# CTC Transcription Alignment of the Bullinger Letters: Automatic Improvement of Annotation Quality

Marco Peer[1], Anna-Scius Bertrand[2] and Andreas Fischer[1]
[1]University of Fribourg, Switzerland
[2]University of Applied Sciences and Arts Western Switerland, Fribourg, Switzerland
marco.peer@unifr.ch, anna.scius-bertrand@hefr.ch, andreas.fischer@unifr.ch

## Abstract

*Handwritten text recognition for historical documents remains challenging due to handwriting variability, degraded sources, and limited layout-aware annotations. In this work, we address annotation errors—particularly hyphenation issues—in the Bullinger correspondence, a large 16th-century letter collection. We introduce a self-training method based on a CTC alignment algorithm that matches full transcriptions to text line images using dynamic programming and model output probabilities trained with the CTC loss. Our approach improves performance (e.g., by 1.1 percentage points CER with PyLaia) and increases alignment accuracy. Interestingly, we find that weaker models yield more accurate alignments, enabling an iterative training strategy. We release a new manually corrected subset of 100 pages from the Bullinger dataset, along with our code and benchmarks. Our approach can be applied iteratively to further improve the CER as well as the alignment quality for text recognition pipelines. Code and data are available via https://github.com/andreas-fischer-unifr/nntp.*

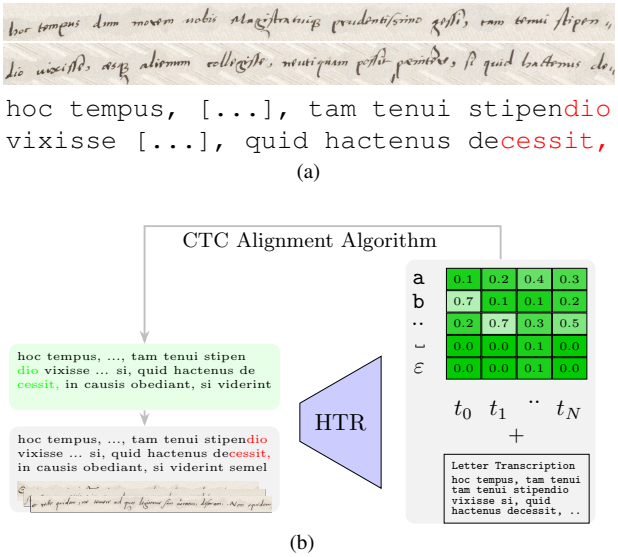



Figure 1. Bullinger dataset. (a) Examples of alignment errors (red) in the Bullinger dataset. (ID: 6_00_r11{19,20}) (b) Overview of our method. We use our CTC alignment algorithm and propose a self-training scheme to improve HTR performance to remove errors caused by hyphenations.

## 1. Introduction

Handwritten Text Recognition (HTR) remains a challenging task in document analysis, particularly for historical documents, due to the high variability of handwriting styles, the degraded condition of the sources, and the diversity of languages and scripts [7]. In addition, the annotation process requires historical expertise to produce accurate transcriptions, which makes it both time-consuming and costly. End-to-end learning approaches based on deep neural networks further increase this issue, as they typically rely on large quantities of annotated training data in the form of text line images paired with their transcriptions [10]. Given the high effort involved in creating such ground truth data, a trade-off often arises between data quality and quantity.

A key difficulty in the context of acquiring ground truth for training lies in the differing requirements of historical scholarship and computational processing: historians, on the one hand, lean towards regularized versions of texts where abbreviations are resolved and special characters are normalized to suit the reading habits of the contemporary audience, whereas HTR systems rely on character-accuracte transcriptions with respect to the visual representation [13]. In this work, we address one major challenge of this process through a case study — namely the hyphenation problem due to an automated alignment — on the Bullinger correspondence [19], a large and linguistically diverse collection of 16th-century letters from Switzerland, for which experts have produced accurate transcriptions of the letters of Heinrich Bullinger, an influential Swiss Reformer and

theologian, and his correspondences. However, the current transcriptions are provided only letter-wise without any layout information to enable HTR training (transcriptions done by historians for the Bullinger collection are publicly available via GitHub[1]). Therefore, to create training data for HTR, the Bullinger dataset contributed by Scius-Bertrand et al.[19] applies an automatic alignment via Transkribus[15] between these letter-level transcriptions and extracted text line images. While this process enables the creation of a large dataset — over 100,000 line images - it also introduces systematic errors, mainly caused by hyphenations at the beginning and end of lines. Although multiple other errors are present in the ground truth, including misaligned parts of hyphenated words, abbreviations that are written out in full and therefore differ from the image, incorrect capitalization, and mismatched punctuation marks, Jungo et al.[10] identified hyphenation as the most common inconsistency within the data collection. It does not only affect the quality of the training data but also the reliability of evaluation results when using the Bullinger dataset for HTR research. An example of a wrong hyphenation in the training set is shown in Figure1a.

To tackle the misalignment, improve the annotation quality and study its influence on the Bullinger collection, we introduce a self-training procedure based on a Connectionist Temporal Classification (CTC) [8] alignment algorithm in our work, as shown in Figure 1b. It uses dynamic programming to find an optimal sequence with respect to the output probabilities of our HTR models trained with the CTC loss function. Furthermore, it takes the full letter transcription (the ground truth - provided by experts, without any layout information), and outputs the aligned text in terms of lines. We align the training set of the Bullinger dataset [19] and finetune the models to, as we show in our results, improve the HTR - in case of the PyLaia framework by 1.1 percentage points on the CER – as well as the alignment accuracy. Secondly, we inspect our algorithm and find that the alignment accuracy is even higher when using a weakly trained model. This process can be efficiently repeated iteratively to further enhance the text recognition. To examine our methodology, results are evaluated on a manually created and corrected dataset consisting of 100 pages in total to ensure precise transcription.

To summarize, our contributions are as follows:

- We provide a new, manually annotated and corrected dataset of the Bullinger collection,
- We introduce a CTC alignment algorithm for text alignment to insert line breaks to match a given layout when ground truth transcriptions are available,
- And we propose a self-training strategy to iteratively improve the HTR performance when using the alignment method.

The code for the alignment as well as the data to benchmark our algorithm will be publicly available. The remainder of the paper is structured as follows: Section 2 describes related work in the domain of alignment and self-training approaches. Section 3 explains our methodology, the evaluation protocol is reported in Section 4, followed by the results in Section 5. We conclude our paper in Section 6.

## 2. Related Work

This section briefly describes related work in the general domain of computer vision, followed by historical document analysis.

