Tab this Folder of Documents: Page Stream Segmentation of Business Documents

Thisanaporn Mungmeeprued
University of Cambridge
Cambridge, United Kingdom
tm714@cam.ac.uk

Yuxin Ma
University of Cambridge
Cambridge, United Kingdom
ym371@cam.ac.uk

Nisarg Mehta
Vector.ai
London, United Kingdom
nisarg@vector.ai

Aldo Lipani
University College London & Vector.ai
London, United Kingdom
aldo.lipani@ucl.ac.uk

ABSTRACT

In the midst of digital transformation, automatically understanding the structure and composition of scanned documents is important in order to allow correct indexing, archiving, and processing. In many organizations, different types of documents are usually scanned together in folders, so it is essential to automate the task of segmenting the folders into documents which then proceed to further analysis tailored to specific document types. This task is known as Page Stream Segmentation (PSS). In this paper, we propose a deep learning solution to solve the task of determining whether or not a page is a breaking-point given a sequence of scanned pages (a folder) as input. We also provide a dataset called TABME (TAB this folder of docuMEnts) generated specifically for this task. Our proposed architecture combines LayoutLM and ResNet to exploit both textual and visual features of the document pages and achieves an F1 score of 0.953. The dataset and code used to run the experiments in this paper are available at the following web link: https://github.com/aldolipani/TABME.

CCS CONCEPTS

• **Computing methodologies** → **Information extraction**; **Computer vision problems**; • **Information systems** → *Digital libraries and archives*; *Document structure*.

KEYWORDS

page-level classification, folder segmentation, deep learning

ACM Reference Format:
Thisanaporn Mungmeeprued, Yuxin Ma, Nisarg Mehta, and Aldo Lipani. 2022. Tab this Folder of Documents: Page Stream Segmentation of Business Documents. In *DocEng '22: ACM Symposium on Document Engineering, September 20–23, 2022, San Jose, CA, USA.* ACM, New York, NY, USA, 10 pages.

DocEng '22, September 20–23, 2022, San Jose, CA, USA

1 INTRODUCTION

Document Intelligence (DI) is the area of scientific research that leverages Machine Learning (ML), Natural Language Processing (NLP), and Computer Vision to extract critical insights from a wide range of business documents (e.g., contracts, emails, purchase orders, sales agreements, financial statements, etc.) to enhance the efficiency of companies' business processes. Business documents are usually natively digital or scanned forms, containing a variety of structures such as straight texts, images, tables, and multi-column formats. Possible format inconsistencies, complicated document structures, and poor scan quality make understanding these documents challenging.

Recently, we have seen many efforts utilizing ML algorithms to understand and interpret documents. Popular research topics in DI for visually rich business documents include classification, information extraction, table detection, semantic structure extraction, etc. A common theme across these topics is that they assume a document as a unit. In the real world, however, business documents usually come in folders, with different types of documents mixed in a single PDF file. This might come from two very common scenarios: 1) In many legal and business situations, all documents pertaining to a single case are filed together, and; 2) many companies resort to bulk scanning services in the process of digitizing documents. As a result, we introduce an automation task to segment the folder into documents based on the page content. We will refer to this task as Page Stream Segmentation (PSS). The PSS is essential in obtaining documents from a folder that will proceed to further analysis tailored to specific document types.

The output of the PSS task is a binary vector with a size equal to the number of pages in the input folder, whose values indicate whether or not a page is a breaking-point, i.e., the beginning of a new document. In Figure 1, we illustrate an example of a PSS task. The PSS can be approached using rule-based systems, conventional ML, or deep learning (DL). The interpretation of the page contents can be based on visual or textual features. For the textual-based interpretation, scanned documents must be preprocessed by an Optical Character Recognition (OCR) before inputting into the system.

We approach the PSS task using visual-based and textual-based DL because 1) it can be applied to heterogeneous datasets containing large variations of document formats and lengths, and the types of documents are not known beforehand, and; 2) recent works on

