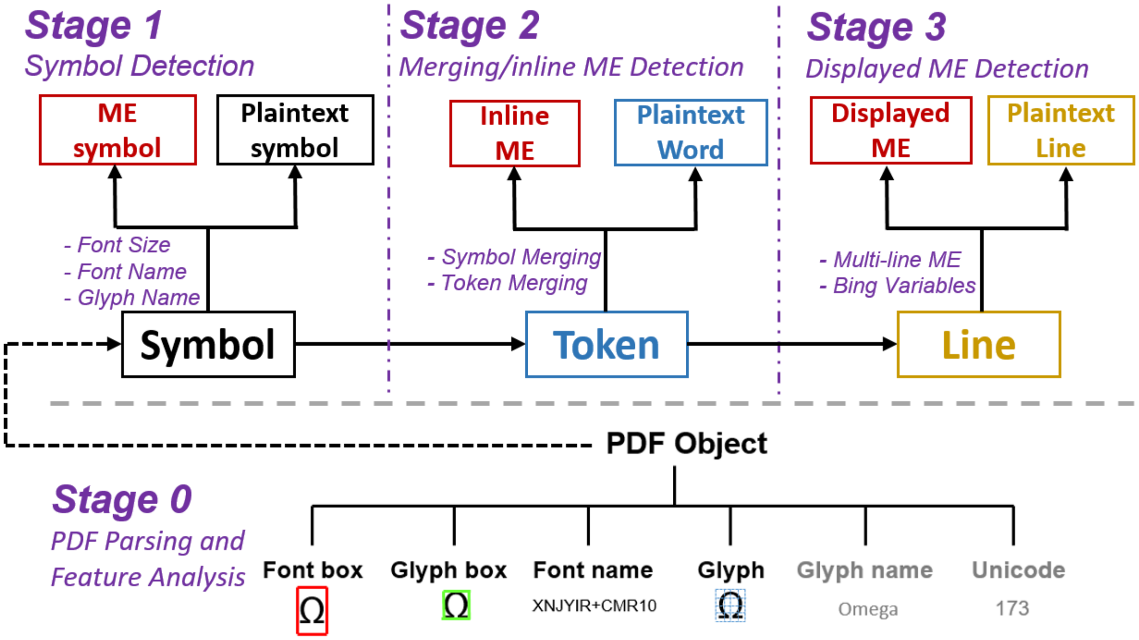


Figure 3 The flow chart of the system model. The bottom line shows an example of the metadata of "Ω". Glyph name and Unicode are shown in grey because they are not always available in PDF documents.

The preprocessing step, stage 0, performs the PDF parsing in order to extract features needed for symbol extraction. At stage 1, symbols are classified into ME vs. plaintext using a likelihood ratio test model based on font size features, and matching of font name and glyph name. At stage 2, symbols are merged into tokens, which are further merged to produce inline ME vs. plaintext words based on a few different heuristic rules. At stage 3, displayed MEs are identified based on a group of spatial rules.

Stage 0: PDF Pre-processing and Feature Design

PDF Pre-processing

Several open source PDF libraries are available for PDF parsing, e.g., Apache PDFBox [6], Apache Tika [17], PDFMiner [3], etc. We developed a customized PDF parser on top of the PDFBox library to extract PDF objects and their metadata including font name, glyph name, Unicode, and two types of bounding boxes (BBox). There are two types of BBoxes: *font box* and *glyph box*. The PDF specification states that a font box is "*the smallest rectangle enclosing the shape that would result if all of the glyphs of the font were placed with their origins coincident and then filled*" [4]. A font box (marked as red boxes in Figure 4) usually has some white space between the box edges and the symbol itself. These boxes are identical to the highlighted boxes when marking text on PDF files. The glyph box is a box in contact with the glyph shape [4] (shown as the green boxes in Figure 4). Effective methods to calculate gylph boxes can be found in [24, 58].

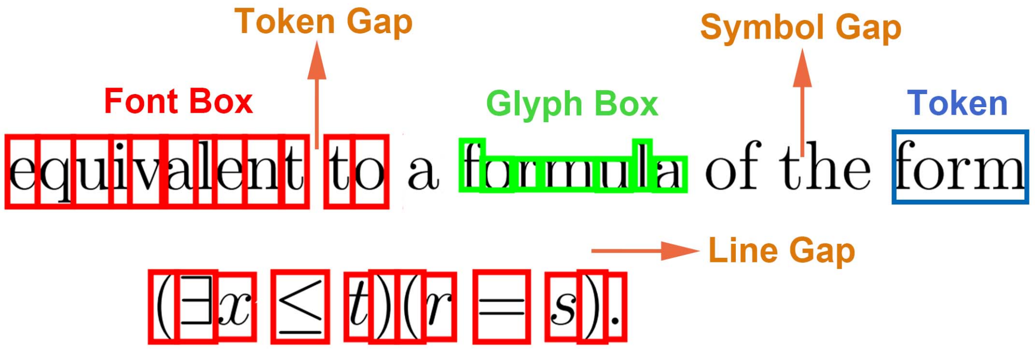

Token Gap
Symbol Gap
Font Box
Glyph Box
Token
equivalent to a formula of the form
Line Gap
$(\exists x \leq t)(r = s).$

Figure 4 Font box, glyph box, and three different types of gaps in a sequence of PDF objects representing words and an ME.

Using the top left coordinates of each glyph box as the reference position of each PDF object, we first sort these PDF objects in the left to right, and top to down sequence. Glyph boxes are not used in the subsequent ME extraction steps, but they are still important for future ME layout analysis (e.g., subscript, superscript). We also adopt the projection profiling cutting (PPC) method to separate text into different columns [21]. The PPC technique is based on calculating the total count of black pixels for each column line.

The blank space between adjacent columns has zero (black pixel) counts, and is used as a delimiter to direct the order of row-by-row PDF objects.

In addition to the column spacing, three other types of spacing gaps among PDF objects are analyzed: the *symbol gap*, *token gap*, and *line gap*, which are illustrated in Figure 4. The physical gap sizes may vary among documents, but their relative relationship on the histogram holds. Figure 5 shows an example of the histogram for gap values of a PDF page. Based on the observation on histograms, we set the first highest peak of the histogram as the symbol gap: the gap between adjacent symbols in a token. We set the second highest peak to be the token gap. Adjacent symbols whose gaps that are smaller than the token gap are grouped as tokens. In plaintext, a token usually corresponds to a natural language word. In MEs, a token could be only part of an ME because the gaps inside MEs are less uniform and can be larger than the token gap value. Thus, the gap-based analysis is necessary for detecting tokens of PDF objects, but additional attributes are needed to recover the structure of MEs.

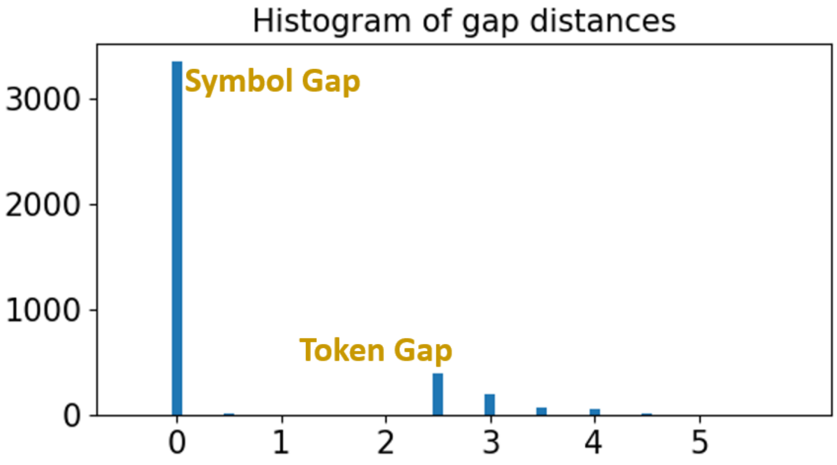


Figure 5 A histogram of gaps between adjacent font boxes on a PDF page. X-axis shows the gap distance in pt. The bin width is 0.1 pt.

Font size as a feature

In our system design, the height of a font box is defined as its font size. Similarly, the height of a glyph box is defined as its glyph size. In most documents, the font size for plaintext remains unchanged, except for headings, special layouts, e.g., highlights, tables, etc. On the other hand, we observed that the font size of symbols in most MEs changes frequently.

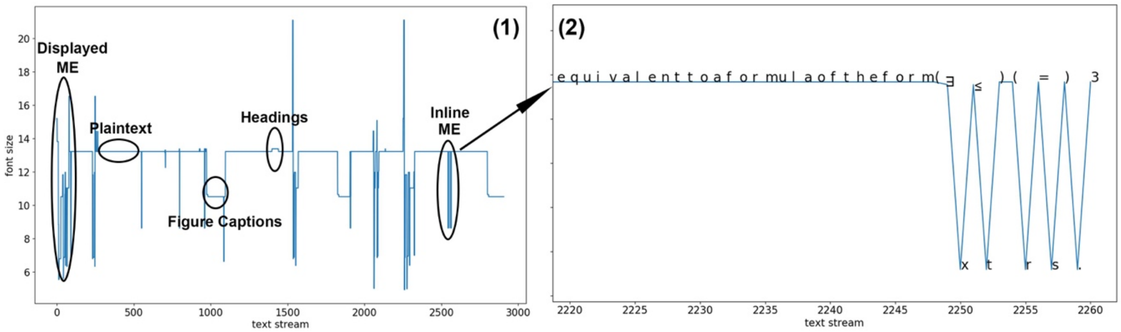


Figure 6 (1) Text stream vs. font size (in pt) of a single PDF page. The black circles mark the occurrences of different PDF elements. (2) A zoom in which shows the phrase and its font sizes.

An example of the font size distribution for a PDF page, which contains the phrase illustrated in Figure 4, is given in Figure 6 (1). The font size for the main body text is around 13pt. However, the font size changes with other components: figure/table captions, headings, inline/displayed MEs, etc. In this particular example, the font sizes in MEs range from 5pt to 21pt, where the very large font sizes correspond to large operators, and the very small ones correspond to sub- or superscripts. Figure 6 (2) gives a zoom in view of font sizes corresponding to the sentence in Figure 4, with the actual symbols plotted on top of the trace line, clearly suggesting the font size fluctuates in MEs. This patterns were consistently observed in numerous examples, except when plaintexts were directly used as MEs. This observation leads to the design of an unsupervised algorithm for symbol-level ME detection, where the font size is used as the detection feature. Details of this algorithm are discussed next.

Stage 1: Symbol-level ME Detection

This stage is consisted of two parts: (1) font size based likelihood ratio test, and (2) font/glyph name based matching.

Font Size-based Likelihood Ratio Test

Although MEs usually have variations in font size, the font size change alone is not sufficient to detect MEs because 1) font size also changes during transitions from the main body text to other elements such as headings, captions, etc.; 2) neighboring symbols inside an ME may have the same font size. Thus, a statistical model is necessary to incorporate the font size information systematically. Here we propose to use a likelihood ratio test model for symbol level classification of ME vs. non-ME (NME). That is, let c be an unknown symbol/character on a PDF page, FS be its font size, and $L \in \{ME, NME\}$ be its label. The likelihood ratio L_r test is based on the font size information, formulated as follows:

$$L_r(c) = \frac{P(L = ME|FS = fs_c)}{P(L = NME|FS = fs_c)}$$

The decision rule for any character c is that: if $L_r(c) > 1$, $L(c) = ME$, otherwise $L(c) = NME$. The key questions is how to estimate the likelihood on the right-hand-side in the equation. One obvious solution is to use ground truth data with human-labeled MEs, and model the likelihood $P(L = ME|FS = fs_c)$ with a supervised training model.

However, ground truth data can be expensive to label and contains human errors. An even more important issue is that the font size feature may not be generalized across different PDF pages. For example, a certain font size may be used intensively for plaintext in one document, but used mainly for MEs in a different document. Here, we instead propose an unsupervised grouping algorithm to generate ME/NME training samples automatically based on the font size variations. We build this model for each page we process independently.

First we observe that the font size changes in MEs occur rapidly within short distances. On the other hand, the font size of plaintext contents, including headings, figure/table captions, etc., changes at much longer distances. Based on this property, we propose a grouping algorithm to classify unknown symbols into two groups: G_{short} and G_{long} based on their distances to font size change, and use the groups as samples to estimate the likelihood of ME vs. NME. The pseudocode of this method is shown below:

Table 2 Pseudocode of the symbol grouping method

Symbol grouping method

```
procedure GROUP( P )
/* P is the sorted collection of all symbols on a page */
current_size = first_symbol.size
G = []
for char in P:
    if char.size != current_size:
        if length(G) <= threshold:
            add all symbols in G to $G_{short}$
        else:
            add all symbols in G to $G_{long}$
        G = []
        current_size = char.size
    add char to G
return $G_{short}$ and $G_{long}$
end GROUP
```

We set the threshold in the above grouping algorithm as 3. To explain this choice, we inspected 562 inline and displayed MEs from the Marmot dataset [59] and visualized the pattern of font size change inside MEs. The result is shown in Figure 7. We observe that the font size in MEs changes within 3 symbols in 95.2% of the cases. Thus, 3 is a reasonable choice for the threshold.

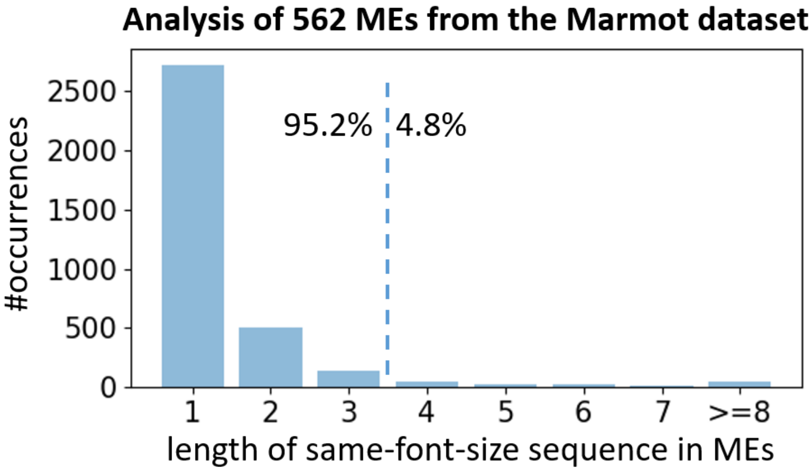


Figure 7 The length of symbol sequence until a font size change.

To derive the decision rule, we hypothesize that symbols in $\mathrm{G_{short}}$ are ME samples, and symbols in $\mathrm{G_{long}}$ are NME samples on an unknown PDF page. Let $\mathrm{Count(c|FS = fs_c)}$ be the total number of symbols with font size $\mathrm{fs_c}$ on a page. We can estimate the likelihood as:

$$\mathrm{P(L = ME|FS = fs_c)} = \frac{\mathrm{Count(c|FS = fs_c, c \in G_{short})}}{\mathrm{Count(c|FS = fs_c)}}$$

$$\mathrm{P(L = NME|FS = fs_c) = \frac{Count(c|FS = fs_c, c \in G_{long})}{Count(c|FS = fs_c)}}$$

Merge the above equations, we have:

$$\mathrm{L_r(c) = \frac{Count(c|FS = fs_c, c \in G_{short})}{Count(c|FS = fs_c, c \in G_{long})}}$$

Font Name and Glyph Name Matching

In addition to the font size feature, font name and glyph name of ME symbols also provide very useful information for identifying MEs. Knowing that numerous custom fonts are available in PDF documents, a systematic enumeration is required to complete the matching approach. As a demonstration, we explored a limited number of PDF documents, and identified that font names containing the following substrings are meant for ME symbols: (“GreekwithMathPi”, “Math”, “+MSBM”, “+CMSY”, “+CMMI”). Glyph names such as operators (“plus”, “equal”, “element”, “summationdisplay”), Greek symbols (“delta”, “gamma”), etc., also indicate ME symbols. Symbols with the above font and glyph names are identified as ME symbols.

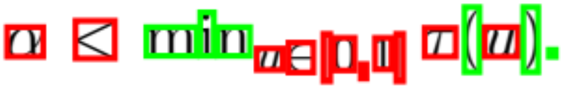


Figure 8 Fragmented detection of symbols in an ME. Symbols in red/green rectangles indicate symbols that are classified as ME/NME.

As it will become clear in Section IV, *Stage 1* alone already yields good performance at the symbol level. That being said, most MEs are composed of a sequence of multiple symbols, which should be aggregated into one single entity, rather than as separate ones. Figure 8 illustrates an example which should be labelled as one single ME, rather than a number of split ME symbols (marked in red) and false negatives (marked in green). In this case, the “min” and the parentheses are not recognized as ME because they have the same font size as plaintext (marked in green). The stage 2 algorithm aims to address this issue based on a number of heuristic rules.

Stage 2: Inline ME Detection

As mentioned earlier, symbols are grouped into a token when their gaps are smaller than the token gap threshold. That being said, additional steps are required to decide if adjacent ME tokens need to be further merged into one ME.

First, a token is designated as an ME token when any of its symbols is a known math symbol, and all symbols in the same token are also labeled as ME symbols. For the example in Figure 8, the word “min” in “$\mathrm{min}_{\mathrm{u}\in[0,1]}$” will be labeled as ME because it is within the same token as "$_{u\in[0,1]}$", which are known math symbols from the previous stage. Based on this rule, we get the token-level ME detection.

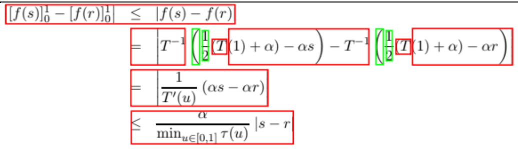
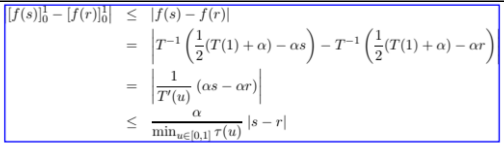
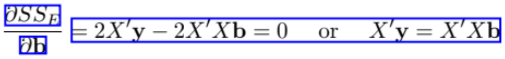
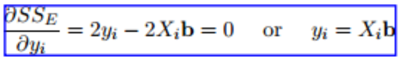
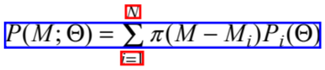
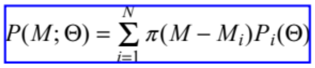
Next, adjacent ME tokens are merged into one ME as an inline ME candidate. Three additional rules are employed in this step: 1) plaintext commas, periods, and semicolons located at the end of the token are excluded from the merging process; 2) when a math operator (“$=$”, “$+$”, “$\in$”, etc.) is at the beginning or end of a token, its neighbor token is also merged; 3) if unmatched parentheses are detected inside a token, we extend the token until parentheses become valid. After the merging process, the ME in Figure 8 becomes complete.

Stage 3: Displayed ME Detection

A line is identified as a displayed ME if it satisfies one of the two rules: 1) the line contains ME symbols and starts or ends with an equation number in the following format: "(x)" where "x" is an integer, and a gap exist between the equation number and the rest symbols; 2) the line contains only ME tokens or white-listed math words such as ("max", "exp", "mean", "sin", etc.), without other common natural language words. For implementation, we used Pattern [60] and Wordnet lemmatizer [61] to normalize the words into their root form, and then match them with the natural language corpus from the Natural Language Toolkit (NLTK) [62].

A lengthy displayed ME may take more than one line. To merge these equations across lines, we check if the beginning or the ending symbol of a displayed ME is a math operator. If so, we merge this line with its neighbor line into one displayed ME. An example is shown on the first row in Table 3. Fraction line, which is often encoded as graphic elements, can cause false split of an equation, as the second example in Table 3. In this case, the displayed ME was falsely labeled into 3 different MEs. This issue can be readily solved by merging lines whenever they are overlapped (line gap<0). A drawback of this rule is that false merging can occur when a displayed ME overlaps with a plaintext line, but these cases are rare. Another scenario is that very often the binding variables of large math operators are falsely detected as separate MEs, as shown on the third row in Table 3. To identify these binding variables, we evaluate the neighbor lines of every displayed ME. The rules for a neighbor line to be binding variables is empirically derived as follows: 1) this line is bounded by the displayed ME on the horizontal-axis, and 2) the height of this line is smaller than the average line height or the line gap between this line and the displayed ME is less than the average line gap. If such a line were detected, we merge it with the displayed ME line.

Table 3 Examples for the displayed ME detection. Green boxes indicate the detected NME. Red Boxes indicate the detected inline ME. Blue boxes indicate the detected displayed ME.

Steps	**Before**	**After**
Multi-line *ME*	 $[f(s)]_0^1 - [f(r)]_0^1 \le \lvert f(s)-f(r)\rvert = \lvert T^{-1}\left(\frac{1}{2}(T(1)+\alpha)-\alpha s\right) - T^{-1}\left(\frac{1}{2}(T(1)+\alpha)-\alpha r\right)\rvert = \lvert\frac{1}{T'(u)}(\alpha s-\alpha r)\rvert \le \frac{\alpha}{\min_{u\in[0,1]}\tau(u)}\lvert s-r\rvert$	 $[f(s)]_0^1 - [f(r)]_0^1 \le \lvert f(s)-f(r)\rvert = \lvert T^{-1}\left(\frac{1}{2}(T(1)+\alpha)-\alpha s\right) - T^{-1}\left(\frac{1}{2}(T(1)+\alpha)-\alpha r\right)\rvert = \lvert\frac{1}{T'(u)}(\alpha s-\alpha r)\rvert \le \frac{\alpha}{\min_{u\in[0,1]}\tau(u)}\lvert s-r\rvert$
Fraction *Line*	 $\frac{\partial SS_E}{\partial \mathbf{b}} = 2X'\mathbf{y} - 2X'X\mathbf{b} = 0 \quad \text{or} \quad X'\mathbf{y} = X'X\mathbf{b}$	 $\frac{\partial SS_E}{\partial y_i} = 2y_i - 2X_i\mathbf{b} = 0 \quad \text{or} \quad y_i = X_i\mathbf{b}$
Binding *Variables*	 $P(M;\Theta) = \sum_{i=1}^{N} \pi(M-M_i)P_i(\Theta)$	 $P(M;\Theta) = \sum_{i=1}^{N} \pi(M-M_i)P_i(\Theta)$

Bigram Regularization Model for Inline ME Detection

Displayed MEs are easier to detect because of their unique spatial layout with respect to other plaintext and inline MEs. Inline MEs are more difficult to detect because of their unrestricted usage of fonts and blurred boundaries with plaintext. Figure 9 describes an inline ME scenario, which is the focus of this paper. The false and miss rate for state-of-the-art inline ME detection is about 10% [21], which is still unsatisfactory. A major issue is that many inline MEs are detected as multiple parts, as shown by the blue highlights in Figure 9.

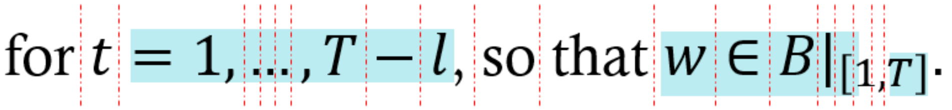


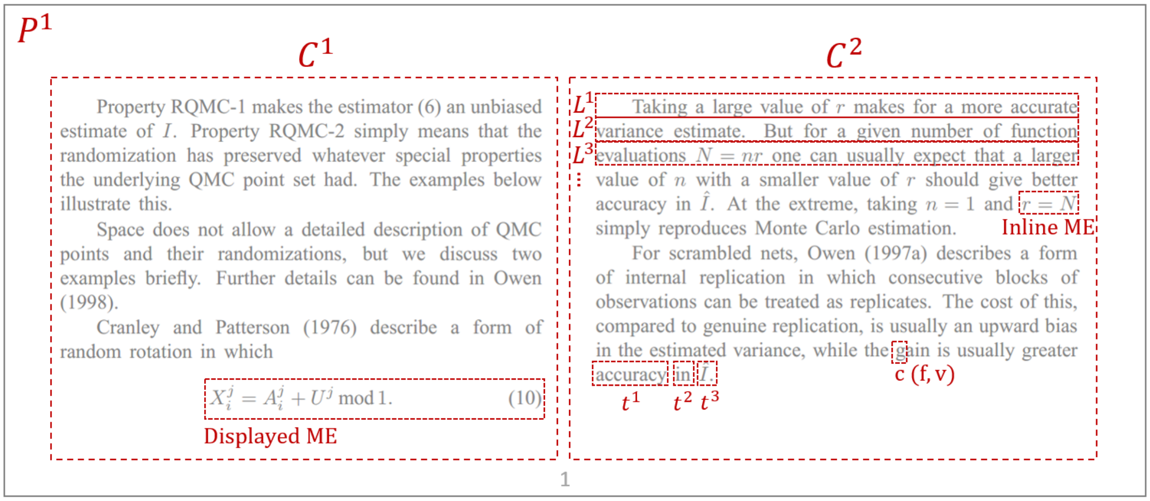
Figure 9 An example of split detections of inline MEs. The parts that are detected successfully are marked in blue. The red lines mark the boundaries of tokens.

Inline MEs are usually identified on the word level, or more precisely, token level. A *token* refers to a group of neighboring characters that are spatially close to each other (boundaries of tokens are marked by red lines in Figure 9). A token typically corresponds to either a natural language word, or a part of an ME. Unfortunately, existing works on ME extraction [12, 21, 59] are mainly pointwise solutions, which means these works only treated tokens as independent units without systematically incorporating their neighboring information. The issue of the pointwise strategies is that the extracted MEs are prone to high split ratios, as MEs could be composed of multiple tokens, but the misidentification of a single token would result in a split detection. This problem is especially common when plaintext is used inside the ME. We observed two facts indicating that neighboring information could be useful to reduce the high split detection ratios: 1) misclassified tokens often have blurred classification boundaries, i.e., similar posterior probability for being an ME or Non-ME (NME); 2) the neighboring tokens of a misclassified token are often detected correctly.

Inspired by the pairwise potentials concept commonly used in the Markov Random Field (MRF) algorithm, this paper proposes a bigram regularization model that utilizes the neighboring information in bigram tokens to fix the misidentified ME labels and reduce the split ratio. Next, we first present the formulation of the inline ME detection. Second, we illustrate some observations on the split cases and show the necessity for bigram modeling. Third, we present the formulation of an objective function that incorporates both unigram and bigram terms. Fourth, we transformed the objective function into a Mixed Integer Programming (MIP) problem. Finally, we explain an implementation with the Bayesian model to derive the ME likelihood for unigram tokens and show a detailed case study of the bigram regularization.