### 2.1. Alignment in computer vision

Image-text alignment in computer vision refers to the process of establishing meaningful correspondences between visual data and textual descriptions, e.g., in the context of sign language recognition, where sequences of video frames are aligned with gloss-level transcriptions [11, 14, 25], aligning videos with textual descriptions [20, 23, 24], or 3D scenes with textual descriptions [26]. The alignment of the visual and the textual modalities serves for automatic transcription, automatic captioning, and cross-modal retrieval, e.g., retrieving a video based on a text or vice versa. Alignment results can also be used for self-learning, using the aligned pseudo-labels for supervised fine-tuning.

### 2.2. Alignment for historical manuscripts

For historical manuscripts, Sudholt and Fink proposed a word-level alignment between images and text with the PHOCNet [21], a CNN-based network that estimates a pyramidal histogram of characters [1] representation of text labels. It allows to retrieve keyword images from the (word-segmented) manuscripts using both text queries and image queries.

Most alignment methods in the literature operate at the line level, aiming at simplifying the navigation in digital libraries (switching easily between the transcription and the original handwriting images) and, more commonly, with the goal of leveraging existing human transcriptions of ancient documents to extract line-level pseudo-labels for training HTR systems. The methods differ in the optical models they use and the algorithms considered for alignment, which typically include some form of dynamic programming.

Early work includes the alignment method proposed by A. Fischer et al. [6] for historical Latin manuscripts, which is based on character hidden Markov models (HMMs) and Viterbi decoding for alignment. Spelling variants are included in the models and it is demonstrated that the alignment task can be solved with high accuracy even for weakly trained models using only few annotated learning samples.

[1] https : / / github . com / bullinger - digital / bullinger-korpus-tei

More recently, Ezra et al. [4] proposed an alignment method for fragments of the Dead Sea Scrolls. After line segmentation, a CNN-RNN based HTR system is used to produce an automatic recognition result, which is then aligned with the transcription using string edit distance. In the work of P. Torras et al. [22], a Seq2Seq model is used to align historical handwritten ciphers based on attention mechanisms. The approach of B. Madi et al. [12] targets the task of image-image alignment between Arabic manuscripts that contain similar texts. YOLO object detection is considered for spotting subwords, subword similarity is established by means of siamese CNN, and alignment is achieved using a longest common subsequence algorithm. YOLO object detection is also considered in the work of A. Sicius-Bertrand et al. [18] for aligning columns of ancient Vietnamese manuscripts with their transcription, utilizing clustering algorithms for the alignment. An interesting aspect of their work is that neither page segmentation nor human-annotated learning samples are needed for aligning the logographic characters.

In this paper, we introduce a line-level alignment method that is based on a deep CNN-RNN with CTC loss. Instead of relying on the recognition result as basis for the alignment, as in [4], we leverage the character CTC posterior probabilities, similar to the character HMM likelihoods in [6], and perform a Viterbi-like dynamic programming to find the optimal correspondence between the character probabilities and the transcription. The CTC character probabilities can be seen as a preliminary character-level image-text alignment performed by the CNN-RNN, which is then used for aligning the transcription of entire letters, consisting of multiple scanned pages.

## 3. Methodology

In this section, we present our methodology, starting with the manual annotation and correction of two subsets of the Bullinger collection[2]. Then, we present the CTC alignment algorithm proposed in our work, followed by the approaches used for HTR. Lastly, the self-training procedure is explained.

### 3.1. Bullinger Dataset

The Bullinger dataset [19] is a historical manuscript collection consisting of 3,622 letters written by 306 authors, drawn from the larger correspondence of Swiss reformer Heinrich Bullinger (1504-–1575). The dataset reflects the linguistic diversity of the period, with letters primarily written in Latin and premodern German, alongside passages in Greek, Italian, French, and Hebrew. Transcriptions are not strictly character-accurate, as abbreviations are often expanded, and the handwriting - particularly that of Bullinger himself — can be highly variable and difficult to read, posing challenges for both human annotators and HTR systems. Additionally, due to the automated alignment, done with the Transkribus platform [15], errors arise at the beginning and end of text lines, particularly in the presence of word breaks.

[2] https://www.bullinger-digital.ch/index.html

| | Letters | Pages | Lines |
|---|---|---|---|
| Subset 1 | 20 | 50 | 902 |
| Subset 2 | 49 | 50 | 1486 |
| Total | 69 | 100 | 2388 |

Table 1. Statistics of the manually corrected dataset.

Therefore, we create a manually annotated and corrected dataset of 100 pages, consisting of 2,388 text lines, split into two subsets. The first subset contains transcriptions made by historians, reflecting how they interpret the text. These transcriptions are written as continuous paragraphs rather than line by line; hyphenated words at line breaks in the images are not preserved. Written abbreviations are expanded, and punctuation is added. This subset also represents the workflow of the annotations provided in the Bullinger dataset [19].

The second subset contains what we call diplomatic transcriptions, meaning they closely reflect the original line-by-line content of the images. Hyphenated words are preserved, abbreviations are kept as written, and no modern punctuation is added. The dataset statistics are shown in Table 1.

### 3.2. Handwritten Text Recognition

We use three different models for HTR: HTRFlor [3] (820k parameters), PyLaia [16] (6.4M), and the method by Retsinas et al. [17] (7.4M). While the use of HTRFlor and PyLaia follows the approach of the original paper [19], the method by Retsinas et al. presents a trade-off between training speed and number of parameters. It retains image aspect ratio, replaces max-pooling with column-wise concatenation to reduce parameters, and introduces a parallel shortcut branch with a 1D convolution and CTC [8] loss to guide the CNN backbone toward learning more discriminative features. The general structure is a CNN-based backbone, followed by either a block of BiLSTM [9] (Retsinas et al., PyLaia) or BGRU [2] (HTRFlor) layers. For our work, we follow the default architectures and training strategies proposed by the authors. The architectures of our models are shown in the supplementary material (Figure 9).

### 3.3. CTC Alignment

The proposed image-text alignment method is inspired by the NNTP token passing algorithm introduced in [5] for HTR using very large vocabularies. Our method is designed for HTR systems that are based on the CTC loss

function [8]. Such systems provide estimated character posterior probabilities $P(c_i|t_j)$ for all characters $c_i \in \mathcal{A}$ of the alphabet $\mathcal{A}$, including the special $\varepsilon$ character indicating "no character", for all time steps $t_j$ when processing the convolutional features of text line images using recurrent neural networks (LSTM or GRU, respectively). While the probabilities are estimated bi-directionally from left-to-right and from right-to-left, the time axis corresponds to the natural reading order of the handwriting.

