B.Comp. Dissertation

Benchmarking and Improving OCR Systems for Southeast Asian Languages

By

Qiu Jiasheng, Jason

Department of Computer Science

School of Computing

National University of Singapore

2024/2025

B.Comp. Dissertation

Benchmarking and Improving OCR Systems for Southeast Asian Languages

By

Qiu Jiasheng, Jason

Department of Computer Science

School of Computing

National University of Singapore

2024/2025

Project ID: H0792230

Supervisor: A/P Min-Yen Kan

Advisor: Tongyao Zhu

Deliverables:

Report: 1 Volume

Abstract

While Optical Character Recognition (OCR) has been widely studied for high-resource languages such as English and Chinese, the efficacy and limitations of OCR models on Southeast Asian (SEA) languages remain largely unexplored. This study aims to bridge this gap by evaluating OCR technologies for SEA languages and exploring script-specific challenges. We propose a pipeline to collect textual data from Wikipedia and benchmark open-source OCR tools. Additionally, we demonstrate the potential of fine-tuning existing models on SEA languages, aiming to expand OCR capabilities for these languages.

Subject Descriptors:

H.3.3 Information Search and Retrieval

I.2.7 Natural Language Processing

I.2.10 Vision and Scene Understanding

Keywords:

Optical Character Recognition, Southeast Asian Languages, Fine-tuning

Implementation Software and Hardware:

Python, Tesseract, EasyOCR, General OCR Theory

Acknowledgements

I would like to thank my supervisor, A/P Kan Min-Yen, and my advisor, Tongyao Zhu, for their invaluable guidance and mentorship. Their encouragement and constructive guidance have been a significant source of inspiration throughout the project.

List of Figures

List of Tables

Table of Contents

Chapter 1

Introduction

Current research in Natural Language Processing (NLP) is heavily concentrated on only 20 of the 7,000 languages in the world (Magueresse et al., 2020). In particular, Southeast Asia (SEA) is home to over 1,000 languages but remains a relatively under-researched region in NLP (Aji et al., 2023). A similar trend can be observed in Optical Character Recognition (OCR) research, where the focus is predominantly on high-resource languages (Salehudin et al., 2023; R. Smith, 2007), leaving many SEA languages underserved.

OCR, the process of converting textual images into machine-readable formats, offers significant potential for languages with limited datasets. While many scanned documents and books in these low-resource languages are available online, the text within them often remains inaccessible due to formats like images and PDFs. By extracting the text from these documents, OCR can generate valuable datasets for low-resource languages, which can then be used for downstream NLP tasks, such as machine translation and named-entity recognition (Agarwal & Anastasopoulos, 2024; Ignat et al., 2022). Therefore, studying OCR performance on SEA languages is crucial to accelerating NLP research and technology development in the region.

While OCR has been widely studied for high-resource languages such as English and Chinese, the efficacy and limitations of OCR models on SEA languages remain largely unexplored. To address this gap, this study presents a pipeline to collect textual data from Wikipedia and benchmark several open-source OCR tools on the collected data.

Additionally, we explore the potential of fine-tuning existing models to improve OCR performance on SEA languages. The primary objective is to evaluate and enhance the performance of OCR technologies on SEA languages, thereby advancing NLP applications in this linguistically diverse region.

Specifically, this project seeks to answer the following research questions (RQs):

- **RQ1:** How do popular OCR tools perform on SEA scripts?
- **RQ2:** What script-related challenges affect OCR accuracy on SEA languages?
- **RQ3:** How can fine-tuning enhance OCR accuracy on SEA languages?

Chapter 2

Related Work

2.1 Overview of OCR Systems

Most OCR systems consist of two stages: text detection and text transcription. Text detection identifies text present in an image and extracts cropped regions containing the detected text. A text transcription model then converts these cropped images into text. Generally, separate models are used for each stage, allowing for greater training flexibility and a clearer understanding of challenges within each component (Subramani et al., 2020). More recently, end-to-end models that combine both stages have shown promise in reducing errors for certain use cases (Feng et al., 2019).

2.1.1 Evolution of OCR Models

Early OCR models employ traditional machine learning techniques, such as K-nearest Neighbors (KNN) and Support Vector Machines (SVMs), to classify textual characters from cropped images. Tesseract, an established OCR engine developed since the 1990s, recognizes character patterns by extracting small fragments of character outlines as features (R. W. Smith, 2013). These features are then classified into character clusters using an optimized KNN algorithm. While effective for structured text, these traditional approaches struggled with variations in handwriting, fonts, and image distortions (Subramani et al., 2020).

The rise of deep learning brought significant advancements in OCR. Convolutional Neural Networks (CNNs) improve feature extraction by automatically detecting edges, textures, and shapes within text images. Unlike traditional handcrafted features, CNNs learn visual patterns by applying small filters across an image. The Character Region Awareness for Text Detection (CRAFT) algorithm, for example, uses a fully convolutional network to achieve state-of-the-art character localization (Baek et al., 2019). For text transcription, Recurrent Neural Networks (RNNs) have been widely adopted due to their ability to model sequential dependencies over time. Tesseract v4 integrated a Long Short-Term Memory (LSTM) model, a specialized type of RNN, to recognize entire lines of text instead of individual characters (Tesseract OCR, 2025). By combining CNNs for feature extraction and RNNs for sequence modeling, Shi et al. (2015) proposed the Convolutional Recurrent Neural Network (CRNN), which significantly improved text recognition accuracy in end-to-end OCR systems.

More recently, transformer-based models have emerged as a powerful alternative. Unlike CNNs and RNNs, transformers process entire input sequences in parallel using self-attention mechanisms, which allows them to capture long-range dependencies in text images more efficiently (Vaswani et al., 2017). This approach avoids image-specific inductive biases present in CNNs, such as the assumption that neighboring pixels are relevant. TrOCR, an end-to-end model that combines an image transformer and a separate text transformer, demonstrates another advantage of transformers: the ability to leverage self-supervised pre-training (M. Li et al., 2021). Since transformers can be pre-trained individually to learn useful patterns from unlabeled images and text, there is less reliance on manually annotated OCR training data to achieve high accuracy. Going beyond traditional text recognition, General OCR Theory (GOT) is another transformer-based

model that extends character recognition capabilities to non-text elements, such as sheet music, charts, and geometric shapes (Wei et al., 2024). By integrating Large Visual-Language Models (LVLMs), GOT seeks to address the bottlenecks of traditional OCR systems, which often struggle with generalization. As transformer-based OCR continues to evolve, these models are expected to push the boundaries of text recognition, enabling more flexible and adaptable OCR systems for diverse applications.

While deep learning techniques for OCR have been widely studied, there is still limited research focused on the performance of transformer-based OCR models, particularly for SEA languages. Our research compares the performance of various OCR tools, including those using transformer-based architectures, on SEA languages. In doing so, we aim to explore how these models handle text recognition in more complex and diverse linguistic contexts.

2.2 Benchmarking OCR on Low-resource Languages

To evaluate OCR performance accurately, textual data in the form of images or PDFs paired with reliable ground truth is essential. Similar to most NLP tasks, data scarcity poses a major obstacle to advancing OCR technology in low-resource languages. The limited availability of annotated textual data restricts both model training and evaluation, leading to disparities in OCR accuracy across different scripts. OCR tools generally perform better on Latin-based scripts (Hegghammer, 2022), partly due to market incentives that prioritize the development of English-language OCR systems, resulting in more extensive training data and refinement. Beyond data availability, the complexity of scripts with ornate diacritics or unique letter shapes often yield lower OCR accuracy (Agarwal & Anastasopoulos, 2024).