Document Model and Problem Formulation

We formulate a PDF document as different hierarchical elements, as shown in Figure 10. A PDF document D_i is consisted of pages $\{P_i^j\}$, where each page P_i^j is composed of columns $\{C_{i,j}^k\}$. A column $C_{i,j}^k$ contains lines $\{L_{i,j,k}^l\}$, where each line could stand for a displayed ME or a mix of inline ME and plaintext. Each line $L_{i,j,k}^l$ is composed of characters which could be organized as a sequence of tokens, $(t_{i,j,k,l}^1, t_{i,j,k,l}^2, \ldots)$. A token could either be a plaintext word or part of an ME, where each character $c \in t$ is associated with glyph name value v_c, font f_c, etc.



Property RQMC-1 makes the estimator (6) an unbiased estimate of I. Property RQMC-2 simply means that the randomization has preserved whatever special properties the underlying QMC point set had. The examples below illustrate this.

Space does not allow a detailed description of QMC points and their randomizations, but we discuss two examples briefly. Further details can be found in Owen (1998).

Cranley and Patterson (1976) describe a form of random rotation in which

$$X_i^j = A_i^j + U^j \bmod 1. \qquad (10)$$

Taking a large value of r makes for a more accurate variance estimate. But for a given number of function evaluations $N = nr$ one can usually expect that a larger value of n with a smaller value of r should give better accuracy in $\hat{I}$. At the extreme, taking $n = 1$ and $r = N$ simply reproduces Monte Carlo estimation.

For scrambled nets, Owen (1997a) describes a form of internal replication in which consecutive blocks of observations can be treated as replicates. The cost of this, compared to genuine replication, is usually an upward bias in the estimated variance, while the gain is usually greater accuracy in $\hat{I}$.

1

Figure 10 The formulation of a PDF page. The red rectangles mark different elements, including columns, lines, tokens, characters, and examples for inline ME and displayed ME.

The inline ME identification will be performed on the lines that are not identified as displayed MEs. Given such a line $L = \{t^1, \dots t^{N_L}\}$, the goal is to predict inline ME label sequence $y = \{y^1, \dots, y^{N_L}\}$, where the subscript is omitted for convenience. N_L is the number of tokens in the line. $y^i \in \{0,1\}$ is the ME label, where 0 stands for NME and 1 stands for ME. The inline ME identification is performed on each line separately. MEs across multiple lines are left separate because the goal is only to find the ME bounding boxes.

Why Bigram Model?

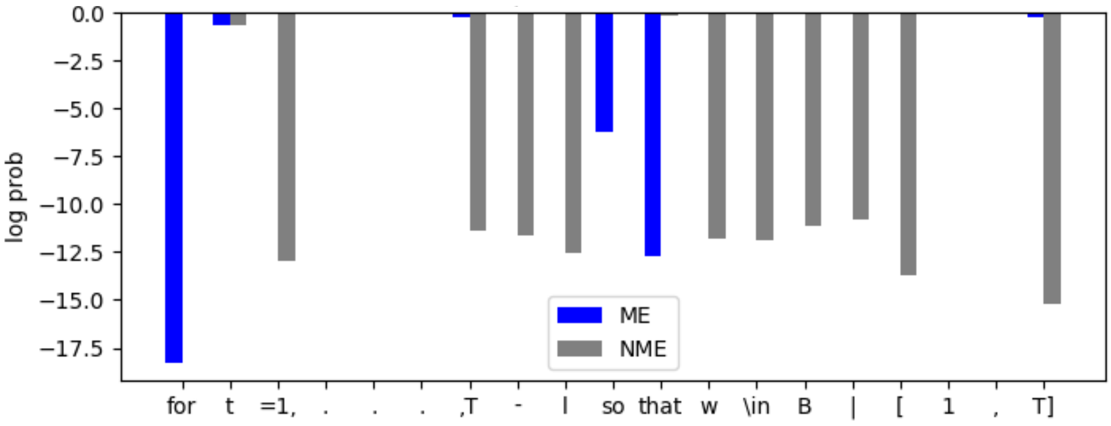


Figure 11 ME likelihood of tokens in a sentence mixed with natural language words and MEs. We took the log of the probabilities for better scalability.

Here we use a case study to demonstrate the necessity for bigram modeling. Figure 11 shows a detailed analysis of the sentence in Figure 9. The x-axis shows the sequence of tokens, while the y-axis shows the log of ME/NME likelihood for each token t^i. The likelihood is derived from the Bayesian model [21]. We observe that the plaintext words (“for”, “so”, “that”) have a large log probability for NME as compared to ME. On the contrary, most ME parts have a large log probability for ME as compared to NME. There are less determinant zones such as “*t*”, “.”, “,”, “*1*”. The over split issue happens because the tokens “*t*”, “*1*”, and “,” in the less determinant zone are misclassified as NME. We further observe that the labels of their direct neighbors are classified correctly. This indicates that a bigram model that utilizes the neighboring information would help correct these misclassifications and hence reduce split detections.

The Bigram Regularization

In this section, we will first brief the unigram (single token) decision model, and then show how neighboring information is incorporated as the bigram regularization term to improve the inline ME decision process.

From the view of a unigram decision process, the goal is to assign the most likely labels y^i to each of the observed tokens t^i, i.e., maximizing the posterior probability $\prod_{i\in[1,N_L]} P(y^i|t^i)$. This is equivalent to minimizing the negation of summation of log probability $-\sum_{i\in[1,N_L]} \log\,(P(y^i|\mathrm{t}^i))$, which can be further formulated as:

$$U(y) = -\sum_i y^i\, log\left(P_{ME}(t^i)\right) + \left(1-y^i\right) log(P_{NME}(t^i))$$

where $P_{[N]ME}(\mathrm{t}^i)$ could be obtained from a classification model, e.g., the Bayesian model [21] or the SVM [12]. For convenience, we write

$$U(y) \equiv -\sum_i y^i\, log\,(LR(t^i))$$

where

$$LR(t^i) = P_{ME}(t^i)/P_{NME}(t^i)$$

$U(y)$ represents the unary potentials. The unary term alone indicates that each token t is treated as independent from others. However, we observed that this is not sufficient because the neighboring tokens could contain useful information to correct the misclassified inline ME labels. In order to incorporate the neighboring information, we propose to add a pairwise term into the objective function. The pairwise term acts as bigram regularization which prefers the label y^i to be similar to its direct neighbors y^{i-1} and y^{i+1}. Mathematically, we add a penalty $P(y)$ to account for label change:

$$P(y) = \sum_{i \in [1, N_L)} \left| y^i - y^{i+1} \right|$$

By merging the unary and pairwise terms, we have the following objective function:

$$f(y) = U(y) + \lambda P(y)$$

where $\lambda > 0$ is a weight parameter for the pairwise penalty.

Regularization Solver Design

Since the pairwise term in the objective function is in absolute value$\left| y^i - y^{i+1} \right|$, $i \in [1, N_L)$, the optimization problem becomes non-linear. Fortunately, we can transform the objective function into a linear function by introducing two auxiliary variables.

Let z_+^i and z_-^i be two auxiliary variables with the following constraint set $\mathcal{C}$:

$$\begin{cases} z_+^i + z_-^i = \left| y^i - y^{i+1} \right| \\ z_+^i - z_-^i = y^i - y^{i+1} \\ z_+^i, z_-^i \in \{0,1\} \end{cases}$$

This way, the objective function is transformed into the following form:

$$f(y, z) = -\sum_{i \in [1, N_L]} y^i \, log\left(LR\left(t^i\right) \right) + \lambda \sum_{i \in [1, N_L)} (z_+^i + z_-^i)$$

subject to the constraint set $\mathcal{C}$ and $y^i \in \{0, 1\}$.

This is essentially a Mixed-Integer Programming (MIP) problem, which could be solved by a linear integer solver. In our implementation, we used the Simplex algorithm in the CyLP software package [63] to solve this function.

Scenario Analysis for the Bigram Regularization

We will study two scenarios based on the example in Figure 11. We first focus on this segment of the sentence: [“*[*”, “*1*”, “*,*”, “*T]*”]. The likelihood for these four tokens is roughly [-14, 0, 0, -15]. The values of the objective function under different predicted labels are enumerated in Table 4.

Table 4 The value of the objective function under different label assignments for token sequence [“[”, “1”, “,”, “T]”].

Label	Objective value	Reduced
[1,0,0,1]	$1*-14+0*0+0*0+1*-15+2\lambda$	$-29+2\lambda$
[1,0,1,1]	$1*-14+0*0+1*0+1*-15+2\lambda$	$-29+2\lambda$
[1,1,1,1]	$1*-14+1*0+1*0+1*-15$	-29

From the table, we can see that if we assign label 0 (NME) to the less determinant tokens (“*1*” and “*,*”), which are between two highly determinant ME (“*[*” and “*T]*”), a penalty of 2λ will be introduced. As long as $\lambda > 0$, the objective function would be minimized when labels [1,1,1,1] are chosen, which means the bigram regularization successfully solves the over split issue in this scenario.

On the other hand, we should not set λ too high, as it would blur the boundaries between ME and NME. To explain this, we look into the bigram [“*that*”, “*w*”] in Figure 11. The log likelihood for these two tokens are roughly [13, -12]. Again, we enumerate the objective function value under different labeling situation in Table 5 below:

Table 5 The value of the objective function under different label assignments for token sequence [“that”, “w”].

Label	Objective value	Reduced
[0,0]	$0*13+0*-12$	0
[0,1]	$0*13+1*-12+\lambda$	$-12+\lambda$
[1,0]	$1*13+0*-12+\lambda$	$13+\lambda$
[1,1]	$1*13+1*-12$	1

This time the ground truth label is [0,1], which corresponds to an objective function value of $-12+\lambda$. In this case, if λ is set to larger than 12, the best prediction becomes [0,0], which would introduce false negatives into the results. Thus, we need to select the value of λ carefully during the decision process. We will present a more detailed analysis in the experiment section on the effect of the parameter λ.

Experiments and Results

The Marmot Dataset

We used the Marmot dataset [59] to evaluate our algorithms. The dataset contains 400 single PDF pages collected from the CiteSeerX digital library. Each page has human-labeled ground truth BBoxes for each MEs. The original dataset contains 1575 displayed MEs and 7907 inline MEs. We also included the 1888 additional MEs annotated in [64] as they are missing from the original ground truth (e.g., in page “*10.1.1.161.9629_10*” alone, 23 inline MEs are missing from the ground truth data.). MEs inside the figure areas are excluded from the evaluation process because they are not labeled in the ground truth data.

Experiment Setup

We will first report the performance on the symbol-level ME detection (*Stage 0*), and then the detailed evaluation on the ME-level detection. For comparison purposes, we implemented the FSB model [21] and compared it with our method.

The evaluation is based on the overlapping of BBoxes. Let M_{gt} be the set of ground truth BBox, M_c be the set of predicted ME symbol BBox, and M_{pd} be the set of predicted ME BBox. The relationship between two BBoxes could either be fully overlapped (OL), fully separated (SP), contained (CT), or partially overlapped (OS). The OL relationship required two BBoxes to have a common area larger than 95%. The OS relationship indicates a common area smaller than 95%.

For symbol-level detection, we report the performance using the three metrics defined below:

- Correct: $\exists m_{gt} \in M_{gt}, rel(m_c, m_{gt}) = CT$
- False: $\forall m_{gt} \in M_{gt}, rel(m_c, m_{gt}) = SP$
- Miss: a symbol is in M_{gt} but not in M_c

For ME-level detection, we report the evaluation metrics defined in [21]. The detailed metrics include correct (Cor), miss (Mis), false (Fal), partial (Par), expanded (Exp), partial and expanded (Pae), merged (Mer), and split (Spl). The general metrics include precision, recall, and F1-score. We report the ME-level performance on inline ME and displayed ME separately, and then the two combined.

Results and Discussions

Symbol-level Performance

Table 6 Symbol-level evaluation results

	Correct	False	Miss
Stage 1.1	69,716	3,902	44,695
Stage 1.2	46,269	1,601	68,142
Stage 1.1+1.2	81,081	4,532	33,330

We first show the detailed performance of *Stage 1*. *Stage 1* consists of two sub-steps: *Stage 1.1*) font size-based likelihood ratio test and *Stage 1.2*) font name and glyph name matching. Here we report the performance of *Stage 1.1* and *Stage 1.2* individually, and the two combined. Table 6 shows the statistics of the evaluation data. In total, there are 114,411 ME symbols in the dataset. *Stage 1.1* alone gives us precision 94.7% and recall 60.9%. *Stage 1.2* alone gives us precision 96.7% and recall 40.4%. With *Stage 1.1* and *Stage 1.2* combined, we get precision 94.7% and recall 70.9%. This result shows that most of the ME symbols are captured by the likelihood ratio test (*Stage 1.1*). While font name and glyph name gives a much lower recall rate, they complement the font size information. Still, we have a non-negligible amount of miss detections even when the two sub-steps are combined. The main cause is the miss detections on plaintext, numbers, and parentheses inside MEs. We will show that the following stages significantly reduce the miss rate.

ME-level Performance

Table 7 Detailed performance statistics for ME-level detection

	Cor	Mis	Fal	Par	Exp	Pae	Mer	Spl
Displayed ME (FSB)	839	208	473	970	57	33	0	31
Displayed ME (Ours)	1181	9	109	238	128	35	2	2
Inline ME (FSB)	4192	829	2290	2971	1491	800	2	2
Inline ME (Ours)	5598	810	776	2580	651	397	5	0

Table 7 summarizes the detailed performance statistics. Compared to the baseline method, our algorithm improves the performance significantly especially on *Cor*, *Fal*, and *Pae*. A large number of expanded boxes and partially detected boxes from the FSB model becomes perfect matches (95% coverage) with our algorithm. Also, false positives are greatly reduced. Next we compare precision, recall, and F1-score. On displayed ME, our algorithm achieves precision 93.6%, recall 99.4%, and F1 96.4%. In comparison, the FSB model has precision 80.3%, recall 90.3%, and F1 85.0%. On inline ME, our algorithm gives precision 92.2%, recall 91.9%, and F1 92.1%. The FSB model has precision 80.5%, recall 91.9%, and F1 85.8%. By combining the two types of MEs together, our algorithm gives precision 92.4%, recall 93.0%, and F1 92.7%. The FSB model gives precision 80.5%, recall 91.7% and F1 85.7%.

Another similar study [13] based on the deep learning model reported 93.4% F1-score on the original Marmot dataset [59]. Knowing that the original Marmot dataset missed a number of cases [64], it would be useful to rerun the experiment with the amended ground truth data to assess its robustness.

Computation Cost

Our experiment was done on a PC with Intel Xeon 3.5Ghz CPU, 16GB RAM. On the Marmot dataset, our algorithm takes an average of 1.09s to extract MEs from one page. The FSB model is a weakly-supervised model, which takes an average of 4.48s to process a page on the same machine. Supervised machine learning methods take longer to train models and make predictions. Take [23] for example, it takes 763 seconds to train word classifiers, and takes about 10 seconds to predict a word.

The MOP Dataset

Knowing that there are large amount of labeling errors in the Marmot dataset, we also developed a more accurate and larger dataset MOP [65] based on a semi-automated LaTeX-based labeling system. MOP dataset contains 1,802 PDF pages, each corresponding to the ground truth bounding boxes of math expressions. The overall semi-automatic pipeline used to generate the dataset is shown in Figure 12. The pages in the MOP dataset are constructed from hep-th papers and their LaTeX sources on arXiv. PDF pages are processed with our PDF parser to output a string of glyph names. LaTeX source files are processed with LaTeXML [66] software to output strings and math expressions with identifiers. The math expressions in PDF are located by matching the strings in the two types of outputs. The bounding boxes are automatically generated and visually inspected to guarantee correctness. The visual inspection is done by a research personnel as a Boolean correction manner – pages with errors are discarded, while pages without errors are kept. In total, there are 1,802 pages generated for this dataset, containing 10,486 MEs and labels of their bounding boxes. On average, there are 28.4 MEs per page. This is more math-dense than the Marmot dataset. At the same time, the MOP dataset also contains more diverse pages, such as those with references. In comparison, pages in Marmot are more uniform in that they are all selected as the central parts of papers.

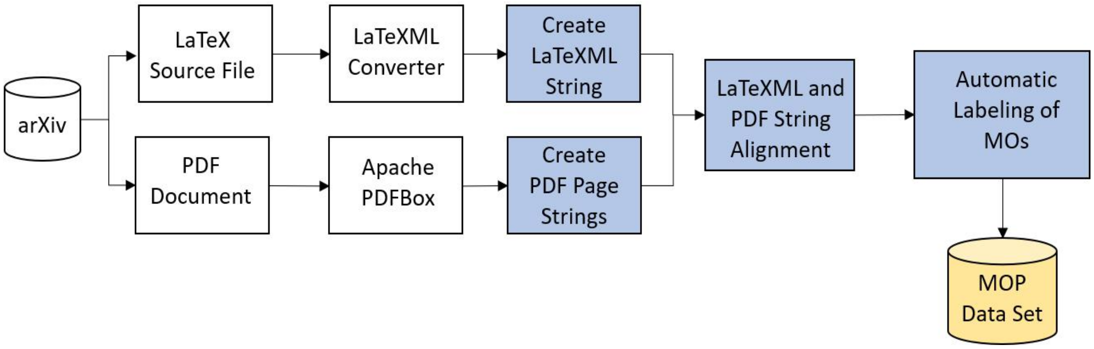


Figure 12 The semi-automatic pipeline used to generate the MOP dataset.

We also tested our multi-stage ME extraction algorithm on this dataset. The evaluation results serve as a benchmark of this dataset. In Table 8, we show the detailed comparison of the performance of the algorithm on this dataset and on the Marmot dataset. As we can see, the F1-score on the two datasets are similar, which suggests that the algorithm performs relatively robust across different types of papers. However, we also observe that the precision on the MOP dataset is noticeably lower. This is due to the diversity of the MOP dataset. For example, the URL links in the references could introduce false positives in the proposed algorithm, because they can also introduce font size changes. On the other hand, the recall score is higher on the MOP dataset, which suggests that the font-based features in the proposed algorithm suits better for pages generated with the LaTeX math mode.

Table 8 Benchmark of the MOP dataset

Data Set	F1	Precision	Recall
MOP	91.8%	86.4%	97.8%
Marmot	93.1%	93.1%	90.5%

Summary

In this chapter, we presented a multi-stage algorithm to detect math expressions from PDF documents. We first extracted metadata from PDF objects and identified the font size as an effective feature. Then we used a grouping method based on the font size feature to generate labels and estimate the likelihood for being ME/NME. Symbol-level detection is done by likelihood ratio test and font name and glyph name matching. Subsequent steps merged ME symbols into inline/displayed MEs based on spatial and semantic heuristics. Next, we proposed a bigram label regularization model to solve the split detection issues during inline ME extraction from scientific publications. The model is composed of a unary term that uses the unigram ME likelihood information, and a pairwise term that incorporates the bigram neighbouring information. The bigram regularization model can greatly reduce the over split issue, which is very important in the later stages of ME parsing. The case study also showed the model's interpretability. The bigram regularization model also serves as a proof-of-concept to incorporate other types of neighbouring constraints and penalties. The algorithm could serve as the foundation to applications such as PDF tagger, structural and semantic analysis on MEs, math-based information retrieval, etc.

CHAPTER IV

MI2LATEX: MATH FORMULA IMAGES TO LATEX TRANSLATION BASED ON DEEP NEURAL NETWORKS*

In this chapter we propose a deep neural network model with an encoder-decoder architecture that translates images of math formulas into their LaTeX markup sequences. The encoder is a convolutional neural network (CNN) that transforms images into a group of feature maps. To better capture the spatial relationships of math symbols, the feature maps are augmented with 2D positional encoding before being unfolded into a vector. The decoder is a stacked bidirectional long short-term memory (LSTM) model integrated with the soft attention mechanism, which works as a language model to translate the encoder output into a sequence of LaTeX tokens. The neural network is trained in two steps. The first step is token-level training using the Maximum-Likelihood Estimation (MLE) as the objective function. At completion of the token-level training, the sequence-level training objective function is employed to optimize the overall model based on the policy gradient algorithm from reinforcement learning. Our design also overcomes the exposure bias problem by closing the feedback loop in the decoder during sequence-level training, i.e., feeding in the predicted token instead of the ground truth token at every time step. The

*Reprinted with permission from "Translating Math Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training" by Wang, Z., Liu, J. C. in *arXiv (2019): arXiv-1908*, Copyright 2019 arXiv.

model is trained and evaluated on the IM2LATEX-100K dataset and shows state-of-the-art performance on both sequence-based and image-based evaluation metrics.

Overview

Math formulas often carry the most significant technical substances in many science, technology, engineering and math (STEM) fields. Being able to extract the math formulas from digital documents and translate them into markup languages is very useful for a wide range of information retrieval tasks. Portable Document Format (PDF) is the *de facto* standard publication format, which makes document distribution very easy and reliable. Although math formulas can be recognized by human readers relatively easily, computer-based math formula recognition in PDF documents remains a major challenge. This is mainly because the PDF format does not contain tagged information about its math contents. Recognizing math formulas from PDF documents is intrinsically difficult because of the presence of unusual math symbols and complex layout structures. In addition, math formulas in PDF documents could partially be represented by blocks of graphics directly rendered from the PDF glyphs, which preserves the correct shapes but misses the meaning of contents. These problems would be readily solved if the markup sources of the PDF documents are available. A good example is the preprint repositories *arXiv.org* which gives readers access to the LaTeX source files along with the PDF files, but it only comprises a small fraction of the existing digital publications. For vast majority of digital documents, advanced techniques are needed to translate the PDF math contents into their markup sources. Being a structured math description language, LaTeX can be used to retrieve math formulas, and can be easily converted to other formats such as MathML [67] to support high-level applications.

With the earliest effort dating back to 1967 [68], different approaches have been developed to recover math contents with different levels of success. Recent advancement in optical character recognition (OCR) techniques has made it possible to recognize text in digital documents at high accuracy. However, recognizing math formulas is difficult, because on top of recognition of individual math symbols, it is also necessary to recognize the structural relationship among symbols, such as sub/sup-scripts, nested fractions, matrix, etc. Researchers have developed rule-based structural analysis methods and syntactic parsers to convert math formulas to their markup languages. One successful example is the INFTY system [8], which was designed to convert documents into structured formats like LaTeX, and was later made into a commercial software called InftyReader for digital document processing. With the rise of deep learning technology, it has been demonstrated that hand-crafted features and rules can now be replaced by learnable feature representations.

Translating math formula images to LaTeX sequences is a joint field of image processing and text processing, which has recently gained increased research interest in the deep learning community [30-32]. The sequence-to-sequence model (seq2seq), also called the encoder-decoder architecture, has been successfully applied to intersect these two fields. The encoder for such applications is usually a convolutional neural network (CNN) which encodes the input images as abstract feature representations, and the decoder is usually a recurrent neural network (RNN) that represents a language model to translate the encoder output into a sequence of *tokens* drawn from a vocabulary. This architecture makes the size of input images and output sequences flexible, and could be trained in an *end-to-end* fashion. Seq2seq model has been successfully used in image captioning [31, 32] and scene text recognition [30] tasks, which shares similar technical requirements with that of the image to LaTeX task. Recently, the authors in [38] successfully applied an attention-based seq2seq model to translate images to LaTeX, which demonstrated the model's capability of handling structural contents.

Leveraging the previous success, in this chapter we propose a new seq2seq model called MI2LS (Math Image to LaTeX Sequence) which focuses on addressing three key problems that have not been investigated in prior works. Firstly, to help the model better differentiate the 2-dimensional spatial relationship of math symbols, we propose to augment the image feature maps by adding sinusoidal positional encoding for richer representation of spatial locality information. Secondly, we propose a sequence-level objective function based on the BLEU (bilingual evaluation understudy) [69] score, which could better capture the interrelationship among different tokens in a LaTeX sequence than the token-level cross-entropy loss. Knowing that the sequence-level evaluation score is discrete and non-differentiable, we propose to solve the optimization problem based on the policy gradient algorithm [70] in reinforcement learning for model training. Thirdly, we eliminate the exposure bias [71] problem by closing the feedback loop during the sequence-level training, i.e., feeding back the predicted token instead of the ground truth token for the next time step. This is made possible because the token alignment problem in token-level training no longer exists in sequence-level training. The overall model architecture includes a CNN encoder, an RNN decoder, and a soft attention mechanism, as shown in Figure 13. The model was trained and evaluated on the IM2LATEX-100K dataset [38], and achieved state-of-the-art performance on both the BLEU score and image similarity measurements.

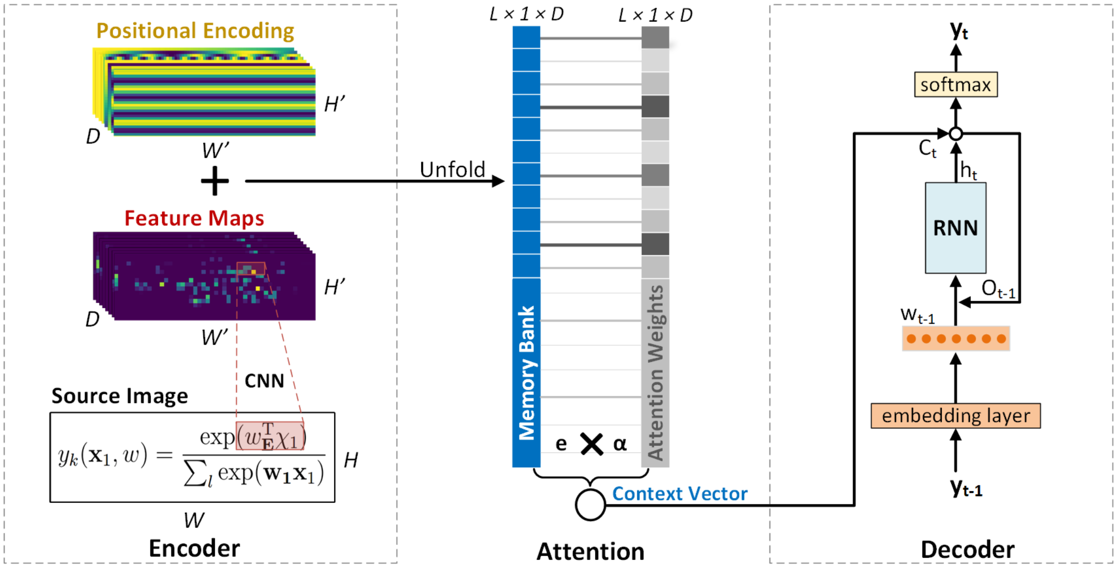


Figure 13 The proposed encoder-decoder architecture of the deep neural network.

