Grenze International Journal of Engineering and Technology, Jan Issue

Document Data Extraction with Optical Character Recognition

Vidit Sanghvi[1], Vijay Ukani[2], Raj Patel[3], Priyanka Sharma[4] and Pooja Shah[5]
[1, 2, 5]Computer Science and Engineering Department, Institute of Technology, Nirma University, Ahmedabad, Gujarat, India
Email: 20mced09@nirmauni.ac.in, vijay.ukani@nirmauni.ac.in, pooja.shah@nirmauni.ac.in
[3-4]Department of Machine Learning, Samyak Infotech Private Limited, Ahmedabad, India
Email: patelraj1677@gmail.com, priyanka.sharma@samyak.com

***Abstract*—The modern solutions for fetching content from PDF documents or images includes fetching document's multiple features like processing using Natural Language Processing (NLP), border-less table contents, processing it's stacked lines, fetching document layout information and working on region-based features from documents. There are new challenges like fetching a table's content where the table is included in borders of other content. In this paper, we have used LayoutLM for parsing the content and RetinaNet for table detection in a particular document. The pretrained models were trained on very less data-set to decide for large scale use. We also have discussed architecture of LayoutLM and tesseract that we used in building the tool with discussing output of some other approaches.**

***Index Terms*— Optical Character Recognition, Document Data Extraction, Tesseract, BERT, LayoutLM, Table Detection, Retinanet.**

I. INTRODUCTION

The digital processing of real-world data has made human effort easy for manipulating files and papers over years now. As in such direction, parsing a digital document's information into computer processable format is not easy. Optical character recognition has been on trend in the past 40 years from now, and it has been made easier to perform by various methods available over time. Earliest ideas of OCR were developed in 1870 to 1931 where Fournier d'Albe's Optophone was developed to help the blind people read. David H. Shephard built a device named "Gismo" that converted printed messages into machine language which was the first ever built system based on OCR. To read printed content for blind people, there was an electro-mechanical device named Optacon. To read price tags on bills and passports, scanners are used currently to scan the printed text and convert it into computer readable form. In past 20 years, OCR products are made online free for use like Adobe Acrobat, Web-OCR, Google Drive and there are numerous products available. The importance of the study we have done is to identify the latest state of the art models' working in parsing complex document formats which include multiple number of entities of information included in it, different layouts, numerous formats of documents including tabular data. We test the approaches on the model explored by us on our dataset and discuss its results. To gain some idea for future of our use of model is better or not, the dataset we have used in model contained less images of documents while training on the model.

Grenze ID: 01.GIJET.9.1.572

II. Problem Statement

The goal of the whole project is to create an application that parses the non-digital content to digital, to be better processed by machine ahead on our custom dataset. So, to loosen the overhead of processing content of a document manually. The dataset contains different format of invoices; i.e. by the means of formats, one particular format contains information at fixed positions in each document of that format. It does not change. Different companies have multiple number of formats of documents like this. The data can be present in PDF format, JPEG format or PNG format. Typically, the goal is to register one time for a user of particular company using specific document format, and then use that format to automatically extract the data for other documents having same structure.

A. Dataset

There are two types of document categories which are “Sales” and “Purchase” documents. The “sales” documents need not to be processed further for data extraction as they are already generated by Tally software in which content can be automatically generated in computer readable format. Purchase documents are hand scanned or computer scanned which is not generated by Tally. As shown in Figure 1, these are sample purchase documents formats:
Typically, many companies contain different number of document formats which are either available in PDF or image formats. There can be 100 formats for a single company and in future, there can be a greater number of formats added to same company. As said above, each format has different document structure. Also, the companies can be added in future too.

