NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

DATA SCIENCE AND INFORMATION TECHNOLOGIES

MSc THESIS

Handwritten Optical Character Recognition

Charalampos P. Vossos

Supervisors: **Stavros Perantonis**, Research Director, NCSR Demokritos
Vassilis Gatos, Research Director, NCSR Demokritos

ATHENS

August 2024

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναγνώριση Χειρόγραφων Εγγράφων

Χαράλαμπος Π. Βόσσος

Επιβλέποντες: Σταύρος Περαντώνης, Διευθυντής Ερευνών, ΕΚΕΦΕ Δημόκριτος
Βασίλης Γάτος, Διευθυντής Ερευνών, ΕΚΕΦΕ Δημόκριτος

ΑΘΗΝΑ

Αύγουστος 2024

MSc THESIS

Handwritten Optical Character Recognition

Charalampos P. Vossos
S.N.: 711512200037

SUPERVISORS: **Stavros Perantonis**, Research Director, NCSR Demokritos
Vassilis Gatos, Research Director, NCSR Demokritos

EXAMINATION COMMITTEE: **Stavros Perantonis**, Research Director NCSR Demokritos
Vassilis Gatos, Research Director NCSR Demokritos
Vassilis Katsouros, Director of ILSP Athena Research Center

Examination Date: 12 August 2024

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αναγνώριση Χειρόγραφων Εγγράφων

Χαράλαμπος Π. Βόσσος
Α.Μ.: 711512200037

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Σταύρος Περαντώνης**, Διευθυντής Ερευνών, ΕΚΕΦΕ Δημόκριτος
Βασίλης Γάτος, Διευθυντής Ερευνών, ΕΚΕΦΕ Δημόκριτος

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Σταύρος Περαντώνης**, Διευθυντής Ερευνών ΕΚΕΦΕ Δημόκριτος
Βασίλης Γάτος, Διευθυντής Ερευνών ΕΚΕΦΕ Δημόκριτος
Βασίλης Κατσούρος, Διευθυντής του ΙΕΛ Ερευνητικό Κέντρο Αθηνά

Ημερομηνία Εξέτασης: 12 Αυγούστου 2024

ABSTRACT

This thesis focuses on recognizing Old Greek handwritten manuscripts, which are written in lowercase and capital letters. Although the text is written in Greek polytonic orthography, predictions are made at the level of letters without the use of diacritical marks. All the pages come from proceedings of the General Board of the Bank of Greece, a collection of handwritten Parliamentary Questions from the Historical Archive (1974-1977) of the Greek Parliament and from an 1842 Greek handwritten translation of Shakespeare's Macbeth, and are split into text line images. Each image corresponds to a ground truth text file with the context. We employ four different models. Firstly, we use the Calamari OCR system, which is an open source toolkit for text line recognition. Calamari follows the encoder decoder architecture with CNN and bidirectional LSTM models, respectively. Since the dataset is not aligned, the CTC loss function has been used during training. Additionally, users can apply augmentations to the dataset and use GPUs for faster training. To test the trained model on unseen line images, at least one model is required for predictions. We employ the default network with one bidirectional LSTM at the decoder, and one larger network with three bidirectional LSTM layers. In addition, we implement two state-of-the-art transformer-based (TrOCR) models with pre-trained image transformers at the encoder and bidirectional LSTM layers at the decoder. We use a large version of the encoder with more parameters and three bidirectional LSTM layers at the decoder, as well as a small version of the encoder with fewer parameters and one bidirectional LSTM decoder. In order to increase the robustness of the model, we apply some popular augmentation techniques. Input images are resized to 384×384 and then are split into sequences of 16×16 patches before being forwarded to the image transformers. As a typical transformer model, a shelf-attention mechanism is applied. Furthermore, we use the CTC loss function for training. We train every model on the same datasets. During inference, we employ greedy search, beam search, and beam search with the n-gram language model decoding methods, keeping the beam search as the most accurate decoding algorithm. Furthermore, we analyze our proposed solution for the Brazilian Essay competition of the ICDAR 2024 Competition on Handwritten Text Recognition in Brazilian Essays – BRESSAY. There are 3 different tasks depending on the input images. Text line images, paragraphs, and entire pages are given for the 3 tasks, respectively. Our approach combines the YOLOv5 model for text line detection (for tasks 2 and 3) and Calamari OCR for text line recognition. The crucial step is the Contrast Limited Adaptive Histogram Equalization (CLAHE) that we apply on the given images, increasing the contrast between text and background. This is beneficial for the text line detection phase. The dataset poses significant challenges due to its diverse authorship, the presence of noise, difficult-to-read words, overwriting, and the presence of strike-through texts. Furthermore, we present our results in terms of the Character Error Rate and the Word Error Rate (WER) metrics. Finally, we compare a popular CNN-based model, like Calamari OCR, with pre-trained transformer-based models using the Character Error Rate (CER), the Word Error Rate metrics (WER), the recall, the F1 score, the precision, and the Mathews Correlation Co-

efficient (MCC). We conclude that a pre-trained transformer-based model outperforms a CNN-based model like Calamari OCR in terms of the accuracy in recognizing characters, but Calamari OCR is more accurate at word-level recognition.

SUBJECT AREA: Handwritten Optical Character Recognition

KEYWORDS: Calamari, OCR, transformers, WER, CER, CNN, LSTM, YOLOv5, TrOCR, Beam Search

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική επικεντρώνεται στην αναγνώριση παλαιών ελληνικών χειρόγραφων εγγράφων, τα οποία είναι γραμμένα με πεζά και κεφαλαία γράμματα. Αν και τα κείμενα είναι γραμμένο με το ελληνικό πολυτονικό σύστημα γραφής, οι προβλέψεις γίνονται σε επίπεδο γραμμάτων χωρίς τόνους και πνεύματα. Όλες οι σελίδες προέρχονται από τα πρακτικά του Γενικού Συμβουλίου της Τράπεζας της Ελλάδος, από μια συλλογή χειρόγραφων κοινοβουλευτικών ερωτήσεων από το ιστορικό αρχείο (1974-1977) της Βουλής των Ελλήνων και από μια ελληνική χειρόγραφη μετάφραση του Μάκμπεθ του Σαίξπηρ από το 1842 και χωρίζονται σε εικόνες γραμμών κειμένου. Κάθε εικόνα αντιστοιχεί σε ένα αρχείο κειμένου με το πραγματικό περιέχομενο. Χρησιμοποιούμε τέσσερα διαφορετικά μοντέλα. Αρχικά, χρησιμοποιούμε το Calamari OCR, το οποίο είναι ένα εργαλείο ανοικτού κώδικα για την αναγνώριση γραμμών κειμένου. Ακολουθεί την αρχιτεκτονική κωδικοποιητή αποκωδικοποιητή με μοντέλα CNN και αμφίδρομου LSTM αντίστοιχα και χρησιμοποιεί τη συνάρτηση απωλειών CTC κατά την εκπαίδευση, δεδομένου ότι το σύνολο δεδομένων δεν είναι ευθυγραμμισμένο. Επιπλέον, οι χρήστες έχουν τη δυνατότητα να εφαρμόζουν επαυξήσεις στο σύνολο δεδομένων και να επιτρέπουν τη χρήση GPU για ταχύτερη εκπαίδευση. Για να δοκιμαστεί το εκπαιδευμένο μοντέλο σε καινούργιες εικόνες γραμμών, απαιτείται τουλάχιστον ένα μοντέλο για προβλέψεις. Χρησιμοποιούμε το προεπιλεγμένο δίκτυο με ένα αμφίδρομο LSTM στον αποκωδικοποιητή και ένα μεγαλύτερο δίκτυο με τρία αμφίδρομα επίπεδα LSTM. Επιπλέον, υλοποιούμε δύο μοντέλα βασισμένα σε transformers (TrOCR) με προ-εκπαιδευμένους transformers εικόνας στον κωδικοποιητή και αμφίδρομα επίπεδα LSTM στον αποκωδικοποιητή. Χρησιμοποιούμε μια μεγάλη έκδοση του κωδικοποιητή με περισσότερες παραμέτρους και τρία αμφίδρομα στρώματα LSTM στον αποκωδικοποιητή, καθώς και μια μικρή έκδοση του κωδικοποιητή με λιγότερες παραμέτρους και έναν αμφίδρομο αποκωδικοποιητή LSTM.

Για να αυξήσουμε την ακρίβεια του μοντέλου εφαρμόζουμε ορισμένες δημοφιλείς τεχνικές επαύξησης. Οι εικόνες εισόδου αλλάζουν μέγεθος σε 384×384 και στη συνέχεια χωρίζονται σε ακολουθίες από 16×16 patches πριν προωθηθούν στους transformers εικόνας. Ως τυπικό μοντέλο transformers εφαρμόζεται ο μηχανισμός shelf-attention. Επιπλέον, χρησιμοποιούμε τη συνάρτηση απωλειών CTC για την εκπαίδευση. Εκπαιδεύουμε και τα δύο μοντέλα στα ίδια σύνολα δεδομένων. Κατά την εξαγωγή προβλέψεων χρησιμοποιούμε τις μεθόδους αποκωδικοποίησης greedy search, beam search και beam search με γλωσσικό μοντέλο n-gram, κρατώντας τον beam search ως τον πιο ακριβή αλγόριθμο αποκωδικοποίησης. Επιπλέον, αναλύουμε την προτεινόμενη λύση μας για τον διαγωνισμό αναγνώρισης βραζιλιάνικων εκθέσεων του ICDAR 2024. Υπάρχουν 3 διαφορετικά προβλήματα ανάλογα με τις εικόνες εισόδου. Για τα 3 προβλήματα δίνονται εικόνες γραμμών κειμένου, παράγραφοι και ολόκληρες σελίδες αντίστοιχα. Η προσέγγισή μας συνδυάζει το μοντέλο YOLOv5 για την ανίχνευση γραμμών κειμένου (για τα προβλήματα 2 και 3) και το Calamari OCR για την αναγνώριση γραμμών κειμένου. Κρίσιμο βήμα είναι η Contrast Limited Adaptive Histogram Equalization (CLAHE) που εφαρμόζουμε στις εικόνες, αυξά-

νοντας την αντίθεση μεταξύ κειμένου και φόντου. Αυτό είναι ευεργετικό για τη φάση της αναγνώρισης γραμμής κειμένου. Το σύνολο δεδομένων είναι εξαιρετικά δύσκολο με αρκετές ειδικές περιπτώσεις, επειδή είναι γραμμένο από διαφορετικούς συγγραφείς και περιέχει θορύβους, δυσανάγνωστες λέξεις, κείμενα με υπεργραφή και διαγραμμένα κείμενα. Επιπλέον, παρουσιάζουμε τα αποτελέσματά μας όσον αφορά τις μετρικές Character Error Rate (CER) και Word Error Rate (WER). Τέλος, συγκρίνουμε ένα δημοφιλές μοντέλο βασισμένο στο CNN όπως το Calamari OCR με ένα προ-εκπαιδευμένο μοντέλο βασισμένο σε transformers χρησιμοποιώντας τις μετρικές Character Error Rate (CER), Word Error Rate (WER), recall, F1 score, precision και Mathews Correlation Coefficient (MCC). Καταλήγουμε στο συμπέρασμα ότι ένα προ-εκπαιδευμένο μοντέλο που βασίζεται σε transformers υπερτερεί έναντι ενός μοντέλου που βασίζεται σε CNN όπως το Calamari OCR όσον αφορά την ακρίβεια στην αναγνώριση χαρακτήρων, αλλά το Calamari OCR έχει μεγαλύτερη ακρίβεια στην αναγνώριση λέξεων.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Αναγνωρίση χειρόγραφων εγγράφων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Χειρόγραφα έγγραφα, προ-εκπαιδευμένα μοντέλα, αλγόριθμοι αποκωδικοποίησης, αναγνώριση χαρακτήρων, αναγνώριση λέξεων

$$1.01^{365} = 37.78$$

$$1^{365} = 1$$

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Mr Stavros Perantonis for introducing me to the team of the Computational Intelligence Laboratory (CIL) of Demokritos and Mr Vassilis Gatos for all his help, observations, and time spent developing this postgraduate thesis. His guidance and support have helped me succeed in this task. I would also like to thank Konstantinos Palaiologos and Eleni Sarafoglou for the great collaboration we had. Last but not least, I would like to thank my family for all their support throughout these years.

CONTENTS

1 INTRODUCTION **25**

1.1 Optical Character Recognition Task . . . **25**

1.2 Thesis Contribution . . . **25**

1.3 Related Work . . . **26**

1.4 Thesis Overview . . . **27**

2 DATASET **29**

3 METHODOLOGY **33**

3.1 TrOCR . . . **33**
3.1.1 Introduction . . . 33
3.1.2 Augmentations . . . 34
3.1.3 TrOCR - LSTM . . . 34
3.1.3.1 Model Architecture . . . 35
3.1.3.2 Encoder . . . 35
3.1.3.3 Decoder . . . 36
3.1.3.4 Model Initialization . . . 36
3.1.3.5 Encoder Initialization . . . 36
3.1.3.6 Decoder Initialization . . . 37
3.1.3.7 Task Pipeline . . . 37
3.1.3.8 Pre-training . . . 37
3.1.3.9 Training Process . . . 37
3.1.3.10 Fine-tuning . . . 38
3.1.3.11 Data Augmentation . . . 38
3.1.3.12 Pre-training Dataset . . . 38
3.1.3.13 Evaluation Metrics . . . 39
3.1.3.14 Results . . . 39
3.1.4 CTC Loss . . . 39
3.1.5 Inference . . . 39

3.2 Calamari OCR . . . **40**
3.2.1 Methods . . . 40
3.2.1.1 Network Architecture Building Blocks . . . 40
3.2.1.2 Finetuning and Codec Resizing . . . 43

3.2.1.3 Voting . . . 43
3.2.2 The Calamari OCR-System . . . 44
3.2.2.1 Preprocessing . . . 44
3.2.2.2 Training . . . 45
3.2.2.3 Prediction . . . 45

3.3 ICDAR 2024 Competition . . . 45
3.3.1 Overview . . . 45
3.3.2 Significance of this Task . . . 46
3.3.3 Our Challenge . . . 46
3.3.4 Impact . . . 46
3.3.5 Challenges . . . 46
3.3.5.1 1st Challenge: Line-Level Recognition . . . 47
3.3.5.2 2nd Challenge: Paragraph-Level Recognition . . . 47
3.3.5.3 3rd Challenge: Page-Level Recognition . . . 47
3.3.6 Dataset Overview and Composition . . . 49
3.3.7 Our Approach . . . 50
3.3.8 BRESSAY Participants . . . 51

4 Evaluation 53

4.1 Evaluation Metrics . . . 53

4.2 Results . . . 55

4.3 ICDAR . . . 63
4.3.1 Line-Level Recognition Results . . . 64
4.3.2 Paragraph-Level Recognition Results . . . 65
4.3.3 Page-Level Recognition Results . . . 66