**Problem statement** The alignment algorithm receives two inputs: the machine-readable transcription of the letter, and the character posterior probabilities for all text line images of the letter. The goal is to insert newline characters at the correct position within the transcription, such that it is aligned with the text line images, resulting in labeled samples for training HTR systems. For each aligned text line, the average character probability, as well as the average probability over the first six and the last six characters is returned as measures of confidence.

Note that in this problem statement, the transcription itself is not changed, i.e. it is not possible to skip words, change or add characters, swap parts of the text, etc. However, if there are parts of the image that are missing in the transcription, the alignment can insert multiple newline characters in order to create a gap in the aligned transcription.

**Alignment process** The alignment proceeds in three steps. In the first step, both the letter transcription and the character posterior probabilities are preprocessed. After discarding all newline characters from the letter transcription, following the principle of CTC, the transcription is extended to a linear, Finite State Automaton (FSA) with skip connections that also include the $\varepsilon$ character, as illustrated in Figure 2. Note that there is no direct connection between two identical characters. At least one $\varepsilon$ needs to be visited between them, such that a character transition can be detected. Regarding the character probabilities, they are concatenated over all lines and pages of a handwritten letter, in order to form a single probability sequence for the entire letter. Afterwards, we compress the sequence by means of a threshold $\theta_\varepsilon$, such that successive time steps with $P(\varepsilon|t_j) > \theta_\varepsilon$ are compressed into a single time step with probability

$$P(\varepsilon|t_{ab}) = \prod_{j=a}^{b} P(\varepsilon|t_j) \; ,$$

and similar for all other characters of the alphabet. This $\varepsilon$ compression is inspired by the observation that for CTC, the $\varepsilon$ character is almost always "active" with nearly 100% probability, interrupted only sporadically by time steps, in which standard characters have a probability significantly larger than 0%.

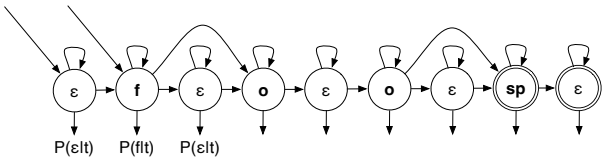



Figure 2. FSA representing the transcription "foo " after $\varepsilon$ extension ('sp' is the whitespace character). Each character state emits posterior probabilities with respect to the time $t$. There are two initial states (diagonal arrows) and two final states (double circles).

In the second, main step, dynamic programming is used to find the optimal character sequence $c^* = c_1^*, \ldots, c_T^*$ with $c_j^* \in \mathcal{A}$, and $T$ the length of the ($\varepsilon$-compressed) letter probability sequence, such that

$$c^* = \underset{c_1,\ldots,c_T \in \mathcal{C}_{FSA}}{\arg\max} \prod_{j=1}^{T} P(c_j|t_j) \; ,$$

where $\mathcal{C}_{FSA}$ contains all valid character sequences according to the FSA of the text transcription (see Figure 2). A schema of the dynamic programming procedure is illustrated in Figure 3, highlighting the maximum probability step for standard characters (3 possible predecessors) and the $\varepsilon$ character (2 possible predecessors) according to the FSA. To avoid numerical problems with small probabilities, the algorithm considers sums of logarithms. For increased memory efficiency, token passing is applied, i.e. only one column is kept in memory in form of the FSA and starting from the initial states, tokens are passed to valid successors in the automaton at each time step. At time $T$, the tokens in the final states hold the optimal solution.

In the third and final step, newline characters are inserted in the transcription according to the line break positions of the text line images in the optimal character sequence $c^*$. Furthermore, the two confidence measures

$$\gamma = \frac{1}{e-s} \sum_{j=s}^{e} P(c_j^*|t_j) \; ,$$

$$\gamma_6 = \frac{1}{6} \sum_{j=s}^{s+6} P(c_j^*|t_j) + \frac{1}{6} \sum_{j=e-6}^{e} P(c_j^*|t_j) \; ,$$

are calculated, where $t_s$ is the starting time and $t_e$ is the ending time of the aligned text line. They reflect the quality of the aligned learning samples with respect to the average character probability, focusing on the beginning and the end of the line in the case of $\gamma_6$. For small text lines with less than 12 characters, $\gamma_6$ is set to zero.

### 3.4. Self-Training

We can now address alignment errors by implementing the self-training pipeline. The procedure begins by extracting

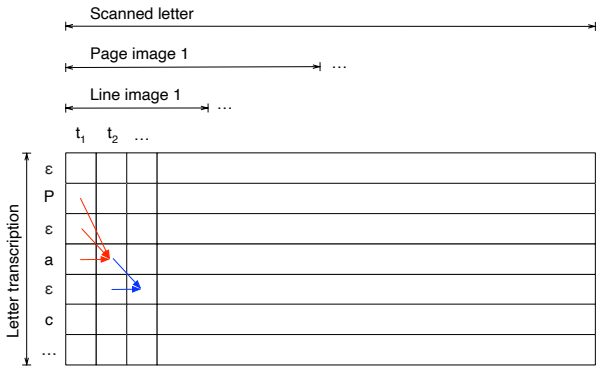



Figure 3. Schema of the dynamic programming algorithm proposed for letter transcription alignment based on CTC character probabilities. Red arrows illustrate the maximum probability step for standard characters, blue arrows for the $\varepsilon$ character.

the probabilities for $P(c_i|t_j),\ c_i \in \mathcal{C}_{FSA}$ for each timestep from the three different HTR models. These probabilities are combined with expert-provided letter transcriptions that lack layout information. Using these inputs, the CTC alignment algorithm performs sequence alignment by inserting line breaks to the corresponding characters in the transcription.

Following alignment, we generate training subsets by applying filtering based on $\gamma_6$, the confidence of the alignment for the first and last six characters. For example, we retain only samples that meet a minimum average character confidence for $\gamma_6$. The filtered subsets of the Bullinger's training and validation set are subsequently used to finetune the HTR models. This adaptation is aimed to improve the HTR performance, in particular when alignment issues, such as hyphenation, are occuring. We show that the procedure can be applied iteratively - improved model predictions are used to generate refined alignments, which in turn support further finetuning.

# 4. Evaluation

## 4.1. Setup

**Data** We use the original Bullinger dataset [19] and remove the letters which occur in our contributed test set, resulting in 108.7k lines for training (instead of 109.6k). The alphabet to generate $\mathcal{C}_{FSA}$ consists of the 79 symbols of the training set (including the $\varepsilon$ of the CTC).

**Training and Finetuning** We use the defaults parameters for training and model architecture of the three frameworks. For finetuning, we decrease the number of epochs to 10% of the initial training phase and lower the learning rate by 0.01. In both phases, a learning rate scheduler is used, and all of our models converge.

