Universität Zürich[UZH]

TrOCR meets CharBERT

Masterarbeit der Wirtschaftswissenschaftliche Fakultät der Universität Zürich

eingereicht von

Yung-Hsin Chen

Matrikelnummer 20-744-322

Institut für Informatik der Universität Zürich

Prof. Dr. Martin Volk

Institut für Computerlinguistik der Universität Zürich

Supervisor: Dr. Simon Clematide, Dr. Phillip Ströbel

Abgabedatum: 20.05.2024

Abstract

test

Acknowledgement

I would like to express my deepest gratitude to all those who made it possible for me to complete this thesis. I am particularly thankful to Dr. Phillip Ströbel, my supervisor, for his invaluable guidance, patience, and support throughout this research study. His profound knowledge and insightful suggestions have been a great source of inspiration and motivation for me.

Special thanks are also due to the Department of Computational Linguistics, Dr. Simon Clematide, and Prof. Dr. Martin Volk for providing the essential facilities and equipment crucial for the completion of this thesis.

I must express my profound gratitude to my parents and my partner for their unwavering support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Finally, I would like to extend my appreciation to the staff of The ScienceCluster for their prompt responses to my requests for computational resources and their assistance with technical issues.

This thesis represents a significant milestone in my academic career, and I am deeply grateful to everyone who has made it possible.

Contents

List of Figures

List of Tables

List of Acronyms

OCR	Optical Character Recognition
LM	Language Model
LLM	Large Language Model
NLM	Noisy Language Model
MLM	Masked Language Modeling
CNN	Convolutional Neural Network
CTC	Connectionist Temporal Classification
SOTA	State of the Art
BPE	Byte-Pair Encoding
ViT	Vision Transformer
TrOCR	Transformer-based Optical Character Recognition
GRU	Gated Recurrent Units
FFNN	Feedforward Neuron Network
Bi-GRU	Bidirectional GRU
GPU	Graphics Processing Unit
CER	Character Error Rate
WER	Word Error Rate
NER	Named Entity Recognition
IR	Information Retrieval
LSTM	Long Short-Term Memory network
SOTA	State of the Art
DeiT	Data-Efficient Image Transformer
BEiT	Bidirectional Encoder Representation from Image Transformers
BERT	Bidirectional Encoder Representations from Transformers
OOV	Out of Vocabulary
GRU	Gated Recurrent Unit
NMT	Neural Machine Translation
JH	Joseph Hooker
GW	George Washington

HTR	Handwritten Text Recognition
NLP	Natural Lanugage Processing
POS	Part-of-Speech

1 Introduction and Motivation

This thesis explores the integration of Optical Character Recognition (OCR) systems with advanced language models to tackle the significant challenges associated with digitizing and processing historical texts. These texts present unique obstacles due to their varied linguistic features and physical deterioration – issues that traditional OCR technologies often struggle to manage effectively. By exploring methodologies that combine state-of-the-art (SOTA) OCR technology with robust language models, this research aims to enhance text recognition accuracy, adaptability, and utility. Here, "adaptability" specifically refers to the development of methods that allow models trained on contemporary data to adeptly handle early modern English, addressing the scarcity of historical labeling data which typically hampers OCR accuracy.

The subsequent contents delve into the current limitations of existing OCR models, highlighting how enhancements in OCR technology can significantly impact the performance of downstream tasks. This discussion underscores the importance of advancing OCR technology. Such an exploration makes a substantial contribution to the field of digital humanities, offering implications for the preservation of cultural heritage and making historical knowledge more accessible and actionable.

OCR has become a key tool for digitizing printed documents [Singh et al., 2012]. While OCR tasks for modern printed materials are typically straightforward, digitizing ancient texts or handwritten documents introduces complex challenges [Neudecker et al., 2019]. Inadequate OCR can significantly affect downstream tasks such as text classification, named entity recognition (NER), information retrieval (IR), etc, leading to poor data utility.

Kettunen et al. [2022] explores the significant impact of OCR quality on the effectiveness of IR tasks, particularly focusing on historical newspaper collections. Through an experimental setup involving 32 users who evaluated search results from historical newspapers, it was found that improved OCR quality significantly enhances the perceived usefulness of historical newspaper articles. The main findings includes 1) Higher OCR quality led to more favorable relevance assessments, with the average

evaluation score for improved OCR results being 7.94% higher than those obtained with older, lower-quality OCR; 2) Users were more likely to find documents relevant when the underlying text was optically recognized with higher accuracy; 3) Investments in better OCR methods could substantially increase the accessibility and utility of historical text archives.

Nguyen et al. [2021] also highlights how OCR errors can significantly affect downstream applications such as IR and various natural language processing (NLP) tasks. OCR inaccuracies can disrupt a wide range of NLP applications including NER, part-of-speech (POS) tagging, text summarization, and sentiment analysis. For instance, as the word error rate (WER) increases, the performance of NER tools drops significantly.

Ehrmann et al. [2023] outlines several challenges faced by NER systems when applied to historical handwritten documents including 1) Noisy Input; 2) Dynamics of Language; 3) Lack of Resources.

Noisy Input Texts derived from historical documents often suffer from quality issues due to the condition of the source material and the process of digitization. OCR and Handwritten Text Recognition (HTR) systems may introduce errors such as misrecognized characters and tokenization problems, severely impacting the accuracy of downstream tasks. The challenge is compounded by the diverse nature of noise in historical texts, ranging from ink bleeds and paper deterioration to varying typographic conventions over time.

Kurar Barakat et al. [2020] identify additional categories of noise in historical texts. Below in Figure 1, some examples of noise in historical texts are illustrated. The top row displays images showing heterogeneous character heights, ink smears, and ink bleed-through, from left to right, respectively. The bottom row illustrates challenges such as touching lines and large ascenders and descenders, from left to right.

Dynamics of Language The language used in historical documents can significantly differ from modern language due to evolutionary changes in spelling, grammar, and syntax. NER systems must cope with historical spelling variations, outdated naming conventions, and the historical context of terms, which may vary significantly from their modern counterparts. Such linguistic dynamics pose a substantial challenge to maintaining high accuracy in entity recognition and classification.

In Present-Day English, the relationships within and between phrases are primarily

Figure 1: These are examples of challenges encountered in the Recent cBAD Dataset for historical handwritten text line detection.
Image souce: Kurar Barakat et al. [2020]

indicated by the order of the words. In contrast, Old English relied more heavily on specific endings attached to words to express these relationships. As a result, Old English exhibited a far more flexible word order than its modern counterpart [Smith, 2013].

For example, in Present-Day English:

These two sentences convey distinctly different meanings, with the word order determining the roles of "the lord" and "the servant" within each sentence. In Old English, however, such clarity was not necessarily dependent on word order due to the grammatical structure that allowed more variability in sentence construction.

Here is how Sentence 1 would be translated into Old English:

In Old English, all three variations effectively convey the same meaning, despite the differing word orders. This flexibility highlights the substantial linguistic shift towards fixed word order in English, underscoring the complexities NER systems must navigate when processing historical texts.

Lack of Resources A significant scarcity of annotated corpora, language models, and other NLP resources specifically designed for historical document processing substantially hinders the development and training of models tailored to these texts. Moreover, the resources that do exist often lack standardization, adding further complexity to the development, training, and evaluation of effective models.

Consequently, Nguyen et al. [2021] suggest that post-OCR correction is a crucial approach to overcome these limitations and enhance the accuracy of digitized data. They further explore advanced post-OCR correction techniques in their survey, which includes the integration of sophisticated language models and the application of both statistical and neural machine learning models. These methods, which range from manual to semi-automatic and fully automatic approaches, significantly enhance the quality of text. This improvement in text quality is crucial for the reliability of subsequent NLP tasks, underscoring the ongoing need for enhancements in post-processing technologies.

Several post-OCR correction methods have been applied and shown significant improvements. Most of these methods function sequentially, not in an end-to-end manner. However, Kang et al. [2021] suggest that allowing backpropagation to influence both the recognizer and language model concurrently can yield better results than training them separately. This integrated approach allows the recognizer to leverage insights from its own processing as well as feedback from the LM, fostering a deeper, more holistic understanding. Furthermore, the system is engineered to evaluate the relevance of the information provided by the LM, enabling the recognizer to selectively prioritize its influence. Consequently, even if the LM and the recognizer are trained on different domains, the composite model remains effective across both. Kang et al. [2021] also argue that the LM can adapt based on the frequent errors produced by the recognizer, thereby enhancing the overall accuracy of the system.

To conclude, the challenge of accurately digitizing and understanding historical texts extends beyond mere academic interest; it is a crucial endeavor aimed at preserving cultural heritage and unlocking the rich historical value embedded within these documents. Traditional OCR systems, though effective with modern texts, frequently encounter difficulties with the unique characteristics of historical languages and scripts. This limitation not only restricts access but also limits the broader scope of digital research.

This study seeks to assess the effectiveness of integrating a recognizer with a language model, not only to enhance performance but also to enable models trained on contemporary texts to adapt to early modern English. We have selected TrOCR, a SOTA OCR model known for its advanced text recognition capabilities, and CharBERT, a variant of the BERT model with an additional character-level processing. By leveraging these sophisticated technologies, our project aims to substantially improve the accuracy and reliability of text digitization processes. This enhancement will make a wider range of historical documents more accessible and useful for

academic research, educational purposes, and public engagement. Improved OCR performance on historical documents is expected to forge new pathways for research in the digital humanities, providing scholars and researchers with more accurate tools for textual analysis, linguistic studies, and cultural investigations based on diverse historical archives.

1.1 Research Questions

In this study, we aim to investigate and address several key research questions that are pivotal to advancing the field of OCR by the integration of OCR with language models. Through this inquiry, we seek to validate new methodologies by proposing a novel model architecture. The research questions will guide the experimental design and analysis, ensuring that the outcomes are robust, insightful, and contribute significantly to the existing knowledge.

1. **Domain Adaptation Through Model Fusion:** In the Candidate Fusion paper , they claimed that fusing the recognizer and language model (LM) can make the LM adjust to the domain-specific data. Can fusing TrOCR and CharBERT achieve the same conclusion? In other words, can CharBERT adjust to historical texts even it was trained on modern texts?

2. **Character-Level Information Enhancement:** In the CharBERT paper [Ma et al., 2020], they claimed that, by using the character level information in addition to the subword level information, the problems of incomplete modelling and fragile representation can be solved. However, the results shown in the paper did not show significant performance improvement over RoBERTa (a strong baseline LM model). Is this statement valid? Can TrOCR combined with CharBERT achieve the claim?

3. **Influence of Language Model on Decoder Output:** Does TrOCR decoder change its output with the presence of the LM?

4. **Impact of Training on Common Errors:** According to Kang et al. [2021], the language model can adapt by learning from the frequent errors generated by the OCR recognizer, thereby enhancing the overall accuracy of the system. If we integrate common errors made by TrOCR into the training process of CharBERT, do the results show improvements?

1.2 Chapter Overviews

Below is an overview of the subsequent chapters. Each chapter explores distinct aspects of the research, starting from foundational studies and progressing towards integrating these findings into a more powerful model. The discussions culminate in evaluating the performance of this composite model.

Chapter 2: Background and Existing Technologies This chapter reviews recent developments in OCR and post-OCR correction tasks. It introduces two pivotal models: TrOCR [Li et al., 2023] and CharBERT [Ma et al., 2020], along with a significant technique, Candidate Fusion [Kang et al., 2021], which form the foundation of our proposed model. The chapter also discusses the benefits of incorporating glyph information into the OCR system, setting the stage for the model's architecture.

Chapter 3: Model Integration and Architecture This chapter details the integration of CharBERT, a BERT-like language model, with TrOCR, a SOTA OCR model. CharBERT will function as an additional corrector within the OCR system, forming an end-to-end post-OCR correction model. The architecture of the composite model is discussed thoroughly, emphasizing its innovative approach.

Chapter 4: Experimental Design and Methodology This chapter details the experimental framework designed to benchmark and assess the adaptability of the model. It specifically evaluates the impact of incorporating glyph information on overall performance and investigates the model's capacity to process texts from diverse domains. The experimental setups are strategically crafted to address the research questions, ensuring that each test contributes directly to validating the model's effectiveness and versatility.

Chapter 5 and 6: Results, Significance, and Future Work These chapters analyze the experimental results, discuss their significance, and explore potential future research directions. The findings are expected to demonstrate the effectiveness of the composite model, and its ability to process historical texts without requiring the OCR component to be trained specifically on such texts.

1.3 Personal Contribution

My contribution to this research is centered on enhancing OCR tasks for historical texts, which are frequently constrained by a scarcity of specialized resources. I aim to develop a robust OCR model that not only outperforms SOTA OCR models in accuracy but is also capable of effectively processing historical images. A key innovation of my approach involves utilizing a language model that has been specifically trained on historical texts. This method allows the OCR model to accurately interpret historical documents without necessitating direct training on these specific types of texts.

To achieve this, I have focused on integrating advanced machine learning techniques with deep linguistic insights, creating a system that bridges the gap between contemporary computational capabilities and the nuanced demands of historical text analysis. This approach not only enhances the model's ability to understand texts with varied linguistic features and degraded physical limitations, but also significantly reduces the need for extensive annotated historical text corpora, which are often difficult and expensive to procure. This research not only advances the technical capabilities of OCR systems but also contributes to the broader field of digital humanities by improving access to valuable historical documents.

2 Related Work

This chapter explores the realm of Optical Character Recognition (OCR), tracing the evolution of this technology and the development of post-OCR correction techniques that enhance the utility of OCR outputs. A detailed exploration of TrOCR, a state-of-the-art (SOTA) OCR model, alongside CharBERT, a large language model (LLM), forms a significant portion of this chapter, with an emphasis on their architectures and training methodologies. These two models are central to this study due to their innovative integration potential. Additionally, the concept of candidate fusion is discussed, which leverages the strengths of both OCR and LMs to improve text recognition accuracy. The chapter also examines the advantages of incorporating glyph analysis into model training, further refining the recognition process. Overall, this chapter reviews the current advancements in OCR technology and sets the stage for the introduction of a novel composite model designed to combine these technologies.

2.1 Optical Character Recognition (OCR)

According to Philips and Tabrizi [2020], the digitization of historical documents involves several critical phases, each leveraging advanced technologies and methodologies to transform ancient images into accessible digital formats. The process begins with image acquisition, where physical documents are converted into digital images through scanning or photography. Following this, preprocessing steps such as binarization, using methods like Otsu's algorithm [Liu and Yu, 2009] to simplify images into binary formats, and layout analysis to determine structural elements like text blocks, are essential for preparing images for text recognition. Text recognition is then performed using technologies like OCR and Handwritten Text Recognition (HTR). Modern approaches often employ Convolutional Neural Networks (CNNs) [Zhang et al., 2017] and Long Short-Term Memory networks (LSTMs) [Breuel et al., 2013] to enhance accuracy. This comprehensive process is depicted in Figure 2.

The paper by Karthick et al. [2019] elaborates on the OCR process: 1) preprocess-

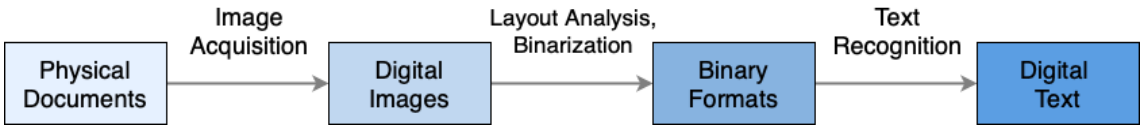


Figure 2: Digitization Process of Historical Documents

ing; 2) feature extraction; 3) recognition. Initially, during preprocessing, the image quality is enhanced by removing noise, adjusting contrast, and segmenting the image to prepare it for analysis. This stage sets the foundation for feature extraction, where unique characteristics of the text are identified, which are crucial for accurate recognition. Before feature extraction, additional steps such as further segmentation are applied to enhance the text structure in the images. Following feature extraction, the recognition stage classifies these features into textual data. This text is then optionally refined during the post-processing stage, where errors are corrected, and text clarity is improved. The process is illustrated in Figure 3.

Figure 3: Stages of the OCR Process

While traditional models typically handle the processes of preprocessing, feature extraction, and recognition in separate, discrete stages, there are modern OCR models that operate on an end-to-end basis [Huang et al., 2021][Neudecker et al., 2019][Belay et al., 2020]. These end-to-end models streamline the text recognition process by integrating all these stages into a single continuous workflow. This approach leverages advanced machine learning techniques, particularly deep learning, to directly process raw images of text into editable text outputs. This not only simplifies the OCR process by eliminating the need for manual feature engineering and separate processing stages, but also enhances the system's ability to handle varied and complex text.

SOTA OCR Models OCR technology has made significant strides in recent years. Notably, transformer models, originally developed for natural language processing (NLP) tasks, have been successfully adapted for OCR applications. Models such as TrOCR [Li et al., 2023], which combines transformers [Vaswani et al., 2017] with pre-trained models like RoBERTa [Liu et al., 2019], have demonstrated remark-

able effectiveness in extracting text from images. Additionally, the incorporation of attention mechanisms in OCR models has enabled the network to sequentially focus on specific parts of an image, mimicking the human reading process such as Attention-based Scene Text Recognizer (ASTER) [Shi et al., 2018] and STAR-Net [Liu et al., 2016]. Both of them have substantially enhanced OCR accuracy. Furthermore, the latest version of Tesseract [Smith, 2007], an open-source OCR engine, has been augmented with LSTM [Hochreiter and Schmidhuber, 1997] networks. This enhancement not only supports a greater range of languages but also improves recognition capabilities, particularly when integrated with customized pre-processing and training techniques. These advancements underscore the dynamic evolution of OCR technology, making it more versatile and effective in processing complex textual data from images.

Relevence to Our Study In our research, we aim to explore potential enhancements in OCR models. We will select a state-of-the-art (SOTA) OCR model as our baseline to evaluate whether modifications made to this model can surpass its original performance. Ideally, we will utilize an end-to-end model, eliminating the need to individually manage steps such as image processing and feature extraction.

2.2 Post-OCR correction

Post-OCR correction refers to the methods and processes used to correct errors in text after it has been converted from images (like scanned documents or photos) to editable and searchable text data. This stage is crucial because OCR technology, despite its advancements, often makes mistakes due to various challenges such as poor image quality, complex layouts, unusual fonts, or language intricacies.