A recent study by Ignat et al. (2022) provides the most relevant benchmarking of OCR on SEA languages. Their benchmark grouped 60 low-resource languages by region and script, including SEA languages such as Khmer, Lao, Burmese, Thai, and Vietnamese. They found that OCR tools perform best on Latin and Cyrillic scripts, with only average performance on SEA languages, supporting Hegghammer (2022)'s findings. Additionally, Ignat et al. (2022) showed that while OCR models perform well on synthetic SEA-language data, their accuracy drops significantly on real-world data. This discrepancy underscores the need for more diverse and realistic training datasets to improve OCR outcomes for SEA languages. Our research aims to address this gap by developing a reusable pipeline for both collecting real-world digital data and generating synthetic data.

2.3 Using Synthetic Data for OCR Evaluation

To bridge the gap in data availability, many studies rely on artificial images and PDFs generated from plain text to create evaluation datasets. For instance, Ignat et al. (2022) generated synthetic PDFs from the Flores 101 dataset, which consists of text from Wikipedia in 101 languages. Expanding on this approach, Gupte et al. (2021) developed an open-source Python package that creates document images from plain text, incorporating several document styling templates. These methods enable the large-scale generation of high-quality, low-resource language data with corresponding ground truth annotations.

However, one challenge with artificial datasets is their tendency to lack the imperfections found in real-world documents. Real-world scanned documents often feature complex layouts, stains, and handwritten scribbles (Hegghammer, 2022). Studies have shown that OCR systems often perform better on synthetic datasets than on real-world

data, highlighting a gap in generalization (Ignat et al., 2022). To address this, researchers frequently apply noise augmentation to synthetic documents. Common techniques include changing the font style, size, color, and letter spacing, as well as adding Gaussian blur, bleed-through effects, and salt-and-pepper noise (Gupte et al., 2021; Ignat et al., 2022). These modifications help artificial datasets to better approximate the challenges of real-world OCR tasks. Our study adopts similar techniques by generating and benchmarking on synthetic data with noise, aiming to better simulate the imperfections found in real-world documents.

2.4 Fine-tuning OCR Systems

To enhance OCR performance in new domains with limited labeled data, many studies explore fine-tuning, or further training pre-trained models on a smaller, task-specific dataset. Instead of training from scratch, fine-tuning updates a model's existing weights, allowing it to adapt to new datasets while retaining prior knowledge. For instance, Parres and Paredes (2023) demonstrated that transformer-based models can successfully adapt to new languages and historical documents with minimal training data, achieving competitive OCR performance. Similarly, Laurent and Lauar (2024) fine-tuned the English TrOCR model for Spanish text, yielding strong results. Fine-tuning thus offers a promising strategy for our research to overcome the scarcity of labeled data in low-resource languages while achieving high accuracy.

Chapter 3

Methodology

To answer the research questions, this study conducted the following three experiments to benchmark and improve OCR performance on SEA languages:

- **Experiment 1:** Benchmarking on Real-world Wikipedia Screenshots
- **Experiment 2:** Benchmarking on Synthetic Data
- **Experiment 3:** Fine-tuning for Vietnamese and Thai

3.1 Experiment Setup

3.1.1 Languages

In this study, we chose to benchmark on English, Indonesian, Vietnamese, and Thai. English serves as a baseline comparison due to its extensive OCR research and established tool support. Meanwhile, Indonesian, Vietnamese, and Thai were selected as a representative subset of SEA languages for several reasons.

Firstly, these three languages encompass a range of script types: Latin scripts for Indonesian, Latin scripts with diacritics for Vietnamese, and Brahmic scripts for Thai. By covering these scripts, we capture a broad spectrum of orthographic features, such as diacritics, tone marks, and complex character shapes. This allows us to examine

how these unique linguistic features impact OCR performance. Furthermore, many other SEA languages, including Malay, Filipino, and Cebuano, use modified Latin scripts, while languages like Khmer, Burmese, and Javanese use Brahmic scripts. Thus, findings from this study can be applied to other languages with similar script types, accelerating OCR research in the region.

Table 3.1: Benchmarked Languages

	Speaker Population	Script Type	Example
English	1.5 billion	Latin	Good morning
Indonesian	252 million	Latin	Selamat pagi
Vietnamese	97 million	Latin with diacritics	Chào buổi sáng
Thai	71 million	Brahmic	**สวัสดีตอนเช้า**

Note: Speaker population data from Wikipedia (2025).

Secondly, the wide usage of these languages makes it feasible to obtain textual data. The high number of speakers, active online communities, and abundant digital content ensure sufficient resources for OCR benchmarking. Their prominence in SEA further highlights their relevance, as improving OCR for these languages benefits a large portion of the region's population.

While this study covers only a small fraction of the languages spoken in SEA, the selection of these languages provides a strong starting point, as they cover popular script types and offer abundant online data for benchmarking.

3.1.2 Data Source

To collect textual data, this study uses Wikipedia due to its accessibility and multilingual scope. Wikipedia articles can be converted into images via screenshots, simulating real-world OCR scenarios. The platform also offers a convenient Application Programming

Interface (API) that allows retrieval of plain text from most articles, serving as a reliable reference for evaluating OCR accuracy and generating synthetic documents. Moreover, the availability of large corpora in various SEA languages, including Thai, Vietnamese, Indonesian, Tamil, and Burmese, makes Wikipedia suitable for this study's language needs ("List of Wikipedias", 2024).

3.1.3 OCR Systems

In our selection of OCR systems for benchmarking, we prioritize open-source solutions that support a diverse range of SEA languages, promoting accessibility and reusability for the proposed evaluation pipeline. Additionally, we aim to include models with different underlying architectures, enabling a more comprehensive assessment of their performance across different languages. Consequently, we selected EasyOCR, Tesseract, and General OCR Theory (GOT), each open-source and representing distinct modeling approaches to OCR.

Table 3.2: Benchmarked OCR Systems

	Architecture	# Supported Languages
EasyOCR	CRAFT + CRNN	83 (includes all benchmarked languages)
Tesseract	LSTM	116 (includes all benchmarked languages)
GOT	VED	2 (English and Simplified Chinese)

EasyOCR is a modern OCR framework that integrates a text detection model based on the Character Region Awareness for Text Detection (CRAFT) algorithm with a recognition model utilizing a Convolutional Recurrent Neural Network (CRNN) (Jaided AI, 2025). Readily available as a Python package, EasyOCR supports 83 languages, including English, Indonesian, Vietnamese, and Thai.

Tesseract is one of the most well-known open-source OCR engines. Since releasing version 4 in 2018, Tesseract uses an underlying Long Short-Term Memory (LSTM) model for line recognition (Tesseract OCR, 2025). Similar to EasyOCR, Tesseract is accessible via a Python package and supports the four chosen languages in this study

GOT is a transformer-based model designed to recognize artificial characters beyond traditional text, such as sheet music, mathematical equations, and charts (Wei et al., 2024). Using a Vision Encoder Decoder (VED) architecture with 580 million parameters, GOT fine-tunes ViTDeT[1] as its vision encoder and Qwen-0.5B[2] as its language decoder. GOT is conveniently available on Hugging Face[3]. One limitation of GOT is that it only officially supports English and Simplified Chinese, and does not support Indonesian, Vietnamese, or Thai. This study seeks to address this limitation by fine-tuning GOT on these languages in Section 3.4.

