

Studies on Text Detection and Character
Image Generation for Advanced Text
Recognition

テキスト認識高度化のためのテキスト検出
と文字画像生成に関する研究

山梨大学大学院
医工農学総合教育部
博士課程学位論文

修了年月 2024年3月

氏名 LEOW CHEE SIANG

Copyright © 山梨大学

2023 年度 山梨大学大学院医工農学総合教育部 工学専攻 システム統合工学コース
博士論文公聴会及び最終審査にて発表済み

公聴会開催日：2024 年 1 月 31 日

開催場所： 山梨大学医工農学総合教育部 工学専攻 システム統合工学コース内

主催： 山梨大学

Studies on Text Detection and Character Image Generation for Advanced Text Recognition

Abstract

In recent years, the digital landscape has seen a dramatic transformation, marking the onset of a new era in information accessibility and processing, largely driven by the swift advancements in deep learning technologies. This particularly impact in the realm of Optical Character Recognition (OCR), which has experienced a revolutionary change, echoing the evolution seen in the creation and distribution of multimedia content. Historically, OCR technology has faced significant challenges similar to those faced by early speech recognition systems, namely the accurate transformation of diverse textual content into machine-readable formats. These challenges were primarily due to the absence of powerful computational tools and sophisticated algorithms needed to manage the variability in text presentations. During this period, OCR systems were relatively basic and struggled with intricate layouts, various fonts, and inconsistent print quality, mirroring the early difficulties in speech recognition with unfamiliar words and different accents.

The introduction of deep learning marked a crucial turning point. The development and widespread adoption of powerful Graphical Processing Units (GPUs), along with the expansion of data storage capabilities via Hard Disk Drives (HDDs) and Solid State Drives (SSDs), provided the necessary infrastructure for sophisticated computational tasks. This evolution in hardware, coupled with the exponential growth of big data, has propelled the advancement of deep learning technologies. Open-source deep learning frameworks like Tensorflow by Google and Pytorch by Meta (formerly Facebook), building on NVIDIA's Compute Unified Device Architecture (CUDA), have significantly enhanced OCR systems' capabilities. These advancements in OCR are reflective of the progress in the area of automatic speech recognition, where deep learning has facilitated more efficient handling and interpretation of vast volumes of data.

Today's OCR technologies, powered by deep learning, demonstrate exceptional proficiency in accurately detecting and recognizing text from various sources. Modern systems are adept at handling multilingual text, deciphering handwritten notes, and processing documents with complex layouts, mirroring recent improvements in speech recognition that enable nuanced understanding and interaction with humans. The field of OCR has benefited from methods such as single-line text detection in multi-line text blocks and innovative data augmentation techniques for character classification, improving the accuracy and reliability of these systems. The societal impact of these advancements in OCR technology is profound. In the business world, OCR systems have become essential for automating data entry, streamlining document management, and improving access to historical archives. In education, OCR facilitates the digitization of materials, making knowledge more accessible. In healthcare, the technology aids in managing patient records, enhancing the delivery of care.

Moreover, the integration of OCR with other technologies, like natural language pro-

cessing and image recognition, opens up new avenues for advanced applications. For example, combining OCR with natural language processing and retrieval technologies can significantly improve information retrieval systems, making them more robust and user-friendly. The evolution of OCR, driven by deep learning and technological advancements, mirrors a broader trend in the digital age, where data processing and accessibility are constantly being redefined. As OCR technology continues to evolve, its integration with emerging technologies is expected to further revolutionize our interaction with and processing of the vast amounts of information available in our increasingly digital world.

However, to build a high-performance OCR system with Deep Learning technology, a large number of data is required. OCR systems that currently exist in the world have high recognition rates for fonts that because of font training data can be generated easily. In contrast, handwritten text data must be written by hand by humans, which requires huge human and financial costs to generate large amounts of data. Today, there are far more documents containing not only machine printed characters but also handwritten characters than in the past, and there are all kinds of patterns of machine printed and handwritten characters, and OCR models based on Deep Learning are considered the most promising technology to handle them. In order to achieve highly accurate character recognition, it is also necessary to have a technology that can accurately detect characters. Due to the influence of digitization, text information printed on documents has become more complex, and it is difficult to accurately detect text because a large amount of text is printed on the commonly used A4 size paper, which is then further handwritten by humans. In particular, even if the characters are the same, they may be printed on multiple lines in a small area, making the boundary between characters ambiguous and making character detection more difficult.

The research objectives of this thesis include improving OCR accuracy using Deep Learning-generated training data, recognizing narrow multi-line characters more accurately, and developing methods for multi-line text recognition. The study focuses on the Y-Autoencoder (Y-AE) and CRAFT models, exploring their application in generating diverse character images and enhancing text detection accuracy. The research also aims to develop simpler approaches for recognizing characters in multi-line text environments, expanding the capabilities of deep learning models beyond traditional one-line recognition methods.

The thesis contributes to text recognition, text detection, and multiple-lines text recognition. It demonstrates that images produced by a deep learning model can enhance character image recognizer performance. The introduction of a novel post-processing method for existing deep learning models improves character recognition rates, especially for characters with narrow line spacing. The research also addresses limitations in conventional TrOCR systems, proposing a pre-processing technique for multi-line character recognition within TrOCR's fixed-size input constraints.

This thesis is organized into the following chapters, providing a comprehensive exploration of character recognition improvement through deep learning and related technolo-

gies:

Chapter 1: Introduction - This chapter introduces prior research that has contributed to the improvement of character recognition with generated image using deep learning model and single-line text detection with improved deep learning model and novel post-processing method. It outlines the scope and aims of the current study, setting the stage for the detailed discussions that follow.

Chapter 2: Text Image Recognition - This chapter delves into the history of character recognition systems. It examines their fundamental principles, the challenges they face, and the roles they play in various applications.

Chapter 3: Deep Learning - An in-depth exploration of deep learning is presented in this chapter. It covers the foundational concepts, the evolution of the technology, and its impact on the field of character recognition.

Chapter 4: Text Detection - Here, the focus shifts to the specific aspect of character detection to text detection. This chapter discusses the methods and technologies used to identify and isolate characters from complex backgrounds and the application of deep learning in enhancing character detection capabilities. It examines the advancements and improvements brought about by integrating deep learning techniques.

Chapter 5: Image Generation Using Deep Learning - This chapter explores how deep learning can be used for character image generation. It highlights the methods and models that have been successful in creating diverse character sets for training and recognition purposes.

Chapter 6: Character Generation with Y-Autoencoder - Here, the Y-Autoencoder's role in character generation is discussed. The chapter elaborates on how this model can create varied character images, contributing significantly to the field of OCR.

Chapter 7: Single-line Text Detection In Multiple-lines Text Images - This chapter discusses a text detection deep learning model that employs an enhanced CRAFT (Character Region Awareness For Text detection). It examines the model's structure, functionality, and advantages with novel post-processing method and experiments results.

Chapter 8: Text Recognition Model - This chapter explore the capability of TrOCR for multiple-lines text recognition by modifying the pre-processing methods. Results shows that the TrOCR is able to trained with Y-AE generated image and also for multiple-lines text image.

Chapter 9: Summary And Future Works - The final chapter summarizes the study, highlighting its key findings, contributions to the field, and potential areas for future research.

Contents

1	Introduction	1
1.1	Background	1
1.2	Related Works	2
1.3	Problem Statement	4
1.4	Research Objective	5
1.5	Research Scope	5
1.6	Contributions	6
1.7	Organization of this thesis	7
2	Text Image Recognition	9
2.1	The History of Character and Text Images	9
2.2	Optical Character Recognition (OCR)	10
2.2.1	Origins and Early Development	10
2.2.2	Pattern Matching in OCR Technology	10
2.3	Handwritten Text Recognition	12
2.3.1	Handwritten Text Recognition: Evolution, Techniques, and Applications	12
2.4	Summary	13
3	Deep Learning	15
3.1	Deep Neural Forward Network	15
3.1.1	Activation Functions	17
3.1.2	Backpropagation	17
3.2	Optimizers in Deep Learning	18
3.2.1	Stochastic Gradient Descent (SGD)	18
3.2.2	Adam Optimizer	18
3.2.3	RMSprop Optimizer	19
3.3	Convolutional Neural Networks (CNNs)	19
3.4	Transformer Model Architecture	22
3.4.1	Encoder	22
3.4.2	Decoder	22
3.4.3	Self-Attention Mechanism	22

3.4.4	Positional Encoding	23
3.4.5	Multi-Head Attention	23
3.5	Vision Transformers (ViT)	23
3.5.1	ViT Architecture	23
3.6	Summary	25
4	Text Detection	26
4.1	Character Detection	26
4.2	Text Detection	28
4.3	Text Detection Using Deep Learning	29
4.3.1	Differentiable Binarization Network (DBNet)	30
4.3.2	DBNet++: Advanced Text Detection with Adaptive Scale Fusion Module	32
4.3.3	Character-Region Awareness For Text detection (CRAFT)	33
4.4	Summary	34
5	Image Generation Using Deep Learning	36
5.1	Autoencoders	36
5.2	Autoencoder-Based Image Reconstruction	37
5.3	Y-Autoencoder	38
5.3.1	Model Architecture	38
5.3.2	Loss Functions	38
5.4	Summary	40
6	Character Generation with Y-Autoencoder	42
6.1	Adaptive Instance Normalization (AdaIN)	42
6.2	Image Generation Model with AdaIN	43
6.3	Y-Autoencoder with AdaIN	44
6.3.1	Model architecture	44
6.3.2	Loss functions	46
6.4	Filtering of generated images	46
6.4.1	MSE-based filtering	47
6.4.2	Classifier-based filtering	48
6.5	Handwritten Character Classifier	48
6.6	Experiments and Discussion	50
6.6.1	Experimental setup	50
6.6.2	Character generation results	53
6.6.3	Character classification results	56
6.6.4	Analysis of Generated Images	57
6.7	Experiment on Kanji Image Generation	59
6.8	Conclusions	61
6.9	Summary	62

7	Single-line Text Detection In Multiple-lines Text Images	63
7.1	Model Architecture	63
7.1.1	Region and Affinity Score Label	65
7.2	Label for Enhanced CRAFT	66
7.2.1	Line Segmentation Label	66
7.3	Post-Processing Algorithm for Multi-Line Text Detection Using Enhanced CRAFT	68
7.4	Post-processing Methodology	68
7.5	Loss Functions	70
7.5.1	Loss Functions for Enhanced CRAFT Model	70
7.6	Experiment Condition and Dataset	71
7.6.1	Train Dataset	71
7.6.2	Test Dataset	71
7.6.3	Evaluation Metrics	72
7.6.4	Model Training and Evaluation Procedures	75
7.7	Experiment Result	76
7.8	Summary	79
8	Text Recognition Model	82
8.1	Model Architecture	82
8.2	Single Line and Multiple Line Text Recognition	83
8.3	Experiment	84
8.3.1	Pre-Training of TrOCR	84
8.3.2	Single Line Image Generation with Y-AE Generated Images	85
8.4	Result	86
8.4.1	Pretraining Results	86
8.4.2	Single-line Training Result	86
8.4.3	Multiple-lines Training Result	92
8.5	Conclusion	93
8.6	Summary	96
9	Summary and Future Works	97
	References	105
	Relationship between publications and this thesis	115
	Publications	116
A	Y-AE Generated Kanji Statistics	I
B	Pre-Training Dataset	XXVII

List of Figures

2.1	The history of printing revolution to recent OCR (Generated by ChatGPT)	9
2.2	Pattern matching-based OCR	11
3.1	Simple Deep Neural Forward Networks	16
3.2	Convolution Operations in CNNs	20
3.3	Transformer model architecture	21
3.4	Vision Transformer model architecture	24
4.1	Example of a pipeline for character detection using image processing techniques.	27
4.2	Example of a pipeline for character detection using Deep Learning.	28
4.3	Differentiable Binarization Network (DBNet) [1]	31
4.4	Adaptive Fusion Module and Spatial Attention architectures [2]	32
4.5	CRAFT architectures	34
5.1	Autoencoder based image reconstruction model architecture	37
5.2	Y-Autoencoder Architecture [3]	39
6.1	Feature maps with AdaIN	43
6.2	The Y-autoencoder architecture.	45
6.3	Example of generated images of Hiragana character “あ.”	47
6.4	Pre-processing of handwritten character images for calculating the MSE.	48
6.5	ResNet-152 Model for Handwritten Japanese Characters	49
6.6	List of training conditions for character classification models.	52
6.7	Example of handwritten character images generated by the Y-AE with and without AdaIN.	54
6.8	Y-AE generated images on different characters	55
6.9	PCA analysis comparing ETL images, generated images, and test images	58
6.10	PCA analysis comparing ETL images, generated images, and test images	60
6.11	Generated result samples of Kanji	61
7.1	The enhanced CRAFT model architecture	64
7.2	Image label for training the CRAFT	66
7.3	Line Segmentation Label	67

7.4	Enhanced CRAFT Post-processing	68
7.5	Height determination process	69
7.6	Text Detection Model Train Dataset Examples	72
7.7	Narrowed multiple text lines image using line spacing pixel which get from the font size height by multiplying the line spacing ratio.	73
7.8	Type of segmentation in texts	74
7.9	Examples of text region detection for each model on font text.	79
7.10	Examples of text region detection for each model on handwritten texts.	80
8.1	TrOCR model architecture	83
8.2	Single line image synthesis	84
8.3	TrOCR model original paper preprocess method and this thesis proposed method	85
8.4	TrOCR model pre-training losses graph	87
8.5	TrOCR model pre-trained model character error rates graph	88
8.6	TrOCR model Kanji single line training, eval losses and eval cers graphs without loss explosion	90
8.7	TrOCR model Kanji single line training, eval losses and eval cers graphs with loss explosion	91
8.8	TrOCR model multiple line training, eval losses and eval cers graphs without loss explosion	94
8.9	TrOCR model multiple line training, eval losses and eval cers graphs with loss explosion	95

List of Tables

6.1	Dataset for the Y-AE model training and the number of generating handwritten character images.	51
6.2	Number of character images used in training for each model.	53
6.3	Statistics of the MSE scale between character images of the same character type. The total number of real images is 18,400, including 92 types of Hiragana and Katakana characters in ETL-5 and ETL-9, 200 images for each character. The number of generated images is also 18,400, including 200 randomly selected images for each character type from the generated images by the Y-AE generators.	56
6.4	Character classification accuracy (acc.) for each model. The architecture of the classification model was the same for all. DA indicates whether the three data augmentation functions were applied to the images in a mini-batch or not when a classifier is trained, ✓: DA is applied, ✗: DA is not applied	57
7.1	Single-line text detection accuracy of each detection method when the IoUs were $0.50 / 0.75$ which separated with / symbol. The numbers in the upper, middle, and lower rows in each cell are the results for the font test set only, the handwritten test set only, and both the test sets, from Test Set A. . . .	76
7.2	OCR accuracy (CER [%]) results for the text detected using each single-line detection method. The CERs are for the typeset dataset only because the OCR engine supports only typeset characters.	77
7.3	Results for spacing: -0.1 (recall, precision, F1 score, correct segmentation, Over Segmentation, and Under Segmentation for IoU of 0.50 and 0.75 on Test Set B.)	78
7.4	Results for spacing: 0.0 (recall, precision, F1 score, correct segmentation, Over Segmentation, and Under Segmentation for IoU of 0.50 and 0.75 on Test Set B.)	78
7.5	Results for spacing: 0.1 (recall, precision, F1 score, correct segmentation, Over Segmentation, and Under Segmentation for IoU of 0.50 and 0.75 on Test Set B.)	78
8.1	The statistics of pre-training handwritten text line images	84

8.2	Character error rates of single-line Hiragana and Katakana text images with randomly generated(RG) image, ✓: DA is applied, ✗: DA is not applied	88
8.3	Character error rates of single line Kanji text images with RG image (DA only applies on ETL or Y-AE based single line synthesis), ✓: DA is applied, ✗: DA is not applied	89
8.4	Character error rates of multiple-lines (Includes Kanji text images by using randomly generated(RG) from single line text images, ✓: DA is applied, ✗: DA is not applied	93

Chapter 1

Introduction

The era of digital technology has witnessed rapid advancements, especially in the areas of deep learning and Optical Character Recognition (OCR). This thesis explores the effects of deep learning on OCR, examining its development, challenges, and future potential. The next sections of this chapter will take readers through OCR's evolution from its beginnings to its present state as an advanced tool enhanced by deep learning, emphasizing its importance in various fields.

1.1 Background

The digital environment has significantly changes mainly due to advancements in deep learning, which also have impacted Optical Character Recognition (OCR) technology. This change is part of a wider shift in how we create and distribute multimedia content, reflecting a larger movement towards better data processing and access. Initially, OCR technology faced challenges similar to those in early speech recognition, such as accurately converting varied text into a format that machines could read. Early OCR systems were limited by basic computational tools and algorithms, struggling with complex layouts, different fonts, and inconsistent print quality. This period saw OCR systems struggling with these issues, indicative of the early stages of the technology.

The advent of deep learning was a pivotal moment for OCR. The development and adoption of powerful Graphical Processing Units (GPUs), along with improvements in data storage with Hard Disk Drives (HDDs) and Solid State Drives (SSDs), and the accessibility of deep learning frameworks like Tensorflow [4] by Google and Pytorch [5] by Meta (formerly Facebook), built on NVIDIA's CUDA [6], set the stage for more sophisticated OCR processes. This technological progress, alongside the explosion of big data, drove deep learning forward, greatly improving OCR technology. Deep learning now enables modern OCR technologies to detect and recognize text from diverse sources with remarkable accuracy. These systems can process multilingual text, understand hand-written notes, and manage documents with complex layouts. This improvement mirrors

recent advances in text recognition, allowing for more sophisticated interactions with humans. Current OCR technologies benefit from innovative techniques like detecting single lines of text within blocks and using new data augmentation methods for better character classification, increasing OCR systems' accuracy and reliability.

The impact of OCR advancements on society is significant. In business, OCR is crucial for automating data entry, streamlining document management, and making historical archives more accessible. Examples include Tegaki/SmartRead [7], a handwriting recognition tool for Japanese characters, and AI Yomi To-ru [8], an OCR solution by NTT East, showing OCR's commercial and practical relevance. In education, OCR facilitates the digitization of materials, broadening access to knowledge. In healthcare, it helps manage patient records, improving services. Moreover, integrating OCR with technologies like natural language processing and image recognition leads to new, advanced applications. For example, combining OCR with language and retrieval technologies greatly enhances information retrieval systems, making them more effective and user-friendly. As OCR continues to evolve with deep learning and technological advancements, it's part of a larger digital trend towards redefining data processing and access. The ongoing development of OCR, along with its integration with new technologies, is set to transform our interaction with the digital information.

1.2 Related Works

The development of efficient OCR systems has advanced significantly with deep learning and data augmentation techniques. Early contributions to OCR began with Denker et al. [9], who developed a neural network recognizer for handwritten zip code digits, demonstrating neural networks' potential in character recognition and setting a foundation for future research. Data augmentation plays a crucial role in image classification by expanding training data without the need for new data collection. Alumentations by Buslaev et al. [10] offers a comprehensive set of augmentation techniques, enhancing image preprocessing's flexibility and speed, vital for robust image classification, object detection, and also OCR system development.

The IAM database [11], created by Marti and Bunke, provides a substantial collection of English sentence images for offline handwriting recognition, supporting numerous system evaluations and developments. The MNIST database [12] is widely used for benchmarking machine learning models, including OCR, and its extension, the EMNIST dataset by Cohen et al. [13], adds handwritten letters, enabling broader character recognition. Chinese handwriting recognition, challenged by character complexity, benefits from the CASIA database [14] and the SCUT-EPT dataset [15], supporting models in achieving high accuracy in Chinese character recognition. Generative models like the SimMIM framework by Xie et al. [16] and Masked Autoencoders by He et al. [17] introduce new pre-training methods using masked image modeling, relevant for synthetic character im-

age generation for OCR training. Innovations continue with masked feature prediction by Wei et al. [18] for self-supervised visual pre-training, which possibly enhancing OCR model training. AutoAugment by Cubuk et al. [19] automates the search for optimal augmentation strategies, beneficial for handwriting recognition with varied writing styles. Image recognition advancements include the Manifold Mixup concept by Bastien M. and colleagues [20], improving character recognition across languages.

The image generation sector achieved a notable advancement with the advent of Generative Adversarial Networks (GANs), as introduced by Goodfellow et al. [21]. This development led to the creation of diverse GAN models, each designed for specific uses. CycleGAN, brought forth by Zhu et al. [22], is a variant aimed at transferring styles between unpaired images, facilitating the conversion of one image style to another without the need for paired samples. B. Chang’s research [23] utilized CycleGAN to generate handwritten Chinese characters by transferring styles from machine-printed fonts, although it was restricted to producing one handwriting style from a single font image. Expanding on GANs and Variational Autoencoders (VAEs), Kong et al. developed cCGAN and cC-VAE models [24] for creating handwritten Chinese characters. Despite their innovative approach, these models struggled with consistent image quality and computational issues related to the Kullback–Leibler (KL) divergence [25, 26] loss.

Simultaneously, Gatys et al. [27] introduced the neural style transfer technique, using neural networks to modify images by applying styles from one domain to another. This approach was significant for its introduction of style and content loss, proving that Convolutional Neural Networks (CNNs) [28] could effectively distinguish between style and content. Ulyanov et al. furthered this field with their TextureNetwork [29], aimed at improving the transfer of complex textures. An important enhancement in this area, also by Ulyanov [30], was the substitution of batch normalization [31] with instance normalization, which boosted the TextureNetwork’s efficiency. This was achieved by calculating the mean and variance independently for each channel and sample, considering the spatial dimensions. In the field of Japanese character recognition, Kitagawa et al. [32] have utilized a Y-Autoencoder (Y-AE) [3] to automatically generate character images for model training. This study indicates that deep learning can improve OCR recognition performance without human effort by generating multiple types of character image data at once from a single image.

The evolution of OCR technology has been paralleled by the advancements in text detection methods. The EAST algorithm by Zhou et al. [33] revolutionized text detection by providing an efficient solution for detecting rotated characters. Meanwhile, Liao et al. [1] and the subsequent improvement with DBNet++ [2] offered robust methods for detecting text by estimating the surrounding regions of character regions. In the endeavor to handle arbitrarily shaped text, Long et al.’s TextSnake [34], ABCNet by Y.L.Liu et al. [35] and the TextFuseNet by Ye et al. [36] have introduced innovative ways to detect text with various shapes and curves. These methods are particularly relevant for OCR systems as they deal with the real-world complexity of text presentation. Character

Region Awareness for Text Detection (CRAFT) [37] has been a significant contribution by Baek et al., focusing on character-by-character detection using Gaussian heatmaps, and it has been further refined for end-to-end text spotting [38]. This level of granularity in detection is crucial for high accuracy OCR in document layout analysis [39, 40, 41, 42, 43, 44, 45].

With the emergence of deep learning, a paradigm shift has occurred in the development of text recognition models. Advanced methodologies like machine translation that use sequence-to-sequence (seq2seq) learning [46], seq2seq contrastive learning for text recognition [47], comprehensive end-to-end training strategies such as Convolutional Recurrent Neural Network (CRNN) [48] trained with Connectionist Temporal Classification (CTC) [49] have significantly contributed to the creation of cutting-edge text recognition systems [50, 51, 52, 53, 54, 55, 56, 57]. To facilitate the transition from research to practical applications in deep learning fields, toolkits such as EasyOCR [58], MMOCR [59], PP-OCR [60] and PP-OCRv2 [61] has made it possible for developers to easily implement state-of-the-art OCR systems in various applications. For instance, EasyOCR provides implementation such as CRNN [48] text recognizer for single-line recognition with different backbones of feature extractor such as ResNet [62], Recursive Recurrent Nets with attention (RCNN) [63], VGG [64], which all with a head of Bidirectional Long Short-Term Memory trained with CTC loss [49]. This toolkit, alongside others, is helping bridge the gap between research outcomes and real-world usage.

The evolution from the initial stages of OCR systems to the current era of deep learning-enhanced technologies marks a profound shift, driven by collaborative progress in creating datasets, innovating models, and advancing methodologies. Each contribution, from datasets like IAM [11], MNIST [12], and CASIA [14], to models and algorithms like SimMIM [16], AutoAugment [19], and CRAFT [37], has played a part in shaping the OCR landscape. As the field continues to evolve, the integration of these developments promises to further enhance the capabilities and applications of OCR technology.

1.3 Problem Statement

The advent of deep learning models has led to various innovations in model structures and data augmentation techniques. However, a significant challenge persists in enhancing character recognition rates: the reliance on human-generated, high-cost data creation, apart from basic image processing-based data expansion. Furthermore, as discussed in section 1.2, most existing research primarily focuses on evaluating methods using generated images, with limited exploration of the real-world impact of these generated images on character recognition rates. This study aims to address this gap by examining the contribution of deep learning-generated images to the improvement of character recognition rates.

Character recognition typically involves detecting individual characters or lines of

text, followed by their recognition using specialized models. However, in modern official and complex documents, characters are frequently handwritten, printed, or typed in constrained spaces. This is often an obstacle to accurate character recognition since several lines of text can be mistaken for a single block of text due to the close spacing between lines, resulting in misrecognition. This research seeks to explore and devise processing methods for the outputs of deep learning models, specifically tailored to overcome these challenges in character recognition in tightly spaced textual environments.

Furthermore, single-line character recognition methods are predominantly utilized in deep learning models. Typically, this approach involves resizing and recognizing characters on a line-by-line basis, as seen in methods like CRNN [48] which trained with CTC loss [49] and Transformer-based Optical Character Recognition with Pre-trained Models (TrOCR) [65]. Alternatively, it involves processing a single line of characters by adeptly integrating features extracted from multiple image blocks, a technique outlined in Daiz’s paper [66]. These methods, however, primarily focus on straightforward scenarios with clear line separations and may not effectively handle complex document layouts where characters are closely packed or overlap.

1.4 Research Objective

This study focuses on three primary research objectives, each aimed at advancing the field of OCR through the application of deep learning techniques:

1. To propose a novel method that improve the accuracy of OCR by generating training data from limited training data using Deep Learning models.
2. To propose a Deep Learning model and a novel and efficient post-processing algorithm that can recognize narrow multi-line characters more accurately, which is a problem with conventional Deep Learning models and algorithmn.
3. To propose a method that can recognize multiple-lines of text in an image, as opposed to the conventional deep learning model that recognizes only single-line text images.

1.5 Research Scope

The research scope in this thesis delves into the application of deep learning in Optical Character Recognition (OCR), primarily focusing on the Y-AE and CRAFT and models. It commences by examining the transformation of OCR systems with the integration of deep learning, highlighting the limitations of traditional OCR and the enhancements brought by these advanced computational methods.

The core of this research is the exploration of the Y-AE model, an innovative approach for generating diverse character images [32]. This model, augmented with Adaptive Instance Normalization, is pivotal in creating a varied dataset for OCR training, particularly enhancing the recognition of handwritten characters. A significant portion of the research is dedicated to the CRAFT model, especially its application in single-line text detection within multi-line text blocks. The integration of a specialized post-processing algorithm with CRAFT plays a crucial role in refining text detection accuracy, particularly in complex OCR scenarios involving diverse fonts and handwriting. The thesis presents a comprehensive evaluation of these models through experimental analysis, using varied datasets to assess their effectiveness in improving OCR performance. The results from these experiments underscore the potential of Y-AE and CRAFT models in advancing the field of OCR. This thesis also aims to explore and develop more simple approaches that can recognize characters in multi-line text environments primary with the state of the art model architecture, TrOCR [65], which only accept a fixed size input image. It expanding the capabilities of deep learning models beyond traditional one-line recognition methods.

1.6 Contributions

The research contributes to three primary areas within the field of OCR, as extensively detailed in the thesis.

Firstly, a major achievement of this study is the development and deployment of a deep learning model designed to create a variety of character images from a single input style image. This model’s ability to produce images that improve the performance of character classifiers marks a noteworthy advancement. It not only generates Hiragana and Katakana images but also Kanji, demonstrating the model’s broad applicability. Furthermore, the research verifies that these images serve not just in training models for single character recognition but also for recognizing single-line text images. This method enhances the accuracy of character recognition while broadening the range of characters that current systems can process.