Post-OCR correction is an effective tool but comes with its own set of challenges, such as distorted outputs from OCR processes or domain-specific terminology within datasets. For example, Karthikeyan et al. [2021] utilize the RoBERTa model, employing a self-supervised pre-training approach to predict masked sections of medical texts. Their goal is to bridge the gaps in OCR transcriptions, particularly when the technology fails to accurately capture parts of documents. The authors discuss and tackle the inherent challenges posed by the varying accuracy of OCR technology, especially in recognizing texts that contain medical terminologies and are often scanned from physical documents where text may be skewed or obscured.

Mokhtar et al. [2018] points out that the structured and predictable nature of OCR

errors makes them suitable for correction using machine learning models. They proposed two deep learning models: a word-based sequence-to-sequence model and a character-based model, allowing them to correct individual characters and handle words not seen during training. The findings show that while the word-based model struggles with unseen words, the character-based model performs well across different datasets. They suggest that its practical applicability in improving OCR accuracy in real-world settings. This insight aligns with the findings from Kang et al. [2021], which suggest that character-level processing can significantly improve overall performance.

SOTA Post-OCR Models Post-OCR correction models extensively utilize advanced machine learning techniques, particularly deep learning, to refine OCR outputs effectively. transformers, renowned for their sequence handling capabilities and scalability, dominate this field. Models like Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2019] and its variants, including RoBERTa [Liu et al., 2019] and GPT [Radford et al., 2018], are effectively fine-tuned for post-OCR text correction. This fine-tuning process involves training on datasets composed of erroneous OCR outputs paired with their correct versions.

Relevence to Our Study In this study, we aim to leverage post-OCR correction to address the inherent limitations of OCR technologies. Drawing from the insights gained in earlier sections, we plan to employ a BERT-like architecture that incorporates character-level processing capabilities. This approach is expected to enhance the accuracy and effectiveness of OCR by refining the outputs, particularly when dealing with complex document types and nuanced text characteristics.

2.3 TrOCR

OCR tasks typically involve text detection and text recognition. TrOCR is a transformer-based model which focuses on the text recognition part of the OCR task, converting images to texts. Therefore, the input should be sliced into line images, each with a single line of transcription on it. Unlike previous advancements in text recognition, TrOCR uses a pre-trained vision transformer (ViT) [Dosovitskiy et al., 2021] for image feature extraction and a text transformer [Vaswani et al., 2017] for sequence-to-sequence learning instead of CNN backbones [Wang et al., 2020a] and Connectionist Temporal Classification (CTC) [Graves et al., 2006]. It eliminates the usage of external LLMs and can be extended for multilingual purposes by lever-

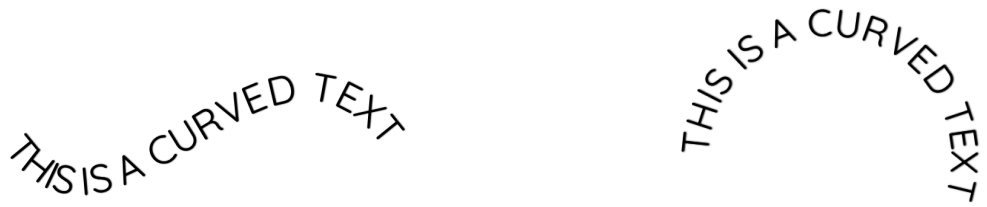
aging pre-trained LMs of different languages. In addition, unlike traditional OCR systems that rely on feature engineering and pre-/post-processing, TrOCR allows contextual learning, leading to SOTA results in challenging scenarios.

The encoder of TrOCR is initialized by pre-trained ViT-style models such as Data-Efficient Image Transformer (DeiT) [Touvron et al., 2021] or Bidirectional Encoder representation from Image Transformers (BEiT) [Bao et al., 2022], while the decoder initialization uses pre-trained BERT-style models such as RoBERTa [Liu et al., 2019] or MiniLM [Wang et al., 2020b]. Both of them are transformer encoder structures. Therefore, the missing encoder-decoder attention in the TrOCR decoder is initialized randomly during training. Due to the fixed input length of transformers, the image is first resized to 384×384, and then split into 16×16 patches before inputting into the transformer. Unlike CNNs, transformers do not have spatial information of the input data. Thus, positional encodings are added to the patches to preserve the spatial structures of the input images.

TrOCR is pre-trained on synthetic data and SROIE [Huang et al., 2019], IAM datasets [Marti and Bunke, 1999] sequentially. $\text{TrOCR}_{\text{LARGE}}$ (total parameters=558M), initialized by $\text{BEiT}_{\text{LARGE}}$ and $\text{RoBERTa}_{\text{LARGE}}$, reached a character error rate (CER) [Klakow and Peters, 2002][Wang et al., 2003] of 2.89 by pre-training on the synthetic data and the IAM dataset.

So far, TrOCR is considered a SOTA model for OCR on both printed and handwritten tasks. Although its design and capabilities represent a significant advancement in the field of OCR, deficiencies still exist. In scenarios where the text layout is curved as shown in Figure 3, TrOCR's performance is compromised. Note that these limitations are not unique to TrOCR but are rather common challenges for most OCR models. In this study, TrOCR will serve as the baseline. This foundational benchmark will be improved through integration with LMs to refine and correct the output results.

Relevence to Our Study The TrOCR handwritten model has been selected as our primary OCR model for text recognition, owing to its SOTA performance. As an end-to-end model, TrOCR simplifies the processing pipeline by eliminating the need for separate image processing and feature extraction steps. This allows it to be easily fine-tuned in conjunction with CharBERT.



(a) (b)

Figure 3: **Curved Text Layout**
TrOCR struggles with curved text recognition. For example, Figure (a) is erroneously recognized as "MASIAS ACURVED TEH" and Figure (b) as "THIS SECRET.COM" by $\text{TrOCR}_{\text{LARGE}}$ instead of the correct "THIS IS A CURVED TEXT."

2.4 CharBERT

CharBERT [Ma et al., 2020] is an enhancement of BERT, which aims to address problems in Byte-Pair Encoding (BPE) [Sennrich et al., 2016] used by pre-trained LMs like BERT and RoBERTa. It has the same model structure and configuration as BERT and RoBERTa, depending on the initialization. During inference, it takes text as input and outputs a representative embedding of the text.

Pre-trained LMs like BERT and RoBERTa have achieved outstanding results in NLP tasks. Both models use BPE to encode input data. BPE is capable of encoding almost all vocabularies, including out-of-vocabulary (OOV) words. It breaks down OOV words into subwords until they are in its vocabulary dictionary. In addition, it allows efficiency in vocabulary space. For instance, BPE can represent different forms of a word (e.g., "run", "running", "runs") with their common subwords, which results in a smaller set of total tokens. However, BPE has the problems of incomplete modeling and fragile representation.

Incomplete modeling refers to the inability of BPE to fully encapsulate the complete aspects of a word. BPE splits words into subword units. While these small pieces are helpful for understanding parts of the word, they may not entirely convey the meaning or nuances of the whole word. For instance, the word "understand", which will be broken down into "under" and "stand" by BPE, means comprehending a concept. However, its meaning is not merely a combination of "under" and "stand".

Fragile representation highlights BPE's sensitivity to minor typos in a word. In other words, a small spelling mistake can result in drastic changes in the set of subwords. Consider the word "backhand," which BPE correctly splits into "back"

and “hand,” effectively capturing the word’s meaning. However, if there is a typo resulting in “bachand” (missing “k”), BPE breaks the word down into “b”, “ach”, and “and.” This segmentation fails to convey the intended meaning and highlights the model’s vulnerability to errors.

To illustrate further, let’s examine character-level tokenization. For “backhand,” the results of the character-level tokenization would be “b,” “a,” “c,” “k,” “h,” “a,” “n,” “d.” In the case of the typo “bachand,” the tokens change slightly to “b,” “a,” “c,” “h,” “a,” “n,” “d.” Despite the typo, the character-level tokenization remains relatively consistent, showing a minor deviation from the original. This example demonstrates that while BPE can drastically misinterpret a word due to a small error, character-level tokenization maintains a closer representation to the original, even with typos. This example is illustrated in Figure 4.

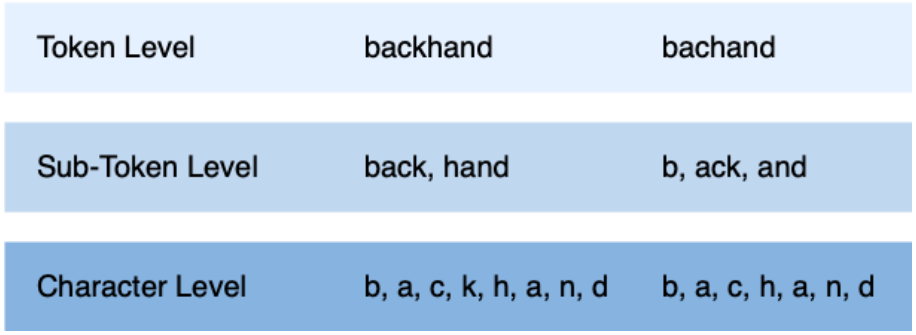


Figure 4: **Fragile Representation Example**
BPE is sensitive to typos, drastically altering subwords: “backhand” splits into “back” and “hand,” but “bachand” (missing “k”) becomes “b”, “ach”, and “and,” losing the original meaning. In contrast, character-level tokenization maintains consistency: “backhand” is tokenized as “b,” “a,” “c,” “k,” “h,” “a,” “n,” “d,” and ”bachand” slightly alters to “b,” “a,” “c,” “h,” “a,” “n,” “d.”

CharBERT aims to address these two problems with two tricks in the pre-training stage: 1) employ a dual-channel architectural approach for the subword and character; 2) utilize noisy language modeling (NLM) and masked language modeling (MLM) for unsupervised representation learning. The first trick processes both subword and character-level information and fuses them, ensuring a more robust representation in case of typos. The second trick involves introducing character-level noise into words or masking words, and trains the model to correct these errors. This approach enhances the model’s ability to handle real-world text with variations and typos.

CharBERT is fine-tuned on several downstream tasks, including question answering, text classification, and sequence labeling. CharBERT outperforms BERT across all

tasks. However, it encounters some challenging competition from RoBERTa, which the authors acknowledge as a robust baseline as shown in Figure 5.

Models	SQuAD 1.1		SQuAD 2.0		Text Classification CoLA	MRPC	QQP	QNLI
	EM	F1	EM	F1	Corr	Acc	Acc	Acc
BERT (Devlin et al., 2019)	80.5	88.5	73.7	76.3	57.4	86.7	90.6	90.7
CharBERT	**82.9**	**89.9**	**75.7**	**78.6**	**59.1**	**87.8**	**91.0**	**91.7**
RoBERTa (Liu et al., 2019)	**84.6**	**91.5**	80.5	83.7	**62.1**	90.2	91.2	92.8
XLNet (Yang et al., 2019)	-	-	80.2	-	60.2	88.2	91.4	91.7
CharBERT$_{\text{RoBERTa}}$	84.0	90.9	**81.1**	**84.5**	61.8	**90.4**	**91.6**	**93.4**

Figure 5: Experimental Results of CharBERT and Strong Pre-trained Models
Image source: Ma et al. [2020]

Although the performance of CharBERT does not significantly exceed that of RoBERTa, the model architecture of CharBERT is intriguing. Since models are usually better at the tasks they are pre-trained on, the NLM pre-training task makes CharBERT a potentially good corrector, which can be paired with the recognizer and compensate for its language deficiencies in our study. In addition, we can further pre-train the NLM with common OCR mistakes instead of randomly introduced errors. This could make the corrector more familiar with OCR-specific errors.

Relevence to Our Study CharBERT will function as the corrector in our post-OCR correction system. As outlined in this paragraph in Section 2.2, this language model meets our requirements, featuring character-level processing and a BERT-like architecture. Additionally, given its pre-training tasks, CharBERT is particularly effective at correction tasks, making it highly suitable for our post-OCR correction needs.

2.5 Candidate Fusion

Candidate fusion [Kang et al., 2021] is a technique that involves integrating the LM within the recognizer in an end-to-end manner, i.e., letting the LM and the recognizer interact. Most SOTA word recognition models and LMs are trained separately. However, with candidate fusion, several advantages can be achieved. First, the recognizer is able to integrate insights from both its own processing and the input from the LM, resulting in a more comprehensive understanding. Second, the system is designed to assess the information supplied by the LM, allowing the recognizer

to selectively weigh its importance. Lastly, the LM can learn from frequent errors generated by the recognizer, improving overall accuracy.

The model utilized for demonstrating candidate fusion have an encoder-decoder structure, where the encoder extracts features from input images, and the decoder converts the image features into text. The encoder is a CNN followed by a Gated Recurrent Unit (GRU) [Dey and Salem, 2017], and the decoder is an unidirectional multi-layered GRU. Both structures in the encoder provide spatial information about the input, allowing the decoder to follow the proper order.

In the paper, they employ two-step training. In the first step, the LM is pre-trained on a large corpus for it to understand general language and grammar. In the second step, the LM is further trained alongside handwritten datasets with the recognizer. This allows the LM to consider both its own knowledge and what the recognizer predicts when outputting the result text. The LM will adjust its predictions by taking into account the recognizer's decisions.

The model is being evaluated on IAM, GW [Fischer et al., 2012], and Rimes [Grosicki and El-Abed, 2011]. Among the datasets, the model shows significant improvement on GW over a strong baseline, Dutta et al. [2018]. However, the performance of this model beyond the GW dataset remains questionable, as the improvements are relatively small [Figure 6]. Additionally, these other datasets encompass more writers, contributing to a greater diversity in handwriting styles. Consequently, the model's marked improvement on the GW dataset might indicate potential limitations in its adaptability to diverse handwriting styles.

Despite the potential deficiency in the model, the idea of fusing the recognizer and the LM and allowing both of them to learn from each other is a quality design worth exploring. This technique provides new opportunities for achieving more context-aware text recognition. In this study, TrOCR and CharBERT will serve as the recognizer and the LM, respectively, and will be combined. The composite model aims to explore the potential enhancements or effects they can bring when used together.

Relevence to Our Study To thoroughly leverage the benefits of candidate fusion, this study will integrate TrOCR and CharBERT as the recognizer and LM, respectively, within the candidate fusion framework. Following the two-step training methodology outlined in the original research, we aim to enable both the recognizer and the language model to interact and learn from each other effectively. This integration seeks to enhance the overall effectiveness of our OCR system, particularly

Method	IAM		GW		Rimes	
	CER	WER	CER	WER	CER	WER
Mor et al. [4]	–	20.49	–	–	–	11.95
Pham et al. [43]	13.92	31.48	–	–	8.62	27.01
Bluche et al. [5]	12.60	–	–	–	–	–
Wiginton et al. [44]	6.07	19.07	–	–	3.09	11.29
Sueiras et al. [10]	8.80	23.80	–	–	4.80	15.90
Kang et al. [11]	6.88	17.45	–	–	–	–
Krishnan et al. [45]	6.34	16.19	–	–	–	–
Toledo et al. [18]	–	–	7.32	–	–	–
Dutta et al. [34][a]	**4.88**	**12.61**	4.29	12.98	**2.32**	**7.04**
Proposed	5.79	15.15	**2.82**	**7.13**	2.59	8.71

Figure 6: Experimental Results of Candidate Fusion
Image source: Kang et al. [2021]

in handling diverse and complex handwriting styles and adapt to texts from various domains.

2.6 Glyph Embedding

Integrating the concept of glyphs into OCR offers substantial advantages, especially for the digitization of historical texts. This approach enhances the accuracy of character recognition by focusing on glyphs—the visual forms of characters—which is crucial for handling the varied typographic styles found in historical documents. Glyph-based OCR is skilled at adapting to these styles and is particularly effective at recognizing characters in texts that are in poor condition, such as those with faded ink or smudges. By understanding the visual aspects of glyphs, such systems significantly reduce OCR errors, minimizing the need for labor-intensive post-correction. Additionally, since this method is based on visual characteristics rather than language, it can be applied across different languages and scripts, making it highly versatile.

Amrhein and Clematide [2018] explore how neural machine translation (NMT), when applied at the character level, can be particularly effective for correcting glyph-based errors. This approach leverages the flexibility of NMT to adapt to the nuances of character shapes and their common misinterpretations, thereby improving the accuracy of OCR text post-correction. Overall, the integration of glyph analysis into OCR workflows as discussed in this paper not only improves the initial accuracy

of text digitization but also enhances the efficiency of processing large volumes of historical documents.

Relevence to Our Study While our study primarily focuses on handwritten images, rather than the printed letters that are often the subject of glyph-based OCR research, we can adaptively incorporate glyph analysis into our approach. Specifically, we plan to integrate commonly misrecognized characters into the training process of the LM based on their occurrence frequencies. This strategy will enable the LM to become familiar with frequent OCR errors and to distinguish between similar character glyphs, thereby enhancing its ability to correct these mistakes effectively and improve the overall accuracy of our OCR system for handwritten documents.

3 Methodology

In this chapter, I will introduce the composite model of TrOCR and CharBERT by leveraging the ideas in Candidate Fusion mentioned in Chapter 2. First, the design concept behind the model will be outlined. Following this, I will elaborate on data collection and preprocessing, and the architectures of the models developed in this study. Finally, I will delve into the details of the training process, including the utilisation of GPU resources, the optimiser and the loss function.

Rewrite this

3.1 Design Concept

CharBERT mitigates the problems of incomplete modelling and fragile representation by including the character encoding in addition to the subword level information. Furthermore, having noisy language model (NLM) as the pre-training task makes CharBERT effective at correcting character level typos, which is a desired feature for OCR correcting. On the other hand, Candidate Fusion claims that having an interaction between the recognizer and the language model (LM) can enhance the performance of OCR. Thus, combining TrOCR and CharBERT is expected have an improvement on the OCR accuracy.

This synergy allows for a dynamic feedback loop where the recognizer's outputs are continuously refined through the language model's contextual corrections, improving OCR accuracy. By combining TrOCR's efficient image-to-text conversion with CharBERT's precision in error correction, the integrated system not only tackles broad structural and fine-grained typographical errors but also performs robustly across diverse document types and conditions, significantly enhancing both the reliability and accuracy of OCR outputs. This comprehensive approach leverages the strengths of both models, making the combined system exceptionally versatile and effective in various OCR scenarios.