3.1.4 Evaluation Metrics

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

Similar to most studies, we utilize Character Error Rate (CER) and Word Error Rate (WER) as our evaluation metrics to measure OCR accuracy (Hegghammer, 2022; Ignat et al., 2022). CER measures the accuracy of character recognition and is calculated using the Levenshtein or edit distance, which represents the minimum number of single-character insertions (I), deletions (D), and substitutions (S) required to transform one word into another. As shown in Equation 3.1, CER is defined as the edit distance between the OCR-

[1]ViTDeT is an object detection model using the Vision Transformer (ViT) as a backbone network (Y. Li et al., 2022).

[2]Qwen-0.5B is a Large Language Model (LLM) with 500 million parameters developed by Alibaba Cloud (Alibaba Cloud, 2025).

[3]https://huggingface.co/stepfun-ai/GOT-OCR2_0

predicted text and ground truth text, divided by the total number of characters in the ground truth text (N). A lower CER value indicates higher accuracy, with 0 representing perfect recognition. Notably, CER can exceed 1 when there is a significant number of insertions. WER serves as the word-based counterpart to CER.

3.2 Experiment 1: Benchmarking on Real-world Wikipedia Screenshots

To explore the performance of OCR tools on SEA scripts (RQ1), Experiment 1 benchmarks OCR systems using screen-captured, real-world data from Wikipedia. Unlike synthetic data, these screenshots contain formatting variations and complex layouts that better reflect real-world OCR challenges. This approach ensures that the evaluation closely mirrors practical use cases, where OCR tools must handle noisy and visually complex text.

While Wikipedia articles do not fully capture the diversity of real-world physical documents, they offer a valuable source for benchmarking OCR tools due to their linguistic richness, accessibility, and relevance to emerging digital use cases. Wikipedia text features varied sentence structures, diacritics, and real-world entities, making it a useful substitute for evaluating OCR performance. Additionally, Wikipedia articles are readily available in many languages, facilitating large-scale testing in SEA languages where real-world documents may be scarce. Furthermore, with the growing use of digital screenshots in applications like ChatGPT, where OCR is used on images or screenshots, the digital nature of Wikipedia screenshots reflects a practical OCR use case.

3.2.1 Data Collection

To ensure substantial data availability across our chosen languages, we compiled a dataset of 100 popular Wikipedia articles. Specifically, we selected the 20 most viewed English articles from each of five categories: people, present countries, cities, life, and buildings and structures ("Wikipedia:Popular pages", 2024). These categories were also chosen to create a diverse corpus in terms of content. Table A.1 lists the articles included in our dataset.

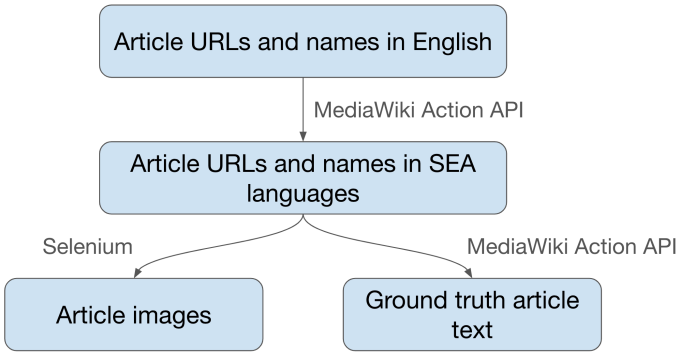

Figure 3.1: Pipeline for data collection from Wikipedia

From the dataset of 100 Wikipedia articles, we collected article images and ground truth article text in our selected languages using Python, Selenium[4], and the MediaWiki Action API[5]. Figure 3.1 illustrates the overall pipeline for data collection. The detailed steps are as follows:

1. Manually compile the dataset's article names and URLs in English.

[4]Selenium is a framework for automating web browsers, commonly used for web scraping by programmatically interacting with websites.

[5]The MediaWiki Action API allows access to wiki page operation features such as search and retrieval.

2. Fetch the article names and URLs in Thai, Vietnamese, and Indonesian from the MediaWiki Action API.

3. Download the article PDFs in all languages using Selenium.

4. Convert the article PDFs into PNG images, with each image representing one page of the PDF.

5. Download the article text into TXT files from the MediaWiki Action API.

The end result is a real-world Wikipedia dataset with 3,590 English images, 1,450 Indonesian images, 1,925 Vietnamese images, and 1,011 Thai images. Figure 3.2 presents sample collected images.

Figure 3.2: Sample English and Thai images collected from Wikipedia

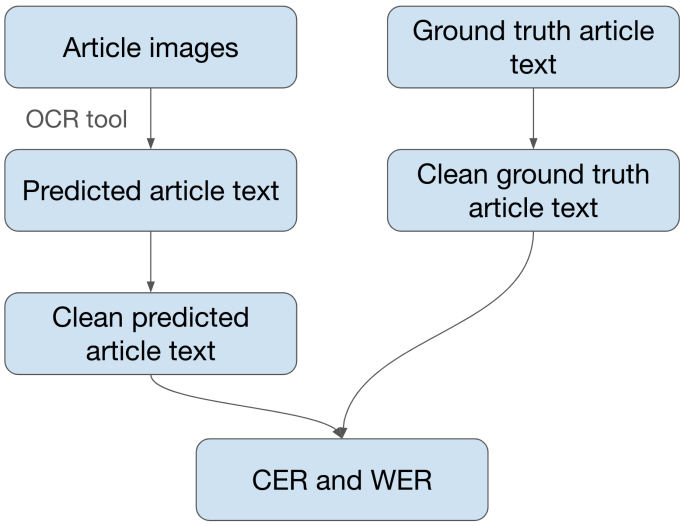

Figure 3.3: Pipeline for OCR evaluation

3.2.2 OCR Evaluation

After collecting the images and corresponding ground truth text, we ran the OCR tools and evaluated the CER and WER for each article. Figure 3.3 summarizes the overall pipeline for OCR evaluation. The detailed steps are as follows:

1. **Apply OCR:** Apply the OCR tools on the article images.

To run EasyOCR, Tesseract, and GOT on all 7,976 images, we used Slurm for job scheduling and execution on the SoC Compute Cluster at the School of Computing, National University of Singapore. Python and shell scripts were utilized to automate OCR execution.

2. **Data Cleaning:** Perform data cleaning on the predicted article text and ground truth article text.

The raw predicted text generated by the OCR tools exhibited some consistent formatting issues. For instance, Tesseract adds an additional space character after every predicted character. The article images also included references and in-text citations, which are not present in the ground truth. To address these issues, we performed data cleaning to align the output text more closely with the ground truth.

3. **Evaluation:** Compute the CER and WER between the predicted text and the ground truth text using JiWER[6].

4. **Data Validation:** Manually review articles with outlier CER values for incorrect ground truth text.

To prevent erroneous results, we implemented a data validation step that automatically checked the CERs for each article. Articles were flagged as outliers if their CER exceeded two standard deviations from the mean (Cousineau & Chartier, 2010). We manually reviewed these outlier articles for anomalies, resulting in the removal of around five articles per language, where the images and ground truth texts contained different content.