Secondly, this study introduces an efficient post-processing technique for enhancing text detection in texts with closely spaced lines, a scenario where conventional models and their post-processing methods fall short. This new approach substantially boosts the text recognition rate per line by mitigating the common issue of multiple lines being misidentified as a single text block. Therefore, this advancement proposes a more effective and precise character detection method, marking a significant stride in text recognition by improving character spacing and line segmentation, thus elevating the overall efficiency and accuracy of text detection systems.

Thirdly, the research tackles a specific challenge in the conventional TrOCR system, which is limited the Vision Transformer (ViT) [67] that it only accepting a fixed size images due to its ViT-based feature extractor and Encoder-Decoder model architecture.

Recognition of multi-line text is often hindered by this limitation. This thesis introduces a solution that maintains the original image width's aspect ratio during pre-processing. By adjusting the image to a set height and aligning it to fit TrOCR's fixed height and width image requirement, it successfully demonstrates the capability for multi-line text recognition within TrOCR's constraints. This method enables systems using a ViT backbone, like TrOCR, to perform multi-line text recognition by applying a strategic pre-processing step to the input images.

1.7 Organization of this thesis

This thesis is organized into the following chapters, providing a comprehensive exploration of character recognition improvement through deep learning and related technologies:

Chapter 1: Introduction - This chapter introduces prior research that has contributed to the improvement of character recognition with generated image using deep learning model and single-line text detection with improved deep learning model and proposed post-processing method. It outlines the scope and aims of the current study, setting the stage for the detailed discussions that follow.

Chapter 2: Text Image Recognition - This chapter delves into the history of character recognition systems. It examines their fundamental principles, the challenges they face, and the roles they play in various applications.

Chapter 3: Deep Learning - An in-depth exploration of deep learning is presented in this chapter. It covers the foundational concepts, the evolution of the technology, and its impact on the field of character recognition.

Chapter 4: Text Detection - Here, the focus shifts to the specific aspect of character detection to text detection. This chapter discusses the methods and technologies used to identify and isolate characters from complex backgrounds and the application of deep learning in enhancing character detection capabilities. It examines the advancements and improvements brought about by integrating deep learning techniques.

Chapter 5: Image Generation Using Deep Learning - This chapter explores how deep learning can be used for character image generation. It highlights the methods and models that have been successful in creating diverse character sets for training and recognition purposes.

Chapter 6: Character Generation with Y-Autoencoder - Here, the Y-Autoencoder's role in character generation is discussed. The chapter elaborates on how this model can create varied character images, contributing by improving the character recognition rate in field of OCR.

Chapter 7: Single-line Text Detection In Multiple-lines Text Images - This chapter discusses a text detection deep learning model that employs an enhanced CRAFT (Character Region Awareness For Text detection). It examines the model's structure, functionality, and advantages with proposed post-processing method and experiments results.

Chapter 8: Text Recognition Model - This chapter explores the architecture and capabilities of TrOCR in recognizing single-line and multi-line texts. It also examines TrOCR's effectiveness with datasets including ETL and Y-Autoencoder generated images explained in chapter 6, highlighting its adaptability in various text recognition scenarios.

Chapter 9: Summary And Future Works - The final chapter summarizes the study, highlighting its key findings, contributions to the field, and potential areas for future research.

Chapter 2

Text Image Recognition

Chapter 2 explores the evolution of Text Image Recognition, highlighting the journey from ancient texts to modern digital formats. It focuses on the pivotal role of Optical Character Recognition (OCR) in changing how we process data, tracing its development and technological progress. Additionally, this chapter examines Handwritten Text Recognition (HTR), pointing out the challenges it faces and how deep learning has improved its precision. The transition from basic OCR to advanced HTR represents significant progress in recognizing and interpreting text, affecting many areas and reshaping our digital text interactions.

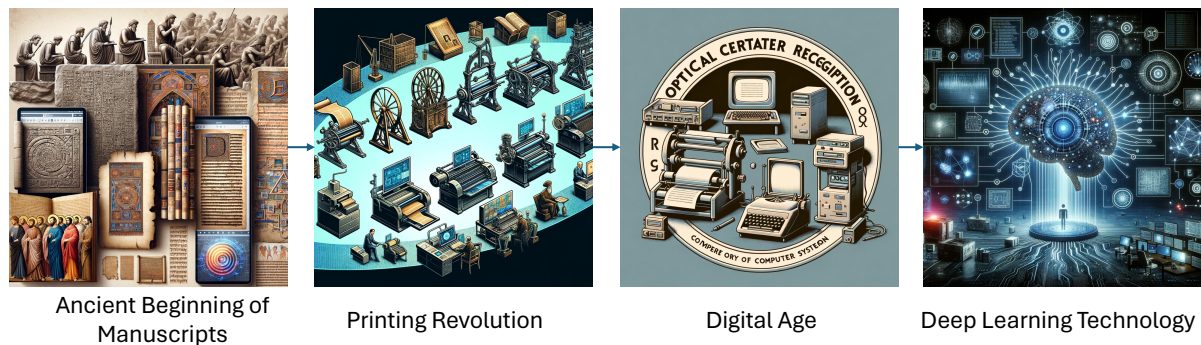


Figure 2.1: The history of printing revolution to recent OCR
(Generated by ChatGPT)

2.1 The History of Character and Text Images

Figure 2.1 illustrates the development of characters and text images from ancient manuscripts through to the digital era and the rise of deep learning. The evolution of text representation reflects human progress, starting with early cave etchings, progressing to papyrus in ancient Egypt, and later parchment and vellum in the Middle Ages. The invention of the printing press by Johannes Gutenberg in the 15th century marked a decisive step forward,

facilitating border access to knowledge and catalyzing intellectual and social shifts. The digital age brought another transformation, making text more dynamic, accessible, and interactive via digital platforms and the internet. This phase has had a transformative impact on the way we interact with text, influencing education, business, and social connections. The transition from physical to digital text highlights the changing dynamics between humans and written communication, emphasizing the continuous evolution of how we share and interact with information.

2.2 Optical Character Recognition (OCR)

2.2.1 Origins and Early Development

The beginnings of OCR trace back to early telegraphy innovations like morse code and computing efforts aimed at converting printed text into a format computers could understand. Initially designed to aid visually impaired people, OCR quickly expanded its reach, transforming how we manage printed materials. A significant figure in OCR's development was Gustav Tauschek, whose Reading Machine [68] in the early 20th century marked a key milestone. Using light rays and photoelectric cells, the machine could interpret text for punchcard calculating devices. Tauschek was a pioneer in OCR, moving it from theoretical concepts to practical use, with over 200 patents to his name, 169 of which were acquired by IBM in 1929 and in the United States in 1935, before World War II [69].

Following Tauschek's innovation, engineers expanded on his ideas, leading to developments like text-to-morse conversion in 1951 and handwriting recognition in 1966 [69]. The creation of the first computer font in 1968 was another milestone, setting the stage for digital text representation. In 1974, Ray Kurzweil's company introduced the first OCR program capable of recognizing various print styles. This technology underpinned the Kurzweil Reading Machine for the blind, which integrated omni-font OCR, CCD flat-bed scanners, and text-to-speech technology [70]. Computer fonts later became fundamental to personal computing, exemplified by the Apple II [71], developed by Steve Jobs and Stephen Gary Wozniak. By the mid-1990s, the emergence of Portable Document Format (PDF) by Adobe [72] marked a new era in digital data storage, facilitating document management for both organizations and individuals with personal computers.

2.2.2 Pattern Matching in OCR Technology

OCR systems initially utilized pattern matching [73, 74], relying on sensors to recognize character patterns. This approach marked a pivotal point in OCR's development, as it leveraged machine learning and image processing to enhance text recognition. In this method, text images are scanned and matched against pre-defined character patterns or templates, facilitating text identification and interpretation. Initially, OCR processes

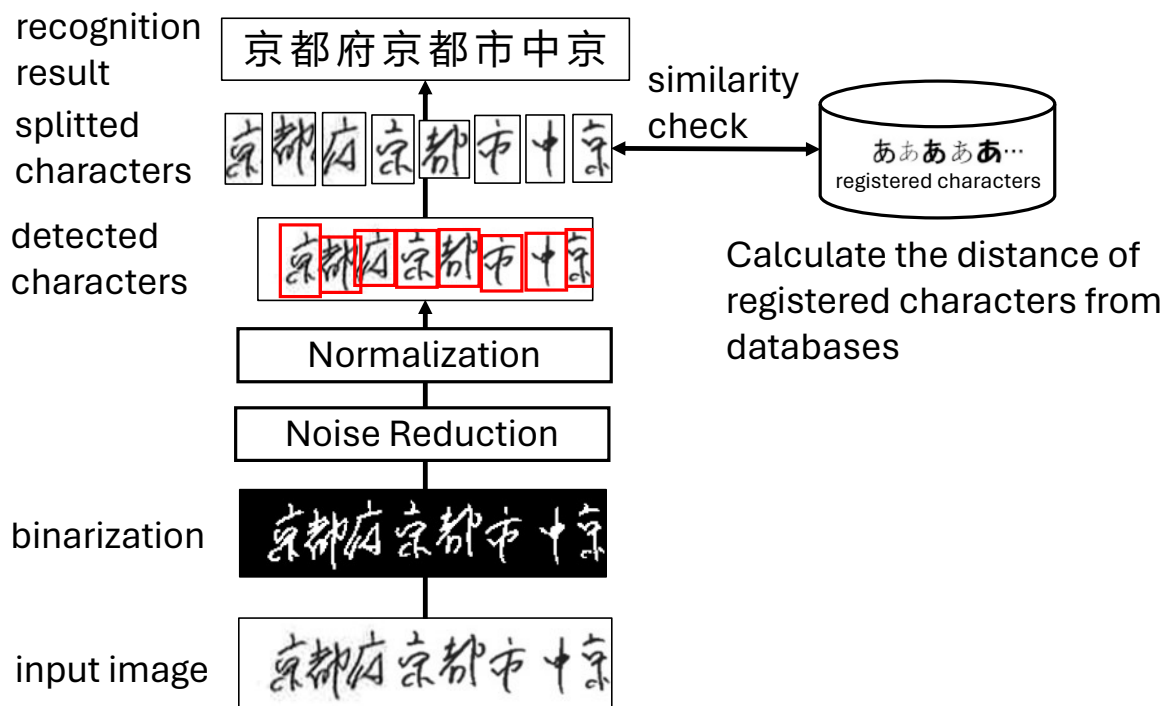


Figure 2.2: Pattern matching-based OCR

were labor-intensive and limited, capable of identifying only a narrow selection of fonts and characters. With the evolution of computing and image processing, pattern matching OCR's abilities expanded considerably, integrating machine learning algorithms. This progress allowed OCR systems to learn and recognize a broader array of fonts and styles, significantly improving accuracy and efficiency.

Early OCR systems faced challenges with input image quality, including issues with text size, image resolution, font style, and background noise. This required image pre-processing, typically involving:

1. *Binarization*: Transforming images into a binary (black and white) format to distinguish text from the background more clearly.
2. *Noise Reduction*: Applying methods like the Sobel filter [75] to reduce image noise.
3. *Normalization*: Standardizing text size and orientation for uniformity across documents.

The Sobel filter [75], an edge detection algorithm, can be mathematically represented as:

$$G = \sqrt{G_x^2 + G_y^2} \quad (2.1)$$

where G_x and G_y are the horizontal and vertical derivatives of the image, respectively. Figure 2.2 demonstrates the character recognition pipeline, applying pre-processing steps

like binarization, noise reduction, and normalization. The detected character image is then compared for similarity with pre-stored characters in the database to determine character recognition.

These early OCR systems' machine learning aspects were fundamental, mainly aimed at refining the pattern matching to accommodate text appearance variations. Adjustments were generally limited to familiar variations within a set of fonts or styles. Despite these limitations, these initial OCR systems merging pattern matching with basic machine learning signified a substantial leap forward. They established the groundwork for more advanced OCR technologies, enabling automated text recognition, data entry, document digitization, and information retrieval. As computer vision and machine learning advanced, OCR technology shifted dramatically. Modern OCR systems employ sophisticated algorithms, including neural networks and deep learning, to recognize a wide range of text types and languages with high accuracy, even in challenging scenarios such as low-quality scans or handwritten text.

2.3 Handwritten Text Recognition

2.3.1 Handwritten Text Recognition: Evolution, Techniques, and Applications

Handwritten Text Recognition (HTR) has become a key component in advancing text processing technologies, marked by its distinct challenges in deciphering human handwriting. Handwriting varies greatly due to personal styles, differences in character shapes and sizes, and inconsistencies in stroke, pressure, and spacing. The task becomes even more complex with cursive writing, where characters are connected, and clear separations are often absent. HTR technologies fall into two main groups: offline and online. Offline HTR analyzes static images of handwritten text, like scanned documents or photos, concentrating on the text's visual features. Initially, offline HTR methods focused on extracting features specific to handwriting for recognition but frequently struggled due to handwriting's inherent complexity and variability.

Online HTR, on the other hand, deals with handwriting captured in real-time, as it is written. It gathers dynamic information such as stroke order, direction, and pressure, offering more information for analysis. M.Okamoto [76] introduced methods for recognizing online handwritten characters by leveraging directional features, non-linear normalization, and writing area characteristics. This approach typically involves digital pens or touchscreens, facilitating a deeper insight into individual handwriting nuances.

The integration of deep learning technologies, particularly Recurrent Neural Networks (RNNs) [77] and Convolutional Neural Networks (CNNs) [28], has substantially improved HTR's capabilities. RNNs, along with their variant Long Short-Term Memory (LSTM) networks [78], are adept at processing sequences, making them ideal for understanding the

contextual continuity of handwriting. Nguyen.H [79] explored online Japanese text recognition, while offline signature verification systems have employed RNNs [77] to analyze the sequential flow of handwritten texts. F.Yang’s work [80] on online handwritten Mongolian character recognition utilized a convolutional-based seq2seq model. These models are proficient in managing the temporal aspects of online HTR and the sequential nature of offline text. CNNs, meanwhile, excel in feature extraction from handwriting images, identifying spatial patterns to discern characters and words. Combining CNNs for feature extraction with RNNs for sequence processing has significantly increased HTR accuracy.

HTR finds applications in various fields, from digitizing archival documents to automating business data entry. It’s instrumental in converting handwritten documents into digital, searchable formats, proving invaluable in archival and record-keeping tasks. Future advancements in HTR aim to improve model robustness, speed, and adaptability to different handwriting styles. Research is directed towards developing systems that can handle multilingual texts, recognizing ambiguous characters, and process low-quality images more effectively. As HTR technology evolves, it’s expected to further close the gap between analog and digital formats, enhancing accessibility and simplifying data management across numerous sectors.

2.4 Summary

This chapter offers a thorough examination of Text Image Recognition, focusing specifically on the historical progression and advancements in OCR and HTR. It begins with the early history of text representation, from ancient manuscripts through to pivotal developments like the printing revolution and the onset of digital technology. Section 2.2 begins with OCR’s origins, highlighting the contributions of early innovators such as Gustav Tauschek and the journey to current OCR technologies. It details the evolution from basic pattern-matching methods to today’s complex algorithms, noting OCR’s significant role in transforming data processing. The section also addresses the initial challenges faced by OCR systems, including limitations related to font and text quality, and discusses image pre-processing techniques (binarization, noise reduction, normalization) developed to mitigate these issues.

The HTR discussion centers on the specific difficulties of interpreting human handwriting. It distinguishes between offline and online HTR, delineating their methods and uses. The introduction of deep learning, especially through RNNs, LSTMs, and CNNs, is spotlighted for its crucial contribution to advancing HTR systems. The chapter also explores HTR’s practical applications across various fields, such as the digitization of historical documents and automation in data entry, demonstrating the broad impact of this technology.

In summary, the chapter provided a comprehensive overview of the journey of text image recognition technologies, underscoring their historical significance and the techno-

logical advancements that have shaped their evolution. It presented a clear picture of how these technologies have revolutionized the way we process, manage, and interact with text in the modern digital era.

Chapter 3

Deep Learning

Deep learning, a subset of machine learning and artificial intelligence (AI), involves training large neural networks to model and understand complex patterns in data. It's inspired by the structure and function of the human brain, particularly the interconnections of neurons. Deep learning models can learn to perform tasks like image recognition, speech recognition, and natural language understanding with a high level of accuracy. Historically, the concept of neural networks dates back to the 1940s, but it wasn't until the 1980s and 1990s that key developments by Geoffrey Hinton [81], Yann LeCun [82], and Yoshua Bengio [83], among others, propelled the field forward. These researchers laid the foundation for many of the algorithms and architectures used in deep learning today.

3.1 Deep Neural Forward Network

Deep neural networks are structured as a series of layers, each composed of multiple nodes or neurons, interconnected in a hierarchical manner. These layers include an input layer, one or more hidden layers, and an output layer. The input layer receives external data, represented as \mathbf{x} , which then flows through the hidden layers before reaching the output layer that produces the final output \mathbf{y} .

Each neuron in a layer is a computational unit that performs specific calculations on its input. The input to a neuron in any hidden layer is the output from the neurons of the preceding layer. This input-output relationship can be expressed mathematically for a neuron as:

$$\mathbf{z} = f(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}) \tag{3.1}$$

where:

- \mathbf{W} represents the weights that the neuron assigns to its inputs.
- \mathbf{x} is the input vector.
- \mathbf{b} denotes the bias, which adjusts the output alongside the weighted sum.

- $\hat{\mathbf{y}}$ is the output vector.

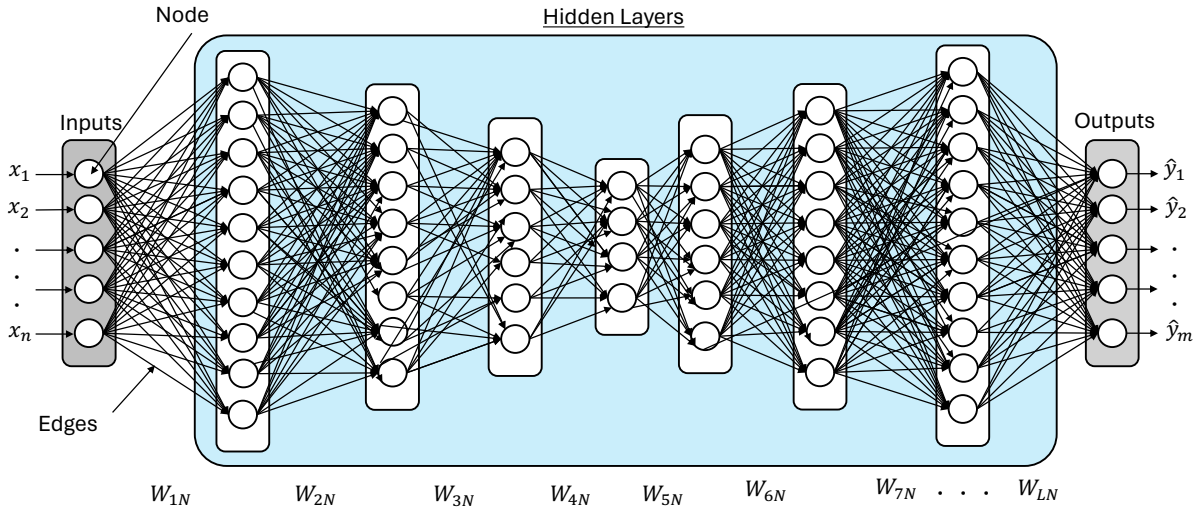


Figure 3.1: Simple Deep Neural Forward Networks

Figure 3.1 shows a simple feed-forward neural network architecture. The process begins at the input layer, where each neuron/node corresponds to a feature in the input data represented by x . The output of these neurons is passed to the first hidden layer. Hidden layers, which are the core of the neural network, consist of neurons that perform computations and transformations on the received inputs. The number of hidden layers and the number of neurons in each layer define the network’s architecture and its capacity to learn complex patterns.

Each neuron in a hidden layer computes the weighted sum of its inputs and applies an activation function to this sum. The weights (represented as $W_{1N}, W_{2N}, \dots, W_{LN}$ for a network with L layers and N neurons) are parameters that the network learns during training. The training process involves adjusting these weights to minimize the difference between the predicted output $\hat{\mathbf{y}}$ and the actual output \mathbf{y} . This is done using algorithms like backpropagation [84] combined with optimization techniques such as stochastic gradient descent [85]. The final layer, the output layer, produces the network’s output. This layer’s neurons are responsible for generating the predictions or decisions of the neural network. The design of the output layer varies depending on the task (e.g., regression, classification). For instance, a softmax activation function might be used in the output layer for a multi-class classification problem, which turns the raw output into probabilities for each class.

In summary, deep neural networks use a series of interconnected layers with trainable weights and biases to transform input data \mathbf{x} into meaningful outputs \mathbf{y} . The network’s ability to learn complex patterns is governed by the architecture and the training process, where the optimal set of weights and biases are determined.

3.1.1 Activation Functions

Activation functions in neural networks are critical for introducing non-linear properties to the model, enabling it to learn and represent more complex patterns that linear models cannot. They determine whether a neuron should be activated or not, based on the weighted sum of its inputs. Common activation functions include:

- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$, a smooth function that outputs values between 0 and 1, making it suitable for binary classification tasks.
- ReLU (Rectified Linear Unit): $f(x) = \max(0, x)$, commonly used in hidden layers, allows models to converge faster and learn effectively by resolving the vanishing gradient problem.
- Tanh (Hyperbolic Tangent): $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, outputs values between -1 and 1, providing a scaled output compared to the sigmoid function and often used in hidden layers for balanced learning.

The choice of activation function could influence the performance and convergence of neural networks. For instance, ReLU is preferred in deep networks due to its computational efficiency and ability to mitigate the vanishing gradient issue, common in networks with sigmoid or tanh activations. In contrast, sigmoid and tanh are often used in scenarios where normalized outputs are crucial, such as in the output layer for binary classification (sigmoid) or when data normalization is essential (tanh).

3.1.2 Backpropagation

Backpropagation [84], a fundamental concept in training neural networks, is the mechanism through which the network learns by adjusting its weights. It involves the computation of the gradient of the loss function with respect to each weight in the network and uses this information to update the weights, thereby minimizing the loss. The process begins with the forward pass, where the input \mathbf{x} is passed through the network to produce a prediction $\hat{\mathbf{y}}$. The prediction is then compared to the actual label \mathbf{y} , and a loss function $L(\hat{\mathbf{y}}, \mathbf{y})$ is calculated to quantify the error of the prediction. The backpropagation process involves computing the gradient of the loss function with respect to each weight in the network. This gradient $\nabla_{\mathbf{W}}L$ signifies how much a small change in each weight would affect the loss. The chain rule of calculus is employed to calculate these gradients, effectively propagating the error information back from the output layer to the input layer. The formula for the gradient computation at each layer is given by:

$$\frac{\partial L}{\partial \mathbf{W}_i} = \frac{\partial L}{\partial \hat{\mathbf{y}}} \times \frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{z}_i} \times \frac{\partial \mathbf{z}_i}{\partial \mathbf{W}_i} \quad (3.2)$$

where:

- $\frac{\partial L}{\partial \hat{y}}$ is the derivative of the loss function with respect to the network's output.
- $\frac{\partial \hat{y}}{\partial z_i}$ is the derivative of the output with respect to the weighted sum at layer i .
- $\frac{\partial z_i}{\partial \mathbf{W}_i}$ is the derivative of the weighted sum at layer i with respect to the weights \mathbf{W}_i of that layer.

Once the gradients are computed for all weights, they are used to update the weights in the direction that minimizes the loss. This is typically done using an optimization algorithm such as stochastic gradient descent (SGD) [85]. The weights are updated according to the formula:

$$\mathbf{W}_i \leftarrow \mathbf{W}_i - \eta \frac{\partial L}{\partial \mathbf{W}_i} \quad (3.3)$$

where η is the learning rate, a small positive value that determines the step size of the weight update. Backpropagation is repeated for many iterations or epochs over the training data, allowing the neural network to learn and adjust its weights to minimize the loss function, thereby improving its prediction accuracy on the given task.

3.2 Optimizers in Deep Learning

Optimizers are algorithms or methods used to change the attributes of the neural network, such as weights and learning rate, to reduce the losses. Optimizers help to minimize (or maximize) an Objective function (another name for Loss function) that maps some set of variables (like the weights and biases in a neural network) to a real number representing how well the neural network performs.

3.2.1 Stochastic Gradient Descent (SGD)

SGD [85] is a simple yet very efficient approach to fitting linear classifiers and convex loss functions such as (linear) Support Vector Machines and Logistic Regression. The core idea is to update parameters in the opposite direction of the gradient of the objective function with respect to the parameters. Mathematically, the update rule for parameter θ is:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (3.4)$$

where η is the learning rate and $\nabla_{\theta} J(\theta)$ is the gradient of the loss function J with respect to θ .

3.2.2 Adam Optimizer

Adam [86] is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. Adam combines the

best properties of the AdaGrad [87] and RMSProp [88] algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems. The update rules for the parameters θ are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_{\theta} J(\theta) \quad (3.5)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_{\theta} J(\theta))^2 \quad (3.6)$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (3.7)$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (3.8)$$

$$\theta = \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \quad (3.9)$$

where m_t and v_t are estimates of the first and second moments of the gradients, respectively, and β_1, β_2 are the exponential decay rates for these moment estimates.

3.2.3 RMSprop Optimizer

RMSprop (Root Mean Square Propagation) [88] is an adaptive learning rate method. It was proposed by Geoffrey Hinton to resolve AdaGrad’s radically diminishing learning rates. The update rule for RMSprop is:

$$S_{\text{dw}} = \beta S_{\text{dw}} + (1 - \beta) \nabla_{\theta} J(\theta)^2 \quad (3.10)$$

$$\theta = \theta - \eta \frac{\nabla_{\theta} J(\theta)}{\sqrt{S_{\text{dw}} + \epsilon}} \quad (3.11)$$

3.3 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) [28] represent a significant innovation in the field of deep neural networks, designed particularly for processing data with grid-like structures, such as images. Their exceptional capability in computer vision is attributed to their proficiency in detecting and learning intricate patterns from visual inputs. Convolutional layers, as illustrated in Figure 3.2, execute convolution operations using a set of learnable filters or kernels. Each filter captures specific features from the input, such as edges or textures. The convolution operation at position (x, y) is mathematically expressed as:

$$O(x, y) = \sum_{i=-k}^k \sum_{j=-k}^k I(x + i, y + j) \cdot F(i, j) \quad (3.12)$$

where O is the output, I denotes the input image, F is the filter, and k represents the kernel size. This operation involves sliding the filter across the input with a size of stride

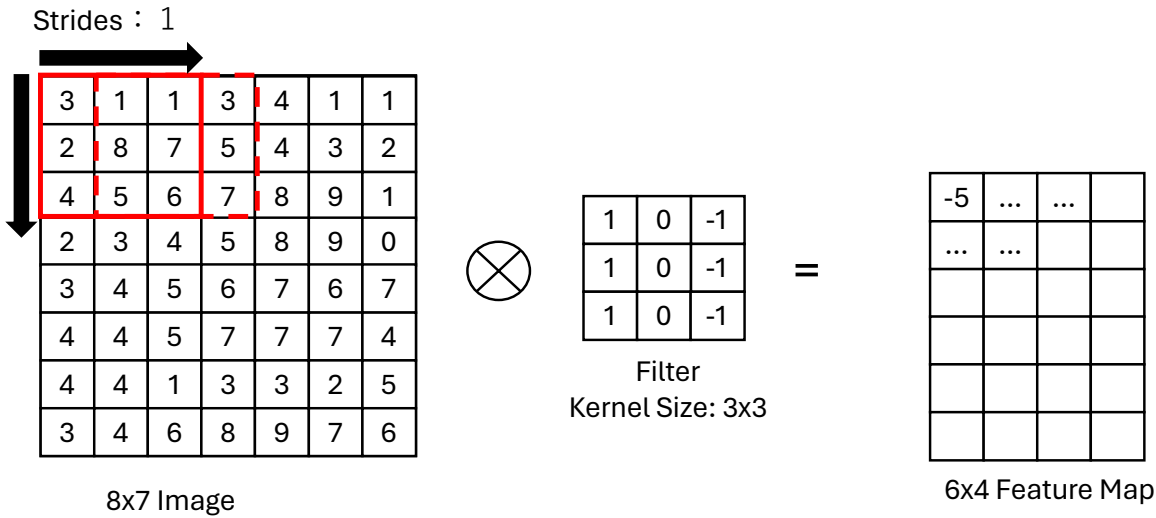


Figure 3.2: Convolution Operations in CNNs

and calculating a dot product at each step, culminating in a summation to produce the output feature map.