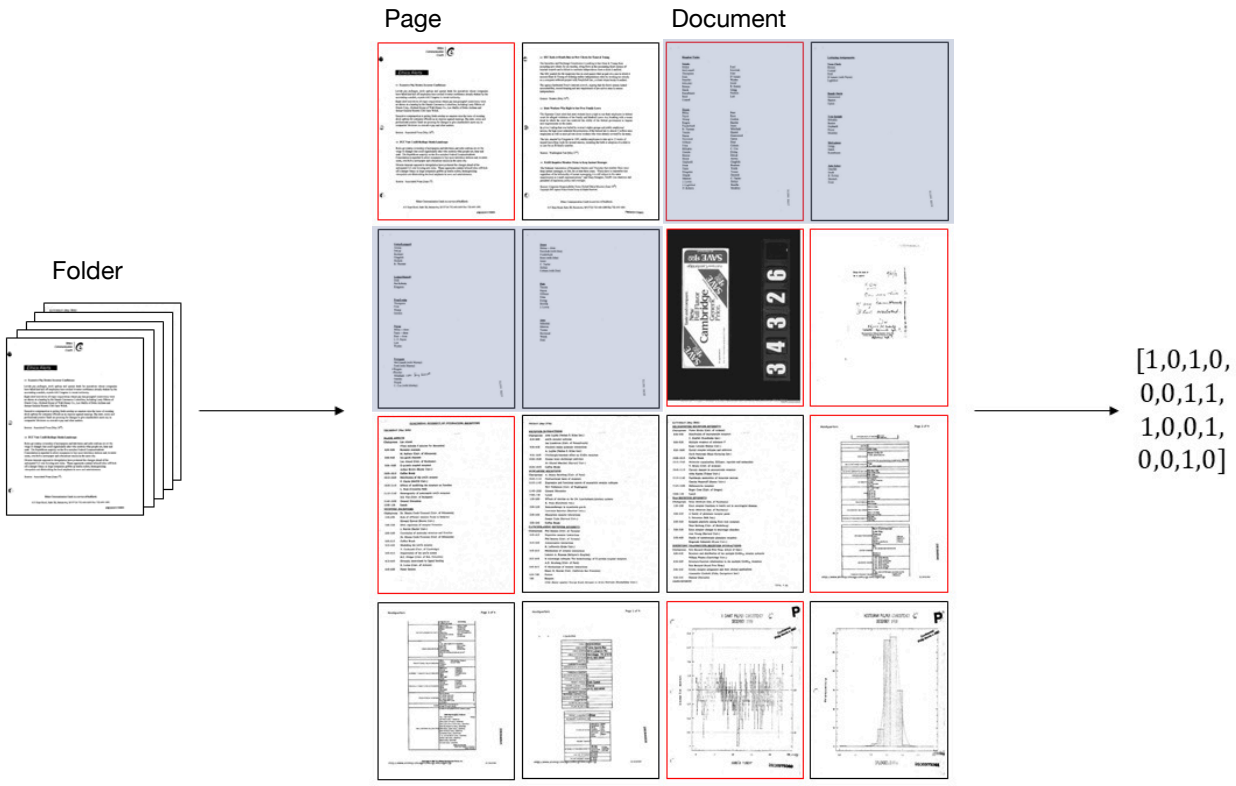


Figure 1: An example of the Page Stream Segmentation (PSS) task: Given as input a 16-page folder, the target output is a 16-digit binary sequence, where 1 indicates a breaking-point and 0 indicates the continuation of the previous document. Red outlines highlight the breaking-point pages. The pages highlighted in blue indicate an example of a document.

similar tasks have shown that both visual-based and textual-based DL models can achieve state-of-the-art performance. Wiedemann and Heyer [14] and Guha et al. [5] introduced similar multi-modal DL approaches. Wiedemann and Heyer employed a GRU-based model as a text feature extractor, while Guha et al. employed a Legal-BERT model. In our work, we chose LayoutLM [15] to extract the textual features, since it takes both texts and layouts into consideration, as opposed to most language models in which the information about layouts is neglected. For the extraction of visual features, the two previous works used the VGG-16 model, while we chose to use the ResNet model [9] since it achieved the highest accuracy on a similar dataset (Tobacco-3432) but for a different task: document image classification [1].

We also found that, while several public document datasets are available, most of them consist of single-page documents that are unsuitable for the PSS task. To facilitate future studies, the dataset and code used in this paper are available at the following web link[1].

In this paper, we make the following contributions:

(1) we present a generated dataset for the PSS task, named TABME, which stands for TAB this folder of docuMEnts;
(2) we propose a deep learning solution: a model architecture that is based on LayoutLM and ResNet, which are a pre-trained BERT-based document understanding model and a pretrained image classification model, and;
(3) we perform an ablation study and an analysis of several potential model biases. With the former, we demonstrate the importance of both kinds of contextual information exploited by the two models, visual and textual. With the latter, we show that the breaking-points positions in the folders do not influence the quality of the predictions.

The remainder of this paper is structured as follows. In Section 2, we present the related work. In Section 3, we introduce and formalize the page stream segmentation task and present a new user-centric evaluation measure. In Section 4, we propose a model to solve this task. Experiments and results are presented in Section 5. We conclude in Section 6.

[1]https://github.com/aldolipani/TABME

2 RELATED WORK

Several works related to the PSS task exist, which can be equivalently referred to as Document Flow Segmentation, Document Stream Segmentation, or Document Separation.

A DL approach that utilizes both textual and visual features was first implemented by Wiedemann and Heyer [14] who combined textual features from the GRU-based model and visual features from the VGG-16 model in an ensemble which outputs the prediction by comparing the previous page with the current page. The proposed model achieved an accuracy of 88.9% on the proprietary German archive dataset and 91.9% on the public U.S. tobacco companies (Tobacco800) dataset. This is the only work where the authors tested their model on a publicly available dataset, and we will use it as a baseline.

Guha et al. [5] improved upon the previous work by replacing the textual feature extractor with the pretrained Legal-BERT model. An F1 score of 97% was reported on the proprietary title insurance dataset. Although the authors claim that this work is an improvement over the previous one, we were not able to test this model against ours due to the unavailability of their source code and the tested dataset. Moreover, this model was optimized only for legal documents, while our dataset is more general.

There are also approaches to the PSS task that use more conventional machine learning approaches. Textual-based approaches include Daher and Belaïd [3] in which various conventional ML classifiers were used (including voted perceptron, which achieved the best performance) to segment pages based on textual features of the current and the previous page obtained from regular expressions. A similar feature extraction method was chosen by Daher et al. [4], but an incremental classifier was used instead. Hamdi et al. [7] extracted textual features using Doc2Vec and made predictions based on a similarity threshold. A follow-up work by the same authors, Hamdi et al. [6], used a purely rule-based approach. The work performed by Karpinski and Belaïd [10] also features a similar approach. Visual-based approaches have been proposed by Agin et al. [2] in which Bag of Visual Words are used as a visual-based feature extractor and Random Decision Forest as the classifier. All of these approaches have been outperformed by the deep learning approach proposed by Wiedemann and Heyer [14]. For this reason, we will not compare against them.

We observe that in most works, the prediction is performed by comparing the current page's content with the previous page. In our work, the classifier consists of a series of 1D convolutional layers that takes into account the information of the previous, current, and subsequent pages, while the information of other pages can also flow to the last layer.

3 PAGE STREAM SEGMENTATION

3.1 Task Description

Here, we formalize the *page stream segmentation task (PSS)* as follows. Let $\mathbf{x} = [x_1, \ldots, x_i, \ldots, x_N]$ with $x_i \in \mathbb{R}^{h \times w}$ and $1 \leq i \leq N$ be a sequence of images representing our input, that is the folder of papers we wish to segment, consisting of N pages where each page is of dimension $h \times w$ pixels. The target output is a binary vector $\mathbf{y} = [y_1, \ldots, y_i, \ldots, y_N]$ with $y_i \in \{0, 1\}$ and $1 \leq i \leq N$, where:

$$y_i = \begin{cases} 1, & \text{if page } x_i \text{ is a breaking-point} \\ 0, & \text{otherwise} \end{cases}, \tag{1}$$

where by breaking-point we intend the first page of a document. Based on these breaking-points we can then reconstruct the structure of the documents forming the folder. However, note that by doing this, we are making the following two assumptions: 1) document pages are input in sequence, that is, all the pages belonging to the same document are input one after the other, and; 2) the pages within each document are input in the correct order. These two assumptions allow us to reduce the more general problem of sorting out a folder of papers into documents to the here-defined PSS task.