3.3 Experiment 2: Benchmarking on Synthetic Data

Unlike Experiment 1, Experiment 2 generates images from plain text for benchmarking. This approach allows us to introduce controlled distortions to the dataset, enabling an analysis of OCR robustness against noise on SEA languages and offering a different perspective on RQ1. Additionally, using synthetic data minimizes annotation errors, allowing us to better isolate script-related errors (RQ2).

[6] JiWER is a Python package designed for fast calculation of CER and WER.

3.3.1 Synthetic Data Generation

Using the article text collected from Experiment 1, we generated synthetic article images with Python and WeasyPrint[7], a Python package for converting HTML pages into PDF documents. To introduce font noise into the dataset, we randomly applied HTML tags to a specified ratio of space-separated words. For instance, a `<b>` tag could be added randomly to make words appear in bold. The following steps summarizes the process of data generation:

1. To add noise, randomly apply HTML tags to a specified ratio of space-separated words.
2. Generate synthetic article PDFs using the ground truth text collected from Experiment 1 with WeasyPrint.
3. Convert the article PDFs into PNG images, with each image representing one page of the PDF.

3.3.2 OCR Evaluation on Different Types of Noise

Table 3.3: Types of noise applied

	HTML Tag	Ratio
Bold	`<b>`	0.3
Italic	`<i>`	0.3
Link	`<a>`	0.3
Heading	`<h1>`	0.03

To investigate how different types of noise impact OCR performance (RQ1), we generated separate datasets, each containing a specific type of noise: bold, italic, link, or

[7] https://pypi.org/project/weasyprint/

heading. The noise was randomly applied to a specified percentage of words in each dataset. Additionally, a control dataset without noise was created for comparison.

Table 3.3 summarizes the noise types, their corresponding HTML tags, and the percentage of words affected. A smaller ratio was applied for headings because headings typically appear less frequently in natural text compared to other formatting elements like bold or italics. Limiting the number of headings also helped reduce the number of pages generated, ensuring a more manageable dataset size. Figure 3.4 presents a sample synthetic image with different types of noise.

Figure 3.4: Sample synthetic image with different types of noise

The selected noise types introduce formatting-based distortions commonly found in digital text and web-based content, posing unique challenges for OCR systems. Bold and italic formatting, frequently used for emphasis in scanned documents and articles, may alter character shapes and affect recognition accuracy. Links introduce underlining and color changes, which can interfere with OCR systems. Headings, often bold and larger in size, may also impact recognition. Evaluating these effects helps to assess OCR robustness in processing real-world digital text.

After generating the five separate datasets, we ran the OCR tools and evaluated their performance on each dataset, following the approach described in Section 3.2.2.

3.3.3 Error Classification by Character Type

Another area of interest in Experiment 2 was the effect of unique script characteristics, such as diacritics and punctuation, on OCR accuracy (RQ2). Using OCR results from the control dataset without noise, we categorized misclassifications by eight character types commonly found in English, Indonesian, Vietnamese, and Thai.

Table 3.4: Character types used for error classification

	Included Characters
Arabic digit	0-9
Latin letter	a-z, A-Z
Latin letter with diacritic	à-ỹ, À-Ỹ
Vietnamese special letter	đ, Đ
Thai letter	ก-ฮ
Special symbol	.,!?;:()-”–’—$%/&+-=[]
Whitespace	␣
Other	

We used the Levenshtein[8] Python package to identify edit operations (insertions,

[8] https://pypi.org/project/Levenshtein/

deletion, and substitutions) and classified misrecognized characters using RegEx[9]. Table 3.4 lists the character types we analyzed, with all uncategorized characters grouped under "Other".

3.4 Experiment 3: Fine-tuning for Vietnamese and Thai

To explore how fine-tuning can enhance OCR accuracy on SEA languages (RQ3), Experiment 3 fine-tunes GOT for Vietnamese and Thai and compares the fine-tuned model with EasyOCR and Tesseract. We chose to fine-tune for Vietnamese and Thai because GOT does not natively support these languages. Although GOT does not support Indonesian, we did not fine-tune for it, since the model already achieves state-of-the-art results for Indonesian, as demonstrated in the results from Experiment 1 (See Section 4.1.2).

3.4.1 Fine-tuning GOT

Fine-tuning our model effectively required a large volume of high-quality data. To ensure the reliability of our training data, we generated additional synthetic data without noise. After collecting more plain text from Wikipedia, we created 960 images for training and 50 images for testing, for both Vietnamese and Thai. To investigate how much training data is needed for effective fine-tuning, we randomly selected samples from the training dataset to create datasets with 50, 100, 200, 400, and 960 images. These sample sizes were chosen to follow an increasing scale, approximating diminishing returns as data size increases. For each dataset, the same base model (GOT) was fine-tuned for three epochs,

[9] https://docs.python.org/3/library/re.html

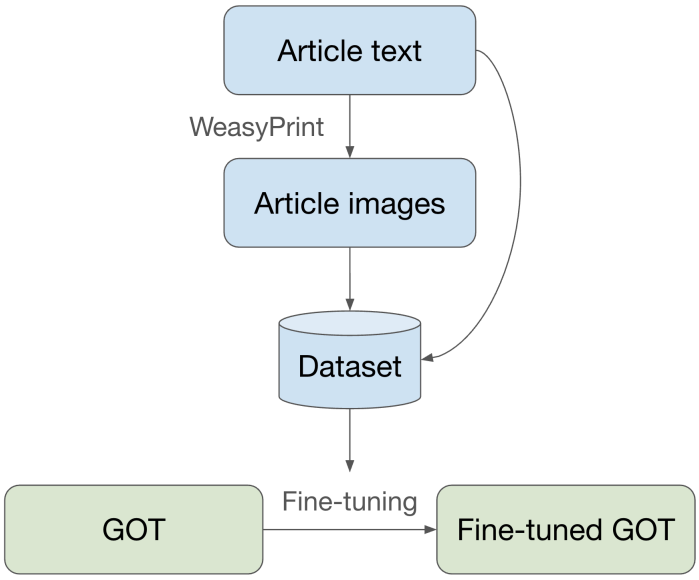

Figure 3.5: Pipeline for fine-tuning GOT

resulting in different fine-tuned models based on the varying dataset sizes.

For fine-tuning, we used SWIFT[10], a user-friendly framework that supports fine-tuning GOT[11]. Using a single NVIDIA Titan V GPU on the SoC Compute Cluster, we fine-tuned our model using Low-Rank Adaptation (LoRA), which optimizes only small adapter layers while keeping the pre-trained model weights frozen (Hu et al., 2021). This approach allows for efficient fine-tuning with reduced memory and computational costs.

3.4.2 OCR Evaluation on Fine-tuned GOT

We ran each fine-tuned GOT model on the test dataset and evaluated their performance, following the approach described in Section 3.2.2. We also ran EasyOCR and Tesseract on the test dataset for comparison.

[10] https://github.com/modelscope/ms-swift
[11] https://github.com/modelscope/ms-swift/issues/2122

Chapter 4

Results and Discussion

In this chapter, we present and analyze the results of the experiments, evaluating their significance and discussing the insights gained in relation to the research questions.