Pooling layers serving to reduce the spatial dimensions of the input feature map, these layers, such as Max Pooling, select the most prominent features, thereby decreasing the computational load and the number of parameters. This down-sampling operation enhances the network’s resilience to small translations in the input. Key parameters in convolutional layers, the performance of these layers is influenced by:

- *Kernel Size (k)*: The dimensions of the filters, often 3×3 or 5×5 , which affect the granularity of feature extraction.
- *Stride (s)*: Dictates how the filter moves across the input, with $s = 1$ moving the filter pixel-by-pixel for fine-grained feature detection.
- *Padding (p)*: Adds zero-value pixels around the image’s edges, maintaining the spatial size of the output relative to the input.

As demonstrated in Figure 3.2, the convolution operation on an 8×7 image with a 3×3 kernel results in a 6×4 feature map, showcasing the intricate transformation process inherent in CNNs. CNNs optimize the weights of filters through training, enabling them to efficiently extract and learn relevant features from visual data. Over the years, groundbreaking CNN architectures like LeNet, AlexNet, VGG, and ResNet have emerged, each advancing deep learning with novel concepts and enhanced capabilities.

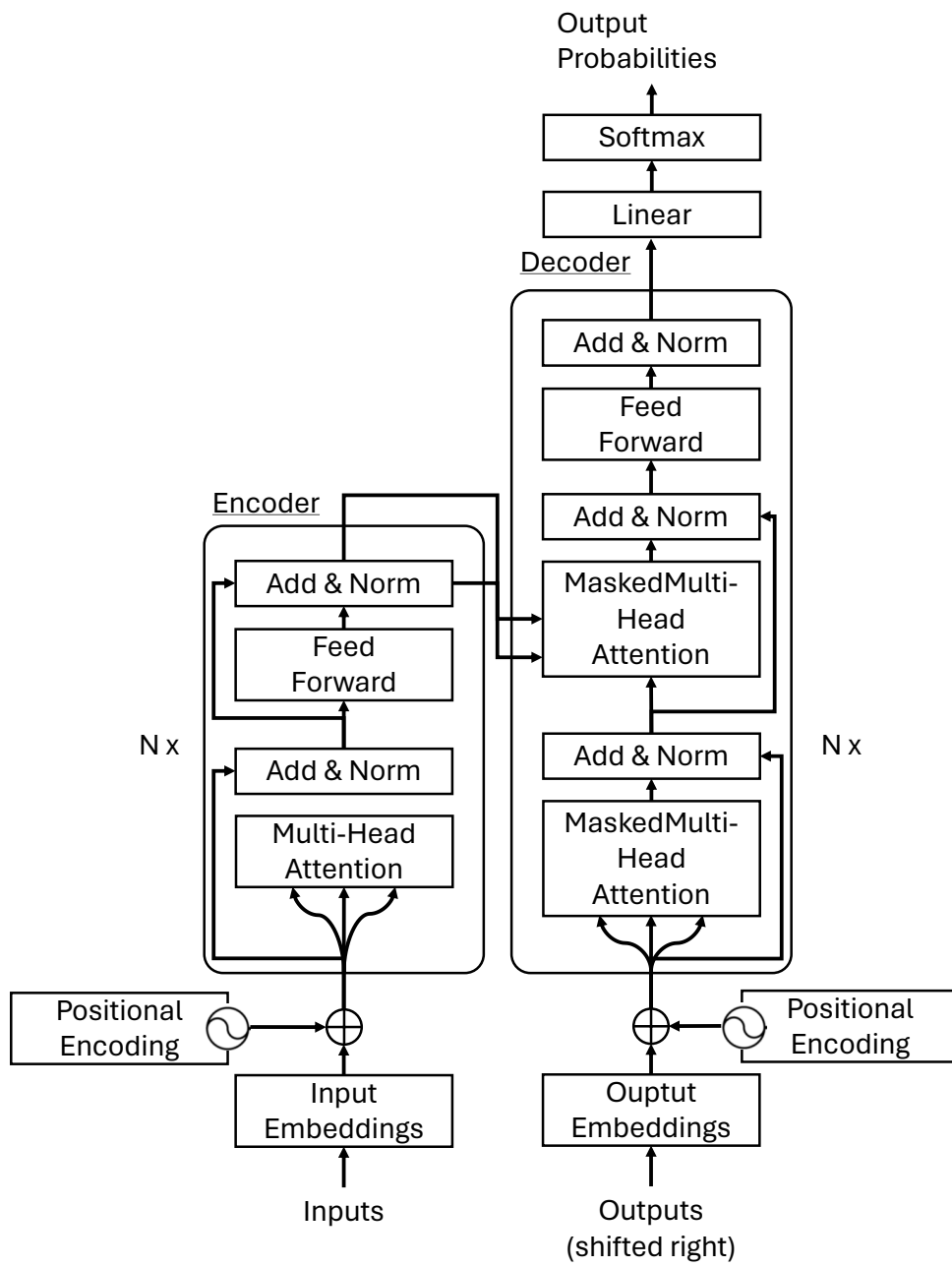


Figure 3.3: Transformer model architecture

3.4 Transformer Model Architecture

The Transformer model, introduced by Vaswani et al. [89], represents a significant advancement in neural network design, especially for tasks involving sequence-to-sequence modeling, such as language translation and text summarization. Its architecture, as shown in Figure 3.3, stands out for its reliance on self-attention mechanisms, eschewing the recurrent layers commonly found in previous models. The Transformer model is divided into two main components: the encoder and the decoder, as illustrated in Figure 3.3. Each of these components consists of a stack of identical layers, with the encoder containing two sub-layers (a self-attention layer and a position-wise fully connected feed-forward network) and the decoder incorporating an additional third sub-layer for encoder-decoder attention.

3.4.1 Encoder

The encoder's role is to process the input sequence and map it into a higher, abstract representation. It does this through a series of layers, each containing:

- A self-attention mechanism that allows the model to weigh the importance of different words in the input sequence.
- A feed-forward neural network that applies to each position separately and identically.

3.4.2 Decoder

The decoder, in turn, takes the encoder's output and generates a sequence of outputs. Its layers are similar to the encoder's but with an added layer that performs attention over the encoder's output. The key operations in the decoder are:

- The masked self-attention layer, which prevents positions from attending to subsequent positions.
- The encoder-decoder attention layer, allowing the decoder to focus on relevant parts of the input sequence.

3.4.3 Self-Attention Mechanism

A core component of the Transformer is the self-attention mechanism. It allows the model to consider other words in the input sequence when encoding a word. The self-attention score for a word is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (3.13)$$

where Q , K , and V represent the query, key, and value matrices, respectively, and d_k is the dimension of the key vector.

3.4.4 Positional Encoding

To account for the order of the words in the input sequence, positional encodings are added to the input embeddings. These encodings have the same dimension as the embeddings, allowing the two to be summed:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (3.14)$$

$$\text{PE}(pos, 2i + 1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \quad (3.15)$$

where pos is the position, i is the dimension, and d_{model} is the dimension of the embedding.

3.4.5 Multi-Head Attention

The Transformer employs multi-head attention to allow the model to jointly attend to information from different representation subspaces:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O \quad (3.16)$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (3.17)$$

where, W_i^Q , W_i^K , W_i^V , and W^O are parameter matrices, and h is the number of heads.

3.5 Vision Transformers (ViT)

The Vision Transformer (ViT) [67] model represents a new approach in the realm of deep learning, applying the transformer architecture—originally designed for natural language processing—to the field of image classification. This adaptation challenges the conventional reliance on CNNs for image processing tasks. Unlike CNNs, which repetitively apply convolution operations across the entire image, ViT treats an image as a sequence of distinct patches, analogous to how transformers process sequences of words in text.

3.5.1 ViT Architecture

The architecture of the ViT introduces a new approach to image processing in the field of vision-based deep learning. By dividing an image into multiple patches, similarly to how sentences are divided into words, ViT processes these patches with a transformer encoder, offering a unique method for handling visual information. The main steps involved in ViT’s processing are as follows:

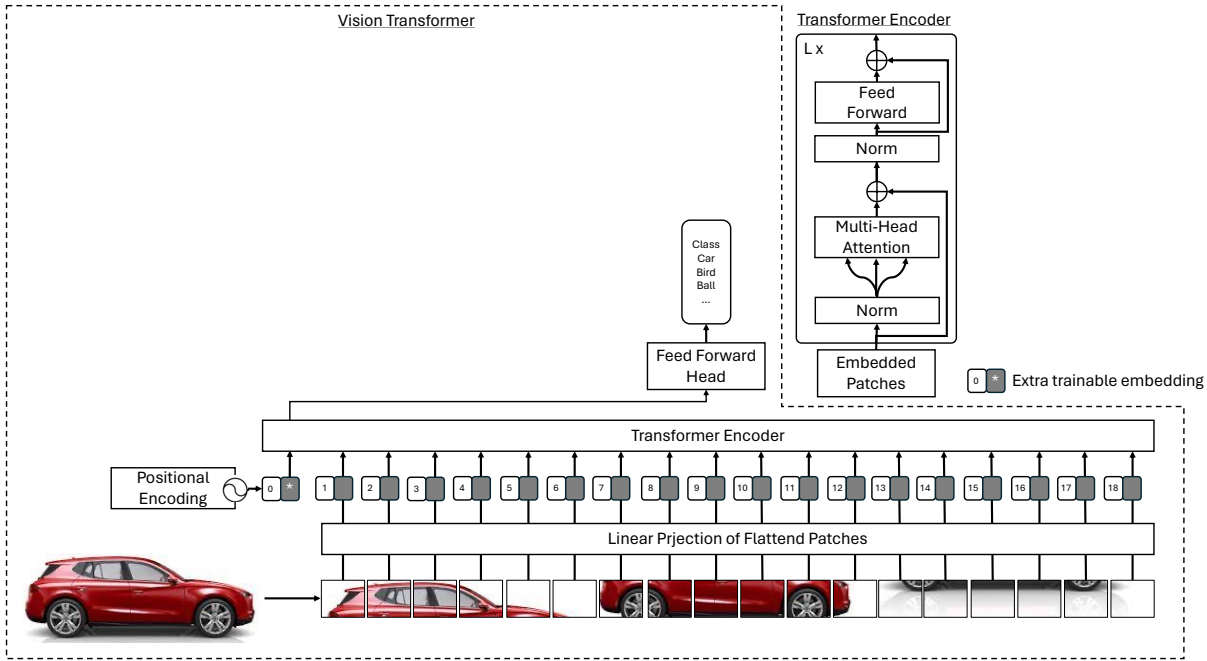


Figure 3.4: Vision Transformer model architecture

1. **Patch Partitioning:** An image is partitioned into a fixed number of patches. These patches are treated as the equivalent of tokens (words) in a language model. This approach allows ViT to capture the intricate details within local regions of an image.
2. **Patch Embedding:** Each patch is flattened and linearly transformed into an embedded vector. This embedding acts as a numerical representation of the patch, analogous to word embeddings in language models.
3. **Positional Encodings:** Similar to the Transformer architecture in NLP (as discussed in Section 3.4), positional encodings are added to the patch embeddings. This is essential to provide spatial context, as transformers, by design, do not inherently understand the order of the input sequence.
4. **Transformer Encoder Processing:** The sequence of embedded patches, now with positional information, is passed through the layers of a standard transformer encoder. The encoder processes these patches, enabling the model to learn and understand the relationships and dependencies between different parts of the image.
5. **Sequence of Encoded Patches:** The transformer encoder outputs a sequence of encoded patches. Each encoded patch now carries global information about the image, having been influenced by every other patch in the sequence.
6. **Classification Token:** For tasks like image classification, a special classification token (often denoted as [CLS]) is prepended to the sequence. The output corre-

sponding to this token, after passing through the transformer layers, is used for the final classification. This output is fed into a linear layer to derive the class scores, making it possible to categorize the image into one of the predefined classes.

The mathematical formulation behind ViT’s processing can be expressed as follows:

$$\mathbf{Z}_0 = [\mathbf{x}_{\text{class}}; \mathbf{x}_1\mathbf{E}; \mathbf{x}_2\mathbf{E}; \dots; \mathbf{x}_N\mathbf{E}] + \mathbf{E}_{\text{pos}}, \quad (3.18)$$

where $\mathbf{x}_{\text{class}}$ is the classification token, \mathbf{x}_i are the flattened patches, \mathbf{E} is the embedding matrix, and \mathbf{E}_{pos} are the positional encodings.

The adoption of the transformer model in image processing, as instantiated by ViT, brings several advantages over traditional CNNs. ViT’s ability to capture long-range dependencies between different segments of an image allows for a more holistic understanding of the visual content. This capability has shown remarkable results, especially in large-scale image datasets where global contextual information is crucial. The ViT model, as outlined in Figure 3.4, marks a significant shift in how we approach image classification tasks. Its success prompted further research and adaptations of the transformer model in various areas of computer vision, proving its generality and effectiveness beyond natural language processing.

3.6 Summary

Chapter 3 offers a concise yet comprehensive overview of Deep Learning, highlighting its roots in the structure of the human brain and the landmark contributions of Geoffrey Hinton, Yann LeCun, and Yoshua Bengio. This chapter discusses the architecture of deep neural networks, emphasizing their multi-layered composition and the critical roles of backpropagation and optimization techniques like SGD and Adam in their functioning.

A significant focus is placed on CNNs and their revolutionary impact on computer vision tasks. The chapter elaborates on the architectural nuances of CNNs, such as convolutional operations, pooling layers, and key parameters that influence their performance. The latter part of the chapter delves into the Transformer model architecture, detailing its encoder-decoder structure, self-attention mechanism, and positional encodings, which have been instrumental in sequence-to-sequence modeling tasks. Finally, the ViT is introduced as an innovative adaptation of the Transformer model for image classification, marking a shift from traditional CNN approaches. This segment underscores the Transformer’s ability to process images as sequences of patches, enhancing the understanding of global contextual information.

In short, this chapter summarizes the evolution, functionality, and impact of deep learning in AI, particularly in advancing technologies like CNNs and transformers, and sets the stage for its continued influence in machine learning and computer vision.

Chapter 4

Text Detection

Chapter 4 presents an integrated overview of Text and Character Detection advancements in OCR. It discusses the evolution from traditional methods to deep learning models, focusing on the challenges of character localization in diverse scenarios. Key highlights include the Differentiable Binarization Network (DBNet) for complex scene text detection, its advanced iteration DBNet++ with enhanced features for handling varied text sizes and backgrounds, and the Character-Region Awareness For Text detection (CRAFT) model, which specializes in individual character segmentation. This chapter underscores the contributions of these models in advancing OCR technology and their broad applicability across different domains.

4.1 Character Detection

Character detection is an important component of OCR, focuses on identifying and localizing individual characters within digital images. It is vital for applications like document digitization, automated data entry, and license plate recognition [90, 91]. The emergence of advanced machine learning and image processing technologies has significantly elevated the precision of character detection, particularly in complex and diverse settings. Character detection faces numerous challenges, including variations in font styles, sizes, colors, and image quality issues like noise, distortion, and uneven lighting. These factors can profoundly impact the accuracy of character detection systems. Figure 4.1 demonstrates a pipeline that includes both pre-processing and post-processing steps to enhance character detection. Image Binarization is a crucial step that involves converting grayscale images to binary format. Binarization simplifies image analysis by reducing complexity and typically employs several methods, each with unique characteristics as following:

- *Otsu's Binarization* [92]: A global thresholding technique that determines an optimal threshold by minimizing within-class variance. The formula is:

$$\sigma_w^2(T) = \omega_0(T)\sigma_0^2(T) + \omega_1(T)\sigma_1^2(T) \quad (4.1)$$

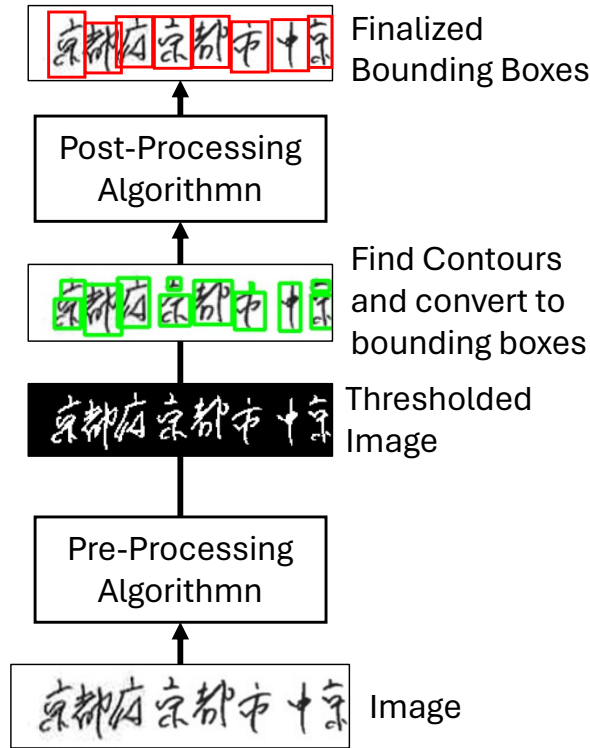


Figure 4.1: Example of a pipeline for character detection using image processing techniques.

where ω_0 and ω_1 are class probabilities separated by threshold T , and σ^2 represents the variances.

- *Niblack's Method [93]*: A local adaptive thresholding technique that calculates the threshold based on local mean and standard deviation. It is defined as:

$$T(x, y) = m(x, y) + k \cdot s(x, y) \quad (4.2)$$

- *Nick's Method [94]*: Uses local statistics to compute the threshold, effective in handling varying contrast levels. The threshold is given by:

$$T(x, y) = m(x, y) - k \cdot \left(\frac{s(x, y)}{R} - 1 \right) \quad (4.3)$$

A common method to detect the characters is to perform contour detection on the image after binarization, which is implemented in OpenCV [95], and convert the detected contour information into a bounding boxes. Also, it might include some post-processing process to determine the final bounding boxes, as in Figure 4.1.

Modern approaches in character detection increasingly use deep learning models, particularly CNNs [28], to extract features and identify characters. These models, trained on

extensive datasets, can recognize a wide array of text styles and representations. Character detection typically encompasses several steps: pre-processing the image, segmenting to isolate characters, and classifying each character. Pre-processing includes binarization, normalization, and noise reduction, while segmentation deals with separating connected characters, a challenge in cursive writing. The overall goal is to enhance text features for accurate detection and recognition.

4.2 Text Detection

Text detection has evolved significantly, transitioning from conventional image processing strategies to the advanced domain of deep learning. Initially, text detection processes leaned on classic image processing methods such as binarization, as discussed in Section 4.1, and morphological operations [96] like erosion and dilation. These operations were crucial for extracting text from images by segmenting the text elements from the background. Despite their simplicity, these traditional techniques were quite effective within controlled environments and remain vital. They frequently act as critical pre-processing steps in contemporary OCR systems, improving image quality for deeper analysis. With the advent of deep learning, the approach to text detection has undergone a profound transformation, as illustrated in Figure 4.2. The introduction of deep learning revolu-

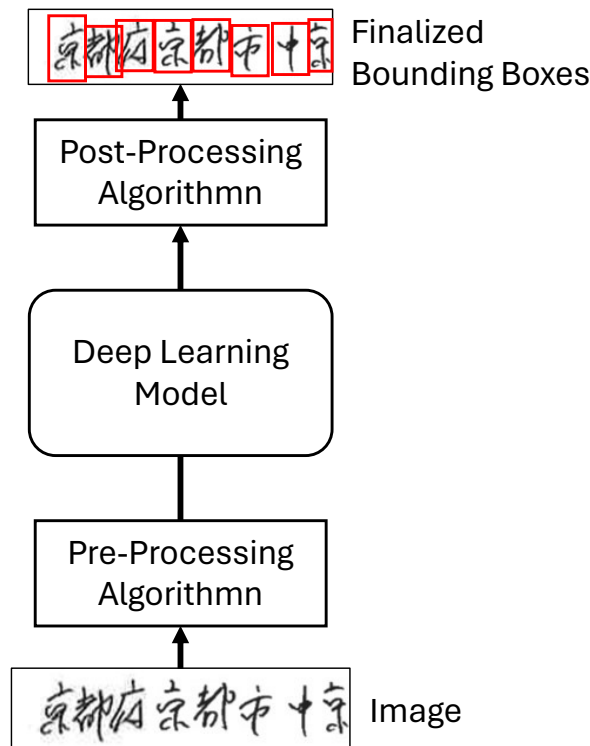


Figure 4.2: Example of a pipeline for character detection using Deep Learning.

tionized the field of text detection and also for character detection. Segmentation-based

approaches, particularly with neural networks like U-Net [97], have shown exceptional performance in complex scenarios where text is integrated with various visual elements. These models operate on a per-pixel basis, effectively differentiating text from non-text regions. This granular approach is particularly beneficial in detecting text in non-standard forms, such as curved or skewed text, which are common in logos or angled photographs. Besides segmentation-based methods, deep learning has streamlined text detection through regression-based approaches. These methods employ CNNs to directly predict bounding boxes around text regions, combining the detection process and confidence assessment in one efficient operation. This technique is especially advantageous in time-sensitive applications like video processing or live feeds, where rapid text detection is crucial.

Hybrid models that blend deep learning with traditional image processing techniques represent an advancement in text detection. In these models, deep learning offers a nuanced understanding of image content, which is then further refined using classical image post-processing methods. For instance, after a neural network segments text regions, operations like erosion and dilation can be applied to enhance the detection results. This includes clarifying the boundaries, reducing noise, and separating connected characters. The hybrid approach brings together the contextual comprehension afforded by deep learning and the precision of traditional methods, yielding improved text detection capabilities, particularly in noisy or low-contrast scenarios.

Overall, the field of text detection has evolved into a diverse and sophisticated discipline that incorporates a wide range of techniques. From foundational image processing to cutting-edge deep learning architectures, each method contributes to the robustness and adaptability of OCR systems. These advancements enable OCR technology to effectively handle a broad spectrum of text types and conditions, opening up new possibilities in document digitization, automated data entry, and beyond.

4.3 Text Detection Using Deep Learning

This section explores advanced deep learning models for character detection, emphasizing their significant contributions to OCR technology. The section begins by detailing the Differentiable Binarization Network (DBNet), a model excelling in text detection within complex natural scenes. It highlights DBNet's key features, including a robust CNN architecture, adaptive thresholding, and a differentiable binarization module for precise text localization. The section then progresses to DBNet++, an enhanced version of DBNet, which incorporates the Adaptive Scale Fusion Module and a Spatial Attention Mechanism. These additions enhance DBNet++'s capabilities, especially in handling text of varying sizes and complex backgrounds. The section also introduces the Character-Region Awareness For Text detection (CRAFT) model, which uniquely focuses on detecting individual character regions. CRAFT's dual scoring approach, consisting of region and affinity scores, enables effective segmentation of characters in densely packed texts.

4.3.1 Differentiable Binarization Network (DBNet)

The Differentiable Binarization Network (DBNet) stands as a significant innovation in the realm of text detection, particularly in processing complex natural scenes. DBNet's architecture, as shown in Figure 4.3, encompasses several critical components, each contributing to its efficacy in detecting text with varying scales, orientations, and levels of distortion.

Key Features of DBNet

The foundation of DBNet's architecture is a robust CNN that serves to extract complex features from input images. A pivotal aspect of DBNet is its differentiable binarization module, which employs an adaptive thresholding mechanism. This mechanism is crucial for generating a probability map of the text regions. The probability map represents the likelihood of each pixel belonging to a text region, providing a granular understanding of text distribution in the image.

Probability Map and Thresholding

The probability map generated by the network is further processed to create an approximate binary map. This binary map is a simplified representation, where the pixels with a high probability of being part of text regions are marked distinctly from the background. The adaptability in the thresholding process enables DBNet to accurately separate text from complex backgrounds, a task that traditional OCR systems often struggle with.

Differentiable Binarization (DB) Module

The DB module of DBNet allows for the fine-tuning of the thresholding process, making it more flexible and capable of handling a diverse range of text presentations. The module's differentiable nature means it can be optimized during the training process, leading to more precise text localization.

Upscaling and Box Formation

Once the approximate binary map is obtained, the next step involves upscaling the features to match the original image's dimensions. This upscaling is necessary to ensure that the spatial resolution of the detected text regions is consistent with the input image, facilitating accurate box formation around each detected text segment. The box formation step is critical for delineating individual text areas, especially useful in scenarios where the text is closely spaced or overlaps with other visual elements.

DBNet's sophisticated approach to text detection, encompassing the generation of probability maps, adaptive thresholding, and precise box formation, makes it an exceptional tool for various applications. Its ability to discern text amidst diverse and challeng-

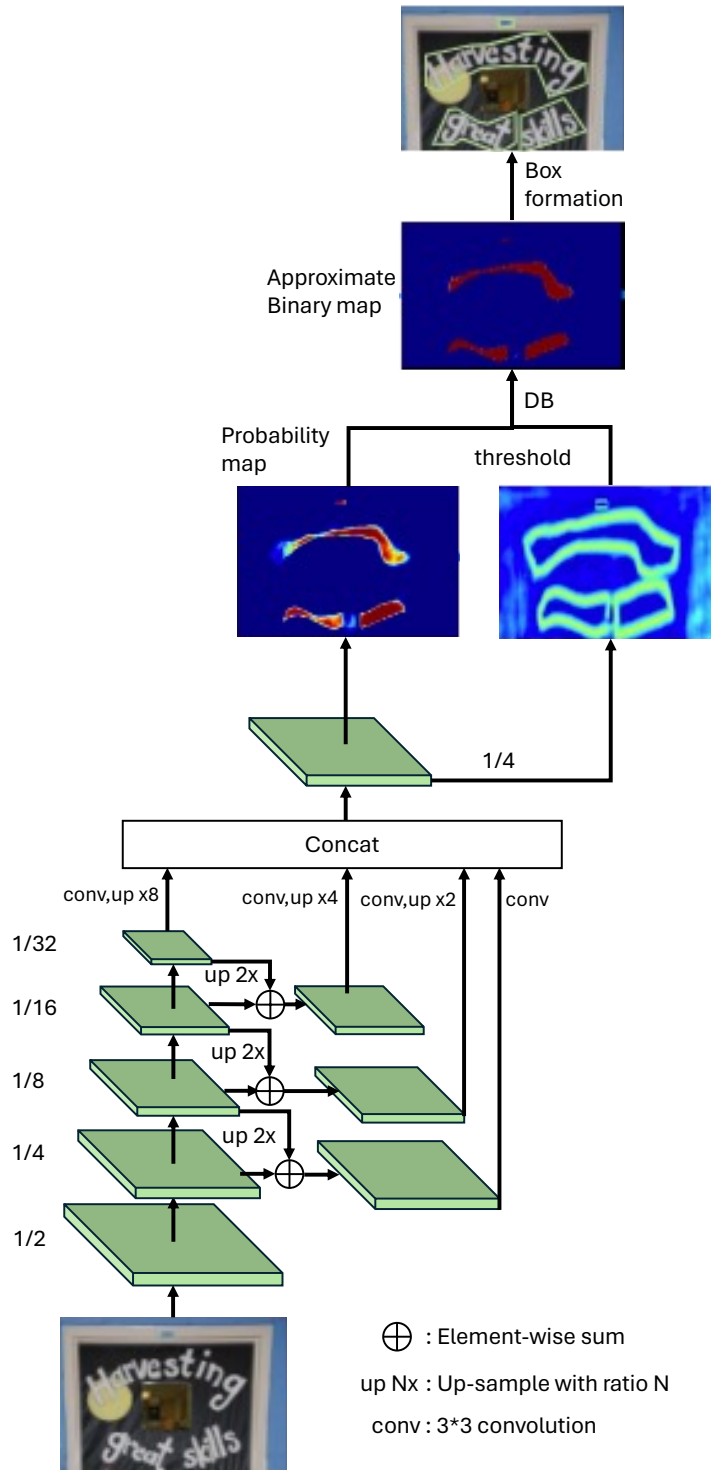


Figure 4.3: Differentiable Binarization Network (DBNet) [1]

ing backdrops has expanded the boundaries of what modern OCR systems can achieve. From document analysis to real-time text recognition in dynamic environments, DBNet's contributions to the field of text detection are profound and far-reaching.