5 Conclusions and Future Work 69

5.1 Conclusions . . . 69

5.2 Future Work . . . 69

ABBREVIATIONS - ACRONYMS 71

REFERENCES 76

LIST OF FIGURES

2.1 Handwritten line image samples from an 1842 Greek handwritten translation of Shakespeare's Macbeth. 29
2.2 Handwritten line image samples of Parliamentary Questions from the Historical Archive (1974-1977). 30
2.3 Handwritten line image samples from the Bank of Greece. 31

3.1 TrOCR architecture. The encoder-decoder network consists of a pre-trained image transformer and a pre-trained text transformer, respectively. Image from [1]. 33
3.2 A typical Convolutional Neural Network architecture (Figure from [2]). . . . 41
3.3 Convolution Operation (Figure from [3]). 42
3.4 Example of confidence voting algorithm (Figure from [4]). 44
3.5 Handwritten line image samples. 47
3.6 Handwritten paragraph image samples. 48
3.7 Handwritten page image samples. 48

LIST OF TABLES

4.1 CER for Different Decoding Strategies using the TrOCR-LSTM Small Model 55
4.2 WER for Different Decoding Strategies using the TrOCR-LSTM Small Model 55
4.3 Precision for Different Decoding Strategies using the TrOCR-LSTM Small Model . . . 56
4.4 Recall for Different Decoding Strategies using the TrOCR-LSTM Small Model 57
4.5 F1 score for Different Decoding Strategies using the TrOCR-LSTM Small Model . . . 57
4.6 Matthews Correlation Coefficient (MCC) for Different Decoding Strategies using the TrOCR-LSTM Small Model . . . 58
4.7 Character Error Rate (CER) for Different Decoding Strategies using the TrOCR-LSTM Large Model . . . 58
4.8 Word Error Rate (WER) for Different Decoding Strategies using the TrOCR-LSTM Large Model . . . 59
4.9 Precision for Different Decoding Strategies using the TrOCR-LSTM Large Model . . . 59
4.10 Recall for Different Decoding Strategies using the TrOCR-LSTM Large Model 60
4.11 F1 score for Different Decoding Strategies using the TrOCR-LSTM Large Model . . . 60
4.12 Matthews Correlation Coefficient (MCC) for Different Decoding Strategies using the TrOCR-LSTM Large Model . . . 61
4.13 OCR Results Comparison - Macbeth Translation - Parliamentary Questions 62
4.14 OCR Results Comparison - Macbeth Translation - Bank Proceedings (284) 62
4.15 OCR Results Comparison - Macbeth Translation - Bank Proceedings (493) 62
4.16 ICDAR Line-Recognition results . . . 64
4.17 Line-Level Recognition Results – Without Punctuation (p), Without Accents (a), and Without Special Tags (s) . . . 64
4.18 Count of Special Tags Recognized in the Line Test Partition. . . . 65
4.19 ICDAR Paragraph-Recognition results . . . 65
4.20 Paragraph-Level Recognition Results – Without Punctuation (p), Without Accents (a), and Without Special Tags (s). . . . 66
4.21 Count of Special Tags Recognized in the Paragraph Test Partition. . . . 66
4.22 ICDAR Page-Recognition results . . . 67
4.23 Page-Level Recognition Results – Without Punctuation (p), Without Accents (a), and Without Special Tags (s). . . . 67
4.24 Count of Special Tags Recognized in the Page Test Partition. . . . 67

PREFACE

Optical Character Recognition (OCR) is the process of converting the text of a page into machine-encoded text. Pages can contain printed or handwritten characters. It can be used in various domains where the fast and accurate text recognition is essential. Some examples are the following:

1. **Digitization and archiving of documents:**
    - **Historical documents and manuscripts:** OCR can be used to digitize and store historical documents, manuscripts and letters. This makes them searchable and accessible for research and educational purposes.
    - **Legal and government documents:** Digitize handwritten legal documents, birth certificates, marriage licenses and other official records for easier retrieval and storage.
2. **Forms processing:**
    - **Surveys and questionnaires:** Automate the processing of handwritten responses to surveys, questionnaires and feedback forms.
    - **Application forms:** Process handwritten application forms for jobs, loans, admissions and other services to streamline data entry and reduce manual labor.
    - **Examinations and assessments:** Grading and processing of handwritten written examinations and evaluation forms at educational institutions.
3. **Mail and address recognition:**
    - **Postal services:** An OCR system can recognize addresses on envelopes and packages written by hand, and automate mail sorting.
    - **Courier and delivery services:** It can be used for recognizing handwritten addresses and instructions making parcel sorting more efficient.
4. **Health care:** Digitization is crucial for more efficient storage and easier access of handwritten prescriptions and medical records.
5. **Banking and financial services:** Recognizing handwritten information on checks to automate banking processes.
6. **Education:** Converting handwritten notes into digital forms may be beneficial for better organization and study. In addition, digital assignments and homework can be submitted and graded easier.

7. **Business and Enterprise:** Digital notes from meetings are obviously easier for analyzing sharing and archiving.
8. **Personal use:** There are some well kown apps for convering handwritten notes into digital text such as Microsoft OneNote, Evernote, and Google Keep. Handwriting input tools on smartphones and tablets that convert handwritten text into typed text as well.

1. INTRODUCTION

1.1 Optical Character Recognition Task

Optical Character Recognition (OCR) is the process of converting images of handwritten, typed, or printed text to machine-encoded text, regardless of the source (document photo, scene photo, or subtitled text) to text. The document can be obtained through scanning, capturing a photo of the document, taking a scene photo, or extracting subtitle text superimposed on an image. Common OCR algorithms have two main modules. One for detecting the text in an image and one for recognizing the text. Text detection is the process of identifying and locating all text blocks in a given image, whether they be individual words or lines of text. The task of text detection is typically regarded as an object detection problem, in which traditional models like YOLOv5 [5] and DBNet [6] can be utilized. Text recognition is a process that seeks to comprehend the meaning of a text image and convert the visual signals into tokens of natural language. For text recognition problems, the most frequent approaches are encoder-decoder architectures. At the encoder, Convolutional Neural Networks (CNNs) are extensively utilized to process the images and extract useful information. At the decoder, text generation is performed using Recurrent Neural Networks (RNNs). This thesis specifically addresses the challenge of recognizing text in handwritten document images, not only in text line images, but also in entire pages. According to the recent advancements in text recognition [7], transformer-based models can lead to significant improvements. Such methods still utilize CNNs as the backbone at the encoder, enhancing them with self-attention mechanisms. Connectionist Temporal Classification (CTC) [8] remains a popular choice for loss function. Furthermore, by adding external language models at the character level, the accuracy of the system can be improved. In spite of the high success of the encoder-decoder models, there is still a lot of space for improvements:

1. These models are trained from scratch without taking advantage of pre-trained large scale models.

2. Pre-trained image transformers that tend to be more and more popular [9], can potentially replace CNN architectures taking advantage of working with image and text transformers simultaneously in a single pipeline of text recognition problems.

1.2 Thesis Contribution

The purpose of this thesis is to explore the idea of Handwritten Optical Character Recognition (OCR). We study how to achieve robust results in OCR by employing four different approaches. Firstly, we utilize the Calamari OCR engine, which is an encoder decoder network with CNN and bidirectional LSTM. We employ the default network with the one bidirectional LSTM at the decoder and one larger network with three bidirectional LSTM

layers. Secondly, we implement an encoder decoder model combining the pre-trained image transformer encoder of the TrOCR model and bidirectional LSTM layers at the decoder. We use a large version of the encoder with more parameters and three bidirectional LSTM layers at the decoder, as well as a small version of the encoder with fewer parameters and one bidirectional LSTM decoder. Additionally, the Calamari OCR, in conjunction with a pre-trained YOLO model, has been used for the ICDAR 2024 Competition on Handwritten Text Recognition in Brazilian Essays – BRESSAY. The YOLOv5 model has been utilized for detecting lines on pages or paragraphs and Calamari OCR for recognizing text. TrOCR and Calamari OCR were evaluated on the same datasets with the Character Error Rate (CER), the Word Error Rate (WER), the recall, the F1 score, the precision, and the Mathews Correlation Coefficient (MCC) and were compared to each other. Furthermore, our proposed solution for the ICDAR competition was evaluated with the CER and WER and was compared to other participants.

1.3 Related Work

CNN models are commonly used in Text Line Recognition (TLR) tasks for feature extraction. These features are passed to an encoder-decoder network to output the predicted sequences. Recent approaches in Handwritten Text Recognition (HTR) have been discussed by Memon et al. [10]. Different attention mechanisms for Seq2Seq algorithms have been explored by Michael et al. [11] in the HTR topic. Some popular architectures that have been used for a while, consist of Recurrent Neural Network (RNN) encoders with CTC decoders. LongShort Term Memory (LSTM) is the most common choice for RNN [12, 13, 14]. Multidimensional LSTM encoders have been investigated as well [15, 16, 17]. Attention decoders have been employed extensively as Seq2Seq methods for HTR [11, 18, 19, 20, 21]. The challenges associated with scaling handwriting recognition were addressed in [22]. Apart from Calamari OCR that we employ in our current thesis, there are some OCR systems like OCRopy, OCRopus3, Tesseract4 and Kraken. These programs offer end-to-end solutions for OCR tasks, from page images to texts. The OCRopy[1] was first published in 2008 by Breuel [23] and in 2013 for the ICDAR, they dealt with printed English and Fraktur using LSTM models [24]. It was the first software where users could train custom LSTM networks with the CTC loss [8]. Numpy is the default option for the computations during training, which can be changed to C-based clstm which is faster. However, GPU or CNN-LSTM networks cannot be used. In order to train the model, a list of text images along with their corresponding ground truth text files are required. Another tool is called Kraken[2]. It has a different API from OCRopy, and it employs clstm as the default option. It is capable of handling bidirectional and top-to-bottom texts as well. Tesseract[3] was released as open source in 2005, and it is still under development. Although it supports deep neural networks like CNN-LSTM, it does not have the option of a GPU. Comparable to Calamari's network prototype language, Tesseract introduces a

[1] https://github.com/NVlabs/ocropus3
[2] http://kraken.re/
[3] https://github.com/tesseract-ocr/tesseract

Variable-size Graph Specification Language (VGSL).

1.4 Thesis Overview

This thesis is divided into 5 chapters. After this introductory chapter, the remainder has the following structure:

- In **Chapter 2**, we describe the datasets in detail and the form of its accompanying transcription files.
- In **Chapter 3**, we describe the proposed networks.
  - In **Section 3.1** we describe the TrOCR model, its components and the custom encoder decoder model that we utilized combining the TrOCR encoder and bidirectional LSTMs. We analyze the training and the testing processes as well.
  - In **Section 3.2** we describe the Calamari OCR model, its architecture and its methods for training as well as for testing.
  - In **Section 3.3** we give insights about the ICDAR 2024 Competition on Handwritten Text Recognition in Brazilian Essays – BRESSAY, and we present our approach for the competition.
- In **Chapter 4** we present the results of our implemented OCR systems, comparing and commenting on the results.
- In **Chapter 5**, we conclude the thesis by summarizing our findings. Finally, we provide some ideas for future work on the OCR task.

2. DATASET

We focus on the problem of recognizing Old Greek handwritten manuscripts and propose recognition techniques that can be applied to a large number of important historical manuscript collections that are written in lowercase and capital letters. Text is written in Greek polytonic orthography. Predictions are made at the level of letters without diacritical marks, though. We employed 3 datasets.

The first one includes handwritten pages from an 1842 Greek handwritten translation of Shakespeare's Macbeth, and a collection of handwritten Parliamentary Questions from the Historical Archive (1974-1977) of the Greek Parliament. There are 175 handwritten pages for training and validation and 50 for testing. Having the ground truth coordinates of the lines inside the pages, we extracted 4133 text lines for training and 1034 text lines for validation. With the same strategy, we extracted 1443 text lines for testing. Figures 2.1 and 2.3 show some examples of this dataset. Each text line image corresponds to one text file with the context. Training and validation datasets contain 238 possible characters.

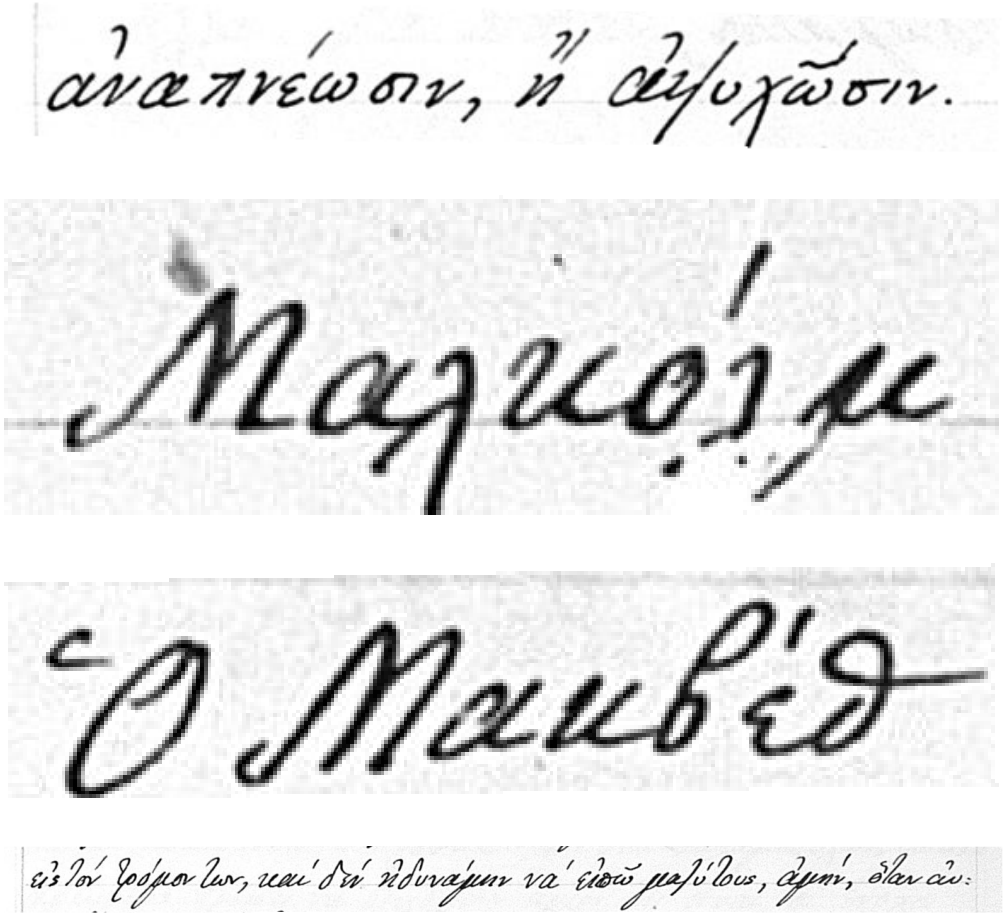

Figure 2.1: Handwritten line image samples from an 1842 Greek handwritten translation of Shakespeare's Macbeth.