Neural Network Architecture

In this section, we first present the formulation of the problem. Next, we introduce the proposed seq2seq architecture as shown in Figure 13, and explain the encoder, which is a convolutional neural network augmented with positional encoding, and the decoder, which is a stacked bidirectional long short-term memory (LSTM). In the end, we explain the soft attention mechanism.

Problem Formulation

The math formula recognition problem is formulated as a sequence prediction problem. Let (x, y) be an image-LaTeX sequence pair. $x \in \mathbb{R}^{H \times W}$ is a grayscale image with height H and width W. $y = [y_1, y_2, \ldots, y_T]$ is the ground truth LaTeX sequence consisting of T tokens that marks up the math formula in the image. x can be rendered by y using the standard TeX compiler. The goal of our task is to recover y given the input image x, i.e., to find a mapping function f so that $f(x) \rightarrow y$. Given a set of N image-LaTeX ground truth pairs $G = \{x^i, y^i\}_{i=1}^{N}$, we use supervised training to build a sequence prediction function $\hat{f}$ that approximates f. During the test time, we use $\hat{f}(x) \rightarrow \hat{y}$ to predict a LaTeX sequence $\hat{y}$ that reconstructs the input image x. Evaluation is done by measuring the similarity between $\hat{y}$ and the ground truth sequence y, and between the rendered image $\hat{x}$ and the ground truth image x.

Encoder

The encoder is used to encode the input images into abstract feature representations. It is composed of a convolutional neural network (CNN) and positional encoding.

Convolutional Neural Network

We use a CNN to extract features from the input images. CNN is consisted of convolution, pooling and activation layers. At each convolution layer, an input image is convolved with a set of kernels, which act as image filters. The kernel values are trainable, which makes the image features data-driven instead of hand-crafted. The pooling layer is usually composed of a max pooling function or average pooling function, which reduces the image size and increases the size of the receptive field. The activation layer adds nonlinearity to the neural network. It is usually a Rectified Linear Units (ReLU) that replaces negative inputs with 0 and keeps the positive inputs unchanged. We use a CNN architecture based on the VGG-VeryDeep that has been adapted particularly for OCR applications [34]. Details of the CNN configuration can be found in Table 9. The feature maps are convolved to a 2D matrix instead of a flattened feature vector in order to retain the spatial locality information, as shown in Figure 13. This practice also allows the model to accept input images of arbitrary size. As a result of the CNN configuration, both the width W' and height H' of the output feature maps are 8 times smaller than that of the input image, and each position is D dimensions deep ($D = 512$ in our implementation).

Table 9 The encoder CNN configurations. #*maps*: the number of feature maps. *k*: kernel size. *p*: padding size. *s*: stride size. *BN*: batch normalization. The sizes are in order (height, width).

Type	#maps	k	p	s
BN	-			
Convolution	512	(3,3)	(1,1)	(1,1)
MaxPooling		(2,1)	(0,0)	(2,1)
BN	-			
Convolution	512	(3,3),	(1,1)	(1,1)
MaxPooling		(1,2)	(0,0)	(1,2)
Convolution	256	(3,3)	(1,1)	(1,1)
BN	-			
Convolution	256	(3,3)	(1,1)	(1,1)
MaxPooling		(2,2)	(0,0)	(2,2)
Convolution	128	(3,3)	(1,1)	(1,1)
MaxPooling		(2,2)	(0,0)	(2,2)
Convolution	64	(3,3)	(1,1)	(1,1)
Input	Gray-scale image			

Positional Encoding

For text recognition, one could simply unfold the feature maps from the encoder to an array and feed it into an RNN decoder without explicitly considering spatial localization, because RNN is capable of capturing left-to-right location ordering. However, in math formulas, the spatial relationship among symbols span along different directions: left-right, top-down, sub/sup-scripts, nested, etc. The positional relationships among math symbols carry critical math semantics. As such, special efforts to preserve spatial locality are necessary. Here we tailor the 1-D positional encoding technique proposed in the Transformer model [9] to 2-D as follows:

$$PE(x, y, 2i) = \sin(x/10000^{4i/D})$$
$$PE(x, y, 2i + 1) = \cos(x/10000^{4i/D})$$
$$PE(x, y, 2j + D/2) = \sin(y/10000^{4j/D})$$
$$PE(x, y, 2j + 1 + D/2) = \cos(y/10000^{4j/D})$$

where x and y specifies the horizontal and vertical positions, and $i, j \in [0, D/4)$ specifies the dimension. These signals are added to the feature maps.

The positional encoding has the same size and dimension as the feature maps. Each dimension of the positional encoding is composed of a sinusoidal signal of a particular frequency and phase, representing either the horizontal or the vertical directions. We use a timescale ranging from 1 to 10000. The number of different timescales is equal to $D/4$, corresponding to different frequencies. For each frequency, we generate a sine/cosine signal on the horizontal/vertical direction. All these signals are concatenated to D dimensions. The first half of the dimensions encodes the horizontal positions, and the second half encodes the vertical positions.

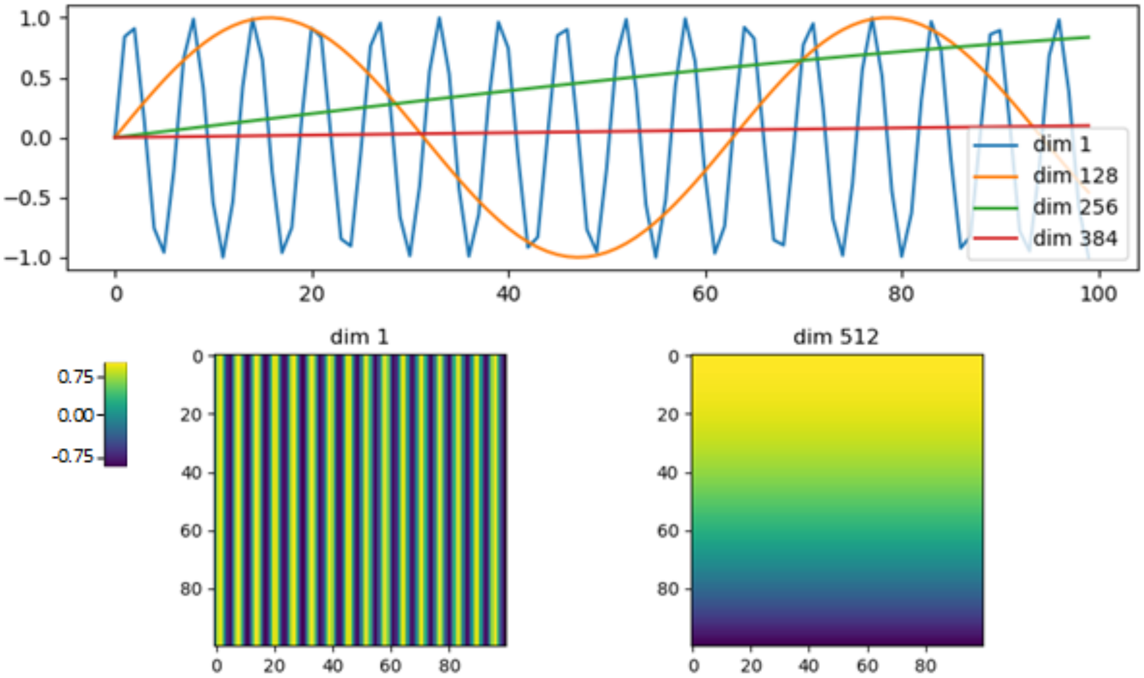


Figure 14 Visualization of the positional encoding.

In Figure 14 we show a visualization of the positional encoding when $D = 512$. The top half of the figure shows the positional encoding signals changing along one axis. The signals span from position 1 to 100 at dimension 1, 128, 256, and 384. The signal frequency decreases as the dimension number increases. The bottom half shows the positional encoding signals in 2D. The signals span from position (1,1) to (100, 100) at dimension 1 and 512. Dimension 1 encodes the horizontal positions, and dimension 512 encodes the vertical positions.

The positional encoding has the same size as that of the feature maps and is defined for every channel ($H'\times W'\times D$). The positional encoding and the feature maps are added together, and then unfolded into a 1-dimensional array $\vec{E} \in \mathbb{R}^{L\times 1\times D}$, where $L = H'\times W'$ is the length of the array. Each vector $e_i \in \vec{E}$ has a dimension of D, which is the feature size. Each such vector corresponds to a certain part of the input image. Note that this position encoding technique has the advantage of not adding new trainable parameters to the neural network. Furthermore, compared to trainable positional embedding, sinusoidal encoding can be scaled to lengths that are unseen in the training data.

Decoder

RNN is well suited for sequence prediction tasks, because it maintains a history of the previous predictions and is able to traverse from the start to the end of sequence at arbitrary length. Let $\vec{V}$ be the vocabulary that contains all the permissible LaTeX tokens. We use an RNN to approximate a language model $p(y_t|y_1, \dots, y_{t-1}, \vec{E})$, which makes a prediction on the probability distribution of the token $y_t \in \vec{V}$ at time t based on the prediction history $y_{i<t}$ and the encoder output $\vec{E}$. Next we introduce the token representation and the structure of the RNN.

Token Embeddings

A LaTeX token refers to a basic processing unit within a LaTeX sequence. The LaTeX source of a formula is first split into a sequence of tokens $y_1, y_2, \dots, y_T$ (details of LaTeX tokenization can be found in the next section). A token can be fed into the RNN in different representations. One straightforward option is to represent each token as a *one-hot* vector, which implies that tokens are orthogonal to each other, and thus it may miss important language semantics. Similar to natural language words, many LaTeX tokens are interrelated. For example, '{' and '}' may have a higher correlation because they need to be used in pair as defined in the LaTeX grammar. As a result, we propose to add a *word* *embedding* [72] layer commonly used in NLP, where a token y_t is projected into a high-dimensional vector w_t:

$$w_t = embedding(y_t)$$

This embedding is trainable and is able to capture the interrelationship between different tokens [72].

Stacked Bidirectional LSTM

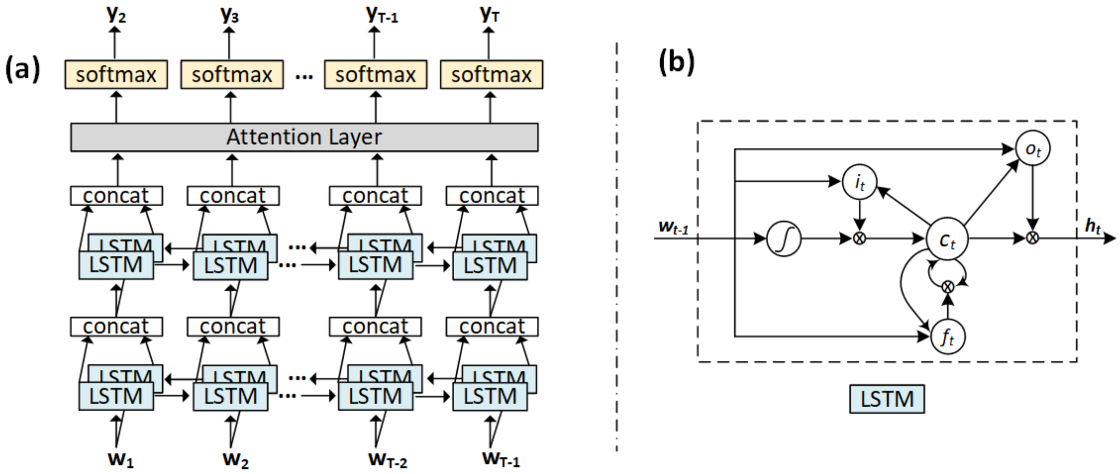


Figure 15 (a) The structure of the stacked bidirectional LSTM with attention layer. (b) The structure of an LSTM cell, where $\boldsymbol{i}$, $\boldsymbol{f}$, $\boldsymbol{o}$ represent input gate, forget gate, and output gate separately.

We propose to use a decoder model based on two layers of bidirectional long-short term memory (LSTM) cells [73]. Stacking multiple layers of LSTM increase the depth of the RNN and thus helps to capture more complex language semantics. Using bidirectional cells in each layer helps to capture the contexts from both forward and backward directions between tokens. Figure 15 (a) shows the structure of the stacked bidirectional LSTM that we used. For convenience, we will simply refer to this network as RNN henceforth.

LSTM is more capable of handling long sequences than the standard RNN, which is subject to the vanishing gradient problem [74] with the growth of sequence length. Figure 15 (b) shows the structure of an LSTM cell. The core to the LSTM is the cell state c_t that records the information that has been observed at time t. The LSTM is capable of adding or removing information from the cell state via three types of gates: input gate i_t, forget gate f_t, and output gate o_t. As implied by their names, these gates control *read* of the current input, *forget* of the current cell state value, or *output* of the current cell value. Each gate is comprised of a sigmoid neural network layer and a pointwise multiplication, expressed as below:

$$i_t = \sigma(W_{ix}w_{t-1} + W_{ih}h_{t-1})$$
$$f_t = \sigma(W_{fx}w_{t-1} + W_{fh}h_{t-1})$$
$$o_t = \sigma(W_{ox}w_{t-1} + W_{oh}h_{t-1})$$
$$c_t = f_t * c_{t-1} + i_t * \tanh(W_{cx}w_{t-1} + W_{ch}h_{t-1})$$
$$h_t = o_t * c_t$$

where h_t represents the RNN hidden state at time t, σ represents the sigmoid function, and W represents the weight matrix.

In NLP applications, the initial hidden state and cell state of the decoder is usually the output of the encoder RNN. However, in our model the encoder is a CNN which does not yield such an output. In order to derive informative initial states for the RNN decoder, we add additional layers to train the initial states based on the encoder output as below:

$$h_0 = \tanh\,(W_h \left(\frac{1}{L}\sum_{i=1}^{L} \vec{e}_i\right) + b_h)$$

$$c_0 = \tanh\,(W_c \left(\frac{1}{L}\sum_{i=1}^{L} \vec{e}_i\right) + b_c)$$

Attention

Theoretically, LSTM can be scaled up to capture long-term memory as needed. However, it is not uncommon that the markup of a complicated math formula extends to over a hundred LaTeX tokens. In such cases, an initial hidden state vector in RNN would be insufficient to compress all the information from the encoder. This problem is even more profound in our model because the CNN encoder does not have memory capability. The attention mechanism [35] has been introduced to solve this problem and has now become a widely adopted approach to enhance the performance on longer sequences. Basically, it maintains the complete encoder output, namely, the memory bank $\vec{E}$, based on which to calculate a context vector C_t for the decoder at every time step t. We adopt the soft attention mechanism, which means that the context vector C_t is calculated as a linear combination of the vectors $e_i \in \vec{E}$ in the memory bank:

$$C_t = \sum_{i=1}^{L} \alpha_{it} e_i$$

where α_{it} is the i^{th} weight at time t.

The attention weights are calculated with an additional feedforward layer by feeding in the previous hidden state of the LSTM h_{t-1} and the memory bank $\vec{E}$, and then pass it through a softmax layer for normalization:

$$a_{it} = \beta^T \tanh\,(W_1 h_{t-1} + W_2 e_i)$$
$$\alpha_{it} = \text{softmax}(a_{it})$$

where the softmax function is used to generate a probability distribution that sums up to 1, defined assoftmax$(a_{it}) = \exp(a_{it})\,/\sum_{k=1}^{L} \exp\,(a_{kt})$. The attention weights indicate which parts of the memory bank should be focused on at the current time step, thus helps the model better capture the salient parts of the input image.

To incorporate the context vector C_t information into the RNN, we compute another hidden state vector O_t based on the context vector C_t and the current hidden state h_t. O_t is called an attentional hidden state vector, which is fed back into the next time step of the RNN. It is also used to compute the probability distribution of the next token:

$$O_t = \tanh\left(W_3[h_t, C_t]\right)$$

We also adopt the input-feeding approach proposed in [35], in which the input embedding vector is concatenated with the attentional vector from the previous time step as the input for the RNN. This way, decisions are made by considering the past alignment information.

$$h_t = RNN(h_{t-1}, [w_{t-1}, O_{t-1}])$$

The prediction probability becomes:

$$p(y_t) = \text{softmax}\left(W_4 O_t\right)$$

which represents the probability distribution of the next token over the vocabulary $\vec{\mathrm{V}}$.

Training objectives

An ideal objective function should be constructed at the sequence level because of the rigorous nature of the LaTeX grammars. In addition, it is highly desirable that the objective function is differentiable for the backpropagation algorithm. In this section we will describe the design of a sequence-level objective function and techniques to compute its derivative based on the policy gradient algorithm. We note that it is infeasible to train the neural network with the sequence-level objective function from a random start, because the neural network may not converge under a poor random prediction policy. To overcome these challenges, we start off by training the neural network with a token-level objective function until it converges. This forms the initial state for the sequence-level training, as such the model can focus on a much smaller search space.

Token-level Objective Function

The objective function of the token-level training is based on the maximum likelihood estimation (MLE). Given a training dataset of image and LaTeX sequence pairs $\{x^i, y^i\}_{i=1}^N$ of size N, where x^i and y^i represents the i^{th} input image and ground truth LaTeX sequence respectively, the goal is to find a set of parameters θ that maximizes the log-likelihood of the training data:

$$\hat{\theta}_{MLE} = \underset{\theta}{\operatorname{argmax}}\{L_{MLE}(\theta)\}$$

where

$$L_{MLE}(\theta) = \sum_{i=1}^{N} p(y^i, x^i) = \sum_{i=1}^{N}\sum_{t=1}^{T} p(y_t^i | y_1^i, \dots, y_{t-1}^i, x^i)$$

This is equivalent to minimizing the cross-entropy loss (XENT):

$$L_{XENT}(\theta) = -\frac{1}{N}\left(\sum_{i=1}^{N} y^i \cdot \log{(\hat{y}^i)}\right)$$

where $\hat{y}^i$ is the prediction. The derivative of the cross-entropy loss can be directly used as the gradient.

The token-level objective function faces two limitations. Firstly, it maximizes the probability of the next correct token, without considering the sequence-level contexts governed by the LaTeX grammar. Secondly, to avoid the token misalignment problem, the ground truth token needs to be fed into the RNN at every time step during the training time, instead of using the RNN's previous prediction. At the prediction time, however, the previous prediction from the RNN is fed back as the next input since the ground truth data is no longer available. As a result, the probability distribution being trained on is $p(y_t|y_{i<t}, \vec{E})$, but the probability distribution being tested on is $p(y_t|\hat{y}_{i<t}, \vec{E})$. This discrepancy is known as the *exposure bias* [71] problem. The sequence-level training objective function aims to overcome these problems.

Sequence-level Objective Function

The formulation of a sequence-level training objective starts with its sequence-level performance metrics. Let (x^i, y^i) be the i^{th} training pair, and $\hat{y}^i$ be the prediction. Let $\mathrm{R}(y^i, \hat{y}^i) \rightarrow [0,1]$ be a function that maps the predicted sequence to a scalar reward, where a larger value indicates a better performance. $\mathrm{R}(y^i, \hat{y}^i)$ could be the BLEU score or any other evaluation metrics. The optimization goal is to maximize the expected reward across the dataset:

$$L_R(\theta) = \sum_{i=1}^{N} \mathbb{E}_{p_\theta(\hat{y}^i|x^i)}[\mathrm{R}(y^i, \hat{y}^i)] = \sum_{i=1}^{N} \sum_{\hat{y}^i \in \Upsilon(x^i)} p_\theta(\hat{y}^i|x^i)\mathrm{R}(y^i, \hat{y}^i)$$

where $\mathbb{E}(\cdot)$ denotes the expectance and $\Upsilon(x^i)$ is the set of all the possible predicted sequences for the input image x^i. The training objective becomes:

$$\hat{\theta}_R = \underset{\theta}{\operatorname{argmax}}\{L_R(\theta)\}$$

This sequence-level objective function aims to optimize the prediction of individual tokens within the context of the entire sequence. It also makes it possible to eliminate the exposure bias problem because the optimization is no longer based on each individual token but on the entire sequence, thus it is no longer necessary to feed in ground truth token at every time step to guarantee token alignment. We can simply close the feedback loop by feeding the predicted token instead of the ground truth token to the next time step during the training time.

Notice that it is computationally infeasible to optimize $L_R(\theta)$ based on exhaustive search due to the exponential growth of the search space of $\Upsilon(x^i)$. Meanwhile the gradient descent is not directly applicable here because the reward function $\mathrm{R}\left(y^i , \hat{y}^i\right)$ is a discrete function of the prediction thus is not differentiable. To address this problem, recent solutions have been proposed in NLP community [71, 75, 76], which proposes to formulate this optimization problem as a reinforcement learning problem. In this setting, the prediction model is treated as an *agent*. Prediction on the next token is an *action*. At completion of the prediction, the predicted sequence is compared against the ground truth sequence to get a sequence-level evaluation score, which is the *reward*. The parameters of the neural network define a *policy*. Even though $L_R(\theta)$ is not differentiable, the policy gradient algorithm [70] can be used to transform the gradient of expectation as an expectation of gradient so that we can avoid taking derivative over the reward function:

$$\nabla_\theta L_R(\theta) = \sum_{i=1}^{N} \mathbb{E}_{p_\theta(\hat{y}^i|x^i)}\left[\mathrm{R}\left(y^i, \hat{y}^i\right)\nabla_\theta \log p_\theta(\hat{y}^i|x^i)\right]$$

In principle, one may leverage the REINFORCE algorithm [77] to estimate the above expectation based on sampling methods. In specific, the expected value can be approximated by taking one sample from the distribution $\tilde{y} \sim p_\theta(y|x^i)$ using multinomial sampling [78]. Unfortunately, it difficult for the neural network to converge this way due to the high variance in gradient estimation. One technique to reduce the variance is to subtract an average reward $\bar{r}$ from the prediction reward [71]. This way, the estimated derivative becomes:

$$\widetilde{\nabla}_\theta L_R(\theta) = \sum_{i=1}^{N} \left[\mathrm{R}\left(y^i, \tilde{y}\right) - \bar{r} \right] \cdot \nabla_\theta \log p_\theta(\tilde{y}|x^i)$$

The average reward $\bar{r}$ was estimated by training a separate neural network layer in [71]. In our work, we simply use Monte Carlo sampling to estimate $\bar{r}$, i.e., taking k samples from the multinomial distribution and calculate the average value. Now that the derivative is obtainable, the backpropagation algorithm can be used for the sequence-level training.

Experiments

In this section, we will first introduce the dataset used to train and evaluate our model, and then discuss the evaluation metrics and other baseline methods, followed by implementation details in the end.

Dataset and Preprocessing

We used the public dataset IM2LATEX-100K [38], which is constructed from the LaTeX sources of publications crawled from *High Energy Physics - Theory* topic on arXiv.org. The dataset contains a total of 103,556 LaTeX sequences representing different math formulas. The length of characters of each sequence ranges from 38 to 997, with mean 118 and median 98. Each math formula is rendered into the PDF format by the pdfLaTeX[1] tool, and then converted to greyscale images in PNG format at resolution 1654

[1] LaTeX (version 3.1415926-2.5-1.40.14)

× 2339. The dataset provides a standard partition of a training set of 83,883 formulas, a validation set of 9,319 formulas, and a test set of 10,354 formulas.

The training of our model starts with constructing a token vocabulary $\vec{V}$. This can be done by tokenizing the LaTeX sources in the dataset. A straightforward approach to tokenize the LaTeX sources is to treat each individual character as a token. A more sophisticated approach is to parse the LaTeX sources into shortest reserved LaTeX words. For example, '*\psi*' stands for " ψ" in LaTeX, which would be treated as one single token, rather than four separate tokens '\', '*p*', '*s*', '*i*'. The second approach has the obvious advantages of reducing the sequence length and avoiding unnecessary prediction errors and computations. However, this approach is not trivial because it needs to have a complete list of LaTeX reserved words and an effective parsing algorithm to segment the LaTeX sources. Here we adopt the LaTeX parser developed in [38]. This parser first converts a LaTeX source into an abstract syntax tree using KaTex [79], and then generates the tokens by traversing through the syntax tree. One can also apply tree transformation during this process to normalize the LaTeX sequences. This normalization step can reduce the LaTeX polymorphic ambiguity since a same math formula image can be produced from different LaTeX source sequences. Details of the normalization rules can be found in [38]. Two utility tokens <*START*> and <*END*> are added to the vocabulary to represent the *start of sequence* and *end of sequence* respectively. The decoder is initialized with the <START> token and keeps making predictions until it encounters the <END> token. We end up with a vocabulary of size of 483.

Images are preprocessed by being cropped to only the formula area, and then downsampled to half of their original sizes for memory efficiency. To facilitate parallelization, images of similar sizes are grouped and padded with whitespaces into buckets of 20 different sizes.[2]

Evaluation Criteria and Baselines

Two types of performance metrics are used to measure the accuracy of the prediction system. The first is the BLEU score between the predicted sequence and the ground truth sequence. Widely used to measure the quality of machine translation on natural languages, the BLEU score measures overlapping of *n*-grams. We report the cumulative 4-gram BLEU score commonly used in the literature. Due to the LaTeX grammar ambiguity, (e.g., x_i^j can be expressed by either *x_i^j* or *x^j_i*), we further report the similarity between the ground truth image and the image rendered from the predicted LaTeX sequence based on four different metrics: image edit distance, exact match, exact match without space, and Image-based Mathematical Expression Global Error (IMEGE) [80]. The image edit distance refers to the column-wise edit distance between the ground truth image and the tested image. To calculate the image edit distance, the image is first binarized, and then converted into a 1D array. Each element in the array is a string representation of that column of data (the string is composed of 0's and 1's). The *edit*

[2] Different sizes of width-height buckets (in pixel): (320, 40), (360, 60), (360, 50), (200, 50), (280, 50), (240, 40), (360, 100), (500, 100), (320, 50), (280, 40), (200, 40), (400, 160), (600, 100), (400, 50), (160, 40), (800, 100), (240, 50), (120, 50), (360, 40), (500, 200).

distance score is equal to $1 - e$, where e is the total number of edit operations divided by the length of the 1D array. We also report the exact match accuracy (i.e., two images are exactly the same), and the exact match after eliminating the whitespace columns. These three metrics were first used in [38]. We further report the IMEGE score proposed in [80], which is based on the idea of image distortion model.