4.1 RQ1: How do popular OCR tools perform on SEA scripts?

4.1.1 OCR Accuracy on Wikipedia Screenshots

Table 4.1: Average Character and Word Error Rate on Wikipedia screenshots

	Character Error Rate			Word Error Rate		
	EasyOCR	Tesseract	GOT	EasyOCR	Tesseract	GOT
English	**0.21**	**0.21**	0.67	**0.27**	0.29	0.67
Indonesian	**0.28**	**0.28**	0.61	**0.36**	0.42	0.71
Vietnamese	0.47	**0.38**	-	0.45	**0.39**	-
Thai	**0.35**	0.55	-	**1.45**	**1.45**	-

Table 4.1 lists the results from benchmarking the OCR tools on real-world Wikipedia screenshots (Experiment 1). A lower Character Error Rate (CER) and Word Error Rate (WER) implies better OCR accuracy. Bolded numbers highlight the best-performing tool for each metric and language. Our observations are as follows:

- **Latin scripts achieved the best performance.** When comparing performance

across script types, Latin scripts (English and Indonesian) had the lowest average CER of 0.38, followed by Latin scripts with diacritics (Vietnamese) with an average CER of 0.42 and Brahmic scripts (Thai) with 0.45. These results align with findings from previous studies, which have shown that OCR tools tend to perform best on Latin-based scripts (Hegghammer, 2022; Ignat et al., 2022).

- **EasyOCR achieved the best overall performance on the Wikipedia screenshots among the OCR tools,** with an average CER of 0.33 across all languages. This was followed by Tesseract with an average CER of 0.35, and GOT with 0.64.
- **WER may not be a reliable metric for non-segmented scripts.** Thai had a particularly high average WER of 1.45, as Thai is a non-segmented script that lacks explicit word boundaries. This characteristic makes word-level evaluation more challenging for OCR systems.

Limitations of Wikipedia Screenshots

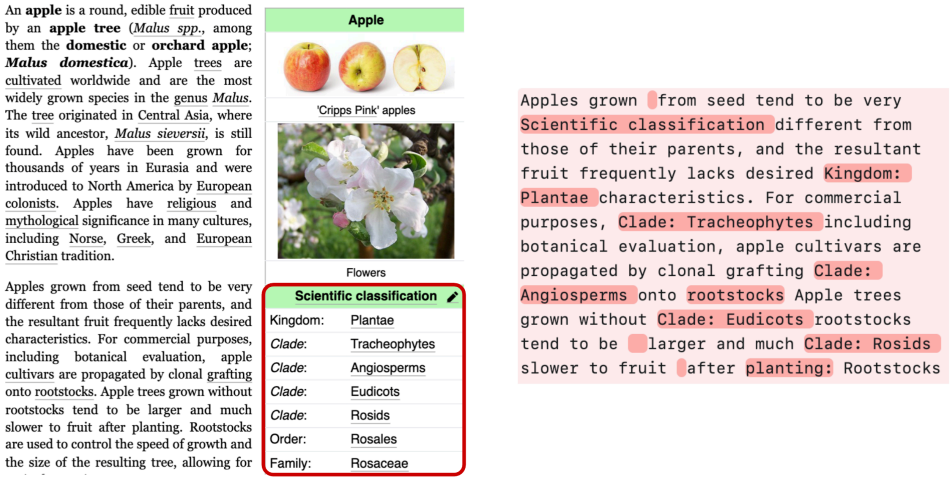

Figure 4.1: Table in Wikipedia articles that interferes with OCR results

Wikipedia articles often have complex layouts and multi-modal elements, such as tables and embedded text in images, that interfere with OCR results. For example, in Figure 4.1, the OCR tool mistakenly recognized the text inside a table as part of the main article text, interweaving it with the surrounding content. Since the ground truth article text collected from the MediaWiki Action API does not include table content, this discrepancy contributed to higher CERs and WERs in the Wikipedia screenshot results.

Accurately recognizing such complex layouts in practical settings requires document layout analysis. Although layout analysis is beyond the scope of this study, this limitation highlights its importance in processing modern digital documents. Future work on document recognition can explore layout-aware approaches.

One possible workaround is to use Selenium to generate bounding boxes around the main article text, restricting OCR to only those regions. However, we felt this would not accurately reflect the challenges in recognizing real-world digital documents, which often contain complex layouts and multi-modal elements. To preserve the realism of the task, we chose to retain the raw OCR results from Wikipedia screenshots. A separate Experiment 2 using noise-free synthetic data was also conducted to better isolate errors specifically related to character recognition.

4.1.2 OCR Accuracy on Synthetic Data

Table 4.2 presents the results from benchmarking the OCR tools on synthetic documents without noise (Experiment 2). Our observations are as follows:

- **Latin scripts achieved the best results on synthetic data,** followed by Latin scripts with diacritics and Brahmic scripts. This finding is consistent with the trend

Table 4.2: Average Character and Word Error Rate on synthetic data

	Character Error Rate			Word Error Rate		
	EasyOCR	Tesseract	GOT	EasyOCR	Tesseract	GOT
English	0.03	0.02	**0.01**	0.09	**0.04**	**0.04**
Indonesian	0.02	0.02	**0.01**	0.09	0.06	**0.05**
Vietnamese	0.10	**0.03**	-	0.17	**0.04**	-
Thai	**0.07**	0.09	-	0.68	**0.64**	-

observed in the benchmarking of real-world Wikipedia data.

- **The OCR tools performed better on synthetic data than on Wikipedia screenshots.** All metrics showed an average decrease in error rate by 0.34. This observation aligns with the findings from Ignat et al. (2022) and Hegghammer (2022).

- **GOT and Tesseract performed well on synthetic data.** On synthetic data, GOT achieved the best results with an average CER of 0.01, followed by Tesseract with 0.04 and EasyOCR with 0.05. This contrasts with the results from the Wikipedia screenshots. This findings suggest that GOT and Tesseract may not be robust to noise, but can achieve high accuracy in noise-free environments, as further discussed in Section 4.1.3.

4.1.3 Impact of Noise on OCR Accuracy

Table 4.3 presents the results from benchmarking EasyOCR on synthetic documents with noise (Experiment 2). Using the noise-free dataset as the baseline, we calculated the percent change for each noise type and language. A negative percent change indicates that the error rate decreased with the added noise, while a positive percent change indicates that the error rate increased. To visually distinguish these changes, positive values are

Table 4.3: Impact of noise on EasyOCR's accuracy

	No Noise CER	Bold % Change	Italic % Change	Link % Change	Heading % Change
English	0.03	**-7.1%**	0.0%	3.6%	**-7.1%**
Indonesian	0.02	**-5.6%**	0.0%	22.2%	**-5.6%**
Vietnamese	0.10	**-1.0%**	**-1.9%**	**-6.7%**	**-11.5%**
Thai	0.07	0.0%	0.0%	7.2%	0.0%

Table 4.4: Impact of noise on Tesseract's accuracy

	No Noise CER	Bold % Change	Italic % Change	Link % Change	Heading % Change
English	0.02	**-4.2%**	0.0%	0.0%	8.3%
Indonesian	0.02	0.0%	0.0%	0.0%	21.7%
Vietnamese	0.03	0.0%	3.8%	3.8%	26.9%
Thai	0.09	3.2%	0.0%	5.4%	4.3%

Table 4.5: Impact of noise on GOT's accuracy

	No Noise CER	Bold % Change	Italic % Change	Link % Change	Heading % Change
English	0.01	27.3%	36.4%	**-18.2%**	218.2%
Indonesian	0.01	9.1%	18.2%	36.4%	63.6%

highlighted in red and negative values in green. Similar results for Tesseract and GOT are shown in Table 4.4 and Table 4.5 respectively. Our observations from these results are as follows:

- **EasyOCR was the most robust to noise among the OCR tools.** EasyOCR had the lowest average percent change of -0.8%, followed by Tesseract with an average percent change of 4.6% and GOT with 48.9%. Notably, adding noise improved the performance of EasyOCR, suggesting that it may be particularly well-suited to handle noisy data.