To address the research questions (RQ), we have designed several experiments (EX), which will be discussed in detail in Chapter 4. The corresponding research questions and their associated experiments are outlined below:

RQ2 Influence of Language Model on Decoder Output: Does TrOCR decoder change its output with the presence of the language model?

EX2 Different combinations of composite model layers will be tested to optimize performance. We will examine whether the text output of this optimized composite model differs from that produced by TrOCR alone.

RQ4 Character-Level Information Enhancement: In the CharBERT paper, they claimed that, by using the character level information in addition to the subword level information, the problems of incomplete modelling and fragile representation can be solved. However, the results shown in the paper did not show significant performance improvement over RoBERTa (a strong baseline LM model). Is this statement valid? Can TrOCR combined with CharBERT achieve the claim?

EX4 The performance of TrOCR + CharBERT will be compared with TrOCR + RoBERTa to ascertain if character-level information substantively enhances model effectiveness.

RQ1 Domain Adaptation Through Model Fusion: In the Candidate Fusion paper, they claimed that fusing the recognizer and LM can make the LM adjust to the domain-specific data. Can fusing TrOCR and CharBERT achieve the same conclusion? In other words, can CharBERT adjust to historical texts even it was trained on modern texts?

EX1 We will train CharBERT with limited historical and modern English data and combine it with TrOCR. The performance of this composite model on historical English will be compared to a version trained on historical data. Due to constraints, we will use CharBERTSMALL as a baseline for comparison.

RQ3 Impact of Training on Common Errors: After integrating common errors made by TrOCR into the training process of CharBERT, do the results show improvements?

EX3 We will identify common errors from TrOCR and integrate these into CharBERT's training process to evaluate whether this enhances the composite model's performance compared to training with randomly introduced errors. As with RQ2, CharBERTSMALL will serve as a baseline for comparison.

This structured approach ensures a thorough evaluation of how well integrated OCR systems can leverage both deep learning architectures and advanced language models to improve text recognition accuracy across diverse datasets.

3.2 Data Collection and Processing

This section is dedicated to detailing the data collection process for OCR task training and $\text{CharBERT}_{\text{SMALL}}$ training. We will outline the types and sources of data harnessed for this study, emphasizing the diversity and volume of the datasets to ensure comprehensive learning. Following the data collection overview, we will delve into the processing techniques applied to the collected data.

3.2.1 Data for OCR

We focus on the performances of handwritten datasets on the composite model. The data used in this study is the George Washington (GW) handwritten dataset[1] and the Joseph Hooker (JH) handwritten dataset[2]. They serve as valuable benchmarks for developing and evaluating handwriting recognition systems. Note that to ensure comparability with existing studies, this research adheres to the established train-validation-test splits of the GW and JH datasets. Detailed information about these datasets is presented in Table 1.

Although the IAM dataset is a more contemporary collection of English handwriting samples, renowned for its diversity in handwriting styles, this study did not use it since TrOCR is pre-trained on the IAM dataset. The most significant risk of fine-tuning the composite on the same dataset is overfitting. The model may become excessively tailored to the IAM dataset, performing well on this specific dataset but poorly on unseen, real-world data or other similar datasets. In addition, if the model has already been trained to a point where it performs optimally on the IAM dataset, further training the composite model on the same data might yield diminishing returns. The model may not improve significantly and could even start to fit the noise in the dataset as mentioned above.

[1]The George Washington handwritten dataset can be downloaded here: https://fki.tic.heia-fr.ch/databases/washington-database

[2]The Joseph Hooker handwritten dataset can be downloaded here: https://github.com/jschaefer738b/JosephHookerHTR

George Washington Dataset The GW dataset is a collection of historical letters and diaries handwritten by George Washington and his secretaries in 1755. This dataset comprises various types of documents, including personal correspondence, military orders, and official communications, which provide a unique window into the early American colonial period. The dataset is frequently utilized in research focused on recognizing historical handwriting, which poses unique challenges due to the use of archaic words and phrases, and the degradation of materials over time. The different handwriting styles in the dataset, ranging from Washington's neat writing to the quicker, less tidy ones, make it very useful for developing strong handwriting recognition systems. This dataset not only supports the preservation and accessibility of these important texts but also aids in advancing the technology needed to interpret and digitize aged manuscripts. An example from the GW dataset is shown in Figure 7.

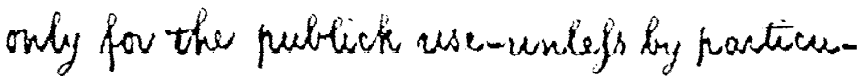


Figure 7: Sample of handwritten text from GW dataset. The excerpt reads: only for the publick use, unless by particu.

Joseph Hooker Dataset The JH dataset consists of correspondence between Joseph Dalton Hooker (1817-1911) and other leading scientists of his time. This collection offers unparalleled insights into the scientific dialogues of the era, featuring interactions with notable figures such as Charles Darwin. As such, it is an invaluable resource for historians specializing in science, botany, and the Victorian period. Hooker's letters are diverse in content, extending beyond mere botanical discussions to include observations from his travels, his perspectives on contemporary scientific debates, his personal reflections, and his professional communications. This rich variety provides a comprehensive view of his intellectual engagement and contributions. An example from the GW dataset is shown in Figure 8.

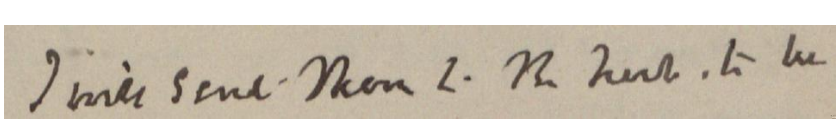


Figure 8: Sample of handwritten text from JH dataset. The excerpt reads: I will send them to the host to be.

The Joseph Hooker and George Washington datasets, while serving distinct academic purposes, both play significant roles in the realm of handwritten text recognition (HTR) tasks. The Joseph Hooker dataset, with its extensive collection of

Metric	GW Dataset	JH Dataset
Text Line	656	6916
Train Data	329	5532
Validation Data	168	691
Test Data	163	693
Unique Word instances[3]	1456	8308
Unique letters[4]	68	84
Average Text Line length[5]	40.23	28.45
Average Word length[6]	6.17	6.89
Percentage of Non-Character[7]	21%	22.4%

[3] The count of distinct words present in the dataset.
[4] The number of distinct characters present in the dataset (include punctuations).
[5] The average number of characters (letters and spaces) per text line.
[6] The average number of characters per word in the dataset.
[7] The percentage of characters in the dataset that are not part of the standard alphabet. This may include punctuation marks and other non-alphabetic characters. (characters other than A-Z, a-z, 0-9)

Table 1: Data Information

19th-century botanical writings and correspondence, provides a unique challenge for HTR technologies due to the scientific terminology and personal handwriting styles found in the documents. This variety enables developers to fine-tune HTR models to handle complex vocabulary and diverse manuscript formats, which is essential for digital humanities projects focusing on scientific archives.

Similarly, the George Washington dataset, consisting of an array of 18th-century materials including letters, diaries, and official documents, presents its own set of challenges for HTR. The historical significance of these documents demands a high level of accuracy in digital transcription. Washington's dataset helps in training HTR systems to recognize older forms of English script and the idiosyncrasies of historical American handwriting, which are crucial for preserving and making accessible key documents from the formative years of the United States.

In both cases, these datasets not only support the preservation and accessibility of

historical texts but also advance the development of HTR systems. They enable the refinement of algorithms capable of interpreting a wide range of handwritten styles and orthographic conventions, thus broadening the utility of HTR technologies across different historical and scientific domains.

Transcription Ground Truth Processing The transcription ground truths of the JH dataset are stored in XML files, where parsing these files is sufficient to retrieve the transcription texts. On the other hand, the transcription ground truths of the GW dataset follow a more complex format. In this dataset, individual characters within words are separated by hyphens (" - "), and words themselves are separated by vertical bars (" | "). Additionally, punctuation marks are represented by special characters, with a specific table detailing the replacements for each punctuation mark. The table for the punctuation replacement can be found in this link. For instance, consider the transcription from the GW dataset for the phrase "of the Virginia Regiment." This is encoded as

`o-f|t-h-e|V-i-r-g-i-n-i-a|R-e-g-i-m-e-n-t-s_pt`

, where `s_pt` is replaced with a period.

Image Processing In this study, images are resized and normalized prior to being input into the model. Resizing and normalizing images are standard preprocessing steps in image processing, particularly in the context of machine learning and computer vision. Each of these processes serves important purposes in preparing image data for models, enhancing model performance, and ensuring consistency. This process is illustrated in Figure 9.

In this study, images are initially resized to 384x384 pixels to comply with the input requirements of the pretrained TrOCR model. Resizing images is a common preprocessing step in image processing pipelines, particularly for deep learning, where input dimensions need to be consistent across all data samples for the model to process them.

Following resizing, the images undergo normalization. The normalization specifies the mean for each of the three color channels (Red, Green, Blue), which are all set to 0.5. In this case, 0.5 is subtracted from each channel. This value shifts the range of pixel values from [0, 1] to [-0.5, 0.5]. The normalization also sets the standard deviation for each color channel to 0.5. Dividing by 0.5 scales the range from [-0.5, 0.5] to [-1, 1]. This practice is common for neural network inputs as it tends to make

the training process more stable and helps it converge faster. Together, resizing and normalizing reposition the data so that its distribution is symmetric around zero, reducing the bias that input values might otherwise introduce into the network's computations. This method ensures that no single pixel range overly influences the network due to its scale, allowing the model to focus more on learning the patterns rather than adapting to the scale of input data.

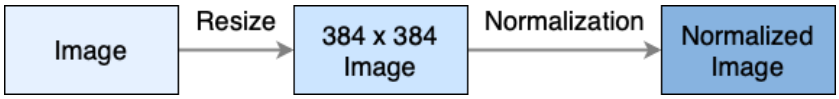


Figure 9: Flowchart illustrating the preprocessing steps for images used in the study.

3.2.2 Data for Training CharBERT$_{\text{SMALL}}$

In the original study, the authors developed CharBERT using a substantial 12GB dataset sourced from Wikipedia. This training was conducted over five days, harnessing the computational capabilities of two NVIDIA Tesla V100 GPUs. Due to constraints in time and available computational resources, our study employs a scaled-down version of CharBERT, hereafter referred to as CharBERT$_{\text{SMALL}}$. This variant has been trained on two considerably smaller datasets: 1.13GB of English Wikipedia data and 637MB of literature from the 16th to 19th centuries, respectively termed as modern CharBERT$_{\text{SMALL}}$ and historical CharBERT$_{\text{SMALL}}$.

The training process for modern CharBERT$_{\text{SMALL}}$ involved randomly sampled sentences from the Wikipedia dataset, such as "Tony Macrie has been president of the Seashore Lines since he formed the railroad in 1984." This approach provided a diverse yet manageable corpus for training, ensuring a broad representation of modern English usage. This methodological adjustment facilitates a comparative analysis between the different implementations of CharBERT: CharBERT$_{\mathcal{P}ij}$ —- which integrates common OCR errors into the model and will be detailed later —- and historical CharBERT$_{\text{SMALL}}$. Such comparisons are crucial for assessing the adaptability and effectiveness of CharBERT when trained with more constrained data resources, highlighting the model's potential flexibility and utility in diverse linguistic contexts.

3.3 Metric

Put this in methods

The metrics used in this study, character error rate (CER) and word error rate

(WER), are commonly employed to evaluate the performance of systems in tasks related to optical character recognition (OCR). Both metrics assess how accurately a system transcribes or recognizes text compared to a ground truth transcription. While CER measures accuracy at the character level, WER evaluates it at the word level.

The primary difference between CER and WER lies in their granularity. CER offers a more fine-grained evaluation, making it particularly useful for tasks where character-level precision is essential, such as OCR. WER, on the other hand, is better suited for scenarios where word-level inaccuracies significantly impact the interpretability of the transcribed text, such as in speech recognition tasks.

Another distinction is their sensitivity to errors. CER is highly sensitive to minor errors, which is critical in situations where a single character change can alter the meaning of a word dramatically. For example, changing "hat" to "hot" by substituting "a" with "o" completely changes the word's meaning. CER effectively captures these subtle yet crucial errors, proving indispensable for tasks requiring meticulous character-level accuracy.

In contrast, WER focuses on capturing errors that affect the reader's or listener's ability to understand the intended message at the word level. An illustrative example is the digitization of a restaurant menu where the original text, "Chicken with Lemon Sauce," was misinterpreted by the OCR system as "Chicken with Lemon Source." This error, which results in a WER of 20%, does not significantly diminish the readability of the item but introduces a notable semantic error. The term "Sauce," referring to a culinary liquid, is replaced by "Source," which implies origin, potentially misleading readers and affecting their understanding of the menu item. Thus, WER highlights how even minor word-level errors can impact the semantic integrity of the content.

Despite these differences, both CER and WER are crucial for benchmarking and improving models in OCR, speech recognition, and similar domains. They enable developers to identify shortcomings, compare different model architectures, and track progress over time.

3.3.1 Character Error Rate (CER)

CER is typically expressed as a percentage that represents the proportion of characters incorrectly predicted by the system. It is calculated based on the number of character-level operations required to transform the system output into the ground

truth text. These operations include insertions, deletions, and substitutions of characters.

The formula for calculating CER is as follows:

$$\text{CER} = \frac{I + D + S}{N} \tag{3.1}$$

In this equation:

- I represents the number of insertions required,
- D denotes the number of deletions,
- S signifies the number of substitutions needed to achieve a match with the ground truth,
- N is the total number of characters in the ground truth text.

This method quantifies the system's accuracy at the character level, highlighting its precision in detail.

3.3.2 Word Error Rate (WER)

WER is calculated in a manner similar to CER but focuses on the word level. It measures the number of word-level insertions, deletions, and substitutions required to transform the system's output into the ground truth.

The formula for calculating WER is as follows:

$$\text{WER} = \frac{I + D + S}{N} \tag{3.2}$$

In this equation:

- I represents the number of insertions needed,
- D denotes the number of deletions,
- S signifies the number of substitutions necessary to match the ground truth,
- N s the total number of words in the ground truth text.

This calculation provides a quantitative measure of the system's performance at recognizing or transcribing words accurately, offering insights into its effectiveness in understanding and processing language at the word level.

3.4 Composite Model Architecture

The end-to-end composite model described in this thesis integrates TrOCR and CharBERT to enhance text recognition accuracy. TrOCR is responsible for the initial text recognition task, while CharBERT functions as a corrector, refining the outputs from TrOCR. Specifically, the input to the TrOCR decoder is first corrected by CharBERT before it is processed by the TrOCR decoder. This step ensures that the TrOCR decoder works with embeddings that have been optimized by CharBERT, potentially improving the accuracy of the text recognition. In this section, we will delve into the architectures of both TrOCR and CharBERT, and discuss the methodology behind their integration within the composite model.

As depicted in Figure 10, the process begins with the image being processed by the TrOCR encoder. Subsequently, the input to the TrOCR decoder is first refined by CharBERT (highlighted in purple) before being fed into the TrOCR Decoder, following the standard transformer architecture.

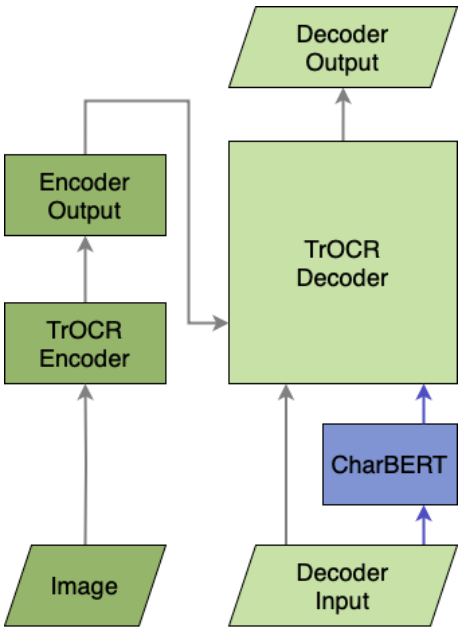


Figure 10: Schematic of the integrated workflow in the composite model combining TrOCR and CharBERT.

3.4.1 Notations

In this report, we denote a one-dimensional tensor (vector) using boldface lowercase letters, while a multi-dimensional tensor appears as boldface uppercase letters. The TrOCR sequence IDs are denoted by $\mathbf{W} = \mathbf{w}_1, ..., \mathbf{w}_i, ..., \mathbf{w}_D$, where each $\mathbf{w}_i$ repre-

sents a token tokenized by the TrOCR tokenizer, and D signifies the length of the sequence.

3.4.2 Recogniser - TrOCR

TrOCR is composed of an image encoder and a text decoder. The image encoder takes the pixel values of an image as input, and the encoder output is then fed into the decoder for text generation. This study involves adapting the decoder to integrate with CharBERT. Therefore, it will be crucial to focus on and provide a detailed explanation of its architecture. The following contents will first focus on the inputs of the decoder and the label that is used for loss calculation. Then, we will elaborate on how the inputs are processed through the decoder.

Decoder Inputs and Labels The text decoder takes the encoder output, decoder input IDs, and padding attention mask[8] as inputs, and outputs the generated text. The decoder output is then compared to the label tensor for loss calculation. This loss is used to update the gradients of the parameters during training.

The encoder output is a set of features extracted from the pixel values of the image. These features represent a transformed version of the input image, capturing not only the patterns but also the arrangements and relationships among elements within the image.

Decoder input IDs are tensors of token IDs converted from the label texts. These include three special tokens: `<bos>`, `<eos>`, and padding. The ID `<bos>`, represented by the number 0, indicates the start of the text, while the ID `<eos>`, denoted by the number 2, signifies the end of the text. Padding, a series of the number 1, is appended after `<eos>` to ensure all input IDs are of uniform length. The decoder input IDs, generated automatically by the TrOCR tokenizer, start with `<bos>` and end with `<eos>` and padding, mimicking the inference process during training[9].