Figure 2.2: Handwritten line image samples of Parliamentary Questions from the Historical Archive (1974-1977).

The second dataset has been split into text line images from 284 full pages ending up with 6881 text line images and texts for training, 1727 for validation. It contains page images and ground truth texts from an 1842 Greek handwritten translation of Shakespeare's Macbeth, a collection of handwritten Parliamentary Questions from the Historical Archive (1974-1977) of the Greek Parliament, and Proceedings of the General Board of the Bank of Greece 2.3. Training and validation datasets contain 156 possible characters.

The third dataset contains 14010 text line images for training, 3512 for validation, which have been extracted from 493 full pages. All these pages are from the same 3 sources as in the second dataset. Each image again corresponds to a ground truth text file with the real context. Training and validation datasets contain 153 possible characters.

Our models were trained on the first, the second and the third datasets separately. The models that were trained on the first dataset were evaluated on the first dataset test partition. The models that were trained on the second and the third dataset were evaluated on the same test dataset that consists of 1300 text line images from the same 3 sources.

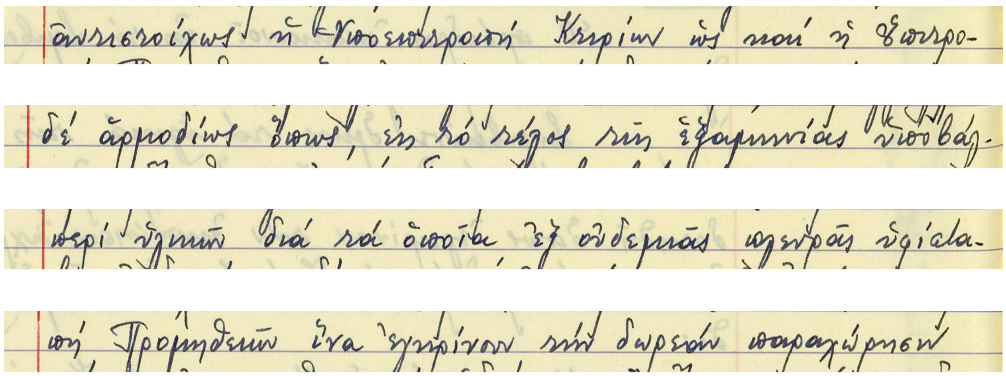

Figure 2.3: Handwritten line image samples from the Bank of Greece.

3. METHODOLOGY

3.1 TrOCR

3.1.1 Introduction

TrOcr is an end-to-end OCR model with transformers. It includes pre-trained computer vision (CV) and natural language processing (NLP) models. Its architecture is depicted in Figure 3.1.

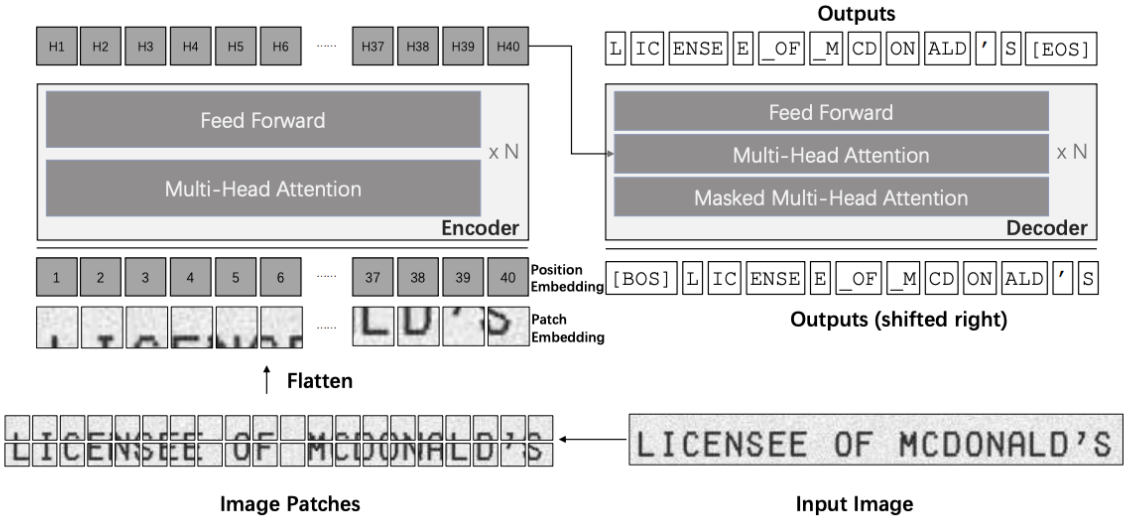

Figure 3.1: TrOCR architecture. The encoder-decoder network consists of a pre-trained image transformer and a pre-trained text transformer, respectively. Image from [1].

TrOCR is a state-of-the-art text recognition model that differs from existing models by not utilizing CNN as its backbone. Despite its simplicity, TrOCR is highly efficient and successful. Contrary to that, according to the study conducted by Dosovitskiy et al. [9], the initial step involves resizing the input text picture to dimensions of 384×384. Subsequently, the image is divided into a series of patches of 16×16, which are then utilized as the input for image transformers. The transformer architecture, which includes the self-attention mechanism, is used in both the encoder and decoder parts. The recognized text from the input image is converted into wordpiece units. In order to train the TrOCR model effectively, the encoder can be initialized with pre-trained ViT-style models (Dosovitskiy et al. [9]; Touvron et al. [25]; Bao et al. [26]), while the decoder can be initialized with pre-trained BERT-style models (Devlin et al. [27]; Liu et al. [28]; Dong et al. [29]; Wang et al. [30]). TrOCR utilizes pre-trained image transformer and text transformer models, using extensive unlabeled data for image comprehension and language modeling, without requiring an external language model. Furthermore, TrOCR does not necessitate the use of a convolutional network as its backbone, and it does not incorporate any image-specific assumptions, hence simplifying the implementation and maintenance of the model. The experiment findings

on OCR benchmark datasets demonstrate that TrOCR can reach state-of-the-art performance on printed, handwritten, and scene text image datasets without the need for any intricate pre/post-processing procedures. Additionally, TrOCR can be enhanced easily to recognize many languages, employing pre-trained models that support several languages during decoding and expanding the dictionary.

The TrOCR contribution can be summarized as follows:

1. It is a transformer-based OCR model that can be used for text recognition (printed or handwritten). It utilizes computer vision (CV) and natural language processing (NLP) models. It is worth mentioning that this is the first study that combines pre-trained image and text transformers for the text recognition task.
2. TrOCR attains cutting-edge performance using a conventional encoder-decoder model based on transformers. This model does not use convolution and does not depend on any intricate pre/post-processing procedures.
3. All the models and code can be accessed by the public at the following link: https://aka.ms/trocr.

3.1.2 Augmentations

In order to increase the diversity of the dataset, the following augmentation techniques have been applied:

- Random Rotate
- Random Brightness
- Random Sharpen
- Random Erode Dilate
- LoG Edge Detection
- Median Filter
- High Boost Filter

3.1.3 TrOCR - LSTM

TrOcr is and end-to-end model that based on transformers. It can be used for text recognition tasks, and contains pre-trained natural language processing and computer vision models. Unlike the commonly used CNN based models, TrOCR resizes the input image to 384×384 and splits it into a sequence of 16×16 patches. These patches are forwarded

to image transformers. This is a typical transformer architecture with shelf-attention mechanisms on both the encoder and the decoder. In our approach, we kept only the encoder part as a feature extractor. More specifically, we employed two different encoders, one smaller with fewer parameters called trocr-small-handwritten[1] and one larger called trocr-large-handwritten[2]. At the decoder of the small TrOCR encoder, one bidirectional LSTM has been used. At the decoder, three bidirectional LSTM layers have been used for the large TrOCR encoder. Both the bidirectional LSTM decoders are followed by a fully connected layer. These networks have been trained using the CTC loss.

3.1.3.1 Model Architecture

TrOCR is constructed using the transformer architecture, which consists of an image transformer for extracting visual data and a text transformer for language modeling. TrOCR system utilizes a transformer encoder-decoder construction. The purpose of the encoder is to extract the visual features from the image patches, while the decoder uses this data along with previous predictions to build a word-piece sequence. In the current thesis, we kept only the encoder part.

3.1.3.2 Encoder

The encoder takes an input image , which belongs to a 3-dimensional space of size $x_{img} \in \mathbb{R}^{3\times H0\times W0}$, and resizes it to a given size (H, W). The transformer encoder is unable to directly process raw images unless they are presented as a sequence of input tokens. Consequently, the encoder breaks down the input image into a batch of $N = HW/P^2$ square patches, each with a fixed size of (P, P). The width (W) and the height (H) of the resized image are ensured to be divisible by the patch size (P). Afterwards, the patches are flattened into vectors and then linearly projected into D-dimensional vectors, which are also known as patch embeddings. D represents the hidden size of the transformer over all of its layers. Like ViT [9] and DeiT [25], authors retain the special token ”[CLS]”, which typically employed for image classification tasks. The ”[CLS]” token consolidates the information from all the patch embeddings and serves as a representation of the entire image. Additionally, while utilizing the DeiT pre-trained models for encoder initialization, they retain the distillation token in the input sequence. This enables the model to acquire knowledge from the instructor model. The patch embeddings and two special tokens are assigned learnable 1D position embeddings based on their absolute positions. Unlike the characteristics derived by the CNN-like network, the transformer models lack image-specific inductive biases and handle the picture as a sequence of patches. This allows the model to easily focus its attention on either the entire image or the individual patches.

[1]https://huggingface.co/microsoft/trocr-small-handwritten
[2]https://huggingface.co/microsoft/trocr-large-handwritten

3.1.3.3 Decoder

At the decoder, we used three bidirectional LSTMs with 256 units for the large TrOCR encoder and one bidirectional for the small TrOCR encoder, with 256 units as well. Both the decoders are followed by a fully connected layer with 512 units (forward and backward pass).

Training the models with the CTC loss, we retrieve the label with the highest probability by setting y as the most probable path of labels π, which is given by:

$$p(\pi|x^n) = \prod_{t=1}^{T} \hat{y}^n_{\pi_t}, \quad \forall \pi \in L^T_\epsilon \tag{3.1}$$

$$\hat{y}^n_t = \arg\max p(\pi|x^n) \tag{3.2}$$

$$\hat{y}^n = B\left(\sum_{t=1}^{T} \hat{y}^n_t\right) \tag{3.3}$$

With L^T_ϵ being the set of all labels that have the length T and $p(\pi|x^n)$ being the probability that path $\pi \in L^T_\epsilon$ is predicted by the DNN. B is a function that removes all predicted blank labels and all repeated labels (e.g. $B(-IC-CC-V) = B(II\text{–}CCC-C\text{–}V-) = ICCV$).

$$\hat{y}^n = B\left(\sum_{t=1}^{T} \hat{y}^n_t\right) \tag{3.4}$$

3.1.3.4 Model Initialization

The encoder and decoder are both initialized using public models that have been pre-trained on extensive labeled and unlabeled datasets.

3.1.3.5 Encoder Initialization

The DeiT [25] and BEiT [26] models are employed to initialize the encoder in the TrOCR models. The image transformer model is trained using DeiT using only the ImageNet dataset [31]. The authors experiment with various hyperparameters and employ data augmentation techniques in order to enhance the efficiency of the model. Furthermore, they condense the expertise of a powerful image classifier into a condensed token within the initial embedding, resulting in a comparable outcome to the models based on Convolutional Neural Networks (CNN). According to the Masked Language Model pre-training task, BEiT introduces the Masked Image Modeling task as a pre-training challenge for the image transformer. Every image will be transformed into two perspectives: image patches

and visual tokens. They tokenize the original image into visual tokens by the latent codes of a discrete VAE [32], mask some image patches randomly, and make the model recover the original visual tokens. BEiT shares the same structure as the image transformer but does not have the condensed token that DeiT does.

3.1.3.6 Decoder Initialization

Although we haven't used the decoder of the model, we would like to mention that the RoBERTa [28] (based on BERT) and the MiniLM [30] models have been used for the decoder initialization.

3.1.3.7 Task Pipeline

In this thesis, the workflow of text recognition starts with the visual feature extraction of text line images. After that, the model predicts the wordpiece tokens depending on the image and the context generated before. Every ground truth token ends with the special token "<eos>" that indicates the "end of sequence". During the training process, we do a backward shift of the sequence by one position and introduce the "start of sequence" token ("<sos>") at the beginning to indicate the initiation of generation. The decoder takes the shifted ground truth sequence, and it is supervised trained with the original ground truth text. CTC loss has been used for training. During the inference, the decoder initiates with the "start of sentence" token ("<sos>") and predicts the output step by step, using the freshly generated output as the input for the next iteration.

3.1.3.8 Pre-training

In the pre-training phase, the text recognition task has been employed, because models can learn the knowledge of visual feature extraction as well as the language model. However, in our case, only the feature extraction has been used. The pre-training is split into two stages. A large scale dataset with hundreds of millions of printed text line images has been used in the first stage. In the second part, two smaller datasets with printed and handwritten downstream problems have been used. Each of them consists of a million text-line images. Furthermore, existing and widely adopted synthetic scene text datasets for the scene text recognition task have been utilized. On these datasets, they pre-train individual models in the second stage, being initialized by the model of the first stage.

3.1.3.9 Training Process

Models were trained with the CTC loss, which is used in tasks with variable length sequences like this. When it comes to the hyperparameters, Stohastic Gradient Descent (SGD) algorithm has been used as the optimizer. The learning rate starts with 0.001, and

it can be reduced by factor 0.1 when the validation loss is not improved with a patience of 5 epochs. The batch size was set to 2 in order to handle our memory resources.

3.1.3.10 Fine-tuning

Apart from the experiments for scene text recognition, the pre-trained TrOCR models are fine-tuned on the downstream text recognition tasks. The outputs of the TrOCR models are based on Byte Pair Encoding (BPE) [33] and SentencePiece [34], and do not depend on any task-related vocabulary.

3.1.3.11 Data Augmentation

For increasing the variety in data and improving robustness, the following augmentation techniques have been applied, keeping the original images as well.