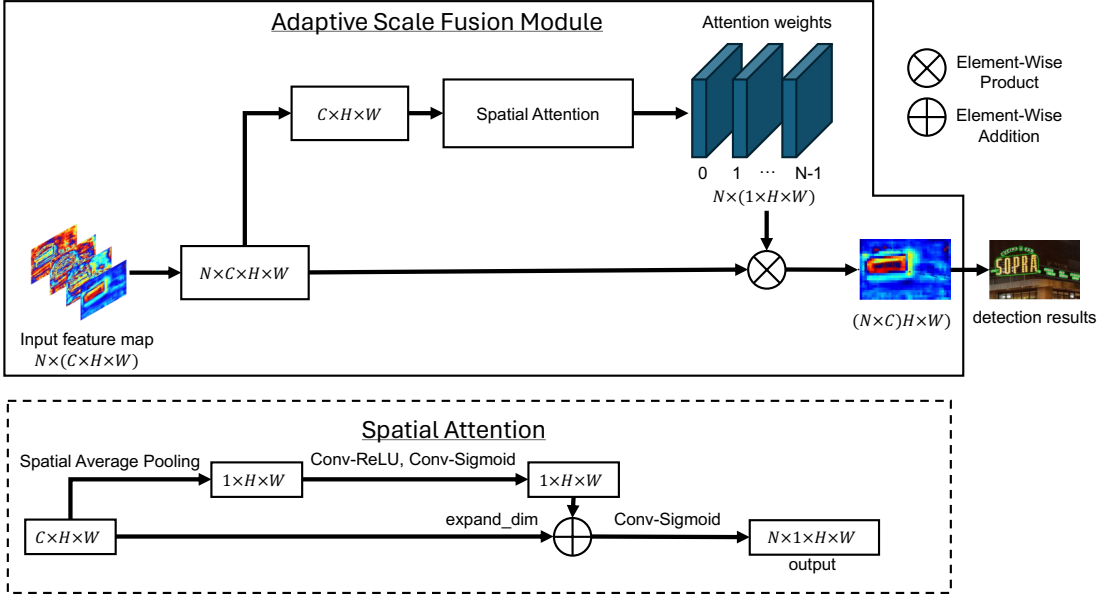


Figure 4.4: Adaptive Fusion Module and Spatial Attention architectures [2]

In summary, DBNet, as illustrated in Figure 4.3, is a groundbreaking model that integrates deep learning with innovative techniques like adaptive thresholding and differentiable binarization. Its proficiency in generating accurate probability maps, creating approximate binary maps, and efficiently performing upscaling and box formation has established it as a benchmark in the domain of advanced text detection.

4.3.2 DBNet++: Advanced Text Detection with Adaptive Scale Fusion Module

DBNet++ is an improved model of DBNet detailed in Section 4.3.1. This model maintains the fundamental principles of DBNet while seamlessly integrating new elements that substantially enhance its text detection capabilities. As a result, DBNet++ stands out for its ability to adeptly navigate the complexities of text detection in various challenging scenarios. The core of DBNet++ lies the Adaptive Scale Fusion Module, a feature prominently illustrated in Figure 4.4. This module is engineered to effectively combine features derived from different scales. Its versatile nature allows the network to accurately detect text, irrespective of its size and shape. From prominently displayed large text to more subtle, smaller text forms, the Adaptive Scale Fusion Module equips DBNet++ with the flexibility to handle a wide array of text characteristics. This adaptability is particularly crucial in enhancing the network’s efficiency in identifying text across varied image contexts.

Another critical advancement in DBNet++ is the refinement of the differentiable binarization process. This enhancement focuses on bolstering the network’s resilience against the varying appearances of text and the complexities of diverse backgrounds. The im-

proved differentiable binarization process in DBNet++ facilitates a finer distinction between text and non-text areas within an image, enabling more accurate and precise text localization. This nuanced approach is key to the enhanced performance of DBNet++ in text detection tasks.

DBNet++ also incorporates a Spatial Attention Mechanism, a feature that further elevates its text detection capabilities. This mechanism is adept to focus on specific areas within an image that are more likely to contain text. By focalizing these regions, the network significantly enhances the quality of text detection. This Spatial Attention Mechanism, working in synergy with the Adaptive Scale Fusion Module, ensures that DBNet++ is not only responsive to text size variations but also acutely aware of its spatial positioning within the image. These features in DBNet++ translates to an improvement in performance metrics. When compared with its predecessor, DBNet++, with its adaptive fusion module and spatial attention architecture, exhibits higher precision and recall rates in text detection tasks. This superior performance is particularly evident in challenging scenarios, such as detecting text presented in unconventional formats or set against highly dynamic backgrounds.

4.3.3 Character-Region Awareness For Text detection (CRAFT)

The Character-Region Awareness For Text detection (CRAFT) model represents a transformative approach in the realm of character detection within images. Distinguished from conventional methods that typically focus on detecting entire lines or words, CRAFT, as shown in Figure 4.5, targets the specific regions occupied by individual characters. This strategy renders CRAFT exceptionally capable in handling text scenarios characterized by close spacing or overlapping characters.

The core of the CRAFT model is the region scoring mechanism, finely designed to ensure areas within an image where characters are most likely to be found can be remarkably identified. This mechanism is adept at highlighting character regions, thereby facilitating the precise localization of each character. In tandem with this, CRAFT employs an affinity scoring system. This system plays a pivotal role in discerning the proximity between adjacent characters, a feature particularly useful in segmenting individual characters within densely populated text areas. The dual scoring approach of CRAFT—encompassing both region and affinity scores enables the model to effectively segment individual characters, even in challenging textual landscapes. This capability is especially beneficial in languages featuring intricate scripts or in documents where lines of text are closely packed. By accurately identifying and segmenting individual character regions, CRAFT significantly enhances the overall accuracy and efficacy of text recognition systems.

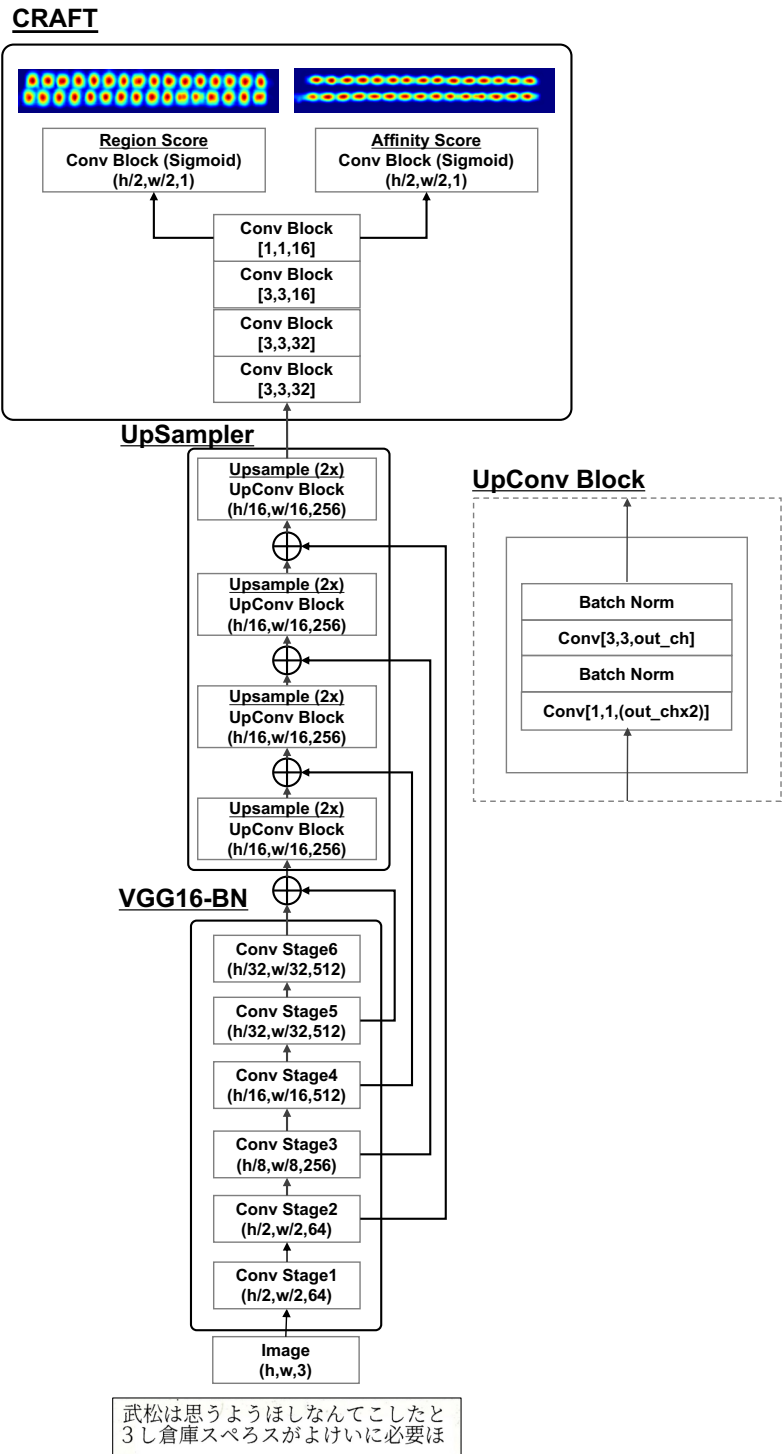


Figure 4.5: CRAFT architectures

4.4 Summary

This chapter explores the progress in technologies for detecting text and characters, highlighting their contribution to OCR systems. The evolution from simple, rule-based

methods to sophisticated deep learning approaches has enhanced OCR's capabilities and widened its use cases. Character detection, which focuses on recognizing individual characters, is vital for detailed text analysis. Models like CRAFT have transformed this area with their dual scoring system for accurate character segmentation, proving especially useful in texts that are closely packed. Meanwhile, text detection is key for understanding the larger context and content of text within images. Techniques such as DBNet and its improved version, DBNet++, stand out here. They utilize strong CNN frameworks, differentiable binarization, Adaptive Scale Fusion Modules, and Spatial Attention Mechanisms, enabling them to efficiently manage text across varied and complex backgrounds. These developments have not only increased the precision and speed of OCR systems but also expanded their practical applications, from converting documents to digital formats to recognizing text in real-time environments. Future integrations with other fields like natural language processing and computer vision promise new, innovative uses, further narrowing the divide between printed text and digital information.

Chapter 5

Image Generation Using Deep Learning

Chapter 5 delves into the realm of deep learning applied to image generation, focusing on the concept of autoencoders and the advanced Y-Autoencoder (Y-AE) architecture. It starts by introducing autoencoders, a type of neural network developed for unsupervised learning, as conceptualized by Geoffrey E. Hinton. This chapter will also explain the changes of autoencoders from image reconstruction usage to vector modifier such as Y-AE that being used to modify the attributes of the compressed vectors output by autoencoder.

5.1 Autoencoders

An autoencoder, introduced by the pioneer of artificial intelligence Geoffrey E. Hinton [98], was a specialized type of neural network used in unsupervised learning. Its fundamental purpose is to learn an efficient encoding or representation of input data. Autoencoders are particularly effective for tasks such as dimensionality reduction, feature learning, and denoising, where the goal is to extract meaningful information from the input while disregarding irrelevant variations or noise. The architecture of an autoencoder is comprised of two primary components: the encoder and the decoder. The encoder's function is to transform the input data into a more compressed and efficient representation, often referred to as the "latent space" or "bottleneck." This process is mathematically represented as:

$$\mathbf{h} = f(\mathbf{W}_e \mathbf{x} + \mathbf{b}_e), \quad (5.1)$$

where \mathbf{x} is the input vector, \mathbf{W}_e represents the encoder weights, \mathbf{b}_e is the encoder bias, and f denotes the activation function. The decoder, on the other hand, aims to reconstruct the original input from the compressed representation. It essentially performs the inverse operation of the encoder:

$$\hat{\mathbf{x}} = g(\mathbf{W}_d \mathbf{h} + \mathbf{b}_d), \quad (5.2)$$

where \mathbf{h} is the encoded vector, \mathbf{W}_d are the decoder weights, \mathbf{b}_d is the decoder bias, and g is the decoder activation function.

The training of an autoencoder involves adjusting the weights and biases ($\mathbf{W}_e, \mathbf{W}_d, \mathbf{b}_e, \mathbf{b}_d$) such that the output $\hat{\mathbf{x}}$ is as close as possible to the input \mathbf{x} . This is achieved through a loss function, typically the mean squared error (MSE) for continuous input data, defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2, \quad (5.3)$$

where n is the number of samples. The training process involves minimizing this loss function, guiding the autoencoder to learn to ignore irrelevant variations (noise) and capture the most salient features of the input data.

Autoencoders find extensive applications in areas such as image processing, anomaly detection, and information retrieval. Variants of the basic autoencoder, like the variational autoencoder (VAE) and the denoising autoencoder, have been developed to handle more complex tasks, including generative modeling and robust feature extraction. In essence, autoencoders serve as a powerful tool in the realm of unsupervised learning, offering an elegant solution to learn compact representations of data, thereby facilitating various downstream tasks in machine learning and artificial intelligence.

5.2 Autoencoder-Based Image Reconstruction

Autoencoder-based image reconstruction, as shown in Figure 5.1 utilizes a neural network with encoder-decoder structure for processing, enhancing or reconstructing images. This architecture effectively captures and reconstructs images from their latent representations, crucial for tasks like image denoising, super-resolution, and restoration of damaged visuals. The encoder in an autoencoder serves to compress the input image, denoted as \mathbf{x} , into

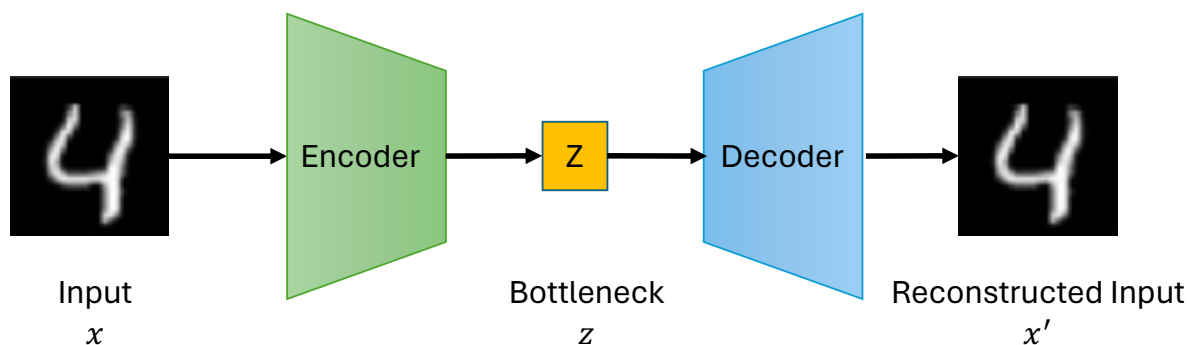


Figure 5.1: Autoencoder based image reconstruction model architecture

a lower-dimensional representation called the latent vector \mathbf{z} . This process is achieved through a series of neural network layers which systematically reduce the dimensionality, extracting and retaining the critical features of \mathbf{x} in \mathbf{z} .

Conversely, the decoder part of the autoencoder focuses on reconstructing the input image from the compressed latent vector \mathbf{z} . It gradually transforms and upscales \mathbf{z} through multiple layers, ultimately producing a reconstructed image \mathbf{x}' . The aim is to make \mathbf{x}' as close as possible to the original \mathbf{x} , effectively restoring or enhancing the initial image. Training an autoencoder involves fine-tuning its weights to minimize the difference between the original image \mathbf{x} and its reconstruction \mathbf{x}' . This is quantified using loss functions such as Mean Squared Error (MSE), which measures the pixel-wise discrepancies between \mathbf{x} and \mathbf{x}' .

Autoencoders can be adapted to various forms of image reconstruction. In image denoising, for instance, they learn to identify and remove random noise from the images. For super-resolution tasks, they are trained to upscale low-resolution images while retaining or reconstructing details. They can also be tailored for more specialized tasks, like repairing damaged or incomplete images, as demonstrated in Figure 5.1. Finally, autoencoder-based image reconstruction, represented in Figure 5.1, offers an effective and versatile method for various image processing applications. By efficiently capturing and reconstructing images through the interplay of encoding and decoding processes, they significantly enhance our capabilities in image restoration and enhancement.

5.3 Y-Autoencoder

5.3.1 Model Architecture

The Y-Autoencoder (Y-AE), as shown in Figure 5.2, features a unique two-branch architecture designed to optimize explicit and implicit losses. The encoder in the Y-AE aims to extract style features from the input image, whereas the decoder focuses on either reconstructing or generating images. This dual-branch approach ensures that Y-AE performs well in a variety of tasks, such as altering styles or reconstructing images. For example, as shown in Figure 5.2, the model can alter a female face image to a male face image by modifying the attributes fed into the decoder on the right branch.

5.3.2 Loss Functions

Y-AE's loss function is a combination of four key components, each targeting different aspects of the encoding and reconstruction phases.

$$L_r = \|\hat{x}_L - x\|^2 \tag{5.4}$$

The first component, as defined in Equation 5.4, is the reconstruction loss L_r used in the left branch. Here, the label y replaces the explicit component e inferred by the encoder. This substitution is crucial in the early stages of training to mitigate instability due to inaccurate classifier predictions. The loss function L_r ensures fidelity in the reconstruction

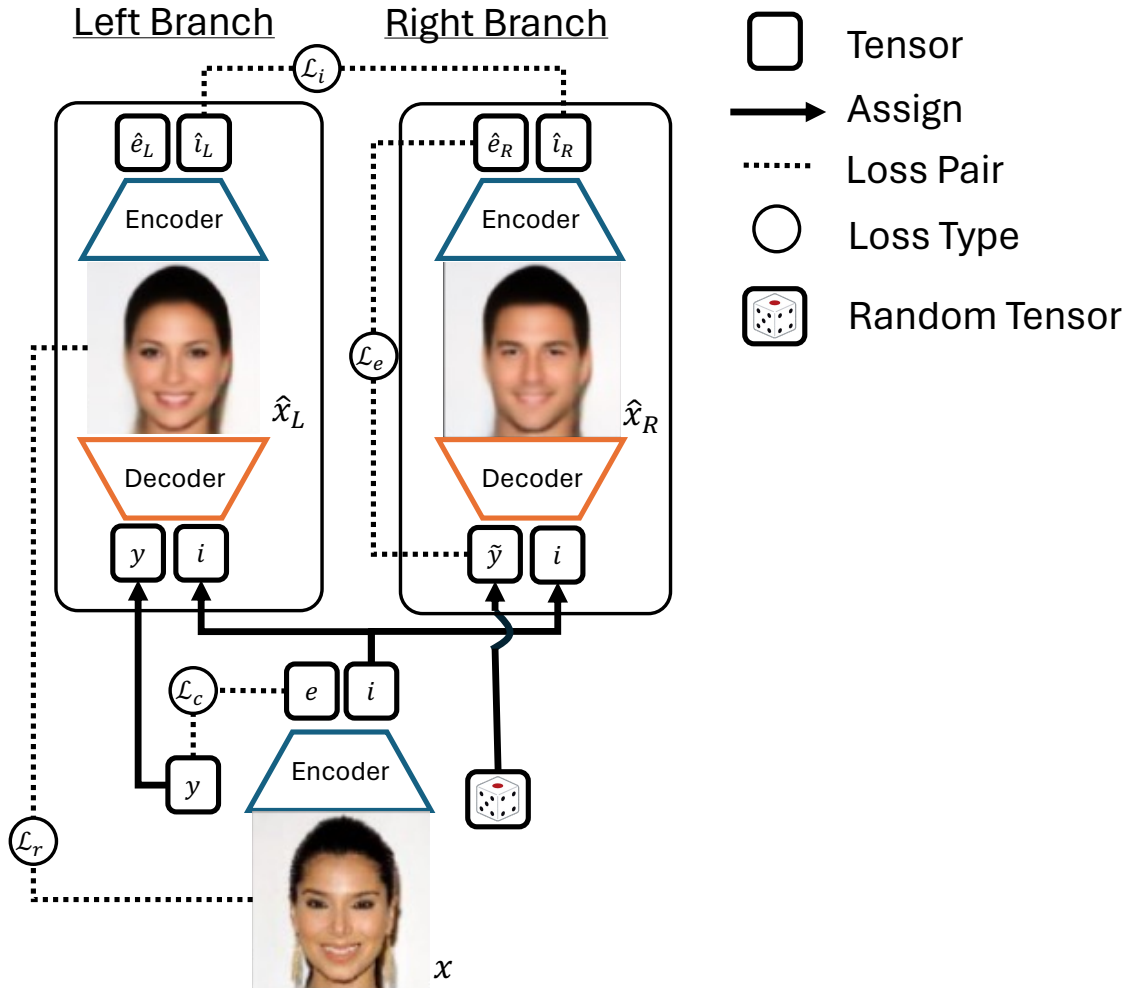


Figure 5.2: Y-Autoencoder Architecture [3]

by penalizing deviations between the reconstructed image \hat{x}_L and the original input x using a standard least-squared error approach.

$$L_c = CE(e, y) = - \sum_j e_j \log y_j \quad (5.5)$$

The second component, detailed in Equation 5.5, is the cross-entropy loss L_c . This loss is applied between the explicit component e and the label y . It serves a dual purpose: firstly, to guide the explicit part of the encoder's output to effectively predict the explicit content type in the input x ; and secondly, to reinforce the predictor aspect of the Y-AE's right branch.

$$L_e = CE(\hat{e}_R, \tilde{y}) = - \sum_j \hat{e}_{R,j} \log \tilde{y}_j \quad (5.6)$$

The third element, as per Equation 5.6, is another cross-entropy loss L_e applied in the right branch to ensure the consistency of the relation between \hat{e}_R and \tilde{y} . This consistency

is vital for verifying that the encoded explicit content aligns closely with the intended attributes.

$$L_i = \|\hat{i}_L - \hat{i}_R\|^2 \quad (5.7)$$

Finally, Equation 5.7 introduces the implicit loss L_i . This loss is applied to the left branch and is integral for maintaining consistency in the implicit information across both branches. Since this information remains unaltered, the implicit vectors \hat{i}_L and \hat{i}_R from both branches should exhibit high similarity, and any deviations are penalized.

$$L = L_r + L_c + \lambda_e L_e + \lambda_i L_i \quad (5.8)$$

The final loss function, as shown in Equation 5.8, integrates these components into a cohesive whole. The weighting factors λ_e and λ_i allow for fine-tuning the influence of explicit and implicit losses, respectively. This nuanced approach, where each loss component plays a specialized role, ensures the Y-AE goes beyond accurate reconstruction to capture the subtleties, explicit and implicit features of the input data. An in-depth analysis of the effects of varying λ_e and λ_i is presented in the experimental section of the paper [3], underscoring the flexibility and adaptability of this loss function.

The Y-AE distinguishes itself with a dual-branch structure that processes implicit and explicit information separately, granting detailed control over the characteristics of the images it generates. This allows the Y-AE to rival more complex models like VAEs and GANs, maintaining simplicity in its training process. Its application range is broad, from separating styles and contents in images, converting images from one form to another. Looking ahead, the thesis combining Y-AEs with AdaIN, to push forward the development of conditional autoencoders for character image generation which will discuss in chapter 6.

5.4 Summary

This chapter provides a thorough understanding of autoencoders, emphasizes their role in image reconstruction, and introduces the Y-Autoencoder, which brings additional control and precision to image generation tasks, leading to the proposed method for character image generation with Y-AE, which will be explained in chapter 6. Autoencoders, as introduced by Geoffrey E. Hinton, was a neural networks designed for unsupervised learning. Their primary function is to learn efficient data encodings, useful in dimensionality reduction, feature learning, and denoising. It consists of two main parts: the encoder, which compresses input data into a latent space representation, and the decoder, which reconstructs the input from this compressed form. Training a autoencoder involves minimizing a loss function, typically the mean squared error, to ensure the output closely matches the input, capturing essential data features while ignoring noise. Here, the encoder compresses an input image into a latent vector, while the decoder reconstructs the image from

this vector. This technique is vital for image denoising, super-resolution, and repairing damaged images. An image reconstruction autoencoder focuses on minimizing the difference between the original and reconstructed images. Furthermore, the introduction of Y-AE has a distinct architecture that divides the encoder’s output into two branches for encoding implicit and explicit information. This design enables fine control over the generated images’ attributes and allows Y-AE to compete with more complex models like VAEs and GANs, maintaining ease of training.

Chapter 6

Character Generation with Y-Autoencoder

This chapter describes the image generation techniques used in this thesis, how they were changed and applied by the proposed method and how the proposed method was evaluated accordingly. In this thesis, the AdaIN layer is adapted to the Y-Autoencoder’s decoder. The Encoder features are learned in two Decoder branches, and distinct labels are assigned with AdaIN layers which allow the decoder to generate labeled images dependent on the style of the input images. This allows for the generation of 1-to-N character images without being limited to a specific label and a specific input image. This contributes to improving the performance of the character classifier by generating a large number of character images with a small amount of training data, compared to conventional methods such as GAN, by stably generating images of labels that are desired to be generated by the input style images. In addition, by filtering the generated images, it is possible to further limit the number of generated images and effectively improve the performance of the character classifier. This chapter describes AdaIN based on Y-AE, including the model structure, filtering methods, character classifiers for verification, and evaluation experiments.

6.1 Adaptive Instance Normalization (AdaIN)

Adaptive Instance Normalization (AdaIN) [99] is a technique used to transfer style in neural networks. It aligns the mean and variance of the content to the style features, thereby enabling style transfer from one image to another. The AdaIN layer is mathematically expressed as:

$$AdaIN(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \quad (6.1)$$

where x is the content input, y is the style input, $\mu(\cdot)$ and $\sigma(\cdot)$ denote the mean and standard deviation, respectively. In the case of use in Neural networks, convolutional

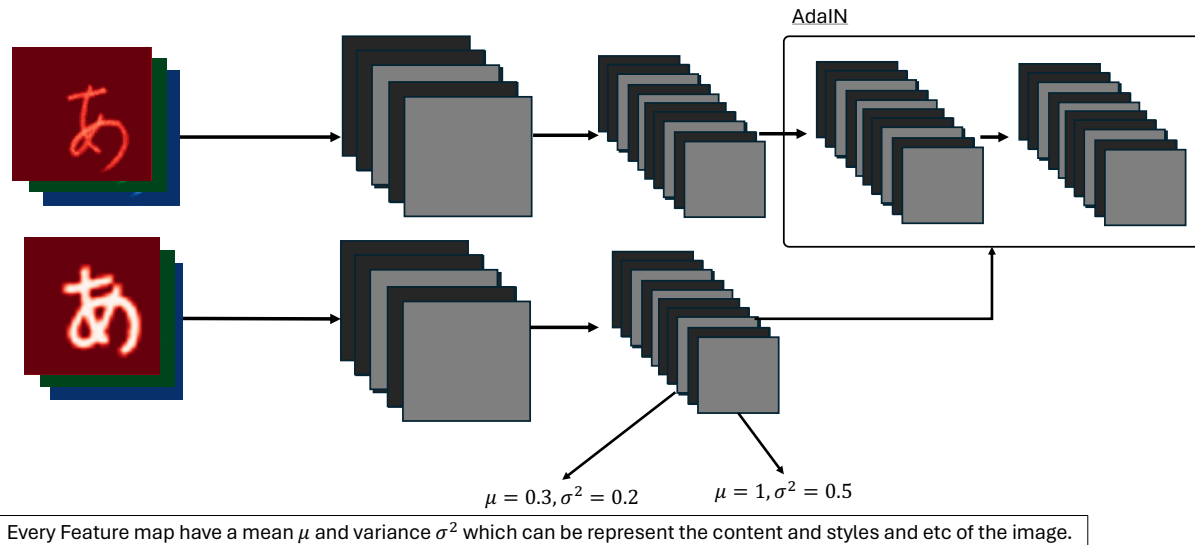


Figure 6.1: Feature maps with AdaIN

layer extract feature maps which each of them have a different mean and variance that shown in Figure 6.1 can be used in equation 6.1. AdaIN’s simplicity and effectiveness make it a popular choice in style transfer applications.