- **Tesseract and GOT were sensitive to noise.** This finding aligns with the results from Section 4.1.2, where Tesseract and GOT performed well on synthetic data but struggled with real-world Wikipedia screenshots, which contain noise and complex layouts.

- **Heading noise impacted the OCR tools the most,** with an average percent change of 31.9%, followed by italic noise (5.6%), link noise (5.4%), and bold noise (2.2%). This result indicates that larger font sizes with bold formatting, as seen in the heading noise, pose the greatest challenge to OCR systems.

Limitations of Noise Types

The selected noise types in our experiment are commonly found in digital text and web-based content but may not fully represent the range of noise found in other document types. For instance, scanned physical documents can contain noise from poor lighting, rotations, and blurry scans (Hegghammer, 2022). Additional artificial noise types could be incorporated in future studies, depending on the specific goals of the research.

4.1.4 Runtime

Table 4.6: Average OCR runtime per page (seconds)

	EasyOCR	Tesseract	GOT
English	**3.2**	11.7	24.3
Indonesian	**2.9**	13.2	31.4
Vietnamese	**3.9**	11.8	-
Thai	**2.3**	16.8	-

Table 4.6 presents the average OCR runtime per page, with the bolded numbers highlighting the fastest tool for each language. The runtimes were recorded when running

the OCR tools on noise-free synthetic data (Experiment 2) on the SoC Compute Cluster. We observed that **EasyOCR is the fastest OCR tool,** with an average runtime of 3.1 seconds, followed by Tesseract at 13.4 seconds and GOT at 27.9 seconds.

4.2 RQ2: What script-related challenges affect OCR accuracy on SEA languages?

4.2.1 Latin Scripts

Error Classification by Character Type in English Articles

	Count	EasyOCR % Missed	Tesseract % Missed	GOT % Missed
Arabic digit	38,324	0.7%	1.9%	0.3%
Latin letter	1,546,964	1.3%	1.8%	0.4%
Latin letter with diacritic	452	100.0%	56.0%	15.9%
Special symbol	55,110	29.5%	3.7%	4.3%
Whitespace	317,587	4.9%	4.3%	3.6%
Other	1,562	103.6%	92.6%	127.9%

Figure 4.2: Error classification by character type in English articles

Figure 4.2 presents the results of classifying errors by character type in the benchmark on English synthetic data without noise (Experiment 2). Results on Indonesian articles are shown in Figure 4.3. The ”Count” column lists the number of characters belonging to each character type in the ground truth text. The remaining columns show the percentage of characters misclassified by each OCR tool.

Notably, these percentages can exceed 100%, as demonstrated in Figure 4.2 under the ”Other” character type. This is similar to how CERs can exceed 1. The reason

Error Classification by Character Type in Indonesian Articles

	Count	EasyOCR % Missed	Tesseract % Missed	GOT % Missed
Arabic digit	24,947	0.4%	1.8%	0.2%
Latin letter	1,208,707	0.5%	1.8%	0.4%
Latin letter with diacritic	276	7.2%	100.0%	16.7%
Special symbol	38,980	22.9%	4.9%	1.4%
Whitespace	207,556	4.8%	5.1%	4.1%
Other	1,262	94.4%	99.6%	64.8%

Figure 4.3: Error classification by character type in Indonesian articles

for this is that the misclassifications are identified using edit operations, which include substitutions, deletions, and insertions. If the OCR tool incorrectly inserts additional characters, the number of misclassified characters can surpass the ground truth.

Our observations from the results on benchmarked Latin scripts are as follows:

- **Latin letters, Arabic digits, and whitespaces achieved high accuracies.** These three character types accounted for 97% of all characters in both English and Indonesian texts.

- **OCR tools for English and Indonesian struggled more with Latin letters containing diacritics than those without diacritics,** as evidenced by the higher error rates for diacritical characters. One possible explanation is that the OCR models were not trained on enough diacritical marks, which is reflected in their low frequency in the English and Indonesian datasets.

  In particular, EasyOCR and Tesseract misclassified all Latin characters with diacritics for English and Indonesian respectively, suggesting that their language-specific

models were not trained to recognize diacritics.

4.2.2 Latin Scripts with Diacritics

Error Classification by Character Type in Vietnamese Articles

	Count	EasyOCR % Missed	Tesseract % Missed
Arabic digit	31,473	1.1%	2.2%
Latin letter	916,667	8.5%	1.8%
Latin letter with diacritic	294,406	14.9%	1.9%
Vietnamese special letter	30,903	4.2%	1.8%
Special symbol	41,655	25.4%	3.1%
Whitespace	367,936	10.9%	5.3%
Other	1,909	105.4%	88.6%

Figure 4.4: Error classification by character type in Vietnamese articles

Figure 4.4 presents the results of classifying errors in the benchmark on Vietnamese data without noise (Experiment 2). We observed that **Tesseract achieved high accuracies in recognizing Vietnamese characters.** Vietnamese characters, which make up 74% of the dataset, consist of Latin letters with and without diacritics, as well as a special character (đ, Đ) categorized under "Vietnamese special letter". Only 2% of these Vietnamese characters were misclassified by Tesseract, which is similar to the performance of English models in recognizing Latin letters. In contrast, EasyOCR struggled more with diacritical characters.

Error Classification by Character Type in Thai Articles

	Count	EasyOCR % Missed	Tesseract % Missed
Arabic digit	22,580	0.9%	6.7%
Latin letter	36,174	100.0%	100.0%
Latin letter with diacritic	101	100.0%	100.0%
Thai letter	1,014,395	1.2%	4.2%
Special symbol	14,246	8.0%	10.6%
Whitespace	58,164	37.5%	37.2%
Other	890	105.2%	110.4%

Figure 4.5: Error classification by character type in Thai articles

4.2.3 Brahmic Scripts

Figure 4.5 presents the results of classifying errors in the benchmark on Thai data without noise (Experiment 2). Our observations from the Thai results are as follows:

- **Thai letters were well-recognized by OCR tools.** Both EasyOCR and Tesseract achieved state-of-the-art results on Thai letters, which make up 88% of the Thai dataset, comparable to the accuracy of English models recognizing Latin letters.
- **Thai OCR models did not recognize Latin letters with or without diacritics,** as evidenced by the 100% misclassification rate for both EasyOCR and Tesseract.
- **Thai OCR models performed worse in recognizing whitespaces compared to other languages.** 37% of whitespaces were misclassified in Thai, which is significantly higher than the misclassification rates for English (4%), Indonesian (5%), and Vietnamese (8%).

We also observed some patterns common across all benchmarked languages:

- **OCR tools misclassified almost all uncategorized characters.** These include non-Latin characters, such as Greek, Cyrillic, Arabic, Chinese, and Japanese characters, as well as mathematical symbols and phonetic symbols.
- **EasyOCR struggled more with special symbols.** EasyOCR misclassified 21% of all special symbols, which is 16% more than Tesseract.