Based on these performance metrics, our method is evaluated against the commercial software InftyReader [8], and three recent works based on deep learning: WYGIWYS [38], Double Attention [43], and DenseNet [40]. For completeness, we also compare our model with the popular commercial software Mathpix [81]. Since it is a closed-source for-profit software, we only run it manually on 100 images and report the evaluation results.

Implementation Details

Given a relatively small vocabulary size of 483, we choose a small token embedding size of 32. The dimension of the CNN feature maps and that of the RNN hidden states are both set to $D = 512$. The mini-batch gradient descent algorithm with Adam optimizer [82] is used to train the parameters, with an initial learning rate of 0.1. Batch size is set to 16 due to GPU memory limitation. To reduce overfitting and improve generalization, the dropout technique [83] with dropout rate of 0.4 is used during training. Randomly dropping out nodes during training can be viewed as a form of simulation to create an ensemble of different neural network configurations.

For sequence-level training, the choice of evaluation metrics is very flexible. For computation efficiency, we use BLEU score as the sequence-level evaluation metric. The initial learning rate is set to 0.00005 for the reinforcement training. The sampling size k for calculating the average reward is set to 20.

To reduce the chance of being trapped at suboptimal solutions, beam search [84] is used while making predictions during the test time. At every time step, beam search selects b tokens with the highest probabilities from the vocabulary. The model stops making new predictions until all b predicted tokens become <EOS>. We use a beam size $b = 5$.

The overall system is implemented in PyTorch [85] to produce a deep learning model consisting of 10,870,595 parameters. It is trained on an 8GB NVIDIA Quadro M5000 GPU with 2048 CUDA cores.

Results and Discussions

General Performance

Table 10 Performance evaluation of different models on the IM2LATEX-100K dataset.

Model	**BLEU**	**Image Edit Distance**	**Exact Match**	**Exact Match (-ws)**	**IMEGE**
INFTY	66.65	53.82	15.60	26.66	-
WYGIWYS	87.73	87.60	77.46	79.88	90.26
Double Attention	88.42	88.57	79.81	-	-
DenseNet	88.25	91.57	-	-	-
MI2LaTeX w/o Reinforce	89.08	91.09	79.39	82.13	95.41
MI2LaTeX with Reinforce	**90.28**	**92.28**	**82.33**	**84.79**	**96.15**

The detailed performance results are reported in Table 10, where the last two rows show the performance of our model without and with the sequence-level reinforcement training. All the four deep learning models achieved a significantly better performance over the InftyReader system. Among different deep learning models, [43] and [40] achieved a better performance over the deep learning baseline [38], which is attributed to the introduction of more sophisticated convolutional networks and attention models. The best performance is achieved by training our model with BLEU score as the reinforcement reward, which shows the highest score on all the five evaluation metrics, with a BLEU score of 90.28%, image edit distance of 92.28%, exact match rate of 82.33%, exact match rate without whitespace of 84.79%, and an IMEGE score of 96.15%. The performance results reaffirm our observation about the importance of preserving positional locality, sequence-level optimization criteria, as well as the elimination of the exposure bias problem.

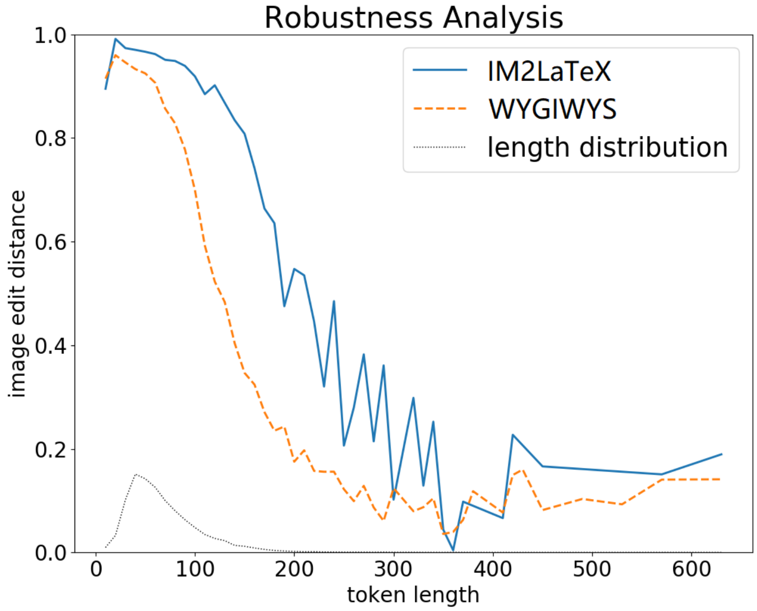


Figure 16 Robustness analysis on token length vs. image edit distance with different models. The black curve shows the density distribution of token lengths in the test set.

Next, we report a robustness analysis of our model vs. WYGIWYS [38] with respect to the sequence length. We use a bin size of 10 to quantize the sequence lengths, and report the average of the image edit distances within a bin as the performance metric. The results of the two models are shown in Figure 16. As expected, the performance of both models declines as the sequence length increases, but at significantly different rates. Knowing that the training set does not contain sequence longer than 150 tokens, this means that the models are also tested on samples with unseen lengths during the test time. At sequence length of 150, the edit distance scores of ours vs. [2] are 0.79 and 0.43, respectively, and at the length of 200, the two scores are 0.54 and 0.17 respectively. Our model shows the capability to handle sequences of unseen length better than the baseline model, especially in the range within 300. Notice that only 3.4% of the test samples have a length longer than 150 tokens, as indicated by the histogram of the token lengths shown in black curve, which makes the performance score after 150 spiky because of the data sparsity. The extra-long LaTeX sequences usually corresponds to large matrices or multi-line math formulas. It remains an open problem to translate them reliably.

In terms of computation cost, the model is first trained for 23 epochs with the MLE as the objective function, which took around 16 hours. The model with the highest token-level accuracy on the validation set is chosen as the candidate model for the sequence-level training. After we switched to the sequence-level objective function, the model is trained for another 15 epochs, which took around 75 hours. The best model was selected as the one with the highest BLEU score on the validation set.

An analysis of Mathpix

We manually ran the Mathpix software on 100 images. These images are selected as the first 100 images in the test set of IM2LATEX-100K. We also re-evaluated our model on this subset of images. Table 11 shows a detailed comparison between the two systems.

Table 11 Mathpix vs. MI2LaTeX on 100 images.

	Mathpix	MI2LaTeX
BLEU	80.64	92.08
Image Edit Dist	76.19	93.38
Exact Match	8.00	82.00
Exact Match (-ws)	34.00	84.00
IMEGE	83.19	97.23

As we can see, MI2LaTeX achieves significantly higher scores than Mathpix on all evaluation metrics. This is inconsistent with our observation that Mathpix is highly accurate. Particularly, the exact match rate of Mathpix is surprisingly low. By investigating further, we found two reasons that lead to this result. The first reason is that the LaTeX coding style of Mathpix is very different from human coding style. This can be observed from the LaTeX sequences generated by Mathpix. Take "*\left*" and "*\right*" operators for example. These two operators appear 273 times in the Mathpix predictions (out of the 100 images) but only 95 times in the ground truth. This instantly brings down the BLEU score of Mathpix. In comparison, this number is 82 in the MI2LaTeX predictions, which is close to the ground truth. This is because MI2LaTeX is trained with human-crafted LaTeX source codes thus it mimics the human coding style. The second reason is that the symbol distances detected by Mathpix is not as precise. This can be seen from the first example in Figure 17 (left column). Even though in this example the inaccurate symbol distances lead to recognition errors, in most cases minor differences on symbol distances does not lead to semantic errors, and these differences may not even be visually observable. This explains why the exact match rate is only 8% (i.e., only 8 out of the 100 images give perfect matches) when we did not visually observe as many errors. To perform a semantic-level evaluation, we visually inspected the images reconstructed by Mathpix, and found that only 12 out of the 100 images contain semantic errors. In comparison, 8 out of the 100 images reconstructed by MI2LaTeX contain semantic errors. This means the two systems have comparable performance on the tested images. There are also cases when Mathpix performs better than MI2LaTeX, such as the second example shown in Figure 17 (right column).

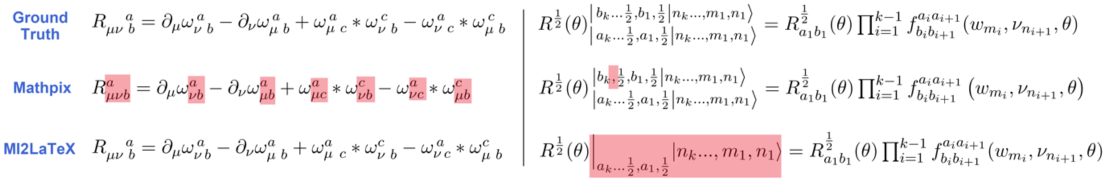


Figure 17 Two examples showing the recognition quality of Mathpix and MI2LaTeX. The reconstruction errors are highlighted as red blocks.

Discussions

The training is end-to-end, which means no explicit information is given about segmentation of symbols in the images, scanning direction of the images, or the grammar for the LaTeX sequence outputs. And the evaluation performance suggests that these information can be learned implicitly by our deep learning model. In

Figure 27 (Appendix A), we give an example that could help us better understand the translation process of our model. The red rectangles in the images show the weights of the soft attention, while deeper color indicates higher weight values. Since the weights are applied on the CNN feature map, each attention weight corresponds to an area of 8×8 pixels in the original image, which is roughly the size of one character. We observe that the trained deep neural network can segment the symbols of different shapes and sizes, some of which are stacked or overlapped, e.g., the superscript “2” inside the square root under the fraction line. The translation process roughly follows a left-to-right order, similar to text recognition. Furthermore, it can also go from top-to-down (e.g., numerator to denominator) or down-to-top (e.g., lower to higher limits in the integral operator). This demonstrates the importance of capturing the spatial locality information. In addition, tokens that are not visible in the input images are also generated. For example, ‘_’, ‘^’ are generated for structural representation. ‘*{*‘, ‘*}*’ are generated for grouping. At every time step, the weights are concentrated on only a few neighborhood regions. The model does not focus on the whitespaces until it reaches the end, in which cases an <EOS> token is generated.

Notice that compared to the DenseNet model [40], our model achieved a higher performance gain on the BLEU score by 2.03%, but a lower performance gain on the image edit distance by 0.71%. A possible explanation is that the sequence-level evaluation metric we used for reinforcement learning is the BLEU score. This would naturally lead to an improvement on the BLEU score performance, but does not necessarily lead to the same amount of improvement on the image edit distance because of the polymorphic ambiguity in LaTeX language. Granted, the image edit distance score of course can be used for sequence-level training, but its drastically increased computing cost makes it an unattractive option, because every LaTeX sequence needs to be compiled to PDF and then converted from PDF to image, which requires a lot of file-level I/O, not to mention the high cost of calculating the image edit distance. One possible future improvement is to distribute this part of computation to a group of machines to facilitate reinforcement training using image edit distance. Notice that unlike [43] and [40], our performance gain over baseline [38] is attributed to adding positional encoding, introducing the sequence-level training objective, and eliminating exposure bias. We believe that our model could be potentially further improved by fusing more recent advancement in deep learning techniques, such as using DenseNet [41] as the encoder, joint attention [42] as the attention mechanism, and GRU [86] or Transformer [9] as the decoder.

Summary

We have proposed MI2LaTeX, a deep neural network model with encoder-decoder architecture to translate images with math formulas into their LaTeX source sequences. The model was trained in an end-to-end manner without explicit labels about image segmentation and grammar information. Nevertheless, the model managed to learn to produce LaTeX output sequence that can reproduce the input image. Using the BLEU score as the reward function and the policy gradient algorithm in reinforcement learning, we successfully trained the model with sequence-level objective function and eliminated the exposure bias problem. MI2LaTeX was evaluated on the IM2LATEX-100K dataset and was compared with other state-of-the-art solutions, and showed the best performance on both sequence-based and image-based measurements. The model also showed more robust performance towards longer LaTeX sequences.

CHAPTER V

PDF2LATEX: A DEEP LEARNING SYSTEM TO CONVERT PDF DOCUMENTS TO LATEX[*]

The mathematical contents of scientific publications in PDF format cannot be easily analyzed by regular PDF parsers and OCR tools. In this paper, we propose a novel OCR system called *PDF2LaTeX*, which extracts math expressions and text in both postscript and image-based PDF files and translates them into LaTeX markup. As a preprocessing step, PDF2LaTeX first renders a PDF file into its image format, and then uses projection profile cutting (PPC) to analyze the page layout. The analysis of math expressions and text is based on a series of deep learning algorithms. First, it uses a convolutional neural network (CNN) as a binary classifier to detect math image blocks based on visual features. Next, it uses a conditional random field (CRF) to detect math-text boundaries by incorporating semantics and context information. In the end, the system uses two different models based on a CNN-LSTM neural network architecture to translate image blocks of math expressions and plaintext into the LaTeX representations. For testing, we created a new dataset composed of 102 PDF pages collected from publications on arXiv.org and compared the performance between PDF2LaTeX and the state-of-the-

[*] Reprinted with permission from “PDF2LaTeX: A Deep Learning System to Convert Mathematical Documents from PDF to LaTeX” by Wang, Z., Liu, J. C. in the *Proceedings of the 20th ACM Symposium on Document Engineering 2020* (DocEng’20) (pp. 1-10), Copyright 2020 by ACM.

art commercial software InftyReader. The experiment results showed that the proposed system achieved a better recognition accuracy measured by the string edit distance between the predicted LaTeX and the ground truth.

Overview

We are in an era of explosive growth in digital publications. According to a research done by the University of Ottawa [1], by the year of 2009, researchers have published 50 million research papers cumulatively. According to the 2019 arXiv statistics report [2], around 150,000 preprints are added to the repository site arXiv.org annually, most of which are in the fields of physics, math, and computer science. This trend is still fast growing. The vast majority of these papers are published in PDF format, despite some criticism about the format [3]. The PDF format in its essence is a mixed representation of different graphical elements, such as predefined fonts or vector graphics [4]. The openness and the evolving nature of the PDF format make it very difficult to recover a stringent markup from it. Textual contents can be embedded into PDF fonts, but they can also be missing or even be wrong [5]. In addition, the format provides no structural information or tags, making it difficult for machines to understand contents beyond text. Particularly, math expressions are heavily used and blended into the main body text in scientific papers, but they are not tagged and are often represented as graphics. This not only harms text-based information retrieval and knowledge mining but also misses the rich technical essence carried inside math expressions. As such, we are motivated to build an automated system that takes mathematical documents in PDF format as inputs and produces their LaTeX markup. Such a system will help both text-based and math information retrieval

(MIR) [19] using the recovered text and LaTeX markup. MIR is important for math-based digital libraries, such as NIST Digital Library of Mathematical Functions [87] and Wolfram Functions Site [88]. The LaTeX markup of math expressions can be further converted to other formats such as MathML [67] for web browsers, and Braille code [89] for blind people. The system can also be very useful for re-digitization of image-based PDF documents.

Existing PDF parsers like PDFMiner [3] and Apache PDFBox [6] are designed to decode PDF text and reconstruct basic page layout structures such as columns and paragraphs. However, they have limited ability on processing math expressions, which are essentially a group of graphical objects, some of which may not even be rendered using fonts (such as a fraction line). In addition, image-based PDF documents would make PDF parser-based solutions in vain. Optical character recognition (OCR)-based approaches can be used to overcome the limitations of PDF parsers. Modern OCR techniques can recognize English text at very high accuracy [7], but recognizing math expressions remains challenging because of their complex layout structures. In recent years, several effective methods have been developed for recognizing math expressions, from explicit layout structure analysis [8, 90] to end-to-end deep learning models [39, 44]. Yet, recognizing mathematical documents requires not only the recognition of plaintext and math expressions individually, but also the exact positions and boundaries of different elements.

In this chapter, we propose a novel OCR system called *PDF2LaTeX*, which leverages recent advancements in machine learning, especially deep learning, to convert mathematical documents from PDF format to LaTeX. PDF documents are first rendered into images as the system input. To locate the text and math contents on the images, we used projection profile cutting (PPC) to segment a page into image blocks of tokens and ordered them line-by-line. Next, we designed a fully convolutional neural network (CNN) with global average pooling to classify each image block into text or math. Then, we trained a conditional random field (CRF) to utilize the semantics and context information for math-text boundary delineation. This way we obtained an ordered sequence of images blocks, each containing either a plaintext word or a math expression. To recognize the contents in each image block, we used two neural networks with CNN-LSTM (Long short-term memory) architecture to translate plaintext words and math expressions into LaTeX respectively. We also created a new dataset with 102 PDF pages using real-world arXiv papers and developed an evaluation tool. Comparison with the state-of-the-art commercial software InftyReader [8] showed that PDF2LaTeX achieved better conversion accuracy. The contributions of this work are summarized below:

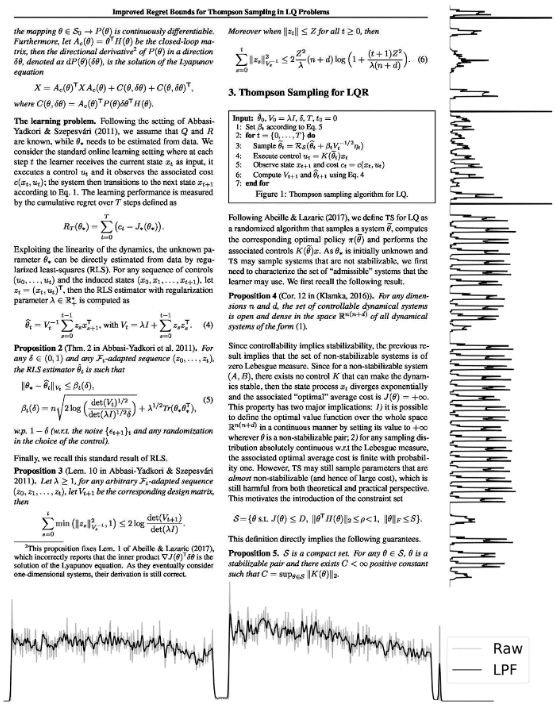
- Proposed a deep learning-based OCR system that models visual, semantics, and context information for mathematical document analysis.
- Generated LaTeX markup from PDF documents with state-of-the-art quality.
- Released a new dataset and an evaluation tool for the PDF-to-LaTeX conversion task.

Segmentation and Detection

In this section, we first discuss the projection profile cutting technique used to segment a PDF page. Then, we give details on the design of a fully convolutional neural network for math/text classification, followed by a conditional random field to enhance the math-text boundary delineation.

Projection Profile Cutting

Projection profile cutting (PPC) is a technique used to detect the structure of a page by repeatedly cutting an image on the horizontal and vertical direction based on projection profiles. A projection profile is a histogram calculated by summing together all the pixel values along either the horizontal or vertical axis. For example, if we project all the black pixels horizontally, we get a horizontal projection profile that can be used for line detection.



Improved Regret Bounds for Thompson Sampling in LQ Problems

the mapping $\theta \in \mathcal{S}_0 \rightarrow P(\theta)$ is continuously differentiable. Furthermore, let $A_c(\theta) = \theta^{\mathsf{T}} H(\theta)$ be the closed-loop matrix, then the directional derivative[2] *of $P(\theta)$ in a direction $\delta\theta$, denoted as $\mathrm{d}P(\theta)(\delta\theta)$, is the solution of the Lyapunov equation*

$$X = A_c(\theta)^{\mathsf{T}} X A_c(\theta) + C(\theta, \delta\theta) + C(\theta, \delta\theta)^{\mathsf{T}},$$

where $C(\theta, \delta\theta) = A_c(\theta)^{\mathsf{T}} P(\theta) \delta\theta^{\mathsf{T}} H(\theta)$.

The learning problem. Following the setting of Abbasi-Yadkori & Szepesvári (2011), we assume that Q and R are known, while θ_* needs to be estimated from data. We consider the standard online learning setting where at each step t the learner receives the current state x_t as input, it executes a control u_t and it observes the associated cost $c(x_t, u_t)$; the system then transitions to the next state x_{t+1} according to Eq. 1. The learning performance is measured by the cumulative regret over T steps defined as

$$R_T(\theta_*) = \sum_{t=0}^{T} \big(c_t - J_*(\theta_*)\big).$$

Exploiting the linearity of the dynamics, the unknown parameter θ_* can be directly estimated from data by regularized least-squares (RLS). For any sequence of controls $(u_0, \ldots, u_t)$ and the induced states $(x_0, x_1, \ldots, x_{t+1})$, let $z_t = (x_t, u_t)^{\mathsf{T}}$, then the RLS estimator with regularization parameter $\lambda \in \mathbb{R}_+^*$ is computed as

$$\widehat{\theta}_t = V_t^{-1} \sum_{s=0}^{t-1} z_s x_{s+1}^{\mathsf{T}}, \text{ with } V_t = \lambda I + \sum_{s=0}^{t-1} z_s z_s^{\mathsf{T}}. \quad (4)$$

Proposition 2 (Thm. 2 in Abbasi-Yadkori et al. 2011). *For any $\delta \in (0,1)$ and any $\mathcal{F}_t$-adapted sequence $(z_0, \ldots, z_t)$, the RLS estimator $\hat{\theta}_t$ is such that*

$$\|\theta_* - \widehat{\theta}_t\|_{V_t} \leq \beta_t(\delta),$$
$$\beta_t(\delta) = n\sqrt{2\log\Big(\frac{\det(V_t)^{1/2}}{\det(\lambda I)^{1/2}\delta}\Big)} + \lambda^{1/2} Tr(\theta_* \theta_*^{\mathsf{T}}), \quad (5)$$

w.p. $1 - \delta$ (w.r.t. the noise $\{\epsilon_{t+1}\}_t$ and any randomization in the choice of the control).

Finally, we recall this standard result of RLS.

Proposition 3 (Lem. 10 in Abbasi-Yadkori & Szepesvári 2011). *Let $\lambda \geq 1$, for any arbitrary $\mathcal{F}_t$-adapted sequence $(z_0, z_1, \ldots, z_t)$, let V_{t+1} be the corresponding design matrix, then*

$$\sum_{s=0}^{t} \min\big(\|z_s\|_{V_s^{-1}}^2, 1\big) \leq 2\log\frac{\det(V_{t+1})}{\det(\lambda I)}.$$

Moreover when $\|z_t\| \leq Z$ for all $t \geq 0$, then

$$\sum_{s=0}^{t} \|z_s\|_{V_s^{-1}}^2 \leq 2\frac{Z^2}{\lambda}(n+d)\log\Big(1 + \frac{(t+1)Z^2}{\lambda(n+d)}\Big). \quad (6)$$

3. Thompson Sampling for LQR

```
Input: $\hat{\theta}_0, V_0 = \lambda I, \delta, T, t_0 = 0$
1: Set $\beta_t$ according to Eq. 5
2: for $t = \{0, \ldots, T\}$ do
3:     Sample $\widetilde{\theta}_t = \mathcal{R}_{\mathcal{S}}(\widehat{\theta}_t + \beta_t V_t^{-1/2} \eta_t)$
4:     Execute control $u_t = K(\widetilde{\theta}_t) x_t$
5:     Observe state $x_{t+1}$ and cost $c_t = c(x_t, u_t)$
6:     Compute $V_{t+1}$ and $\widehat{\theta}_{t+1}$ using Eq. 4
7: end for
```

Figure 1: Thompson sampling algorithm for LQ.

Following Abeille & Lazaric (2017), we define TS for LQ as a randomized algorithm that samples a system $\widetilde{\theta}$, computes the corresponding optimal policy $\pi(\widetilde{\theta})$ and performs the associated controls $K(\widetilde{\theta})x$. As θ_* is initially unknown and TS may sample systems that are not stabilizable, we first need to characterize the set of "admissible" systems that the learner may use. We first recall the following result.

Proposition 4 (Cor. 12 in (Klamka, 2016)). *For any dimensions n and d, the set of controllable dynamical systems is open and dense in the space $\mathbb{R}^{n(n+d)}$ of all dynamical systems of the form* (1).

Since controllability implies stabilizability, the previous result implies that the set of non-stabilizable systems is of zero Lebesgue measure. Since for a non-stabilizable system (A, B), there exists no control K that can make the dynamics stable, then the state process x_t diverges exponentially and the associated "optimal" average cost is $J(\theta) = +\infty$. This property has two major implications: *1)* it is possible to define the optimal value function over the whole space $\mathbb{R}^{n(n+d)}$ in a continuous manner by setting its value to $+\infty$ wherever θ is a non-stabilizable pair; *2)* for any sampling distribution absolutely continuous w.r.t the Lebesgue measure, the associated optimal average cost is finite with probability one. However, TS may still sample parameters that are *almost* non-stabilizable (and hence of large cost), which is still harmful from both theoretical and practical perspective. This motivates the introduction of the constraint set

$$\mathcal{S} = \{\theta \text{ s.t. } J(\theta) \leq D, \ \|\theta^{\mathsf{T}} H(\theta)\|_2 \leq \rho < 1, \ \|\theta\|_F \leq S\}.$$

This definition directly implies the following guarantees.

Proposition 5. *$\mathcal{S}$ is a compact set. For any $\theta \in \mathcal{S}$, θ is a stabilizable pair and there exists $C < \infty$ positive constant such that $C = \sup_{\theta \in \mathcal{S}} \|K(\theta)\|_2$.*

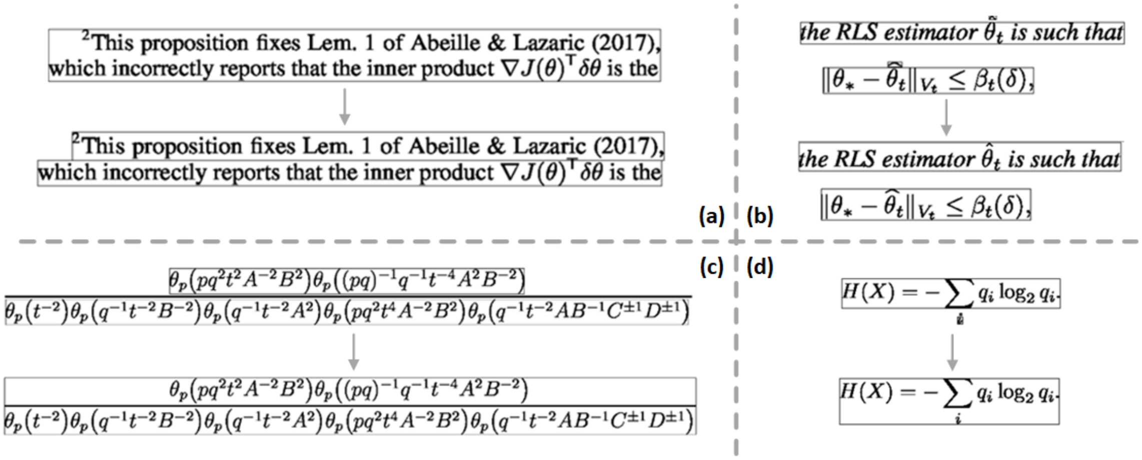
[2] This proposition fixes Lem. 1 of Abeille & Lazaric (2017), which incorrectly reports that the inner product $\nabla J(\theta)^{\mathsf{T}} \delta\theta$ is the solution of the Lyapunov equation. As they eventually consider one-dimensional systems, their derivation is still correct.