1. Random rotation (-10 to 10 degrees)
2. Gaussian blurring
3. Image dilation
4. Image erosion
5. Downscaling
6. Underlying

3.1.3.12 Pre-training Dataset

In order to construct a comprehensive and superior dataset, the researchers extract a sample of two million document pages from PDF files that are publicly accessible on the Internet. By transforming digital-born PDF files into page images and extracting the text lines together with their cropped images, they may obtain visually appealing printed text line images. The first pre-training dataset consists of a total of 684 million text lines. They utilize a total of 5,427 handwritten fonts to create handwritten text line images. This is achieved through the usage of TRDG, an open-source text recognition data generator. The text utilized for generating is obtained by systematically extracting information from various sites of Wikipedia. The second-stage pre-training handwritten dataset comprises 17.9 million text lines, which includes the IIIT-HWS dataset [35]. Furthermore, they gather approximately 53,000 photographs of receipts from real-life scenarios and employ commercial OCR engines to accurately identify and extract the text from these images. Based on the outputs, they crop the pages into text lines using their coordinates and rectify them into normalized images. In addition, the TRDG has been used to generate one million

printed text line pictures using two receipt fonts and the pre-installed printed fonts. The printed dataset consists of 3.3 million text lines. The additional pre-training data used for scene text recognition is the MJSynth (MJ) [36]. The IAM Handwriting Database is used for handwritten English text recognition, and Aachen's partition of the dataset is used for training and testing. More specifically, they used 6,161 lines with 747 different forms for training, 966 lines from 115 forms for validation, and 2,915 lines from 336 forms for testing.

3.1.3.13 Evaluation Metrics

In the handwritten text recognition task that we are interested in, the IAM dataset was evaluated by the case-sensitive Character Error Rate (CER).

3.1.3.14 Results

Different combinations of encoder and decoder were examined. For encoders, they compared DeiT, BEiT, and ResNet-50 networks. They concluded that the BEiT encoder had the best performance. In this thesis we employed the encoder of the TrOCR large model, which has the BEiT large encoder. At the Decoder, we did not follow their solution of the RoBERTa language model, using LSTMs.

The majority of the methods in handwritten text recognition use CNN architectures in their encoders. According to their experiments, transformer-based encoders achieve similar or even better results than the traditional CNNs, using the information of the image patches. Therefore, transformer architectures are a competitive solution for visual feature extraction.

3.1.4 CTC Loss

As a loss function, we used the Connectionist Temporal Classification (CTC). The CTC is a loss function used in training of deep learning models. Particularly in speech recognition, handwriting recognition, and other sequence labeling problems where the timing varies. It does not need an aligned dataset and makes the training process more straightforward and robust. It is used as a measure of the difference between the predicted output and the actual label sequence.

3.1.5 Inference

In order to test our model, we used our test sets (section 2). We employed three different algorithms for decoding:

- Greedy search

- Beam search
- Beam search with an n-gram language model

Greedy search is the least complex algorithm among the three. It computes the probability of all the possible characters at each time step and predicts the character with the highest one. Beam search explores every possible path of characters, extending the sequence and measuring the total probability of the current sequence. The top k sequences that have the higher total probabilities are considered candidates. This algorithm ends up with the most possible sequence. Adding an n-gram language model, decoding can take into consideration some language principles that can be beneficial. For example, grammar, vocabulary, or syntax rules that each specific language has. This algorithm predicts the next word in a sequence considering the n-1 previous words. In our case, an n of 5 has been used. From the above methods, the simple beam search leads to higher accuracy. The n-gram did not improve the results, as we expected. That may be because our task aims to predict at the character level, whereas a language model focuses on the coherence between words.

3.2 Calamari OCR

3.2.1 Methods

Calamari[3] utilizes many methods to obtain state-of-the art outcomes on both modern and historical prints. In addition to supporting several DNN architectures, it also enables confidence voting for distinct predictions and allows for finetuning using codec adaptation. The following techniques will be introduced in the presentation.

3.2.1.1 Network Architecture Building Blocks

The primary function of the Deep Neural Network (DNN) and its decoder is to process the image of a segmented text line and provide a corresponding textual output. Convolutional Neural Networks (CNNs) are a branch of Deep Neural Networks (DNNs). The purpose of DNNs is to model a high degree of abstraction in the input samples using multiple-level non-linear transform architectures. A CNN is a feed forward neural network. It is a state-of-the-art solution that is generally used in image, video, and sound recognition tasks. The network consists of the following basic components:

1. Convolution layer.
2. Activation function.

[3]https://github.com/Calamari-OCR

3. Pooling-sub sampling.
4. Fully-connected (dense) layer.

Figure 3.2 shows a typical CNN model:

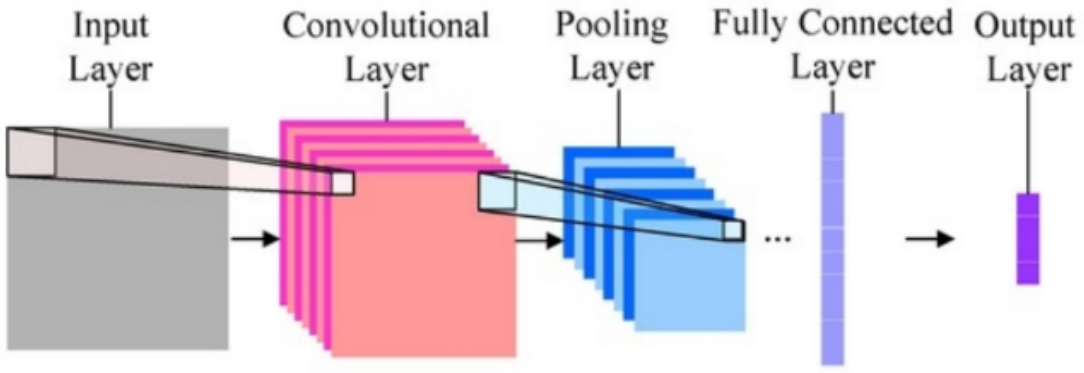

Figure 3.2: A typical Convolutional Neural Network architecture (Figure from [2]).

In CNNs, a filter is represented by a convolutional kernel that is much smaller spatially than the input. Each convolution layer consists of several filters, arranged in such a way that their output response corresponds to the same input area (which is also called the receptive field). The filters scan the input so that the receptive fields consist of overlapping areas of the input in order to extract a smoother representation of the inserted image sample (Figure 3.3). This method is called weight sharing.

Each convolution layer is followed by an activation function. Some of the most common activation functions in neural networks are the following:

1. Binary step
2. Sigmoid
3. Identity
4. Relu

This sequence-to-sequence problem is trained using the CTC algorithm, which was published by Graves [8]. The technique enables the prediction of shorter, albeit arbitrary, label sequences from an input sequence. In this case, the input sequence is a line image treated as a sequence. The network generates a probability distribution for every character in the alphabet for each horizontal pixel position on the line. Therefore, a picture with dimensions $h \times w$ and an alphabet size of $|L|$ will produce a matrix of type $(x, l) \in \mathbb{R}^w \times |L|$, where $(x, l) \in \mathbb{R}^w \times |L|$, represents a probability distribution for every x. Due to the network's requirement of observing multiple slices in width to provide accuracy in predicting

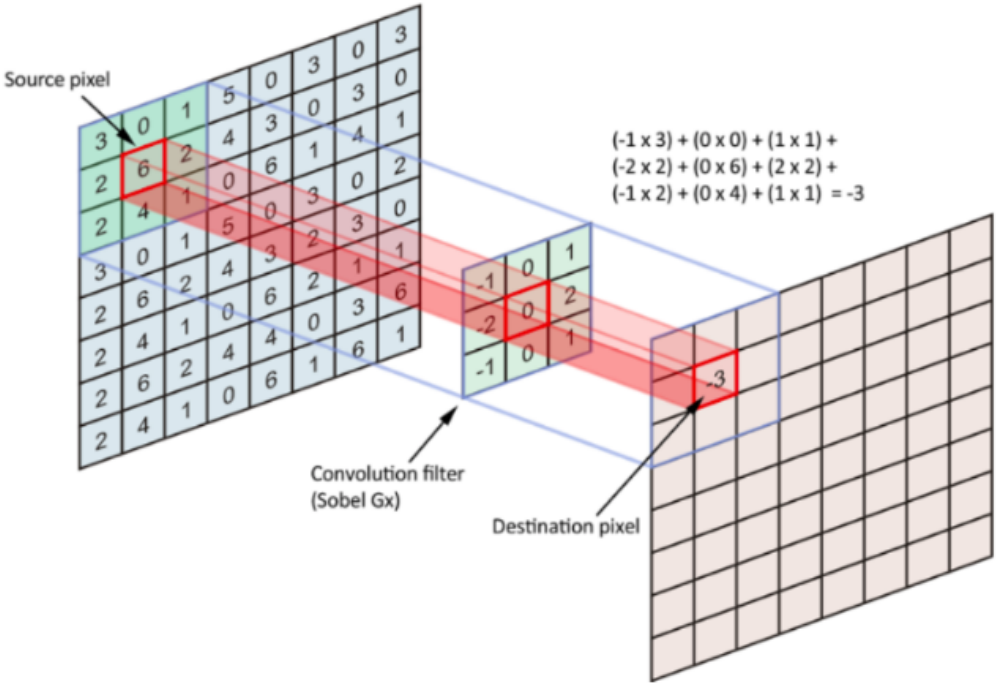

Figure 3.3: Convolution Operation (Figure from [3]).

a single character, it sometimes lacks the necessary information to make accurate predictions. Therefore, the CTC algorithm includes an additional label in the alphabet that is disregarded by the decoder. This label enables the network to provide empty or uncertain predictions with a high likelihood. Indeed, the used greedy decoder primarily selects the character with the highest probability at each position x, resulting in a tendency to forecast blank labels. Only when the character has a width of one or two pixels, can the decoder accurately recognize it. Subsequently, the ultimate decoding outcome is obtained by consolidating adjacent forecasts of identical characters and eliminating all empty labels. For instance, the sentence ”AA–B–CAA-” of length w is condensed to ”ABCAA”.

The network is trained using the CTC loss function, which calculates the likelihood of the ground truth label sequence based on the network’s probability output. The probability is calculated by adding up all potential routes through the probability matrix P that result in the ground truth (GT), using an effective forward-backward procedure. Calamari supports CNN-LSTM-Hybrid networks, which are capable of processing an entire line in one step. The features are passed to a bidirectional LSTM recurrent network to calculate the probability matrix P. Max Pooling is a prevalent method used in CNN to decrease the computational cost and retain only the most significant information. The pooling layer is used to reduce the number of parameters and consequently the computations of the network. It also decreases the possibility of overfitting. The most common functions of this layer are Max Pooling and Average Pooling.

Therefore, it is crucial for the last layer of the CNN in the whole network to have an image width that is sufficiently large to generate the entire series of labels. For instance, if the

ground truth (GT) consists of 40 characters, a minimum of 80 predictions is necessary to accommodate a blank prediction between any two consecutive characters. Therefore, in order to include two 2×2 pooling layers in the CNN, the width of the image lines must be a minimum of $w = 2 \cdot 2 \cdot 2 \cdot 80$px $= 320$px.

3.2.1.2 Finetuning and Codec Resizing

A common strategy for increasing the accuracy of a model on a certain dataset is to take an existing pre-trained model that has been used on this task and train it on your custom dataset [37]. That is the fine-tuning approach. However, the vocabulary of the custom dataset is different from the dataset that has been used for training. Therefore, the output layer must be replaced. In the case of OCR tasks, many letters, numbers, or punctuation might be the same, and only a few of them need to be added. For instance, German umlauts when starting from an English model are erased, e.g., the character ”@,” which does not exist in historical manuscripts. It is reasonable to keep those weights as is and add or remove new or unneeded labels instead of training the output layer from scratch. In the domain of historical printed books, a distinct model for each book must be trained to achieve robust results for OCR. To reduce the human labor required for manually transcribing ground truth lines, the OCR model should be trained using as few lines as possible. On the other hand, when employing only a limited number of lines, certain characters, such as capital letters or numerals, may be missing. Therefore, a whitelist functions to explicitly indicate which characters should be retained and not deleted from the underlying model. Thus, the resulting model maintains the capacity to forecast specific characters, even if they were not observed during the fine-tuning procedure. One additional benefit of using a pre-trained model is the reduced training time. Since the initial weights are not chosen arbitrarily, but instead assigned to important traits that are predicted to be applicable to new data, only slight modifications are required to improve performance on the new data.

3.2.1.3 Voting

In order to increase robustness, voting the outputs of several models is a beneficial method. The impact of voting is significantly influenced by the variance of the individual voters. In cases where the voters predict comparable results, the likelihood of removing errors decreases. On the other hand, models with higher variance were more capable of removing errors. In OCR tasks, confidence voting can lead to higher overall accuracy. This voting technique considers not only the most probable character but also takes into account alternatives with their corresponding probabilities.

A simple example of confidence voting is shown in Figure 3.4.

There are three different voters, possible characters have been predicted with an individual confidence. Choosing a single voter, the most probable character would be an “I” for a majority vote. However, when summing up the confidences among the three voters, the

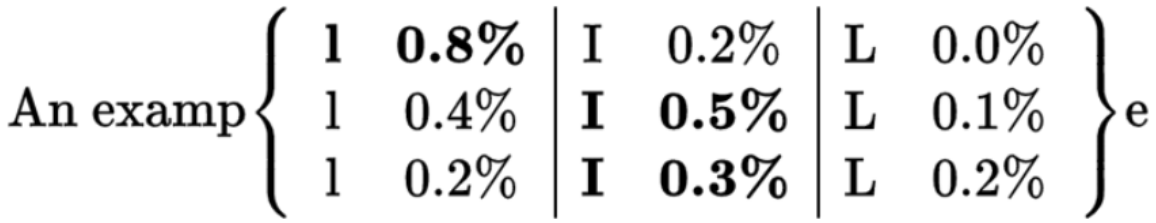

Figure 3.4: Example of confidence voting algorithm (Figure from [4]).

most probable character is an “l”. To come up with different voters for a dataset, several approaches can be applied:

- Using different training datasets
- Changing network architectures
- Employing models for finetuning

It has been proven that a straightforward and robust method called cross-fold training can generate variable voters [38].

3.2.2 The Calamari OCR-System

The Code of Calamari is written in Python3 with Tensorflow for neural networks. For decreasing the time for training and testing, Calamari supports the usage of GPUs with highly optimized CuDNN kernels, which are provided by NVIDIA. In addition, there are easy instructions for users about the available methods, networks, and parameters that can be used.

3.2.2.1 Preprocessing

Calamari preprocesses both the lines and the text for all the experiments. The line images are converted to grayscale and are then proportionally resized to a fixed height of 48 pixels. Optionally, the lines can be dewarped using OCRopy’s dewarping algorithm. Predicting the initial and final pixels within a line might be challenging for bidirectional LSTMs. This, can be considered as transient behavior in the internal LSTM state. Consequently, 16-pixel white padding is applied to both sides of the line. Textual preprocessing, such as, translating Roman unicode digits to Latin letters or merging repeated white space, can solve many visual uncertainties. Additionally, support for mixed left-to-right text is included. This overcomes a troublesome task known as mirrored symbols. For example, when opening or closing brackets rely on reading order that can alter in a line.