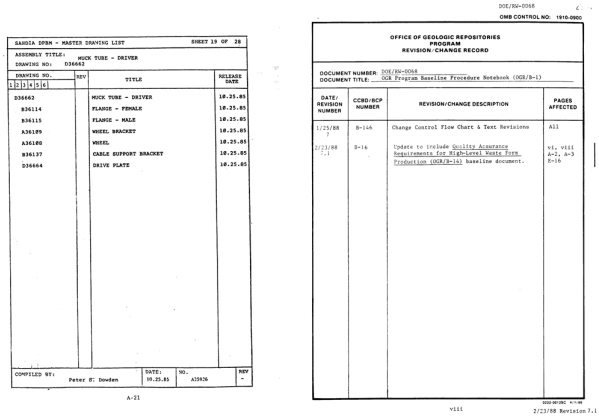

Figure 1: Sample Formats of Documents

III. Literature Review

Optical character recognition comprises of two steps: Character Detection and Character Recognition. Text and words detection is considered as object detection. Transformer [1] has been showing a significant improvement in text recognition using NLP and layout features. In a transformer, for text in image, CNN is used as backbone architecture using self-attention on the top and it’s called encoders. However, we first reviewed typical methods for parsing printed and hand-written characters. We discuss it first and then we refer the models which provide document and layout analysis. In a study of using support vector machine for English characters [2], they took NIST database as their dataset and they used lower case and upper-case characters of English as in training. For feature extraction, they used freeman chain code and support vector system for the recognition step. The project showed 86% accuracy with lower case characters as training data-set, 88% accuracy as uppercase letters used in training, and 73% as the combination of both. In the one with a single layer of artificial neural network on hand written English characters [3], they used row-wise segmentation technique which is efficient in extracting some common features of hand-writings of people. Though this technique cannot be useful when the characters are deviated from their positions. The character N grams as a primitive method for can be used for recognition of heavily degraded documents printed in Indian Languages. The result was tested on Malayalam and English text images which showed considerable improvement for degraded quality of documents which might be useful in our dataset with low quality documents. The error rate was reduced by 15% in word recognition. It showed good results with this approach in comparison to classical methods [4].

Based on a new technique called Spiral Run Length Smearing Algorithm (SRLSA), the authors are managed to extract handwritten document images with above 85% success rate [5].
As we had almost all documents printed with different style of printing characters, for documents having different text styles we referred one source where the authors have taken some text to train the system which basically works on extracting features of text character by character and then determining that character with feature matching. In training, they have taken only 3 types of characters styles and tested it with 5 images which shows accuracy with the same types as training very good and with different types of styles, the accuracy is degraded [6].
The two latest tools available for OCR which are highly portable and free for use that is Tesseract and Transym. It has described how tesseract is overpowered than transym and also tesseract requires very less time (in the matter of seconds) to extract the data [7]. Also, standard deviation of accuracy is more consistent in case of tesseract which is assures us to give more consistently correct extracted data which is in case of transym, it is deviating.
For the use of content fetching, we have documents of specific formats for which we can apply optical character recognition with tesseract and it gives good printed character recognition in all the documents with supporting different styles of English characters while used.
For the hand written characters to be recognised, there are many impressive solutions has been seen like in case of Odia character recognition effort done [8], there is very higher chance of getting true positives about the characters that the model recognises and almost none faults.
In past 20 years, OCR products are made online free for use like Adobe Acrobat, Web-OCR, Google Drive and there are numerous products available. In 2000, Web-OCR was developed in a cloud environment for real time translation of foreign language signs. Tesseract is a free open-source tool which was developed by Hewlett Packard in 2005 which has almost accurate recognition capacity for printed documents. For the first version of our application, the minimum overhead of time is required while processing the content in real time to our clients. After reviewing typical approaches as above, we find to use tesseract for position-based content extraction in documents.
Since we are having issue in fetching tabular contents, it is better to fetch layout and visual information from the given document such as table detection, the relativity of position of question and answer (visual information), text-label prediction image features etc.
The latest approaches which fetches document layout information are explored. The first use of CNN in table detection was applied in 2016 [9]. They combined loose rules to gain table-like areas from portable documents which supported tables with grid, tables with only horizontal lines, and table with no horizontal or vertical lines. The task of classification of chosen areas was implemented using CNN to determine the chosen area is a table or not. However, information contained in portable documents makes significant improvement in accuracy. However, their method has a limitation of detecting extra regions with table that belongs to other objects in documents.
As discussed above, portable document contains metadata within it, by which it becomes easier to detect rows and columns of tables, but it does not work with scanned images. DeepDeSRT [10] achieves state of the art accuracy with a non-table problem which includes detecting bar chart as a table. For the tables in images, it successfully detects table and also identifies its structures. In background, it uses pretrained CNN models (trained for object detection) for table detection and validates it by achieving state of the are performance having 97.40% precision.
For structured document, layout information is so much important for better prediction of labels to the content available in documents. Carlos X. Soto explains how FRCNN is useful for contextual feature extraction on a scientific articles' dataset [11]. FRCNN detect regions of interest and labels them as a table, a set of textual information, figures etc. It shows that adding very simple contextual information improves performance of classification over the baseline by 50% and achieves state of the art performance then single stage detectors.
PubLayNet is a dataset made by Xu Zhong, Jianbin Tang and Antonio Jimeno Yepes in 2019 which was developed for better layout feature extraction for recent convolutional neural models [12]. By training the FRCNN and MRCNN models on the dataset, the models outperforms than pretrained models trained on MS-COCO and ImageNet datasets in all the classes (text, list, figures) except table detection. In table detection, fine-tuning FRCNN on COCO dataset has better precision than fine-tuning it on PubLayNet.
MFCN (Multimodal fully convolutional neural network) provides document's semantic structure recognition and extraction [13]. This model uses both visual and textual information for better prediction unlike the ones we discussed above which provides only document's layout information. To gain textual features, the sentence from the document is embedded and it is mapped to the sentence where it is represented.