Figure 18 Visualization of PPC. The bottom plot shows the vertical projection profile of the entire page. The right plot shows the horizontal projection profile of the right column.

In our system, PPC is used to segment a page into columns, lines, and tokens. Specifically, we first render a PDF page into a grayscale image at 250 dpi, then binarize it and calculate the vertical projection and horizontal projection alternatively. Figure 18 shows a visualization of PPC on a page. The design of our PPC algorithm is based on the assumption that tables and figures are already detected and removed. We refer the readers to [50, 51] for table detection and [48, 52] for figure detection algorithms.

Column detection. The first step is to determine if a page is single-column or double-column. To do this, we first calculate the vertical projection profile based on the entire image. The rationale is that a double-column page will have an empty region in the center of the vertical projection profile. Notice that titles and headers can act as noise in the center region. To remove noise, we designed a finite impulse response low-pass filter (LPF) to smooth the profile. The filter has an order of 9 and a cut-off frequency of 0.125 Hz. The original vertical projection profile and the profile filtered by the LPF are shown at the bottom of Figure 18. We then set values that are smaller than 10% of the maximum value of the profile to zero. If consecutive zeros are observed in the center of the profile, we split the page into two columns around the zeros and process the line detection and token detection algorithms on each column separately. Otherwise, we process the entire page as one column. This algorithm can also be easily generalized to process more than two columns.

Line detection. We calculate the horizontal projection profile per column and split the profile into multiple segments of non-zero values. Each such segment becomes the horizontal boundary of a candidate line. An example is shown in the plot on the right side of Figure 18. This process produces preliminary results for line segmentation, which are further processed by several heuristic rules, as detailed below:



[2]This proposition fixes Lem. 1 of Abeille & Lazaric (2017), which incorrectly reports that the inner product $\nabla J(\theta)^{\mathsf{T}}\delta\theta$ is the

[2]This proposition fixes Lem. 1 of Abeille & Lazaric (2017), which incorrectly reports that the inner product $\nabla J(\theta)^{\mathsf{T}}\delta\theta$ is the

the RLS estimator $\widehat{\theta}_t$ *is such that* $\|\theta_* - \widehat{\theta}_t\|_{V_t} \le \beta_t(\delta)$,

the RLS estimator $\widehat{\theta}_t$ *is such that* $\|\theta_* - \widehat{\theta}_t\|_{V_t} \le \beta_t(\delta)$,

$$\frac{\theta_p(pq^2t^2A^{-2}B^2)\theta_p((pq)^{-1}q^{-1}t^{-4}A^2B^{-2})}{\theta_p(t^{-2})\theta_p(q^{-1}t^{-2}B^{-2})\theta_p(q^{-1}t^{-2}A^2)\theta_p(pq^2t^4A^{-2}B^2)\theta_p(q^{-1}t^{-2}AB^{-1}C^{\pm1}D^{\pm1})}$$

$$\frac{\theta_p(pq^2t^2A^{-2}B^2)\theta_p((pq)^{-1}q^{-1}t^{-4}A^2B^{-2})}{\theta_p(t^{-2})\theta_p(q^{-1}t^{-2}B^{-2})\theta_p(q^{-1}t^{-2}A^2)\theta_p(pq^2t^4A^{-2}B^2)\theta_p(q^{-1}t^{-2}AB^{-1}C^{\pm1}D^{\pm1})}$$

$$H(X) = -\sum_i q_i \log_2 q_i.$$

$$H(X) = -\sum_i q_i \log_2 q_i.$$

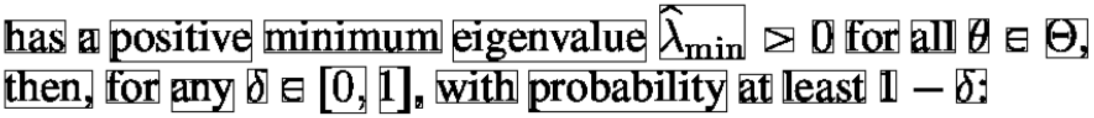
Figure 19 Examples of line detection heuristics. (a) Split overlapped lines. (b) Merge hats. (c) Merge fraction lines. (d) Merge binding variables.

- *Overlapped lines.* Two separate lines can be falsely merged together if their pixels overlap horizontally. For example, in Figure 19 (a), the superscript "T" on the second line overlaps with the pixels on the first line, making their projection profiles connected together. To handle such errors, we split a profile segment into two from the center if the height of this line is larger than the most common line height on the page, but the values around the center are smaller than 10% of the median of the profile values.
- *Variable hats.* The hat of a math variable can be falsely detected as an independent line. An example is shown in Figure 19 (b). To merge hats, if a line is mostly blank but has small groups of connected pixels, this line becomes a candidate for variable hats. If the distance between this candidate and its closest neighbor line is smaller

than the most common line distance of this page, this line is merged to its closest neighbor line as a hat.

- *Fraction expressions.* A fraction expression can be falsely detected as three separate lines: the numerator, the denominator, and the fraction line, as shown in Figure 19 (c). To merge these, we first detect black lines as long narrow lines filled with black pixels. Next, we merge this black line with its two neighbor lines (numerator and denominator) if the widths of the neighbor lines are shorter than the black line and the distance to the black line is smaller than the most common line distance of this page.
- *Binding variables.* Binding variables are math variables bonded with big operators such as a big sum or a big product operator. They can be falsely detected as independent lines, as shown in Figure 19 (d). To merge the binding variables with the big operators, we first need to find lines that potentially contain big operators, which are defined as lines whose heights are larger than twice of the most common line height. We also need to find the binding variables lines, which are defined as the neighbors of the big operator lines whose height is smaller than the most common line height and the distance to the big operator line is smaller than the most common line distance. Once binding variables are detected, we merge them into their corresponding big operator lines.

Token detection. The process is similar to line detection. We first calculate the vertical projection profile per line and split the profile into multiple segments of non-zero values. Each such segment becomes the vertical boundary of a candidate character. We define character distance as the distance between two neighboring non-zero segments. The most common character distance corresponds to the distance of characters within tokens, which is smaller than the distance between tokens. As such, we use twice the most common character distance as the threshold to group characters into tokens. An example result of token segmentation is shown in Figure 20. As we can see, the algorithm works well for plaintext words but poor for math expressions. In fact, math expressions are almost always detected as broken math segments, because the character distances within math expressions do not have fixed patterns like that of text. This problem can be easily fixed if the labels of each token are given (math vs. text), because complete math expressions can be obtained by simply merging the neighboring math segments. In the following subsection, we will introduce two machine learning models to classify each token into math or plaintext.



has a positive minimum eigenvalue $\hat{\lambda}_{\min} > 0$ for all $\theta \in \Theta$, then, for any $\delta \in [0, 1]$, with probability at least $1 - \delta$:

Figure 20 Token segmentation results. The bounding boxes mark the boundary of each token.

CNN Classifier

Plaintext words and math segments have different visual features. For example, the layout structure of math segments is less restricted than plaintext because it can include sub/sup-scripts, different font sizes, etc. In addition, math segments can include Greek letters and math operators that are not present in plaintext words. These features can be easily captured by a convolutional neural network (CNN) [28]. Below we will first describe the process of synthesizing the training data for the CNN, then elaborate the design of a fully convolutional neural architecture for math/text classification, and finally evaluate the performance of the classifier on the synthetic data.

The synthetic data for training and testing are generated from the 2003 KDD Cup dataset [91], which includes approximately 29,000 LaTeX source files of hep-th papers from arXiv. We used LaTeXML [66] to parse the LaTeX source files into XML format, in which the LaTeX code of math expressions and plaintext words are extracted and explicitly tagged. Next, we used TeX Live [92] to compile each math expressions and plaintext words into PDF files and then render them into images. This gives us a lot of images of math expressions and plaintext words, together with their ground truth labels. Notice that our classification is performed on math segments vs. plaintext words, thus we went further to apply PPC on the math expression images and split them into smaller image blocks of math segments. These images are generated in different resolutions and different fonts (bold, italic, etc.) which ensures that they cover special cases like section titles. We padded both the plaintext images and math segments images with four-pixel-wide whitespaces. It is very important that all the images get the exact same padding, otherwise the CNN would capture the padding space as the main feature and give false results. In total, the synthetic dataset contains ~75k plaintext words and ~75k math segments, which are randomly split into a training set of 119,500 images, a validation set of 13,259 images, and a test set of 16,073 images.

Given the large amount of data we have, we decided to train a deep convolutional neural network for math detection from scratch. Since the input images have various sizes, we decide to use a fully convolutional network proposed in [93], which handles flexible input sizes by using global average pooling to avoid scaling the images or adding additional fully connected layers. The architecture of the CNN is presented in Table 12. It is similar to the VGG-VeryDeep architecture that has been adapted particularly for OCR applications [34]. We used a global average pooling layer as the last layer, which takes the average of the two output feature maps and produces two scalar values. The two scalar values are further passed through a softmax function which gives the likelihood of being a plaintext word or a math segment.

Table 12 CNN Configuration. *#maps*: the number of feature maps. *k*: kernel size. *p*: padding size. *s*: stride size. *BN*: batch normalization. *GlobalAvgPool*: global average pooling. The sizes are in order (height, width).

Type	#maps	k	p	s
GlobalAvgPool				
Convolution	2	(3,3)	(1,1)	(1,1)
MaxPooling		(2,1)	(0,0)	(2,1)
BN				
Convolution	512	(3,3)	(1,1)	(1,1)
MaxPooling		(1,2)	(0,0)	(1,2)
Convolution	256	(3,3)	(1,1)	(1,1)
BN				
Convolution	256	(3,3)	(1,1)	(1,1)
MaxPooling		(2,2)	(0,0)	(2,2)
Convolution	128	(3,3)	(1,1)	(1,1)
MaxPooling		(2,2)	(0,0)	(2,2)
Convolution	64	(3,3)	(1,1)	(1,1)
Input		Gray-scale image		

The CNN model is implemented in PyTorch [85]. It contains 2,150,685 parameters and is trained for 5 epochs which took 0.2 hours on an Nvidia Quadro M5000 GPU (8GB, 2048 CUDA cores). We used Adam optimizer [82] with a mini-batch size of 32. We also applied the dropout technique [83] during training with a dropout rate of 0.4. For testing, an image block is classified as positive (math) if the likelihood is above 0.5, and negative (plaintext) if otherwise. Out of 16,073 test samples, 15,482 are classified correctly, which gives an accuracy of 96.3%. By looking closer at the confusion matrix in Table 13, we observed a lot more false positives (540) than false negatives (51). The confusion matrix converts to precision 93.6%, recall 99.4%, and F1-score 96.4%. In Section 3.3 we will dive into the cause and the solution to the large false positive number.

Table 13 Confusion matrix of the CNN output.

	Predicted Plaintext	Predicted Math
Plaintext	7,922	540
Math	51	7,560

Conditional Random Field

The CNN classifier gives us a decent math/plaintext classification accuracy on the synthetic dataset. However, there are still many false positives as we have observed in Table 13. This problem becomes more obvious when the model is tested on real publications, as shown in Figure 21 (a). As we can see, the false positives mainly come from the stop words such as “it”, “is”, “of”, “and”, etc., which are falsely classified as math. The reason behind is that these stop words are occasionally included as part of math expressions in the training set. On the other hand, we did not resample these words as plaintext samples when constructing the dataset (the resampling frequency would also be tricky to determine), thus the classification of these words is biased towards positive. At the same time, false negative errors are also noticeable. For example, the math segment “sup $\{G(\Delta(t,$” circled in red is falsely classified as plaintext.

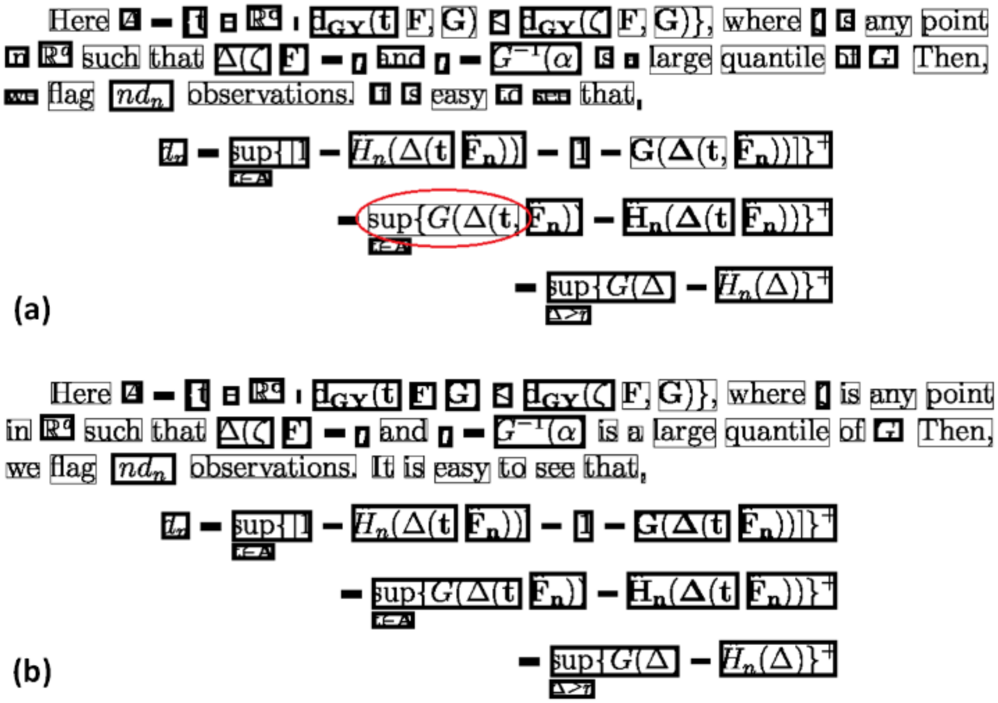


Figure 21 Label assignment before CRF (a) and after CRF (b). Bold bounding boxes mark math labels.

These observations indicate that more important features need to be included in addition to the visual features given by the CNN classifier. One example is the text information, which can be extracted by English OCR engines from the images. The text information carries semantical meanings that can be used to identify the stop words mentioned above. In addition, if the recognized text is a meaningless string instead of a valid English word, it is more likely that the image contains a math segment instead of a plaintext word. In addition to the text information, the math likelihood of the neighbors and the physical distance to the neighboring images are also useful features. They contain context information that can be used to correct errors such as a misclassified segment in the middle of a long math expression. Taking these into consideration, we decide to use a conditional random field (CRF) to model these features. CRFs are a type of supervised machine learning model for sequence labeling purposes, while in our application a sequence is a line of tokens. Assuming that the length of a line is T, we want to find a set of labels l that best describes each token x, i.e., maximizes the conditional probability $p(l|x)$. The label $l \in \{0,1\}$, where 0 indicates plaintext and 1 indicate math. Suppose the CRF has K different feature functions f_k, each with a corresponding weight λ_k. $p(l|x)$ is modeled as:

$$p(l|x) = \frac{1}{Z(x)} \prod_{t=1}^{T} exp\left\{\sum_{k=1}^{K} \lambda_k f_k(l_t, l_{t-1}, x_t)\right\}$$

where $Z(x)$ is a normalization factor that sums over all possible labels l', which is defined as below:

$$Z(x) = \sum\nolimits_{l'} \prod_{t=1}^{T} exp\left\{\sum_{k=1}^{K} \lambda_k f_k(l_t, l_{t-1}, x_t)\right\}$$

The feature functions are composed of pointwise features and pairwise features. Pointwise features describe individual tokens, including the math likelihood of the tokens given by the CNN classifier, the text content, and the image block size. Pairwise features describe the relationship between a token and its immediate neighbors. All the features are summarized in Table 14. Text features are one-hot encoded.

Table 14 Full list of CRF features.

Pointwise features	
Math likelihood	The math likelihood given by the CNN classifier
Text	The text string recognized by the OCR engine
is_plaintext	Whether the text string is a valid English word
Width	The width (in pixel) of the image block
Height	The height (in pixel) of the image block
Pairwise features	
Math likelihood	The math likelihood of the neighbor token given by the CNN classifier
Text	The text string of the neighbor token recognized by the OCR engine
Distance	The distance (in pixel) to the neighboring image block

The CRF model is implemented with sklearn-crfsuite [94]. The parameters λ_k can be learned by the gradient descent algorithm with maximum likelihood estimation as the cost function. Since the number of parameters is relatively small, we created a small dataset to train the CRF. We picked 10 PDF pages from the 2003 KDD Cup dataset and ran our PPC algorithm to segment the pages into tokens. For each token, we used the binary CNN classifier to generate the math likelihood, then used an OCR engine (which will be illustrated in Section 4.2) to generate the text, and manually labeled each token as either math or plaintext as the ground truth. This results in 312 lines, 2,726 tokens of data in total. For evaluation, we applied 10-fold cross-validation by training a model with nine pages and testing with the remaining one page. We report precision, recall, and F1 score in Table 15. The scores are calculated as the average value of the cross-validation results.

Table 15 CRF performance with 10-fold cross-validation.

Performance	Precision	Recall	F1
Before CRF	60.2%	98.7%	74.8%
Pointwise Features Alone	89.8%	95.7%	92.6%
Pointwise + Pairwise	**95.8%**	97.4%	**96.6%**

Table 15 shows the performance without CRF, with CRF using pointwise features alone, and with CRF using the pointwise and pairwise features together. As we can see, before CRF is applied, the precision is fairly low because of the large amount of misclassified stop words. By including the pointwise features, the precision is greatly improved because the semantics information is considered. By including the pairwise features, we achieved the best precision and F1-score, which is attributed to the neighboring context information.

After we apply the CRF on each line, each image block is assigned a refined label. By merging the neighboring math image blocks into larger images, we obtain complete math expressions, which define the math/text boundaries.

Neural Translators

We used PPC and math classification algorithms to decompose a page into a group of image blocks with labels. In this section, we focus on the final step of the system, which is to recognize each image block of plaintext words and math expressions into LaTeX. Math expressions are intrinsically more difficult to recognize than plaintext words. This is because 1) unusual symbols in different sizes are present in math expressions, 2) the structural layout of the symbols needs to be understood, and 3) the generated LaTeX needs to follow correct grammar. In comparison, recognizing plaintext only requires to recognize the individual characters one by one. In section 4.1, we will introduce a neural network with CNN-LSTM architecture which has recently shown success in recognizing math expressions and generating markup LaTeX. In section 4.2, we will use the same neural architecture to train a customized plaintext OCR model. Since the LaTeX generation process is very similar to the translation task in natural language processing, we refer to these neural networks as translators.

Math Translator

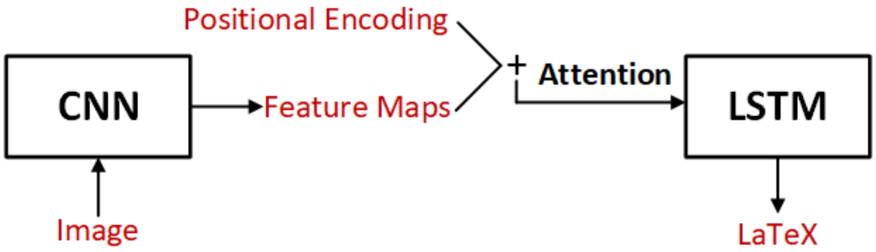


Figure 22 CNN-LSTM Neural Architecture.

To translate images of math expressions into LaTeX, we adopt the neural network model proposed in [44]. The model has an encoder-decoder architecture, as shown in Figure 22. The encoder is a CNN that processes the input images, and the decoder is an LSTM that generates sequential outputs, in our case, a sequence of LaTeX tokens. The CNN first extracts visual features from the original image and produces a set of feature maps. To preserve the spatial location information, the feature maps are combined with positional encoding [9], which is essentially a set of sinusoidal signals that represent positions. The decoder is a two-layer bidirectional LSTM [73] which is used to translate the encoder output into a sequence of LaTeX tokens. At every step, the encoder output is fed into the decoder via the soft attention mechanism [35], which means the input to the LSTM is a weighted combination of the encoder output, where the weights are learned by a separate network. The attention mechanism enables the neural network to selectively focus on different regions of the input images at different steps. The network is first trained using the maximum likelihood estimation as the training objective, which is token-level optimization. Once the token-level training converges, we advance further to a sequence-level training objective using the policy gradient algorithm [70] in reinforcement learning. The reward function for the policy gradient algorithm is chosen as the BLEU score [69]. We refer the readers to [44] for more details.

We trained the model using the public dataset IM2LATEX-100K [38] which contains ~100k images of math expressions and their LaTeX ground truths. The dataset is split into ~90k of training data and ~10k of test data. Each LaTeX string is tokenized into a sequence of reserved LaTeX tokens, such as "\frac", "\sigma", "a", "{", etc. Each image is preprocessed by cropping out the surrounding white spaces and downsampling two times for memory efficiency. The model is implemented in PyTorch and contains 10,870,595 parameters in total. It is trained with a mini-batch size of 16 and a dropout rate of 0.4. The training took around 90 hours on our 8GB Nvidia GPU. We applied beam search [84] during the prediction time, with a beam size of 5. For evaluation, we used the cumulative 4-gram BLEU score as the criteria. The experiment results showed a BLEU score of 90.28% on the test set.

While applying this math translator to PDF pages, we further applied divide-and-conquer approaches to some large math expressions for reducing complexity. This is helpful because even though LSTM has long-term memory that handles longer sequences better than vanilla recurrent neural networks, as math expressions become longer and more complicated, the neural network still produces more errors, especially on the math expressions that expand to multiple lines. To remediate this problem, math expressions expanding to multiple lines are translated line-by-line and then concatenated together. For fraction expressions, numerator and denominator are translated separately and then merged together.

Plaintext Translator

Existing commercial and open-source OCR tools can be used to recognize English text in plaintext images. However, it is difficult to efficiently parallelize these OCR tools on batches of single-word image blocks. Take Tesseract OCR [45] for example, it runs fast on larger images with a collection of sentences, but runs much slower on batches of small images because the software overhead becomes a major performance bottleneck. Furthermore, these tools cannot be used with GPUs. For better efficiency, we decided to train a customized plaintext OCR engine.

Notice that the previous CNN-LSTM architecture can be directly applied to plaintext recognition because the two tasks are very similar. The differences are that plaintext does not have complex layout structures like math expressions, and the character set is smaller (mostly English alphabets). As a result, we decide to inherit the same neural network specifications from the previous section for this task. Given that the feature distribution of plaintext is quite different from math expressions, we decided to train a new model from scratch with a new dataset. We generated a synthetic dataset by collecting 75,581 plaintext words from the 2003 KDD Cup dataset and rendered them into images in different sizes and fonts. We used 60,000 images for training and 15,581 images for testing.

The training procedure is similar to that of the math translator. Since this task is relatively simpler, we only trained it with the token-level objective function. The ground truths are tokenized into individual characters instead of reserved LaTeX tokens. The model was trained with a mini-batch size of 20 and a dropout rate of 0.4. The training took 2.5 hours on our 8GB Nvidia GPU. Experiment results show that 15,527 out of 15,581 words were recognized perfectly, which gave an accuracy of 99.65%.

Experiments and Results

In this section, we first describe a new dataset we created for evaluating the proposed PDF2LaTeX system. We then describe normalization rules and evaluation criteria, and the baseline results generated by the state-of-the-art system InftyReader. In the end, we compare the performance of the two systems.

Dataset

The recognition accuracy of the InftyReader system was previously reported as 98.51% in [8] on a dataset of 476 pages. We inspected the 476-page dataset available on the Infty website[3], and found that these pages are synthesized with displayed math expressions and repeating English words in different fonts and sizes. Our target task, on the other hand, is to recognize real-world scientific publications that follow a different data distribution. As a result, we decided to create a more sophisticated dataset using real-

[3] http://www.inftyproject.org/en/database.html

world papers from arXiv. The dataset consists of 102 PDF pages collected from arXiv papers covering different domains including physics, AI, economics, signal processing, statistics, machine learning, and genomics. We manually extracted the LaTeX source of each page and removed tables, figures, footnotes, references, citations, and comments. Then we compiled these sources into PDF files with TeX Live [92] and rendered them into images at 250 dpi resolution. In the LaTeX ground truth, without counting space, there are 116,970 characters used to construct math expressions and 126,187 characters used to construct plaintext. There are in total 2,233 math expressions in the dataset, with an average of 22 math expressions per page. The dataset and the evaluation tool are publicly available on Github[4].

Baseline and Evaluation Criteria

We generated a baseline for our dataset by processing each page with the state-of-the-art commercial software InftyReader (version 3.2.0 released in November 2019). InftyReader recognizes PDF pages by OCR and can generate the recognition results in LaTeX format. We will compare the LaTeX output of our system with this baseline. Before the evaluation, we need to normalize all the LaTeX sources as follows:

1. Remove the preambles such as macros, i.e., the content before *\begin{document}* and after *\end{document}*.
2. Replace all math claimers with *$*, e.g., *\begin{equation}*, *\begin{eqnarray}*, *\end{gather}*, etc.

[4] https://github.com/wzlxjtu/PDF2LaTeX-dataset

3. Remove section and paragraph claimers, including *\section{}*, *\subsection{}*, *\paragraph{}*, etc.
4. Remove spaces
5. Ignore cases
6. Normalize math polymorphism

Math polymorphism refers to the problem that a same math expression can be expressed by different LaTeX strings. For example, X_a^b can be expressed by either *X_a^b* or *X^b_a*. We used the LaTeX parser developed in [38] to handle the math polymorphism problem. The parser uses KaTeX [79] to convert a LaTeX string into an abstract syntax tree, and then traverses the tree in fixed orders to generate the normalized LaTeX string. Notice that the parser can only solve the polymorphism problem to a limited extent.