**CTC Alignment** The threshold for the $\theta_\varepsilon$ is set to 0.99 in our experiments. If not stated otherwise, we use the PyLaia model for HTR and finetune it on the alignments filtered with $\gamma_6 > 0.5$.

## 4.2. Metrics

In the following, we briefly describe the metrics used for evaluation.

**Line-level Accuracy** To quantitatively evaluate the quality of predicted text lines against the ground truth in our handwritten text recognition system, we adopt a line-level accuracy metric. For each letter $i$ (note that a letter usually consists of more than one page), the transcription is segmented into lines by our algorithm and compared to the corresponding ground truth lines in the same order. Let the ground truth lines be $G_i = \left(g_i^{(1)}, g_i^{(2)}, \ldots, g_i^{(L_i)}\right)$, where $L_i$ is the number of ground truth lines, and the predicted lines be $P_i = \left(p_i^{(1)}, p_i^{(2)}, \ldots, p_i^{(M_i)}\right)$, where $M_i$ is the number of predicted lines. We define a per-line match indicator

$$\delta_i^{(j)} = \begin{cases} 1 & \text{if } p_i^{(j)} = g_i^{(j)} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } j = 1, \ldots, \min(L_i, M_i). \tag{1}$$

The number of correctly matched lines for sample $i$ is then

$$m_i = \sum_{j=1}^{\min(L_i, M_i)} \delta_i^{(j)}. \tag{2}$$

Finally, the overall line-level accuracy across the dataset of $N$ samples is computed as the total number of correctly matched lines normalized by the total number of ground truth lines.

**HTR** For evaluating the performance of the HTR models, we provide Character Error Rate (CER) and Word Error Rate (WER) metrics. Additionally, we report the CER metric for the first and last $n$ characters of a line, denoted as $\text{CER}_n$, to specifically evaluate the influence of the text alignment on the HTR performance.

# 5. Results

In this part, we present the main results of the proposed methodology.

## 5.1. Baseline Results for Bullinger

First, we evaluate our three HTR models on the Bullinger dataset [19] and compare it with the results provided in the original work, shown in Table 2. The worst model is HTR-Flor (with the fewest number of parameters), followed by Retsinas et al. and PyLaia, that performs best with 8.9 % / 9.9 % CER on the frequent/nonfrequent subset of the

| | Frequent | | Nonfrequent | |
|---|---|---|---|---|
| | CER | WER | CER | WER |
| HTRFlor [3] | 11.8 | 43.7 | 12.8 | 46.4 |
| PyLaia [16] | **8.9** | **31.5** | **9.9** | **35.0** |
| Retsinas et al. [17] | 9.0 | 29.6 | 10.3 | 33.7 |
| Baseline [19] | 8.4 | 29.6 | 9.9 | 34.4 |

Table 2. HTR performance on the two writer subsets of the Bullinger dataset [19].

| | Subset 1 | | Subset 2 | |
|---|---|---|---|---|
| | CER | WER | CER | WER |
| HTRFlor [3] | 15.7 | 52.4 | 22.8 | 64.9 |
| PyLaia [16] | **7.6** | **28.9** | **18.6** | **54.0** |
| Retsinas et al. [17] | 13.0 | 40.1 | 21.7 | 58.6 |

Table 3. HTR performance on the 100 manually corrected pages.

| | Subset 1 | Subset 2 |
|---|---|---|
| HTRFlor [3] | **86.1** | **79.7** |
| PyLaia [16] | **86.1** | 77.3 |
| Retsinas et al. [17] | 81.5 | 73.0 |

Table 4. Line-level alignment accuracy.

dataset. Scius-Bertrand et al. [19] also provide the binarized version of the line images to the input (resulting in a four channel image), that might explain the small gap in performance.

Next, we evaluate the same models as in Table 2 on our manually corrected dataset, which consists of two subsets of 50 pages each. The results are presented in Table 3. PyLaia outperforms the other two approaches by 5.4/8.1% and 3.1/4.2% CER on the two subsets, respectively. However, the performance on Subset 2 is consistently worse across all approaches, which may be due to differences in annotation style. The models were trained on Annotation Set 1, where abbreviations were expanded and punctuation was added. As a result, the models tend to reproduce this normalized format rather than the original transcription as it appears in Subset 2.

### 5.2. Text Alignment

We evaluate the toking passing algorithm by reporting the line-level accuracy on both subsets in Table 4. All models achieve accuracies above 80 and 70%, resp., on both of our subsets. Similarly to the HTR performance, the results are worse for Subset 2. Interestingly, the smallest model, HTRFlor, outperforms PyLaia, the best model in terms of CER, on Subset 2. We will evaluate the influence of the model size in our ablation study.

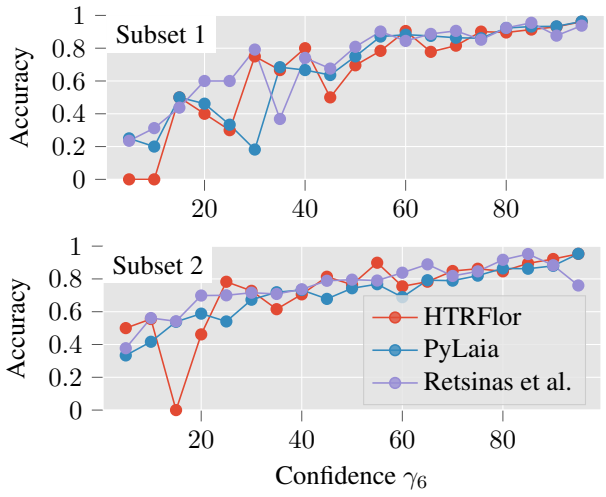



Figure 4. Confidence values of the first and last six characters vs. line-level accuracy. Higher confidence values correspond to higher line-level accuracies.

In Figure 4, we show the line-level accuracy for each approach depending on the confidence of the first/last six characters on our test set. We notice that higher confidence scores also correspond to higher line-level accuracies, with Retsinas for Subset 2 being the only outlier for higher confidence thresholds, explaining the lower accuracy in Table 4.

| | $T_{\text{Line}}$ | $T_{\text{L}}^{*}$ | $T_{\text{L}}$ | $\varepsilon$ | $d$ in s |
|---|---|---|---|---|---|
| HTRFlor [3] | 256 | 11242 | 6897 | 1.63 | 3.54 |
| PyLaia [16] | 214 | 8915 | 5643 | 1.58 | 2.70 |
| Retsinas et al. [17] | 126 | 5759 | 4840 | 1.19 | 2.28 |

Table 5. Average length of the line ($T_{\text{Line}}$) and the letter probability sequence ($T_{\text{L}}^{*}/T_{\text{L}}$ after/before the compression), the achieved compression ratio $\varepsilon$ when applying thresholding with $\theta_\varepsilon = 0.99$ and the average run-time per letter $d$ for different recognition models.