4.2.4 Implications for OCR Development on SEA Languages

A common preconception is that OCR systems are less accurate on SEA scripts. Prior studies have consistently reported that SEA scripts tend to yield lower accuracy than Latin scripts (Ignat et al., 2022). This view appears to be supported by the language-level trends observed in Sections 4.1.1 and 4.1.2, where Latin-script languages outperformed Vietnamese and Thai in both CER and WER. However, after classifying OCR errors by character type, we found that OCR tools are actually effective at recognizing the more complex Vietnamese and Thai scripts. In fact, the error rates for these scripts are comparable to those for Latin characters by English models. These findings challenge the claim by Agarwal and Anastasopoulos (2024) that OCR tools perform poorly on scripts with ornate diacritics or unique character shapes.

One major contributor to the lower overall accuracy of Vietnamese and Thai lies in the misclassification of characters that are irrelevant to the target language. For example, Latin letters that appear in the Thai dataset were consistently misclassified, as they are not part of the Thai script. These irrelevant characters often occur in metadata, URLs, or borrowed text segments within documents, and their misclassification inflates

the CER. This suggests that the lower performance of OCR on Vietnamese and Thai is not necessarily due to the inherent complexity of the scripts themselves, but rather the OCR tools' limitations in handling mixed-script or multilingual content effectively.

This highlights the need for OCR systems to better handle mixed-script content, especially in real-world documents where such mixing is common. Future work could focus on developing models that are script-aware or capable of dynamically switching recognition modes based on the detected script. Incorporating script identification as a pre-processing step or training models with more diverse multilingual datasets could also improve robustness in mixed-script contexts.

4.3 RQ3: How can fine-tuning enhance OCR accuracy on SEA languages?

4.3.1 Performance of Fine-tuned GOT

Figure 4.6 presents the performance of the fine-tuned GOT models on Vietnamese (Experiment 3). Each GOT model was fine-tuned on different numbers of training images, and their CERs on the test dataset are plotted. For comparison, the performance of EasyOCR and Tesseract is also included. From the Vietnamese model results, we observed the following:

- **Our best fine-tuned Vietnamese GOT model achieved a CER of 0.10.** Using 200 training images resulted in a 33% reduction in error rate compared to the base GOT model without fine-tuning, which achieved a CER of 0.15.

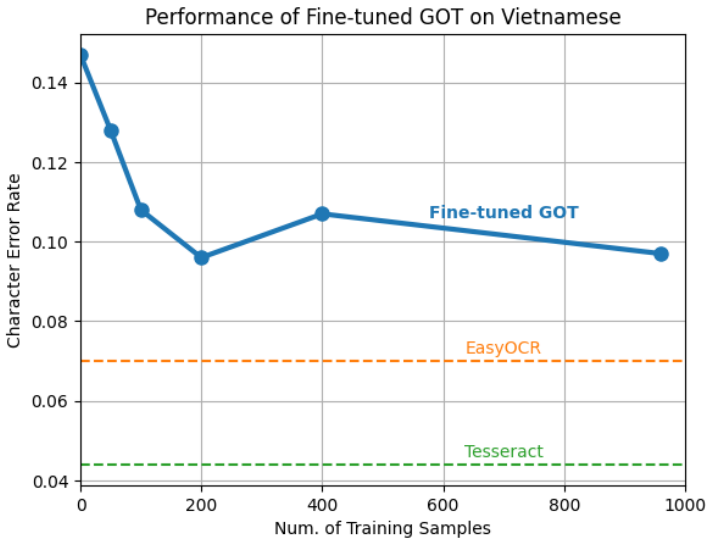

Figure 4.6: Performance of fine-tuned GOT on Vietnamese

- **The fine-tuned Vietnamese model underperformed relative to EasyOCR and Tesseract.** EasyOCR achieved a CER of 0.07, and Tesseract achieved a CER of 0.04.

- **Model performance did not consistently improve with more training data.** Instead, we observed a non-monotonic trend, peaking at 200 training samples and plateauing thereafter.

  This behavior may be due to the strong pre-training of the base GOT model on English. Since Vietnamese uses Latin scripts with diacritics, many characters and visual patterns are shared with English. The base model likely already encodes strong representations for Latin characters, enabling it to adapt to Vietnamese with relatively little fine-tuning data. As a result, most of the learning occurs early on, and additional training data provides diminishing returns once the model has sufficiently generalized to the task.

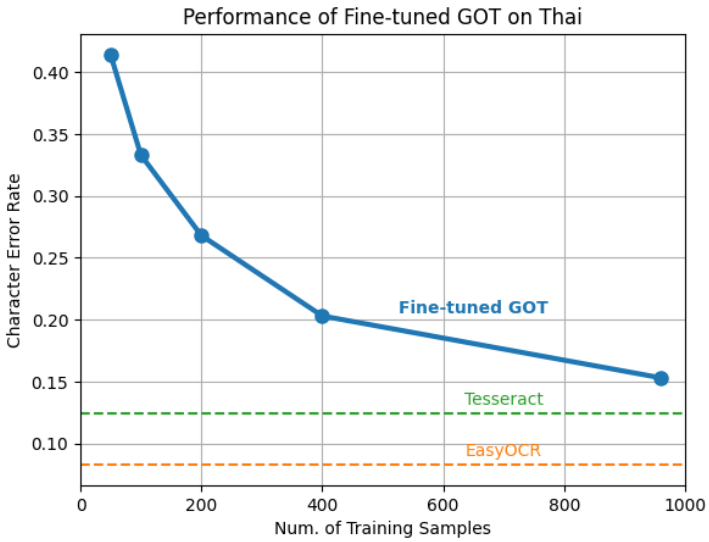

Figure 4.7: Performance of fine-tuned GOT on Thai

Figure 4.7 presents the performance of the fine-tuned GOT models on Thai (Experiment 3). From the Thai model results, we observed the following:

- **Our best fine-tuned Thai GOT model achieved a CER of 0.15.** In comparison, the base GOT model failed to recognize Thai characters, producing mostly Greek characters and yielding a CER of 0.90 on a sample image.
- **The fine-tuned Thai model also underperformed compared to EasyOCR and Tesseract,** which achieved CERs of 0.08 and 0.12 respectively.
- **Unlike the Vietnamese model, the Thai model consistently improved with more training data,** achieving its best performance when trained on all 960 training samples. While the CER improved with additional data, the reduction in CER diminished as the sample size increased: from 0.08 with 100 samples, to 0.06 with 200 samples, to 0.06 with 400 samples, and finally to 0.05 with 960 samples.

4.3.2 Implications on Adapting OCR Systems to New Languages

Our results demonstrate that fine-tuning is an effective strategy for adapting OCR systems to new languages. Even with relatively small amounts of training data, it can significantly improve OCR performance. This makes fine-tuning especially promising for low-resource languages, where large annotated datasets are often costly and time-consuming to obtain. Our findings show that reasonable accuracy can be achieved with modest datasets, highlighting the practicality of this method in resource-constrained settings.

While fine-tuning was successful, our experiments were limited to fine-tuning GOT on noise-free synthetic data in Vietnamese and Thai. Future work could incorporate noisy data to improve model robustness in real-world scenarios. Additionally, surpassing state-of-the-art tools like EasyOCR and Tesseract may require more sophisticated fine-tuning techniques. Although increasing the amount of training data improved performance for Thai, our Vietnamese results suggest that more data does not always lead to better accuracy. As such, further gains may come from exploring alternative models and optimizing fine-tuning strategies, including hyperparameter tuning.