3.2.2.2 Training

The default model consists of two convolutional layers. The first one has 64 filters, and the second has 128. Each filter has a size of 3×3. The convolution is performed using zero padding, a stride of 1×1 and ReLU activation function. Each convolutional layer is followed by a max pooling layer with a pool size of 2×2. Forwards and backwards LSTM layers have 200 hidden units, respectively, that are concatenated and are used as input for the final output layer. In order to prevent overfitting, dropout in 50% of the neurons has been applied to the concatenated LSTM. Apart from the default model, we utilized a network by adding two additional LSTM layers with 200 and 100 nodes each, and a dropout of 0.5 each. The augmentation parameter of the Calamari-OCR engine was set to 5. The CTC algorithm has been used for computing the loss during training. Regarding the optimizer, Adam [39] has been selected with a learning rate of 0.001. To overcome the problem of exploding gradients of the LSTMs, users can apply gradient clipping to the global norm of all gradients as suggested in [40]. However, we did not use this feature, because we did not have exploding gradient issues. A list of images with text lines and their corresponding text files are expected as inputs to Calamari for training. During training, the whole dataset is loaded and remained in memory. This is more efficient than reading only the current training sample iteratively.

3.2.2.3 Prediction

To test our trained models on unseen line images, at least one model is required for predictions. If more than one model is used, then Calamari votes among the results of each of them to generate the final output sequence. In some cases, it might be beneficial to obtain extra information about the predicted output. Consequently, Calamari enables the generation of information related to the position and the confidence of every single character and the input full probability distribution. On our test datasets (sections 2, 3.3.6), we employed the recognition engine described in [41, 42] that is based on the open-source, TensorFlow-based Calamari-OCR engine [43].

3.3 ICDAR 2024 Competition

3.3.1 Overview

This competition relies on the implementation of innovative solutions for the Handwritten Text Recognition (HTR) problem. The specific competition in which we take part has to do with the transcription and analysis of essays that are written in Brazilian Portuguese. More specifically, we participated in Handwritten Text Recognition in Brazilian Essays – BRESSAY competition of the 18th International Conference on Document Analysis and Recognition (ICDAR 2024) [44]. A total of 14 contestants from different countries participated, of which 4 teams including ours (Demokritos) submitted their solutions. A total of

11 proposals for the 3 challenges were submitted. The participants achieved great results by implementing innovative algorithms outperforming baseline methods. Some key strategies include preprocessing techniques, synthetic data and advanced deep learning models.

3.3.2 Significance of this Task

Writing essays plays a vital role in education in Brazil. Higher education is achieved through the National High School Exam for Brazilian students. Therefore, the ability of fast and efficient processing of essays is crucial. Robust and accurate HTR systems can have a significant impact. However, recognizing handwritten context may be tough due the variety of styles, erasures and inconsistencies.

3.3.3 Our Challenge

This competition provides an extremely diverse dataset with different writing styles, varying paper textures, and ink qualities, giving the participants the opportunity to implement end-to-end HTR systems for real problems. The final goal is to transcribe these essays.

3.3.4 Impact

Since there is still a gap in the array of handwritten text recognition, especially in the Brazilian Portuguese, participants may contribute to an improvement in the educational technology and help modify the future of academic exams.

3.3.5 Challenges

The purpose of the competition is to get an essay image as input and output the text. There are 3 challenges regarding the level of complexity: line, paragraph, and page recognition.

1. Line-Level

    a) The images contain a single text line of an essay. The model should recognize each line taking into consideration the different writing styles and inconsistencies.

2. Paragraph-Level

    a) The model here should have the capability of recognizing breaks and alignments that exist in a paragraph, maintaining the structure and the context of the text.

3. Page-Level

    a) This is the most challenging task, because the algorithm should deal with variable margins, spacing between paragraphs apart from the text itself.

3.3.5.1 1st Challenge: Line-Level Recognition

The first challenge focuses on the implementation of models to recognize and transcribe text line images from handwritten texts. Algorithms should be able to deal with variable writing styles and inconsistencies. The purpose is to accurately transform line images into text. Figure 3.5 shows a sample text line image.

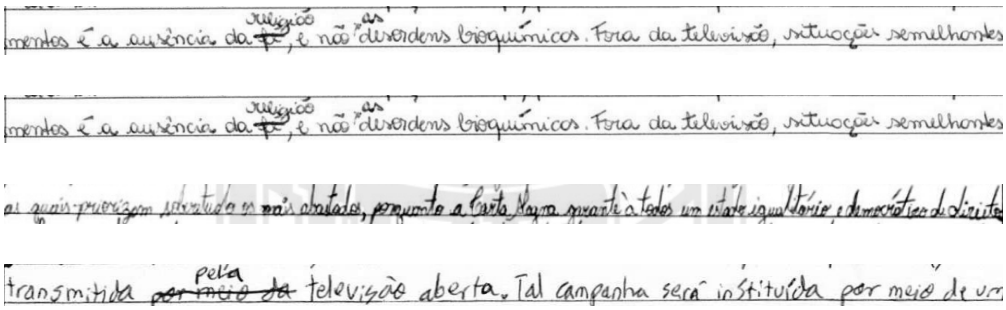

Figure 3.5: Handwritten line image samples.

3.3.5.2 2nd Challenge: Paragraph-Level Recognition

The second challenge is increasing the level of complexity. It requires models that must deal with entire paragraphs. It is crucial for algorithms to maintain the integrity of the text and to capture the context and the special characteristics that define a paragraph. Figure 3.6 shows relative paragraph images.

3.3.5.3 3rd Challenge: Page-Level Recognition

In this task, models should be able to process and transcribe entire pages of handwritten texts. Different margins, spacing, and the overall text arrangement should be considered to produce an accurate transcription. A thorough understanding of the context is vital as well. The above reasons, make this task the most challenging among the three. Figure 3.7 depicts some page images.

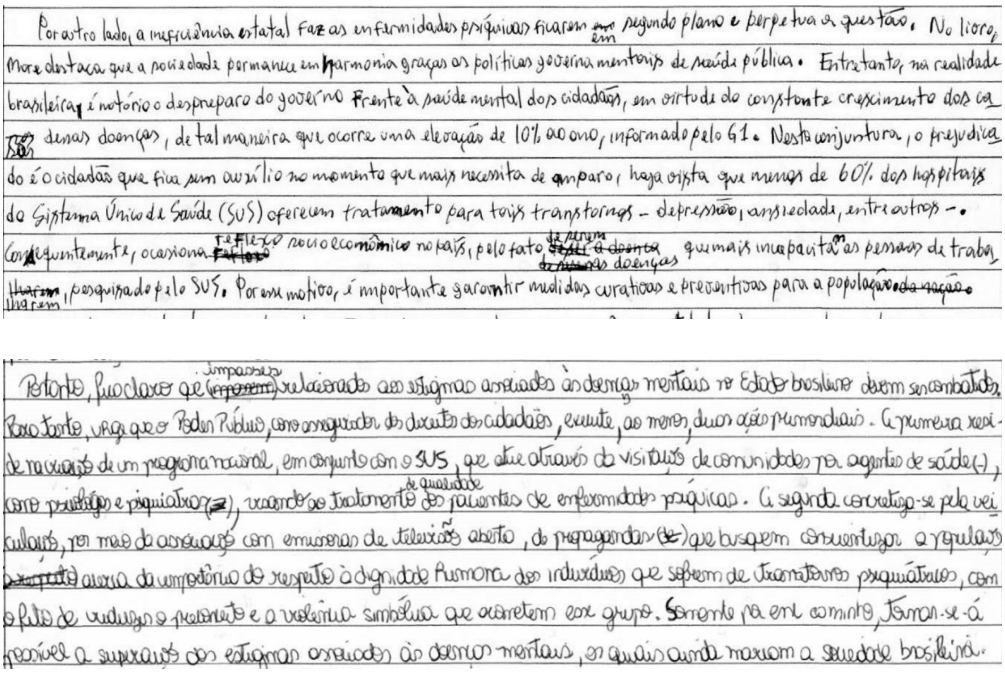

Figure 3.6: Handwritten paragraph image samples.

Figure 3.7: Handwritten page image samples.

3.3.6 Dataset Overview and Composition

The BRESSAY dataset [45] comprises images of handwritten essays in Brazilian Portuguese, providing a diverse challenge for the optical models. The images span a range of resolutions, from 450×500 to 2100×2400, requiring competitors to create automatic and adaptive solutions. Furthermore, the dataset incorporates real-world challenges, written by different people (one per page) in a limited time about different themes. Therefore, since the writer was subjected to some constraints when writing his/her essay, it is common to find in this dataset several challenges, such as hard readable words, connect words, noises, overwriting, and strike-through texts. The dataset includes a thousand scanned images of Brazilian Portuguese essays. Each page image captures unique handwriting styles, paper textures, and writing qualities, reflecting real-world challenges like erasures, overwriting, and striking-through texts. The dataset is organized to support different levels of transcription analysis, including lines, paragraphs, and full pages. This comprehensive collection is designed as a resource for testing and developing optical character recognition models. The dataset presents its content in PNG images, paired with corresponding plain text transcriptions in TXT files. Organized into ”lines,” ”pages,” and ”paragraphs” within the main `data/` directory, the dataset offers a structured approach for both focused and comprehensive algorithm testing. Each recognition level includes its respective image and transcription, allowing for versatile algorithm application. Furthermore, the dataset is divided into training, validation, and test partitions, to facilitate model training and evaluation. Within the `sets/` directory are partition files: `test.txt`, `validation.txt`, and `training.txt`. These files list page image names for each set, with each name representing a page and all its related content (lines and paragraphs) grouped in the corresponding partition. It is worth mentioning that the annotations in the TXT files encompass various real-world handwriting complexities, including overwritten text, subscripted text, and different forms of illegible or crossed-out text. In this case, we have special annotations in the ground truth and should be considered in the evaluations. Annotations used in the dataset:

- `##@@???@@##`: Unidentifiable superscript text. Superscript text that has become unidentifiable and unreadable.
- `$$@@???@@$$`: Unidentifiable subscript text. Subscript text that has become unidentifiable and unreadable.
- `@@???@@`: Unidentifiable text. Text that cannot be read or identified due to its illegibility.
- `##-xxx-##`: Superscript and illegibly crossed out. Text that has been added as a superscript and subsequently crossed out, rendering it illegible.
- `$$-xxx-$$`: Subscript and illegibly crossed out. Text that has been added as a subscript and subsequently crossed out, rendering it illegible.

- `-xxx-`: Unreadable strikethrough. Text that has been crossed out in a way that makes it unreadable.
- `##-text-##`: Superscript and legibly crossed out. Text that has been added as a superscript and subsequently crossed out, but remains legible.
- `$$-text-$$`: Subscript and legibly crossed out. Text that has been added as a subscript and subsequently crossed out, but remains legible.
- `##text##`: Superscript text in the line. Text added as a superscript in the line, typically as a correction or additional note.
- `$$text$$`: Subscript text in the line. Text added as a subscript in the line, typically as a correction or additional note.
- `-text-`: Readable strikethrough. Text that has been crossed out but remains readable.

3.3.7 Our Approach

First of all, we manually extracted the text lines from the given pages (images and texts). Then, we transformed the images into CLAHE (Contrast Limited Adaptive Histogram Equalization) in order to enhance the contrast between the text and the background. This will be beneficial for the text line detection phase. Apart from the first challenge of line level recognition, YOLOv5 model has been employed for the text line detection part. YOLOv5 has been trained on a large corpus, different than the given dataset. After that, we trained the Calamari OCR on the text line images and their corresponding ground truth texts. We inspired by the [46]. Steps for extracting the predictions for lines (section 3.3.5.1), paragraphs (section 3.3.5.2), and pages (section 3.3.5.3):

1. Transform images into CLAHE in order to enhance the contrast between the text and the background.
2. Detect the lines from pages for tasks 2 and 3 with the YOLOv5 network.
3. Crop page images to lines, regarding the extracted predicted coordinates for the previous step.
4. Predict the text with the Calamari OCR.
5. Concatenate the lines into paragraphs or pages for tasks 2 and 3, respectively.

3.3.8 BRESSAY Participants

- **LITIS (Rouen, France)** - The LITIS team from the University of Rouen Normandya and INSA Rouen Normandyb participated in all the 3 tasks. They utilized the Document Attention Network for Information Extraction and Labeling (DANIEL) [47]. In the encoder they used the Document Attention Network (DAN) [48, 49] and the transformer-based model DOcumeNt Understanding Transformer (DONUT) in the decoder [50]. To adapt the model to Portuguese they augmented the training data with synthetic images generated with articles from Portuguese Wikipedia. They also applied an upsampling method based on the Hybrid Attention Transformer (HAT) [51] to increase the resolution in low-resolution images. The DANIEL model was employed in all the 3 challenges with minor adaptations.

- **Lule˚a, (Sweden)** - The LTU team from the Lule˚a University of Technology (Lule˚a Tekniska Universitet) participated in the first task of competition (line-level recognition). Their method utilizes the seq2seq architecture [52, 53] with Convolutional Recurrent Neural Network for feature extraction in the encoder [54] with variations to adapt to smaller and larger text lines. Multiple decoders were used for text and tags as well as an additional step combining the predictions. In addition, they introduced a complementary approach in which the most accurate predictions from 22 models are selected under the assumption that their errors average out to zero and taking into account a random bias centered around zero. The output that has the smallest overall edit distance compared to all other predictions is identified.

- **PERO (Brno, Czech Republic)** - The PERO team from the Faculty of Information Technology, Brno University of Technology participated in all the 3 challenges. Its method is based on the pero-ocr pipeline and includes text line detection, text line recognition and correction with language model [55]. For text line detection they based on the ParseNet [56]. For text recognition they utilized a CRNN neural network [26] with the VGG16 in the encoder and LSTM in the decoder [57]. The language model, an LSTM with 2 layers me 2048 units in each layer was trained on the Brazilian Portuguese Carolina Carolina Corpus using the ”University” set. In addition, they employed the beam search algorithm for decoding. Finally, they converted the special tags into unique characters during training and returned them to the original form during inference.

4. EVALUATION

4.1 Evaluation Metrics

This thesis utilizes standard metrics to gauge the effectiveness of Handwritten Text Recognition (HTR) algorithms. Key metrics include the Character Error Rate (CER), the Word Error Rate (WER), the recall, the F1 score, the precision, and the Mathews Correlation Coefficient (MCC). These metrics are essential for measuring the robustness of a machine learning model because they evaluate the accuracy from multiple aspects.

Character Error Rate (CER) measures the percentage of characters that incorrectly predicted. It is calculated as:

$$\text{CER} = \frac{\text{S} + \text{D} + \text{I}}{\text{N}}$$

where:

- **S** is the number of substitutions.
- **D** is the number of deletions.
- **I** is the number of insertions.
- **N** is the total number of characters in the reference.