### 5.3. Computational Analysis

We analyze the computational differences regarding the alignment between the models in Table 5. First, the number of time steps per algorithm varies: HTRFlor uses 256 steps per line, while Retsinas et al. use 128. However, they propose inserting whitespace at the beginning and end of each line during training (removed at inference), effectively resulting in 126 steps per line. PyLaia applies adaptive pooling, with an average of 214 steps per line.

Interestingly, we observe that $\varepsilon$-compression reduces the effective sequence length per letter more efficiently for HTRFlor and PyLaia, a result of the sparsity of the CTC

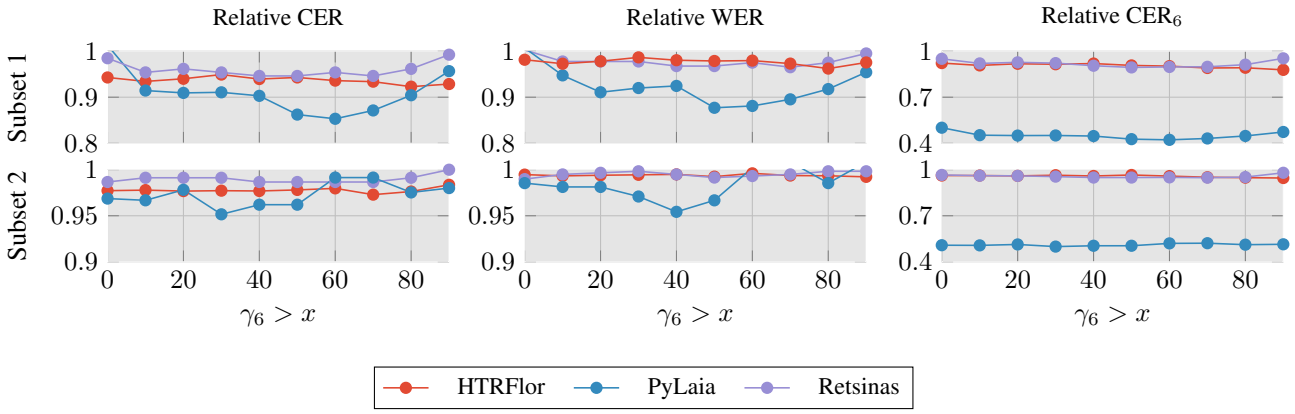



Figure 5. Finetuning the models on the aligned training set for different thresholds $\gamma_6$. (a) Normalized CER/WER improvements for the models. The best $\gamma_6$ is a trade off between training data used and confidence (30 – 60%), while PyLaia shows the largest improvement.

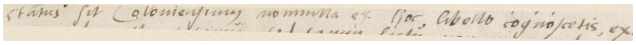
probabilities ("peaks") and the relative difference of the steps per line. Overall, we have an average run-time for an alignment of a letter (consisting of 34.6 lines on average) of 2.28 - 3.54s, depending on the algorithm. We conclude that the CTC alignment algorithm can even be applied to longer sequences (or HTR models with a larger number of steps) by using the proposed $\varepsilon$-compression.

### 5.4. Self-Training

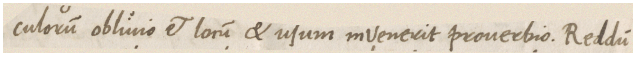
After the text alignment, we run the algorithm on the training set of the Bullinger dataset and create subsets to finetune our HTR models. Firstly, we evaluate the influence of the confidence threshold. The relative improvements on the baseline CER and WER metrics are shown in Figure 5. We notice that the improvement for PyLaia is the largest (which can particularly be observed when evaluating the CER$_6$, where the performance improves by 50%), and the improvement is more pronounced for Subset 1. In general, all models benefit from thresholding, but the gains vary depending on the model and metric. PyLaia exhibits the strongest sensitivity to confidence filtering, while HTRFlor and Retsinas show more moderate but consistent trends.

We observe that the optimal threshold is a trade-off between data quantity and quality, as the highest improvements are obtained when considering samples with a confidence higher than 30–60%, depending on the approach and subset. This is also in line with the findings by Jungo et al. [10]. When increasing the confidence threshold, the number of training lines decreases significantly (see Figure 10 in the supplementary), which can negatively affect the finetuning process if too few samples are considered for training. On the other hand, lower thresholds allow for more but also faulty data (see Figure 4).

Table 6 presents the absolute CER and WER scores for each model when finetuned using a confidence threshold of 50%. Consistent with the trends observed in Figure 5, PyLaia achieves the best overall performance across both subsets and metrics (with an improvement of 1.1% on the CER on Subset 1). However, our algorithm and finetuning strategy improves the results of all three models. We also show qualitative results in Figure 6, where we observe that the finetuned model performs better at the beginning and the end of the line, while the baseline tends to drop the first/last word of the line.



**GT** status sit Coloniensium nonnulla ex hoc libello cognoscetis ex
**FT** status sit Coloniensium nonnila ex hoc libello cognoscetis ex
**BL** sit Coloviensium nomnia ex hoc libello cognoscetisex



**GT** culorum obliuio et lorum & usum invenerit prouerbio. Redduo
**FT** clorum oblivio et locum et usum invenerit proverbio. Reddum
**BL** oblivio et locum et usum invenerit proverbio. Reddum

**GT** Es wayr sye, das vnser kunning in Inglandt gestorben
**FT** es waist sye, das unser kunning in iuglandt gestothen
**BL** es wait sye, das unser kunning in Inglandt gestorben

Figure 6. Qualitative comparison of the finetuned (FT) and the baseline (BL) PyLaia [16] model. Red and green words highlight differences that are only correct in one of the predictions.

| | Subset 1 | | Subset 2 | |
|---|---|---|---|---|
| | CER | WER | CER | WER |
| HTRFlor [3] | 14.8 ↓0.9 | 51.2 ↓1.2 | 22.2 ↓0.6 | 64.4 ↓0.5 |
| PyLaia [16] | **6.5** ↓1.1 | **25.4** ↓3.5 | **17.9** ↓0.7 | **52.2** ↓1.8 |
| Retsinas [17] | 12.2 ↓0.8 | 38.8 ↓1.3 | 21.3 ↓0.4 | 58.2 ↓0.4 |

Table 6. Absolute finetuning improvements for a threshold of 50%.