3.2 The TABME Dataset

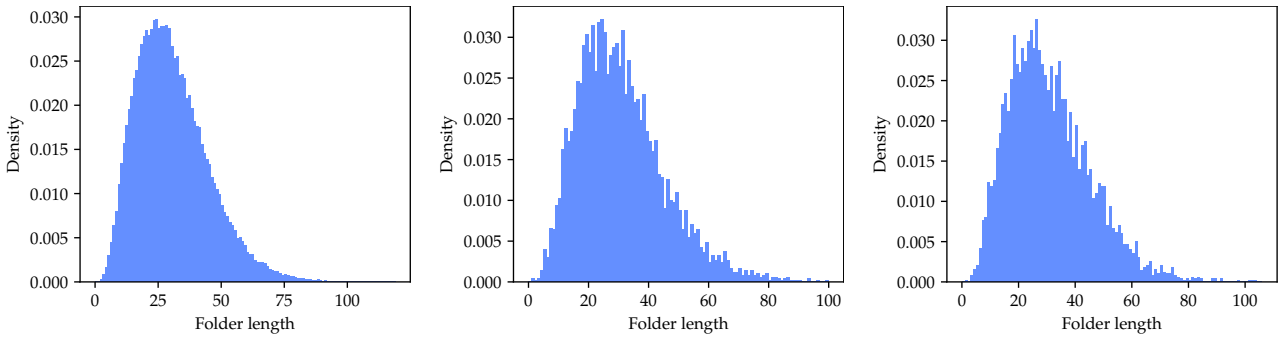
A commonly used dataset for the understanding and classification of business documents is the IIT-CDIP test collection [11]. From this test collection also other datasets have been derived like the RVL-CDIP dataset [8] that was specifically created for a document type classification task. The IIT-CDIP test collection was built by collecting documents from the tobacco litigation. However, this dataset is unsuitable for our purposes since it only contains images and there is no information about to which document these images belong.

For the creation of our dataset, we followed a similar approach. We downloaded a portion of the documents publicly hosted by the University of California, San Francisco (UCSF) library[2]. This collection consists of a sample of 44,769 PDF documents from the Truth Tobacco Industry Documents (TTID) archive. The TTID archive includes several document types (leaflet, letter, email, etc.) and each document can range from one to a hundred pages. In the sampled documents, we made sure not to include corrupted files (i.e., files that could not be opened without an error) or documents longer than 20 pages. Based on this sample, we aim to build a synthetic dataset for our task.

To generate an unbiased dataset we had to perform two preprocessing steps. The first step consisted in cropping the bottom margin of all pages. This was needed because each PDF file contained an identifier of the document printed in them. The second step consisted in transforming all PDFs to gray-scale and resizing them to fit within a square with a side length of 1,000 pixels. This was needed in order to avoid inconsistencies among the documents. Next, to avoid any data leakage among the training, validation, and test sets, we performed the split at this level: We randomly sampled 90% of the documents for the training set (40,292 documents; 110,132 pages), 5% for the validation set (2,238 documents; 6,089 pages), and 5% (2,238 documents; 6,237 pages) for the test set.

We now simulate the digitization of folders of documents following the procedure described in Algorithm 1. Using the sampled documents we simulate the length and variety of document types we find in real-world scenarios. Given a sample of documents $\mathcal{D}$ and the number of folders we wish to generate M, we first sample an integer L following a Poisson distribution. We choose the Poisson distribution because it is often used to model the number of

[2] https://www.industrydocuments.ucsf.edu/



(a) Training set. **(b) Validation set.** **(c) Test set.**

Figure 2: Distribution of folder lengths in the TABME dataset.

Algorithm 1: Generation of the folders of documents, where $\sim$ and $||$ are the sampling and concatenation operators.

Data: $\mathcal{D}$ (set of documents), λ (parameter of the Poisson distribution), M (number of folders to generate).
Result: $\mathcal{F}$ (set of folders of documents).
$\mathcal{F} \leftarrow \emptyset$;
foreach $n \in [1, \ldots, M]$ **do**
  $L \sim Poisson(\lambda)$;
  $f \leftarrow []$;
  foreach $i \in [1, \ldots, L]$ **do**
    $d \sim \mathcal{D} \setminus \{d' \in f\}$;
    $f \leftarrow f \,||\, d$;
  end
  $\mathcal{F} \leftarrow \mathcal{F} \cup \{f\}$
end

Table 1: Summary of the TABME dataset.

Split	Number of folders	Number of pages in each folder			
		Mean	SD	Min	Max
Training	100,000	30.13	14.30	0	120
Validation	5,000	30.36	14.36	1	101
Test	5,000	30.43	14.41	1	107

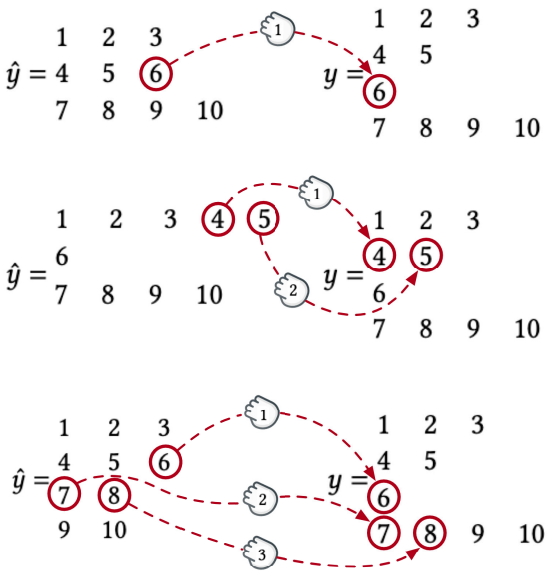


Figure 3: Examples of computation of the MNDD required to change the predictions (on the left-hand side) into the ground-truth (on the right-hand side).

events happening in a given interval of time. In this case, we use it to model the number of documents in a folder. We set the parameter of the Poisson distribution $\lambda = 11$ based on the observation we have in our applications. The combination of this parameter choice and the sampled documents generates folders with a mean length of around 30 pages. Then, we sample without repetition L documents and concatenate them to generate a folder. Finally, we add this folder to the set of generated folders and return this set ($\mathcal{F}$). We perform this process for each split and generate 100,000 folders for the training set, 5,000 folders for the validation set, and 5,000 folders for the test set. The summary of the TABME dataset is given in Table 1. In Figure 2, we show the distribution of folder lengths in the training, validation, and test sets.

For the sake of reproducibility, we also release the further preprocessing done to get the OCR results. These results are obtained using the Tesseract OCR engine. This engine was configured to assume English as the language to be recognized and to output tab-separated values files containing the recognized words and their associated box coordinates and confidence levels.

