EditTrans: Speedy Edit-based Detailed Transformation of Academic Documents into Markup

Anonymous ACL submission

Abstract

Academic Documents stored in PDF format can be transformed into plain text structured markup languages to enhance accessibility. Markup languages allow for easier updates and customization, making academic content more adaptable and accessible to diverse usage, such as linguistic corpus compilation.

Existing end-to-end decoder transformer models can transform screenshots of documents into markup language, their flexibility is superior to encoder transformers based on Document Layout Analysis. However, decoder transformers have more parameters and operate more slowly. Their token-by-token decoding from scratch wastes a lot of inference steps in generating dense text, which can be directly copied from PDF files.

To solve this problem, we introduce EditTrans, whose features allow identifying a queue of to-be-edited text from a PDF before starting to generate markup language. EditTrans contains a lightweight classifier that is fine-tuned from a Document Layout Analysis model on 162,127 pages of documents from arXiv. In our evaluations, EditTrans reduced the number of generation steps by 42.9% compared to end-to-end decoder transformer models.

1 Introduction

Transforming Academic Documents (AD) from PDF to markup languages such as HTML or Markdown significantly enhances their accessibility and usability. This conversion not only improves web accessibility but also boosts document interactivity, enhances search-ability and indexing, and guarantees compatibility across different platforms (Frankston et al., 2024). Such documents typically delivered in PDF format contain complex elements including mathematical formulas, figures, headers, and tables, as well as densely layouted text. ADs vary greatly in layout and content, posing challenges for in computational document processing (Li et al., 2020b). In order to overcome these challenges and implement a faithful extraction process, a precise Document Understanding (DU) is required, which enables accurate reproduction of text, figures, and tables in a structured format, ensuring the integrity and functionality of the original document are maintained in the new markup file.

DU predominantly refers to the process of automated classifying, and extracting information with rich typesetting formats from digital-born documents or scanned documents (Cui et al., 2021). One method involves using a transformer encoder for Document Layout Analysis, followed by text content extraction and understanding (Huang et al., 2022). Recent works focus on document screenshots due to the generality and complexity of the models (Lee et al., 2023). For instance, Donut (Kim et al., 2022) is an end-to-end transformer decoder model for DU from screenshots. Based on Donut's development, Nougat (Blecher et al., 2023) was introduced as a method that transforms academic PDFs into Markdown, a markup language.

However, Nougat has drawbacks due to processing speed because it generates text token-by-token from scratch which significantly slows down the overall document transformation process. Given that ADs frequently contain dense text that can be directly copied from PDFs, adopting an edit-based approach should speed up the transformation process and save computational costs.

Text-editing models have become a prominent alternative for monolingual text-generation tasks with high degree of textual overlap between the source and target, such as Grammatical Error Correction, Style Transfer, and Text Simplification (Malmi et al., 2022). These models focus on making minimal changes to adapt or correct the existing text, which also fits the paradigm of AD transformation.

In this paper, our contributions are:

- EditTrans which can identify and put copyable text from PDF into the edit queue before Nougat generation starts.

- It is lightweight with only 1.1M trainable parameters and a weights file size of less than 5MB.

- We release the dataset-making scripts as well as the arXiv numbers of the documents in the experiments to enhance reproducibility and observe copyright.

2 Related Work

2.1 Academic Documents Transformation

GROBID (GRO, 2008–2024) is a machine learning library for extracting, parsing, and re-structuring documents including PDF into structured XML encoded documents. However, it is not flexible because it converts formulas and tables into images thus hampering subsequent accessibility. docTR (Mindee, 2021) and DocBed (Zhu et al., 2022) first identify the document layout and then extract text content. Donut (Kim et al., 2022), is a Document Understanding model consisting of a visual encoder and language model decoder without obtaining texts directly from the document. Nougat (Blecher et al., 2023) follows Donut in implementing screenshot-to-Markdown transformation of Academic Documents. LOCR (Sun et al., 2024) solves the problem of Nougat's hallucination and repetition using an additional location prompt. Kosmos-2.5 (Lv et al., 2023) and DocOwl-1.5 (Hu et al., 2024) implement a more generalized screenshot-to-Markdown transformation with Vision-Language methods and larger model size.