**Line-level Accuracy** Table 7 reports the line-level alignment accuracy after finetuning the HTR models. Overall, the improvements in transcription quality lead to better alignment accuracy for most models. HTRFlor shows the highest post-finetuning accuracy on both subsets, with a gain of 3.4% on Subset 1. Retsinas et al. also improves across both subsets, though to a lesser extent. Interestingly, PyLaia shows a slight drop in accuracy on Subset 1, likely due to a saturation effect, as its pre-finetuning performance on Subset 1 is already very high. This suggests that for stronger models, further finetuning with filtered data may not always translate into better alignment, especially when initial transcriptions are already of high quality in terms of CER. With the improved alignment accuracy, the models can then be finetuned on the enhanced training data again.

| | Subset 1 | Subset 2 |
|---|---|---|
| HTRFlor [3] | **89.5** ↑3.4 | **81.0** ↑1.3 |
| PyLaia [16] | 85.6 ↓0.5 | 78.9 ↑1.6 |
| Retsinas et al. [17] | 82.9 ↑1.4 | 73.5 ↑0.6 |

Table 7. Line-level accuracy of the alignment after finetuning the HTR models.

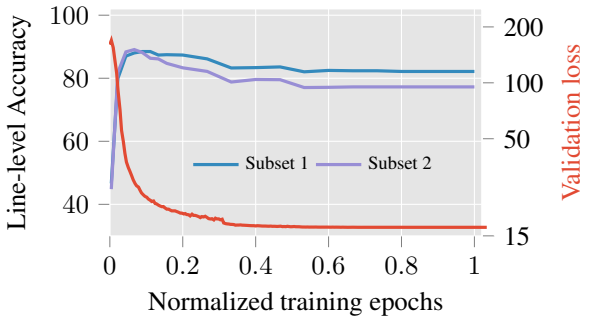



Figure 7. Line-level accuracy during training

## 5.5. Weakly Trained Model

Next, we examine the influence of the training steps on the alignment accuracy to check whether a weakly trained model has an improved benefit from the alignment since it has a minor training on the faulty data. The results are shown in Figure 7. The model is trained for 225 epochs, and the probabilities of the weakly trained model do result in a better line-level accuracy – 88.5% vs. 86.1% on Subset 1 and 89.1% vs. 77.3% on Subset 2, both reached after 15 epochs of training. We also find that the line-level accuracy does converge after approximately half of the training time in our case. This result highlights that the self-training is more beneficial when the model is trained for a reduced amount of optimization steps, making our approach more efficient with respect to time.

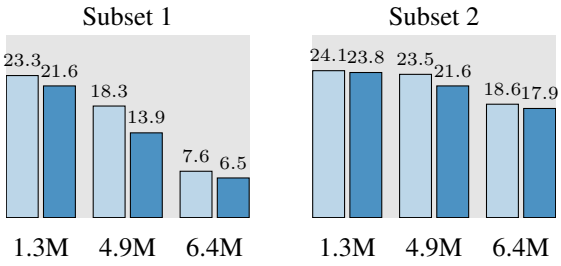



Figure 8. Influence of the model size before and after finetuning on the CER.

## 5.6. Model size

We evaluate the impact of model size by simplifying the PyLaia architecture; details of the reduced models are in the supplementary. The small model uses a 2-layer RNN with 128 units, while the mid and large models use 3 layers with 256 units. As shown in Figure 8, larger models lead to lower CERs both before and after fine-tuning. On Subset 1, the CER drops from 23.3% to 7.6% as model size increases from 1.3M to 6.4M parameters (6.5% after fine-tuning). A similar trend is observed on Subset 2, where CER decreases from 24.1% to 18.6% (17.9% after fine-tuning).

# 6. Conclusion

This work introduced a self-training method based on a CTC alignment algorithm to improve annotation quality within the Bullinger dataset. We showed improved performance of three different HTR approaches by finetuning on aligned subsets of the initial training data. Our algorithm is efficient in terms of computational complexity, even for longer sequences. Additionally, results indicated that weakly trained models are more suitable for self-training in terms of line-level accuracy, making an iterative process feasible. Lastly, we provided 100 annotated and manually corrected pages of the Bullinger dataset for further research.

For future work, this paper is the first step toward acquiring annotations close to the original visual text. We plan to extend the algorithm to skip words, enabling detection of missing words or abbreviations, common in expert annotations of historical manuscripts. This extension would help obtain character-accurate annotations of the Bullinger data collection to further improve HTR performance.


## Acknowledgements

This work has been supported by the Hasler Foundation, Switzerland. We thank the contributors of the Bullinger Digital project, in particular Tobias Hodel, Anna Janka, Raphael Müller, Peter Rechsteiner, Patricia Scheurer, David Selim Schoch, Raphael Schwitter, Christian Sieber, Phillip Ströbel, Martin Volk, and Jonas Widmer for their contributions to the dataset and ground truth creation.

## References


[1] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566, 2014. 2

[2] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Association for Computational Linguistics, 2014. 3

[3] Arthur Flor de Sousa Neto, Byron Leite Dantas Bezerra, Alejandro Héctor Toselli, and Estanislau Baptista Lima. Htr-flor++: A handwritten text recognition system based on a pipeline of optical and language models. In *DocEng '20*, page 1–4. ACM, 2020. 3, 6, 7, 8, 11, 12

[4] Daniel Stökl Ben Ezra, Bronson Brown-DeVost, Nachum Dershowitz, Alexey Pechorin, and Benjamin Kiessling. Transcription alignment for highly fragmentary historical manuscripts: The dead sea scrolls. In *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 361–366. IEEE, 2020. 3

[5] Andreas Fischer. *Handwriting Recognition in Historical Documents*. Ph.d. dissertation, University of Bern, 2012. 3

[6] Andreas Fischer, Volkmar Frinken, Alicia Fornés, and Horst Bunke. Transcription alignment of latin manuscripts using hidden markov models. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, pages 29–36, 2011. 2, 3

[7] Andreas Fischer, Marcus Liwicki, and Rolf Ingold. *Handwritten historical document analysis, recognition, and retrieval - state of the art and future trends*. World Scientific Publishing, Singapore, Singapore, 2020. 1

[8] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*, page 369–376. ACM Press, 2006. 2, 3, 4

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 3

[10] Michael Jungo, Lars Vögtlin, Atefeh Fakhari, Nathan Wegmann, Rolf Ingold, Andreas Fischer, and Anna Scius-Bertrand. Impact of the ground truth quality for handwriting recognition. In *Proceedings of the 12th International Symposium on Information and Communication Technology*, page 135–140. ACM, 2023. 1, 2, 7

[11] Ahmet Alp Kındıroglu, Oğulcan Özdemir, and Lale Akarun. Aligning accumulative representations for sign language recognition. *Machine Vision and Applications*, 34(1):12, 2023. 2

[12] Boraq Madi, Ahmad Droby, and Jihad El-Sana. Textline alignment on the image domain. *International Journal on Document Analysis and Recognition (IJDAR)*, 25(4):415–427, 2022. 3