Word Error Rate (WER) measures the percentage of words that incorrectly predicted. It is calculated as:

$$\text{WER} = \frac{\text{S} + \text{D} + \text{I}}{\text{N}}$$

where:

- **S** is the number of substitutions.
- **D** is the number of deletions.
- **I** is the number of insertions.
- **N** is the total number of words in the reference.

Precision measures the accuracy of the detected characters. It is calculated as:

$$\text{Precision} = \frac{\text{Correct matches}}{\text{The number of the detected characters}}$$

where:

- **Correct matches** is the count of characters correctly recognized by the OCR system.
- **Number of detected characters** is the total number of characters detected by the OCR system.

Recall measures how often the OCR model correctly identifies positive instances (true positives) from all the actual positive samples in the dataset. It is calculated as:

$$\text{Recall} = \frac{\text{Correct Matches}}{\text{Number of the ground truth Characters}}$$

where:

- **Correct Matches** is the count of characters correctly recognized by the OCR system.
- **Number of the ground truth Characters** is the total number of characters that exist in the reference text.

The F1 score combines precision and recall into a single metric. It describes the harmonic mean of the precision and recall of a classification model. It is calculated as:

$$\text{F1 score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where:

- **Precision** and **Recall** are as defined above.

The Matthews Correlation Coefficient (MCC) for multi-label classification is expressed using the following formula:

$$\text{MCC} = \frac{c \times s - \sum_{k}^{K} (P_k \times t_k)}{\sqrt{\left(s^2 - \sum_{k=1}^{K} P_k^2\right) \times \left(s^2 - \sum_{k}^{K} t_k^2\right)}}$$

where:

- **k** is the number of class 1 to K.
- **s** is the number of samples.
- **c** is the number of samples correctly predicted.
- $\mathbf{t}_k$ is the number of times class k truly occurred.
- $\mathbf{p}_k$ is the number of times class k was predicted.

4.2 Results

Tables 4.1, 4.2, 4.3, 4.4, 4.5, and 4.6 present the results of the three different decoding algorithms that we have employed (greedy search, beam search, beam search with n-gram) in our TrOCR-LSTM small model.

Table 4.1: CER for Different Decoding Strategies using the TrOCR-LSTM Small Model

Decoding	**Macbeth Translation - Parliamentary Questions**	**Macbeth Translation - Parliamentary Questions - Bank Proceedings (284)**	**Macbeth Translation - Parliamentary Questions - Bank Proceedings (493)**
Greedy search	28.11%	42.09%	6.65%
Beam search	27.95%	41.95%	6.59%
Beam search - n-gram	29.48%	44.11%	8.06%

Regarding the model that was trained on our first dataset, the beam search algorithm achieved the best accuracy with a CER of 27.95%, the greedy search algorithm achieved a CER of 28.11%, and the beam search with an n-gram language model had the worst performance, attaining a CER of 29.48%.

For the model that was trained on our second dataset, the beam search algorithm achieved the best accuracy with a CER of 41.95%, the greedy search algorithm was slightly worse, achieving a CER of 42.09%, and the beam search with an n-gram language model had the worst performance, achieving a CER of 44.11%.

Using the model that was trained on our third dataset, the beam search algorithm continued to have the best performance with a CER of 6.59%, the greedy search was second, achieving a CER of 6.65%, and the beam search with an n-gram language model had again the poorest results with a CER of 8.06%.

Table 4.2: WER for Different Decoding Strategies using the TrOCR-LSTM Small Model

Decoding Strategy	**Macbeth Translation - Parliamentary Questions**	**Macbeth Translation - Bank Proceedings (284)**	**Macbeth Translation - Bank Proceedings (493)**
Greedy Search	69.08%	82.66%	25.53%
Beam Search	68.88%	82.65%	25.36%
Beam Search - n-gram	70.38%	83.99%	27.52%

Concerning the model that was trained on our first dataset, the beam search algorithm achieved the best accuracy with a WER of 68.88%, the greedy search algorithm achieved

a WER of 69.08%, and the beam search with an n-gram language model had the worst performance, attaining a WER of 70.38%.

For the model that was trained on our second dataset, the beam search algorithm achieved again the best accuracy with a WER of 82.65%, slightly better than the greedy search algorithm, which achieved a WER of 82.66%. The beam search with an n-gram language model had the worst performance, attaining a WER of 83.99%.

Employing the model that was trained on our third dataset, the beam search algorithm continued to have the best performance with a WER of 25.36%, the greedy search had the second-best performance, achieving a WER of 25.53%, and the beam search with an n-gram language model had again the poorest results with a WER of 27.52%.

Table 4.3: Precision for Different Decoding Strategies using the TrOCR-LSTM Small Model

Decoding Strategy	**Macbeth Translation - Parliamentary Questions**	**Macbeth Translation - Bank Proceedings (284)**	**Macbeth Translation - Bank Proceedings (493)**
Greedy Search	0.8463	0.8381	0.9648
Beam Search	0.8471	0.8397	0.9649
Beam Search - n-gram	0.8497	0.8386	0.9605

The model we trained on our first dataset had the highest precision of 0.8497 for the beam search with an n-gram language model, a slightly lower precision of 0.8471 for the beam search algorithm, and the lowest precision of 0.8463 for the greedy search.

When the TrOCR-LSTM small model was trained on our second dataset, the beam search algorithm achieved the highest precision of 0.8397; the beam search with an n-gram language model achieved a slightly lower precision of 0.8386. Finally, the greedy search yielded a precision of 0.8381.

Utilizing the model that was trained on our third dataset, the beam search achieved the highest precision with 0.9649, followed closely by the beam search algorithm with a precision of 0.9648. The beam search with an n-gram language model had a precision of 0.9605.

The beam search algorithm achieved a recall of 0.8040 on our first dataset, the greedy search algorithm yielded a recall of 0.8039, and the beam search with an n-gram language model achieved a recall of 0.7795.

For the model that was trained on our second dataset, the greedy search achieved the highest rate with a recall of 0.7638; the beam search was slightly worse, achieving a recall of 0.7637. The beam search with an n-gram language model was the least accurate, having a recall of 0.7451.

When it comes to the model that was trained on our third dataset, the greedy search and the beam search had the same score, achieving the highest rate with a recall of 0.9609.

Table 4.4: Recall for Different Decoding Strategies using the TrOCR-LSTM Small Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	0.8039	0.7638	0.9609
Beam Search	0.8040	0.7637	0.9609
Beam Search - n-gram	0.7795	0.7451	0.9478

Finally, the beam search with an n-gram language model was the least accurate, having a recall of 0.9478.

Table 4.5: F1 score for Different Decoding Strategies using the TrOCR-LSTM Small Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	0.8221	0.7958	0.9622
Beam Search	0.8226	0.7964	0.9622
Beam Search - n-gram	0.8088	0.7848	0.9532

Regarding the model that was trained on our first dataset, the beam search algorithm achieved an F1 score of 0.8226, which was the highest rate; the greedy search was slightly worse with an F1 score of 0.8221. Finally, the beam search with an n-gram language model achieved an F1 score of 0.8088.

The beam search had an F1 score of 0.7964, while the greedy search came in second with an F1 score of 0.7958, and the beam search with an n-gram language model had the lowest accuracy with an F1 score of 0.7848.

Finally, the greedy and beam search algorithms on our third dataset achieved the highest rates, with an F1 score of 0.9622, while the beam search with an n-gram language model was very close, with an F1 score of 0.9532.

Concerning the model that was trained on our first dataset, the beam search algorithms achieved an MCC of 0.8235, which was the highest rate. The greedy search had the second-best MCC, at 0.8229. The beam search with an n-gram language model came in third, achieving an MCC of 0.8168.

With an MCC of 0.7738 for the beam search, the greedy search came in second with an MCC of 0.7731, and the beam search with an n-gram language model came in third with an MCC of 0.7633 for the model that we trained on our second dataset.

Table 4.6: Matthews Correlation Coefficient (MCC) for Different Decoding Strategies using the TrOCR-LSTM Small Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	0.8229	0.7731	0.9586
Beam Search	0.8235	0.7738	0.9588
Beam Search - n-gram	0.8168	0.7633	0.9522

Finally, the beam search achieved the highest MCC of 0.9588 using the model we trained on our third dataset. The greedy search had a slightly worse MCC of 0.9586. The beam search with an n-gram language model yielded an MCC of 0.9522.

Tables 4.7, 4.8, 4.9, 4.10, and 4.11, 4.12 show the results of the three different decoding algorithms that we have employed (greedy search, beam search, beam search with n-gram) in our TrOCR-LSTM large model. The model was trained on our three different datasets, and the greedy search, the beam search, the beam search with n-gram language model, were employed as decoding methods.

Table 4.7: Character Error Rate (CER) for Different Decoding Strategies using the TrOCR-LSTM Large Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	6.72%	6.60%	3.77%
Beam Search	6.69%	6.55%	3.76%
Beam Search - n-gram	6.84%	6.75%	3.88%

With respect to the model that was trained on our first dataset, the beam search algorithm achieved the best accuracy with a CER of 6.69%, the greedy search algorithm achieved a CER of 6.72%, and the beam search with an n-gram language model had the worst performance, attaining a CER of 6.84%.

For the model that was trained on our second dataset, the beam search algorithm achieved again the best accuracy with a CER of 6.55%, the greedy search algorithm was slightly worse, achieving a CER of 6.60%, and the beam search with an n-gram language model had the worst performance, attaining a CER of 6.75%.

Using the model that was trained on our third dataset, the beam search algorithm continued to have the best performance with a CER of 3.76%, the greedy search algorithm was

really close, achieving a CER of 3.77%, and the beam search with an n-gram language model had again the poorest results with a CER of 3.88%.

Table 4.8: Word Error Rate (WER) for Different Decoding Strategies using the TrOCR-LSTM Large Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	29.43%	27.91%	16.69%
Beam Search	29.31%	27.74%	16.64%
Beam Search - n-gram	29.68%	27.87%	17.04%

For the model that was trained on our first dataset, the beam search algorithm achieved the best accuracy with a WER of 29.31%, the greedy search algorithm achieved a WER of 29.43%, and the beam search with an n-gram language model had the worst performance, attaining a WER of 29.68%.

With regard to the model that was trained on our second dataset, the beam search algorithm achieved again the best accuracy with a WER of 27.74%, the beam search with an n-gram language model was second, achieving a WER of 27.87%, and the greedy search was the least accurate, attaining a WER of 27.91%.

Using the model that was trained on our third dataset, the beam search algorithm continued to have the best performance with a WER of 16.64%, the greedy search algorithm was slightly worse, achieving a WER of 16.69%, and the beam search with an n-gram language model was third with a WER of 17.04%.

Table 4.9: Precision for Different Decoding Strategies using the TrOCR-LSTM Large Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	0.9451	0.9663	0.9778
Beam Search	0.9451	0.9670	0.9780
Beam Search - n-gram	0.9431	0.9676	0.9790

In terms of the model that was trained on our first dataset, the beam search algorithm and the greedy search achieved the same precision of 0.9451, while the beam search with an n-gram language model yielded a precision of 0.9431.

With a precision of 0.9676, the beam search with the n-gram model achieved the highest rate for our second dataset, followed by the beam search with a precision of 0.9670, and

the greedy search with a precision of 0.9663, which was the least accurate.

Utilizing the model we trained on our third dataset, the beam search with the n-gram algorithm maintained its top performance with a precision of 0.9790, followed by the beam search algorithm with a precision of 0.9780, and the greedy search with a precision of 0.9778.

Table 4.10: Recall for Different Decoding Strategies using the TrOCR-LSTM Large Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	0.9386	0.9622	0.9777
Beam Search	0.9386	0.9619	0.9776
Beam Search - n-gram	0.9332	0.9579	0.9760

The beam search algorithm and the greedy search, when trained on our first dataset, both achieved a recall of 0.9386, while the beam search with an n-gram language model achieved a recall of 0.9332.

For the model that was trained on our second dataset, the greedy search achieved the highest rate with a recall of 0.9622, the beam search was slightly worse, achieving a recall of 0.9619, and the beam search with an n-gram language model was the least accurate, having a recall of 0.9579.

When it came to the model that was trained on our third dataset, the greedy search achieved the highest rate with a recall of 0.9777, the beam search was slightly worse, with a recall of 0.9776, and the beam search with an n-gram language model was the least accurate, with a recall of 0.9760.

Table 4.11: F1 score for Different Decoding Strategies using the TrOCR-LSTM Large Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	0.9411	0.9636	0.9774
Beam Search	0.9412	0.9639	0.9774
Beam Search - n-gram	0.9387	0.9619	0.9771

In terms of the model that was trained on our first dataset, the beam search algorithm achieved an F1 score of 0.9412, which was the highest rate; the greedy search was slightly worse with an F1 score of 0.9411. Finally, the beam search with an n-gram language model achieved an F1 score of 0.9387.

With respect to the model that was trained on our second dataset, the beam search achieved the highest rate with an F1 score of 0.9639, the greedy search was second, achieving an F1 score of 0.9639, and the beam search with an n-gram language model was the least accurate, having an F1 score of 0.9619.

Finally, when the TrOCR-LSTM large model trained on our third dataset, the greedy and beam search algorithms achieved the highest rates with an F1 score of 0.9774, while the beam search with an n-gram language model was very close, with an F1 score of 0.9771.

Table 4.12: Matthews Correlation Coefficient (MCC) for Different Decoding Strategies using the TrOCR-LSTM Large Model

Decoding Strategy	Macbeth Translation - Parliamentary Questions	Macbeth Translation - Bank Proceedings (284)	Macbeth Translation - Bank Proceedings (493)
Greedy Search	0.9392	0.9582	0.9762
Beam Search	0.9392	0.9584	0.9762
Beam Search - n-gram	0.9391	0.9579	0.9756

When TrOCR-LSTM large model trained on our first dataset, the beam search and greedy search algorithms achieved an MCC of 0.9392, the highest rate; the beam search with an n-gram language model achieved an MCC of 0.9391, which was almost identical to the other algorithms. In our second training dataset, the beam search yielded the highest rate with an MCC of 0.9584, followed by the greedy search with an MCC of 0.9582, and the beam search using an n-gram language model with an MCC of 0.9579, which was the least accurate. Finally, using the model that was trained on our third dataset, the beam search and the greedy search algorithms achieved an MCC of 0.9762, which was the highest rate; the beam search with an n-gram language model yielded an MCC of 0.9756.

The above extensive analysis indicates that the simple beam search generally outperforms the other decoding methods in the majority of the metrics and for both the TrOCR-LSTM models (small and large). The beam search with an n-gram model performs the worst, possibly due to overfitting or inappropriate constraints for this particular handwritten OCR task. In addition, the large TrOCR-LSTM model consistently surpassed the small TrOCR-LSTM model in every metric across all the decoding methods, suggesting that the TrOCR-LSTM model is more robust. After concluding that the beam search algorithm is the most effective, we proceed to use it for the predictions of our TrOCR-LSTM models, aiming to compare them with the Calamari OCR models.