3.3 Evaluation Measures

To evaluate and compare the performance of the ML algorithms applied to this task we use standard evaluation measures like Precision, Recall, and F1-score. We can use these evaluation measures since this task is a sequential binary classification problem whose focus is on the identification of the breaking-points. Moreover, given that the number of breaking-points is much lower than the number of non-breaking-points (35% vs 65%) these evaluation measures are preferred over measures like accuracy that weigh the identification of both classes equally. However, these metrics are system-centric evaluation measures and may not reflect the actual satisfaction of a user using a system. This is because not every misclassification requires users the same amount of work. For this reason, we formulate a new user-centric evaluation measure inspired by a graphical user interface used in our applications. This measure aims to count the Minimum Number of Drag-and-Drops (MNDD) a user would need to perform should the pages of the identified documents are not separated correctly.

In Figure 3, we illustrate with three examples what the MNDD computes. We have, on the left-hand side, the model predictions ($\hat{y}$) and, on the right-hand side, the ground-truth (y). Each row is a document, and each value represents the number of pages in the folder. In the first example, we see that page 6 was identified as the last page of the second document, while it should have been identified as a new one-page document. To remedy this error, the user would need to perform only one drag-and-drop action. In the second example, we see that pages 4 and 5 were identified as the continuation of the first document, while they should have been identified as a new document. To remedy this error, the user would need to perform at least two drag-and-drop actions. Following this line of reasoning, the reader can infer what is happening in the third example where at least 3 drag-and-drop actions are required.

Based only on the first two examples we can see that the misclassification of 1 breaking-point has caused the user to perform a different amount of work, which is reflected in this evaluation measure and not in the previous ones. Moreover, note that the illustrated solutions are not unique. There are other ways of rearranging the pages of the folder into the correct order. However, here we care about the minimum number of drag-and-drops required to do so and not about the actual drag-and-drops actions to be performed. Also, the order of the documents (rows) does not matter. The proposed measure is in fact invariant to the order of the documents in the folder.

4 THE PROPOSED MODEL

The model architecture (illustrated in Figure 4) relies on both Computer Vision (CV) and Natural Language Processing (NLP) approaches.

The input folder containing N images of size $h \times w$ is represented by:

$$\mathbf{x} = [\mathbf{x_1}, \ldots, \mathbf{x_N}] \in \mathbb{R}^{N \times h \times w}. \tag{2}$$

Each image x_i is fed into ResNet and LayoutLM. We chose these pre-trained models since they have achieved state-of-the-art performance on other tasks. For LayoutLM, we first need to extract the words and the bounding boxes using the Tesseract OCR engine.

$$\mathbf{h_r} = \text{ResNet}(\mathbf{x}), \tag{3}$$

$$\mathbf{h_l} = \text{LayoutLM}(\text{OCR}(\mathbf{x})). \tag{4}$$

where $h_r \in \mathbb{R}^{N \times 512}$ and $h_l \in \mathbb{R}^{N \times 768}$ are feature vectors extracted by the two pretrained models respectively.

The outputs $\mathbf{h_r}$ and $\mathbf{h_l}$, representing the visual and textual embeddings of the document pages, are then concatenated:

$$\mathbf{h} = [\mathbf{h_r}, \mathbf{h_l}] \in \mathbb{R}^{N \times 1280}. \tag{5}$$

This concatenated vector is then fed into a stack of six 1D convolutional layers with a kernel size of 3 and a stride of 1. The convolution operation is done along the series of pages. Therefore, setting the kernel size to 3 means that each layer can see the features of the previous, current, and subsequent page. This allows the model to learn the similarities and differences of the pages in comparison with the adjacent pages.

These convolutional layers follow an encoder structure where the dimension of the feature vector decreases as it is fed into the next layer. The output vectors of the six convolutional layers are of size $N \times 1067$, $N \times 854$, $N \times 641$, $N \times 428$, $N \times 215$, and finally $N \times 2$. The number of convolutional layers and the size of their output vectors are hyper-parameters of this architecture. The dimension along the length of the documents is preserved by circular padding. A rectified linear unit (ReLU) and dropout layer with a probability of 0.2 are added after each convolutional layer. Formally we have:

$$\mathbf{c} = \text{CNN}(\mathbf{h}), \tag{6}$$

where $\mathbf{c} \in \mathbb{R}^{N \times 2}$.

For each page, the two output classes indicate whether or not the page is the breaking-point. Because of the flexibility of the convolutional layers, the input folder can have any number of pages. In the case of folders with many pages where the input cannot fit into the GPU memory, the pages can be separated and input into the model using a sliding window. Formally we have:

$$\mathbf{s} = \text{Softmax}(\mathbf{c}) \tag{7}$$

$$\hat{\mathbf{y}} = \left[\underset{b \in \{0,1\}}{\arg\max}(s_{i,b+1}) : 1 \leq i \leq N \right] \tag{8}$$

where $\mathbf{s} \in \mathbb{R}^{N \times 2}$ and $\hat{\mathbf{y}} \in \mathbb{R}^{N}$ are the logits matrix and the predicted vector.

This architecture is trained using an unweighted binary cross entropy loss function:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log P(\hat{y}_i = 1) + (1 - y_i) \log P(\hat{y}_i = 0)] \tag{9}$$

where $\mathbf{y} \in \mathbb{R}^{N}$ is the ground-truth label.

All layers including those in the pretrained models are unfrozen and trained for a maximum of 30 epochs using an early stopping strategy to avoid overfitting. We use Adam as the optimizer. The stopping strategy is set to stop training after the validation loss has not improved for at least 5 epochs. The learning rate is set to $5 \cdot 10^{-5}$. The model takes around 40 minutes to finish one epoch when trained on an Nvidia GeForce RTX 3090 GPU. On average, the training terminates after around 20 epochs. The rest of the training and validation details are provided in the Appendix.