For evaluation, we used the string edit distance between the predicted LaTeX and the ground truth LaTeX as the performance metric. In addition to the string edit distance, we also report the edit distance relative to the document size, i.e., edit distance rate, defined below:

$$edit\ dist\ rate = 1 - \frac{edit\ dist}{\#characters\ in\ the\ document}$$

The edit distance rate is more intuitive to understand since higher values (closer to 1) indicate better performance. We also separate LaTeX sources into math parts and plaintext parts, and report the edit distance rate on math and plaintext separately. The evaluation script is released together with the dataset.

Results

In Table 16, we show the evaluation results of InftyReader and PDF2LaTeX system on our 102-page dataset. The results include the string edit distance and the edit distance rate on the overall documents and on the math part and plaintext part alone. In overall, the string edit distance of InftyReader is 76,012, while the string edit distance of PDF2LaTeX is 48,820. This is relative to the total number of 258,383 characters in the ground truth, which is after normalization. By converting these numbers to edit distance rate, InftyReader achieves an overall edit distance rate of 70.6%, while PDF2LaTeX achieves 81.1%. By separating the LaTeX source into math and plaintext, InftyReader achieves an edit distance rate of 86.6% on plaintext, and 46.0% on math. In comparison, PDF2LaTeX achieves an edit distance rate of 94.8% on plaintext, and 65.9% on math.

Table 16 Results reported in Edit Distance (Rate).

Performance	InftyReader	PDF2LaTeX
Subset		
Edit Dist Rate (text)	86.6%	**94.8%**
Edit Dist Rate (math)	46.0%	**65.9%**
Overall		
Edit Dist Rate	70.6%	**81.1%**
Edit Dist	76,012	**48,820**

Next, we report the processing speed of the two systems. This experiment was done on a PC with Intel Xeon 3.5Ghz CPU, 16GB RAM. InftyReader took an average of 25 seconds to process a page. In comparison, PDF2LaTeX took an average of 40 seconds to process a page on the same machine. Notice that our model has the advantage of being able to execute on a GPU. By simply deploying the neural networks on our 8GB Nvidia

Quadro M5000 GPU (without further parallelizing across different pages), the time to process a page is reduced to an average of 14 seconds.

Discussions

In Figure 28 in Appendix B we visualize two examples from the experiment results. Each column of the table corresponds to part of a document in the dataset. The first row shows the original documents. The second row shows the pages reconstructed from the InftyReader output. The third row shows the pages reconstructed from the PDF2LaTeX output. The reconstruction errors are highlighted in red. We can see clearly that PDF2LaTeX produces significantly fewer errors than InftyReader. In fact, most of the errors for both systems occur inside math expressions, which means the performance gain of PDF2LaTeX is largely attributed to the better math translator. Yet, another important advantage of PDF2LaTeX is the better segmentation results. This can be reflected in two parts: 1) for displayed math expressions, InftyReader can mistakenly split sup/sub-scripts and fraction expressions into different lines, as shown in both examples in Figure 28, while our PPC algorithm demonstrated better robustness; 2) for inline math expressions, PDF2LaTeX can detect the math/plaintext boundary more clearly. This is largely attributed to the conditional random field, which utilizes not only the visual features, but also the semantics and context information. The poorer math/plaintext boundary explains why InftyReader only gives an edit distance rate of 86.6% on plaintext in this dataset, as compared to the 99.44% reported in [8].

We have seen from the previous section that the edit distance between our output and the ground truth is 48,820, which is an average of 479 edit distance per page. This is far from perfect, but it is also important to understand that this is not equivalent to 479 errors per page. For example, a minor error like failing to recognize the italic font of a character would result in 9 edit distance error because the italic command *\textit{}* has 9 characters. Furthermore, we need to keep in mind that the math polymorphism problem is not entirely solved even after the normalization technique is applied. To fully solve the polymorphism problem, we will need to convert the recognized math expressions into their semantical representations such as Content MathML [67] before making comparisons. This conversion remains an open problem to date.

The current version of PDF2LaTeX still has limitations and requires future works. For example, the current system only supports limited font styles, because all the training and test data were generated with TeX Live. It is necessary to label more training data covering all sorts of fonts in order to increase the robustness of the system. In addition, noise removal and deskewing techniques are needed in order to process scanned documents and historical publications. Finally, figure and table detection algorithms need to be integrated with the current segmentation algorithm.

Summary

In conclusion, we have proposed a novel OCR system that converts mathematical documents from PDF format into their markup LaTeX. The system used projection profile cutting to segment a page into an ordered sequence of tokens, and then used a convolutional neural network and a conditional random field to classify the token labels and refine the segmentation results. In the end, the system used two CNN-LSTM neural networks to translate the detected plaintext words and math expressions into their markup

LaTeX. The system was evaluated on a new 102-page dataset composed of real-world scientific publications and achieved a better recognition rate than the previous state-of-the-art commercial software InftyReader.

CHAPTER VI

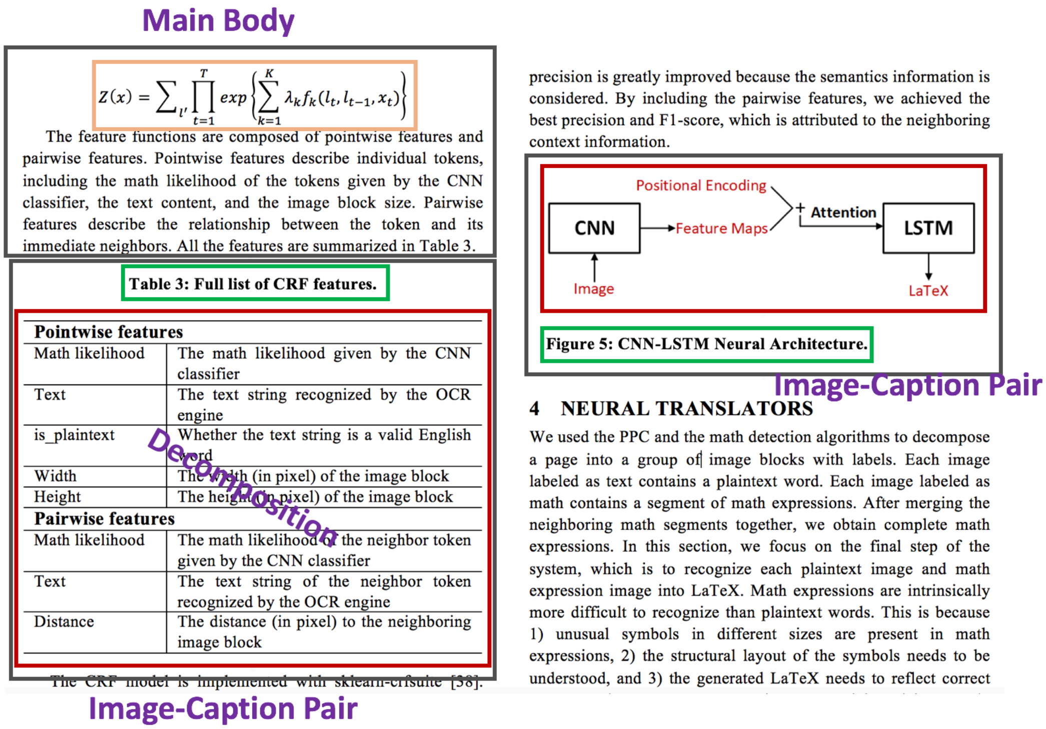
FIGURE AND TABLE EXTRACTION IN THE PDF2LATEX SYSTEM

Figures and tables play a critical role in presenting technical details and experiment results in research papers. In PDF documents, positions of figures and tables are unknown. Fortunately, several external tools have been developed for extracting these elements from PDF documents automatically. In this work, we will integrate the PDFFigures software into the PDF2LaTeX system, which can locate the bounding boxes of figures and tables, and can further extract their captions. We will also integrate the Camelot software, which can parse the detected tables and extract the cell contents from these tables. The extracted figures, tables, and their captions will be saved and formatted for information retrieval purposes. With the information given by these software, the page segmentation step in the PDF2LaTeX system will be able to skip table and figure areas, which enables the system to convert a much larger variety of real-world publications into their LaTeX sources. The proposed system is evaluated on a new dataset with 25 PDF pages, each containing text, math expressions, and figures/tables. The experiment results show that the proposed system achieved a comparable performance to the previous version of the PDF2LaTeX system on LaTeX conversion, and has extended capabilities to process pages with figures and tables.

Overview

Given the fast-growing number of digital publications, it is becoming increasingly important to develop algorithms that can automatically extract contents and mine knowledge from existing publications. While the PDF format helps the publishers and the readers to distribute digital publications very conveniently, it also brings difficulties on extracting non-textual elements from PDF documents, because the PDF format does not contain tags about its contents. The PDF2LaTeX system proposed in task 3 leveraged state-of-the-art machine learning algorithms and successfully extracted math expressions and converted documents to LaTeX [5, 44]. However, the system was not able to extract figures and tables, which are widely present in research papers and often contain important technical details and summaries of experiment results. In addition, the presence of figures and tables in a page can disrupt the page segmentation process of the PDF2LaTeX system and lead to recognition errors. As a result, a figure and table extraction module becomes a necessary add-on to the current PDF2LaTeX system. Even though information about figures and tables on a PDF page is not directly available, these elements usually occupy exclusive page areas and follow rigid layout patterns, thus are relatively easy to detect. A number of research tools and commercial software have been develop for figure and table extraction, such as PDFFigures [10] and Camelot [11]. Nevertheless, a comprehensive system that can parse various components (including text, math expressions, figures, and tables) in scientific papers still does not exist.

In this chapter, we will integrate the figure and table extraction modules into the existing PDF2LaTeX system to make it capable of processing pages with figures and tables. Firstly, we used the PDFFigures software to extract the bounding boxes of figures and tables, as well as the caption text associated with them. The segmentation method -- projection profile cutting used in the PDF2LaTeX system is updated to skip the figure and table areas, which are provided by the PDFFigures software. Figures contain diverse types of images and can be difficult to parse. As a result, we save the detected figures as image files for visualization and use their captions for indexing. Tables are different from figures in that their structure can be recognized relatively easily, thus their cell data can be extracted. We propose to use the Camelot software to extract the cell data from tables. The proposed system is a further step to the semantical understanding of paper contents. The resulting system is able to extract text, math expressions, figures, tables, and their captions all together and index the recognized contents for information retrieval purposes, as shown in Figure 23. The detected figures and tables can help the readers quickly grasp the gist of the technical contents. To evaluate the proposed system, we constructed a new dataset with 25 PDF pages, each containing a mixture of text, math expressions, and figures/tables. The proposed system converted these pages into their LaTeX sources, which were compare with their ground truth LaTeX sources. The evaluation results measured by string edit distance shows that the proposed system achieved a comparable performance to the previous version of the PDF2LaTeX system, which was evaluated on pages without figures and tables.



$$Z(x) = \sum_{l'} \prod_{t=1}^{T} exp\left\{\sum_{k=1}^{K} \lambda_k f_k(l_t, l_{t-1}, x_t)\right\}$$

The feature functions are composed of pointwise features and pairwise features. Pointwise features describe individual tokens, including the math likelihood of the tokens given by the CNN classifier, the text content, and the image block size. Pairwise features describe the relationship between the token and its immediate neighbors. All the features are summarized in Table 3.

Table 3: Full list of CRF features.

Pointwise features	
Math likelihood	The math likelihood given by the CNN classifier
Text	The text string recognized by the OCR engine
is_plaintext	Whether the text string is a valid English word
Width	The width (in pixel) of the image block
Height	The height (in pixel) of the image block
Pairwise features	
Math likelihood	The math likelihood of the neighbor token given by the CNN classifier
Text	The text string of the neighbor token recognized by the OCR engine
Distance	The distance (in pixel) to the neighboring image block

The CRF model is implemented with sklearn-crfsuite [38].

precision is greatly improved because the semantics information is considered. By including the pairwise features, we achieved the best precision and F1-score, which is attributed to the neighboring context information.

Figure 5: CNN-LSTM Neural Architecture.

4 NEURAL TRANSLATORS

We used the PPC and the math detection algorithms to decompose a page into a group of image blocks with labels. Each image labeled as text contains a plaintext word. Each image labeled as math contains a segment of math expressions. After merging the neighboring math segments together, we obtain complete math expressions. In this section, we focus on the final step of the system, which is to recognize each plaintext image and math expression image into LaTeX. Math expressions are intrinsically more difficult to recognize than plaintext words. This is because 1) unusual symbols in different sizes are present in math expressions, 2) the structural layout of the symbols needs to be understood, and 3) the generated LaTeX needs to reflect correct

Figure 23 The PDF2LaTeX system with figure and table detection capabilities. The main body is converted to LaTeX. The detected figures and captions are saved as image-caption pairs. The tables are decomposed into cell data.

Method

In this section, we first introduce two existing tools that can extract figures and tables from PDF documents. Next, we discuss how these tools are integrated into the PDF2LaTeX system, which enables it to process PDF documents with graphics components and convert them into LaTeX sources.

Figures and Tables Extraction Tools

he first step of extracting figures and tables is to locate their positions on a page. The *PDFFigures software* [10] has been developed by Clark et al. for this purpose. The cell data in tables also provides very useful information, thus we propose to use the

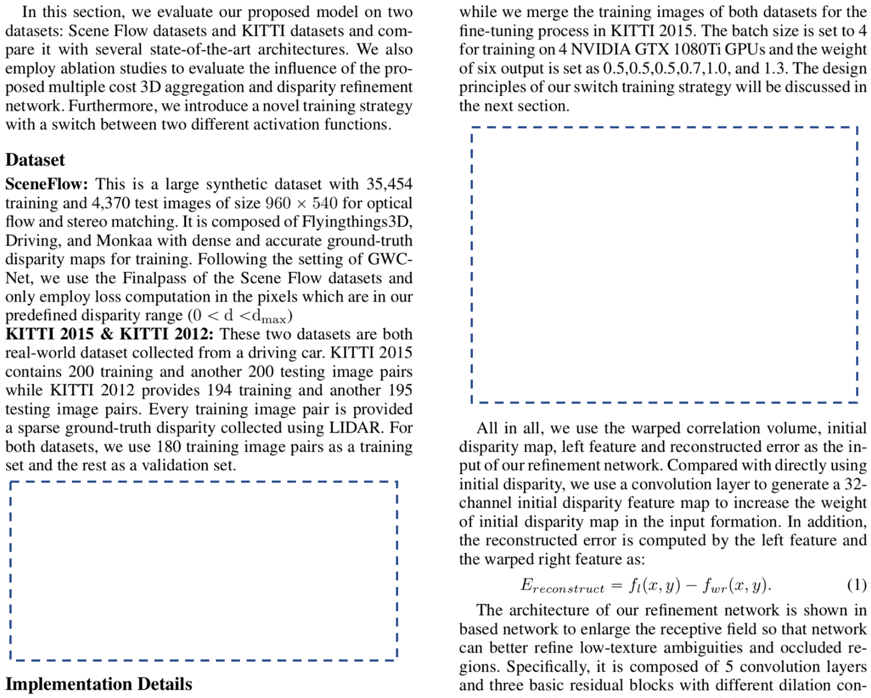
Camelot software [11] to decompose the detected tables into cell data. Below we briefly review the mechanisms of these tools.

PDFFigures is capable of extracting the bounding boxes of figures, tables, and their corresponding captions from a PDF page. It first extracts the text from a PDF page using the Apache PDFBox library [6]. Next, texts are grouped into text blocks which represent paragraphs, titles, captions, etc. Captions are then located using regular expression matching and a few heuristic rules about the caption words and fonts. Regions of figure and table are proposed by expanding the caption regions towards different vertical and horizontal directions without touching text blocks. Each proposed region is scored as potential figure and table regions by heuristic rules and assigned to captions.

Camelot software can be used to decompose tables in PDF documents into its cell data. The texts are extracted using PDFMiner [3]. Camelot provides two different algorithms to decompose a table. One algorithm is called *Lattice*, which first detects horizontal and vertical line boundaries in a table. Next, the intersection points of these lines are determined by using the "AND" operation of the line pixels. Finally, the cell data is located as the text in between the intersection points. Another algorithm is called *Stream*, which infers the location of cell data based on the text spatial layout. In this method, texts are first grouped into words and lines using spatial statistics. Next, the number of columns are determined as the mode of number of words in each row. Lines are then split into a list of column ranges. Finally, table cells are located as the intersection of rows and columns. By default, the choice of the algorithm need to be specified manually.

Integration with the PDF2LaTeX System

The previous PDF2LaTeX system was proposed to convert mathematical documents from the PDF format to the LaTeX format. The system effectively recognizes plaintext and math expression in PDF pages, but does not handle figures and tables. This is because the presence of figures and tables on a page would disrupt the projection profile cutting method used for page segmentation. This greatly limits the type of pages that PDF2LaTeX can process, because figures and tables are frequently used in many scientific publications. Knowing that figures and tables can be extracted with tools such as PDFFigures, we propose to integrate PDFFigures software into the existing PDF2LaTeX system.



In this section, we evaluate our proposed model on two datasets: Scene Flow datasets and KITTI datasets and compare it with several state-of-the-art architectures. We also employ ablation studies to evaluate the influence of the proposed multiple cost 3D aggregation and disparity refinement network. Furthermore, we introduce a novel training strategy with a switch between two different activation functions.

Dataset

SceneFlow: This is a large synthetic dataset with 35,454 training and 4,370 test images of size 960 × 540 for optical flow and stereo matching. It is composed of Flyingthings3D, Driving, and Monkaa with dense and accurate ground-truth disparity maps for training. Following the setting of GWC-Net, we use the Finalpass of the Scene Flow datasets and only employ loss computation in the pixels which are in our predefined disparity range ($0 < \mathrm{d} < \mathrm{d}_{\max}$)
KITTI 2015 & KITTI 2012: These two datasets are both real-world dataset collected from a driving car. KITTI 2015 contains 200 training and another 200 testing image pairs while KITTI 2012 provides 194 training and another 195 testing image pairs. Every training image pair is provided a sparse ground-truth disparity collected using LIDAR. For both datasets, we use 180 training image pairs as a training set and the rest as a validation set.

Implementation Details

while we merge the training images of both datasets for the fine-tuning process in KITTI 2015. The batch size is set to 4 for training on 4 NVIDIA GTX 1080Ti GPUs and the weight of six output is set as 0.5,0.5,0.5,0.7,1.0, and 1.3. The design principles of our switch training strategy will be discussed in the next section.

All in all, we use the warped correlation volume, initial disparity map, left feature and reconstructed error as the input of our refinement network. Compared with directly using initial disparity, we use a convolution layer to generate a 32-channel initial disparity feature map to increase the weight of initial disparity map in the input formation. In addition, the reconstructed error is computed by the left feature and the warped right feature as:

$$E_{reconstruct} = f_l(x, y) - f_{wr}(x, y). \qquad (1)$$

The architecture of our refinement network is shown in based network to enlarge the receptive field so that network can better refine low-texture ambiguities and occluded regions. Specifically, it is composed of 5 convolution layers and three basic residual blocks with different dilation con-

Figure 24 Figure and table areas are masked out by white rectangles, as indicated by the dash lines.

First, we use PDFFigures to locate the bounding boxes of figures and tables. Next, we use the PyPDF Python library to overlay a white rectangle in the area specified by the bounding box, which masks out the existing figures and tables with white pixels. An example is shown in Figure 24. The PDF page with the white mask is then rendered into an image, and processed by the remaining PDF2LaTeX steps. The rationale of this approach is that the white masks do not add pixel values to the projection profile during segmentation. The main body part of the page (plaintext and math expressions) is converted to LaTeX in the same way as before. For visualization purposes, the detected figures and tables will both be saved as image files. The detected captions of the figures and tables will be saved along with their corresponding figures and tables for indexing and retrieval purposes.

Comparing to figures, tables usually follow strict layout structures thus can be recognized relatively easily. The Camelot software has been designed to detect tables on a PDF page and decompose them into cell data. We observed that Camelot can be erroneous during the table detection phase, which results in failures during the decomposition phase. To solve this problem, instead of using the Camelot for table detection, we specify the locations of the tables and only use Camelot for table decomposition in the specified areas. The coordinates of tables are provided by the PDFFigures software. Notice that in the Camelot coordinate system, (0,0) is defined as the bottom left corner of the page. In PDFFigures, (0,0) is defined as the top left corner. As a result, we apply the following coordinates conversion:

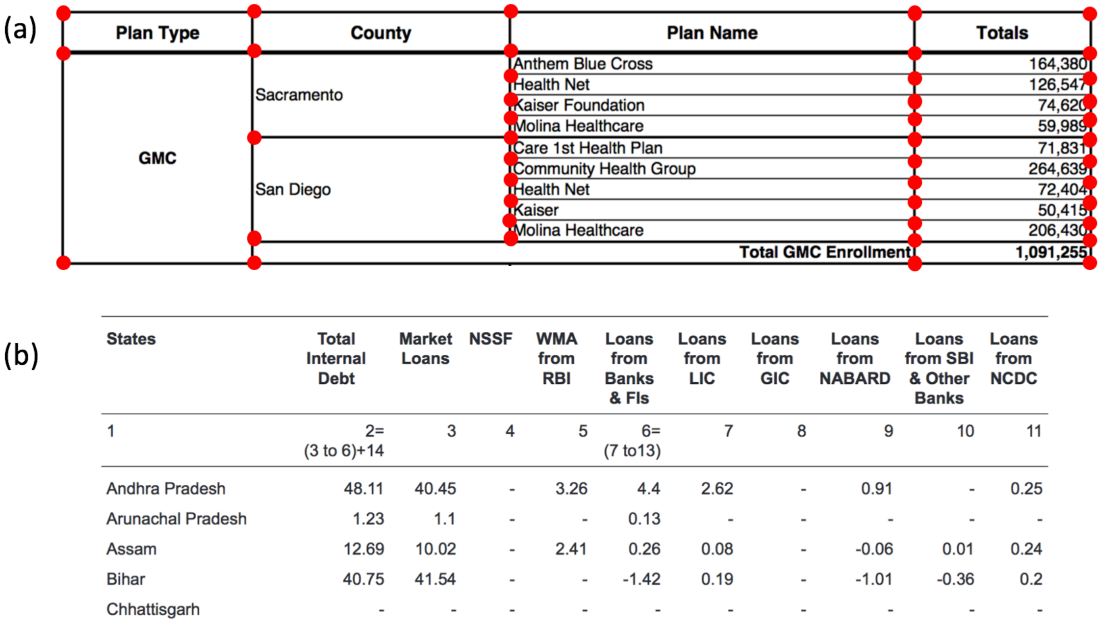
$$y_{1_{camelot}} = page_{height} - y_{2_{pdffigures}}$$

$$y_{2_{camelot}} = page_{height} - y_{1_{pdffigures}}$$

We also observed that for table decomposition, the two algorithms in Camelot (Lattice vs. Stream) have pros and cons on different kinds of tables. For example, the Lattice method works better for tables with full-line boundaries. The Stream method works better for tables whose lines are hidden and cell elements are indicated by spaces. Since there is not a method to automatically choose between the two algorithms, we propose to use an auto-selection rule as follows:

$$ratio = \frac{\#cells}{\#intersections}$$

$\#cells$ indicates the number of cells, which is provided by the Stream method. $\#intersections$ indicates the number of line intersections, which is provided by the Lattice method. If $\#intersections$ is zero, we set the ratio to 1. We use this ratio as a decision rule for algorithm selection. If the ratio is larger than a threshold, we select the Stream method, because a larger ratio indicates fewer line boundaries, which are likely to be hidden. If the ratio is smaller than a threshold, we select the Lattice method, because a smaller ratio indicates more line boundaries. In our implementation, we set the threshold to 0.8.


(a)

Plan Type	County	Plan Name	Totals
GMC	Sacramento	Anthem Blue Cross	164,380
		Health Net	126,547
		Kaiser Foundation	74,620
		Molina Healthcare	59,989
	San Diego	Care 1st Health Plan	71,831
		Community Health Group	264,639
		Health Net	72,404
		Kaiser	50,415
		Molina Healthcare	206,430
		Total GMC Enrollment	1,091,255

(b)

States	Total Internal Debt	Market Loans	NSSF	WMA from RBI	Loans from Banks & FIs	Loans from LIC	Loans from GIC	Loans from NABARD	Loans from SBI & Other Banks	Loans from NCDC
1	2= (3 to 6)+14	3	4	5	6= (7 to13)	7	8	9	10	11
Andhra Pradesh	48.11	40.45	-	3.26	4.4	2.62	-	0.91	-	0.25
Arunachal Pradesh	1.23	1.1	-	-	0.13	-	-	-	-	-
Assam	12.69	10.02	-	2.41	0.26	0.08	-	-0.06	0.01	0.24
Bihar	40.75	41.54	-	-	-1.42	0.19	-	-1.01	-0.36	0.2
Chhattisgarh	-	-	-	-	-	-	-	-	-	-

Figure 25 Lattice vs. Stream method for different kind of tables. (a) Lattice is chosen. (b) Stream is chosen.

An example of applying this rule is shown in Figure 25. The table in Figure 25 (a) contains 43 intersection points and 27 cells. This gives a ratio of 0.63, which is smaller than the threshold, thus the Lattice method is applied. On the other hand, the table in Figure 25 (b) has no intersection point, which yields a ratio of 1. This is larger than the threshold, thus the Stream method is applied. In our integrated system, the table contents recognized by Camelot are saved as *.CSV* files.

Experiments and Dataset

The performance of the proposed system is categorized by the accuracy of figure and table detection, the decomposition of table contents, and the LaTeX generated from the main body. The performance related to figures and tables has been previously reported by the authors of PDFFigures and Camelot [52]. As a result, we focus the system evaluation on the predicted LaTeX of the main body. The accuracy of the main body recognition is measured by the string edit distance between the predicted LaTeX and the corresponding ground truth LaTeX. A lower edit distance indicates a better performance. We also report the edit distance rate, which is defined as below:

$$edit\ dist\ rate = 1 - \frac{edit\ dist}{\#characters\ in\ the\ document}$$

The edit distance rate is more intuitive in that it is relative to the size of the document. A value closer to 1 indicates a better performance. We also applied some preprocessing steps before the evaluation. For the ground truth files, we removed the preambles before *\begin{document}* and *\after{document}*. We also replaced all the math claimers with '*$*', and removed section claimers, spaces, and ignored cases. For LaTeX that marks math expressions, we used the KaTeX parser [79] to normalize them in order to reduce the math polymorphism problem.