The approaches described above has two limitations: One is the models are trained on less human labelled data and second is they either grab layout features or textual features from the documents. So, they are not trained on both types of features jointly.
Compare to this, LayoutLM [14] addresses this issue and achieves even better accuracy in label prediction task using visual and textual features of documents. It takes advantage of multimodal inputs, token and layout embeddings and image embeddings. LayoutLM is trained on IIT-CDIP dataset which contains more than 6 million documents with 11 million scanned images. LayoutLM outperforms all the state of art pretrained models.

IV. Proposed Methods

We describe position-based text extraction using open-source library tesseract first, which is used in our current application; We discuss about its output and drawbacks of the current version of our app. Then we move further to tackle some issues in our current application; we discuss transformer approach, its output on handling issue, its advantages and drawbacks on our dataset and on our application. The results of some other models on our dataset to overcome the current issue are compared.

A. Handling Data with Structure-Position Format

As we have mentioned above, the dataset has different types of structures under different companies in which a particular structure contains same type of information on same position of document as observed by us. We first decided to go for position-based content extraction in which position of all contents of a structured document is known to us by means of registration of that particular structure first and then automatic content extraction on those positions registered with us as shown in Figure 2:

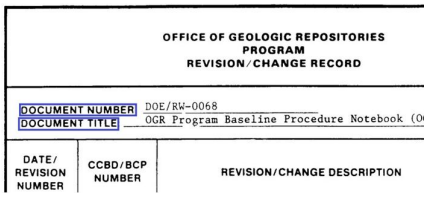

Figure 2: Registration by Snipping Interesting Fields

In Figure 2 i.e., in registration process, user snips interesting fields as shown as blue boxes, and the position of those boxes are stored permanently in JSON format at server. Next, user selects type of documents in second page of application and the content from the same position with respect to registered document is fetched automatically from uploaded documents.

Results And Issues

We used tesseract provided by PYOCR library which is open-source. PYOCR provides functionalities like text extraction with bounding boxes, orientation detection. It extracts all of the contents of documents even with rotated content 99% successfully. The footer lines' characters are small in some documents which it is unable to fetch. After registering for one format of document, contents of any documents having same format is automatically extracted.
The issue with current version of our app is we fail to grab all the contents of any tabular information present in documents as the number of rows in table may be variable in each document having same format. Here, position-based content extraction does not work.