6.2 Image Generation Model with AdaIN

Adaptive Instance Normalization (AdaIN) has transformed the field of image generation, as highlighted in a key study [99]. It has emerged as a fundamental approach in neural style transfer, facilitating models that adeptly transfer styles between images. This process merges the content of one image with the style of another. The effectiveness of AdaIN is based on its mathematical approach, which adjusts the mean and variance of content features to match those of the style features, enabling this seamless style integration. The integration of AdaIN into image generation models like CycleGAN [100] and StyleGAN [101] has marked an advancement in style transfer. CycleGAN [100], for example, employs AdaIN in its architecture for unsupervised image-to-image translation tasks of converting CT image from high-dose to low-dose. This usage allows CycleGAN to effectively learn and transfer styles between two distinct and unpaired image domains, a capability particularly useful in domain transfer models. In addition, StyleGAN [101] also utilizes AdaIN layer, but at each layer of its generator network. This multi-layer implementation of AdaIN in StyleGAN allows for a nuanced control of style at various scales throughout the image generation process. By manipulating style at different levels, StyleGAN can generate high-resolution and diverse character images, each exhibiting intricate and varied styles such as mixing face images.

AdaIN’s role in the this thesis’s Y-Autoencoder enhanced for character generation.

The Y-Autoencoder architecture, as detailed in Figure 5.2 designed focus on the task of character generation. In this model, AdaIN adjusts the style of the generated characters by aligning their feature statistics with those of the desired style. This alignment is achieved through the AdaIN equation, which effectively modifies the content features from the encoder with the style features, resulting in a stylized output. Such a capability allows for the generation of characters in various handwriting styles, significantly contributing to the diversity and effectiveness of OCR training datasets.

The impact of AdaIN in these generative models is a testament to its versatility and efficacy in creating diverse and realistic character images. As discussed in original paper [99], AdaIN’s simplicity and efficiency in performing real-time style transfer make it a highly favorable choice in applications requiring style manipulation. Its integration into character generation models for OCR systems demonstrates its potential in advancing the field of computer vision and machine learning, particularly in areas where data variety and richness are crucial for model performance. The application of AdaIN in such models not only enhances their ability to generate diverse character styles but also paves the way for the development of more robust and efficient OCR systems, capable of recognizing a wide array of text styles and formats.

The incorporation of AdaIN into image generation models represents a significant stride in the field of style transfer and character generation. The models like CycleGAN and StyleGAN, enhanced with AdaIN, exemplify the transformative impact of this technique in generating diverse character images for OCR systems. The ongoing developments in this area, as exemplified by the findings in this thesis, continue to push the boundaries of what is capable in style transfer, character generation. The process of generating character images utilizing Y-Autoencoder with AdaIN will be detailed in the next section.

6.3 Y-Autoencoder with AdaIN

6.3.1 Model architecture

The architecture of the Y-AE model used in this thesis to generate Hiragana and Katakana character images is shown in Figure 6.2. The model is based on the original Y-AE architecture [3], which consists of an encoder and a conditional decoder. The encoder uses a VGG16 [64] backbone feature extractor to encode the RGB image in the shape of $(128, 128, 3)$ and output the style representation i and character label e of the input image. Both the style expression i and character label e are the inputs to the decoder, which generates a handwritten character image. The character label e is converted into a 512-dimensional embedding vector by the embedding layer. The embedding vector is input to three fully-concatenated (FC) layers, from which the content features shown in Equation (6.2) are extracted. This embedding vector allows for the intended character images.

$$content_feature(s) = FC(Emb(s)) \tag{6.2}$$

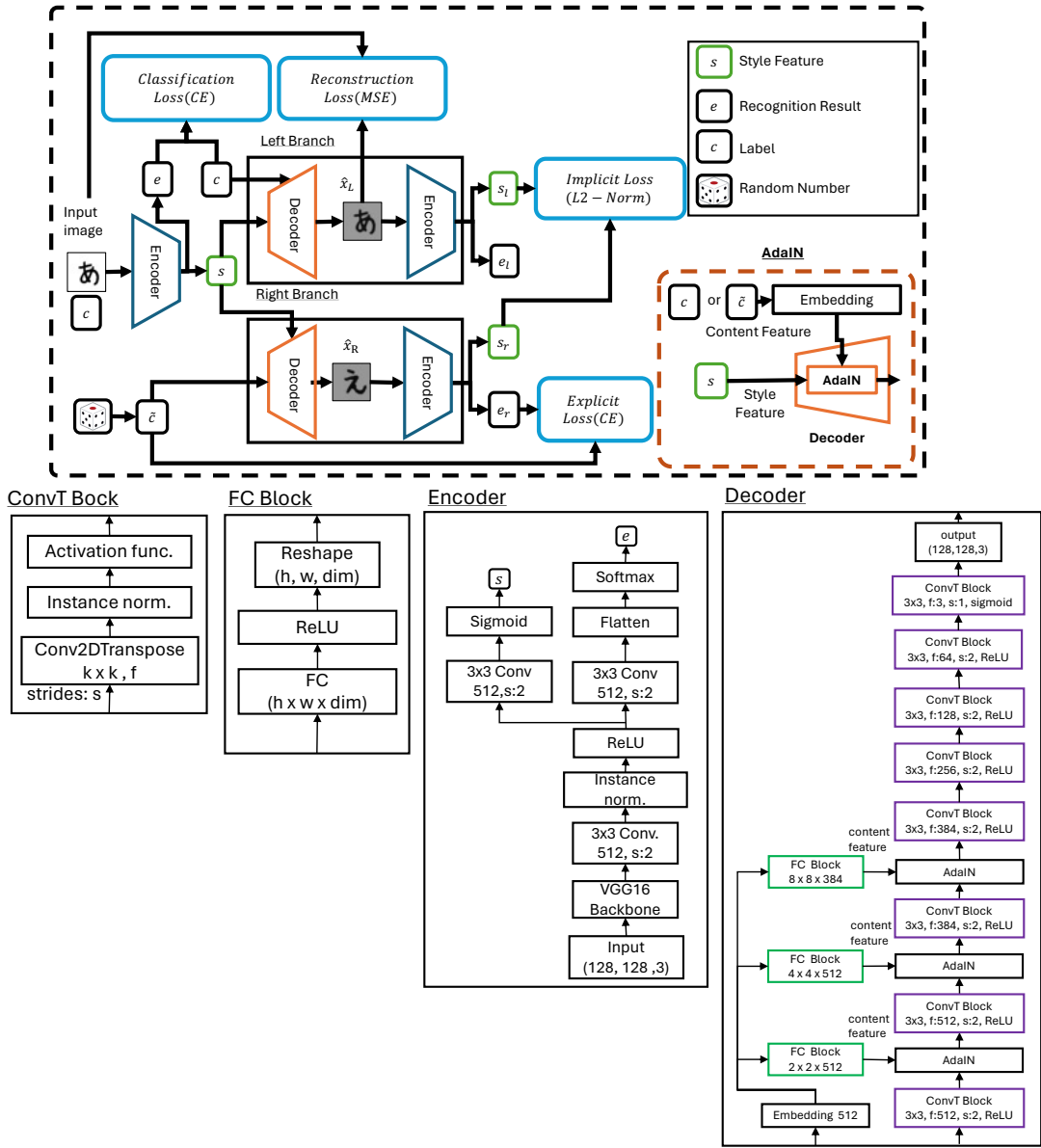


Figure 6.2: The Y-autoencoder architecture.

where FC is an FC layer, Emb is the embedding layer, and s is the character label \tilde{y} or label y . The output dimension of each FC layer must fit the shape of the intermediate up-sampling feature. Therefore, the output vector of each FC layer is reshaped to the exact shape of the up-sampling feature.

To generate the intended character images, a character label is converted into the content features through the FC layers. These content features are injected into the convolution blocks of the decoder using AdaIN [99], as denoted by Equation (6.1).

$$x = content_feature(s) \quad (6.3)$$

$$y = ConvT(i) \quad (6.4)$$

where $ConvT$ in Equation (6.4) is a transposed convolutional layer, and i is the style representation encoded from the input character image. The kernel size used in each convolutional and up-sampling layer is set to 3×3 with strides 2×2 and apply AdaIN with ReLU activation. However, the last convolution layer outputs a style representation i in the encoder and the last ConvT block of the decoder outputs a generated image \hat{x} are also applied a Sigmoid activation function. The last convolutional layer of the encoder outputting estimated character label \tilde{e} uses a softmax activation function.

6.3.2 Loss functions

The loss functions used in this thesis are based on the original Y-AE [3] loss functions, which consists of four separate components. First is the classification loss, which computes the cross-entropy (CE) between the output e with label y using Equation (6.5). The MSE shown in Equation (6.6), as follows, is utilized for the reconstruction loss of the generated image.

$$\mathcal{L}_{cls} = CE(e, y) \tag{6.5}$$

$$\mathcal{L}_{reconst} = \|x_L - x\|^2 \tag{6.6}$$

where e is the output of the encoder, y is the label of the character image, \hat{x}_L is a decoded image decoded in the left-side branch network, and x is the input image. Next, we can calculate the implicit loss, that is the L2-norm with the following Equation (6.7):

$$\mathcal{L}_{im} = \|i_r - i_l + \epsilon\|_2 \tag{6.7}$$

where the i_r and i_l are the style outputs from the left branch and right branch networks, respectively. ϵ is set to $1.0e - 15$. Thirdly, there is the explicit loss, the CE as shown in following Equation (6.8):

$$\mathcal{L}_{ex} = CE(e_r, \tilde{y}) \tag{6.8}$$

where the e_r is the output e of the right branch network, and \tilde{y} is the random character label. Finally, the total loss is shown in Equation (6.9), which is used to back-propagate the gradients to the Y-AE model.

$$\mathcal{L}_{total} = \mathcal{L}_{reconst} + \mathcal{L}_{cls} + \mathcal{L}_{im} + \mathcal{L}_{ex} \tag{6.9}$$

6.4 Filtering of generated images

The character images generated by the Y-AE models do not always represent the correct character form. For example, Figure 6.3 shows some results of the generated character “あ.” As shown in Figure 6.3, some images may not be generated with the correct character. If these images are used to augment the training data of a character classifier,

the presence of noisy images may prevent the formation of a highly accurate character classifier. Therefore, a filtering method for the generated images is introduced. Two

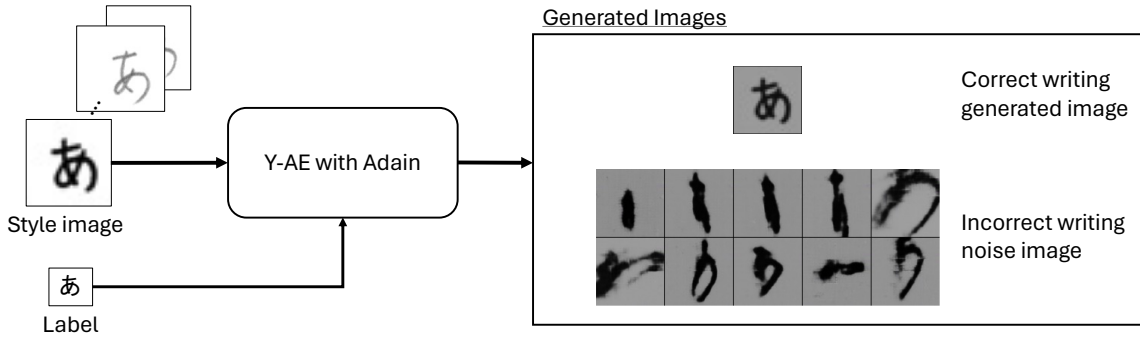


Figure 6.3: Example of generated images of Hiragana character “あ.”

filtering techniques—an MSE-based approach and a character classifier-based approach are investigated in this thesis. The MSE-based approach employs a generated image whose MSE scale to the real images is large. In the character classifier-based approach, a character classifier trained with the original character images (i.e., the baseline classifier) is used to recognize character images, and only correctly recognized character images are adopted for data augmentation.

6.4.1 MSE-based filtering

In the MSE-based filtering approach, the distance between two images is calculated using the following equation:

$$\text{MSE_filter}(A, B) = \frac{1}{w \times h} \sum_{x=0}^w \sum_{y=0}^h \{A(x, y) - B(x, y)\}^2 \quad (6.10)$$

where w and h are the width and height of an image, respectively and $A(x, y)$ or $B(x, y)$ is the pixel value of the (x, y) coordinate in images A and B , respectively. The MSE value is 0 for images in which A and B are exactly the same, and this increases for images in which A and B are different. In other words, the generated images with larger MSE values can be considered more suitable for data augmentation.

By calculating the MSE between the generated images and all the real images of the same character type, the generated images with the a high average MSE value are adopted as the image for data augmentation. Note that when calculating the MSE, a pre-processing as shown in Figure 6.4 is performed to eliminate factors due to the background of the generated images and the size of the characters. As shown in Figure 6.4, the pre-processing was performed by the following steps:

1. A character image is converted to a binary image using Otsu’s binarization [92].

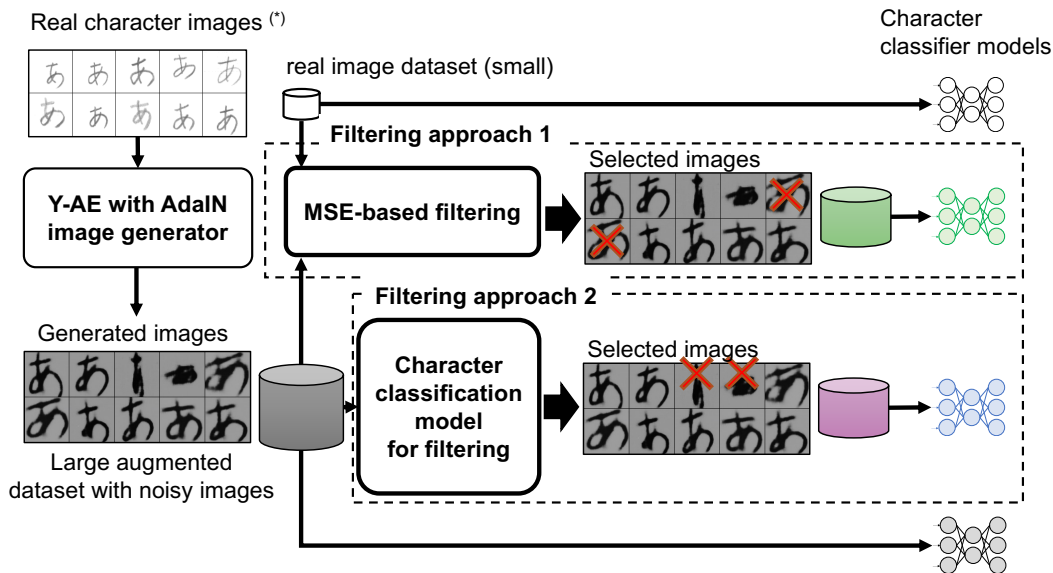


Figure 6.4: Pre-processing of handwritten character images for calculating the MSE.

2. Extraction of the character box in the image.
3. Cropping the character border area.
4. Reshape the image into a (128, 128) square so that margins of at least 10 pixels are added to the top, bottom, left, and right sides of the image.

6.4.2 Classifier-based filtering

A generated image is input to the character classifier. The generated image is not noisy and is considered to have retained the style of the characters if the classifier can correctly classify the image into the proper class. The character classifier used for filtering is trained on the same dataset used to train the Y-AE models. The architecture of the character classifier is described in Section 6.5. A generated image is input to the character classifier, and if the classification result is correct with a posterior probability of 90% or higher at the time, the image is adopted as the image for data augmentation. Note that in training the character classifier using the generated images, only the top n images with the highest posterior probability are used for data augmentation in order to keep the number of images per character class the same. n is explained in Section 6.6.1.

6.5 Handwritten Character Classifier

This section focuses on examines the ability of computer-generated images for handwriting characters recognition. This is tested using a simple version of the ResNet-152 model [62], a popular tool for image recognition.

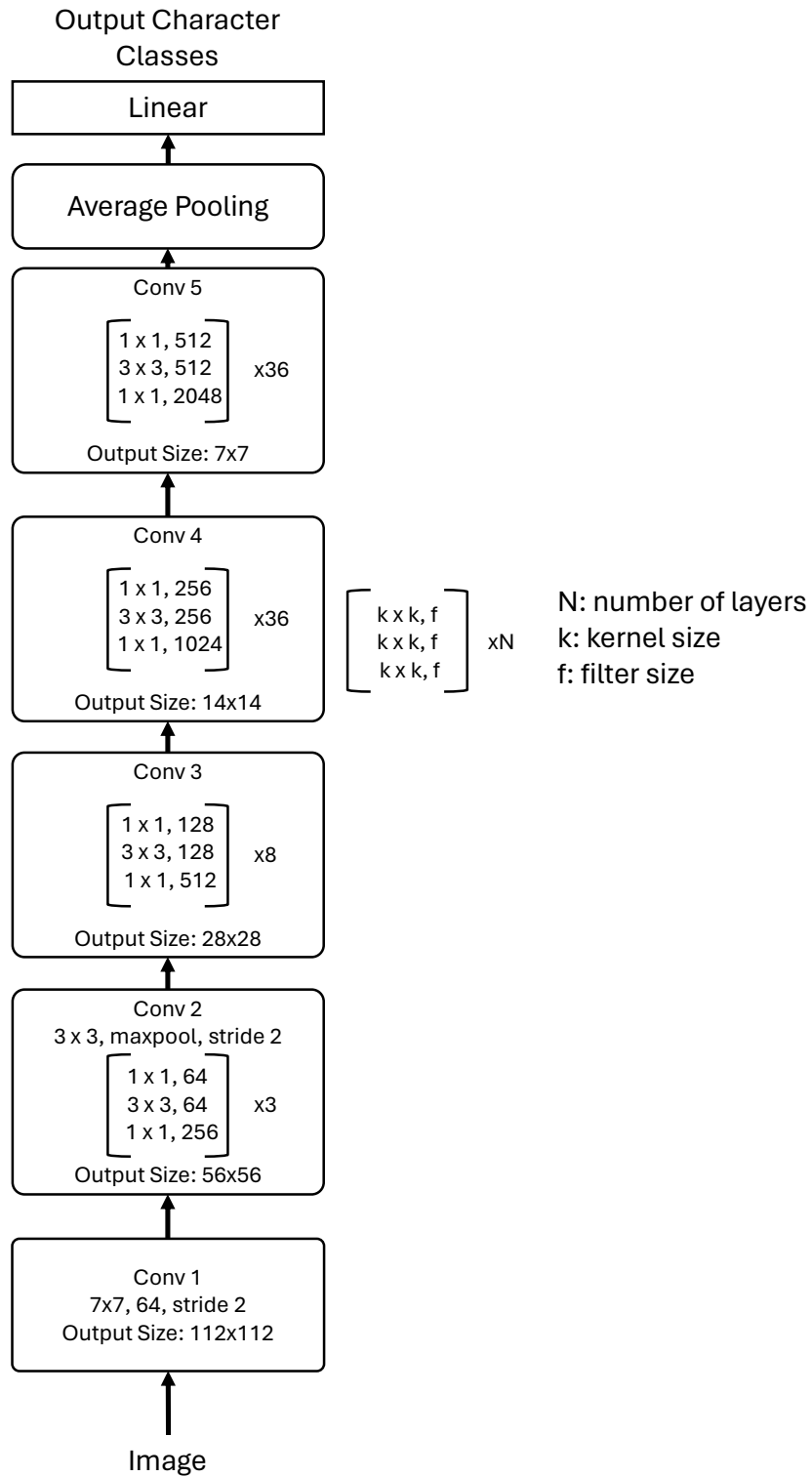


Figure 6.5: ResNet-152 Model for Handwritten Japanese Characters

The main goal is to correctly identify 92 different Japanese characters, which include both Hiragana and Katakana scripts. These handwritten characters images are each sized at 128×128 pixels, are fed into the ResNet-152 model. The model then tries to identify these images, categorizing each one into one of the 92 character types. As shown in Figure 6.5, our version of the ResNet-152 model has been slightly adjusted to specifically work with these Japanese characters. The main changes are in the parts where images are put into the model (input layer) and where the model decides what character it sees (output layer). Other than that, the model works pretty much like the standard ResNet-152.

An important note about the proposed approach is that this model is trained without using any previously learned information (no pre-training). This means that any improvement in recognizing characters comes purely from learning with the images provided, including computer-generated images. This approach reveals whether the generated images contribute to model training for better handwriting character recognition.

6.6 Experiments and Discussion

6.6.1 Experimental setup

Y-AE model training

(1) Dataset

In the current study, instead of training a single Y-AE model that can generate Hiragana and Katakana characters simultaneously, two Y-AE models are involved, one of which is responsible for generating the Hiragana character image, while the other is responsible for generating the Katakana character image. This is because there are some characters in Japanese with similar shapes in Hiragana and Katakana (e.g., “ \wedge ” and “ \wedge ”), and these characters may not be generated well if Hiragana and Katakana images are trained simultaneously using a single Y-AE model.

Table 6.1 shows the dataset used in this thesis. ETL has a total of nine subsets. 46 Hiragana characters from ETL-9 and 46 Katakana characters from ETL-5 are utilized. For training the Y-AE model for generating Hiragana images, 200 handwritten character images are used for each Hiragana, for a total of 9,200 images. To train the Y-AE model for generating Katakana images, 208 handwritten characters are used for each Katakana, here for a total of 9,568 images. When generating character images using the trained Y-AE models for Hiragana and Katakana characters generation, the same character images as used for model training are also used. In other words, the maximum number of character images generated is 423,200 ($=9,200 \times 46$) for Hiragana and 440,128 ($=9,568 \times 46$) for Katakana.

Table 6.1: Dataset for the Y-AE model training and the number of generating handwritten character images.

Target char. type	ETL subset	# of images	# of generated images
Hiragana	ETL-9	9,200 (46×200)	423,200
Katakana	ETL-5	9,568 (46×208)	440,128

(2) Pre-processing of character images

In training Y-AE, it is known that if ETL image data are used without adjustment, the training is not optimal because the sizes of the character images are different [32]. Therefore, Y-AE model can be easily trained by pre-processing the original image to standardize the size of the character regions. This pre-processing also has the advantage of making it easier to evaluate the diversity of the generated character images. The pre-processing is the same approach as the method used for MSE-based filtering shown in Figure 6.4. This pre-processing removes the background of the original character image and places the text in the center of the image.

(3) Y-AE model training

The same hyper-parameters were used for the Katakana-generated Y-AE model and the Hiragana-generated Y-AE model. For the training conditions of Y-AE, the number of epochs was 500, the mini-batch size was eight, Adam was used as the optimization function, and the learning rate was set to 1e-4. For data augmentation, three functions of *ElasticTransform*, *Affine*, and *GaussianBlur* are employed from a tool for image data augmentation called “Albumentations” [10]. Each of these three functions was applied with a probability of 50% during the generation of a mini-batch at the time of model training.

Character classifier training

First, in this thesis, character classification models is trained for 92 Japanese Hiragana and Katakana characters using multiple datasets for comparison. In addition, exactly the same data augmentation functions as used to train the Y-AE models were applied during training. Figure 6.6 shows a list of the training conditions for the character classification models. In addition, six different datasets were used in this thesis. The number of images used in the training of each model is summarized in Table 6.2.

The generated images used in the training of Models (7), (8), (9), and (10) were filtered using the MSE scale and the baseline character classifier. Note that in this case, the filtering was performed so that there would be 263 images per character class ($n = 263$). The reason for limiting the number to 263 is that the character class with the lowest number of images was 263 when the baseline character classifier was used for filtering.







- (1) only ETL (original) images w/o data augmentation (DA) (**baseline**)
- (2) only ETL images w/ general DA

 ETL original images
- (3) only all images generated by the Y-AE generators w/o DA
- (4) only all images generated by the Y-AE generators w/ general DA

 Generated images
- (5) ETL images and all images generated by the Y-AE generators w/o DA
- (6) ETL images and all images generated by the Y-AE generators w/ general DA

- (7) ETL Images and selected images generated by the Y-AE generators using the MSE scale w/o DA
- (8) ETL Images and selected images generated by the Y-AE generators using the MSE scale w/ DA

- (10) ETL Images and selected images generated by the Y-AE generators using the classifier (baseline model) w/o DA
- (11) ETL Images and selected images generated by the Y-AE generators using the classifier (baseline model) w/ DA

- (11) ETL Images + the other ETL subset images (all real images) w/o DA
- (12) ETL Images + the other ETL subset images (all real images) w/ DA


Figure 6.6: List of training conditions for character classification models.

The number of images per character was exactly the same, and there was no difference in the number of images per class. For further comparison, the models ((Models (11) and (12)) were trained with approximately the same number of real handwritten character images as the generated images to demonstrate the usefulness of the generated images.

All character classifiers were subjected to the same training conditions except for the training data and the number of epochs. The mini-batch size was set to 8, Adam was used as the optimization function, and the learning rate was set to $1e-4$. The model structure was ResNet-152, and the number of classes was 92, consisting of 46 Hiragana and 46 Katakana. The ETL and generated images input to the model were binary images of a 128×128 image size, applying Otsu's binarization method to eliminate factors other than character shape.

The validation and test sets for testing the classifier models consisted of real character images of Hiragana and Katakana characters included in ETL-1 and ETL-7. Each character was evaluated with 200 images, totaling 18,400 images. The classification accuracy is used as the evaluation measure. Since there is no difference in the number of ground truth images for each class, the classifiers were evaluated based on the classification accuracy.

6.6.2 Character generation results

Figure 6.7 shows the handwritten character images generated by the Y-AE generators with AdaIN for Japanese Hiragana and Katakana. As shown in top of Figure 6.7, both Hiragana and Katakana handwritten characters were generated as if they were real. The original Y-AE model did not use AdaIN; the handwritten character images generated by the Y-AE generator without AdaIN are shown in bottom of Figure 6.7. As can be seen by comparing top and the bottom of Figure 6.7, the use of AdaIN clearly enabled the generation of a wide variety of handwritten characters.

Table 6.2: Number of character images used in training for each model.

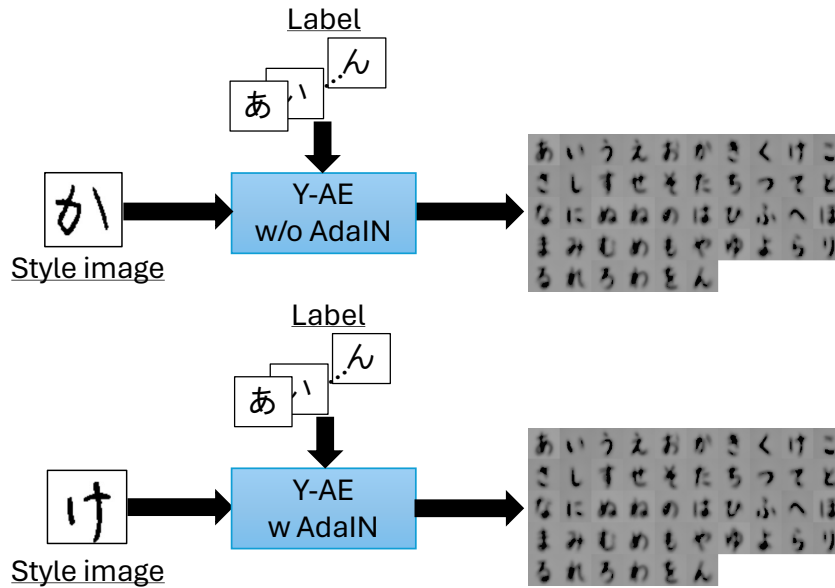
Model no.	# of original ETL images	# of generated images
(1), (2)	18,768 (ETL-5, ETL-9)	—
(3), (4)	—	863,328
(5), (6)	18,768 (ETL-5, ETL-9)	863,328
(7), (8)	18,768 (ETL-5, ETL-9)	24,196
(9), (10)	18,768 (ETL-5, ETL-9)	24,196
(11), (12)	18,768 (ETL-5, ETL-9) + 25,760 (ETL-4, ETL-6, ETL-8)	—

Additionally, Figure 6.8 showcases the Y-AE with AdaIN’s capability to generate Japanese Hiragana and Katakana characters, leveraging different input images. Figure 6.8 distinctly demonstrates that the Y-AE with AdaIN excels not only in creating diverse characters but also in mimicking the style of the input image. For instance, characters such as あ, ア, and う are generated with a larger form, while い and り exhibit a thinner structure in their midsections, and ん is slanted, indicating a keen ability of the Y-AE with AdaIN to capture and replicate the distinct stylistic nuances of the style image. This adaptability is particularly beneficial for enhancing the performance of character classifiers, as it allows for the generation of a wide array of character image variations, encompassing different styles and characters.