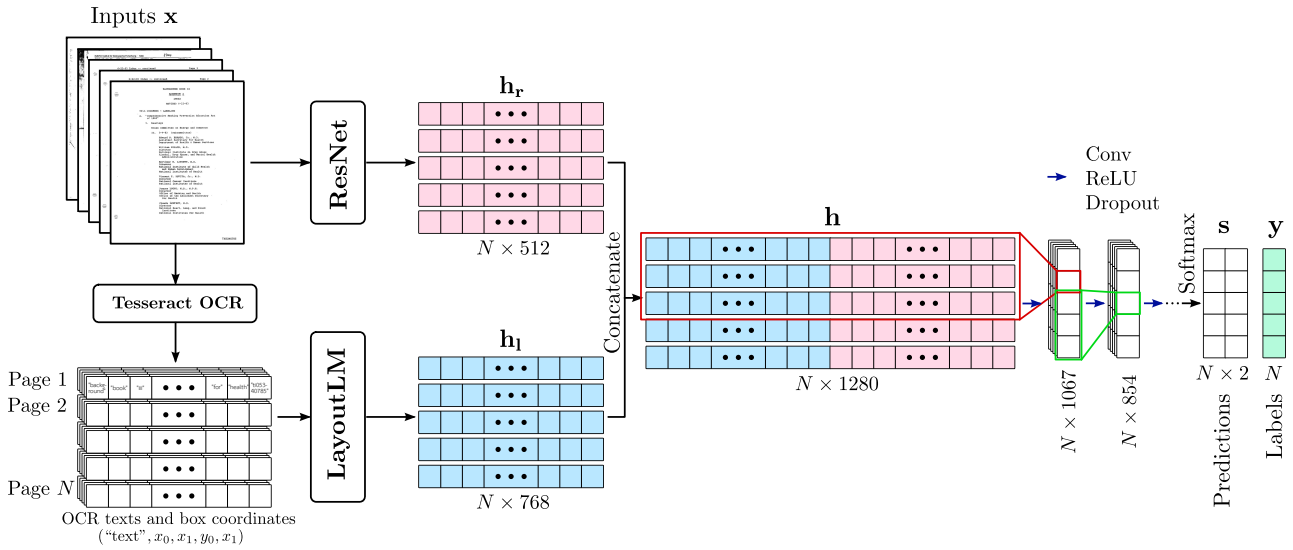


Figure 4: The model architecture consisting of ResNet and LayoutLM model followed by 1D convolutional layers.

We hypothesized that the combination of the two approaches results in an improvement in the performance of the model. Relying on visual inputs alone might not be sufficient because most documents are in black and white and contain visually similar layouts. The same applies to the OCR inputs, where the quality of OCR results is sometimes poor and the OCR fails on handwritten texts and texts with uncommon fonts. Thus, the combination of both inputs gives the model a better view of the documents and a better chance to make the correct prediction. This hypothesis is tested later with an ablation experiment.

5 EXPERIMENTAL SETUP

In this section we aim to answer the following 3 research questions:

RQ1 How does the proposed model perform on the TABME task?

The model was compared against two trivial baselines on the proposed dataset that we generated containing 6,237 images from 2,238 documents. The baselines consist of a random classifier that predicts a breaking-point 35% of the times (named *random*) and a model that predicts as a breaking-point only the first page of the folder (named *only first page*). To evaluate these models, we will use the dataset and evaluation measures introduced in Section 3.

We also compare the proposed model against the only public baseline we found provided by Wiedemann and Heyer [14] on the Tobacco800 dataset [12]. We test the model on their test set which was sampled from the last 260 images from 150 documents according to the alphabetical order of the filenames. Although this dataset and ours are generated separately, they are from the same original source. To ensure the absence of any data leak between this test set and our training and validation sets, we compared the document IDs and did not find any shared document. To ensure a fair comparison we compared against this model using the same metric recommended by the authors, i.e., accuracy.

Table 2: Performance comparison of the proposed model against the baselines. For MNDD the lower the better, for Precision, Recall and F1, the higher the better.

Model	MNDD	Precision	Recall	F1
Our model	**3.440**	**0.947**	**0.964**	**0.953**
Random	15.256	0.415	0.498	0.432
Only first page	20.548	1.000	0.102	0.182

RQ2 What is the contribution of the ResNet and LayoutLM components?

To assess the contribution of the components of the combination of ResNet and LayoutLM, we perform an ablation study. This ablation study consists of evaluating the performance of each model independently and comparing their performance to the combined model. When LayoutLM is ablated, we replace the OCR input with a constant embedding equivalent to the case where there is no OCR box detected; When the ResNet is ablated, we replace the image embeddings with an embedding of a white blank page.

RQ3 Are there any biases in terms of the position of the breaking-point in the folder, the length of the folder, or the length of the documents in the folder?

To assess the presence of any of such biases, we will assess the model in function of the position of the breaking-points in the folder, the length of the folder, and the length of the documents.

5.1 TABME Task (RQ1)

In Table 2, we show the performance of our model against the naive baselines on the proposed dataset. We observe that the proposed model is able to achieve performances way above the naive baselines. This indicates that the model is not behaving in a naive way. The proposed model is able to classify correctly the breaking-points

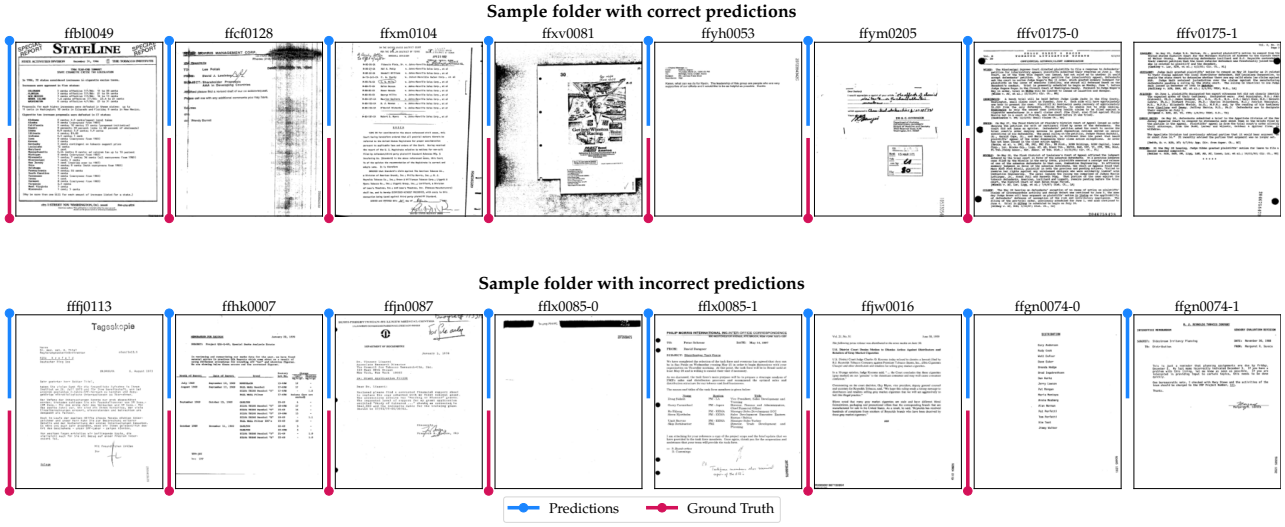
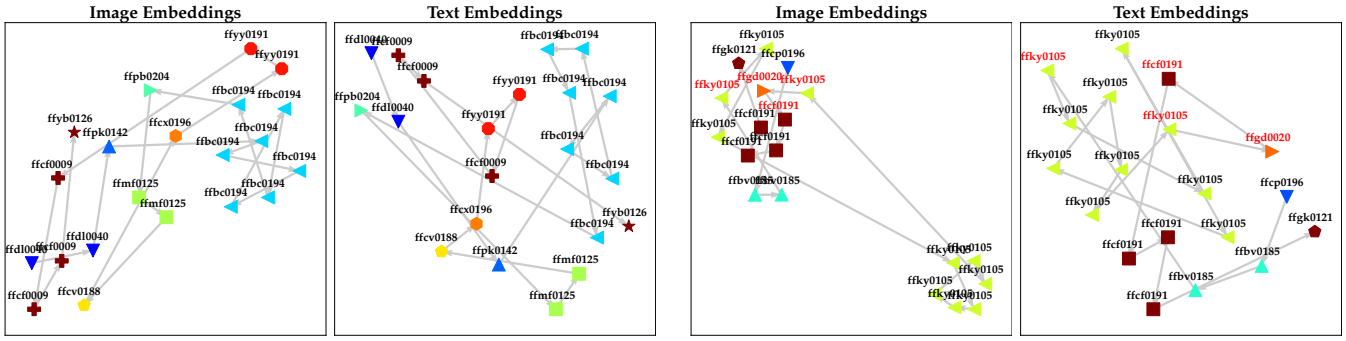