[13] Martin Mayr, Julian Krenz, Katharina Neumeier, Anna Bub, Simon Bürcky, Nina Brolich, Klaus Herbers, Mechthild Habermann, Peter Fleischmann, Andreas Maier, and Vincent Christlein. Nuremberg letterbooks: A multi-transcriptional dataset of early 15th century manuscripts for document analysis. *Scientific Data*, 12(1), 2025. 1

[14] Yuecong Min, Aiming Hao, Xiujuan Chai, and Xilin Chen. Visual alignment constraint for continuous sign language recognition. In *proceedings of the IEEE/CVF international conference on computer vision*, pages 11542–11551, 2021. 2

[15] Guenter Muehlberger, Louise Seaward, Melissa Terras, Sofia Ares Oliveira, Vicente Bosch, Maximilian Bryan, Sebastian Colutto, Hervé Déjean, Markus Diem, Stefan Fiel, Basilis Gatos, Albert Greinoecker, Tobias Grüning, Guenter Hackl, Vili Haukkovaara, Gerhard Heyer, Lauri Hirvonen, Tobias Hodel, Matti Jokinen, Philip Kahle, Mario Kallio, Frederic Kaplan, Florian Kleber, Roger Labahn, Eva Maria Lang, Sören Laube, Gundram Leifert, Georgios Louloudis, Rory McNicholl, Jean-Luc Meunier, Johannes Michael, Elena Mühlbauer, Nathanael Philipp, Ioannis Pratikakis, Joan Puigcerver Pérez, Hannelore Putz, George Retsinas, Verónica Romero, Robert Sablatnig, Joan Andreu Sánchez, Philip Schofield, Giorgos Sfikas, Christian Sieber, Nikolaos Stamatopoulos, Tobias Strauß, Tamara Terbul, Alejandro Héctor Toselli, Berthold Ulreich, Mauricio Villegas, Enrique Vidal, Johanna Walcher, Max Weidemann, Herbert Wurster, and Konstantinos Zagoris. Transforming scholarship in the archives through handwritten text recognition: Transkribus as a case study. *Journal of Documentation*, 75 (5):954–976, 2019. 2, 3

[16] Joan Puigcerver. Are multidimensional recurrent layers really necessary for handwritten text recognition? In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, pages 67–72, 2017. 3, 6, 7, 8, 11, 12

[17] George Retsinas, Giorgos Sfikas, Basilis Gatos, and Christophoros Nikou. Best practices for a handwritten text recognition system. In *Document Analysis Systems*, page 247–259, 2022. 3, 6, 7, 8, 11, 12

[18] Anna Scius-Bertrand, Michael Jungo, Beat Wolf, Andreas Fischer, and Marc Bui. Transcription alignment of historical vietnamese manuscripts without human-annotated learning samples. *Applied Sciences*, 11(11):4894, 2021. 3

[19] Anna Scius-Bertrand, Phillip Ströbel, Martin Volk, Tobias Hodel, and Andreas Fischer. The bullinger dataset: A writer adaptation challenge. In *Document Analysis and Recognition - ICDAR 2023 - 17th International Conference, San José, CA, USA, August 21-26, 2023, Proceedings, Part I*, pages 397–410, 2023. 1, 2, 3, 5, 6

[20] Xiujun Shu, Wei Wen, Haoqian Wu, Keyu Chen, Yiran Song, Ruizhi Qiao, Bo Ren, and Xiao Wang. See finer, see more: Implicit modality alignment for text-based person retrieval. In *European Conference on Computer Vision*, pages 624–641. Springer, 2022. 2

[21] Sebastian Sudholt and Gernot A. Fink. Evaluating Word String Embeddings and Loss Functions for CNN-based Word Spotting. In *Proc. Int. Conf. on Document Analysis and Recognition*, 2017. 2

[22] Pau Torras, Mohamed Ali Souibgui, Jialuo Chen, and Alicia Fornés. A transcription is all you need: Learning to align

through attention. In *Document Analysis and Recognition–ICDAR 2021 Workshops: Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part I 16*, pages 141–146. Springer, 2021. 3

[23] Ziyang Wang, Yi-Lin Sung, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Unified coarse-to-fine alignment for video-text retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2816–2827, 2023. 2

[24] Jianwei Yang, Yonatan Bisk, and Jianfeng Gao. Taco: Token-aware cascade contrastive learning for video-text alignment. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11562–11572, 2021. 2

[25] Jiangbin Zheng, Yile Wang, Cheng Tan, Siyuan Li, Ge Wang, Jun Xia, Yidong Chen, and Stan Z Li. Cvt-slr: Contrastive visual-textual transformation for sign language recognition with variational alignment. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 23141–23150, 2023. 2

[26] Ziyu Zhu, Xiaojian Ma, Yixin Chen, Zhidong Deng, Siyuan Huang, and Qing Li. 3d-vista: Pre-trained transformer for 3d vision and text alignment. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2899–2909. IEEE Computer Society, 2023. 2

## 7. Model architectures

Figure 9 presents the default architectural configurations of the three HTR systems evaluated in this work: Retsinas et al. [17], PyLaia [16], and HTRFlor [3]. The model proposed by Recsinas et al. (Figure 9a) employs a deep residual convolutional neural network with multiple ResBlocks and intermediate max pooling layers, followed by a column-wise max pooling operation. Sequence modeling is performed by a single BiLSTM layer with 256 hidden units, followed by a dense output layer. Additionally, the architecture incorporates a CTC shortcut path consisting of a $1 \times 1$ convolution for intermediate supervision. PyLaia (Figure 9b) adopts a relatively shallow CNN feature extractor composed of five convolutional layers with batch normalization and LeakyReLU activations. This is followed by a stack of five BiLSTM layers, each with 256 hidden units per direction. The final output is produced by a softmax layer applied after a dense transformation.

HTRFlor (Figure 9c) is based on gated convolutions and uses PReLU activations combined with Batch Renormalization and MaxNorm regularization. The encoder consists of six gated convolutional blocks with increasing filter sizes, interleaved with dropout and normalization layers. Max pooling and tiling are applied before the recurrent decoder, which comprises two stacked BGU layers, each with 128 hidden units. The final dense layer maps the outputs to character probabilities, followed by a softmax operation. In our experiments, we use the unmodified, default versions of all three architectures as provided in their respective implementations.