Chapter 5

Conclusion

References

Agarwal, M., & Anastasopoulos, A. (2024). A concise survey of OCR for low-resource languages. In M. Mager, A. Ebrahimi, S. Rijhwani, A. Oncevay, L. Chiruzzo, R. Pugh, & K. von der Wense (Eds.), *Proceedings of the 4th workshop on natural language processing for indigenous languages of the americas (americasnlp 2024)* (pp. 88–102). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.americasnlp-1.10

Aji, A. F., Forde, J. Z., Loo, A. M., Sutawika, L., Wang, S., Winata, G. I., Yong, Z.-X., Zhang, R., Doğruöz, A. S., Tan, Y. L., & Cruz, J. C. B. (2023). Current status of NLP in South East Asia with insights from multilingualism and language diversity. *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, 8–13. https://aclanthology.org/2023.ijcnlp-tutorials.2

Alibaba Cloud. (2025). Qwen. https://github.com/QwenLM/Qwen

Baek, Y., Lee, B., Han, D., Yun, S., & Lee, H. (2019). Character region awareness for text detection. *CoRR, abs/1904.01941*. http://arxiv.org/abs/1904.01941

Cousineau, D., & Chartier, S. (2010). Outliers detection and treatment: A review. *International Journal of Psychological Research*, *3*, 58–67. https://doi.org/10.21500/20112084.844

Feng, W., He, W., Yin, F., Zhang, X.-Y., & Liu, C.-L. (2019). Textdragon: An end-to-end framework for arbitrary shaped text spotting. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Gupte, A., Romanov, A., Mantravadi, S., Banda, D., Liu, J., Khan, R., Meenal, L. R., Han, B., & Srinivasan, S. (2021). Lights, camera, action! A framework to improve NLP accuracy over OCR documents. *CoRR*, *abs/2108.02899*. https://arxiv.org/abs/2108.02899

Hegghammer, T. (2022). OCR with Tesseract, Amazon Textract, and Google Document AI: A benchmarking experiment. *Journal of Computational Social Science*, *5*, 861–882. https://doi.org/https://doi.org/10.1007/s42001-021-00149-1

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2021). Lora: Low-rank adaptation of large language models. *CoRR*, *abs/2106.09685*. https://arxiv.org/abs/2106.09685

Ignat, O., Maillard, J., Chaudhary, V., & Guzmán, F. (2022). OCR improves machine translation for low-resource languages. In S. Muresan, P. Nakov, & A. Villavicencio (Eds.), *Findings of the association for computational linguistics: Acl 2022* (pp. 1164–1174). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-acl.92

Jaided AI. (2025). Easyocr. https://github.com/JaidedAI/EasyOCR

Laurent, V., & Lauar, F. (2024). Spanish trocr: Leveraging transfer learning for language adaptation. https://arxiv.org/abs/2407.06950

Li, M., Lv, T., Cui, L., Lu, Y., Florêncio, D. A. F., Zhang, C., Li, Z., & Wei, F. (2021). Trocr: Transformer-based optical character recognition with pre-trained models. *CoRR*, *abs/2109.10282*. https://arxiv.org/abs/2109.10282

Li, Y., Mao, H., Girshick, R., & He, K. (2022). Exploring plain vision transformer backbones for object detection. https://arxiv.org/abs/2203.16527

List of languages by total number of speakers. (2025). https://en.wikipedia.org/wiki/List_of_languages_by_total_number_of_speakers

List of wikipedias. (2024). https://en.wikipedia.org/wiki/List_of_Wikipedias

Magueresse, A., Carles, V., & Heetderks, E. (2020). Low-resource languages: A review of past work and future challenges. *CoRR*, *abs/2006.07264*. https://arxiv.org/abs/2006.07264

Parres, D., & Paredes, R. (2023). Fine-tuning vision encoder–decoder transformers for handwriting text recognition on historical documents. In G. A. Fink, R. Jain, K. Kise, & R. Zanibbi (Eds.), *Document analysis and recognition - icdar 2023* (pp. 253–268). Springer Nature Switzerland.

Salehudin, M., Basah, S., Yazid, H., Basaruddin, K., Safar, M., Som, M. M., & Sidek, K. (2023). Analysis of optical character recognition using easyocr under image degradation. *Journal of Physics: Conference Series*, *2641*(1), 012001. https://doi.org/10.1088/1742-6596/2641/1/012001

Shi, B., Bai, X., & Yao, C. (2015). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *CoRR*, *abs/1507.05717*. http://arxiv.org/abs/1507.05717

Smith, R. (2007). An overview of the tesseract ocr engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, *2*, 629–633. https://doi.org/10.1109/ICDAR.2007.4376991

Smith, R. W. (2013). History of the Tesseract OCR engine: what worked and what didn't. In R. Zanibbi & B. Coüasnon (Eds.), *Document recognition and retrieval xx* (p. 865802). SPIE. https://doi.org/10.1117/12.2010051

Subramani, N., Matton, A., Greaves, M., & Lam, A. (2020). A survey of deep learning approaches for OCR and document understanding. *CoRR*, *abs/2011.13534*. https://arxiv.org/abs/2011.13534

Tesseract OCR. (2025). Tesseract. https://github.com/tesseract-ocr/tesseract

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *CoRR*, *abs/1706.03762*. http://arxiv.org/abs/1706.03762

Wei, H., Liu, C., Chen, J., Wang, J., Kong, L., Xu, Y., Ge, Z., Zhao, L., Sun, J., Peng, Y., Han, C., & Zhang, X. (2024). General ocr theory: Towards ocr-2.0 via a unified end-to-end model. https://arxiv.org/abs/2409.01704

Wikipedia:popular pages. (2024). https://en.wikipedia.org/wiki/Wikipedia:Popular_pages

Appendix A

Wikipedia Dataset

Table A.1: Dataset of 100 Wikipedia articles used for benchmarking real-world Wikipedia screenshots in Experiment 1 (See Section 3.2)

Category	**Articles**
People	Elizabeth II, Barack Obama, Michael Jackson, Elon Musk, Lady Gaga, Adolf Hitler, Eminem, Lionel Messi, Justin Bieber, Freddie Mercury, Kim Kardashian, Johnny Depp, Steve Jobs, Dwayne Johnson, Michael Jordan, Taylor Swift, Stephen Hawking, Kanye West, Donald Trump, Cristiano Ronaldo
Present countries	United States, India, United Kingdom, Canada, Australia, China, Russia, Japan, Germany, France, Singapore, Israel, Pakistan, Philippines, Brazil, Italy, Netherlands, New Zealand, Ukraine, Spain
Cities[a]	New York City, London, Hong Kong, Los Angeles, Dubai, Washington, D.C., Paris, Chicago, Mumbai, San Francisco, Rome, Monaco, Toronto, Tokyo, Philadelphia, Machu Picchu, Jerusalem, Amsterdam, Boston, Angelsberg
Life	Cat, Dog, Animal, Lion, Coronavirus, Tiger, Human, Dinosaur, Elephant, Virus, Horse, Photosynthesis, Evolution, Apple, Bird, Mammal, Potato, Polar bear, Shark, Snake
Buildings and structures[b]	Taj Mahal, Burj Khalifa, Statue of Liberty, Great Wall of China, Eiffel Tower, Berlin Wall, Stonehenge, Mount Rushmore, Colosseum, Auschwitz concentration camp, Great Pyramid of Giza, One World Trade Center, Empire State Building, White House, Petra, Large Hadron Collider, Hagia Sophia, Golden Gate Bridge, Panama Canal, Angkor Wat

[a] Singapore was replaced because it's already listed under present countries.

[b] Machu Picchu was replaced becuase it's already listed under cities.