The padding attention mask, another input for the decoder, determines which parts of the data the TrOCR should ignore, consisting of 0s and 1s where the 0s cover the

[8]Note that the padding attention mask should not be confused with the causal attention mask. The padding attention mask prevents the model from attending to the paddings, while the causal attention mask ensures that the prediction for a specific token in the sequence can only be influenced by previously generated tokens.

[9]During inference, the decoder output from the previous step becomes the new decoder input. However, as no output sequence has been generated at the first step, the initial decoder input comprises only the `<bos>` token. This is processed by the decoder to produce an output, which is then used to update the decoder input.

padding portions of the decoder input IDs.

The label tensor used for loss calculation is the decoder input IDs minus the <bos> token. Since the TrOCR model is based on the VisionEncoderDecoderModel, it requires label paddings to be set to -100. Thus, the padding token in the label tensor is replaced with -100 instead of 1s. The whole process is illustrated in Figure 11.

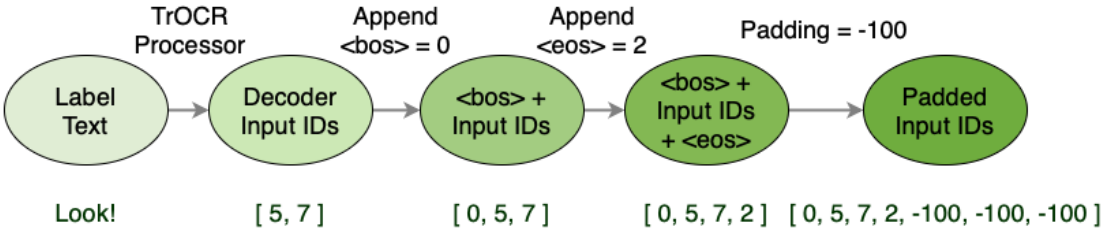


Figure 11: Processing pipeline for transforming label text into padded input IDs for the TrOCR model.

Inside the Decoder Let's first consider the original decoder before adapting it to integrate with CharBERT. The decoder receives three primary inputs: the decoder input IDs, the padding attention mask, and the encoder output. Initially, the decoder input IDs are converted into decoder embeddings. These embeddings are then combined with positional encoding, which preserves spatial information, before being processed by the decoder stacks. The decoder stacks include masked multi-head attention, multi-head attention, and a feedforward neural network (FFNN). The padding attention mask is applied during both the masked multi-head attention and multi-head attention stages to prevent the TrOCR from attending to the padding tokens. Finally, the output from the decoder stacks passes through a linear layer and a softmax layer to generate the final output sequence.

3.4.3 Corrector - CharBERT

CharBERT features a dual-channel architecture, comprising token and character channels. The token channel mirrors the structure of BERT or RoBERTa, depending on the initialization. In this study, we will specifically focus on $\text{CharBERT}_{\text{RoBERTa}}$. Henceforth in this document, any reference to "CharBERT" will pertain exclusively to $\text{CharBERT}_{\text{RoBERTa}}$, unless otherwise specified. Unlike BERT and RoBERTa, which output a single embedding, CharBERT generates both token-level and character-level embeddings. Given that the token channel's architecture is identical to that of BERT/RoBERTa, the subsequent sections will concentrate on the character channel and its heterogeneous interactions.

Character Channel & Heterogeneous Interaction The character channel of CharBERT begins by splitting the input text into individual characters and converting these characters into IDs via a dictionary lookup. These IDs are then embedded, followed by processing through a bidirectional GRU (Bi-GRU) layer to generate the output embeddings. These embeddings, at both the character and token levels, are subsequently passed through a transformer and undergo heterogeneous interaction, which is comprised of two parts: fusion and divide. The fusion process allows the embeddings to enrich each other through the use of a FFNN and a convolutional neural network (CNN), enhancing their mutual characteristics. Conversely, the divide process ensures that each set of embeddings retains its unique features, facilitated by an FFNN and a residual connection. This residual connection is crucial as it helps preserve the distinct information from each embedding type. CharBERT repeats the transformer and heterogeneous interaction cycles to capture more features and information inherent in the input texts.

3.4.4 Composite Model

The composite model is ingeniously designed to integrate TrOCR and CharBERT. During the inference phase, the decoder output is recycled back as input in a feedback loop. Before this recycled input is fed back into the decoder, it undergoes correction and refinement by CharBERT. Consequently, CharBERT is positioned between the decoder input and the decoder stacks to ensure that the input to the decoder is optimized in each iteration of the process. However, integrating these systems presents several challenges: 1) The TrOCR decoder accepts token IDs as input, whereas CharBERT outputs embeddings; 2) CharBERT requires textual inputs, but the input to the TrOCR decoder is a tensor; 3) The embedding representations of TrOCR do not align with those of CharBERT; 4) The input to the TrOCR decoder is a single tensor, while CharBERT produces dual-channel outputs. The subsequent sections will discuss these issues in detail and explore potential approaches to resolve them.

Adapted TrOCR Before addressing the first problem, it is crucial to thoroughly understand the issue. The TrOCR decoder is specifically designed to accept token IDs as input, which are then mapped to embeddings. These embeddings are subsequently augmented with positional encoding before being fed into the Transformer decoder. The challenge arises from the outputs of CharBERT, which are embeddings rather than token IDs, leading to a compatibility issue. If we were to resolve this problem by converting CharBERT's embeddings back into token IDs for input into

the TrOCR decoder, a new issue surfaces. Using token IDs as intermediaries between model components is problematic because token IDs are integers, while during the training process, the model weights are updated as floating-point numbers. Converting these weights to integers is not practical; while rounding could technically convert floats to integers, this approach would likely distort the learning process and could render the results meaningless for the intended task. Therefore, the only viable solution is to adapt the TrOCR decoder to accept embeddings directly.

A straightforward solution involves repositioning the embedding layer from the TrOCR decoder to precede CharBERT. This adjustment ensures that token IDs are initially converted to TrOCR embeddings, which are then input into CharBERT for correction. Consequently, the adapted TrOCR can accept embeddings directly, bypassing the need for token IDs. This modification not only ensures that the outputs from CharBERT are seamlessly integrated into the TrOCR decoder but also eliminates the use of token IDs between model components. This entire modification process is illustrated in Figure 12.

Adapted CharBERT The second problem addresses the mismatch between the data types of input and output. According to the modifications described in Adapted TrOCR, the TrOCR decoder input is now an embedding, which should be processed by CharBERT for correction. However, CharBERT traditionally only accepts text as input. Therefore, it becomes essential to develop an adapted version of CharBERT that can handle token and character embeddings directly as inputs. In this revised model, CharBERT no longer converts text into IDs and then into embeddings; instead, it receives pre-processed embeddings directly. This adaptation allows both token and character embeddings to be processed through their respective channels in CharBERT. A comparison between the original and the adapted CharBERT models is illustrated in Figure 13.

Tensor Transform The third problem presents a more complex challenge. The TrOCR decoder relies on embeddings that are unique to TrOCR. Even for identical texts, the embedding representations generated by TrOCR and CharBERT differ significantly, further complicated by CharBERT's dual-channel embeddings. Not only do the models represent the same text differently, but their embedding dimensions are also incompatible. As a result, even though CharBERT is capable of processing embeddings directly, the TrOCR decoder input cannot be directly integrated into CharBERT without modifications. To overcome this issue, an architecture comprising CNN and FFNN is utilized. This strategy is designed to adjust the dimensions

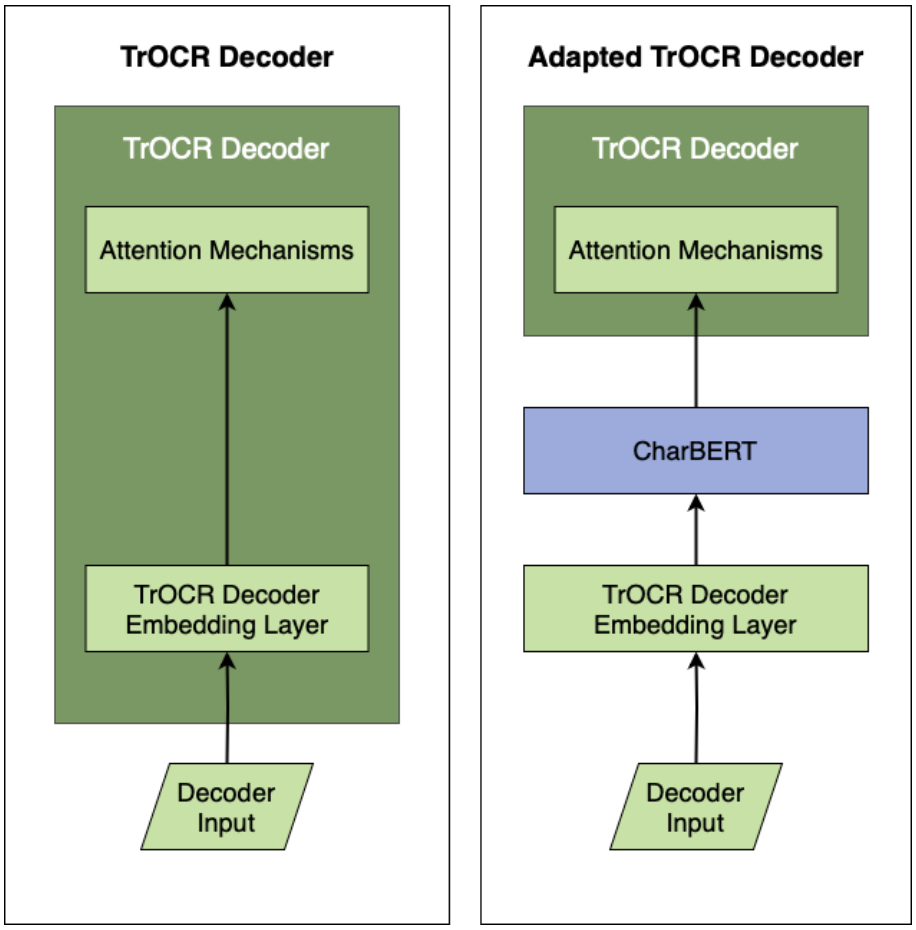


Figure 12: **Comparison of the TrOCR Decoder architecture**
On the left, the original TrOCR Decoder structure includes the Decoder Input, TrOCR Decoder Embedding Layer, and Attention Mechanisms. On the right, the adapted TrOCR Decoder structure shows the repositioning of the TrOCR Decoder Embedding Layer below CharBERT, indicating the integration of CharBERT into the decoder process to enhance input handling before the attention mechanisms.

of the TrOCR decoder input to match those of the CharBERT token and character embeddings. Additionally, this dimensional transformation facilitates an effective alignment of representations between TrOCR and CharBERT. To clarify the architecture of the tensor transformation, dimensions will be noted after each input and output, using the format `(batch size, embedding size)`.

The objective of the tensor transform architecture is to adjust the dimensions of the TrOCR decoder input, which are initially set at `(batch size, 512, 1024)`, to match the dimensions required by CharBERT's token embeddings `(batch size, 510, 768)` and character embeddings `(batch size, 3060, 256)`. In this architec-

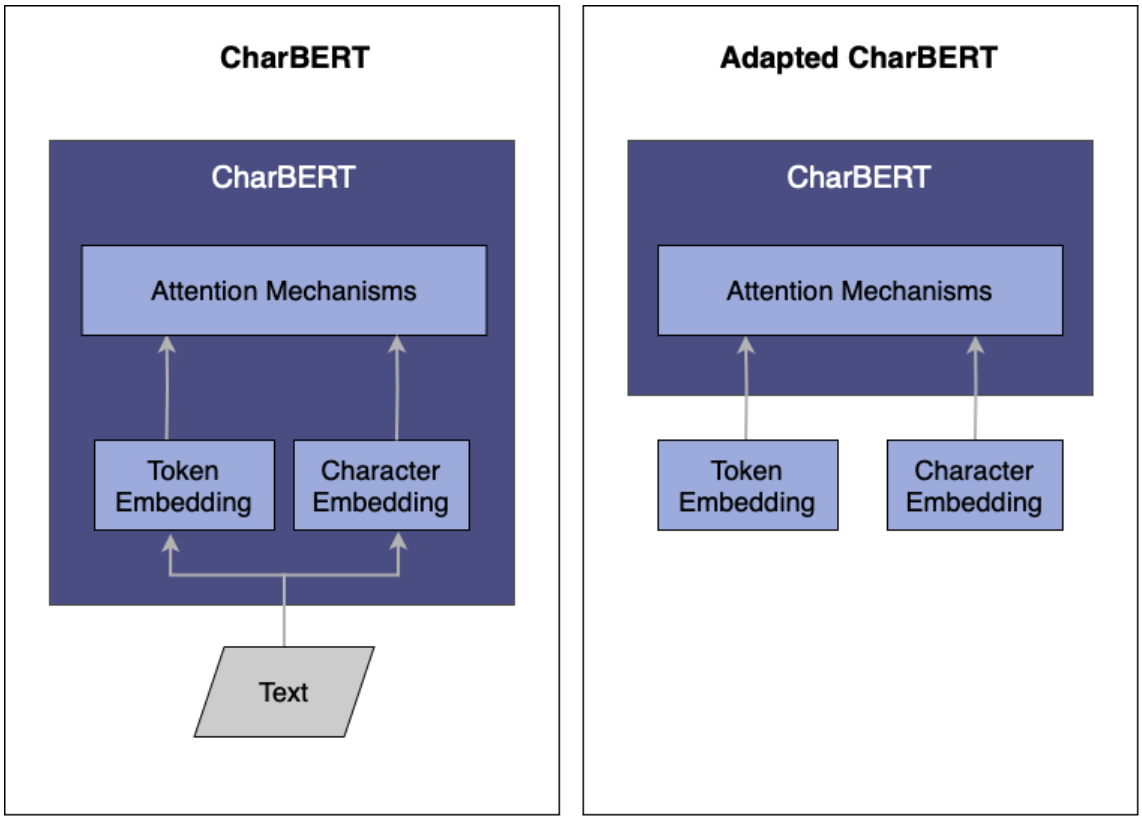


Figure 13: **Comparison of the CharBERT architecture**
On the left, the original CharBERT model processes text into token and character embeddings, which are then fed into the attention mechanisms. On the right, the adapted CharBERT directly receives token and character embeddings as inputs, bypassing the initial text processing stage, and processes them through the same attention mechanisms.

ture, the second dimension represents the sequence length, and the third dimension represents the embedding size. The transformation process is divided into two distinct stages. In the first stage, the decoder input passes through a series of CNN layers, which are interspersed with LeakyReLU activation functions and batch normalization, specifically designed to adjust the sequence dimension (dim=1). Subsequently, the output from the first stage is processed through FFNN layers, with interspersed LeakyReLU activations, in the second stage to modify the embedding dimension (dim=2).

$$\begin{aligned}
\mathbf{t}_{1,j} &= \text{LeakyReLU}(\mathbf{b}_1 + \sum_{k=1}^{3} \mathbf{W}_{1,k} \cdot \mathbf{e}_{i+k-1}) \quad ; \quad \mathbf{t}'_1 = \text{Batch_Norm}(\mathbf{t}_1) \\
\mathbf{t}_{2,l} &= \text{LeakyReLU}(\mathbf{b}_2 + \sum_{k=1}^{3} \mathbf{W}_{2,k} \cdot \mathbf{t}'_{1,j+k-1}) \quad ; \quad \mathbf{t}'_2 = \text{Batch_Norm}(\mathbf{t}_2) \\
\mathbf{t}_{3,p} &= \text{LeakyReLU}(\mathbf{b}_3 + \sum_{k=1}^{3} \mathbf{W}_{3,k} \cdot \mathbf{t}'_{2,l+k-1})
\end{aligned} \tag{3.3}$$

After the application of the CNN layers, the dimension of the decoder input is adjusted to (batch size, 510, 1024), aligning the sequence dimension (dim=1) with the desired dimension for the CharBERT token embedding. The CNN layers are specifically chosen for their ability to expand or contract the sequence length while preserving spatial information, making them more suitable than FFNNs for adjusting sequence dimensions. Additionally, it is important to note the inclusion of batch normalization steps between the convolutional layers. These batch normalizations stabilize the deep model and maintain healthier gradients, especially important due to the presence of activation functions[10]. Following this, the tensor transformation process moves to the second stage, which utilizes FFNN layers.

$$\begin{aligned}
\mathbf{T}_4 &= \text{LeakyReLU}(\mathbf{b}_4 + \mathbf{W}_4 \cdot \mathbf{T}_3) \\
\mathbf{T}_5 &= \text{LeakyReLU}(\mathbf{b}_5 + \mathbf{W}_5 \cdot \mathbf{T}_4) \\
\mathbf{T}_n &= \text{LeakyReLU}(\mathbf{b}_6 + \mathbf{W}_6 \cdot \mathbf{T}_5)
\end{aligned} \tag{3.4}$$

The resulting tensor, the CharBERT token embedding, is denoted as $\mathbf{T} = \mathbf{t}_1, ..., \mathbf{t}_n, ..., \mathbf{t}_N$, where N represents the token sequence length and the tensor size is `(batch size, 510, 768)`.

Similarly, operations are applied to the TrOCR decoder input but adjusted for different dimension expansion to produce the CharBERT character embedding $\mathbf{C} = \mathbf{c}_1, ..., \mathbf{c}_m, ..., \mathbf{c}_M$, where M denotes the character sequence length. It is established that M is typically six times that of N, based on the assumption that the average word contains six characters.

The Tensor Transform layer is specifically designed to convert the TrOCR decoder input into CharBERT token and character embeddings. The CNN and FFNN layers are instrumental not only in aligning the dimensions between the tensors but also

[10]With activation functions in deep networks, gradients can easily explode or vanish. Batch normalization helps maintain a healthier gradient flow in the network, which can improve the efficiency of backpropagation and thereby enhance the learning process.

in learning to map the contextual information from TrOCR embeddings to those of CharBERT, effectively adapting the embeddings for integrated processing.

Tensor Combine The fourth problem arises because CharBERT produces two separate tensors—token and character representations—while the TrOCR decoder input requires a single tensor. To address this, the solution involves combining the two output tensors from CharBERT into a single tensor. Additionally, to stabilize the deep model, a residual connection from the original TrOCR decoder embedding can be added. This residual connection helps to reuse features from the original TrOCR decoder embedding and prevents gradient vanishing.