The approach described in this paper is an attempt to edit Nougat's input sequence to speed up the transformation.

2.2 Document Layout Analysis (DLA)

Recent DLA models have become increasingly powerful thanks to the availability of large-scale document layout datasets (Zhong et al., 2019; Li et al., 2020b; Pfitzmann et al., 2022; Jaume et al., 2019). Computer Vision models have been able to extract layouts in screenshots of documents (Yang and Hsu, 2021; Li et al., 2020a; Wu et al., 2021). Language models have also been applied to recognize layouts. LayoutLM (Xu et al., 2020) and its variant VILA (Shen et al., 2022) are transformer encoder models that analyze document layouts from the texts and their 2D coordinates. LayoutLMv2 and 3 (Xu et al., 2021; Huang et al., 2022) additionally attaches visual features to the transformer encoder.

In this work, our Copyable Text Identification model is fine-tuned from LayoutLMv3 (Huang et al., 2022).

2.3 Text Generation with Text-Editing Models

Transformers decoder models generate outputs token-by-token from scratch thus making them slow at inference time. Text-editing models provide several benefits over decoder models including faster inference speed, higher sample efficiency as well as better control and interpretability of the outputs (Malmi et al., 2022). LaserTagger (Malmi et al., 2019) implements the Sentence Fusion task with three actions: `KEEP`, `DELETE`, and `REPLACE`. FELIX (Mallinson et al., 2020) and EdiT5 (Mallinson et al., 2022) also achieve text reordering. PIE (Awasthi et al., 2019), Seq2Edits (Stahlberg and Kumar, 2020) and GECToR (Omelianchuk et al., 2020) edit the text using the Iterative Refinement approach. There is a blog post[1] about Google Search correcting user input using EdiT5 (Mallinson et al., 2022) with low-latency features.

Our work organizes copyable text into edit queues, which mimics the behavior of Text-editing models.

3 Methodology

EditTrans streamlines academic document transformation into three steps: (1) EditTrans begins by classifying spans extracted from PDF pages and identifying which portions of the spans are copyable; (2) EditTrans then organizes the classified spans into an edit queue and delineates a stop criterion for each edit needed; (3) For each span requiring editing, EditTrans utilizes the pre-trained Nougat. (Blecher et al., 2023) model to execute the necessary edits. Figure 1 briefly demonstrates how we can copy text from a PDF and save Nougat's inference steps.

EditTrans is expected to produce the same output as Nougat (i.e. generate Markdown). But, EditTrans requires a PDF page as input while Nougat requires a screenshot of the PDF page.

[1] `https://research.google/blog/grammar-checking-at-google-search-scale/`

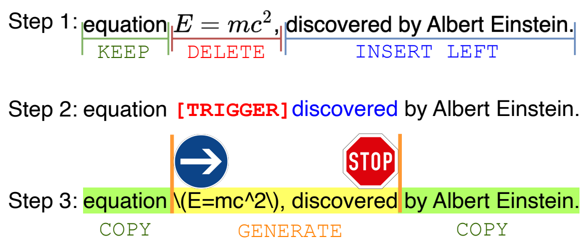

Figure 1: An overview of how EditTrans works. Step 1 detects whether the span is copyable or not. Step 2 constructs an edit queue, [TRIGGER] is the edit trigger, and blue word is the edit stop sign. Step 3 executes the edit, where the green part is copied from the PDF and the yellow part is generated by Nougat.

3.1 Copyable Text Identification

Inspired by DLA-related work, we assert that whether the text is copyable or not is highly correlated with its layout information. Specifically, we suggest that: (1) Dense plain text found in paragraphs should be preserved in its entirety. (2) Page elements such as mathematical formulas, tables, and titles should be modified to align with Markdown formatting standards. (3) Elements that do not convey relevant content, including page headers, footers, and page numbers, should be excluded from the final document.

Following VILA (Shen et al., 2022), we assume that text copyability is homogeneous at the span level. We use PyMuPDF[2] to extract span-level text and bounding boxes from the PDF. Subsequently, we fine-tune the LayoutLMv3 model for token classification using LoRA (Hu et al., 2022), omitting global 1D position embeddings to prevent potential biases in layout judgment (Tu et al., 2023).