Tables 4.13, 4.14, 4.15 present our results for the TrOCR-LSTM and the Calamari OCR models. The TrOCR-LSTM small model refers to the model with the small TrOCR encoder and the one bidirectional LSTM layer. The TrOCR-LSTM large model refers to the model with the large TrOCR encoder and the three bidirectional LSTM layers. The Calamari OCR large model refers to the model with the one bidirectional LSTM layer. The Calamari OCR

large model refers to the model with the three bidirectional LSTM layers. It's important to note that we haven't utilized Calamari OCR's voting feature.

Table 4.13: OCR Results Comparison - Macbeth Translation - Parliamentary Questions

OCR System	CER	WER	Precision	Recall	F1 Score	MCC
Calamari OCR small	11.11%	40.40%	0.9206	0.8932	0.9049	0.9175
Calamari OCR large	7.40%	28.60%	0.9306	0.9248	0.9269	0.9293
TrOCR-LSTM small	27.95%	68.88%	0.8471	0.8040	0.8226	0.8235
TrOCR-LSTM large	6.69%	29.31%	0.9451	0.9386	0.9412	0.9392

Table 4.14: OCR Results Comparison - Macbeth Translation - Bank Proceedings (284)

OCR System	CER	WER	Precision	Recall	F1 Score	MCC
Calamari OCR small	13.47%	46.97%	0.9363	0.9146	0.9236	0.9193
Calamari OCR large	7.30%	25.81%	0.9552	0.9499	0.9517	0.9478
TrOCR-LSTM small	41.95%	82.65%	0.8397	0.7637	0.7964	0.7738
TrOCR-LSTM large	6.55%	27.74%	0.9670	0.9619	0.9639	0.9584

Table 4.15: OCR Results Comparison - Macbeth Translation - Bank Proceedings (493)

OCR System	CER	WER	Precision	Recall	F1 Score	MCC
Calamari OCR small	7.94%	29.00%	0.9663	0.9508	0.9576	0.9528
Calamari OCR large	3.97%	15.82%	0.9762	0.9725	0.9740	0.9709
TrOCR-LSTM small	6.59%	25.36%	0.9649	0.9609	0.9622	0.9588
TrOCR-LSTM large	3.76%	16.64%	0.9780	0.9776	0.9774	0.9762

In terms of the first dataset (table 4.13), the TrOCR-LSTM large model achieved the highest recognition rates at character level recognition, attaining a CER of 6.69%, a precision of 0.9451, a recall of 0.9386, an F1 score of 0.9412, and an MCC of 0.9392. According to the WER metric, the Calamari OCR model demonstrated the best accuracy with a WER of 28,60%. The second-best performance was achieved by the Calamari OCR model at the character level recognition rates with a CER of 7.40%, a precision of 0.9306, a recall of 0.9248, an F1 score of 0.9269, and an MCC of 0.9293. At word-level recognition, the TrOCR-LSTM large model yielded the second-best rates with a WER of 29,31%. The third-best performance, across all the metrics, was achieved by the Calamari OCR small network that yielded a CER of 11.11%, a WER of 40,40%, a precision of 0.9206, a recall of 0.8932, an F1 score of 0.9049, and an MCC of 0.9175. Finally, the TrOCR-LSTM small model had the lowest accuracy in the first dataset, achieving a CER of 27.95%, a WER of 68.88%, a precision of 0.8471, a recall of 0.8040, an F1 score of 0.8226, and an MCC of 0.8235.

Regarding the second dataset (table 4.14), the TrOCR-LSTM large model had the most accurate results at character level recognition, achieving a CER of 6.55%, a precision of 0.9670, a recall of 0.9619, an F1 score of 0.9639, and an MCC of 0.9584. According to the WER metric, the Calamari OCR model demonstrated the best accuracy with a WER of 25,81%. The Calamari OCR model had the second-best rates at character level recognition, with a CER of 7.30%, a precision of 0.9552, a recall of 0.9499, an F1 score of 0.9517, and an MCC of 0.9478. At word level recognition, the TrOCR-LSTM large model was second with a WER of 27,74%. The Calamari OCR small network had again the third-best accuracy for both the character and the word level recognition rates, achieving a CER of 13.47%, a WER of 46,97%, a precision of 0.9363, a recall of 0.9146, an F1 score of 0.9236, and an MCC of 0.9193. Finally, the TrOCR-LSTM small model did not perform well in the second dataset, achieving a CER of 41,95%, a WER of 82,65%, a precision of 0,8397, a recall of 0,7637, an F1 score of 0,7964, and an MCC of 0,7738.

When it comes to the third dataset (table 4.15), the TrOCR-LSTM large model performed the best at character level recognition rates, achieving a CER of 3.76%, a precision of 0.9780, a recall of 0.9776, an F1 score of 0.9774, and an MCC of 0.9762. According to the WER metric, the Calamari OCR large model yielded the lowest error rate with a WER of 15,82%. The Calamari OCR model was second at character level recognition metrics with a CER of 3.97%, a precision of 0.9762, a recall of 0.9725, an F1 score of 0.9740, and an MCC of 0.9709. At word-level recognition, the TrOCR-LSTM large model achieved the second-best results with a WER of 16,64%. The TrOCR-LSTM small model was third in both the character and the word level recognition rates achieving a CER of 6.59%, a WER of 25,36%, a precision of 0.9649, a recall of 0.9609, an F1 score of 0.9622, and an MCC of 0.9588. Finally, the Calamari OCR small model achieved a CER of 7,94%, a WER of 29,00%, a precision of 0,9663, a recall of 0,9508, an F1 score of 0,9576, and an MCC of 0,9528.

Generally, it seems that the larger the dataset, the higher the accuracy. The TrOCR-LSTM large model outperforms Calamari OCR at character-level recognition rates. On the other hand, the Calamari OCR large model surpassed the others at word-level recognition. Both the large models indicated superior performance in recognizing words and characters, taking advantage of the three bidirectional LSTM layers at their decoders. The usage of pre-trained models is also beneficial for faster convergence and higher accuracy, considering prior knowledge from large datasets of handwritten texts. The TrOCR-LSTM models are significantly larger and more complex than Calamari OCR models and require more time for training and greater memory usage. For example, the third dataset, which is the largest one, requires approximately eight days to be trained with the TrOCR-LSTM large model, while the Calamari OCR large model needs one.

4.3 ICDAR

The results of all the participants for all the 3 challenges of the ICDAR 2024 Competition on Handwritten Text Recognition in Brazilian Essays—BRESSAY (presented in section

3.3.5) are following in sections 4.3.1, 4.3.2, and 4.3.3. Teams were flexible to choose the tasks they would participate in. Thus, the teams that participated in the competition are as follows: four teams in the first challenge (line-level recognition), three teams in the second challenge (paragraph-level recognition), and three teams in the third challenge (page-level recognition). Our team participated in all three challenges. In addition, three baseline solutions were provided in order to have some initial benchmarks; Bluche [12], Flor [58], and Puigcerver [59], are traditional methods in the HTR domain. Models were evaluated with CER and WER metrics. In case of a tie, WER prevailed. The accuracy in recognizing the special tags of the given dataset was also measured. Finally, execution times and GPU memory usage were reported to complete the analysis.

4.3.1 Line-Level Recognition Results

Table 4.16: ICDAR Line-Recognition results

Metric	Value
Character Error Rate (CER)	7.89%
Word Error Rate (WER)	25.01%
CER (Case Insensitive)	7.78%
WER (Case Insensitive)	24.66%
CER (No Punctuation)	7.53%
WER (No Punctuation)	23.72%
CER (No Accents)	7.63%
WER (No Accents)	24.16%
CER (No Special Tags)	7.81%
WER (No Special Tags)	24.75%

Baseline/Team	CER	WER	CERp	WERp	CERa	WERa	CERs	WERs
Bluche	18.02%	48.55%	18.16%	47.55%	17.72%	47.99%	20.03%	49.42%
Flor	10.26%	31.58%	10.43%	30.44%	10.08%	31.12%	12.49%	31.59%
Puigcerver	8.95%	25.06%	9.01%	23.90%	8.81%	24.74%	11.22%	25.44%
Demokritos	8.21%	25.12%	7.98%	23.77%	7.93%	24.24%	8.95%	24.96%
LITIS	4.62%	11.91%	4.90%	10.97%	4.53%	11.67%	5.91%	11.53%
LTU (ensemble)	3.20%	10.12%	2.89%	8.67%	3.12%	9.81%	4.71%	9.47%
LTU (main)	3.35%	10.51%	2.86%	8.93%	3.27%	10.18%	3.62%	9.67%
PERO	2.88%	9.39%	3.26%	8.67%	2.81%	9.09%	5.86%	10.02%

Table 4.17: Line-Level Recognition Results – Without Punctuation (p), Without Accents (a), and Without Special Tags (s)

As far as the line-level recognition is concerned, the PERO team had the best performance with a CER of 2,88% and a WER of 9,39%. In terms of time and GPU memory allocation, they needed 0.43 seconds on average to process each item and used 1 GB of the GPU

Special Tag	Demokritos	LITIS	LTU (ens.)	LTU (main)	PERO	Total
##@@???@@##	0	0	0	0	0	2
$$-xxx-$$	0	0	1	0	0	3
##-xxx-##	0	0	3	7	1	10
@@???@@	1	50	23	31	2	55
-xxx-	104	230	180	192	83	246
##text##	48	51	346	350	248	387
-text-	254	533	572	540	254	577

Table 4.18: Count of Special Tags Recognized in the Line Test Partition.

memory. The LTU had the second-best accuracy in the recognition metrics with the main and the ensemble models. The ensemble model achieved a CER of 3.20% and a WER of 10.12% outperforming the main model, which had a CER of 3.35% and a WER of 10.51%. The main model required an average of 0.04 seconds per item, and the ensemble 1.06 seconds per item. Both models allocated 5 GB of the GPU memory. The LITIS was third in error rates with a CER of 4.62% and a WER of 11.91%. On average, their model took 0.41 seconds to process one item using 5 GB of GPU memory. Finally, our team achieved a CER of 8.21% and a WER of 25.12%. The test of our solution was made on a CPU, so access times and GPU memory usage were not measured. Tables 4.16 and 4.17 present our results and the results of all the participants, respectively. The results show that the PERO team was better in CER and WER. However, because they were not very successful at recognizing special tags, their accuracy in CER_s and WER_s was reduced. On the other hand, the other groups had more balanced results across all measures. Table 4.18 shows the number of recognized special tags in the line test partition.

4.3.2 Paragraph-Level Recognition Results

Table 4.19: ICDAR Paragraph-Recognition results

Metric	Value
Character Error Rate (CER)	7.28%
Word Error Rate (WER)	21.73%
CER (Case Insensitive)	7.18%
WER (Case Insensitive)	21.46%
CER (No Punctuation)	6.75%
WER (No Punctuation)	20.49%
CER (No Accents)	7.16%
WER (No Accents)	21.38%
CER (No Special Tags)	6.97%
WER (No Special Tags)	21.30%

Team	CER	WER	CERp	WERp	CERa	WERa	CERs	WERs
Demokritos	8.42%	22.72%	7.59%	21.20%	8.28%	22.39%	7.86%	22.20%
LITIS	6.63%	14.48%	6.35%	13.42%	6.51%	14.24%	6.53%	13.82%
PERO	3.75%	10.48%	3.40%	9.14%	3.68%	10.18%	3.60%	10.10%

Table 4.20: Paragraph-Level Recognition Results – Without Punctuation (p), Without Accents (a), and Without Special Tags (s).

Special Tag	Demokritos	LITIS	PERO	Total
##@@???@@##	0	0	0	2
$$-xxx-$$	0	0	0	3
##-xxx-##	0	0	1	10
@@???@@	1	55	0	55
-xxx-	89	196	97	246
##text##	47	37	260	387
-text-	317	507	268	577

Table 4.21: Count of Special Tags Recognized in the Paragraph Test Partition.

The paragraph recognition challenge involves recognizing entire paragraphs. In this challenge, 3 teams, including ours, proposed solutions. The PERO team achieved the lowest error rates, having a CER of 3,75% and a WER of 10,48%. Their method processed each item in an average of 3.63 seconds and used approximately 1 GB of GPU memory. Second was the LITIS team, achieving a CER of 6.63% and a WER of 14.48%. Their method processed each item in an average of 2.21 seconds and used about 6 GB of GPU memory. We had the third-best results with a CER of 8.42% and a WER of 22.72%. We made the predictions on a CPU, as stated in section 4.3.1; therefore, computational costs were not evaluated. The results are in tables 4.19 and 4.20. Regarding the special tags, the LITIS team demonstrated a higher level of accuracy, especially in crossout texts and unreadable terms. All groups showed more balanced performance across all the metrics, improving their accuracy compared to the line-level recognition task. Table 4.21 presents the number of recognized special tags in the paragraph test partition.

4.3.3 Page-Level Recognition Results

The page recognition challenge involves recognizing entire pages. In this challenge, three teams, including ours, proposed solutions. The PERO team achieved the lowest error rates, having a CER of 3,77% and a WER of 10,08%. Their method processed each item in an average of 18.09 seconds and used approximately 1 GB of GPU memory. The LITIS team had the second-best performance, achieving a CER of 8.80% and a WER of 17.66%. Their method processed each item in an average of 6.63 seconds and used about 7 GB of GPU memory. Third was us, having a CER of 6.62% and a WER of 19.49%. We made the predictions on a CPU, as stated in section 4.3.1; therefore, computational costs were not evaluated. The results are in tables 4.22 and 4.23. In special tags, the

Table 4.22: ICDAR Page-Recognition results

Metric	Value
Character Error Rate (CER)	5.74%
Word Error Rate (WER)	18.50%
CER (Case Insensitive)	5.67%
WER (Case Insensitive)	18.24%
CER (No Punctuation)	5.30%
WER (No Punctuation)	17.39%
CER (No Accents)	5.65%
WER (No Accents)	18.16%
CER (No Special Tags)	5.39%
WER (No Special Tags)	18.03%

Team	CER	WER	CERp	WERp	CERa	WERa	CERs	WERs
Demokritos	6.62%	19.49%	5.92%	18.27%	6.51%	19.13%	6.01%	18.89%
LITIS	8.80%	17.66%	8.38%	16.53%	8.67%	17.41%	8.39%	16.85%
PERO	3.77%	10.08%	3.35%	8.86%	3.70%	9.78%	3.55%	9.64%

Table 4.23: Page-Level Recognition Results – Without Punctuation (p), Without Accents (a), and Without Special Tags (s).

Special Tag	Demokritos	LITIS	PERO	Total
##@@???@@##	0	0	0	2
$$-xxx-$$	0	0	0	3
##-xxx-##	0	0	1	10
@@???@@	0	55	1	55
-xxx-	73	189	91	246
##text##	42	47	257	387
-text-	331	526	266	577

Table 4.24: Count of Special Tags Recognized in the Page Test Partition.