Among the three tensors involved, two are derived from CharBERT. Consequently, these representations are not inherently compatible with the TrOCR decoder, and their dimensions do not match. Thus, they must first undergo a transformation via the Tensor Transform process before they can be effectively combined.

$$\begin{aligned} \mathbf{T} \in \mathbb{R}^{d_N} &\rightarrow \mathbf{T}' = \text{Tensor\_Transform}(\mathbf{T}) \in \mathbb{R}^{d_D} \\ \mathbf{C} \in \mathbb{R}^{d_M} &\rightarrow \mathbf{C}' = \text{Tensor\_Transform}(\mathbf{C}) \in \mathbb{R}^{d_D} \end{aligned} \tag{3.5}$$

where d_D = `(batch size, 512, 1024)`; d_N = `(batch size, 510, 768)`; d_M = `(batch size, 3060, 256)`.

After the tensor transform module is applied, the three tensors will all conform to the size `(batch size, 512, 1024)`, matching the original input size of the TrOCR decoder. These tensors can then be merged and fed into the TrOCR decoder stack. In this study, we explore four different architectures for the tensor combination module: 1) simple addition of all tensors; 2) mean pooling; 3) utilization of linear layers as an attention network; 4) implementation of convolutional layers as an attention network.

Tensor Combine 1: Adding The first tensor transform module aggregates three tensors into a single tensor by performing an element-wise addition of the three input tensors. Specifically, this involves the TrOCR decoder embeddings $\mathbf{E}$, the CharBERT transformed token representation $\mathbf{T}'$, and the CharBERT transformed character representation $\mathbf{C}'$. Each of these contributes equally to the formation of the combined tensor, ensuring that the information from each input tensor is weighted equally. The operation can be mathematically expressed as follows:

$$\mathbf{E}'_{1,ijk} = \mathbf{E}_{ijk} + \mathbf{T}'_{ijk} + \mathbf{C}'_{ijk} \tag{3.6}$$

where $i = 1, 2, ...,$ batch size; $j = 1, 2, ..., 512$; $k = 1, 2, ..., 1024$.

Tensor Combine 2: Mean Pooling The second tensor combination module is designed to dynamically allocate attention weights to each word across the three input tensors. These weights are derived by performing max pooling along the embedding axis of the three tensors, once they are stacked together. After obtaining these weights, they are applied to the stacked embeddings. It is important to note that all three tensors undergo feature-wise normalization prior to stacking. This normalization aids in faster model convergence and promotes better generalization.

$$
\begin{aligned}
&\mathbf{T'}_{\text{norm}} = \text{Feature\_Norm}(\mathbf{T'}) \\
&\mathbf{C'}_{\text{norm}} = \text{Feature\_Norm}(\mathbf{C'}) \\
&\mathbf{E''}_{\text{norm}} = \text{Feature\_Norm}(\mathbf{E'}) \\
&\mathbf{S} = \text{Stack}(\mathbf{E'}, \mathbf{T'}_{\text{norm}}, \mathbf{C'}_{\text{norm}}; \text{axis} = 1) \in \mathbb{R}^{(\text{batch size, 3, 512, 1024})} \\
&\mathbf{P} = \frac{1}{1024} \sum_{j=1}^{1024} \mathbf{S}_{a,b,c,j} \in \mathbb{R}^{(\text{batch size, 3, 512})} \\
&\mathbf{E''} = \mathbf{P} \cdot \mathbf{S}
\end{aligned}
\tag{3.7}
$$

Tensor Combine 3: Linear Layers as Attention Net The third tensor combination module builds upon the second module by incorporating linear layers as the attention network. Rather than merely performing mean pooling, this module employs linear layers interspersed with activation functions to more effectively capture relevant information when determining the weights. The process is described as follows:

$$
\begin{aligned}
&\mathbf{T'}_{\text{norm}} = \text{Feature\_Norm}(\mathbf{T'}) \\
&\mathbf{C'}_{\text{norm}} = \text{Feature\_Norm}(\mathbf{C'}) \\
&\mathbf{P} = \text{Stack}(\mathbf{E}, \mathbf{T'}_{\text{norm}}, \mathbf{C'}_{\text{norm}}; \text{axis} = 1) \in \mathbb{R}^{(\text{batch size, 3, 512, 1024})}
\end{aligned}
\tag{3.8}
$$

$$
\begin{aligned}
&\text{For } i = 1, 2, ..., 512: && \text{Conv Attention Net} \\
&\quad \mathbf{S}' = \mathbf{S}[:, :, i, :] \in \mathbb{R}^{(\text{batch size, 3, 1024})} \\
&\quad \mathbf{S}'_1 = \text{LeakyReLU}(\mathbf{b}_7 + \mathbf{W}_7 \cdot \mathbf{S}') \\
&\quad \mathbf{S}'_2 = \text{LeakyReLU}(\mathbf{b}_8 + \mathbf{W}_8 \cdot \mathbf{S}'_1) \\
&\quad \mathbf{S}'_3 = \mathbf{b}_9 + \mathbf{W}_9 * \mathbf{S}'_2 \\
&\quad \mathbf{P}_i = \text{softmax}\Big(\frac{1}{1024} \sum_{j=1}^{1024} \mathbf{S}'_{3,(a,b,j)}\Big) \in \mathbb{R}^{(\text{batch size, 3})}
\end{aligned}
\tag{3.9}
$$

$$
\begin{aligned}
&\mathbf{P} = [\mathbf{P}_1; \mathbf{P}_2; ...; \mathbf{P}_{512}] \in \mathbb{R}^{(\text{batch size, 3, 512})} \\
&\mathbf{E}' = \mathbf{P} \cdot \mathbf{S}
\end{aligned}
\tag{3.10}
$$

Tensor Combine 4: Convolutional Layers as Attention Net The fourth tensor combination module substitutes linear layers with convolutional layers in the attention network. The equations below illustrate this attention network, which differs from the one described in Tensor Transform - Linear Layers as Attention Net.

$$
\begin{aligned}
&\text{For } i = 1, 2, ..., 512: && \text{Linear Attention Net} \\
&\quad \mathbf{S}' = \mathbf{S}[:, :, i, :] \in \mathbb{R}^{(\text{batch size, 3, 1024})} \\
&\quad \mathbf{S}'_1 = \text{LeakyReLU}(\mathbf{b}_7 + \mathbf{W}_7 \cdot \mathbf{S}') \\
&\quad \mathbf{S}'_2 = \text{LeakyReLU}(\mathbf{b}_8 + \mathbf{W}_8 \cdot \mathbf{S}'_1) \\
&\quad \mathbf{S}'_3 = \mathbf{b}_9 + \mathbf{W}_9 * \mathbf{S}'_2 \\
&\quad \mathbf{P}_i = \text{softmax}\Big(\frac{1}{1024} \sum_{j=1}^{1024} \mathbf{S}'_{3,(a,b,j)}\Big) \in \mathbb{R}^{(\text{batch size, 3})}
\end{aligned}
\tag{3.11}
$$

Change to conv version

3.5 Training CharBERT$_{\text{SMALL}}$

The training of CharBERT$_{\text{SMALL}}$ closely follow the methodology outlined by the original authors. The model undergoes training using pre-training objectives, specifically masked language modeling (MLM) and noisy language model (NLM), across a large text corpus. Training entails backpropagation and weight optimization to minimize prediction errors for both MLM and NLM tasks.

While maintaining adherence to the hyperparameters recommended by the original

authors, we made adjustments to the batch size, increasing it from the suggested size of 4 to 16, to expedite the computation process. The training process utilized five A100 GPUs, each equipped with 80GB of RAM, and spanned six days, comprising five training epochs.

We train

3.6 Glyph Incorporation

In our methodology, we enhance the training process by specifically targeting commonly misrecognized characters, such as "." and "," or "O" and "o", with the aim of reducing the likelihood of these errors in future recognitions. To achieve this, we commence by determining the probability $\mathcal{P}_{ij}$, where i represents the correct character that has been erroneously recognized as character j. This strategy is a deviation from the CharBERT NLM training approach, which incorporates character-level errors into the text at random.

By leveraging $\mathcal{P}_{ij}$, we refine our training methodology to introduce errors in a more systematic manner, based on the observed probabilities of specific misrecognitions. This targeted approach allows us to focus the model's learning on correcting these particular errors, enhancing its accuracy and reliability in distinguishing between characters that are commonly confused.

3.6.1 Get $\mathcal{P}_{ij}$

To obtain $\mathcal{P}_{ij}$, it is first necessary to calculate the frequency of each character misrecognized by the recognizer (TrOCR), termed as *misrecognized frequency*. This process begins with employing the basic TrOCR model to process the GW and JH datasets. After generating text outputs with TrOCR, these are compared with the corresponding text labels to determine the frequencies of characters that were misrecognized.

To align the generated outputs with the labels, we utilize the "Bio" package in Python, specifically referring to Biopython. This toolkit is primarily designed for biological computation, handling data such as DNA, RNA, and protein sequences, and offers extensive features for sequence analysis and alignments. It also provides extensive features for sequence analysis and alignments, making it highly suitable for our needs. Below, we present an example of how the Bio package aligns OCR output with a label. For the OCR output "Company" and the label "company", the

toolkit not only matches each corresponding character but also identifies missing or misrecognised ones with vertical lines. This clear visualization allows us to easily determine which characters are frequently misrecognised by our recognition model. This example is particularly useful for understanding common recognition errors and adjusting the model accordingly.

```
0 *C-ompany*  9
0 |--|||||||  10
0 *-company*  9
```

Although Biopython is not inherently designed for processing natural language texts, its tools for sequence matching and regular expressions can be adapted for text analysis applications. For instance, in our context, we can apply Biopython's methodologies to perform precise character-by-character matching within text strings, identifying and annotating discrepancies. This approach demonstrates how Biopython's core functionalities can be repurposed for non-biological applications, providing valuable methods for text data analysis and enhancing our understanding of text misrecognition patterns.

3.6.2 Training CharBERT$_{\mathcal{P}_{ij}}$

To train CharBERT$_{\mathcal{P}_{ij}}$, we meticulously adhere to the methodologies outlined in the original CharBERT study. This involves processing text through the model's dual-channel architecture, which includes both token and character channels, as detailed in Section 3.4.3. We also faithfully follow the hyperparameter settings recommended in the CharBERT publication, which are documented extensively at this link.

In the conventional CharBERT training regimen, the model is typically conditioned by randomly adding, deleting, or swapping characters within the input text to simulate typical typographical errors, thereby training CharBERT to correct them. For our specific application focusing on OCR corrections, we have modified this approach by replacing the random swapping of characters with targeted replacements. These replacements are guided by the misrecognition probabilities $\mathcal{P}ij$, which were derived from the analysis of character frequencies in misrecognized words by the TrOCR system, as discussed earlier. This modification reflects the error patterns more commonly observed in OCR outputs, ensuring that our training process closely mimics real-world OCR challenges and optimizes CharBERTSMALL for more accurate error correction in this context.

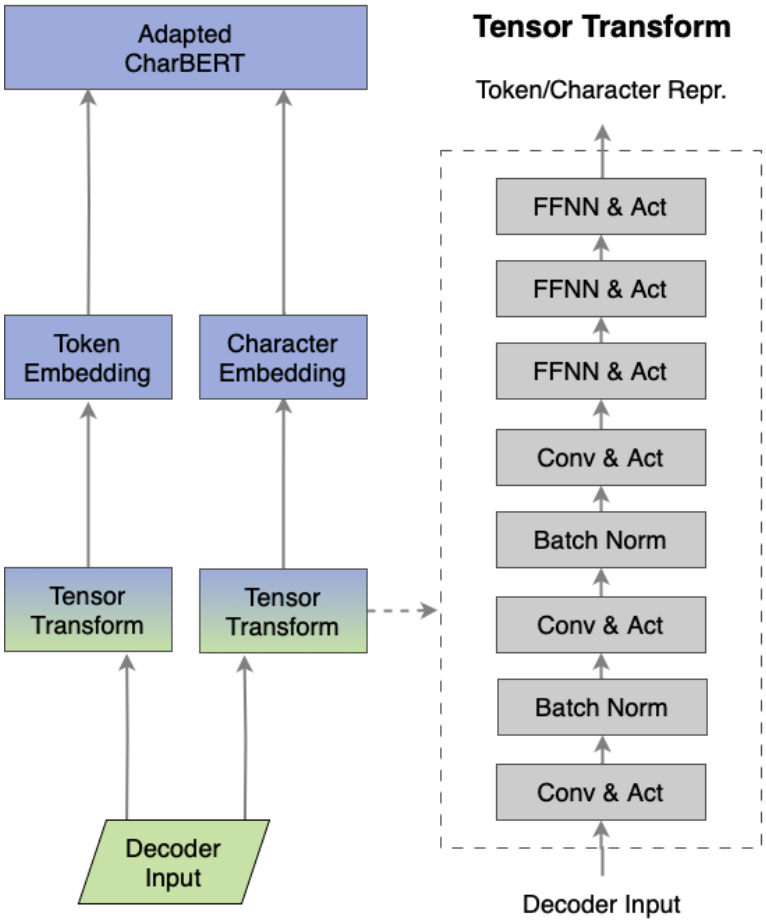


Figure 14: **Architectural flow of the adapted CharBERT and its integration with the TrOCR decoder through tensor transformation** The diagram shows how token and character embeddings from the adapted CharBERT are processed through separate tensor transformation pathways to align their dimensions and formats with the requirements of the TrOCR decoder input. Each transformation pathway involves a series of convolutional (Conv) and activation (Act) layers, interspersed with batch normalization (Batch Norm) and further refined by FFNN and additional activation layers, culminating in the combined decoder input.

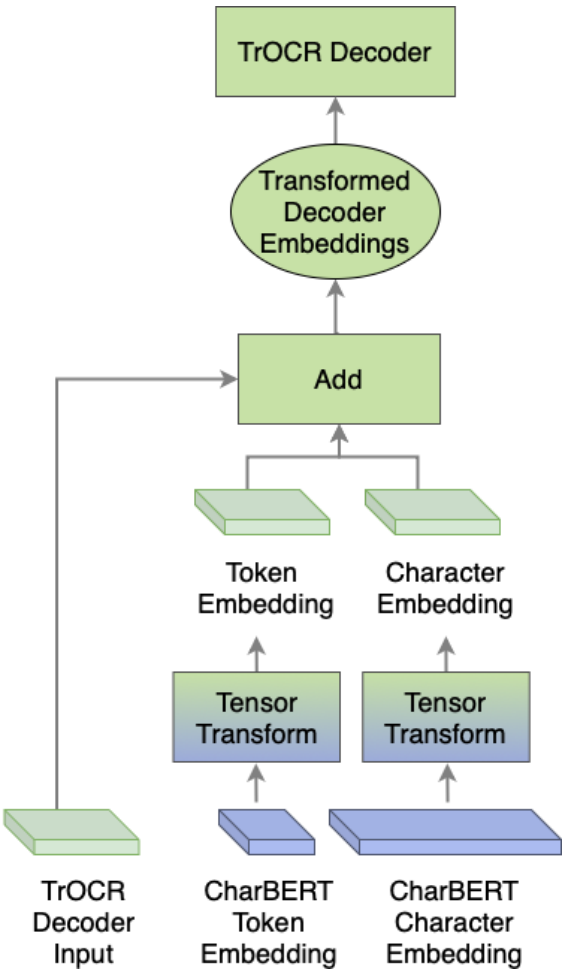


Figure 15: **Architecture of the Tensor Combine Module 1**
This diagram illustrates the flow from the TrOCR decoder and CharBERT token and character embeddings through their respective tensor transform processes. The transformed embeddings are then combined using an element-wise addition to form the transformed decoder embeddings, which are subsequently fed back into the TrOCR decoder.

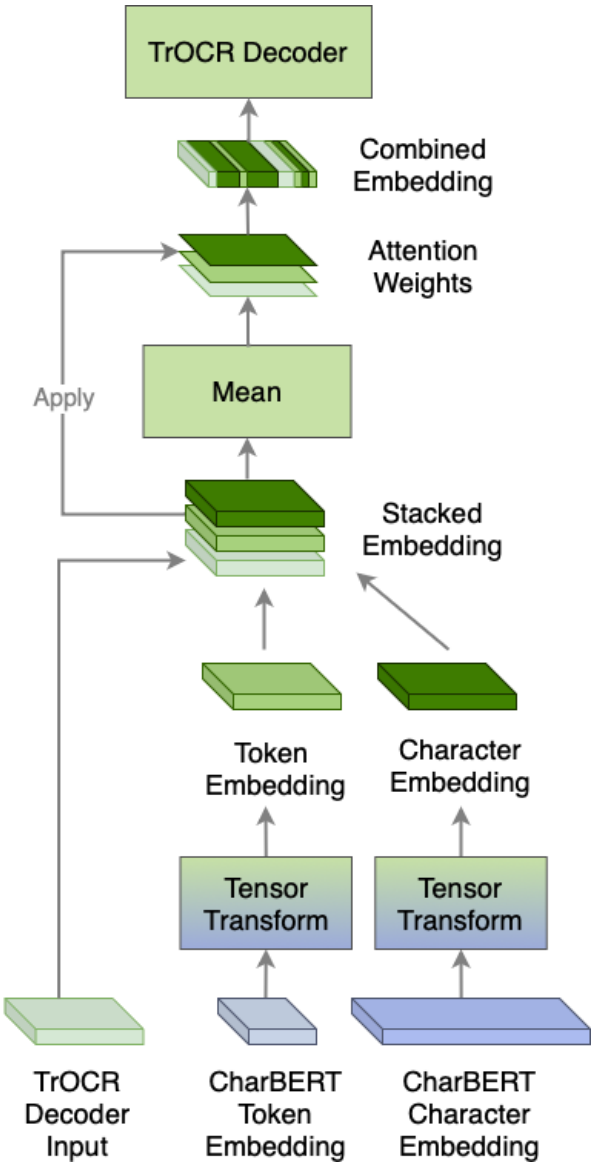


Figure 16: **Architecture of the Tensor Combine Module 2**
This module utilizes mean pooling to dynamically allocate attention weights across the three input tensors. After the tensors are stacked, attention weights are derived and applied, followed by mean pooling to merge the embeddings into a single combined embedding, which is then processed by the TrOCR decoder.