B. Transformer Approach

We explored NLP based approach in which transformer is one state of the art concept that is famous worldwide. We use LayoutLM [14] for label prediction of content of documents. LayoutLM is trained on FUNSD [15] dataset which contains its own labels (question, answer, header and other). For experiments, we have used 10-17 images for training and tested the fine-tuned model on sample images. As, the annotation for one image was so much time-consuming process and there is not entity linked to any other entity in dataset, we still try to get some idea of how would it be with different scenarios in available dataset of 17 images. We have explained basics of bidirectional transformer used in LayoutLM and BERT architecture ahead.

Bidirectional Transformer

Bidirectional transformer treats any sequence of input in an NLP task as a representation caused by having attention according to different positions of that input. This is called masking of language model (MLM) [1].

This bidirectional transformer has encoder-decoder architecture in which an input sequence depicted as symbols (x_1, x_2, ... x_n) is transformed in continuous representations (z_1, z_2, ... z_n) and decoder generates output (y_1, y_2, ... y_n) of each token. Here, in input sequence, previously generated output is also added in next input sequence.
The typical architecture of transformer consists of 6 blocks of layers (one encoder and one decoder per layer). These layers are identical.

BERT Architecture

The architecture of BERT model [16], contains multi-layer bidirectional Transformer encoder as shown in Figure 5. $BERT_{BASE}$ model consist of 12 bidirectional transformer blocks, 768 hidden states and 12 self-attention heads. It is first state of the art model which has the highest accuracy on sentence-level and token-level predictions. BERT model has two steps: pretraining and fine-tuning. In pretraining step, all the parameters are consumed of pretraining weights and in fine-tuning, those parameters are fine-tuned. In pretraining, model is trained on unlabelled data and in fine-tuning, model is fine-tuned on labelled data. Unlike the standard unidirectional models.

- Masked Language Model (MLM): - To train on an input sequence, randomly 15% of total number of available tokens in that sequence are masked, and those tokens are predicted. This procedure is called as Masked Language Modelling (MLM) approach. The final hidden vectors of masked tokens are fed into softmax layer.
- Next Sentence Prediction (NSP): - Language modelling can't capture any relationship between two sentences. NSP is binarization method used for whether two sentences are consecutive or not. In Figure 5, C is used for next sentence prediction. [16]

The main advantage of fine-tuning BERT model is that it performs very well with very less data on downstream tasks with randomly initialized parameters.

LayoutLM

LayoutLM additionally has two more layers in existing $BERT_{BASE}$ architecture [14]. Typically, it improves language representations of documents that are visually rich. LayoutLM uses BERT as backbone of it and adds two new embeddings called image embedding and 2-D position embedding.