We altered LayoutLMv3's classification head to predict labels as `KEEP`, `DELETE`, or `INSERT_LEFT`. `KEEP` indicates that the span should be included in the Markdown output without editing, `DELETE` indicates that the span should be deleted, and `INSERT_LEFT` indicates that a trigger for Nougat generation should be inserted before this span.

This approach is notably different from already existing text-editing models. Our editing logic in this paper is to delete text that should not be copied, and then insert tokens that should be generated by the decoder, e.g., a mathematical formula is deleted from the PDF, and an equivalent expression is generated before the next span of copiable text.

As we have fine-tuned LayoutLMv3 for token classification, and each span may contain more than one token, a voting classifier (Diem et al., 2011) is applied to decide the prediction of the spans. Details of the fine-tuning hyperparameters are documented in Appendix A.

[2] `https://github.com/pymupdf/PyMuPDF`

3.2 Edit Queue Building

Once we have the span-level edit annotation, we can turn it into an edit queue Q that prompts the pre-trained Nougat model for which portion of the text to edit. Each edit queue Q starts with an edit trigger, followed by a sequential processing of each span. We iterate through each span in edit annotation.

If next span is predicted to label `KEEP`, we add span text to Q. Note that if the length of the text characters in this span is less than 5, we do not add it and instead expect Nougat to generate it because too short a text makes it difficult to match where Nougat should stop generating. We then match the first word in the text of this span with a character length greater than 3 to sign the Nougat model stop generation and start copy.

If next span is predicted as `DELETE`, we do not add anything to Q.

If next span is `INSERT_LFET` we will add an edit trigger first, and then add this span's text sequence to Q. Similarly to the `KEEP` span, we will match the first word with a character length greater than 3 as an edit stop sign to Nougat.

At the end, we will add an extra edit trigger to allow Nougat to generate end-of-sentence tokens.

3.3 Markup Edits Generation

In this step, we initialize an empty tokens sequence S and traverse the edit queue Q.

If the next element in Q is an edit trigger, a screenshot of the page and S are fed into the Nougat model. Nougat's generation phase involves the decoding of an auto-regressive model, where it predicts the next token in sequence until a stopping criterion is met. We set the stopping criteria to be the pre-selected stop sign of the next to-be-copied span or end-of-sentence tokens. The output of the Nougat model is added to S.

If the next element in the queue is a to-be-copied span, we simply tokenize the text of this span and add it to the end of S.

In a nutshell, we copy the simple text from the PDF and leave Nougat in charge of generating the complex parts, such as formulas and tables. Finally, S is outputted and detokenized into Markdown format.

4 Dataset Building

As there is no existing dataset released as PDFs at this time, we downloaded the LaTeX source code bundles for the July and August 2023 papers from arXiv. Then we use a framework (Duan et al., 2023) that compiles LaTeX to PDF, plus annotates for semantic labels, reading order, and LaTeX code corresponding to mathematical formulas and tables for each element on a page. A part of the downloaded source code of the papers was not annotated successfully, because it was written in a way that the framework could not parse. A total number of 14,320 papers were annotated.

Spans are extracted from these pages and are labeled as either `KEEP`, `DELETE`, or `INSERT_LEFT`, based on the results of the semantic annotation of the previous step. We mark the captions of figures and tables as `DELETE` because they are reordered to the end of the page in Nougat.

Pages that are empty or challenging to read, such as those containing full-page images, long tables, or bibliographies, are excluded from the dataset. Finally, a dataset was assembled consisting of 180.146 pages, each annotated with span-level text copyable labels and their corresponding bounding boxes. We randomly split the training set size to 162,127 and the test set to 18,019. The vast majority of the pages in this dataset are in English.

We then attached a Markdown target for each page, which emulates Nougat's style of inserting mathematics formulas and tables as LaTeX code. LaTeX is quite flexible because it allows user-defined macros. Therefore, we normalize the formula and table LaTeX codes with LaTeXML[3].

The method in this paper extracts text spans from PDFs, which requires access to the full-text of academic papers. As arXiv does not grant permission to repost the full-text[4], we publish the scripts for creating the datasets plus the dataset's arXiv numbers to provide reproducibility.