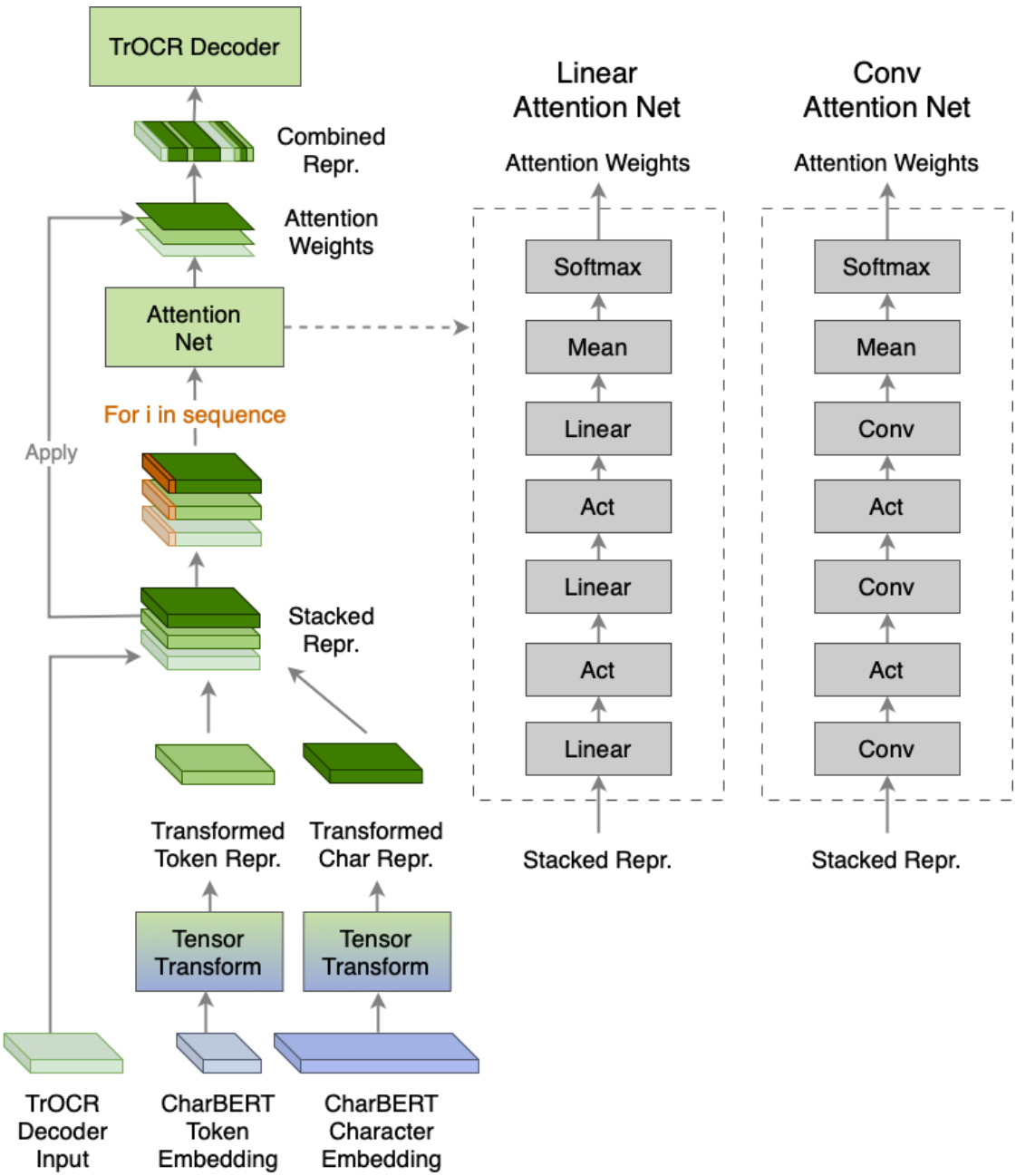


Figure 17: **Architecture of the Tensor Combine Module 3 and 4**
For each input sequence, the tensors are stacked and transformed through either linear or convolutional layers (detailed in the right two panels) to generate attention weights via a softmax function. The weights are then applied to compute a weighted mean, resulting in an attention-modified combined representation, which is subsequently fed into the TrOCR decoder.

4 Experiment

This chapter presents a detailed examination of the experimental setup and evaluation criteria in our study. Beginning with the data collection and processing section, we outline the sources and types of data utilised to benchmark the performance of our composite model and the training of $\text{CharBERT}_{\text{SMALL}}$. This is followed by a discussion on the baseline model, where we compare both the pre-trained and fine-tuned versions of the TrOCR model to establish performance benchmarks. The subsequent sections delve into the training and evaluation criteria for the composite model and $\text{CharBERT}_{\text{SMALL}}$. They highlights the strategic choices made in terms of optimizers, loss functions, and hyperparameters. In addition, we address common training challenges and our solutions. Finally, the analysis section explores the impact of specific model components and features on overall performance. This comprehensive approach not only enables a deeper understanding of the models' capabilities and limitations but also guides the refinement of OCR technologies for enhanced performance and efficiency.

Rewrite this

4.1 Baseline Model

In this study, we consider both the pre-trained and fine-tuned versions of the TrOCR model as baseline models for comparison. The rationale behind using the fine-tuned TrOCR model alongside its pre-trained counterpart is to ascertain the performance benchmark. If the fine-tuned TrOCR model outperform the composite model, it would suggest that further fine-tuning of the TrOCR model is a more efficient approach than employing the larger, more complex composite model. This comparison allows us to evaluate the efficiency of the fine-tuning process relative to the integration of additional model components.

Pre-trained TrOCR We use the pre-trained handwritten large TrOCR as the baseline model. The model is trained on synthetic data and IAM handwritten dataset, and evaluated on IAM handwritten dataset.

Fine-tuned TrOCR Another stronger baseline of ours is the fine-tuned TrOCR on GW and IAM handwritten datasets. We further include this baseline model for the following reasons:

First of all, fine-tuning TrOCR on a specific dataset can enhance its performance significantly. Camparing the composite model with both the original and fine-tuned TrOCR allows us to assess the added value of integrating CharBERT. In other words, demonstrating that the composite model outperforms not just the original but also a fine-tuned version of TrOCR helps justify this added complexity.

In addition, including both the original and fine-tuned versions of TrOCR as baselines provides a more comprehensive view of how the composite model performs across different stages of model optimization. This comparison is crucial for understanding whether the improvements come from fine-tuning, integrating CharBERT, or both.

Lastly, fine-tuning allows the model to adapt to the specific characteristics and distribution of your data, which might be significantly different from the data TrOCR was originally trained on. By evaluating against a fine-tuned baseline, we ensure that comparisons take into account the model's ability to handle data-specific challenges.

4.2 Composite Model Training Setups

In this section, we delve into the training and evaluation process for the composite model. The section will cover key aspects such as the choice of optimizer, the loss function employed, and the hyperparameters set for the training. Additionally, the challenges commonly faced when training deep learning models, and insights into strategies and solutions to mitigate these issues will also be discussed.

4.2.1 Training Details

When training deep learning models, the selection of optimisation, loss computation techniques, and the hyperparameters settings play a crucial role in the effeiciency and effectiveness of the training process. In this study, we use Adam as the optimiser and cross-entropy for loss computation.

Optimiser Utilising Adam as the optimiser provides a sophisticated approach by adopting an adaptive learning rate, which is particularly adept at managing sparse

gradients tasks such as NLP problems. Besides, due to its efficient computation of adaptive learning rates, Adam often leads to faster convergence on training data. This can significantly reduce the time and computational resources needed to train deep models, making the process more efficient.

For the training of the composite model, the learning rate has been meticulously set to 1e−5. This settingn ensures a controlled and gradual adaptation of the model parameters, facilitating a smooth convergence towards the local minima. Additionally, a weight decay parameter of 1e−5 is employed to enhance the model's ability to navigate the optimization landscape efficiently. This careful calibration prevents the optimizer from overshooting the local minima, thereby promoting stability in the training process and improving the model's overall performance.

Loss Function OCR tasks involves predicting the probability distribution of possible characters for a given input image. It is a multi-class classification problem, with each character representing a unique class. Cross-entropy loss is naturally suited for multi-class settings. This makes it directly applicable to the task of classifying images into characters. In addition, unlike other loss functions that might focus solely on the accuracy of the classification, cross-entropy loss encourages the model not just to predict the correct class but to do so with high confidence. High-confidence wrong predictions are penalised more, encouraging the model to be cautious about making predictions when unsure, which is often the case with less frequent characters.

4.2.2 Training Challenges and Solutions

Training deep learning models involves navigating a series of common challenges that can significantly impact their performance and effectiveness. Common challenges include overfitting, vanishing/exploding gradients, high computational costs, and data quantity. The following discussion will delve into these challenges and the strategies used in this study to mitigate them.

Overfitting Overfitting occurs when a model learns the training data too well, capturing noisein the training set instead of learning the underlying patterns, which results in poor generalisation to new, unseen data. Deep learning models, by their very nature, have a large number of parameters, allowing them to model intricate patterns and relationships in the data. However, it also means they have the capacity to memorise irrelevant details in the training data, leading to overfitting. Lack of regularisation techniques, or poorly chosen learning rate and batch size can easily

lead to overfitting.

To mitigate overfitting, we implemented dropout layers between both linear and convolutional layers within the composite model architecture. Dropout serves as a form of regularization that, by temporarily dropping out units from the network, prevents the model from becoming overly dependent on any single element of the training data, thereby enhancing its generalization capabilities.

Additionally, we explored the impact of batch size on model training. Smaller batch sizes result in more noise during the gradient updates, which can have a regularizing effect. However, very small batch sizes can lead to extremely noisy gradient estimates, which might make training unstable or lead to convergence on suboptimal solutions. Through iterative testing, we determined that a batch size of 8 strikes an optimal balance, offering sufficient regularization to mitigate overfitting while maintaining stable and effective training dynamics.

Vanishing/Exploding Gradients Training deep models often encounters exploding or vanishing gradient problems due to their complex architectures and the long chains of computations involved. If the gradients are large (greater than 1), they can exponentially increase as they propagate back through the layers, leading to exploding gradients. Conversely, if the gradients are small (less than 1), they can exponentially decrease, leading to vanishing gradients.

Certain activation functions, like the sigmoid or tanh, squish a large input space into a small output range in a non-linear fashion. For inputs with large magnitudes, the gradients can be extremely small, leading to vanishing gradients. In addition, improper initialization of weights can exacerbate the exploding or vanishing gradient problems. For instance, large initial weights can lead to exploding gradients, while small initial weights can contribute to vanishing gradients.

To mitigate the vanishing/exploding gradients, we deploy strategies such as Xavier initialisation, Leaky ReLU activation function, gradient clipping, residual connection and batch normalization in the composite model architecture. The details of residual connection and batch normalisation is discussed in Composite Model.

Make sure they are mentioned

High Computational Costs Deep models, especially those with many layers and parameters, require significant computational resources and time to train. Hardware accelerators like GPUs can reduce training times. In this study, a single A100 GPU, equipped with 80GB of RAM, was utilized to accelerate the training process. When applied to the GW dataset and the IAM dataset, the training durations were

approximately 2 minutes per epoch and 30 minutes per epoch, respectively.

Data Quantity Deep models typically require large datasets to effectively learn and generalize due to their complex architectures and the vast number of parameters they contain. Training these models from scratch on limited data often leads to overfitting. To mitigate this issue, we employ transfer learning. In this study, we take pre-trained CharBERT and TrOCR model, which are developed on large and comprehensive datasets, and adapt it to our specific task.

4.2.3 Evaluation

The validation set serves the purpose of hyperparameter tuning and model selection. Meanwhile, the testing set is reserved for the final evaluation. For the GW dataset, we employ a 4-fold cross-validation approach. This method divides the dataset into four equally sized segments, using each in turn for testing while the remaining three serve as the training set. The final results for the GW dataset represent the average performance across these four folds.

In this study, we focus on word error rate (WER) and character error rate (CER) as our primary evaluation metrics. These metrics are critical for assessing the model's accuracy in recognizing and reproducing text, providing the model's performance in terms of both word-level and character-level precision.

4.3 Composite Model Analysis

In this section, we conduct an in-depth analysis of composite neural network models, aiming to understand their performance, robustness, and adaptability across various tasks and datasets. Through a series of component analyses and benchmarking exercises, we investigate the efficacy of different model configurations and techniques, such as freezing specific layers, integrating dropout mechanisms, comparing tensor modules, and benchmarking against state-of-the-art models like GPT-4. Additionally, we delve into methods for validating model adaptability, exploring strategies for testing the model's performance on diverse datasets and assessing its ability to transfer learned patterns to new, unseen data environments. By examining the interplay between different components and techniques within composite models, we aim to provide comprehensive insights into their effectiveness and suitability for real-world applications.

4.3.1 Component Analysis and Benchmarking

In this section, we delve into various strategies aimed at enhancing the performance and robustness of our neural network models. To understand the contribution of specific components or features of a model to its overall performance, we conducted a comprehensive analysis. This analysis encompassed strategies such as 1) freezing specific layers; 2) integrating dropout mechanisms; 3) comparing combined tensor modules; 4) benchmarking against GPT-4. Each of these approaches was carefully implemented to isolate and understand the impact of various model elements on its efficiency. By comparing different combinations of tensor modules and loss functions, we gain valuable insights into the architecture's effectiveness and identify optimal configurations. Moreover, benchmarking our models against industry standards like GPT-4 provides a comprehensive evaluation of their capabilities, shedding light on areas of improvement and innovation.

Freezing Specific Layers When a layer is frozen, its weights remain unchanged during the training updates. This is often done to understand the contribution of specific layers to the model's learning and performance. By freezing specific layers and training the remaining layers, we can assess how changes in certain parts of the model affect its overall performance. This helps identify which layers are critical for learning the task at hand and which might be redundant or less influential.

Integrating Dropout Mechanisms Dropout is a regularization technique used to prevent overfitting in neural networks. By integrating dropout mechanisms into different parts of the model, we can analyze its effect on reducing overfitting and improving generalization. This helps to understand the importance of each component in achieving a balance between model complexity and its ability to perform well on unseen data.

Comparing Combined Tensor Modules This involves experimenting with different tensor combination modules within the model. By comparing how different combinations of these modules affect performance, we can understand how the architecture and data flow within the model contribute to its success or limitations.

Integrating Batch Normalization Batch normalization is a technique used in deep learning to improve the stability and speed of training neural networks. It helps to stabilize the learning process by normalizing the inputs to each layer. This reduces

the likelihood of vanishing or exploding gradients, which can occur when training deep neural networks, especially those with many layers. In addition, by normalizing the inputs, batch normalization allows for faster convergence during training. This means that the network can reach satisfactory performance levels with fewer training iterations, speeding up the overall training process.

Comparing Loss Functions Given the involvement of two models in our study, we explored various combinations of loss functions derived from the outputs of both models. This method not only allow us to assess the individual contributions of each model's output to the overall task, but also identify the most effective loss functions for the

Benchmarking Against GPT-4 Benchmarking against a model like GPT-4 provides a high standard for performance and innovation. It allows us to gauge how far your model has come in terms of understanding and generating text compared to leading models in the field. This comparison can highlight the strengths and weaknesses of our approach, offering insights into what components or features are contributing most to its performance.

4.3.2 Validating Model Cross-Dataset Performance

In this subsection, we explore methods to validate and enhance the adaptability of our composite model. Through a combination of dataset amalgamation and cross-training approaches, we investigate how our model can effectively learn from and adapt to varied data characteristics, ultimately ensuring its reliability and versatility in real-world applications.

The primary purpose of this method is to determine how well the model can adapt to different datasets and to identify any potential biases or limitations in its training. By testing the model on both the dataset it was trained on and an unrelated dataset, it's possible to gauge how effectively the model can transfer its learned patterns and predictions to new, unseen data, which is crucial for real-world applications.

This model evaluation strategy involves a cross-training and testing approach using two separate datasets, GW dataset and JH dataset. In this step, the model is first trained on GW dataset and then tested on both the test set of GW dataset and the test set of JH dataset. This process is then repeated by training the model on JH dataset and similarly testing it on the test sets of both datasets. This dual-training

approach allows the model to be evaluated under two different learning conditions, which helps in assessing both its dataset-specific performance and its generalization capabilities across varied data types.

This method provides a comprehensive understanding of the model's performance and robustness, offering insights into how training on one dataset might influence performance on another. This method also highlights the model's ability to generalize, which is critical for deploying machine learning models in environments where they will encounter varied types of data. Additionally, this strategy helps in fine-tuning the model by identifying specific weaknesses or overfitting issues, guiding further refinements to enhance its accuracy and adaptability. Overall, Step 2 ensures that the model not only performs well in a controlled, homogeneous dataset environment but is also effective and reliable when faced with diverse and dynamic data conditions.

4.3.3 Validating Model Domain Adaptability

The Candidate Fusion paper claimss that the integration of a recognizer with a language model (LM) not only enhances recognition accuracy but also enables the LM to adapt to domain-specific nuances. To examine whether a similar enhancement and adaptability can be achieved by integrating TrOCR with CharBERT, we embarked on an experimental setup involving two distinct versions of CharBERT: one trained on a dataset of historical English texts and the other on modern English texts, termed historical $\text{CharBERT}_{\text{SMALL}}$ and modern $\text{CharBERT}_{\text{SMALL}}$, respectively. These specialized models were then each combined with TrOCR to form composite systems designed to tackle the challenges of domain-specific text recognition.

Our primary objective is to evaluate how well these composite models perform when exposed to the George Washington (GW) dataset, which consists of historically significant documents with unique linguistic features that differ markedly from contemporary text forms. By assessing the performance of both TrOCR-historical $\text{CharBERT}_{\text{SMALL}}$ and TrOCR-modern $\text{CharBERT}_{\text{SMALL}}$ on this dataset, we aim to determine the extent of each model's ability to adapt to the domain-specific characteristics inherent in historical documents.

If both versions of the composite model—those incorporating historical and modern training—show comparable effectiveness on the GW dataset, it would not only validate the models' capability to transcend their training domains but also support the adaptability claims made in the Candidate Fusion paper. Such an outcome

would illustrate the potential of advanced language models, when combined with sophisticated OCR technologies, to provide flexible, domain-agnostic solutions that maintain high levels of accuracy across varied textual environments. This adaptability is crucial for deploying OCR systems in diverse real-world scenarios where the ability to understand and process different types of text is paramount.