Since there is neither existing tools nor dataset for this task, we constructed a new dataset for testing. The dataset is composed of 25 PDF pages collected from real-world publications on arXiv. We manually selected pages that contain a mixture of text, math expressions, figures, and tables. Footnotes, references, and citations are removed from the pages. The portion of LaTeX sources that marks figures and tables are also removed during evaluation. There are 21 figures and 11 tables in total. Every PDF page in the dataset contains at least one figure or table. The main body part consists of 84,537 characters in total. For comparison, we used the previous PDF2LaTeX system without the figure and table extraction module as the baseline. Ideally, the performance of the new system on this new dataset should be comparable to the previous system evaluated on the PDF2LaTeX-102 dataset which does not contain figures or tables.

Results and Discussions

Table 17 Performance of the PDF2LaTeX system with and without figure/table extraction modules.

	Performance w/o figures/tables	Performance with figures/tables
Edit Dist (page)	479	446
Edit Dist (total)	48,820	11,155
Edit Dist Rate	81.1%	86.8%

In Table 17, we report the detailed evaluation results of the new system evaluated on the 25-page dataset. For the baseline, we present the evaluation results of the previous PDF2LaTeX system without figure/table extraction modules evaluated on the PDF2LaTeX-102 dataset, which does not contain figures and tables. As we can see, the new system achieved an edit distance of 446 per page. This is lower than the old system, which has an edit distance of 479 per page. When converted to the edit distance rate, the new system achieved an edit distance rate of 86.8%, while the old system achieved 81.1%. This means the performance of the proposed new system on pages with figures and tables is comparable to the performance of the old system on pages without figures or tables.

As we observed, the performance of the new system is slightly higher than the old system, even though the algorithm that processes the main body remains the same. The performance gain is due to the fact that there are fewer complicated math expressions in the 25-page dataset. This is reasonable because figures and tables present more often in the result sections, while math expressions present more often in the method sections. In other words, the pages with figures and tables tend to contain fewer math expressions, which are the main sources of LaTeX recognition errors.

For the computation cost, the new PDF2LaTeX system takes an average of 50s to process a page. This is evaluated on a machine with Intel Xeon 3.5Ghz CPU and 16GB RAM. Note that the main body part of a page is recognized by the neural networks, which can be parallelized on a GPU to speed up the system.

A Demo Application

The PDF2LaTeX system serves as the backend for parsing the PDF documents. This opens up a lot of possibilities for different applications. In this section, we demonstrate a PDF content viewer we developed as a front-end application for readers. The software aims to improve the reading experience of professionals who need to have access to the data inside the PDF documents. The software is written in Python Tkinter[5].

This application is a graphical-based PDF viewer, in which PDF pages are represented as images. Each element including tokens, MEs, figures, tables, etc. is wrapped in a rectangle object, which is augmented by the underlying data. Features of this interface include:

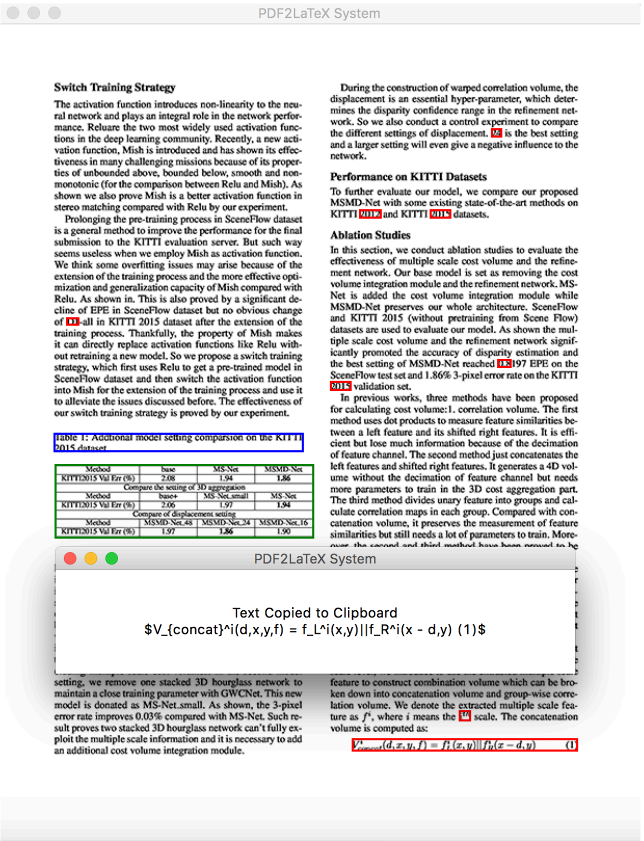
- highlights the critical elements on a page including MEs, figures, tables, and captions
- left-click to copy the underlying text of an object to clipboard
- left-click to open a table as a *.csv* file

[5] https://docs.python.org/3/library/tkinter.html (Accessed Aug 2020)

- right-click to copy the MathML of an ME to clipboard

MathML is generated in the backend based on the recognized markup LaTeX of an ME. The LaTeX to MathML conversion is realized by the Python software package *latex2mathml*[6]. MathML can be directly used in different environments, such as 1) display math expressions on webpages or 2) paste MathML in Microsoft Word which becomes an editable math formula object.

Figure 26 shows the interface of this software. The pop-up window shows that the source LaTeX of an ME being left-clicked is copied to the clipboard.



Switch Training Strategy

The activation function introduces non-linearity to the neural network and plays an integral role in the network performance. Reluare the two most widely used activation functions in the deep learning community. Recently, a new activation function, Mish is introduced and has shown its effectiveness in many challenging missions because of its properties of unbounded above, bounded below, smooth and nonmonotonic (for the comparison between Relu and Mish). As shown we also prove Mish is a better activation function in stereo matching compared with Relu by our experiment.

Prolonging the pre-training process in SceneFlow dataset is a general method to improve the performance for the final submission to the KITTI evaluation server. But such way seems useless when we employ Mish as activation function. We think some overfitting issues may arise because of the extension of the training process and the more effective optimization and generalization capacity of Mish compared with Relu. As shown in. This is also proved by a significant decline of EPE in SceneFlow dataset but no obvious change of [illegible]-all in KITTI 2015 dataset after the extension of the training process. Thankfully, the property of Mish makes it can directly replace activation functions like Relu without retraining a new model. So we propose a switch training strategy, which first uses Relu to get a pre-trained model in SceneFlow dataset and then switch the activation function into Mish for the extension of the training process and use it to alleviate the issues discussed before. The effectiveness of our switch training strategy is proved by our experiment.

Table 1: Additional model setting comparsion on the KITTI 2015 dataset

Method	base	MS-Net	MSMD-Net
KITTI2015 Val Err (%)	2.08	1.94	**1.86**
Compare the setting of 3D aggregation			
Method	base+	MS-Net_small	MS-Net
KITTI2015 Val Err (%)	2.06	1.97	**1.94**
Compare of displacement setting			
Method	MSMD-Net_48	MSMD-Net_24	MSMD-Net_16
KITTI2015 Val Err (%)	1.97	**1.86**	1.90

setting, we remove one stacked 3D hourglass network to maintain a close training parameter with GWCNet. This new model is donated as MS-Net_small. As shown, the 3-pixel error rate improves 0.03% compared with MS-Net. Such result proves two stacked 3D hourglass network can't fully exploit the multiple scale information and it is necessary to add an additional cost volume integration module.

During the construction of warped correlation volume, the displacement is an essential hyper-parameter, which determines the disparity confidence range in the refinement network. So we also conduct a control experiment to compare the different settings of displacement. 24 is the best setting and a larger setting will even give a negative influence to the network.

Performance on KITTI Datasets

To further evaluate our model, we compare our proposed MSMD-Net with some existing state-of-the-art methods on KITTI 2012 and KITTI 2015 datasets.

Ablation Studies

In this section, we conduct ablation studies to evaluate the effectiveness of multiple scale cost volume and the refinement network. Our base model is set as removing the cost volume integration module and the refinement network. MS-Net is added the cost volume integration module while MSMD-Net preserves our whole architecture. SceneFlow and KITTI 2015 (without pretraining from Scene Flow) datasets are used to evaluate our model. As shown the multiple scale cost volume and the refinement network significantly promoted the accuracy of disparity estimation and the best setting of MSMD-Net reached 0.8197 EPE on the SceneFlow test set and 1.86% 3-pixel error rate on the KITTI 2015 validation set.

In previous works, three methods have been proposed for calculating cost volume:1. correlation volume. The first method uses dot products to measure feature similarities between a left feature and its shifted right features. It is efficient but lose much information because of the decimation of feature channel. The second method just concatenates the left features and shifted right features. It generates a 4D volume without the decimation of feature channel but needs more parameters to train in the 3D cost aggregation part. The third method divides unary feature into groups and calculate correlation maps in each group. Compared with concatenation volume, it preserves the measurement of feature similarities but still needs a lot of parameters to train. Moreover, the second and third method have been proved to be

feature to construct combination volume which can be broken down into concatenation volume and group-wise correlation volume. We denote the extracted multiple scale feature as f^i, where i means the [illegible] scale. The concatenation volume is computed as:

$$V^i_{concat}(d,x,y,f) = f^i_L(x,y)||f^i_R(x-d,y) \quad (1)$$

Text Copied to Clipboard

```
$V_{concat}^i(d,x,y,f) = f_L^i(x,y)||f_R^i(x - d,y) (1)$
```

Figure 26 The graphical user interface of the PDF2LaTeX system.

[6] https://pypi.org/project/latex2mathml/ (Accessed Aug 2020)

Summary

We have proposed a new version of the PDF2LaTeX system, which is designed to extend the previous version of the system to process pages with figures and tables. The PDFFigures software was integrated to extract figures and tables from PDF pages. The Camelot software was integrated to decompose the extracted tables into cell data. In addition, captions of the figures and tables were also extracted and saved. In the end, the main body including the plaintext and math expressions were converted to LaTeX. For evaluation, we constructed a new dataset of 25 PDF pages from research papers which contains figures and tables. The evaluation results suggest that the performance of the new system is comparable to the previous system, but has extended capabilities for processing figures and tables. To our knowledge, this is the first system than can systematically parse real-world scientific publications and convert them to LaTeX. The system serves as the foundation to knowledge mining and information retrieval based on the vast amount of existing PDF documents.

CHAPTER VII

CONCLUSIONS

This dissertation presented a comprehensive system for content extraction and recognition in scientific publications, with a focus on the PDF format. These contents mainly include math expressions, figures, and tables, which carry important technical details but are very difficult for machine to process. The proposed content extraction algorithms help machines to locate the positions of different elements on a page and assign labels of their identities. The proposed recognition algorithms help machines understand the contents of these elements, and convert them to the LaTeX format, which can be directly used for NLP, searching, etc. This system serves as the foundation to applying a lot of different types of information retrieval and data mining technologies on top of PDF-based scientific publications.

Summary of Findings

In the first task, we designed a group of algorithms to extract math expressions from PDF documents. This is a necessary step not only for PDF documents with missing fonts, but also for those with correct fonts embedded, because the existence of MEs could disrupt text extraction and NLP analysis. Existing ME extraction algorithms either rely on the font information or require global training. The former approaches fail when correct fonts are missing. The latter approaches are usually supervised learning-based algorithms that are fine-tuned for specific datasets and thus do not generalize well outside the training dataset. This is a real problem when applying supervised machine learning algorithms in this field due to the lack of training data. A full exploitation of the font information helped us design the algorithm based on the font size feature, which is adaptive and robust, meanwhile does not require training. Another advantage is that the critical feature – font size, is not affected by the missing fonts problem. The multi-stage algorithm exploits the hierarchical relationship from symbols to the structural layout of MEs. At the same time, the design of the bigram regularization model borrows the idea from the Markov Random Field, which utilizes neighboring information for label predictions. The model stabilizes the predicted labels by penalizing label changes, thus reduces split detection errors. The model is simple and does not require training other than hand-tuning a penalty parameter. The mixed integer programming algorithm was used to efficiently optimize the bigram objective function. Finally, we introduced a semi-automated algorithm to generate the ME ground truth bounding boxes in an efficient manner. This is possible because the LaTeX sources of the PDF documents are essentially a variant of the labels of the content in PDF documents. The semi-automated solution we proposed links the labels in the LaTeX sources to the PDF documents. The outcome of this method is a large-scale dataset we released. The dataset not only serves as a new evaluation platform for future researchers, but also a better training source for supervised machine learning algorithms.

In the second task, we proposed a deep neural network to recognize MEs into LaTeX. To overcome the missing fonts problem, we proposed to recognize MEs as images. Recognition of MEs in images has been a research topic for decades. Traditional approaches involve symbol segmentation, individual symbol recognition, structural analysis, and finally, language model for sequence generation. In these approaches, an error generated in any stage cascades to the following steps. As compared to these approaches, we used an end-to-end prediction model, i.e., given input images, the model directly generates the output LaTeX strings. A deep learning model is made possible thanks to the tons of data that are easily available from the real-world LaTeX sources of existing scientific publications. The encoder-decoder architecture suits well for generating sequences from the images. The encoder CNN process the input images and encodes them into feature maps. The decoder RNN translates the feature maps into a sequence of LaTeX tokens, which marks up the contents in the input images. With the help of the attention mechanism, the model acts intelligently to focus on different parts of the image at every prediction step. Knowing the importance of the relative position of different symbols, we also introduced the 2-dimensional positional encoding, which adds sinusoidal signals to the feature maps as positional information. The sequence-level training objective also helped to enforce sequence-level correctness by considering the full sequence instead of individual tokens, which suits well for the rigid grammar of the LaTeX language. The policy gradient algorithm made it possible to train the model using a sequence-level evaluation metric in that it helps avoid taking derivatives on the discrete evaluation function but use the log-derivative of the likelihood instead.

In the third task, we introduced the PDF2LaTeX system, which focuses on page-level recognition. The system can process documents not only in PDF format, but also in image format. This requires an OCR-based solution, which is not subject to the limitations of PDF fonts. However, OCR-based solutions do not utilize the rich information provided by the PDF format. Fortunately, the recognition of text and MEs can be based on the success of the OCR engine proposed in the previous task, while individual English words and MEs are translated into LaTeX using the deep neural network. There still remains a challenge to locate the positions of text and MEs and identify their boundaries. We proposed to use the projection profile cutting algorithm to segment pages into columns, lines, and tokens. To classify if each token is an ME or plaintext, we trained a CNN as a binary classifier, which captures the visual features of the characters. Furthermore, we proposed to use a CRF to capture other features, especially the features of the neighboring tokens. The context information introduced by the CRF brought some additional performance gain. The deep neural network proposed in task 3 was directly used to recognize MEs and translates them into LaTeX. Plaintext recognition is considered a solved problem, but for efficiency, we trained another neural network with the same encoder-decoder architecture to recognize plaintext image blocks in batches. Since there exist no comprehensive dataset that are composed of real-world publications, we composed a new dataset with pages of mixed text and MEs. The PDF2LaTeX system is evaluated on this new dataset, and is compared with the commercial software InftyReader, which is the only know system that serves similar purposes. The PDF2LaTeX outperforms the InftyReader software, especially on differentiating ME vs. text and recognizing complex MEs.

In the fourth task, we integrated figure and table extraction modules into the PDF2LaTeX system. Although the system we proposed in task 3 was able to recognize mathematical documents at the state-of-the-art accuracy, it has strict assumptions that elements other than MEs and text do not exist. These elements are mainly composed of figures and tables, which often contain important information such as experiment results or technical specifications. Processing figures and tables not only provides valuable information, but is also a necessity in the PDF2LaTeX system, because the areas being occupied by figures and tables must be handled explicitly to make the projection profile cutting algorithm to give correct segmentation results. We proposed to integrate the PDFFigures software into our system for detection. The PDFFigures software helps locate the bounding boxes of figures, tables, as well as their captions. These elements are saved as image-caption pairs, which can be used in the information retrieval systems. The area of these elements are masked out by overlapping white padding on the corresponding page areas. We also integrated the Camelot software to decompose the detected tables into cell data, which are saved as .CSV files for easy data access. This greatly broadens the scope of the PDF2LaTeX system.

Future Work

We observed in task 1 that one obstacle of existing supervised learning-based ME detection solutions is the lack of training data. The MOP dataset is a pioneer work to generate large-scale ground truth data automatically. It contains around 1,800 pages of technical writings based on the LaTeX source files in the KDD dataset [91], which is sufficient as a test set. However, this may still not be enough for training some data-hungry but most effective machine learning models such as deep neural networks. The volume of pages is not the only problem since all the pages in the MOP dataset are from the high-energy physics subject on arXiv. To build more robust machine learning models, papers from a wider range of research areas are needed. Thus, a sensible extension to this work is to crawl more LaTeX sources of publications from different areas and possibly different publishers.

The math to LaTeX translator is an end-to-end neural network model, which conveniently bypasses the intermediate steps of math expressions recognition, which could be error-prone. These steps include character segmentation, symbol recognition, structural analysis, language model with LaTeX grammar, etc. While this is an exquisite design, at the same time it is also a compromise due to the lack of intermediate training data. The obvious disadvantage is the lack of interpretability. When a prediction error occurs in such models, it is nearly impossible to track the source of errors and correct it. Generating images of math expressions from LaTeX can be fully automated, but generating labels of individual symbols and their structural relationship requires tedious human labels. However, if such labels become available in the future, it could be very helpful to use several neural networks for different functions and train them separately. Another direction to improve the model is to replace the CNN and LSTM with more advanced neural architectures. For example, the ResNet [95] has been proposed very recently which outperforms CNN, thanks to the residual connections across different layers. This architecture can potentially help with extracting more precise representations of ME images.

We have tested both the PDF2LaTeX system and the InftyReader system on the PDF2LaTeX-102 dataset, and demonstrated that PDF2LaTeX can recognize pages in scientific documents at a better accuracy. That being said, as a research product, PDF2LaTeX has much more limited scope of applications as compared to the mature commercial software InftyReader. PDF2LaTeX is trained on arXiv papers rendered by TeX, and is tested on a dataset generated in the same way. Its performance would suffer when tested on unseen data. To make PDF2LaTeX as robust as the InftyReader, much larger and diverse data need to be fed into the training process. Such data is not yet available and the process requires some engineering arts. This is beyond the scope of this dissertation but is certainly a path towards a better tool for real-world usage. For future research related to the PDF pages-to-LaTeX task, it may also be helpful to introduce more comprehensive evaluation metrics in addition to the string edit distance. For example, it would be very beneficial to differentiate between the edit distance introduced by difference in font styles and the edit distance introduced by structural recognition errors, which would give insights to further improve the performance.

The figure and table extraction tools we integrated into the PDF2LaTeX system have decent performance but can always be easily replaced for better performance. PDFFigures and Camelot are PDF parser-based solutions, which has limitations when being applied to PDF documents with bad font encodings. An alternative is to use OCR-based figure and table detection approaches. For example, recent fast and accurate object detection neural networks such as YOLO [96] can be trained and integrated for figure and table detection purposes.

Finally, being a tool to give access to the various contents inside PDF documents, the PDF2LaTeX system opens many opportunities to information retrieval systems and data mining algorithms. Future researchers could build math information retrieval (MIR) systems such as math expression-based search engine in PDF documents. MIR has been a topic in the research community but it has not been able to work on PDF documents. With the help of PDF2LaTeX, the search engines can utilize the LaTeX being recovered to match the math keywords queries. A naive ranking criterion can be as simple as using the string edit distance between the query LaTeX and the target LaTeX. More advanced matching criteria may also be developed, such as matching the operator trees of MEs. One can simply convert MEs from LaTeX to Presentational MathML trees, or even to Content MathML trees which contains more semantical meanings. Efficient algorithms such as the pq-Gram [97] can be used to approximated tree edit distance very efficiently (in n*log*n time and linear space), which makes large-scale applications possible. One can go further to dive deep into the ME contents and extracts the declarations of math variables, which brings more semantical-level information for future processing. Some pioneer works on this direction can be found in [98, 99].

REFERENCES

[1] A. E. Jinha, "Article 50 million: an estimate of the number of scholarly articles in existence," *Learned Publishing,* vol. 23, no. 3, pp. 258-263, 2010.

[2] (2020, Feb 20). *arXiv submission rate statistics*. Available: https://arxiv.org/help/stats/2019_by_area

[3] (2020, Feb 7th). *PDFMiner*. Available: https://pypi.org/project/pdfminer/

[4] "Document management - Portable document format - Part 1: PDF 1.7," *Adobe Systems Incorporated,* p. 242, July 1st 2008.

[5] Z. Wang, D. Beyette, J. Lin, and J.-C. Liu, "Extraction of Math Expressions from PDF Documents based on Unsupervised Modeling of Fonts," in *IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, Australia, 2019: IEEE.

[6] (2020, Feb 7th). *Apache PDFBox*. Available: https://pdfbox.apache.org/

[7] S. Singh, "Optical character recognition techniques: a survey," *Journal of emerging Trends in Computing and information Sciences,* vol. 4, no. 6, pp. 545-550, 2013.

[8] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "INFTY: an integrated OCR system for mathematical documents," in *Proceedings of the 2003 ACM symposium on Document engineering*, 2003, pp. 95-104: ACM.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998-6008.

[10] C. Clark and S. Divvala, "Pdffigures 2.0: Mining figures from research papers," in *2016 IEEE/ACM Joint Conference on Digital Libraries (JCDL)*, 2016, pp. 143-152: IEEE.

[11] (2020, April 27th). *Camelot: PDF Table Extraction for Humans*. Available: https://camelot-py.readthedocs.io

[12] X. Lin, L. Gao, Z. Tang, X. Hu, and X. Lin, "Identification of embedded mathematical formulas in PDF documents using SVM," in *Document Recognition and Retrieval XIX*, 2012, vol. 8297, p. 82970D: International Society for Optics and Photonics.

[13] L. Gao, X. Yi, Y. Liao, Z. Jiang, Z. Yan, and Z. Tang, "A Deep Learning-Based Formula Detection Method for PDF Documents," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017, pp. 553-558: IEEE.

[14] J. B. Baker, A. P. Sexton, and V. Sorge, "Faithful mathematical formula recognition from PDF documents," in *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, 2010, pp. 485-492: ACM.

[15] K. Inoue, R. Miyazaki, and M. Suzuki, "Optical recognition of printed mathematical documents," in *Proceedings in the Third Asian Technology Conference on Math*, 1998, pp. 280-289.

[16] A. Kacem, A. Belaïd, and M. B. Ahmed, "Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context," *International Journal on Document Analysis and Recognition,* vol. 4, no. 2, pp. 97-108, 2001.

[17] (2019, Feb 7th). *Apache Tika*. Available: https://tika.apache.org/

[18] (2019, Feb 7th). *Poppler*. Available: https://poppler.freedesktop.org/

[19] R. Zanibbi and D. Blostein, "Recognition and retrieval of mathematical expressions," *International Journal on Document Analysis and Recognition (IJDAR),* vol. 15, no. 4, pp. 331-357, 2012.

[20] U. Garain, B. Chaudhuri, and A. R. Chaudhuri, "Identification of embedded mathematical expressions in scanned documents," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004, vol. 1, pp. 384-387: IEEE.

[21] X. Wang and J.-C. Liu, "A Font Setting Based Bayesian Model to Extract Mathematical Expression in PDF Files," in *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017, vol. 1, pp. 759-764: IEEE.

[22] K. Iwatsuki, T. Sagara, T. Hara, and A. Aizawa, "Detecting In-line Mathematical Expressions in Scientific Documents," in *Proceedings of the 2017 ACM Symposium on Document Engineering*, 2017, pp. 141-144: ACM.

[23] X. Lin, L. Gao, Z. Tang, J. Baker, and V. Sorge, "Mathematical formula identification and performance evaluation in PDF documents," *International*

Journal on Document Analysis and Recognition (IJDAR), vol. 17, no. 3, pp. 239-255, 2014.

[24] J. B. Baker, A. P. Sexton, and V. Sorge, "A linear grammar approach to mathematical formula recognition from PDF," in *International Conference on Intelligent Computer Mathematics*, 2009, pp. 201-216: Springer.

[25] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition,* vol. 3, no. 1, pp. 3-15, 2000.

[26] X. Wang, Z. Wang, and J.-C. Liu, "Bigram Label Regularization to Reduce Over-Segmentation on Inline Math Expression Detection," in *IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, Australia, 2019: IEEE.

[27] H. M. Twaakyondo and M. Okamoto, "Structure analysis and recognition of mathematical expressions," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 1995, vol. 1, pp. 430-437: IEEE.

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[29] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369-376: ACM.

[30] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep structured output learning for unconstrained text recognition," *arXiv preprint arXiv:1412.5903,* 2014.

[31] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156-3164.

[32] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048-2057.

[33] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks," in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, 2012, pp. 3304-3308: IEEE.

[34] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence,* vol. 39, no. 11, pp. 2298-2304, 2016.

[35] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025,* 2015.

[36] J. Zhang, J. Du, and L. Dai, "A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition," in *2017*

14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, vol. 1, pp. 902-907: IEEE.

[37] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognition,* vol. 71, pp. 196-206, 2017.

[38] Y. Deng, A. Kanervisto, and A. M. Rush, "What you get is what you see: A visual markup decompiler," *arXiv preprint arXiv:1609.04938,* vol. 10, pp. 32-37, 2016.

[39] Y. Deng, A. Kanervisto, J. Ling, and A. M. Rush, "Image-to-markup generation with coarse-to-fine attention," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2017, pp. 980-989: JMLR. org.

[40] J. Wang, Y. Sun, and S. Wang, "Image To Latex with DenseNet Encoder and Joint Attention," *Procedia computer science,* vol. 147, pp. 374-380, 2019.

[41] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700-4708.

[42] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T.-S. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5659-5667.

[43] W. Zhang, Z. Bai, and Y. Zhu, "An Improved Approach Based on CNN-RNNs for Mathematical Expression Recognition," in *Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing*, 2019, pp. 57-61: ACM.

[44] Z. Wang and J.-C. Liu, "Translating Mathematical Formula Images to LaTeX Sequences Using Deep Neural Networks with Sequence-level Training," *arXiv preprint arXiv:1908.11415,* 2019.

[45] R. Smith, "An overview of the Tesseract OCR engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007, vol. 2, pp. 629-633: IEEE.

[46] L. Gao, X. Yi, Z. Jiang, L. Hao, and Z. Tang, "ICDAR2017 competition on page object detection," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017, vol. 1, pp. 1417-1422: IEEE.

[47] 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.