5 Results

Following Nougat (Blecher et al., 2023), we use edit distance (Levenshtein, 1966) and F-measures to evaluate transformation quality. The baseline model is a pre-trained `nougat-base`[5] model. Our fine-tuned LayoutLMv3 (Huang et al., 2022) model achieves an F1-score of 0.92 on the Copyable Text Identification task.

Models	Edit dist ↓	F1 ↑	Steps ↓
nougat-base	0.1119	0.882	495.03
EditTrans	0.1114	0.901	**282.77**

Table 1: Comparative performance results on the arXiv test set. Findings demonstrate that for pages amenable to transformation by Nougat, EditTrans significantly reduces the number of inference steps required and ensures a high-quality document transformation.

We noticed that on some pages, especially if the page contains many formulas or tables, Nougat tends to hallucinate, becomes repetitive, and simply fails to hit EditTrans' stop sign in the edit queue which results in EditTrans not working. Therefore, we selected the test set samples that could be transformed to Markdown by Nougat without too many errors, specifically, we chose test set sample pages with an edit distance of less than 0.25 in Nougat's baseline transformation results.

Table 1 shows that EditTrans saves 42.9% inference steps while maintaining transformation quality. We provide code, weights, and example data in supplementary material, they will be open-sourced.

6 Conclusion and Future Work

In this paper, we introduce EditTrans, a lightweight text-editing PDF to Markdown Academic Documents Transformation tool, which is based on off-the-shelf models LayoutLMv3 (Huang et al., 2022) and Nougat (Blecher et al., 2023). We performed minimal fine-tuning and the weights file size is less than 5MB. EditTrans accelerates the transformation by saving 42.9% of the decoding steps.

We observed that some documents could not be fully transformed by Nougat due to issues with hallucination and repetition. These issues persist with EditTrans which does not control Nougat during the generation phase. LOCR (Sun et al., 2024) addresses these problems by correcting Nougat's output through visual positional guidance, significantly reducing hallucination and repetition errors. Since LOCR complements Nougat's output, it should integrate seamlessly with EditTrans. We are closely monitoring LOCR's development and plan to incorporating it with EditTrans upon its release.

Another observed issue is that Nougat discards figures from pages, while LayoutLMv3 can extract figures. We will further explore how to insert figures properly into Markdown output.

[3] https://math.nist.gov/~BMiller/LaTeXML/
[4] https://info.arxiv.org/help/license/reuse.html#full_text
[5] https://huggingface.co/facebook/nougat-base

7 Limitations

Due to the limitations of LayoutLMv3 (Huang et al., 2022), our method currently limits the output to a maximum of 512 tokens, but we have observed that many pages exceed this token count. Secondly, full-page formulas and tables cannot benefit from our method. Additionally, our method may be less efficient in batch generation due to synchronization.

References

2008–2024. Grobid. https://github.com/kermitt2/grobid.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. 2023. Nougat: Neural optical understanding for academic documents. *Preprint*, arXiv:2308.13418.

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. 2021. Document ai: Benchmarks, models and applications. *Preprint*, arXiv:2111.08609.

Markus Diem, Florian Kleber, and Robert Sablatnig. 2011. Text classification and document layout analysis of paper fragments. In *2011 International Conference on Document Analysis and Recognition*, pages 854–858.

Changxu Duan, Zhiyin Tan, and Sabine Bartsch. 2023. LaTeX rainbow: Universal LaTeX to PDF document semantic & layout annotation framework. In *Proceedings of the Second Workshop on Information Extraction from Scientific Publications*, pages 56–67, Bali, Indonesia. Association for Computational Linguistics.

Charles Frankston, Jonathan Godfrey, Shamsi Brinn, Alison Hofer, and Mark Nazzaro. 2024. Html papers on arxiv – why it is important, and how we made it happen. *Preprint*, arXiv:2402.08954.

Anwen Hu, Haiyang Xu, Jiabo Ye, Ming Yan, Liang Zhang, Bo Zhang, Chen Li, Ji Zhang, Qin Jin, Fei Huang, et al. 2024. mplug-docowl 1.5: Unified structure learning for ocr-free document understanding. *arXiv preprint arXiv:2403.12895*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM International Conference on Multimedia*, MM '22, page 4083–4091, New York, NY, USA. Association for Computing Machinery.

Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. Funsd: A dataset for form understanding in noisy scanned documents. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 2, pages 1–6.