Figure 5: Sample predictions of the proposed model. In red (below) we indicate the correct breaking-point and in blue (above) the predicted one.



(a) A folder with all correctly identified breaking-points.

(b) A folder with some incorrectly identified breaking-points.

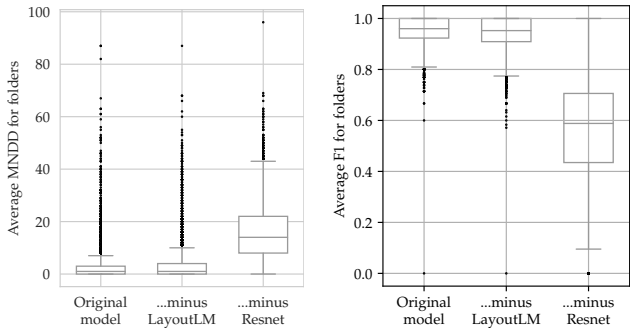
Figure 6: UMAP of the image and text embeddings for folders with correctly (left) and incorrectly (right) identified breaking-points. A point represents a page. Each point is labeled with its document identifier. Points of the same color belong to the same document. The identifiers are marked in black if the prediction is correct and in red if it is incorrect. The arrows connecting the points indicate the order of the pages in the folder and documents.

as demonstrated by the high score achieved, generating a result that in expectation requires only around 2 drag-and-drops corrections by the user as indicated by the MNDD metric. In Figure 5, we show two sample predictions. In the first sample, the model performs correctly. In the second sample, the model incorrectly identified the 5th and the 8th pages as breaking-points when none should have been inferred. The source of this mistake could be explained by observing that rarely in our dataset a new document starts with a blank page and the format of the 7th page is different from the 8th. We also compare the performance of our model against the baseline provided by Wiedemann and Heyer [14] on the same test set. The baseline achieves an accuracy of 0.919, while our model achieves an accuracy of 0.977, outperforming the baseline.

To gain some intuitions about how the model operates, we perform a Uniform Manifold Approximation and Projection (UMAP) [13] on the embeddings generated by ResNet and LayoutLM, $\mathbf{h_r}$ and $\mathbf{h_l}$. This is shown in Figure 6. UMAP is a nonlinear dimensionality reduction technique often used to visualize embeddings in a lower-dimensional space, 2 in this case. Here we find that page embeddings from the same document are generally closer in the vector space than those from other documents. The incorrect predictions generally occur when the breaking-points are close to the previous page, or when the non-breaking-points are far away from the previous page. Here we observe that the clustering is more apparent in the image embeddings than in the text embeddings. However, when the prediction is correct, both kinds of information are necessary since when pages belonging to the same document are not close in the image embedding's space they are close in the text embedding's space and vice versa.

Table 3: The ablation study.

Model	MNDD	Precision	Recall	F1
Our model	**3.440**	**0.947**	**0.964**	**0.953**
... minus LayoutLM	3.910	0.944	0.948	0.942
... minus ResNet	15.928	0.940	0.421	0.559



(a) MNDD (b) F1 score

Figure 7: Box plot of the MNDD (a) and F1 scores (b) for the ablation study.

5.2 Ablation Study (RQ2)

In Table 3, we present the results of the ablation study. We observe that the model clearly benefits from both models by performing much better when the two components are used. We also observe that for this task the ResNet has a stronger influence on the final performance by achieving a better MNDD score. This is not a surprise since this task has a strong visual component. This finding is corroborated by also looking at the higher F1 score achieved when removing this component. This was also observed when looking at the embeddings in Section 5.1, where we saw that the image embeddings tend to cluster more the pages belonging to the same document than the text embeddings.

In Figure 7 we present the box-plots of the MNDD and F1 score metrics, where we can observe that there is likely to be a difference between the full model and the model with LayoutLM ablated since in both measures its median line lies outside of the box of the combined model. In contrast, this is not observed when ablating ResNet. To determine if there is a statistical difference between the models, we perform a paired t-test between the full model and the ablated ones. All differences between our model and the ablated ones are statistically significant with a p-value < 0.05. Detailed statistics are available in the Appendix.

5.3 Analysis of Model Biases (RQ3)

To assess if the model is biased against the position of the breaking-points, for each page number i, we filtered out the folders in the test sets which have page i as a breaking-point, and calculated their average F1 score. In Figure 8, we observe that there is no substantial bias in terms of the position of the breaking-point within the folder. The model is able to correctly classify the breaking point at any position in the folder. Note that the high variance we observe at the end of the plot is due to the limited number of folders considered for those breaking-point positions.