LITIS group demonstrated a higher level of accuracy as in paragraph-level recognition, especially in crossout texts and unreadable terms. Table 4.24 presents the number of recognized special tags in the page test partition.

With respect to our results for the ICDAR 2024 Competition on Handwritten Text Recognition in Brazilian Essays – BRESSAY, we are satisfied being between the top 4 competitors. The accuracy in the text line recognition task (table 4.16) is competitive, regarding the challenges and the complexity of the given datasets, as we have already mentioned in 3.3.5 and 3.3.6. In addition, extracting lines from pages for tasks 2 and 3 with the YOLOv5 model was absolutely successful, leading to even lower CER and WER for paragraphs and page tasks (tables 4.19, 4.22).

5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, we explored the recognition of old Greek handwritten manuscripts utilizing the Calamari OCR system and a state-of-the-art transformer-based model (TrOCR) accompanied with bidirectional LSTM layers. Both approaches employ the encoder-decoder architecture. We implemented two different architectures for the TrOCR-LSTM model, as well as two for the Calamari OCR. According to the TrOCR-LSTM, we used a large version of the encoder with more parameters and three bidirectional LSTM layers at the decoder, as well as a small version of the encoder with fewer parameters and one bidirectional LSTM decoder. In terms of the Calamari OCR, we used two models with three bidirectional LSTM layers and one bidirectional LSTM layer at the decoders, respectively. We trained and tested each model on the same datasets, which included text line images from the General Board of the Bank of Greece, a collection of handwritten Parliamentary Questions from the Historical Archive (1974-1977) of the Greek Parliament, and an 1842 Greek handwritten translation of Shakespeare's Macbeth. The Calamari OCR engine, which employs an encoder-decoder architecture with CNN and bidirectional LSTM models, demonstrated robustness on the given datasets, taking advantage of the augmentation techniques that it offers. The TrOCR-LSTM networks comprised a pre-trained image transformer encoder and bidirectional LSTM layers at the decoder. Every model was trained with the CTC loss due to the not aligned datasets. To make predictions with the TrOCR-LSTM models, we used the beam search decoder, the greedy search decoder, and a combination of the beam search with an n-gram language model for the inference. The most accurate decoding method was the beam search. In addition, we presented our proposed solution for the ICDAR 2024 Competition on Handwritten Text Recognition in Brazilian Essays - BRESSAY, which included the YOLOv5 model for text line detection and the Calamari OCR for text line recognition. The CLAHE preprocessing, which enhances the background and text, played a crucial role in the text line detection process. We evaluated our methods using the Character Error Rate (CER), the Word Error Rate, the recall, the F1 score, the precision, and the Mathews Correlation Coefficient (MCC), and confirmed that the large TrOCR-LSTM was the best in terms of character recognition. On the other hand, the Calamari OCR with the three bidirectional LSTM layers had the best performance at word-level recognition. In addition, we achieved relatively competitive results in BRESSAY, especially considering the dataset's challenges.

5.2 Future Work

The promising results of our study open several avenues for further investigation. Considering the advancements in Natural language processing, we could add large language models at the decoder instead of Recurrent Neural Networks like LSTM. It would also be beneficial to use pre-trained language models like BERT or GPT. Such models can

be used for post-processing as well. Apart from this, implementing post-processing techniques like error correction algorithms could enhance the accuracy of the recognized text. Last but not least, we can explore more augmentation techniques, such as synthetic data generation, in order to increase the diversity of the data and the model's robustness and generalization capabilities.

ABBREVIATIONS - ACRONYMS

WER	Word Error Rate
CER	Character Error Rate
CTC	Connectionist Temporal Classification
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory (LSTM)
HTR	Handwritten Text Recognition
GT	Ground Truth
DNN	Deep Neural Network
NLP	Natural Language Processing
CV	Computer Vision
VGSL	Variable-size Graph Specification Language
MCC	Mathews Correlation Coefficient

REFERENCES

[1] M. Li, T. Lv, J. Chen, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei, "Trocr: Transformer-based optical character recognition with pre-trained models," 2022. [Online]. Available: https://arxiv.org/abs/2109.10282

[2] M. Peng, C. Wang, T. Chen, and G. Liu, "Nirfacenet: A convolutional neural network for near-infrared face identification," *Information*, vol. 7, no. 4, 2016. [Online]. Available: https://www.mdpi.com/2078-2489/7/4/61

[3] "An intuitive guide to convolutional neural networks," 2018, accessed: 2024-07-03. [Online]. Available: https://www.freecodecamp.org/news/an-intuitive-guide-to-convolutional-neural-networks-260c2de0a050/

[4] C. Wick, C. Reul, and F. Puppe, "Calamari - a high-performance tensorflow-based deep learning package for optical character recognition," 2018. [Online]. Available: https://arxiv.org/abs/1807.02004

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[6] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," 2019. [Online]. Available: https://arxiv.org/abs/1911.08947

[7] D. H. Diaz, S. Qin, R. Ingle, Y. Fujii, and A. Bissacco, "Rethinking text line recognition models," 2021. [Online]. Available: https://arxiv.org/abs/2104.07787

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 369–376. [Online]. Available: https://doi.org/10.1145/1143844.1143891

[9] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[10] J. Memon, M. Sami, R. A. Khan, and M. Uddin, "Handwritten optical character recognition (ocr): A comprehensive systematic literature review (slr)," *IEEE Access*, vol. 8, pp. 142 642–142 668, 2020.

[11] J. Michael, R. Labahn, T. Grüning, and J. Zöllner, "Evaluating sequence-to-sequence models for handwritten text recognition," 2019. [Online]. Available: https://arxiv.org/abs/1903.07377

[12] T. Bluche and R. Messina, "Gated convolutional recurrent neural networks for multilingual handwriting recognition," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, 2017, pp. 646–651.

[13] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21. Curran Associates, Inc., 2008. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf

[14] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," 2014. [Online]. Available: https://arxiv.org/abs/1312.4569

[15] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21. Curran Associates, Inc., 2008. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf

[16] P. Voigtlaender, P. Doetsch, and H. Ney, “Handwriting recognition with large multidimensional long short-term memory recurrent neural networks,” 10 2016, pp. 228–233.

[17] J. Puigcerver, “Are multidimensional recurrent layers really necessary for handwritten text recognition?” in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 01, 2017, pp. 67–72.

[18] L. Kang, P. Riba, M. Rusiñol, A. Fornés, and M. Villegas, “Pay attention to what you read: Non-recurrent handwritten text-line recognition,” 2020. [Online]. Available: https://arxiv.org/abs/2005.13044

[19] J. Poulos and R. Valle, “Attention networks for image-to-text,” 12 2017.

[20] A. Chowdhury and L. Vig, “An efficient end-to-end neural model for handwritten text recognition,” 2018. [Online]. Available: https://arxiv.org/abs/1807.07965

[21] T. Bluche, “Joint line segmentation and transcription for end-to-end handwritten paragraph recognition,” 2016. [Online]. Available: https://arxiv.org/abs/1604.08352

[22] R. R. Ingle, Y. Fujii, T. Deselaers, J. Baccash, and A. C. Popat, “A scalable handwritten text recognition system,” 2019. [Online]. Available: https://arxiv.org/abs/1904.09150

[23] T. Breuel, “The ocropus open source ocr system,” vol. 6815, 01 2008, p. 68150.

[24] A. Ul-Hasan, F. Shafait, and T. Breuel, “High-performance ocr for printed english and fraktur using lstm networks,” 08 2013.

[25] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, “Training data-efficient image transformers & distillation through attention,” 2021. [Online]. Available: https://arxiv.org/abs/2012.12877

[26] H. Bao, L. Dong, S. Piao, and F. Wei, “Beit: Bert pre-training of image transformers,” 2022. [Online]. Available: https://arxiv.org/abs/2106.08254

[27] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[28] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “Roberta: A robustly optimized bert pretraining approach,” 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[29] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, “Unified language model pre-training for natural language understanding and generation,” 2019. [Online]. Available: https://arxiv.org/abs/1905.03197

[30] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, “Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers,” 2020. [Online]. Available: https://arxiv.org/abs/2002.10957

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[32] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, “Zero-shot text-to-image generation,” 2021. [Online]. Available: https://arxiv.org/abs/2102.12092

[33] R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” 2016. [Online]. Available: https://arxiv.org/abs/1508.07909

[34] T. Kudo and J. Richardson, “Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing,” 2018. [Online]. Available: https://arxiv.org/abs/1808.06226

[35] P. Krishnan and C. V. Jawahar, “Generating synthetic data for text recognition,” 2016. [Online]. Available: https://arxiv.org/abs/1608.04224

[36] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Synthetic data and artificial neural networks for natural scene text recognition,” *ArXiv*, vol. abs/1406.2227, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:11072772

[37] C. Reul, C. Wick, U. Springmann, and F. Puppe, “Transfer learning for ocropus model training on early printed books,” 2017. [Online]. Available: https://arxiv.org/abs/1712.05586

[38] C. Reul, U. Springmann, C. Wick, and F. Puppe, “Improving ocr accuracy on early printed books by utilizing cross fold training and voting,” in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, Apr. 2018. [Online]. Available: http://dx.doi.org/10.1109/DAS.2018.30

[39] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[40] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” 2013. [Online]. Available: https://arxiv.org/abs/1211.5063

[41] P. Kaddas, K. Palaiologos, B. Gatos, V. Katsouros, and K. Christopoulou, “A system for processing and recognition of greek byzantine and post-byzantine documents,” in *Document Analysis and Recognition - ICDAR 2023: 17th International Conference, San José, CA, USA, August 21–26, 2023, Proceedings, Part IV*. Berlin, Heidelberg: Springer-Verlag, 2023, p. 366–376. [Online]. Available: https://doi.org/10.1007/978-3-031-41685-9_23

[42] P. Kaddas, B. Gatos, K. Palaiologos, K. Christopoulou, and K. Kritsis, “Text line detection and recognition of greek polytonic documents,” in *Document Analysis and Recognition – ICDAR 2023 Workshops: San José, CA, USA, August 24–26, 2023, Proceedings, Part II*. Berlin, Heidelberg: Springer-Verlag, 2023, p. 213–225. [Online]. Available: https://doi.org/10.1007/978-3-031-41501-2_15

[43] C. Wick, C. Reul, and F. Puppe, “Calamari - a high-performance tensorflow-based deep learning package for optical character recognition,” 2018. [Online]. Available: https://arxiv.org/abs/1807.02004

[44] A. Neto, B. Bezerra, S. Araujo, W. Souza, K. Alves, M. Oliveira, S. Lins, H. Hazin, P. Rocha, and A. Toselli, “Icdar 2024 competition on handwritten text recognition in brazilian essays – bressay,” in *18th International Conference on Document Analysis and Recognition (ICDAR)*. Athens, Greece: Springer, September 2024.

[45] ——, “Bressay: A brazilian portuguese dataset for offline handwritten text recognition,” in *18th International Conference on Document Analysis and Recognition (ICDAR)*. Athens, Greece: Springer, September 2024.

[46] B. Gatos, G. Louloudis, N. Stamatopoulos, G. Retsinas, G. Sfikas, A. Giotis, F. S. Liwicki, V. Papavassiliou, and V. Katsouros, *OldDocPro: Old Greek Document Recognition*, ch. Chapter 9, pp. 157–174. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9789811203244_0009

[47] T. Constum, P. Tranouez, and T. Paquet, “Daniel: A fast document attention network for information extraction and labelling of handwritten documents,” 2024. [Online]. Available: https://arxiv.org/abs/2407.09103

[48] D. Coquenet, C. Chatelain, and T. Paquet, “Dan: A segmentation-free document attention network for handwritten document recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 7, p. 8227–8243, Jul. 2023. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2023.3235826

[49] ——, “Faster DAN: Multi-target Queries with Document Positional Encoding for End-to-end Handwritten Document Recognition,” in *Document Analysis and Recognition - ICDAR 2023*, ser. Lecture Notes in Computer Science. Springer Nature Switzerland, Jan. 2023, vol. 14190, pp. 182–199. [Online]. Available: https://hal.science/hal-03957156

[50] G. Kim, T. Hong, M. Yim, J. Nam, J. Park, J. Yim, W. Hwang, S. Yun, D. Han, and S. Park, “Ocr-free document understanding transformer,” 2022. [Online]. Available: https://arxiv.org/abs/2111.15664

[51] X. Chen, X. Wang, W. Zhang, X. Kong, Y. Qiao, J. Zhou, and C. Dong, “Hat: Hybrid attention transformer for image restoration,” 2023. [Online]. Available: https://arxiv.org/abs/2309.05239

[52] J. Michael, R. Labahn, T. Grüning, and J. Zöllner, “Evaluating sequence-to-sequence models for handwritten text recognition,” 2019. [Online]. Available: https://arxiv.org/abs/1903.07377

[53] S. Tarride, A. Lemaitre, B. B. Coüasnon, and S. Tardivel, “A comparative study of information extraction strategies using an attention-based neural network,” in *15th IAPR International Workshop on Document Analysis Systems*, La Rochelle, France, May 2022. [Online]. Available: https://hal.science/hal-03677908

[54] G. Retsinas, G. Sfikas, B. Gatos, and C. Nikou, “Best practices for a handwritten text recognition system,” in *Document Analysis Systems: 15th IAPR International Workshop, DAS 2022, La Rochelle, France, May 22–25, 2022, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2022, p. 247–259. [Online]. Available: https://doi.org/10.1007/978-3-031-06555-2_17

[55] A. F. d. S. Neto, B. L. D. Bezerra, and A. H. Toselli, “Towards the natural language processing as spelling correction for offline handwritten text recognition systems,” *Applied Sciences*, vol. 10, no. 21, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/21/7711

[56] O. Kodym and M. Hradiš, “Page Layout Analysis System for Unconstrained Historic Documents,” J. Lladós, D. Lopresti, and S. Uchida, Eds., vol. 12822. Cham: Springer International Publishing, 2021, pp. 492–506, book Title: Document Analysis and Recognition – ICDAR 2021 Series Title: Lecture Notes in Computer Science. [Online]. Available: https://link.springer.com/10.1007/978-3-030-86331-9_32

[57] M. Kišš, K. Beneš, and M. Hradiš, *AT-ST: Self-training Adaptation Strategy for OCR in Domains with Limited Transcriptions*. Springer International Publishing, 2021, p. 463–477. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-86337-1_31

[58] A. F. d. Sousa Neto, B. Leite Dantas Bezerra, A. Hector Toselli, and E. Baptista Lima, “A robust handwritten recognition system for learning on different data restriction scenarios,” *Pattern Recogn. Lett.*, vol. 159, no. C, p. 232–238, jul 2022. [Online]. Available: https://doi.org/10.1016/j.patrec.2022.04.009

[59] J. Puigcerver, “Are multidimensional recurrent layers really necessary for handwritten text recognition?” 11 2017, pp. 67–72.