Geewook Kim, Teakgyu Hong, Moonbin Yim, JeongYeon Nam, Jinyoung Park, Jinyeong Yim, Wonseok Hwang, Sangdoo Yun, Dongyoon Han, and Seunghyun Park. 2022. Ocr-free document understanding transformer. In *European Conference on Computer Vision (ECCV)*.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2struct: screenshot parsing as pretraining for visual language understanding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

Kai Li, Curtis Wigington, Chris Tensmeyer, Handong Zhao, Nikolaos Barmpalios, Vlad I. Morariu, Varun Manjunatha, Tong Sun, and Yun Fu. 2020a. Cross-domain document object detection: Benchmark suite and method. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12912–12921.

Minghao Li, Yiheng Xu, Lei Cui, Shaohan Huang, Furu Wei, Zhoujun Li, and Ming Zhou. 2020b. DocBank: A benchmark dataset for document layout analysis. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 949–960, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Tengchao Lv, Yupan Huang, Jingye Chen, Lei Cui, Shuming Ma, Yaoyao Chang, Shaohan Huang, Wenhui Wang, Li Dong, Weiyao Luo, et al. 2023. Kosmos-2.5: A multimodal literate model. *arXiv preprint arXiv:2309.11419*.

Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. EdiT5: Semi-autoregressive text editing with t5 warm-start. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2126–2138, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. FELIX: Flexible text editing through tagging and insertion. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1244–1255, Online. Association for Computational Linguistics.

Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. Text generation with text-editing models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 1–7, Seattle, United States. Association for Computational Linguistics.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Mindee. 2021. doctr: Document text recognition. `https://github.com/mindee/doctr`.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Birgit Pfitzmann, Christoph Auer, Michele Dolfi, Ahmed S. Nassar, and Peter Staar. 2022. Doclaynet: A large human-annotated dataset for document-layout segmentation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3743–3751, New York, NY, USA. Association for Computing Machinery.

Zejiang Shen, Kyle Lo, Lucy Lu Wang, Bailey Kuehl, Daniel S. Weld, and Doug Downey. 2022. VILA: Improving structured content extraction from scientific PDFs using visual layout groups. *Transactions of the Association for Computational Linguistics*, 10:376–392.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. Association for Computational Linguistics.

Yu Sun, Dongzhan Zhou, Chen Lin, Conghui He, Wanli Ouyang, and Han-Sen Zhong. 2024. Locr: Location-guided transformer for optical character recognition. *Preprint*, arXiv:2403.02127.

Yi Tu, Ya Guo, Huan Chen, and Jinyang Tang. 2023. LayoutMask: Enhance text-layout interaction in multi-modal pre-training for document understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15200–15212, Toronto, Canada. Association for Computational Linguistics.

Xingjiao Wu, Ziling Hu, Xiangcheng Du, Jing Yang, and Liang He. 2021. Document layout analysis via dynamic residual feature fusion. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6.

Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, Min Zhang, and Lidong Zhou. 2021. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2579–2591, Online. Association for Computational Linguistics.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 1192–1200, New York, NY, USA. Association for Computing Machinery.

Huichen Yang and William H. Hsu. 2021. Vision-based layout detection from scientific literature using recurrent convolutional neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6455–6462.

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

Wenzhen Zhu, Negin Sokhandan, Guang Yang, Sujitha Martin, and Suchitra Sathyanarayana. 2022. Docbed: A multi-stage ocr solution for documents with complex layouts. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):12643–12649.

A Details for Fine-tuning LayoutLMv3 (Huang et al., 2022)

- Base Model: `layoutlmv3-base`[6]

[6] `https://huggingface.co/microsoft/layoutlmv3-base`

- Batch Size: 64
- Epochs: 10
- Weight Decay: 1×10^{-5}
- Dropout rate: 0.1
- Optimizer: AdamW (Loshchilov and Hutter, 2019)
    - Learning Rate: 2×10^{-5}
    - ϵ: 1×10^{-6}
- LoRA (Hu et al., 2022):
    - Rank: 32
    - α: 64
- All Parameters: 126,512,776
- Trainable Parameters: 1,201,156 (0.94%)
- The LayoutLMv3 model was fine-tuned on an $1\times$A100 cloud server for 9 hours.