Next, the MSE scale was also used to evaluate how much the generated handwritten character images differed from the real images used to train the Y-AE models. The MSE was the same as the calculation method used in the image filtering described in Section 6.4.1. Table 6.3 shows the statistics of the MSE scale. For images of the same character type, the smaller the MSE value, the more the characters can be considered to be of the same handwriting style. Conversely, the larger the MSE value, the more likely it is that the characters had a completely different handwriting style. In Table 6.3, the MSE values between images of the same character type (200 images for each character) were calculated on an all-possible combinations of all the images, and the mean, variance, and minimum MSE values are shown.

As shown in Table 6.3, the MSE values between the real images in ETL had a larger

Y-AE without AdaIN generated images



Y-AE with AdaIN generated images

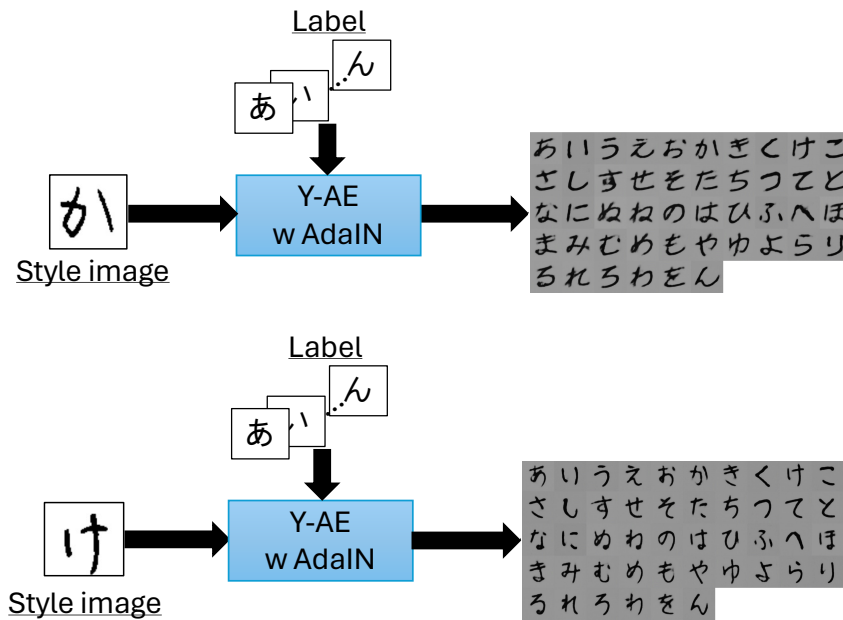


Figure 6.7: Example of handwritten character images generated by the Y-AE with and without AdaIN.

mean and variance, indicating that there was more variation in the handwriting style. On the other hand, the statistics of MSE between the real and the generated images show that the values were smaller than those of MSE between the real images, and it can be considered that the variation of handwriting style was more limited than that of the real

Input image

Generated image

ア

ア イ ウ エ オ カ キ ク ケ フ
サ シ ス セ ソ タ チ ツ テ ト
ナ ニ ヌ ネ ノ ハ ヒ フ ハ ホ
マ ミ ム メ モ ヤ ュ ヨ ラ ソ
ル レ ロ ワ ヲ ン

イ

ア イ ウ エ オ カ キ ク ケ フ
サ シ ス セ ソ タ チ ツ テ ト
ナ ニ ヌ ネ ノ ハ ヒ フ ハ ホ
マ ミ ム メ モ ヤ ュ ヨ ラ ソ
ル レ ロ ワ ヲ ン

ウ

ア イ ウ エ オ カ キ ク ケ フ
サ シ ス セ ソ タ チ ツ テ ト
ナ ニ ヌ ネ ノ ハ ヒ フ ハ ホ
マ ミ ム メ モ ヤ ュ ヨ ラ ソ
ル レ ロ ワ ヲ ン

あ

あ い う え お か き く け こ
さ し す せ そ た ち つ て と
な に ぬ ね の は ひ ふ へ ほ
ま み む め も や ゆ よ り
ら れ ろ わ を ん

い

あ い う え お か き く け こ
さ し す せ そ た ち つ て と
な に ぬ ね の は ひ ふ へ ほ
ま み む め も や ゆ よ り
ら れ ろ わ を ん

う

あ い う え お か き く け こ
さ し す せ そ た ち つ て と
な に ぬ ね の は ひ ふ へ ほ
ま み む め も や ゆ よ り
ら れ ろ わ を ん

Figure 6.8: Y-AE generated images on different characters

images. However, since the minimum value was 1.280, which is non-zero, no character was output exactly the same as the images used in the Y-AE model training. This shows that the Y-AE generator can be used to generate character images of handwriting style that are different from the training dataset. In the next section, the evaluation is done on the usefulness of the generated character images as expanded images by training a character classifier.

Table 6.3: Statistics of the MSE scale between character images of the same character type. The total number of real images is 18,400, including 92 types of Hiragana and Katakana characters in ETL-5 and ETL-9, 200 images for each character. The number of generated images is also 18,400, including 200 randomly selected images for each character type from the generated images by the Y-AE generators.

Comparison target	Average	Variance	Minimum
Real images vs. real images	9.671	5.972	2.224
Real images vs. generated images	5.330	2.255	1.280

6.6.3 Character classification results

Table 6.4 shows the character classification accuracy of each model for the test set. The baseline model (Model(1)) was trained from only ETL-5 and ETL-9 real character images without any data augmentation, resulting in an accuracy of 0.8832 and 0.9061 on the validation and test sets, respectively. By applying three typical data augmentation functions on the same training set (Model (2)), the classification accuracies improved to 0.9159 and 0.9302. On the other hand, the model trained by adding images generated by our proposed Y-AE character generator as data augmentation images (Model(5)) showed an accuracy of 0.8993 on the test set, which was worse than the baseline. The model (Model (6)) trained by applying the data augmentation functions to this training set improved the accuracy to 0.9127, but was not as good as the model trained from the ETL alone. From this result, it is assumed that the character images generated by the Y-AE models contained many characters that were not well formed. In other words, there were a certain number of noisy images that are not useful for training the character classification model. These noisy images can be considered to be an obstacle to the training of the character classifier model. In fact, the accuracy of the classification models trained using only images generated by the Y-AE model alone was 0.8035 (Model(3)) and 0.8620 (Model(4)) on the test set. Considering that the number of images was 46 times larger than the baseline but worse than the baseline, it can be concluded that there were a lot of noise images in the automatically generated images.

Therefore, the accuracies of the classifiers trained with the MSE scale-based and the baseline-based filtering methods improved to 0.9217 (Model(7)) and 0.9310 (Model(9)),

respectively, on the test set when the classification models were trained with the ELT and the image data filtered from the generated images using the MSE scale and the baseline character classifier. Furthermore, applying the same three data augmentation functions as in Model(2) further improved the accuracies to 0.9474 (Model(8)) and 0.9555 (Model(10)). The same results were obtained for the validation set. These results indicate that the generated handwritten character images from the Y-AE generators trained on a limited data set can be sufficiently used as image data for data augmentation by eliminating noise.

A model’s accuracy is further tested by increasing “real data” with additional realistic handwritten character images from different subsets of ETL. Model (11) and Model (12) are the models involved. The results show that the classification accuracies were 0.9598 (w/o DA) and 0.9554 (w/ DA) for the test set, which were not significantly different from those of Model (10). On the other hand, for the validation set, Model (10) was still slightly less competitive with the models trained using the real image data, because a gap of 0.01 could still be observed. However, from these series of experiments, we can claim that the images generated from the Y-AE character generator trained from a limited data set (but with filtering) could generate data close to the real ones.

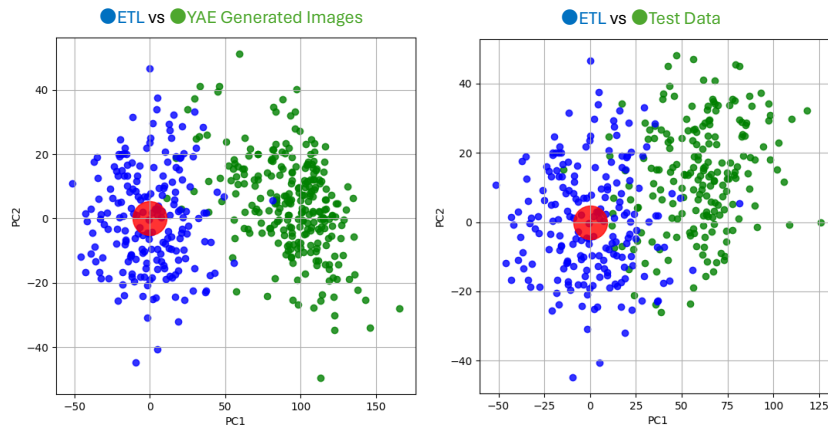
Table 6.4: Character classification accuracy (acc.) for each model. The architecture of the classification model was the same for all. DA indicates whether the three data augmentation functions were applied to the images in a mini-batch or not when a classifier is trained, ✓: DA is applied, ✗: DA is not applied

Model no.	Dataset description	DA	Valid. acc.	Test acc.
(1)	ETL only (baseline)	✗	0.8832	0.9061
(2)	ETL only	✓	0.9159	0.9302
(3)	Generated images (GIs) only	✗	0.7979	0.8035
(4)	GIs only	✓	0.8637	0.8620
(5)	ETL + GIs (all)	✗	0.8910	0.8993
(6)	ETL + GIs (all)	✓	0.9079	0.9127
(7)	ETL + GIs (filtered by MSE)	✗	0.9066	0.9217
(8)	ETL + GIs (filtered by MSE)	✓	0.9411	0.9474
(9)	ETL + GIs (filtered by Model(1))	✗	0.9176	0.9310
(10)	ETL + GIs (filtered by Model(1))	✓	0.9428	0.9555
(11)	ETL + other ETL images	✗	0.9554	0.9598
(12)	ETL + other ETL images	✓	0.9535	0.9554

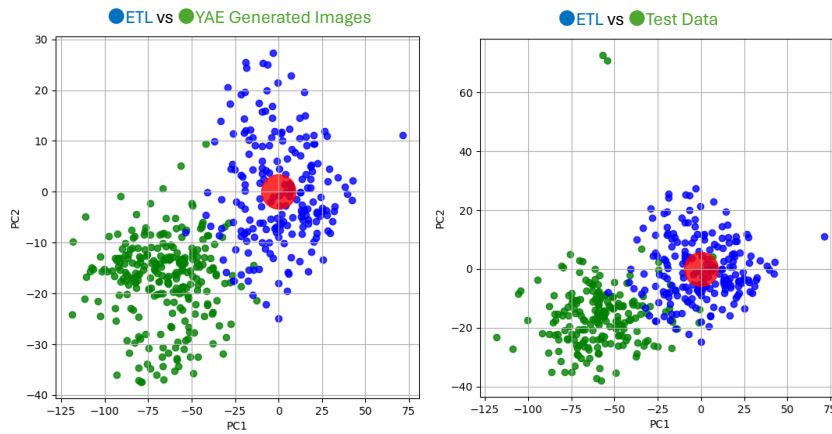
6.6.4 Analysis of Generated Images

A crucial aspect of evaluating the performance of character recognition models lies in understanding how the images generated by the model contribute to recognition accuracy. This understanding is particularly relevant when comparing the characteristics of the

PCA analysis of the features 「あ」



PCA analysis of the features 「い」



PCA analysis of the features 「う」

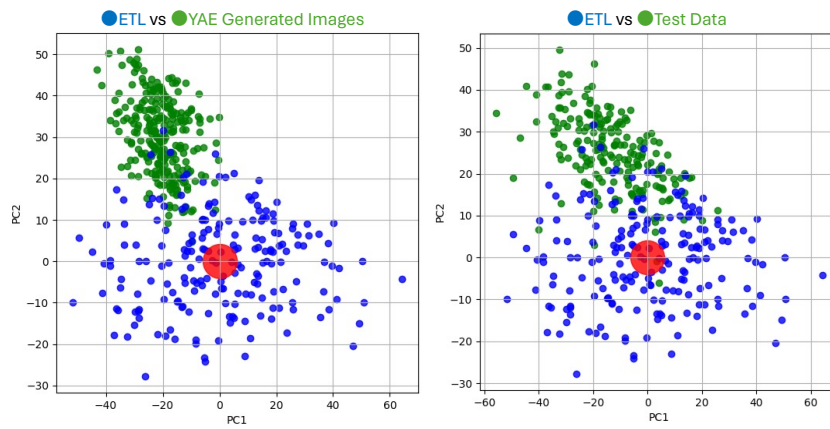


Figure 6.9: PCA analysis comparing ETL images, generated images, and test images

training dataset, generated images, and test images. To this end, a Principal Component Analysis (PCA) was conducted on images from the ETL dataset, images generated by the Y-AE with AdaIN, and the test dataset.

The PCA analysis aimed to visualize and compare the feature spaces of different sets of images. The backbone of the model, specifically its feature extractor, was employed to process the generated images. The resulting features were then analyzed using PCA, a statistical procedure that converts a set of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. Figures 6.9 and 6.10 present the PCA results, showing the distribution of 208 ETL images, 263 Y-AE generated images, and 200 test images. The test images include characters such as あ, い, う, え, and お. A notable observation from these figures is the distinct separation between the ETL images and the Y-AE generated images in the first and second components of PCA. This separation clearly indicates that these two sets of images do not share the same distribution.

An intriguing aspect observed from the PCA analysis is the difference in distribution between the ETL training data and the test data. While the ETL images are used as training data, their distribution significantly differs from that of the test data. This disparity in distribution could be a contributing factor to the variations in recognition accuracy. The analysis suggests that the addition of the Y-AE generated images to the training dataset may help bridge the gap between the training and test distributions. By encompassing a broader range of features, the model is better equipped to handle the variability present in the test data, potentially leading to improved recognition accuracy.

The PCA analysis provides valuable insights into the relationship between training data, generated images, and test data in the context of character recognition. The clear distinction between the distributions of these image sets underlines the importance of incorporating diverse data sources during the training phase to enhance the model’s generalization capabilities.

6.7 Experiment on Kanji Image Generation

To assess the feasibility of generating Kanji characters using Y-AE, an experiment with same training setup explained in section 6.6 was conducted with a subset of ETL9’s Kanji characters. Out of the 2965 Kanji characters listed in the ETL9 dataset, 2921 were selected for training, primarily due to the extensive resources required for the YAE’s training and generation process. The methodology involved training separate Y-AE models for distinct groups of Kanji characters, each categorized by stroke count. This segmentation was necessary because training all Kanji characters in a single model can’t be trained to generate character images.

The experiment’s outcome, as shown in Figure 6.11, demonstrated successful generation of Kanji characters with Y-AE incorporating AdaIN. The diversity of the generated

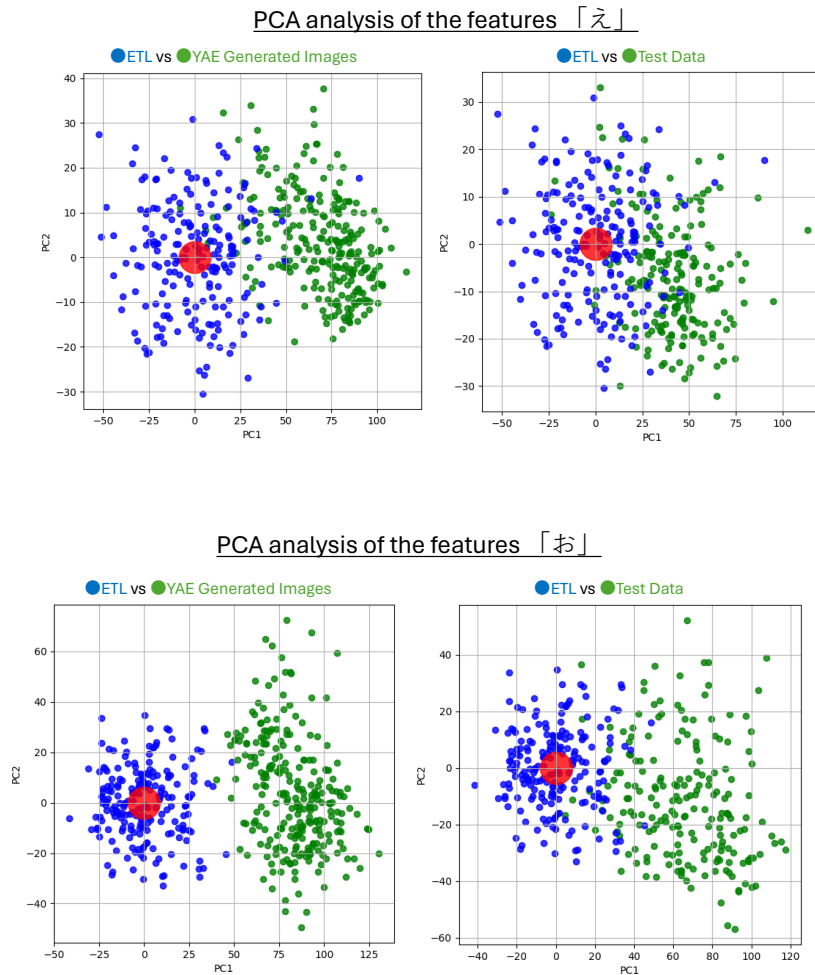


Figure 6.10: PCA analysis comparing ETL images, generated images, and test images

images indicated the method’s potential for enhancing the accuracy of character classifiers, particularly in line-by-line recognition scenarios. Each Kanji character led to the generation of 200 distinct images, translating to a staggering 118,600,000 images overall, considering all character types. However, after applying a filtering process via a character classifier, the usable image count was reduced to 37,199,046. The specific generated image statistics are shown in appendix A. This reduction, accounting for approximately 68.63% of the generated images being deemed unsuitable, highlighted the need for further improvements in the Y-AE model, especially regarding the generation quality of Kanji characters.

The generated Kanji images, despite the filtration process reducing their number, provided a significant dataset to validate the effectiveness of single-line and multi-line text classifiers, as discussed in chapter 7. This validation was crucial in determining the practical applicability of the Y-AE model in real-world OCR scenarios, where Kanji characters are prevalent.

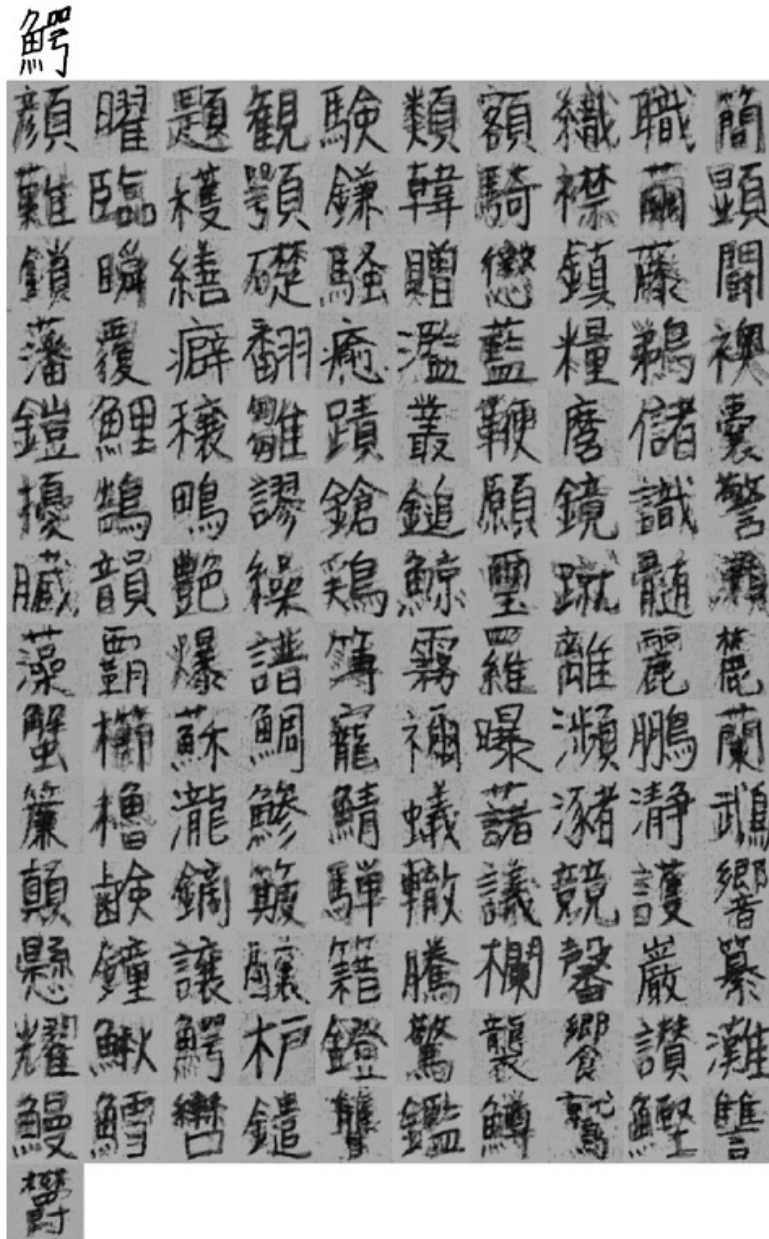


Figure 6.11: Generated result samples of Kanji

6.8 Conclusions

In this thesis, a data augmentation technique is proposed to train a character classifier with deep learning by automatically adding generated images to the training set using a Y-AE-based conditional generation model. Because the original Y-AE model [3] could not represent handwriting with rich variations in handwritten style, an improvement was proposed and performed by applying AdaIN. The Y-AE model successfully generated handwritten character images with a wide variety of handwriting. On the other hand, the generated character image set contained noise (not suitable for training a character

classifier); therefore, character similarity using the MSE scale and character filtering using the character classifier trained with the real handwritten character images dataset only are applied.

The effectiveness of the proposed method as a data augmentation was evaluated in terms of the accuracy of the character classifier. The experimental results showed that the character images generated by the Y-AE generator alone were not as good as the character classifier trained only with real handwritten character images; however, they were very useful as an extension to the dataset of real handwritten character images. In addition, it was also shown that existing data augmentation functions, such as Affine transformations, could also be applied to the generated character images. Finally, the character classification accuracy of the baseline model on the test set was 0.9061, while our proposed method achieved 0.9555, which was a significant improvement of 0.0494 points. This was a 47.4% improvement in the character error rate.

6.9 Summary

This chapter presented the Y-Autoencoder enhanced with AdaIN for generating diverse character images, the filtering methods for filtering the generate character images which is an essential step for improving OCR systems. The integration of AdaIN into Y-AE enables the model to produce a wide range of realistic and varied character styles, thereby enriching the training datasets for OCR. Also, by using the filtering methods, the removal of unuseful generated character images is performable. The successful implementation and evaluation of this model mark an improvement in the field of OCR technology, opening new possibilities for robust and efficient character recognition systems.

Chapter 7

Single-line Text Detection In Multiple-lines Text Images

Chapter 4 discusses methods using image recognition and text detection based on deep learning, which have been used recently. This chapter describes the improvements made to the CRAFT base of previous studies. A description of the model structure that adds a Line Segmentation branch to the Region Score and Affinity Score outputs of the previous CRAFT study to allow the detection of narrow multi-line segments is made. Post-processing to enable detection of narrow line spacing in text will also be described. The proposed method significantly reduces the text error rate of the Text Recognizer. In addition to the model structure, this chapter describes in detail the training method, how to create labels for image data, and the experimental conditions and results for evaluating the proposed method and comparing it with other methods.

7.1 Model Architecture

As highlighted in Chapter 4, this study builds upon the Character Region Awareness For Text (CRAFT) model [37], renowned for its character-level text detection capabilities. Our primary innovation lies in augmenting the original CRAFT framework with an additional line segmentation branch, as shown in Figure 7.1. The original CRAFT model adeptly estimates character regions through its region score and gauges the connections between characters using the affinity score, collaboratively delineating text regions. However, the model's reliance solely on region and affinity scores occasionally leads to misinterpretations, such as mistaking individual characters for radicals in Chinese characters or misclassifying characters in multi-line texts.

Original CRAFT

Line Segmentation Branch

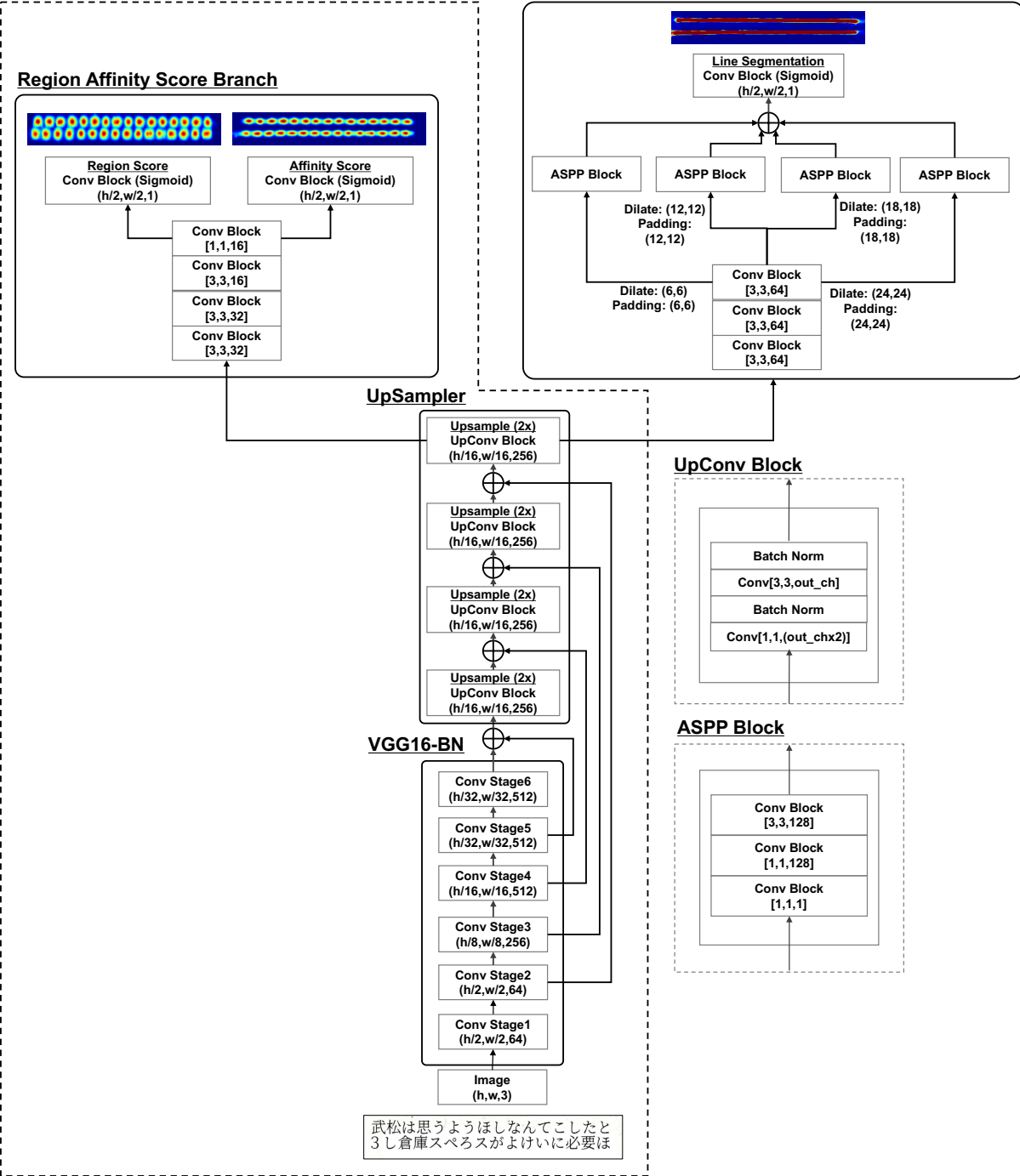


Figure 7.1: The enhanced CRAFT model architecture

The integration of a task to explicitly estimate the region of each line is performed to circumvent these limitations and enhance the precision of line detection in multi-line texts. This novel line segmentation branch, inspired by and adapted from the LinkRefiner approach by Baek et al. [102], refines the affinity score in the CRAFT framework. The crux of our enhancement strategy diverges from the original LinkRefiner methodology; rather than treating it as a sequential or separate module, our model synergistically estimates region scores, affinity scores, and line segmentation in a unified process. This concurrent estimation approach is pivotal, allowing the model to capture the nuances of line-based text structures with greater accuracy and efficiency. Our modifications to the CRAFT model do not just build upon its existing strengths but significantly expand its capabilities, particularly in handling complex, multi-line textual layouts.