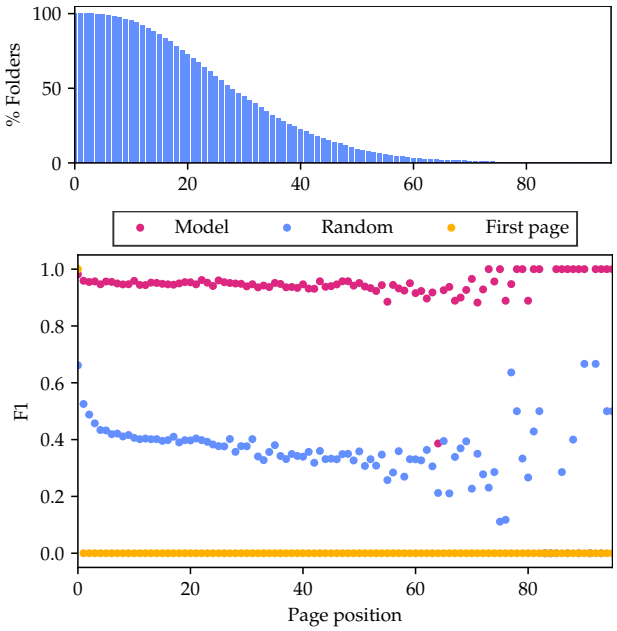


Figure 8: Results with respect to the page position: the histogram of folders' frequency (above) and the average F1 score (below).

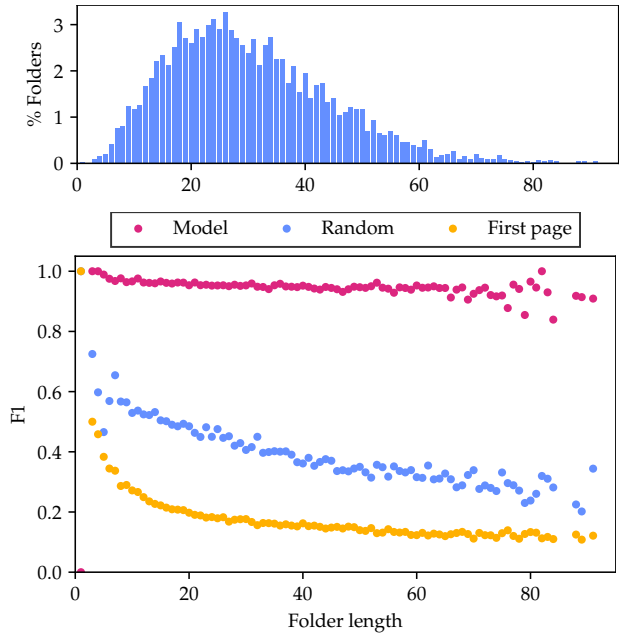


Figure 9: Results with respect to the folder length: the histogram of folders' frequency (above) and the average F1 score (below).

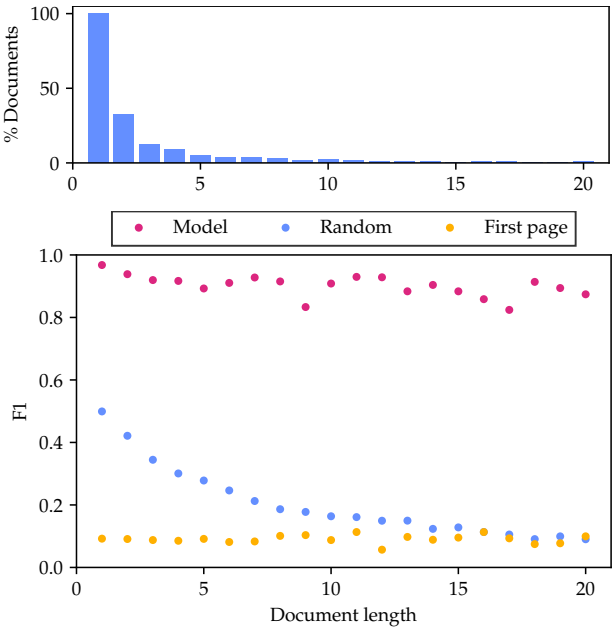


Figure 10: Results with respect to the document length: the histogram of folders' frequency (above) and average F1 score (below).

To assess how the model performance degrades with the length of the folder, in Figure 9 we plotted the average F1 score across folders with different lengths. The higher variance we observe when the folder length increases is due to the fact that the number of folders with a given folder length exponentially decreases when the folder length increases. In this plot, we observe that measuring the F1 score across the folder lengths has a natural bias against longer folders. This is because there is a higher chance to measure a smaller F1 score when the folder length increases, as also shown by the performance measured on the baseline models. However, for our model, we observe a relatively constant performance.

To assess how the model performance degrades with the length of the documents, we extracted all the documents from all folders with a given length and calculated the average F1 score across them. Our model again has a relatively constant performance across the document lengths, in comparison with a quick decrease in performance in the *random* model. This decrease in value is due to the same rationale provided in the paragraph above.

6 CONCLUSION AND FUTURE WORK

In this paper, we introduced a user-centric evaluation measure for the Page Stream Segmentation (PSS) task, the MNDD, which measures the Minimum Number of Drag-and-Drops the user would need to perform if the model inferred the wrong segmentation, and constructed the TABME dataset. We also provided a model that exploits both visual and textual information using ResNet and LayoutLM as backbones. This baseline achieves good performance and further experimentation demonstrated the robustness of this model. With the ablation study, we also demonstrated the importance of including a purely visual component.

In the TABME task, we assume that the pages of a document in a folder are continuous and sorted. In future work, we aim to investigate when these two assumptions are not true. This new task could be considered as a generalization of the proposed one. Such a task, in fact, naturally requires the use of graphs to represent documents, i.e., nodes to represent pages and links to represent their order. For this reason, we are currently experimenting with graph neural networks.

REFERENCES

[1] Muhammad Zeshan Afzal, Andreas Kölsch, Sheraz Ahmed, and Marcus Liwicki. 2017. Cutting the Error by Half: Investigation of Very Deep CNN and Advanced Training Strategies for Document Image Classification. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 01. 883–888. https://doi.org/10.1109/ICDAR.2017.149

[2] Onur Agin, Cagdas Ulas, Mehmet Ahat, and Can Bekar. 2015. An approach to the segmentation of multi-page document flow using binary classification. In *Sixth International Conference on Graphic and Image Processing (ICGIP 2014)*, Vol. 9443. International Society for Optics and Photonics, 944311.