[48] R. Saha, A. Mondal, and C. Jawahar, "Graphical Object Detection in Document Images," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2019, pp. 51-58: IEEE.

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, 2009, pp. 248-255: Ieee.

[50] 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 *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, 2017, vol. 1, pp. 1162-1167: IEEE.

[51] M. O. Perez-Arriaga, T. Estrada, and S. Abad-Mota, "TAO: system for table detection and extraction from PDF documents," in *The Twenty-Ninth International Flairs Conference*, 2016.

[52] C. Clark and S. Divvala, "Looking beyond text: Extracting figures, tables and captions from computer science papers.. 2015," in *AAAI 2015 Workshop on Scholarly Big Data*, 2015.

[53] D. D. Lewis and K. S. Jones, "Natural language processing for information retrieval," *Communications of the ACM,* vol. 39, no. 1, pp. 92-101, 1996.

[54] H. Déjean and J.-L. Meunier, "A system for converting PDF documents into structured XML format," in *International Workshop on Document Analysis Systems*, 2006, pp. 129-140: Springer.

[55] E. Oro and M. Ruffolo, "TREX: An approach for recognizing and extracting tables from PDF documents," in *ICDAR'09. 10th International Conference on Document Analysis and Recognition*, 2009, pp. 906-910: IEEE.

[56] F. Rahman and H. Alam, "Conversion of PDF documents into HTML: a case study of document image analysis," in *The Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, 2003, vol. 1, pp. 87-91: IEEE.

[57] U. Schöneberg and W. Sperber, "POS Tagging and its Applications for Mathematics," in *Intelligent Computer Mathematics*: Springer, 2014, pp. 213-223.

[58] M. Suzuki, Y. Terada, T. Kanahori, and K. Yamaguchi, "New Tools to Convert PDF Math Contents into Accessible e-Books Efficiently," *Studies in health technology and informatics,* vol. 217, pp. 1060-1064, 2015.

[59] X. Lin, L. Gao, Z. Tang, X. Lin, and X. Hu, "Performance evaluation of mathematical formula identification," in *10th IAPR International Workshop on Document Analysis Systems (DAS)*, 2012, pp. 287-291: IEEE.

[60] T. D. Smedt and W. Daelemans, "Pattern for python," *Journal of Machine Learning Research,* vol. 13, no. Jun, pp. 2063-2067, 2012.

[61] G. A. Miller, "WordNet: a lexical database for English," *Communications of the ACM,* vol. 38, no. 11, pp. 39-41, 1995.

[62] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[63] (2019, Jan 30). *CyLP software package*. Available: https://github.com/coin-or/CyLP

[64] X. Wang, "Missing MEs in Marmot dataset, in www.icst.pku.edu.cn/cpdp/data/marmot_data.htm, web posting: http://rtds.cse.tamu.edu/resources/, posted at Aug. 2017,"

[65] D. Beyette, Z. Wang, J. Lin, and J.-C. Liu, "Semi-Automatic LaTeX-Based Labeling of Mathematical Objects in PDF Documents: MOP Data Set," in

Proceedings of the ACM Symposium on Document Engineering 2019, 2019, pp. 1-4.

[66] B. Miller, "LaTeXML: A Latex to XML Converter. url: https://dlmf.nist.gov/LaTeXML/," *LaTeXML/(visited on 03/03/2020)*.

[67] P. Ion, R. Miner, S. Buswell, and A. Devitt, *Mathematical Markup Language (MathML) 1.0 Specification*. World Wide Web Consortium (W3C), 1998.

[68] R. H. Anderson, "Syntax-directed recognition of hand-printed two-dimensional mathematics," in *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, 1967, pp. 436-459: ACM.

[69] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, 2002, pp. 311-318: Association for Computational Linguistics.

[70] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057-1063.

[71] M. A. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732,* 2015.

[72] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in neural information processing systems*, 2014, pp. 2177-2185.

[73] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[74] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks,* vol. 5, no. 2, pp. 157-166, 1994.

[75] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, "Minimum risk training for neural machine translation," *arXiv preprint arXiv:1512.02433,* 2015.

[76] L. Wu, F. Tian, T. Qin, J. Lai, and T.-Y. Liu, "A study of reinforcement learning for neural machine translation," *arXiv preprint arXiv:1808.08866,* 2018.

[77] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning,* vol. 8, no. 3-4, pp. 229-256, 1992.

[78] S. Chatterjee and N. Cancedda, "Minimum error rate training by sampling the translation lattice," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 2010, pp. 606-615: Association for Computational Linguistics.

[79] (2019, Aug 25). *KaTex*. Available: https://katex.org/

[80] F. Álvaro, J.-A. Sánchez, and J.-M. Benedí, "An image-based measure for evaluation of mathematical expression recognition," in *Iberian Conference on Pattern Recognition and Image Analysis*, 2013, pp. 682-690: Springer.

[81] (2020, May 6th). *Mathpix Snip*. Available: https://mathpix.com/

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

[83] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* vol. 15, no. 1, pp. 1929-1958, 2014.

[84] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711,* 2012.

[85] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[86] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078,* 2014.

[87] (2020, Feb 20th). *NIST Digital Library of Mathematical Functions*. Available: https://dlmf.nist.gov/

[88] (2020, Feb 20th). *Wolfram Functions Site*. Available: http://functions.wolfram.com/

[89] E. Foulke, "Reading braille," *Tactual perception: A sourcebook,* vol. 168, 1982.

[90] K. Ashida, M. Okamoto, H. Imai, and T. Nakatsuka, "Performance evaluation of a mathematical formula recognition system with a large scale of printed formula

images," in *Second International Conference on Document Image Analysis for Libraries (DIAL'06)*, 2006, pp. 12 pp.-331: IEEE.

[91] J. Gehrke, P. Ginsparg, and J. Kleinberg, "Overview of the 2003 KDD Cup," *Acm Sigkdd Explorations Newsletter,* vol. 5, no. 2, pp. 149-151, 2003.

[92] (2020, March 3). *TeX Live*. Available: https://www.tug.org/texlive/

[93] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400,* 2013.

[94] M. Korobov, "sklearn-crfsuite (2015)," ed, 2019.

[95] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770-778.

[96] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779-788.

[97] N. Augsten, M. Böhlen, and J. Gamper, "The pq-gram distance between ordered labeled trees," *ACM Transactions on Database Systems (TODS),* vol. 35, no. 1, pp. 1-36, 2008.

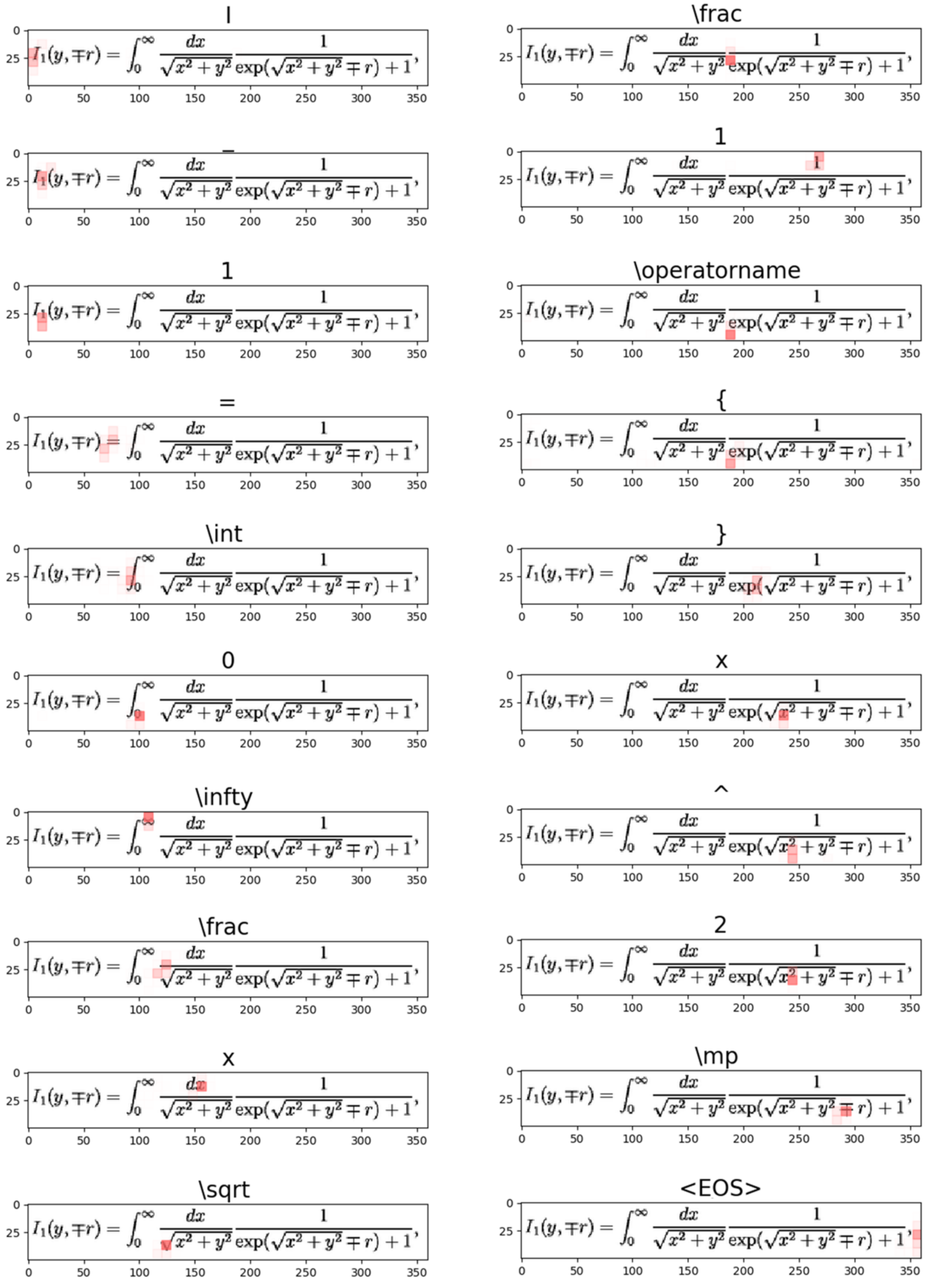
[98] J. Lin, X. Wang, Z. Wang, D. Beyette, and J.-C. Liu, "Prediction of Mathematical Expression Declarations based on Spatial, Semantic, and Syntactic Analysis," in *Proceedings of the ACM Symposium on Document Engineering 2019*, 2019, pp. 1-10.

[99] X. Wang, J. Lin, R. Vrecenar, and J.-C. Liu, "QuQn map: Qualitative-Quantitative mapping of scientific papers," in *Proceedings of the ACM Symposium on Document Engineering 2018*, 2018, pp. 1-4.

APPENDIX A

VISUALIZATION OF MI2LATEX

Figure 27 Visualization of the translation process for an input image. The image sequences are ordered vertically. The title of each image represents the token being produced at that certain time step. The red rectangles represent the attention weights. Darker color indicates a larger weight. We sampled 20 out of 77 predicted LaTeX tokens for concise presentation.



I

_

1

=

\int

0

\infty

\frac

x

\sqrt

\frac

1

\operatorname

{

}

x

^

2

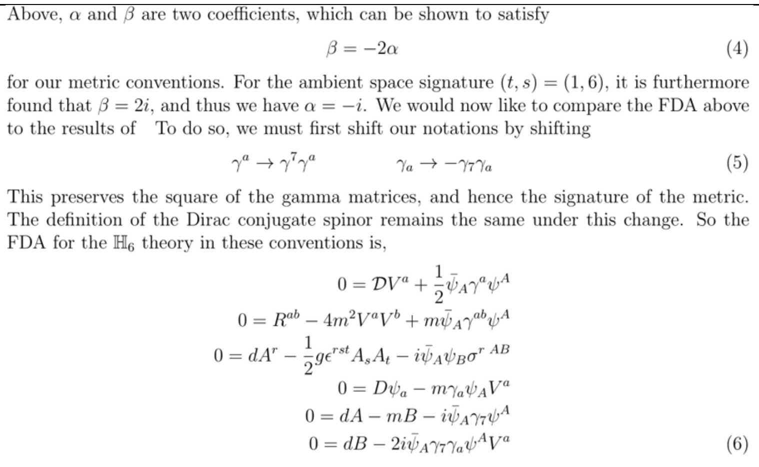
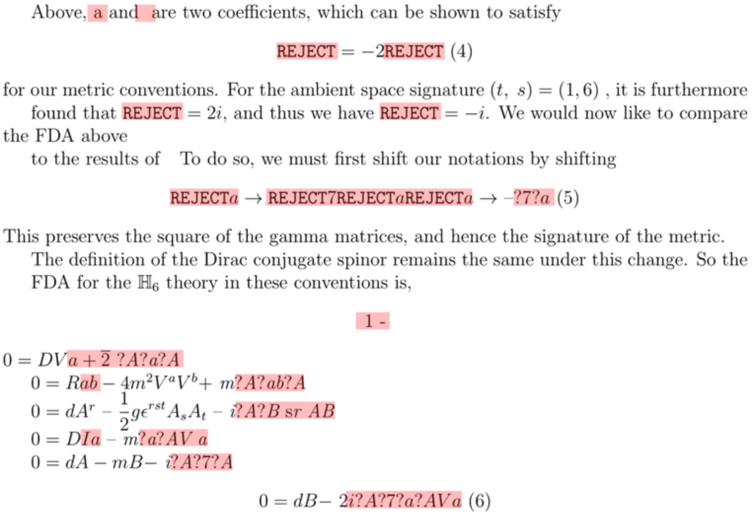
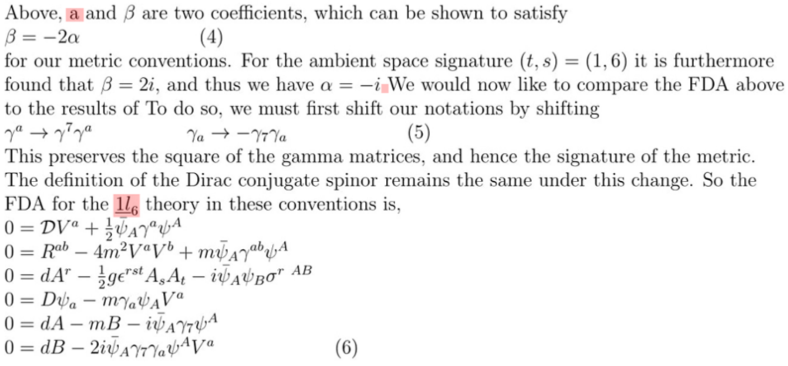
\mp

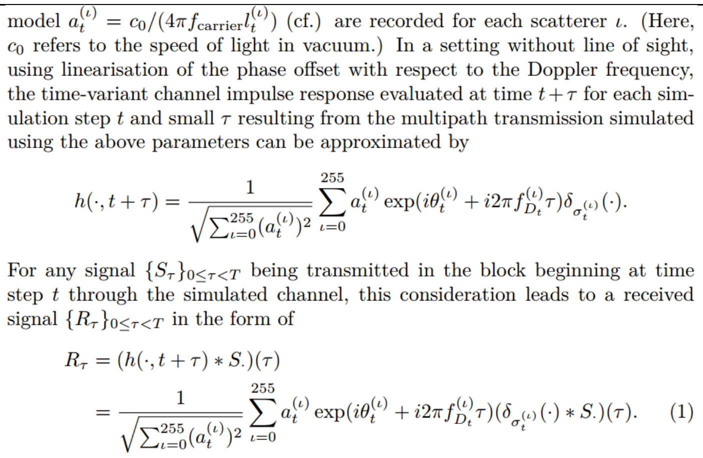
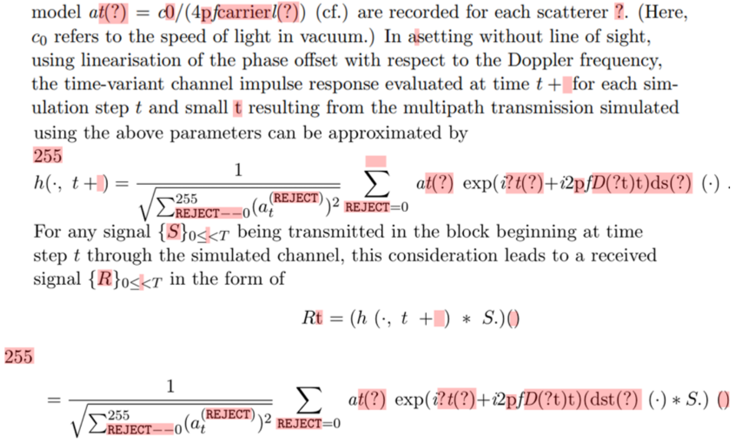
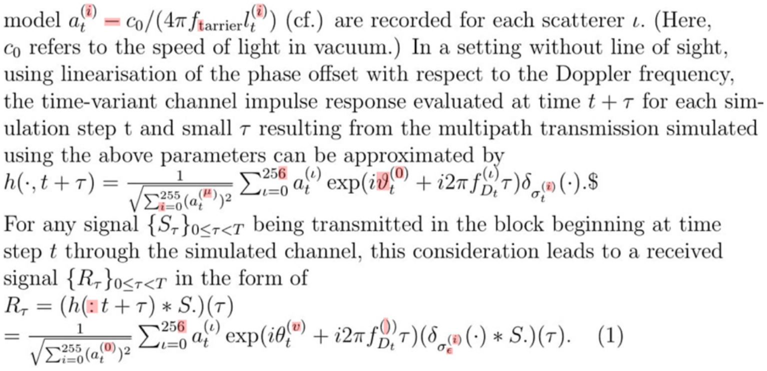
<EOS>

APPENDIX B

VISUALIZATION OF PDF2LATEX

Figure 28 Visualization of two examples from the PDF2LaTeX-102 dataset. The reconstruction errors are marked in red.

	“28.pdf”
Original	 Above, α and β are two coefficients, which can be shown to satisfy $\beta = -2\alpha$ (4) for our metric conventions. For the ambient space signature $(t, s) = (1, 6)$, it is furthermore found that $\beta = 2i$, and thus we have $\alpha = -i$. We would now like to compare the FDA above to the results of To do so, we must first shift our notations by shifting $\gamma^a \to \gamma^7\gamma^a \qquad \gamma_a \to -\gamma_7\gamma_a$ (5) This preserves the square of the gamma matrices, and hence the signature of the metric. The definition of the Dirac conjugate spinor remains the same under this change. So the FDA for the $\mathbb{H}_6$ theory in these conventions is, $0 = \mathcal{D}V^a + \frac{1}{2}\bar{\psi}_A\gamma^a\psi^A$ $0 = R^{ab} - 4m^2V^aV^b + m\bar{\psi}_A\gamma^{ab}\psi^A$ $0 = dA^r - \frac{1}{2}g\epsilon^{rst}A_sA_t - i\bar{\psi}_A\psi_B\sigma^{r\ AB}$ $0 = D\psi_a - m\gamma_a\psi_AV^a$ $0 = dA - mB - i\bar{\psi}_A\gamma_7\psi^A$ $0 = dB - 2i\bar{\psi}_A\gamma_7\gamma_a\psi^AV^a$ (6)
InftyReader	 Above, a and are two coefficients, which can be shown to satisfy REJECT = −2REJECT (4) for our metric conventions. For the ambient space signature $(t,\ s) = (1, 6)$, it is furthermore found that REJECT $= 2i$, and thus we have REJECT $= -i$. We would now like to compare the FDA above to the results of To do so, we must first shift our notations by shifting REJECTa → REJECT7REJECT$a$REJECT$a$ → −?7?a (5) This preserves the square of the gamma matrices, and hence the signature of the metric. The definition of the Dirac conjugate spinor remains the same under this change. So the FDA for the $\mathbb{H}_6$ theory in these conventions is, 1 - $0 = DVa + \bar{2}\ ?A?a?A$ $0 = Rab - 4m^2V^aV^b + m?A?ab?A$ $0 = dA^r - \frac{1}{2}g\epsilon^{rst}A_sA_t - i?A?B\ sr\ AB$ $0 = DIa - m?a?AV\ a$ $0 = dA - mB - i?A?7?A$ $0 = dB - 2i?A?7?a?AVa$ (6)
PDF2LaTeX	 Above, a and β are two coefficients, which can be shown to satisfy $\beta = -2\alpha$ (4) for our metric conventions. For the ambient space signature $(t, s) = (1, 6)$ it is furthermore found that $\beta = 2i$, and thus we have $\alpha = -i$ We would now like to compare the FDA above to the results of To do so, we must first shift our notations by shifting $\gamma^a \to \gamma^7\gamma^a \qquad \gamma_a \to -\gamma_7\gamma_a$ (5) This preserves the square of the gamma matrices, and hence the signature of the metric. The definition of the Dirac conjugate spinor remains the same under this change. So the FDA for the $\underline{1l}_6$ theory in these conventions is, $0 = \mathcal{D}V^a + \frac{1}{2}\bar{\psi}_A\gamma^a\psi^A$ $0 = R^{ab} - 4m^2V^aV^b + m\bar{\psi}_A\gamma^{ab}\psi^A$ $0 = dA^r - \frac{1}{2}g\epsilon^{rst}A_sA_t - i\bar{\psi}_A\psi_B\sigma^{r\ AB}$ $0 = D\psi_a - m\gamma_a\psi_AV^a$ $0 = dA - mB - i\bar{\psi}_A\gamma_7\psi^A$ $0 = dB - 2i\bar{\psi}_A\gamma_7\gamma_a\psi^AV^a$ (6)

	“66.pdf”
Original	 model $a_t^{(\iota)} = c_0/(4\pi f_{\text{carrier}} l_t^{(\iota)})$ (cf.) are recorded for each scatterer ι. (Here, c_0 refers to the speed of light in vacuum.) In a setting without line of sight, using linearisation of the phase offset with respect to the Doppler frequency, the time-variant channel impulse response evaluated at time $t+\tau$ for each simulation step t and small τ resulting from the multipath transmission simulated using the above parameters can be approximated by $$h(\cdot,t+\tau) = \frac{1}{\sqrt{\sum_{\iota=0}^{255}(a_t^{(\iota)})^2}} \sum_{\iota=0}^{255} a_t^{(\iota)} \exp(i\theta_t^{(\iota)} + i2\pi f_{D_t}^{(\iota)}\tau)\delta_{\sigma_t^{(\iota)}}(\cdot).$$ For any signal $\{S_\tau\}_{0\le\tau<T}$ being transmitted in the block beginning at time step t through the simulated channel, this consideration leads to a received signal $\{R_\tau\}_{0\le\tau<T}$ in the form of $$R_\tau = (h(\cdot,t+\tau)*S.)(\tau)$$ $$= \frac{1}{\sqrt{\sum_{\iota=0}^{255}(a_t^{(\iota)})^2}} \sum_{\iota=0}^{255} a_t^{(\iota)} \exp(i\theta_t^{(\iota)} + i2\pi f_{D_t}^{(\iota)}\tau)(\delta_{\sigma_t^{(\iota)}}(\cdot)*S.)(\tau). \quad (1)$$
InftyReader	 model $at(?) = c0/(4pfcarrierl(?))$ (cf.) are recorded for each scatterer ?. (Here, c_0 refers to the speed of light in vacuum.) In asetting without line of sight, using linearisation of the phase offset with respect to the Doppler frequency, the time-variant channel impulse response evaluated at time $t+$ for each simulation step t and small t resulting from the multipath transmission simulated using the above parameters can be approximated by 255 $h(\cdot,\ t+\ ) = \frac{1}{\sqrt{\sum_{\text{REJECT}--0}^{255}(a_t^{(\text{REJECT})})^2}} \sum_{\text{REJECT}=0} at(?)\ \exp(i?t(?)+i2pfD(?t)t)ds(?)\ (\cdot)$. For any signal $\{S\}_{0\le<T}$ being transmitted in the block beginning at time step t through the simulated channel, this consideration leads to a received signal $\{R\}_{0\le<T}$ in the form of $$Rt = (h\ (\cdot,\ t\ +\ )\ *\ S.)()$$ 255 $$= \frac{1}{\sqrt{\sum_{\text{REJECT}--0}^{255}(a_t^{(\text{REJECT})})^2}} \sum_{\text{REJECT}=0} at(?)\ \exp(i?t(?)+i2pfD(?t)t)(dst(?)\ (\cdot)*S.)\ ()$$
PDF2LaTeX	 model $a_t^{(i)} = c_0/(4\pi f_{\text{tarrier}} l_t^{(i)})$ (cf.) are recorded for each scatterer ι. (Here, c_0 refers to the speed of light in vacuum.) In a setting without line of sight, using linearisation of the phase offset with respect to the Doppler frequency, the time-variant channel impulse response evaluated at time $t+\tau$ for each simulation step t and small τ resulting from the multipath transmission simulated using the above parameters can be approximated by $h(\cdot,t+\tau) = \frac{1}{\sqrt{\sum_{i=0}^{255}(a_t^{(\mu)})^2}} \sum_{\iota=0}^{256} a_t^{(\iota)} \exp(i\vartheta_t^{(0)} + i2\pi f_{D_t}^{(\iota)}\tau)\delta_{\sigma_t^{(i)}}(\cdot).\$$ For any signal $\{S_\tau\}_{0\le\tau<T}$ being transmitted in the block beginning at time step t through the simulated channel, this consideration leads to a received signal $\{R_\tau\}_{0\le\tau<T}$ in the form of $R_\tau = (h(:t+\tau)*S.)(\tau)$ $= \frac{1}{\sqrt{\sum_{i=0}^{255}(a_t^{(0)})^2}} \sum_{\iota=0}^{256} a_t^{(\iota)} \exp(i\theta_t^{(v)} + i2\pi f_{D_t}^{()}\tau)(\delta_{\sigma_\epsilon^{(i)}}(\cdot)*S.)(\tau). \quad (1)$

APPENDIX C

EXTERNAL SOURCES

Table 18 External sources used including datasets and tools.

Datasets	
Marmot Dataset	https://www.icst.pku.edu.cn/cpdp/sjzy/index.htm
KDD Cup 2003 Dataset	https://research.cs.cornell.edu/kddcup/datasets.html
Tools	
PDFFigures 2.0	http://pdffigures2.allenai.org/
Camelot	https://camelot-py.readthedocs.io/en/master/
sklearn-crfsuite	https://sklearn-crfsuite.readthedocs.io/en/latest/
KaTeX	https://katex.org/
PDFBox	https://pdfbox.apache.org/
PDFMiner	https://pdfminer-docs.readthedocs.io/pdfminer_index.html
InftyReader	http://www.inftyreader.org/