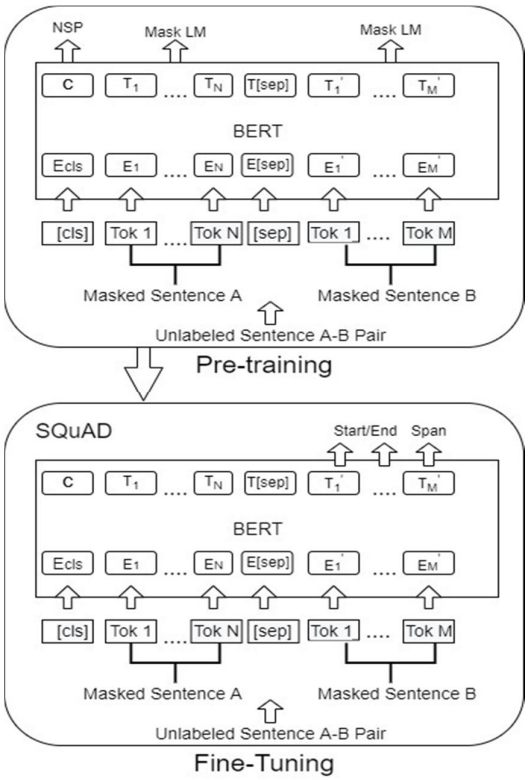

Figure 5: BERT Architecture

- Document's Layout Information - The relative positions of different categories of content (i.e., question and its answer) is taken as a feature in 2-D positions in LayoutLM framework. LayoutLM takes an image of document as a coordinate system of pixels and processes each content information with positions of that content. Any input tokens or word is located by (x_0, y_0, x_1, y_1) positions in which (x_0, y_0) is top-left position of a particular content and (x_1, y_1) is bottom-right. The images according to content present in document is extracted and features are fetched on those images using Faster-RCNN as image embeddings.
- Visual Information - Here, image features are taken such that any segment of document could be identified by those features. The document can represent a layout which can be identified as a feature. Any document might consist of bold, italic, underlined contents which also can be taken as features and may further help in classification of content.
- Pretraining LayoutLM - LayoutLM uses Masked Language Model representation to train on the words which are available in dataset with positions. It masks some of the input tokens randomly and keeps 2-D position embeddings. Model uses contexts to predict these masked tokens. By using contexts and positions of contents it combines linguistic and image features to predict best. For better document-level presentation, model uses Multi-Label Classification Loss (MDC) from which model learns better document representation. LayoutLM is pretrained on form-understanding, document-understanding and receipt understanding tasks.
- Pretraining Dataset - LayoutLM is trained on IIT-CDIP test collection dataset which contains 6 million documents and 11 million scanned document images. The text content is produced by OCR.
- Fine-tuning Dataset - The model is fine-tuned on FUNSD, SROIE and RVL-CDIP datasets. FUNSD dataset contains 199 scanned forms which are annotated with 31k almost words. Also, words are interlinked with each other and having labels as classification properties which contains 4 classes - Header, Question, Answer, Other. It has 149 training and 150 testing samples.
- Model Architecture - LayoutLM's architecture is same as BERT except position embedding layer. In LARGE setting, to sum up, it contains overall 24 layers of transformer with 1,024 hidden size and 16 attention heads initialized by $BERT_{LARGE}$'s 343M parameters. The model was trained having Adam optimizer and learning rate of $5e^{-5}$.

Results

The model was trained on 50, 100 and 500 epochs with learning rate of $5e^{-5}$, $5e^{-10}$ in each experiment but as dataset was too less, there is not much difference in precision including different epochs. However, less learning rate yields better result. The result is shown in table 1.

- Labelling All Entities from Document: In this experiment, we have annotated all the phrases (like "Roll no.", "Phone Number") and its sub-words (like "Roll" and "No." in "Roll No.") giving unique labels individually to each type of information and for all contents present in one document. Here, each information of document is given "entityname_q" and "entityname_a" labels like a question-answer type pair. Overall number of labels are approximately 25 for a document type.
  "Buyer's address" is a label predicted in some of fields which is correct prediction. Prediction for "GSTIN" and "PAN" fields is not correct. Although, for tabular contents, index numbers are labelled correctly as "index a" and some of the tabular contents were predicted almost accurate deviating by a confusion of model between two labels in which name of labels contained same word in them. In some cases, it lacked by not be able even predicting some of the content of tables.
  This approach did not work for us as the output on each content in document contained multiple labels prediction for a particular content often, though some content was rightly predicted. But again, giving labels this way won't work for us as we have encountered this later on, there can be new labels ahead in future in new document types for which we would have to retrain the model each time new format is encountered and it is costly.
- Giving Few Labels with No Sub-labels - We took this approach to extract interesting regions only those customers are interested in for fetching content of those regions. Even on less data that model was trained on, result is better with a smaller number of labels but not accurate with all the contents of documents tested on often. However, we have excluded sub-words of an entity such as in "Invoice No.", whole "Invoice No." and its actual value is labelled as "Invoice No.". As this, there are "invoice date", "buyer's name", "buyer's gstin number", "buyer's address" and "buyer details" labels. Buyer details contains all details including buyer's address and gstin number.

- Giving Few Labels with Sub-labels - This approach worked better than the previous approach. In this, an entity name (i.e., Invoice Number etc.) and its value were given same label separately. As "buyer" keyword appears in lot of document formats, we labelled it as "Buyer". Buyer's address, buyer's state, index numbers of tables (given "row" as a label) and heading of those number (labelled as "index"), additionally we labelled invoice number, buyer's gstin and invoice date.