[3] Hani Daher and Abdel Belaïd. 2014. Document flow segmentation for business applications. In *Document Recognition and Retrieval XXI*, Vol. 9021. International Society for Optics and Photonics, 90210G.

[4] Hani Daher, Mohamed-Rafik Bouguelia, Abdel Belaid, and Vincent Poulain D'Andecy. 2014. Multipage Administrative Document Stream Segmentation. In *2014 22nd International Conference on Pattern Recognition*. 966–971. https://doi.org/10.1109/ICPR.2014.176

[5] Abhijit Guha, Abdulrahman Alahmadi, Debabrata Samanta, Mohammad Zubair Khan, and Ahmed H. Alahmadi. 2022. A Multi-Modal Approach to Digital Document Stream Segmentation for Title Insurance Domain. *IEEE Access* 10 (2022), 11341–11353. https://doi.org/10.1109/ACCESS.2022.3144185

[6] Ahmed Hamdi, Mickaël Coustaty, Aurélie Joseph, Vincent Poulain d'Andecy, Antoine Doucet, and Jean-Marc Ogier. 2018. Feature Selection for Document Flow Segmentation. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. 245–250. https://doi.org/10.1109/DAS.2018.66

[7] Ahmed Hamdi, Joris Voerman, Mickaël Coustaty, Aurelie Joseph, Vincent Poulain d'Andecy, and Jean-Marc Ogier. 2017. Machine Learning vs Deterministic Rule-Based System for Document Stream Segmentation. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 05. 77–82. https://doi.org/10.1109/ICDAR.2017.332

[8] Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis. 2015. Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*. 991–995. https://doi.org/10.1109/ICDAR.2015.7333910

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Romain Karpinski and Abdel Belaïd. 2016. Combination of Structural and Factual Descriptors for Document Stream Segmentation. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. 221–226. https://doi.org/10.1109/DAS.2016.21

[11] D. Lewis, G. Agam, S. Argamon, O. Frieder, D. Grossman, and J. Heard. 2006. Building a Test Collection for Complex Document Information Processing. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '06)*. Association for Computing Machinery, New York, NY, USA, 665–666. https://doi.org/10.1145/1148170.1148307

[12] David Lewis, Gady Agam, Shlomo Argamon, Ophir Frieder, David Grossman, and Jefferson Heard. 2006. Building a test collection for complex document information processing. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. 665–666.

[13] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. 2018. UMAP: Uniform Manifold Approximation and Projection. *The Journal of Open Source Software* 3, 29 (2018), 861.

[14] Gregor Wiedemann and Gerhard Heyer. 2021. Multi-modal page stream segmentation with convolutional neural networks. *Language Resources and Evaluation* 55, 1 (2021), 127–150.

[15] 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. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (Jul 2020). https://doi.org/10.1145/3394486.3403172

A TRAINING AND VALIDATION

A.1 Hyper-Parameter Search

We study the influence of the structure of the 1D convolutional layers. We have experimented with the size of output vector for each convolutional layer. This size starts at $N \times 1280$ (the dimensionality of the embeddings) and drops to $N \times 2$ (the dimensionality of the predictions). We found that the performance of the architecture is better in terms of validation loss when the size of the convolutional layers between the input and output layers decays linearly along the forward layers than when the size decays geometrically. Therefore, based on the linear decay, we explored architectures with 0, 1, 2, 3, 4, 5, and 6 hidden layers using a learning rate of $5 \cdot 10^{-5}$. The minimum validation loss across five repeats was found in the model with 5 hidden layers. The results from the hyper-parameter search are shown in Table 4.

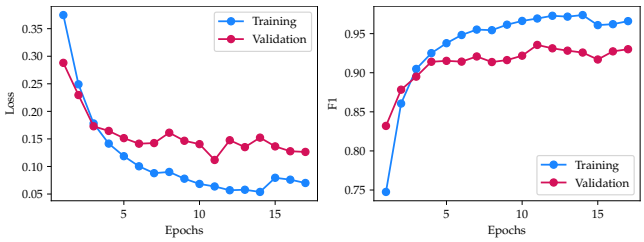
Table 4: Results from the hyper-parameter search.

# of Hidden Layers	Validation Loss	Validation F1
0	0.380	0.783
1	0.132	0.927
2	0.121	0.930
3	0.113	0.929
4	0.114	0.934
5	**0.104**	**0.941**
6	0.118	0.934

A.2 Validated Architecture

The breakdown of 1D convolution layers in our model is shown in Table 5. The layers Permute, Squeeze and Unsqueeze are added to facilitate the convolution operation along the series of pages that is specific to the way this can be implemented with PyTorch.

A.3 Learning Curves



(a) Loss **(b) F1**

Figure 11: Learning curves of the validated architecture.

The learning curves for the training of the model with the best hyper-parameters configuration are provided in Figure 11.

B ABLATION STUDY

To determine if there is a statistically significant difference between our model and the baseline models, we perform paired t-tests of the corresponding F1 and MNDD scores. The detailed test statistics are shown in Table 6.

Table 5: Breakdown of the convolutional layers.

Layer	Output Shape	# of Parameters
Input	(N, 1280)	0
Permute	(1280, N)	0
Unsqueeze	(1, 1280, N)	0
Conv1d	(1, 1067, N)	4098347
ReLU	(1, 1067, N)	0
Dropout	(1, 1067, N)	0
Conv1d	(1, 854, N)	2734508
ReLU	(1, 854, N)	0
Dropout	(1, 854, N)	0
Conv1d	(1, 641, N)	1642883
ReLU	(1, 641, N)	0
Dropout	(1, 641, N)	0
Conv1d	(1, 428, N)	823472
ReLU	(1, 428, N)	0
Dropout	(1, 428, N)	0
Conv1d	(1, 215, N)	276275
ReLU	(1, 215, N)	0
Dropout	(1, 215, N)	0
Conv1d	(1, 2, N)	1292
Permute	(1, N, 2)	0
Squeeze	(N, 2)	0
Softmax	(N, 2)	0
Total		9,576,777

Table 6: T-tests for ablation studies

Test	p-value	t-stat (df=4999)
F1: full v.s. −ResNet	~ 0.0	136.863
F1: full v.s. −LayoutlM	$4.44 \cdot 10^{-40}$	13.369
MNDD: full v.s. −ResNet	~ 0.0	-84.211
MNDD: full v.s. −LayoutlM	$9.21 \cdot 10^{-06}$	-4.440