4.3.4 TrOCR-CharBERT$_{\mathcal{P}_{ij}}$ Analysis

The primary objective of this analysis is to evaluate whether integrating common errors identified in TrOCR outputs into the training of CharBERT can improve the performance of the combined model. For this purpose, a new variant of CharBERT, referred to as CharBERT$_{\mathcal{P}ij}$, is trained specifically to address these identified inaccuracies.

To ensure a fair comparison of the enhanced capabilities of TrOCR-CharBERT$_{\mathcal{P}ij}$, it is inappropriate to compare it directly against a composite model using the pretrained standard CharBERT, which has not been adjusted for OCR-specific errors. Instead, we compare TrOCR-CharBERT$_{\mathcal{P}ij}$ to TrOCR-CharBERT$_{\text{SMALL}}$. This latter model employs the original CharBERT architecture but is trained on a more limited dataset. This comparison aims to isolate the impact of the targeted error correction training in CharBERT$_{\mathcal{P}_{ij}}$ and to demonstrate the potential improvements in OCR accuracy stemming from this bespoke training approach.

5 Results and Discussion

5.1 $\mathcal{P}_{ij}$ Matrix

As detailed in Subsection 3.4.1, the $\mathcal{P}_{ij}$ matrix encapsulates the probabilities of a correct character i being misrecognized as another character j. This matrix is a pivotal component in our analysis, providing a quantitative measure of the likelihood of error between different character transitions within the OCR process. Below, we present a subset of the matrix, highlighting the top 14 most commonly misrecognized characters. These entries have been selected due to their higher frequency of occurrence, which suggests significant areas where the CharBERT model may benefit from further training. The visual representation of these misrecognition probabilities, as shown in Figure 18, provides a clear and immediate insight into the error dynamics of the model.

5.2 Component Analysis and Benchmarking Results

As outlined in Subsection 4.4.1, several analytical experiments were conducted to optimize the model's performance. These experiments included comparing the effects of freezing specific layers to determine their impact on learning, integrating dropout mechanisms to enhance generalization, evaluating different combinations of tensor modules for optimal processing, assessing various loss functions to improve training efficiency, and benchmarking the model against GPT-4 to gauge its competitive standing.

5.2.1 Baseline Models

For the GW dataset, fine-tuning significantly improved model accuracy. When fine-tuning was not applied, the WER and CER were relatively high at 37.76% and 15.40%, respectively. However, after fine-tuning, these rates dropped to 14.44% for

WER and 4.78% for CER, showcasing a substantial enhancement in model performance.

Similarly, for the JH dataset, the benefits of fine-tuning were pronounced. Without fine-tuning, the model performed poorly, with a WER of 91.31% and a CER of 58.57%. Post fine-tuning, these figures improved to 36.97% for WER and 20.28% for CER, indicating a marked improvement in the model's ability to accurately transcribe text.

These results clearly demonstrate that fine-tuning has a significant positive effect on reducing error rates and thereby enhancing the overall accuracy of the model across different datasets.

Dataset	Fine-Tune	WER	CER	File Name
GW	False	37.76	15.40	gw
GW	True	14.44	4.78	gw_ft
JH	False	91.31	58.57	iam
JH	True	36.97	20.28	iam_ft

Table 2: Baseline Model Results

Output Texts Analysis - Over-Correction Upon examining the GW dataset TrOCR outputs without fine-tuning, it becomes apparent that TrOCR tends to over-correct the text. As illustrated below, TrOCR autocorrects "Expamples" (an original misspelling by George Washington) to "Examples," the correct form of the word. Additionally, it completes the truncated word "gene" as "general" presuming the word was inadvertently cut short.

```
Test on GW Dataset Without Fine-Tuning
label:  have a very good effect. The Commonalty in gene
output: have a very good effect. Then commonly in general

Test on GW Dataset Without Fine-Tuning
label:  est occasion for Expamples, will be morally im
output: cut occasion for Examples, will be morally in-
```

This tendency to over-correct is particularly noticeable at the end of sentences where the last word is truncated, a frequent occurrence in the George Washington (GW)

dataset. TrOCR often attempts to complete these cut-off words, or it may substitute them with a different word that, while seemingly appropriate, is irrelevant to the original token image. In some cases, TrOCR even transforms the incomplete word into a completely fabricated term.

Notably, this tendency to over-correct persists even after fine-tuning, indicating that TrOCR consistently aims to standardize text regardless of the context. In the case of the George Washington (GW) testing dataset, which includes 30 labels ending with unfinished words, the fine-tuned TrOCR correctly resolved only 5 of these instances. Often, it either attempted to complete them or altered them into entirely different words. In contrast, the best-performing integration of TrOCR with CharBERT correctly handled 10 of these cases, attempting to complete the word only once, compared to the eight times observed with the fine-tuned TrOCR. This improvement demonstrates that adding a language model (LM) to the end-to-end OCR system provides valuable access to both language and image information, enhancing the model's ability to process and preserve the textual integrity of historical documents. The distribution of these outcomes is depicted in Figure 19.

The findings from the TrOCR outputs highlight its inherent ability to correct and complete words based on its learned understanding from training data. While this functionality showcases TrOCR's robustness in rectifying spelling errors and incomplete entries, it also introduces potential inaccuracies when dealing with OCR tasks where original spellings and phrasings are crucial for authenticity. The model's tendency to standardize text could be a drawback in contexts where the preservation of original language nuances is necessary.

Output Texts Analysis - Poor Performance on JH Dataset The performance on the JH dataset is notably poor. This may be attributed, firstly, to Joseph Hooker's tendency to write in an ascending manner, which often results in cropping that inadvertently includes lines from adjacent text, as depicted in Figure 19a. Such cropping may distract the TrOCR from accurately recognizing the text. Secondly, the elegance of Joseph Hooker's handwriting, while aesthetically pleasing, presents readability challenges due to its cursive style and unique letter formations. Additionally, a few images (fewer than 10) contain printed rather than handwritten letters, as illustrated in Figure 19b. In this example, TrOCR recognized "THE Camp." instead of the correct label "THE CAMP." as shown below. Despite the fact that TrOCR's handwritten model can recognize printed letters, it struggles with correct capitalization. Although these instances are relatively rare, they still contribute to the overall error rate of the task.

```
Test on JH Dataset Without Fine-Tuning
label:  THE CAMP.
output: THE Camp.
```

5.2.2 Freezing Specific Layers

For the GW dataset, results showed significant variation depending on the configuration of frozen components. The lowest WER and CER were observed when both the TrOCR encoder and decoder were frozen. This suggests that the model performs optimally when only the components connecting CharBERT and TrOCR, as well as the CharBERT components themselves, are trainable. Other configurations involving freezing combinations did not yield as favorable outcomes.

Conversely, the JH dataset exhibited less variability in response to component freezing. Although freezing both the TrOCR encoder and decoder also yielded marginally better results compared to other configurations, the improvements were not as pronounced as those observed in the GW dataset. This indicates a dataset-specific response to the freezing of model components, highlighting the need for tailored strategies depending on the characteristics of the data being processed.

This experience underscore a significant variation in the performance of the TrOCR model across two datasets, GW and JH, in response to different component freezing configurations. For the GW dataset, the best performance—indicated by the lowest WER and CER—was achieved when both the TrOCR encoder and decoder were frozen, suggesting optimal functionality when only the CharBERT-related components are active. In contrast, the JH dataset demonstrated less sensitivity to component freezing, with marginal improvements observed when both the encoder and decoder were frozen, compared to other configurations. These results highlight the importance of customizing model training strategies to accommodate the specific attributes and requirements of different datasets.

Output Texts Analysis - TrOCR combined with CharBERT When TrOCR is combined with CharBERT, the enhanced model achieves more precise post-corrections. Unlike TrOCR alone, which may over-correct or erroneously complete words, this hybrid approach maintains the authenticity of the original images. For example, as discussed in Subsection 5.3.1, TrOCR incorrectly extends “gene” to “general” and correct “Expamples” to “Examples.” By integrating CharBERT, the model leverages both visual information and linguistic knowledge, enabling it to make more informed decisions about when to amend text and when to preserve the original

Dataset	Freeze	WER	CER	File Name
GW	None	17.13	10.17	gw_none
GW	TrOCR encoder	25.09	15.39	gw_encoder
GW	TrOCR decoder	17.46	10.49	gw_decoder
GW	TrOCR encoder, decoder	12.84	5.88	gw_01
JH	None	36.06	22.11	iam_none
JH	TrOCR encoder	34.74	22.32	iam_encoder
JH	TrOCR decoder	35.17	21.95	iam_decoder
JH	TrOCR encoder, decoder	35.63	21.90	iam_encoder_decoder

Table 3: Freezing Specific Layers Results

input. This is particularly effective in the JH dataset, where the combined model correctly recognizes "THE CAMP.," accurately handling printed letters without the capitalization errors seen with TrOCR alone.

```
Test on GW Dataset with Encoder, Decoder Freezed
label:  have a very good effect. The Commonalty in gene
output: have a very good effect. The Commonalty in gene

Test on GW Dataset with Encoder, Decoder Freezed
label:  est occasion for Expamples, will be morally im
output: est occasion for Expamples, will be morally im

Test on JH Dataset with Encoder, Decoder Freezed
label:  THE CAMP.
output: THE CAMP.
```

Output Texts Analysis - Effects on Freezing Different Layers Although the integration of TrOCR with CharBERT significantly enhances OCR and post-OCR correction tasks, these benefits are primarily observed when the encoder and decoder are frozen. Models with other component configurations perform suboptimally. They do not over-correct or complete sentences as TrOCR alone does, yet the accuracy of their output remains questionable, as demonstrated by the following

examples:

```
Test on GW Dataset with None Freezed
label:  have a very good effect. The Commonalty in gene
output: have a very good effect. The Commonwealthalty in get

Test on GW Dataset with Encoder Freezed
label:  have a very good effect. The Commonalty in gene
output: have a very good effect. The Commally in g

Test on JH Dataset with None Freezed
label:  & education has extended to
output: & observations has returned to
```

This discrepancy may stem from several factors:

Feature Stability Freezing the encoder and decoder in TrOCR maintains the stability of the features extracted from images. As a transformer-based OCR model, TrOCR is adept at effectively extracting text. By freezing these components, the pre-trained weights, which are proficient at capturing crucial image features, remain unaltered during additional training with CharBERT. This preservation is essential for accurate OCR.

Avoiding Catastrophic Forgetting In fine-tuning deep learning models for new tasks, there is a risk of catastrophic forgetting, where the model loses its previously learned knowledge—here, its OCR capabilities. Freezing the encoder and decoder helps prevent this loss.

Focus on Language Understanding With the OCR components frozen, CharBERT primarily enhances language understanding and error correction, based on the context it detects. CharBERT, designed to handle textual inaccuracies and contextual understanding, thus focuses on refining the text output.

Reducing Overfitting Allowing the TrOCR encoder and decoder to be trainable may lead to overfitting, where the model becomes excessively specialized to the training data nuances, compromising its ability to generalize. Freezing mitigates this risk.

Balanced Training Dynamics Activating both components could cause CharBERT's gradient during backpropagation to negatively impact TrOCR's feature extraction, degrading core OCR functionality. Freezing focuses optimization on linguistic corrections without undermining OCR capabilities.

5.2.3 Integrating Dropout Mechanisms

For both datasets, varying the dropout rates demonstrated minimal impact on the model's performance. However, the GW dataset exhibited a more pronounced response to changes in dropout rates compared to the JH dataset. This suggests that the influence of dropout is dataset-specific, highlighting the necessity for tailored dropout configurations that are carefully calibrated based on the characteristics of the dataset and the desired outcomes of the model. This approach ensures optimized performance while addressing the unique demands of each dataset.

Dataset	Dropout Rate	WER	CER	File Name
GW	-	13.11	5.78	gw_00
GW	0.1	12.84	5.88	gw_01
GW	0.2	12.88	6.26	gw_02
GW	0.5	13.80	6.39	gw_05
JH	-	35.30	21.33	iam_00
JH	0.1	35.63	21.90	iam_encoder_decoder
JH	0.2	35.13	22.76	iam_02
JH	0.5	35.30	22.77	iam_05

Table 4: Integrating Dropout Mechanisms Results

Output Texts Analysis - Effects of Different Dropout Rates The impact of varying dropout rates on the output texts is evident, although it is challenging to determine which rate enhances language understanding the most. Each rate results in some correct and some incorrect recognitions, suggesting a nuanced effect on model performance. This observation may be attributed to the model's architecture. Both TrOCR and CharBERT, being based on transformer architecture and designed for error correction, respectively, likely exhibit a high degree of inherent robustness against overfitting due to their sophisticated designs and pre-trained weights. As

dropout is primarily a tool to prevent overfitting, its contribution may be less significant if the model already generalizes well. Moreover, given that TrOCR and CharBERT are tasked with interpreting complex dependencies and contextual nuances, the random deactivation of neurons through dropout could disrupt more than assist. This disruption is particularly problematic in tasks requiring the interaction of all neurons to capture complex data patterns. Notably, lower dropout rates (such as 0 or 0.1) tend to perform better compared to higher rates (0.2 and 0.5), possibly due to these factors.

5.2.4 Comparing Combined Tensor Modules

For both datasets, Module 3 performs sligthly better than Module 1 and Module 4, but significantly better than Module 2.

These results indicate that the effectiveness of specific tensor modules varies significantly between the two datasets, with certain configurations like linear layers as attention mechanisms consistently outperforming others such as mean pooling. This variability underscores the importance of selecting tensor modules based on dataset-specific characteristics to optimize model performance. The study clearly illustrates the need for meticulous configuration and testing of different components within OCR models to achieve optimal results.

Dataset	Combined Tensor Modules	WER	CER	File Name
GW	Module 1	13.63	6.61	gw_c1
GW	Module 2	30.66	18.85	gw_c2
GW	Module 3	12.84	5.88	gw_01
GW	Module 4	13.31	6.29	gw_c4
JH	Module 1	34.52	20.61	iam_c1
JH	Module 2	44.99	30.14	iam_c2
JH	Module 3	35.63	21.90	iam_encoder_decoder
JH	Module 4	43.00	25.24	iam_c4

Table 5: Comparing Combined Tensor Modules Results

Output Texts Analysis - Effects of Different Combined Tensor Modules In the comparative analysis of tensor combine modules applied to the GW and JH datasets, it is evident that Module 2 consistently underperforms. This module, which utilizes mean pooling to dynamically allocate attention weights across the embedding dimensions of stacked tensors, simplifies the model by focusing on average features. However, this approach likely leads to an oversimplification of the representation, obscuring critical nuances and distinct handwriting traits that are essential for accurate character recognition. The inherent simplicity of mean pooling fails to capture the complex variations and subtleties in historical handwritten texts, resulting in suboptimal performance.

Conversely, Module 3, which employs linear layers for attention-based feature weighting, excels with the GW dataset. This module's sophisticated attention mechanism adaptively enhances and focuses on relevant features, which proves particularly effective for the GW dataset. The dataset, characterized by subtle variations in handwriting style from multiple authors, benefits from the nuanced differentiation that the linear layers provide, thereby improving both accuracy and robustness.

On the other hand, the JH dataset shows the best performance with Module 1, which uses a simple element-wise addition. This outcome can be attributed to the inherent variability and less consistent text layouts within the JH dataset. The straightforward additive approach of Module 1, though simpler, may inadvertently offer robustness by avoiding over-emphasis on any particular feature set, thus maintaining a general representation that accommodates the diverse handwriting styles found in the dataset.

Module 4, which replaces linear layers with convolutional layers, aims to exploit spatial correlations within the data. Although this strategy is typically effective for input types where spatial relationships are key, such as images, its benefits may not translate as effectively to tensor combinations derived from text embeddings.

This analysis highlights the critical importance of tailoring the tensor combining strategy to match the specific characteristics and challenges of each dataset. Module 3's adaptive capabilities are ideally suited to datasets like GW, which require nuanced differentiation, while the simpler, more robust approach of Module 1 is more appropriate for handling the varied and unpredictable features of the JH dataset. This strategic alignment is essential for optimizing model performance across different handwriting analysis tasks.

5.2.5 Comparing Loss Functions

The table titled "Comparing Loss Functions Results" provides a detailed analysis of how three different loss functions impact the performance of OCR systems across two distinct datasets: the George Washington (GW) and Joseph Hooker (JH) datasets. For each dataset, the effectiveness of each loss function is evaluated based on the Word Error Rate (WER) and Character Error Rate (CER). Notably, Loss Function 1 appears to yield the lowest WER and CER in the GW dataset, suggesting it may be the most effective of the three for this particular set of data. However, the performance varies significantly in the JH dataset, where the error rates are generally higher, indicating a greater challenge in character recognition or possibly the complexity of the dataset itself. This comparison is crucial for understanding which loss functions might provide the best balance between word and character recognition accuracy in different OCR applications.

Dataset	Loss Function	WER	CER	File Name
GW	1	12.84	5.88	gw_01
GW	2	13.30	5.81	gw_loss2
GW	3	14.19	6.85	gw_loss3
JH	1	35.63	21.90	iam_encoder_decoder
JH	2	44.71	26.20	iam_loss2
JH	3	40.72	27.92	iam_loss3

Table 6: Comparing Loss Functions Results

5.2.6 Benchmarking Against GPT-4

Model	WER	CER
TrOCR-CharBERT		
GPT-4 Post Correction		
GPT-4 Image Recognition		

Table 7: Benchmarking Against GPT-4 Results

5.3 Model Cross-Dataset Performance Results

The model's performance when trained on the George Washington (GW) dataset and tested on the Joseph Hooker (JH) dataset highlights a significant domain shift that the model struggles to manage effectively. The limited variability or the non-generalizable nature of the features learned from the GW dataset results in poor adaptability to the JH dataset. Conversely, the model trained on the JH dataset, although not excelling, performs better when tested on the GW dataset. These results suggest that the JH dataset likely offers a richer and more varied range of handwriting styles and complexities, making models trained on it more robust when applied to different datasets.