Here is the accuracy we achieved every-time we trained as discussed above:

TABLE I: RESULTS OF LAYOUT LM

Annotation Strategy	mAP	GPU
Labelling All Entities	36.8	Google colab
Multi labels With No Sub-labels (7 classes)	50	Google colab
Multi Labels With Sub-labels	60	Google colab

Few things to be noted after model's evaluation, are:

1. For the classification of larger number of labels, model requires more data.
2. The main advantage of using LayoutLM is it is NLP based approach on large sequence of data. Due to we didn't give any links between question answer (field name and its value) pair, the model lacks its accuracy in predicting labels on each field.

Because we encountered that there can be more type of contents that may present in a new document format, we couldn't find a way to fetch all the contents in new types as our main target was that. For that, model needs more training each time a new document format is arrived and we have to add additional code for mapping new fields' labels given by user in application to the ones which our model refers to as. As the problem is fetching tabular content, we further discuss about some models which were built for table detection.

C. Table Detection

As we mentioned above, for the document types as shown in Figure 1, due to the document format having borders around it's all of the content including table, the most of solutions we explored detects table in that format incorrectly.

We applied camelot open-source library on non-scanned images, tabula library, Cascadetabnet[17] and MultiType-TD-TSR[18] models for the detection of tables in above format and all failed in successfully detecting table at mid-position that we required for fetching contents.

However, we fine-tuned Retinanet [19] pretrained on MS-COCO dataset [20] on 29 images and evaluation on 6 images in 1st experiment. We experimented on custom made dataset that is made by us each time having a greater number of images including in dataset. 1st dataset included 29 images of the dataset that client provided to us. After training for 10 epochs having batch-size of 2, the model successfully detects tables in document formats having borders around it's all content with less-precision (as it is very less number of images trained on).

In 2nd experiment, we trained the model on 364 images having border-less tables and different formats outside of client's dataset. The model had 54.3 mean average precision after training of 3 epochs on google colab having classification loss under 20%. The model was not able to detect border-less tables in some images and in some images it detected correctly. With tables having borders, the model most of the times detects table correctly as shown in Figure 6. The results are included in table II.

TABLE II: RESULTS OF TABLE DETECTION USING RETINANET PRETRAINED ON MS-COCO

Train Size (Number of images)	Test size (Number of images)	Epochs	mAP
29(Custom Dataset)	50	10	34.5
364(Custom Dataset + Table Bank)	50	3	54.3

16.31 The data table below is typical of standard tables of thermophysical properties of liquids and gases used widely in chemical engineering practice, especially in process simulation. This specific data set shows the temperature dependence of the heat capacity C_p of methylcyclohexane.

Table 0.000

Temperature Kelvin	Heat Capacity $C_p, KJ/kg-K$	Temperature Kelvin	Heat Capacity $C_p, KJ/kg-K$
150	1.426	230	1.627
160	1.447	240	1.661
170	1.469	250	1.696
180	1.492	260	1.732
190	1.516	270	1.770
200	1.541	280	1.801
210	1.567	290	1.848
220	1.596	300	1.888

First fit a linear model

$$C_p = \theta_0 + \theta_1 T$$

and check the significance of the parameters, the residuals, and the R^2 and R^2_{adj} values. Then fit the quadratic model,

$$C_p = \theta_0 + \theta_1 T + \theta_2 T^2$$

Figure 6: Table Detection After 1st experiment

V. Conclusion and Future Scope

We understood tesseract's working in our application and discussed its results on our dataset. The application had tabular content extraction issues and also content displacement issue in some new documents of same structure. We saw that for multi-class classification in NLP, LayoutLM does not give bad prediction in classifying data even with very less dataset for a greater number of multi-label classes. As it can work upon long sequences of data, we are targeting for better in training on a huge amount of data. We explained basics of bidirectional transformer encoder and decoder structure and basics of BERT. After training Retinanet [19] on our dataset, it provides good accuracy with successful table detection on bordered images. However, sometimes it lacks detecting bordered less tables which has no column/row lines. As far as, dataset is in concerned, the reason it lacks in accuracy is that it's trained on less data.