7.1.1 Region and Affinity Score Label

In this study, deep learning models are utilized, specifically the original CRAFT model [37] and its enhanced iteration, which pivot on the concepts of region and affinity scores for character detection. These scores are critical in identifying individual characters and the spaces between them, represented as two-dimensional Gaussian heatmaps derived from character bounding box data. The region score is pivotal in pinpointing the central probability of a character within its bounding box. This score is crucial for accurately locating each character, especially in complex text layouts. Traditional Gaussian heatmap representations, being uniformly shaped, often fall short in encapsulating the nuanced variations in character sizes and shapes. To counter this limitation, as illustrated in Figure 7.2, a modified Gaussian heatmap is employed. This heatmap adapts to the bounding box's specific dimensions, thereby offering a more precise and tailored representation of each character's central region.

The affinity score plays a complementary role, focusing on the inter-character regions. It is calculated by considering pairs of adjacent character bounding boxes. By drawing a diagonal across each bounding box, two triangles per character pair are formed. The process involves using the centroids of these triangles to define new bounding boxes that represent the space between adjacent characters which shown in Figure 7.2. This innovative approach to affinity score calculation allows the model to effectively discern the proximity and connectivity between characters, a key aspect in handling closely spaced text and improving overall text detection accuracy.

Both the region and affinity scores are integral to the CRAFT model's ability to perform nuanced text detection. Their heatmap representations, enhanced to adapt to the varying shapes and sizes of characters, are a testament to the model's advanced capabilities in handling diverse text scenarios. This level of precision is essential for the model's application in complex OCR tasks, where accurately distinguishing individual characters and their relational spacing is paramount.

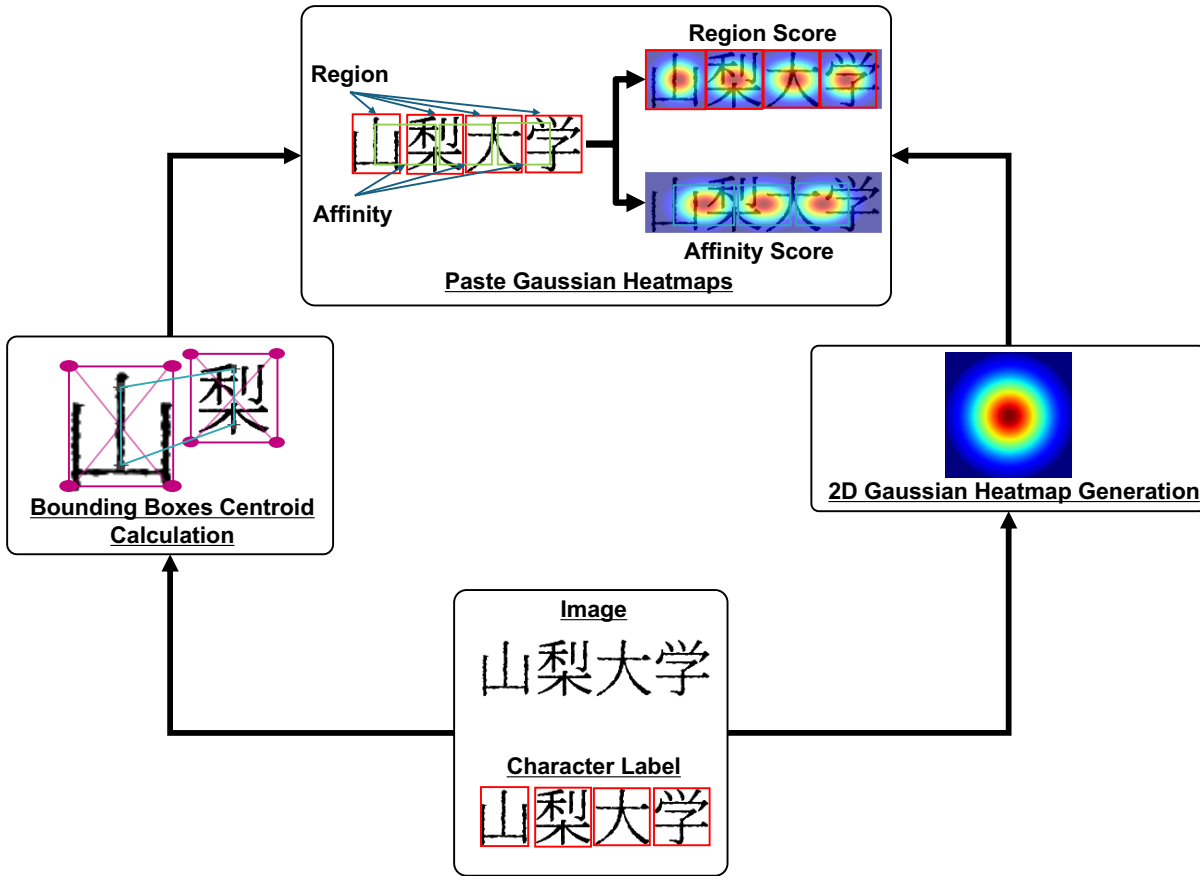


Figure 7.2: Image label for training the CRAFT

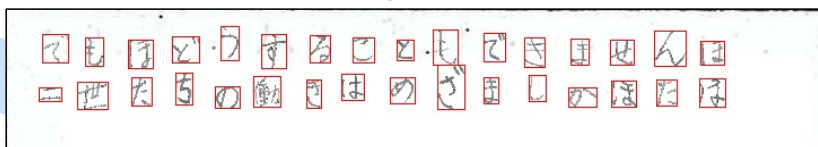
7.2 Label for Enhanced CRAFT

7.2.1 Line Segmentation Label

The generation of line segmentation labels, a crucial aspect of training the enhanced CRAFT model, leverages the character bounding box information inherently required by CRAFT. Illustrated in Figure 7.3, this automated label creation process capitalizes on the detailed bounding box data for each character. The procedure initiates by calculating the centroid of each character's bounding box within a line, subsequently connecting these centroids to form a trajectory. This step is followed by determining the average height of all character bounding boxes within a line, denoted as h' .

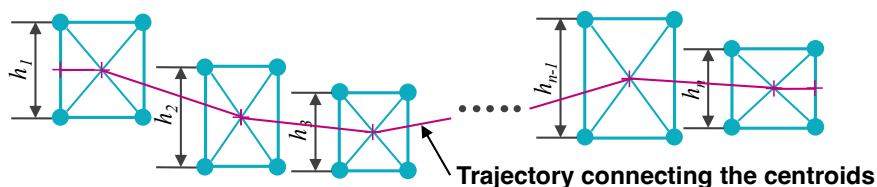
Next, the thickness of this trajectory is expanded to a factor of h'/r , where r is a strategically chosen hyperparameter. The selection of r balances a critical trade-off: a smaller r results in a thicker trajectory, aiding in the separation of closely spaced lines but potentially complicating the distinction of individual lines. Conversely, a larger r value yields a thinner trajectory, enhancing line detection but possibly hindering the model's ability to separate closely spaced text. After extensive experimentation, an optimal value of $r = 5$ was identified for this study, striking a balance between line separation and

Character-wise bounding box



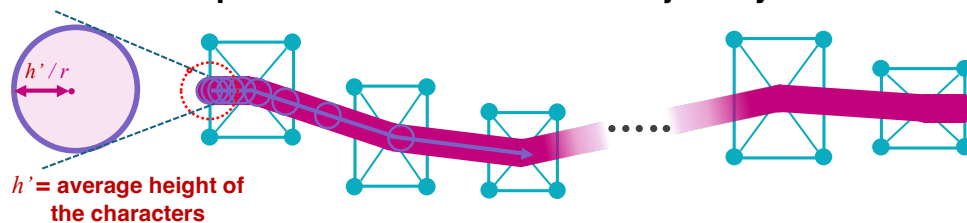
Step(1)

Calculation of the centroid point of each bounding box and the average height of all the character bounding boxes



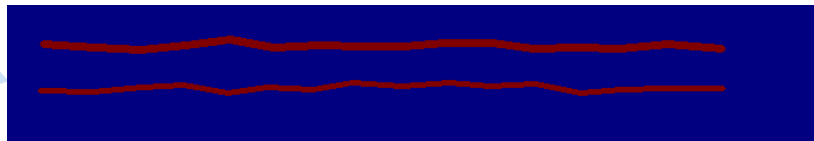
Step(2)

Expand the thickness of the trajectory to h'/r times



Step(3)

Fix line segmentation label



 Character bounding box  Centroid of a character bounding box

Figure 7.3: Line Segmentation Label

detection accuracy.

The expanded trajectories are designated as line segmentation labels, providing a clear representation of the line structure within the text. For bounding boxes at the extremities of a line, the midpoints of their sides are also integrated into the trajectory, ensuring a comprehensive depiction of the character's spatial domain. This method even enables the representation of line segmentation for isolated characters, thereby establishing a robust annotation framework that comprehensively expresses the inter-character connectivity.

7.3 Post-Processing Algorithm for Multi-Line Text Detection Using Enhanced CRAFT

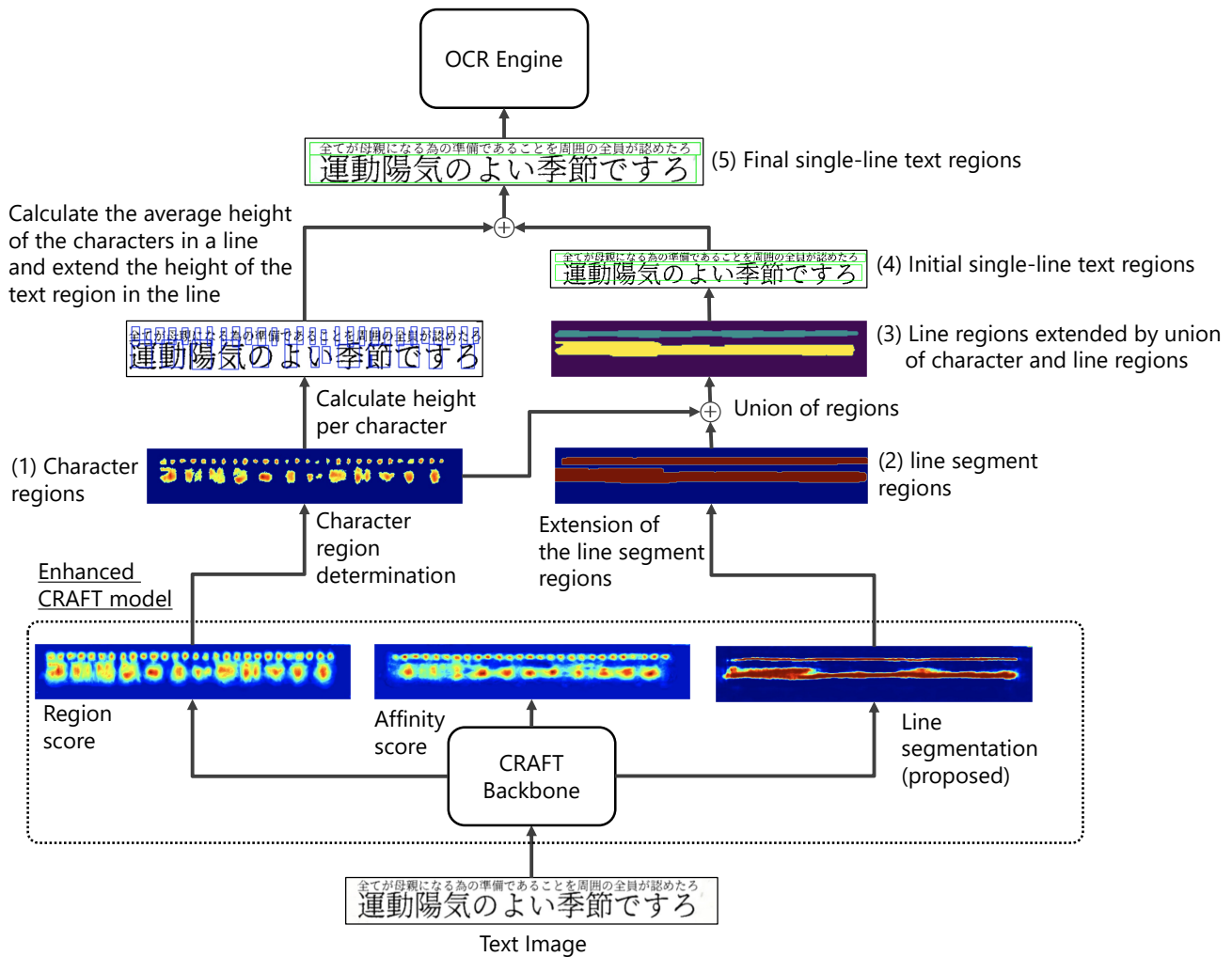


Figure 7.4: Enhanced CRAFT Post-processing

7.4 Post-processing Methodology

The incorporation of the line segmentation branch into our model, as shown in Figure 7.1, significantly enhances the accuracy of single-line text detection compared to the original model without this enhancement. However, challenges arise when parts of characters within the detected text region extend beyond the designated area, rendering them unsuitable for character recognition and subsequent OCR system processing. To address this issue, an innovative post-processing technique that combines the single-line text region estimated from line segmentation with the character region outputs from the original

CRAFT model is proposed. This novel approach not only boosts single-line text detection performance but also ensures the generation of images that are optimally cropped for OCR processing. It is important to highlight that our proposed method is tailored to enhance text line detection, particularly in document data where text lines are demarcated by clear horizontal lines.

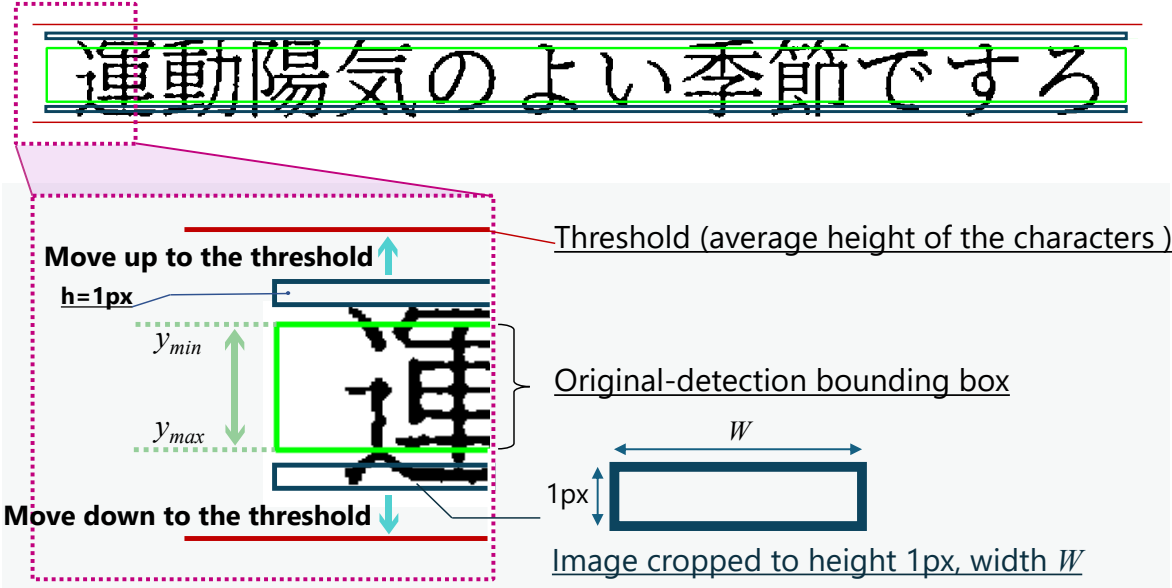


Figure 7.5: Height determination process

The procedure of our post-processing method is visually represented in Figure 7.4. Initially, an image serves as the input to the CRAFT model, from which character regions, the connectivity between adjacent characters, and line segmentation regions are delineated. The model’s region score branch computes a probability distribution for character regions, with areas exhibiting a probability greater than 0.6 identified as character regions, as marked (1) in Figure 7.4. The model further expands the single-line segmentation region, labeled (3) in Figure 7.4, by amalgamating these character regions with the single-line segmentation region identified by the line segmentation branch, indicated as (2) in Figure 7.4. Consequently, a text bounding box, shown as (4) in Figure 7.4, can be extracted from this expanded line segmentation. However, images cropped using this text bounding box are often unsuitable for OCR, primarily due to the bounding box’s limited width, which results in missing text. To mitigate this, we can calculate the average height of each character in image (2), and accordingly expand the text region in image (4) to incorporate the missing text segments, ultimately deriving the final single-line text region (5). The algorithm for determining region (5) is detailed in Figure 7.5. This process involves a meticulous expansion methodology, where the calculated average character height serves as a threshold, represented by the red line in Figure 7.5. The parameters y_{min} and y_{max} from the originally detected bounding box (in green) guide the cropping of a 1 px height and W px width image segment (in blue), iteratively adjusted both upwards and

downwards. Each iteration involves a pixel-by-pixel evaluation of the cropped image to ascertain if its average value reaches 255, indicating a completely blank image. If such a scenario occurs, the y-coordinate is set as the final height for the intended cropped image; otherwise, the final y_{min} or y_{max} values are methodically expanded by the average character height, ensuring a precise and comprehensive text region for OCR applicability.

7.5 Loss Functions

7.5.1 Loss Functions for Enhanced CRAFT Model

Mean Squared Error (MSE) and Binary Focal Loss are employed as the principal loss functions in this thesis. These functions play a pivotal role in refining the model’s accuracy for text detection tasks. The following sections detail these loss functions, along with Binary Cross Entropy Loss, which forms the basis of Binary Focal Loss.

Mean Squared Error for Gaussian Heatmaps

Mean Squared Error (MSE) is a fundamental statistical tool used to measure the average of the squares of errors, i.e., the difference between estimated and actual values. In our model, MSE is specifically applied to calculate the loss for Gaussian heatmaps representing region and affinity scores. This is crucial for accurately identifying text regions and the connections between characters. MSE is mathematically defined as:

$$MSE = \frac{1}{N} \sum_{i=1}^n (d_i - p_i)^2 \quad (7.1)$$

Here, p_i denotes the model’s predicted Gaussian heatmap values, and d_i represents the label heatmap values. By emphasizing the square of errors, MSE ensures that the model is fine-tuned to precisely detect text regions and connections, especially in documents with complex layouts.

Binary Focal Loss for Line Segmentation

Binary Cross Entropy Loss (BCE) is a crucial measure in binary classification models, including text detection systems. It calculates the divergence between the predicted probabilities and the actual binary labels, guiding the model to improve its classification accuracy. The BCE is represented by the formula:

$$BCE = - \sum_{i=0} [d_i \log p_i + (1 - d_i) \log(1 - p_i)] \quad (7.2)$$

In this context, p_i represents the model’s output probabilities, and d_i indicates the actual binary labels. BCE is essential for differentiating text regions from non-text regions for

the line segmentation effectively.

Binary Focal Loss (BF) is a loss function proposed to correct learning failures on unbalanced data, and can dynamically scale the BCE Loss for unbalanced data. In the case of determining whether a line is a background or a written line, as in the line segmentation in this study, the majority of the image is background, and most of the learning is dominated by a simple background determination. Therefore, BF is used to scale the loss to a small number of examples that are successfully and easily classified.

BF can be shown in the following equation (7.3). Note that where γ is the scaling parameter and p_t is the probability of belonging to the positive class. The t in p_t corresponds to the index of the positive class in the case of binary classification such as the line segmentation estimation task in this thesis. In this thesis, the parameter of BF, α was 0.25, γ was 2.00 for training the proposed model.

$$BF(p_t) = -\alpha(1 - p_t)^\gamma \log p_t \quad (7.3)$$

7.6 Experiment Condition and Dataset

7.6.1 Train Dataset

In this thesis, evaluation is conducted on single-line text detection capabilities across the models, including our proposed model. For training the models, the training dataset was created by downloading from 3,398 document formats available on the Yamanashi Prefectural Government’s website. These documents were converted to images, and line-by-line and character-by-character bounding box labels for text and blank areas that were manually labeled to the images by human, totaling 143,948 of them. By randomly pasting fonts or handwritten characters to each of these labels area, we generated Gaussian heat maps and row-by-row segmentation maps, which are the teacher data needed for CRAFT training. When training the model, we used randomly cropped images and their corresponding label images. In addition to the fonts, the randomly pasted characters were used to create the training data, the ETL database [103] was used for the handwritten images and Balanced Corpus Contemporary Written Japanese [104] for the text. Two examples document formats that labeled with character bounding boxes is shown as Figure 7.6.

7.6.2 Test Dataset

To rigorously evaluate our models, “Test Set A” is designed to contain 600 images, split evenly between font and handwritten text, with two to four lines of text per region each. These images exhibit a range of line spacings, some with narrowly spaced lines, others with overlapping bounding boxes, offering a challenging environment for our detection algorithms. In addition, to test the text detection performance of narrowed multiple lines

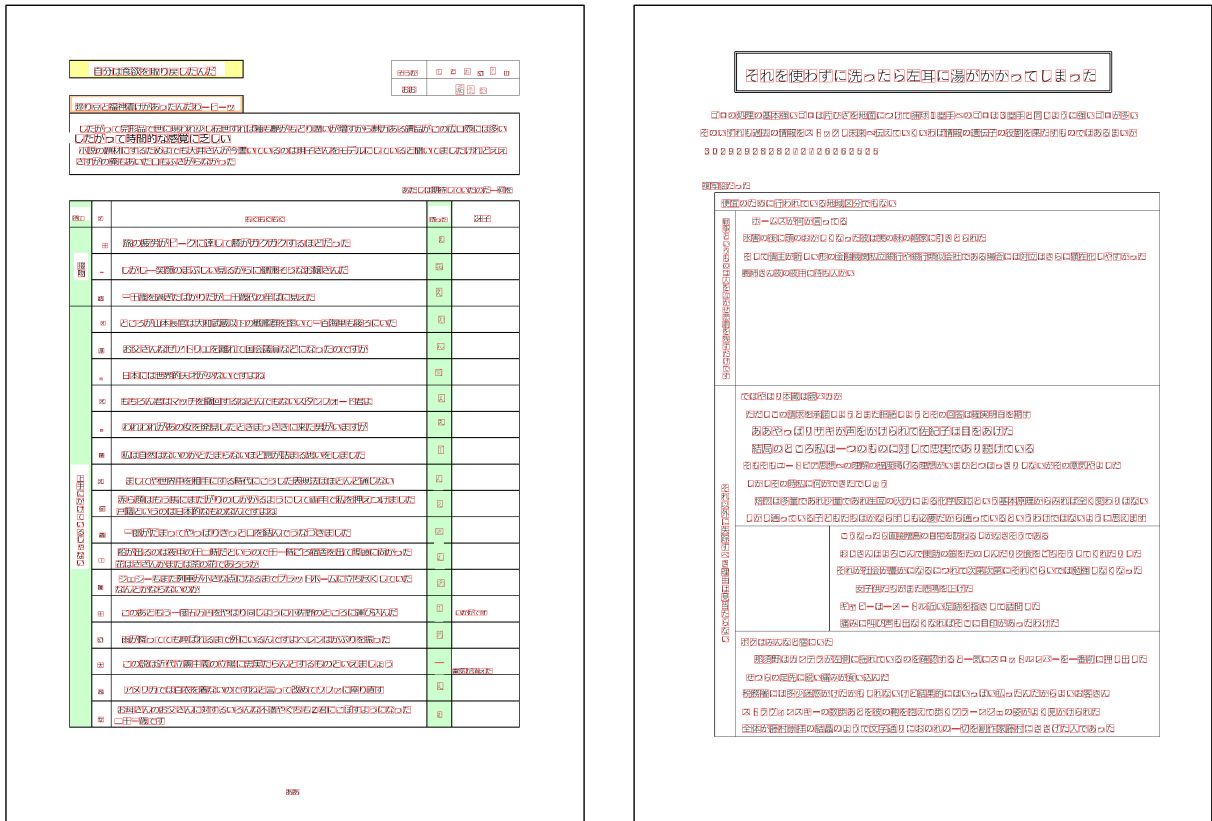


Figure 7.6: Text Detection Model Train Dataset Examples

images in this study, the narrowed test image height was fixed at 256 pixels and 2 to 4 lines of characters were randomly generated as shown in Figure 7.7. In doing so, the line spacing pixel of the characters was narrowed by multiplying “line spacing ratio” by the number of pixels for the height of the generated characters, and the calculation formula is as follows.

$$\text{line spacing pixel} = \text{line spacing ratio} \times \text{character font size height} \quad (7.4)$$

A “Test Set B” containing 300 images categorized by line spacing ratios of -0.1 , 0.0 , and 0.1 is created. This set was instrumental in assessing our models’ ability to handle varying line spacings, a critical factor in accurate text detection and OCR performance.

7.6.3 Evaluation Metrics

Intersection over Union (IoU)

The Intersection over Union (IoU) metric is particularly useful in text detection tasks for evaluating the accuracy of detected text regions compared to ground truth annotations. The IoU is calculated using the coordinates of the predicted and ground truth bounding boxes. Consider the coordinates of the bottom-left corner and the top-right corner of the

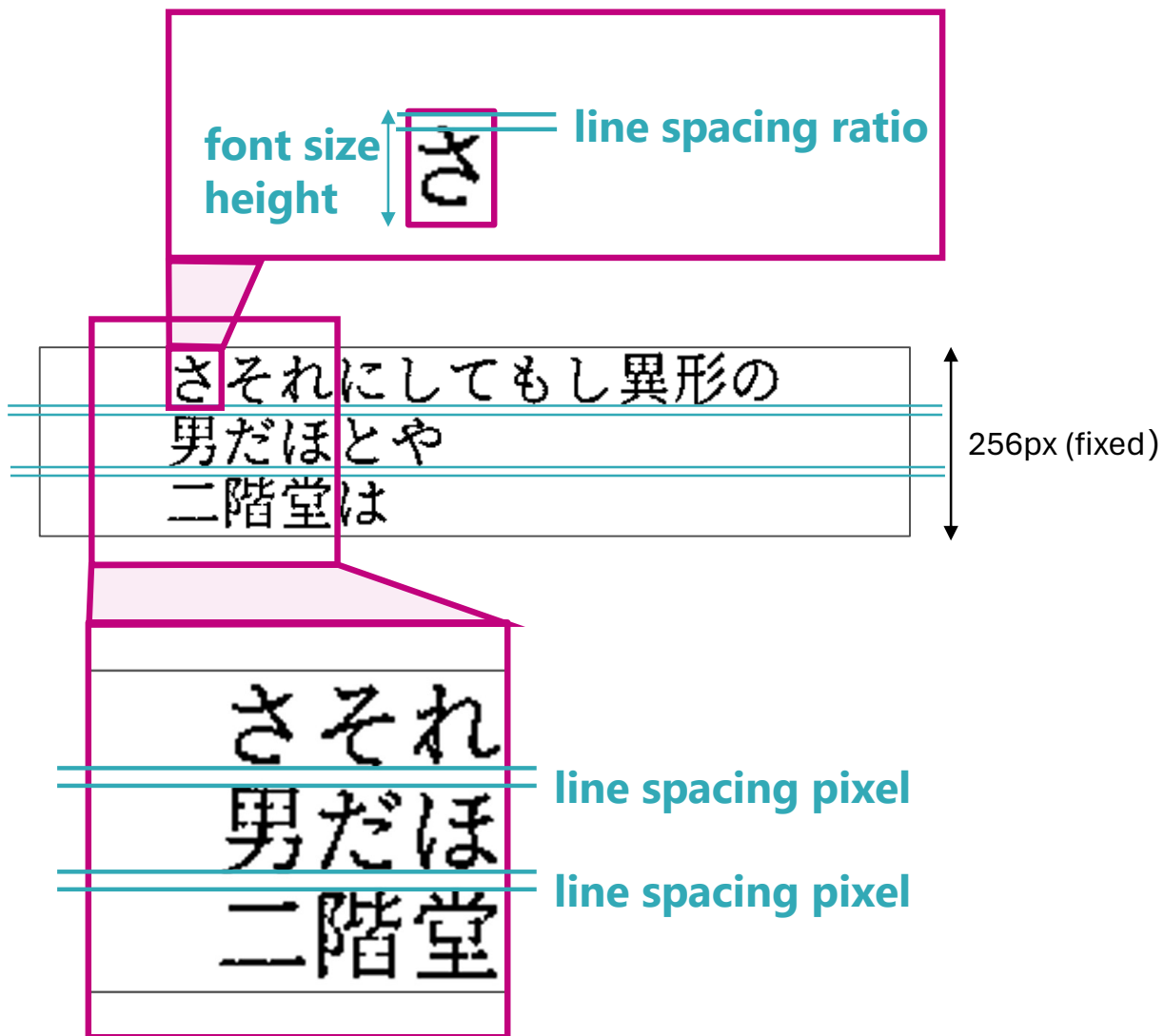


Figure 7.7: Narrowed multiple text lines image using line spacing pixel which get from the font size height by multiplying the line spacing ratio.

bounding boxes:

- Predicted bounding box: $(x_{p1}, y_{p1}), (x_{p2}, y_{p2})$
- Ground truth bounding box: $(x_{g1}, y_{g1}), (x_{g2}, y_{g2})$

The IoU is computed as follows:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (7.5)$$

where the Area of Overlap is given by:

$$\text{Area of Overlap} = \max(0, \min(x_{p2}, x_{g2}) - \max(x_{p1}, x_{g1})) \times \max(0, \min(y_{p2}, y_{g2}) - \max(y_{p1}, y_{g1})) \quad (7.6)$$

and the Area of Union is calculated by adding the areas of both bounding boxes and then subtracting the Area of Overlap:

$$\text{Area of Union} = \text{Area}(\text{predicted}) + \text{Area}(\text{ground truth}) - \text{Area of Overlap} \quad (7.7)$$

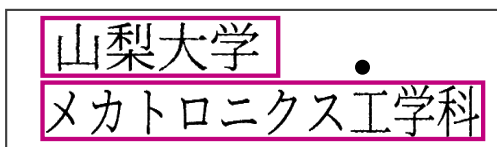
This metric helps in assessing the precision of text localization by the detection model. Furthermore, in text detection tasks, the evaluation metrics of F1 Score, Recall, and Precision, especially at IoU thresholds of 0.5 and 0.75, are crucial.