This inference aligns with earlier discussions in subsection 5.2.4, suggesting that the JH dataset's complexity and diversity might explain why Module 3 do not perform as well on it. In contrast, the GW dataset appears to be more homogeneous and specific in style, which hampers its generalization capabilities when confronted with the varied and complex data characteristics of the JH dataset. This analysis underscores the need for OCR models that can adapt across significantly diverse handwriting datasets and highlights the importance of incorporating a broad spectrum of data characteristics during the training phase to enhance model robustness and versatility.

Training Dataset	Testing Dataset	WER	CER	File Name
GW	JH	96.96	82.64	gw_adapt
JH	GW	40.30	27.23	iam_adapt

Table 8: Model Cross-Dataset Performance Results

5.4 Model Domain Adaptability Results

5.5 TrOCR-CharBERT$_{\mathcal{P}_{ij}}$ Results

Training Dataset	LM Training Data	WER	CER	File Name
GW	Contemporary English	13.88	6.51	gw_small
GW	15^{th} – 18^{th} Century English			gw_pij
JH	Contemporary English			iam_small
JH	15^{th} – 18^{th} Century English			iam_pij

Table 9: Model Domain Adaptability Results

Training Dataset	Model	$\mathcal{P}_{ij}$	WER	CER	File Name
GW	TrOCR_CharBERT$_{\text{SMALL}}$	False	13.88	6.51	gw_small
GW	TrOCR_CharBERT$_{\mathcal{P}_{ij}}$	True	12.94	6.03	gw_pij
JH	TrOCR_CharBERT$_{\text{SMALL}}$	False			iam_small
JH	TrOCR_CharBERT$_{\mathcal{P}_{ij}}$	True			iam_pij

Table 10: TrOCR-CharBERT$_{\mathcal{P}_{ij}}$ Results

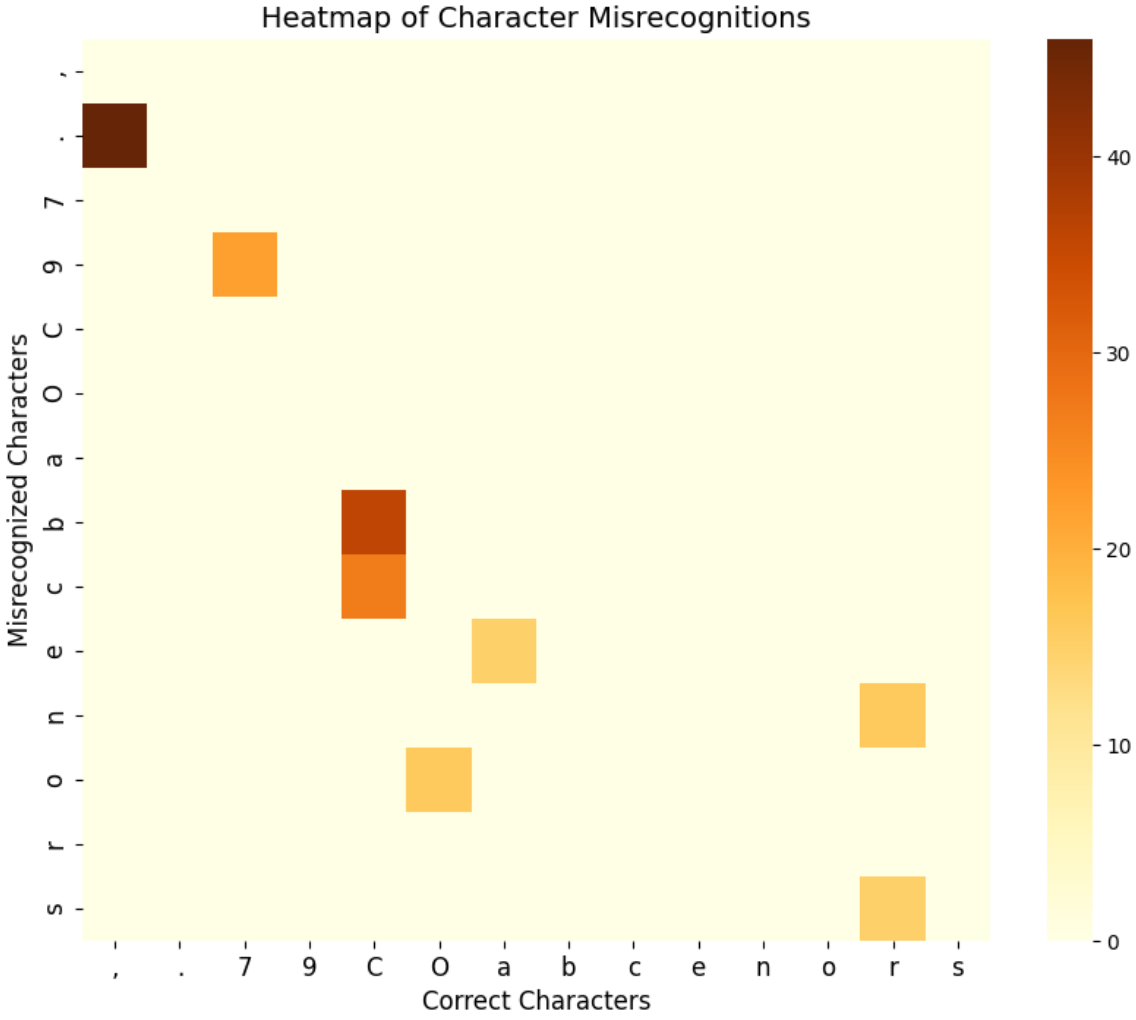


Figure 18: **Heatmap of Character Misrecognitions $\mathcal{P}_{ij}$**
This visual representation illustrates the frequency of common misrecognitions between various characters, as outlined in our study. The x-axis represents the correct characters, and the y-axis shows the characters as they were misrecognized by the OCR system. The color gradient, ranging from light yellow to dark brown, indicates the relative frequency of each misrecognition event, with darker shades representing higher frequencies. This analysis helps identify specific characters that are prone to misrecognition, facilitating targeted improvements in the OCR model.

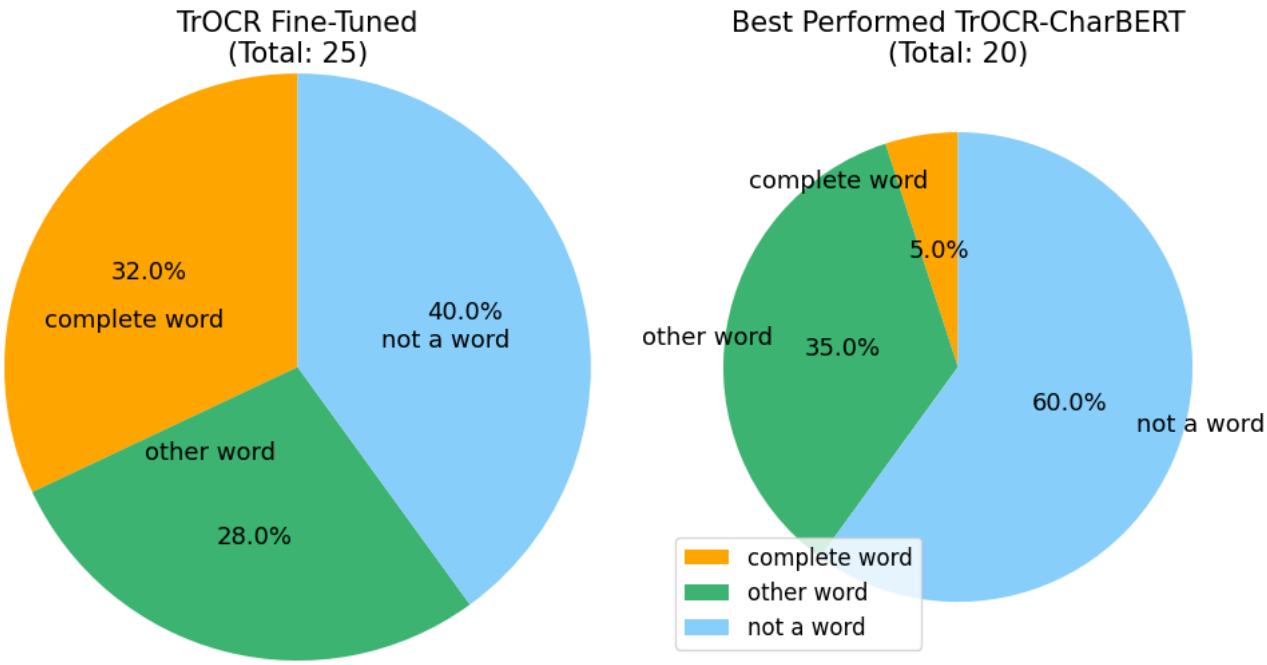
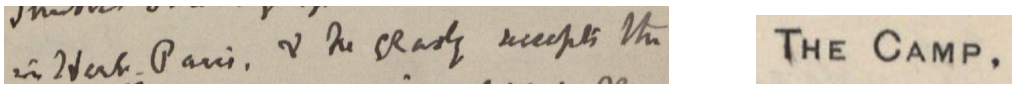


Figure 19: **Comparison of Outcomes for TrOCR Fine-Tuned vs. Best Performed TrOCR-CharBERT**
This figure illustrates the percentage of outcomes for unfinished word scenarios within the GW dataset, comparing the fine-tuned TrOCR and the TrOCR-CharBERT. The categories "Complete word," "Other word," and "Not a word" indicate whether the model attempted to complete the unfinished words, substituted them with a different word it deemed fit, or transformed them into non-words, respectively. The pie charts reveal that the integration with CharBERT significantly reduces the instances of attempting to complete words erroneously, demonstrating its ability to more accurately preserve the original text integrity.



(a) (b)

Figure 19: **JH Dataset Poor Image Examples**
The subpar performance on the JH dataset is largely due to Joseph Hooker's ascending handwriting style, which often leads to line cropping and text overlap, as shown in Figure a. This can interfere with accurate text recognition. Moreover, Hooker's cursive handwriting, while visually appealing, is challenging to read due to unique letter formations. A few instances of printed text, rather than handwritten, are also present, shown in Figure b, contributing to the overall error rate.

6 Conclusion

In this project we have done so much.[1]

We could show that ...

6.1 Future Work

[1] Thanks to many people that helped me.

Glossary

Of course there are plenty of glossaries out there! One (not too serious) example is the online MT glossary of Kevin Knight [2] in which MT itself is defined as

> techniques for allowing construction workers and architects from all over the world to communicate better with each other so they can get back to work on that really tall tower.

accuracy A basic score for evaluating automatic **annotation tools** such as **parsers** or **part-of-speech taggers**. It is equal to the number of **tokens** correctly tagged, divided by the total number of tokens. [...]. (See **precision and recall**.)

clitic A morpheme that has the syntactic characteristics of a word, but is phonologically and lexically bound to another word, for example *n't* in the word *hasn't*. Possessive forms can also be clitics, e.g. The dog*'s* dinner. When **part-of-speech tagging** is carried out on a corpus, clitics are often separated from the word they are joined to.

[2] Machine Translation Glossary (Kevin Knight): http://www.isi.edu/natural-language/people/dvl.html

References

C. Amrhein and S. Clematide. Supervised OCR error detection and correction using statistical and neural machine translation methods. *Journal for Language Technology and Computational Linguistics (JLCL)*, 33(1):49–76, 2018.

H. Bao, L. Dong, S. Piao, and F. Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=p-BhZSz59o4.

B. Belay, T. Habtegebrial, M. Meshesha, M. Liwicki, G. Belay, and D. Stricker. Amharic OCR: an end-to-end learning. *Applied Sciences*, 10(3):1117, 2020.

T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait. High-performance OCR for printed English and Fraktur using LSTM networks. In *2013 12th international conference on document analysis and recognition*, pages 683–687. IEEE, 2013.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. URL https://aclanthology.org/N19-1423.

R. Dey and F. M. Salem. Gate-variants of gated recurrent unit (GRU) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

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. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar. Improving CNN-RNN hybrid networks for handwriting recognition. In *2018 16th international conference on frontiers in handwriting recognition (ICFHR)*, pages 80–85. IEEE, 2018.

M. Ehrmann, A. Hamdi, E. L. Pontes, M. Romanello, and A. Doucet. Named entity recognition and classification in historical documents: A survey. *ACM Computing Surveys*, 56(2):1–47, 2023.

A. Fischer, A. Keller, V. Frinken, and H. Bunke. Lexicon-free handwritten word spotting using character HMMs. *Pattern recognition letters*, 33(7):934–942, 2012.

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*, pages 369–376, 2006.

E. Grosicki and H. El-Abed. ICDAR 2011-French handwriting recognition competition. In *2011 International Conference on Document Analysis and Recognition*, pages 1459–1463. IEEE, 2011.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

J. Huang, G. Pang, R. Kovvuri, M. Toh, K. J. Liang, P. Krishnan, X. Yin, and T. Hassner. A multiplexed network for end-to-end, multilingual OCR. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4547–4557, 2021.

Z. Huang, K. Chen, J. He, X. Bai, D. Karatzas, S. Lu, and C. Jawahar. ICDAR 2019 competition on scanned receipt OCR and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE, 2019.

L. Kang, P. Riba, M. Villegas, A. Fornés, and M. Rusiñol. Candidate fusion: Integrating language modelling into a sequence-to-sequence handwritten word recognition architecture. *Pattern Recognition*, 112:107790, 2021.

K. Karthick, K. Ravindrakumar, R. Francis, and S. Ilankannan. Steps involved in text recognition and recent research in OCR; a study. *International Journal of Recent Technology and Engineering*, 8(1):2277–3878, 2019.

S. Karthikeyan, A. G. S. de Herrera, F. Doctor, and A. Mirza. An OCR post-correction approach using deep learning for processing medical reports.

IEEE Transactions on Circuits and Systems for Video Technology, 32(5): 2574–2581, 2021.

K. Kettunen, H. Keskustalo, S. Kumpulainen, T. Pääkkönen, and J. Rautiainen. OCR quality affects perceived usefulness of historical newspaper clippings – a user study. In *Proceedings of the 18th Italian Research Conference on Digital Libraries (IRCDL 2022)*, CEUR-WS.org, Padova, Italy, 2022. CEUR Workshop Proceedings. URL https://ceur-ws.org/Vol-3160/paper1.pdf.

D. Klakow and J. Peters. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1-2):19–28, 2002.

B. Kurar Barakat, R. Cohen, A. Droby, I. Rabaev, and J. El-Sana. Learning-free text line segmentation for historical handwritten documents. *Applied Sciences*, 10(22):8276, 2020.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13094–13102, 2023.

D. Liu and J. Yu. Otsu method and K-means. In *2009 Ninth International conference on hybrid intelligent systems*, volume 1, pages 344–349. IEEE, 2009.

W. Liu, C. Chen, K.-Y. K. Wong, Z. Su, and J. Han. Star-net: a spatial attention residue network for scene text recognition. In *BMVC*, volume 2, page 7, 2016.

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. *arXiv preprint arXiv:1907.11692*, 2019.

W. Ma, Y. Cui, C. Si, T. Liu, S. Wang, and G. Hu. CharBERT: Character-aware pre-trained language model. In D. Scott, N. Bel, and C. Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online), Dec. 2020. International Committee on Computational Linguistics. URL https://aclanthology.org/2020.coling-main.4.

U.-V. Marti and H. Bunke. A full English sentence database for off-line handwriting recognition. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition. ICDAR'99 (Cat. No. PR00318)*, pages 705–708. IEEE, 1999.

K. Mokhtar, S. S. Bukhari, and A. Dengel. OCR error correction: State-of-the-art vs an NMT-based approach. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 429–434. IEEE, 2018.

C. Neudecker, K. Baierer, M. Federbusch, M. Boenig, K.-M. Würzner, V. Hartmann, and E. Herrmann. OCR-D: An end-to-end open source OCR framework for historical printed documents. In *Proceedings of the 3rd international conference on digital access to textual cultural heritage*, pages 53–58, 2019.

T. T. H. Nguyen, A. Jatowt, M. Coustaty, and A. Doucet. Survey of post-OCR processing approaches. *ACM Computing Surveys (CSUR)*, 54(6):1–37, 2021.

J. Philips and N. Tabrizi. Historical document processing: A survey of techniques, tools, and trends. *KDIR*, pages 341–349, 2020.

A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. OpenAI White Paper, 2018. URL https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf.

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. In K. Erk and N. A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. URL https://aclanthology.org/P16-1162.

B. Shi, M. Yang, X. Wang, P. Lyu, C. Yao, and X. Bai. ASTER: An attentional scene text recognizer with flexible rectification. *IEEE transactions on pattern analysis and machine intelligence*, 41(9):2035–2048, 2018.

A. Singh, K. Bacchuwar, and A. Bhasin. A survey of OCR applications. *International Journal of Machine Learning and Computing*, 2(3):314, 2012.

J. J. Smith. *Essentials of early English: Old, middle and early modern English*. Routledge, 2013.

R. Smith. An overview of the tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)*, volume 2, pages 629–633. IEEE, 2007.

H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh. CSPNet: A new backbone that can enhance learning capability of CNN. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020a.

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. *Advances in Neural Information Processing Systems*, 33: 5776–5788, 2020b.

Y.-Y. Wang, A. Acero, and C. Chelba. Is word error rate a good indicator for spoken language understanding accuracy. In *2003 IEEE workshop on automatic speech recognition and understanding (IEEE Cat. No. 03EX721)*, pages 577–582. IEEE, 2003.

H. Zhang, D. Liu, and Z. Xiong. CNN-based text image super-resolution tailored for OCR. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, pages 1–4. IEEE, 2017.

Curriculum vitae

Personal Information

Yung-Hsin Chen

yung-hsin.chen@uzh.ch

Education

2016-2020	Bachelor's degree in Physics at National Tsing-Hua University (NTHU)
since 2020	Master's degree in Informatics at University of Zurich (UZH)

Part-time Activities and Internships

2021	Delta Electronics Inc. Data Scientist Intern
2022-2023	Swiss Re Data Analyst

A Tables

Part of speech	POS type	POS	number of labels in my corpus
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	DET	**35**	280
14	Total	**35**	280

Table 11: Some very large table in the appendix

B List of something

This appendix contains a list of things I used for my work.

- apples
  - `export2someformat`
- bananas
- oranges
  - `bleu4orange`
  - `rouge2orange`