## 8. PyLaia model architectures

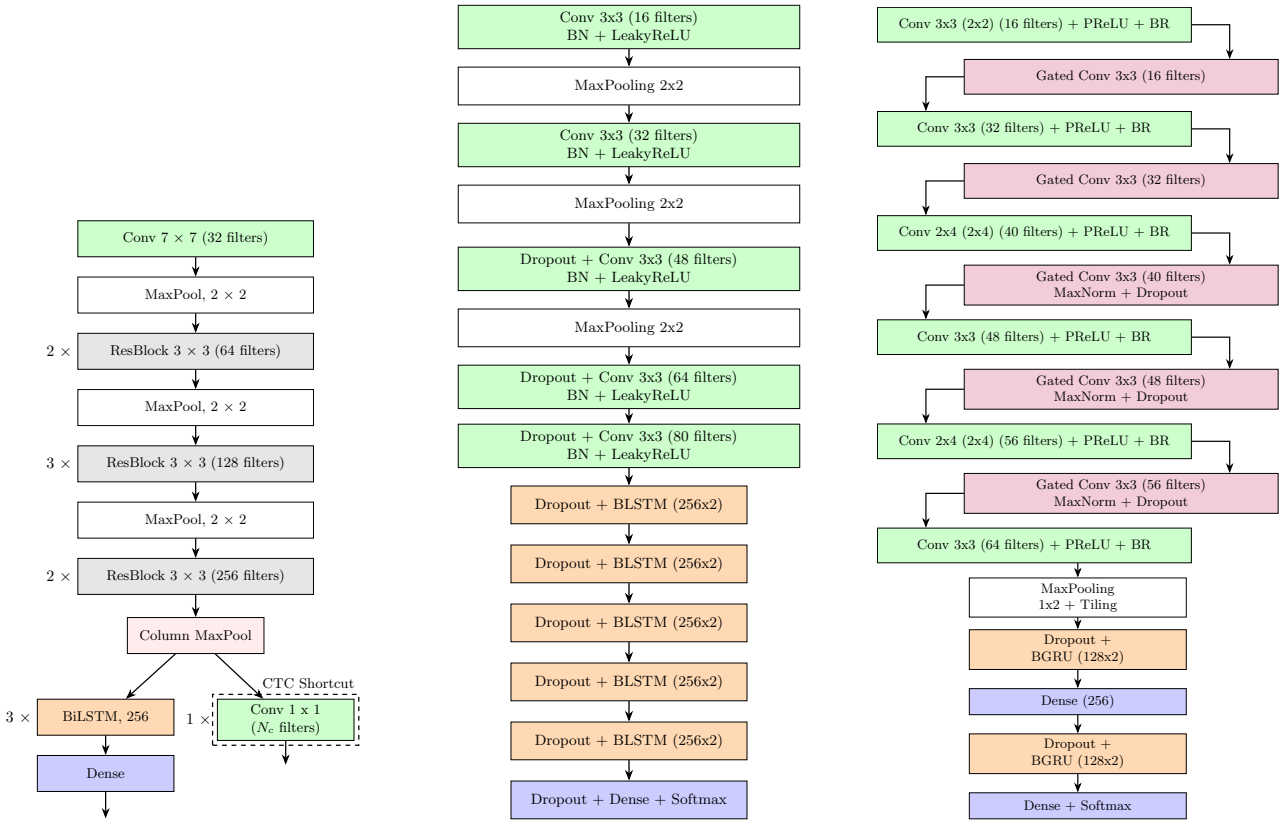
In Table 8, the detailed architectures of the PyLaia model for the ablation study on the model size are reported. Large corresponds to the default architecture and is also the default provided in the PyLaia Framework. The main difference between the mid and large model is the reduced size of the CNN backbone (3 vs. 4), while the small and mid model differ mainly in terms of the reduced size of the RNN.

### 8.1. Subset size for finetuning

Figure 10 illustrates the number of training samples retained for finetuning at various confidence thresholds. As the confidence threshold increases from 0 to 90, the number of samples decreases across all three models: HTRFlor, PyLaia, and Retsinas. HTRFlor consistently maintains a higher number of samples compared to the other models at most thresholds, while Retsinas shows the steepest decline. This shows how stricter confidence criteria reduce the training dataset size, which impacts the model finetuning by reducing the amount of faulty data (while also having less data).

| Parameter | Small | Mid | Large |
|---|---|---|---|
| **CNN** | | | |
| Features | [16, 24, 48] | [16, 24, 36] | [16, 16, 32, 32] |
| Kernel Size | [3, 3, 3] | [3, 3, 3] | [3, 3, 3, 3] |
| Stride | [1, 1, 1] | [1, 1, 1] | [1, 1, 1, 1] |
| Dilation | [1, 1, 1] | [1, 1, 1] | [1, 1, 1, 1] |
| Pool Size | [2, 2, 2] | [2, 2, 2] | [2, 2, 2, 0] |
| Dropout | 0 | 0 | 0 |
| Activation | LeakyReLU | LeakyReLU | LeakyReLU |
| **RNN** | | | |
| Layers | 2 | 3 | 3 |
| Units | 128 | 256 | 256 |
| Type | LSTM | LSTM | LSTM |
| Dropout | 0.5 | 0.5 | 0.5 |
| **Linear** | | | |
| Dropout | 0.5 | 0.5 | 0.5 |

Table 8. Comparison of Small (1.3M), Mid (4.9M), and Large (6.4M) PyLaia model configurations. (T = True, F = False)




(a) Retsinas et al. [17] (b) PyLaia [16] (c) HTRFlor [3]

Figure 9. Comparison of the three architectural designs used for text recognition.

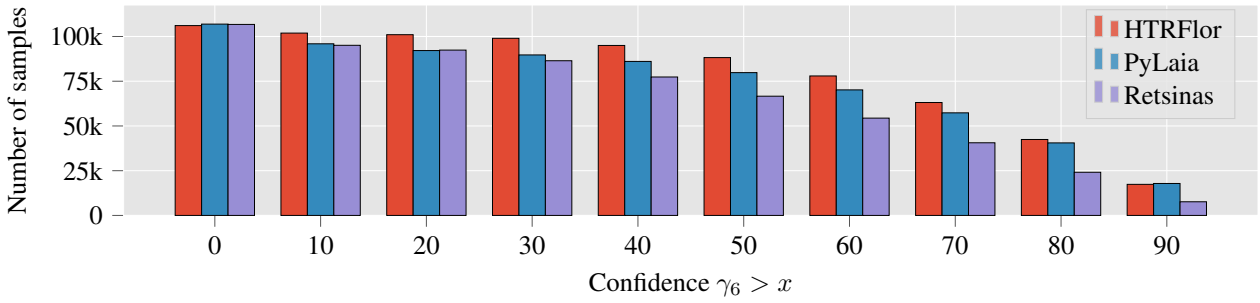



Figure 10. Number of training samples used for finetuning for different thresholds.