For now, as LayoutLM's results are explored as above with good amount of accuracy on less data and as we found it is also trained on RVL-CDIP dataset which has more than 40 number of multi-label classes, we are going to finetune LayoutLM on approximately 500-1000 documents (depending on number of multi-label classes) each would be having different formats than each other. For tabular data extraction, we will have to train Retinanet on larger dataset (planning to include approximately 1900 images in training dataset) for table detection and handle the content of it separately using open-cv.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[2] D. Nasien, H. Haron, and S. Yuhaniz, "Support vector machine (svm) for english handwritten character recognition," *Computer Engineering and Applications, International Conference on*, vol. 1, pp. 249–252, 01 2010.

[3] R. K. Mandal and N. R. Manna, "Hand written english character recognition using row- wise segmentation technique (rst)," *IJCA Proceedings on International Symposium on Devices MEMS, Intelligent Systems Communication (ISDMISC)*, no. 2, pp. 5–9, 2011, full text available.

[4] S. Dutta, N. Sankaran, K. Sankar, and C. Jawahar, "Robust recognition of degraded documents using character n-grams," *Proceedings - 10th IAPR International Workshop on Document Analysis Systems, DAS 2012*, 03 2012.

[5] S. Malakar, S. Halder, R. Sarkar, N. Das, S. Basu, and M. Nasipuri, "Text line extraction from handwritten document pages using spiral run length smearing algorithm," in *2012 International Conference on Communications, Devices and Intelligent Systems (CODIS)*, 2012, pp. 616–619.

[6] R. Jana, A. Chowdhury, and S. Islam, "Optical character recognition from text image," *International Journal of Computer Applications Technology and Research*, vol. 3, pp. 240–244, 04 2014.

[7] C. Patel, A. Patel, and D. Patel, "Optical character recognition by open source ocr tool tesseract: A case study," *International Journal of Computer Applications*, vol. 55, pp. 50–56, 10 2012.

[8] R. Sethi and K. Mohanty, "Optical odia character classification using cnn and transfer learning: A deep learning approach," vol. 07, pp. 3885–3890, 07 2020.
[9] L. Hao, L. Gao, X. Yi, and Z. Tang, "A table detection method for pdf documents based on convolutional neural networks," in *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, 2016, pp. 287–292.
[10] S. Schreiber, S. Agne, I. Wolf, A. R. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, pp. 1162–1167, 2017.
[11] C. Soto and S. Yoo, "Visual detection with context for document layout analysis," in *EMNLP*, 2019.
[12] X. Zhong, J. Tang, and A. Jimeno-Yepes, "Publaynet: Largest dataset ever for document layout analysis," *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1015–1022, 2019.
[13] X. Yang, E. Yumer, P. Asente, M. Kraley, D. Kifer, and C. L. Giles, "Learning to extract semantic structure from documents using multimodal fully convolutional neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4342–4351.
[14] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, "LayoutLM: Pre-training of text and layout for document image understanding," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery &amp Data Mining*. ACM, aug 2020. [Online]. Available: https://doi.org/10.1145%2F3394486.3403172
[15] G. Jaume, H. Kemal Ekenel, and J.-P. Thiran, "Funsd: A dataset for form understanding in noisy scanned documents," in *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, vol. 2, 2019, pp. 1–6.
[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.
[17] D. Prasad, A. Gadpal, K. Kapadni, M. Visave, and K. A. Sultanpure, "Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2439–2447, 2020.
[18] P. Fischer, A. Smajic, G. Abrami, and A. Mehler, "Multi-type-td-tsr – extracting tables from document images using a multi-stage pipeline for table detection and table structure recognition: From ocr to structured table representations," in *KI 2021: Advances in Artificial Intelligence*, S. Edelkamp, R. Moller, and E. Rueckert, Eds.¨ Cham: Springer International Publishing, 2021, pp. 95–108.
[19] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection,"´ *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.
[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in context,"´ in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 740–755.