- **Recall** is defined as the ratio of correctly detected text regions to the total number of ground truth text regions.
- **Precision** is the ratio of correctly detected text regions to the total number of detected text regions by the model.
- **F1 Score** is the harmonic mean of Precision and Recall, balancing both metrics.

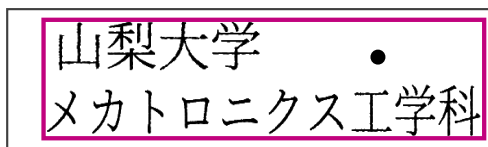
These metrics provide a more comprehensive understanding of a model's performance in text detection, taking into account the accuracy of overlap between predicted and actual text regions. In this thesis, we evaluate the predicted bounding boxes with IoU values of 0.50 and 0.75.

Additionally, metrics like Correct Line Segmentation, Over Segmentation, and Under Segmentation are introduced, providing detailed insights into each model's performance in Test Set B, which features varied line spacing scenarios. Figure 7.8 displays the seg-

Correct-line Segmentation



Under Segmentation



Over Segmentation

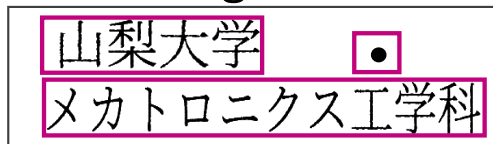


Figure 7.8: Type of segmentation in texts

mentation types developed for text detection tasks. Correct Line Segmentation happens when the number of bounding boxes in the prediction equals that in the ground truth. For instance, if an image in the ground truth contains five bounding boxes and the model predicts the same number, it's labeled as correct. Under Segmentation is identified when the model predicts fewer bounding boxes than the ground truth, each instance contributing to the Under Segmentation count. Similarly, Over Segmentation occurs when the model predicts more bounding boxes than the ground truth, with each such instance counted as Over Segmentation. These metrics are crucial for assessing the precision of an OCR model in detecting the correct number of text regions as per the ground truth. The equations for these metrics are as follows:

$$\text{Correct Line Segmentation} = \frac{\text{Number of Images Correctly Detected}}{\text{Total Number of Images}} \quad (7.8)$$

$$\text{Under Segmentation} = \frac{\text{Number of Images Under Detected}}{\text{Total Number of Images}} \quad (7.9)$$

$$\text{Over Segmentation} = \frac{\text{Number of Images Over Detected}}{\text{Total Number of Images}} \quad (7.10)$$

These formulas help quantify the model's accuracy in terms of detecting the exact, fewer, or more text regions compared to the actual number in the ground truth.

Character Error Rates (CER)

Character Error Rates (CER) are crucial in evaluating the performance of Optical Character Recognition (OCR) models. CER measures the accuracy of the recognized text against the ground truth at the character level. It is calculated using the formula:

$$\text{CER} = \frac{\text{Substitutions} + \text{Insertions} + \text{Deletions}}{\text{Total Number of Characters in Ground Truth}} \quad (7.11)$$

- **Substitutions** are the number of characters incorrectly recognized.
- **Insertions** are the additional characters that were not in the ground truth.
- **Deletions** are the characters from the ground truth that were not recognized.

CER is expressed as a percentage, with lower values indicating better text recognition accuracy. This metric is especially important in OCR systems where precise character recognition is essential.

7.6.4 Model Training and Evaluation Procedures

The Adam optimization function [86] is employed for model training, setting an initial learning rate of 0.0001 for the CRAFT model and its line segmentation variant, and

0.001 for DBNet and DBNet++. Our training method included a suite of basic data augmentation methods ¹, such as cropping, rotation, and color modification, to boost our models’ robustness. After 2,000 epochs, the models exhibiting the highest F1 scores on the validation dataset were selected for further testing.

EasyOCR [58] was employed as the OCR engine, focusing solely on its OCR core engine and excluding its text detection component. This strategic choice allowed us to concentrate on the core OCR capabilities, pivotal in our evaluation of single-line text detection and character recognition. Our evaluation methodology was comprehensive, employing recall, precision, and F1 score metrics at IoU values of 0.50 and 0.75, to ascertain the models’ efficacy in correctly identifying text regions.

7.7 Experiment Result

Table 7.1: Single-line text detection accuracy of each detection method when the IoUs were *0.50 / 0.75* which separated with / symbol. The numbers in the upper, middle, and lower rows in each cell are the results for the font test set only, the handwritten test set only, and both the test sets, from Test Set A.

Detection Methods	Recall	Precision	F1 score
CRAFT (baseline)	0.378 / 0.313	0.558 / 0.462	0.450 / 0.373
	0.543 / 0.370	0.335 / 0.228	0.414 / 0.282
	0.464 / 0.342	0.396 / 0.293	0.427 / 0.316
DBNet	0.848 / 0.432	0.911 / 0.464	0.879 / 0.448
	0.745 / 0.605	0.828 / 0.672	0.784 / 0.636
	0.794 / 0.522	0.868 / 0.571	0.830 / 0.545
DBNet++	0.886 / 0.805	0.922 / 0.837	0.904 / 0.821
	0.559 / 0.354	0.642 / 0.406	0.597 / 0.378
	0.715 / 0.569	0.783 / 0.623	0.747 / 0.595
CRAFT+line seg. w/o post-processing	0.394 / 0.329	0.587 / 0.490	0.471 / 0.394
	0.547 / 0.367	0.388 / 0.261	0.454 / 0.305
	0.474 / 0.349	0.448 / 0.330	0.461 / 0.339
CRAFT+line seg. w/ post-processing (proposed)	0.895 / 0.824	0.911 / 0.839	0.903 / 0.832
	0.880 / 0.682	0.876 / 0.679	0.878 / 0.680
	0.887 / 0.750	0.893 / 0.755	0.890 / 0.752

Table 7.1 presents a comparative analysis of single-line text detection performance across various methodologies when evaluated at IoU threshold of 0.50 and 0.75. At the lower IoU threshold (0.50), where the detection criteria are less stringent, both DBNet and DBNet++ demonstrate superior performance. In contrast, at this IoU level, the original CRAFT model and its extension with the line segmentation branch lag behind in detection accuracy. However, the integration of our advanced post-processing step in the

¹<https://pytorch.org/vision/stable/transforms.html>

Table 7.2: OCR accuracy (CER [%]) results for the text detected using each single-line detection method. The CERs are for the typeset dataset only because the OCR engine supports only typeset characters.

Detection methods	CER
CRAFT [37] (baseline)	56.2
DBNet [1]	44.8
DBNet++ [2]	38.9
CRAFT+line seg. w/o post-processing	54.4
CRAFT+line seg. w/ post-processing (proposed)	16.0
Oracle	4.1

CRAFT+line segmentation model marks a notable improvement in accuracy, surpassing DBNet++ across both typeset and handwritten test sets.

The scenario shifts at a higher IoU threshold (0.75), where stricter accuracy requirements lead to a general decline in detection performance. DBNet and DBNet++ exhibit a more pronounced decrease in performance due to deviations from the true text region, often caused by narrow line spacing and other factors, as illustrated in Figure 7.9 (b) and (c). This discrepancy raises concerns about the accuracy of text images fed into the OCR system. Our proposed method, however, maintains a robust F1 score of 0.752 even at this higher IoU level, demonstrating a lesser decline in accuracy compared to other models and ensuring closer alignment with true text regions. The detection of handwritten text is conducted as a supplementary detection experiment to confirm the efficacy for handwritten text recognition. The results are shown in Figure 7.10. As can be seen from the results, DBNet++ in (c) and the proposed method (d) are also effective in detecting multiple lines of handwriting.

The distinction in test set types (typeset vs. handwritten) reveals a notable disparity, with the handwritten test set consistently showing lower accuracy across all models. This can be primarily attributed to under-segmentation in the detected text line regions: a change in IoU from 0.50 to 0.75 led to a significant drop in F1 score from 0.878 to 0.680, even for our proposed method. The inherent variability in shape, size, and consistency of handwritten characters poses challenges in accurately estimating character and line segmentation regions, leading to under-segmentation issues.

Table 7.2 details the OCR accuracy for text detected using each method. The OCR engine, trained exclusively on typeset characters, could not evaluate handwritten text. Hence, the OCR assessment was confined to the typeset dataset. The 'Oracle' in Table 7.2 represents the ideal CER for perfect single-line text detection. A mismatch between detected and actual character regions results in increased CER due to compromised character recognition accuracy. The existing models, CRAFT, DBNet, and DBNet++, record significantly higher CERs than the Oracle, illustrating their limitations as OCR preprocessors due to frequent character protrusions from the detected text region, especially at higher IoU values.

The introduction of the line segmentation branch in the CRAFT model (without post-processing) exhibited a marginal CER improvement of 1.8% over the baseline model, underscoring its moderate effectiveness. However, the application of our post-processing method, utilizing line segmentation results, significantly enhanced OCR accuracy, reducing the CER to 16.0%, a substantial improvement of 40.2% over the baseline model. This indicates that our proposed method adeptly captures the necessary character images for the OCR system without overextending the text region or including irrelevant characters from adjacent lines.

Table 7.3: Results for spacing: -0.1 (recall, precision, F1 score, correct segmentation, Over Segmentation, and Under Segmentation for IoU of 0.50 and 0.75 on Test Set B.)

Detection methods	Recall (0.50 / 0.75)	Precision (0.50 / 0.75)	F1 (0.50 / 0.75)	Correct lines seg.	Over seg.	Under seg.
DBNet	0.16 / 0.13	0.34 / 0.26	0.22 / 0.17	0.17	0.00	0.83
DBNet++	0.17 / 0.13	0.33 / 0.24	0.23 / 0.17	0.16	0.01	0.83
CRAFT	0.10 / 0.02	0.25 / 0.02	0.14 / 0.01	0.00	0.00	1.00
CRAFT+l.s. w/o p.p	0.11 / 0.00	0.29 / 0.00	0.16 / 0.00	0.02	0.00	0.98
CRAFT+l.s w/ p.p	0.33 / 0.11	0.54 / 0.19	0.41 / 0.14	0.18	0.01	0.81

Table 7.4: Results for spacing: 0.0 (recall, precision, F1 score, correct segmentation, Over Segmentation, and Under Segmentation for IoU of 0.50 and 0.75 on Test Set B.)

Detection methods	Recall (0.50 / 0.75)	Precision (0.50 / 0.75)	F1 (0.50 / 0.75)	Correct lines seg.	Over seg.	Under seg.
DBNet	0.50 / 0.39	0.71 / 0.56	0.58 / 0.46	0.48	0.01	0.51
DBNet++	0.54 / 0.42	0.73 / 0.57	0.62 / 0.48	0.57	0.01	0.42
CRAFT	0.12 / 0.08	0.25 / 0.17	0.16 / 0.11	0.07	0.01	0.92
CRAFT+l.s. w/o p.p	0.08 / 0.05	0.18 / 0.11	0.11 / 0.10	0.10	0.01	0.89
CRAFT+l.s w/ p.p	0.83 / 0.60	0.86 / 0.62	0.85 / 0.61	0.77	0.08	0.15

Table 7.5: Results for spacing: 0.1 (recall, precision, F1 score, correct segmentation, Over Segmentation, and Under Segmentation for IoU of 0.50 and 0.75 on Test Set B.)

Detection methods	Recall (0.50 / 0.75)	Precision (0.50 / 0.75)	F1 (0.50 / 0.75)	Correct lines seg.	Over seg.	Under seg.
DBNet	0.90 / 0.58	0.95 / 0.61	0.93 / 0.59	0.88	0.01	0.11
DBNet++	0.88 / 0.78	0.92 / 0.81	0.90 / 0.80	0.88	0.03	0.09
CRAFT	0.61 / 0.59	0.77 / 0.75	0.68 / 0.66	0.54	0.01	0.45
CRAFT+l.s. w/o p.p	0.49 / 0.46	0.69 / 0.65	0.57 / 0.53	0.43	0.01	0.56
CRAFT+l.s w/ p.p	0.98 / 0.87	0.95 / 0.84	0.97 / 0.85	0.90	0.09	0.01

Table 7.3 further elucidates the recall, precision, F1 score, and segmentation accuracies for Test Set B. It is evident that a line spacing ratio of -0.1 leads to predominantly under-segmented results, with multiple lines of text often being misinterpreted as a single

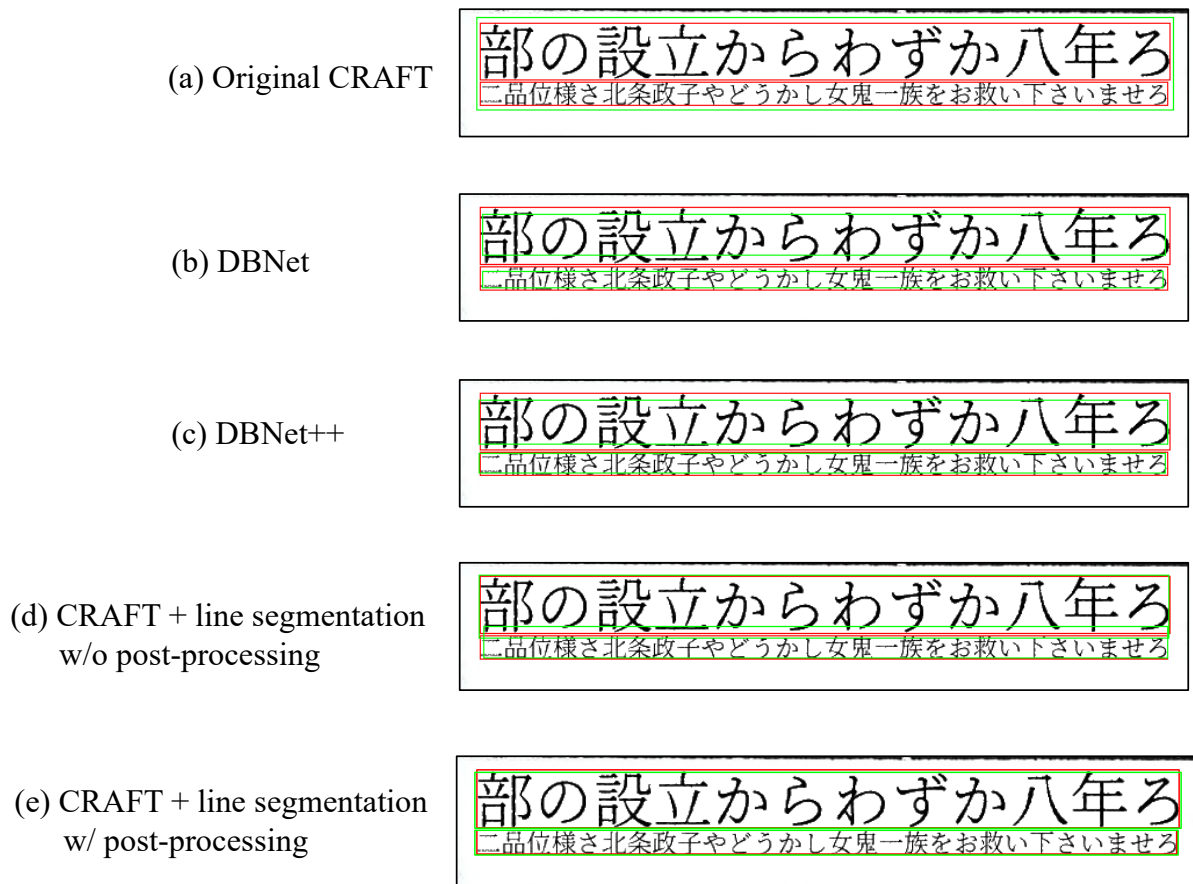


Figure 7.9: Examples of text region detection for each model on font text.

block. Contrastingly, at a line spacing ratio of 0.0 which shown in table 7.4, where text is distinctly separable, the CRAFT model with our post-processing method outshines others in accurately detecting multiple text lines. Additionally, our method exhibits the lowest under-segmentation rate, reinforcing its robustness in text line detection. Comparisons between IoU values of 0.50 and 0.75 highlight our method’s superior performance across most metrics, save for over-segmentation, where it is slightly more susceptible due to the CRAFT model’s heightened sensitivity to character heatmaps.

These insights underscore the efficacy of our proposed single-line text detection method in line segmentation detection, marking a significant advancement in the preprocessing phase of character recognition. While it may not demonstrate a dramatic improvement in text detection accuracy over traditional methods, its contribution to enhancing OCR accuracy is undeniable.

7.8 Summary

This chapter summarizes the key advancements and findings of our research on enhancing the Character Region Awareness For Text (CRAFT) model for optical character recogni-

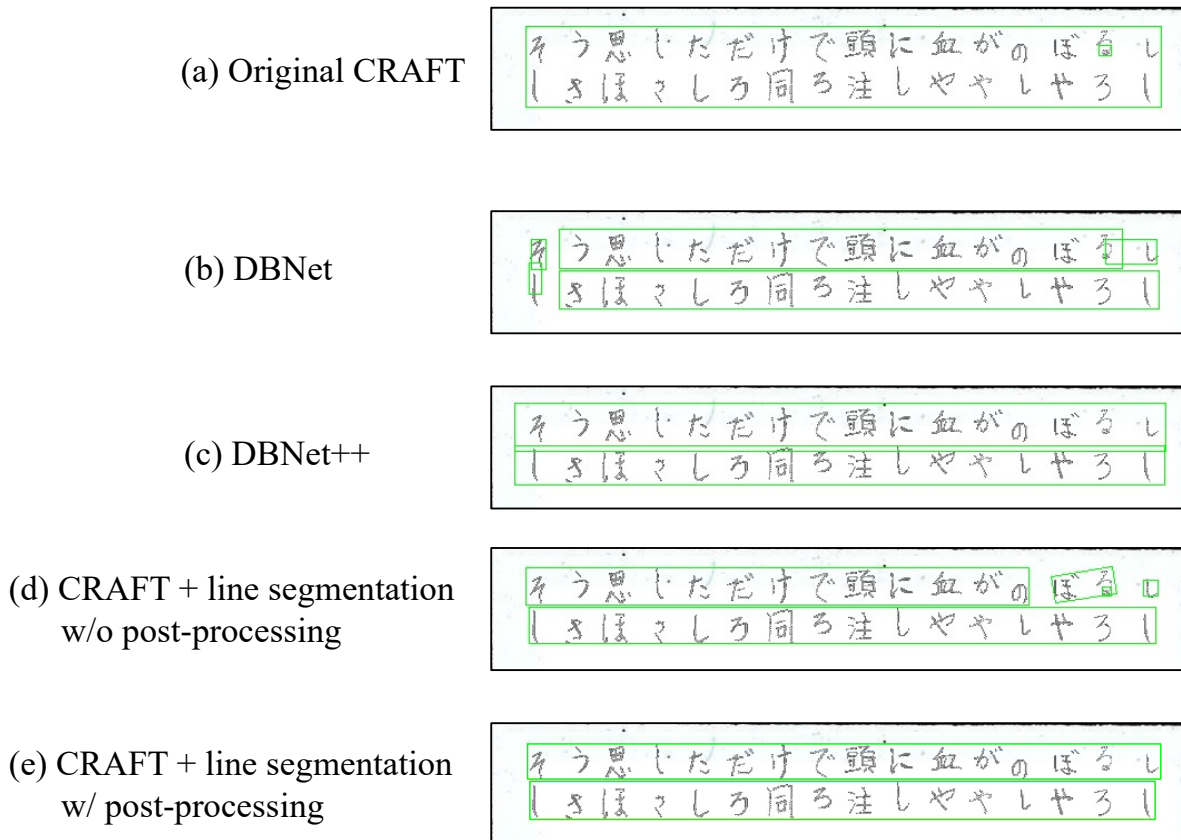


Figure 7.10: Examples of text region detection for each model on handwritten texts.

tion (OCR) systems. Our primary contribution lies in the integration of a line segmentation branch into the original CRAFT framework, significantly improving the accuracy of single-line text detection in multi-line texts. This enhancement enables the model to handle complex textual layouts with narrow line spacings more effectively.

The enhanced CRAFT model architecture, as shown in Figure 7.1, synergistically estimates region scores, affinity scores, and line segmentation. This integrated approach is critical in capturing the nuances of line-based text structures, thereby augmenting the model’s capabilities, especially in handling multi-line texts. The region and affinity scores, crucial for character detection, have been modified to adapt to the varying shapes and sizes of characters, enhancing the model’s precision in text detection tasks. Our post-processing methodology, detailed in Figure 7.4, combines the estimated single-line text region from line segmentation with the character region outputs from the original CRAFT model. This step is vital in generating optimally cropped images for OCR processing, particularly in documents where text lines are demarcated by clear horizontal lines. The loss functions employed, including Mean Squared Error (MSE) and Binary Focal Loss, play a pivotal role in refining the model’s accuracy for text detection tasks. These functions ensure that the model is fine-tuned to precisely detect text regions and connections.

Our experimental setup, outlined in Section 7.6, utilized a unique dataset comprising

a mix of font and handwritten texts from public document forms and the ETL Character Database. The evaluation metrics, including recall, precision, and F1 score, were applied at Intersection over Union (IoU) values of 0.50 and 0.75, demonstrating the effectiveness of our proposed method in single-line text detection and OCR accuracy. Tables 7.1, 7.2, 7.3, 7.4, and 7.5 present a comprehensive comparison of our method against other models, highlighting its superiority in handling various text layouts and line spacings.

In conclusion, our research presents a significant leap in the field of text detection and character recognition. The enhanced CRAFT model, with its novel line segmentation branch and efficient post-processing technique, offers a robust solution for accurately detecting text in multi-line documents, paving the way for more effective OCR systems in the future.

Chapter 8

Text Recognition Model

TrOCR represents a significant advancement in OCR by incorporating Transformer models (discussed in Chapter 3) to excel at recognizing text, particularly in single-line text image. This chapter explores the intricate design of TrOCR and its effectiveness in managing text recognition challenges. This chapter also reexamines the Hiragana and Katakana, and also Kanji generated images by Y-AE from Chapter 6 by randomly generating single-line text image for training the TrOCR model. This chapter also introduce a pre-processing method for the input images of ViT feature extractor of TrOCR and evaluating it's performance on multiple-lines text images.

8.1 Model Architecture

TrOCR combines the strengths of Transformer technology to set a new standard in OCR. Its architecture, shown in Figure 8.1, includes a ViT as image encoder for extracting features and a sequence Transformer for decoding text. The ViT encoder splits text images into patches and multiplies with positional embeddings, which then feeds them into a Transformer encoder. This step is crucial for identifying key features and patterns in the text. The sequence Transformer decodes these features into textual information, leveraging a self-attention mechanism to accurately interpret sequences and contextual relationships, thereby ensuring reliable text recognition.

During its training phase, TrOCR uses markers such as `<s>` and `</s>` for the beginning and end of text sequences, respectively, and `<PAD>` for padding. A key training strategy involves shifting the model's predictions to match target sequences, vital for correctly ending sequences. The model applies a "Label Smoother" for computing loss, blending label smoothing with its outputs. The loss function, as follows:

$$\text{Loss} = (1 - \epsilon) \times \text{nll_loss} + \epsilon \times \text{smoothed_loss} \quad (8.1)$$

This equation 8.1 uses ϵ to balance the negative log likelihood loss (`nll_loss`) and a smoothed loss across the vocabulary, enhancing prediction confidence. This approach

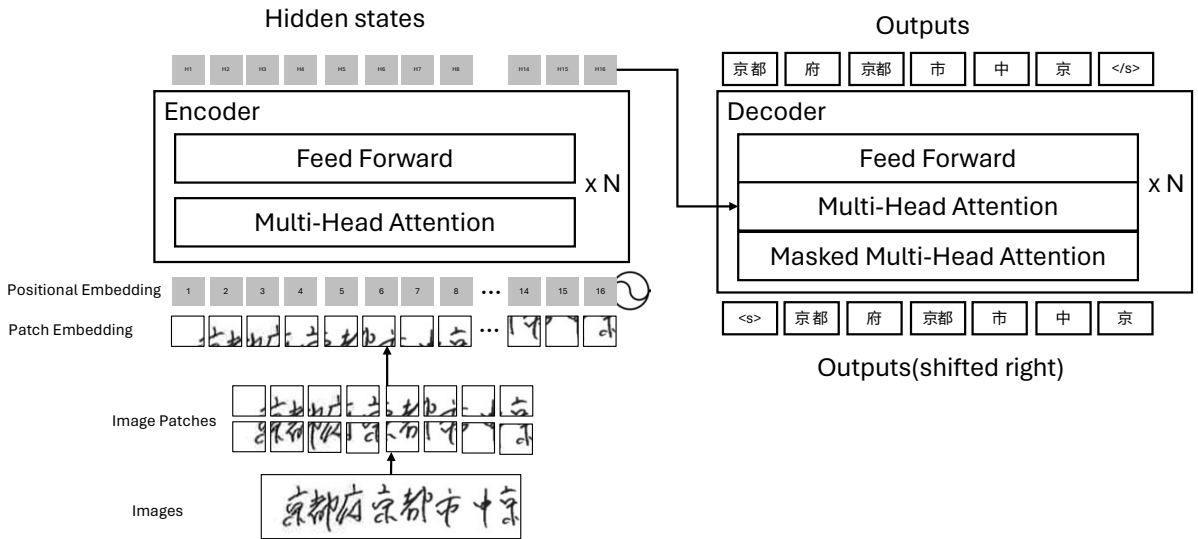


Figure 8.1: TrOCR model architecture

enables TrOCR to handle both single and multi-line text effectively, ensuring its flexibility for different OCR tasks, with a focus on shifted outputs and balanced loss.

8.2 Single Line and Multiple Line Text Recognition

TrOCR excels at processing single-line text, commonly found in names, signs, and number plates. It utilizes the Transformer’s attention mechanism to focus sequentially on each character, ensuring accurate recognition across different text types and styles. The synthesis of single-line text images starts with selecting characters from a database, as illustrated in Figure 8.2, and combining them to form a coherent single-line image, ensuring precise character depiction and alignment. Moreover, TrOCR’s capability extends to multi-line text processing, adeptly managing line breaks and spacing, ideal for complex documents and signage. Its comprehension of context and character interrelations across lines boosts its accuracy in such situations. Figure 8.3 shows the preprocessing method used in the original paper [65] and the proposed preprocessing method in this thesis. In the original TrOCR paper, it directly resizes images to 384×384 pixels, which potentially distorting character appearance. To prevent this, this thesis suggests resizing text lines to a fixed height of 32 pixels, concatenating them into a single long line, then splitting this line into sections of 384 pixels width, and stacking these sections vertically for processing. This method aims to preserve character integrity for improved recognition accuracy.

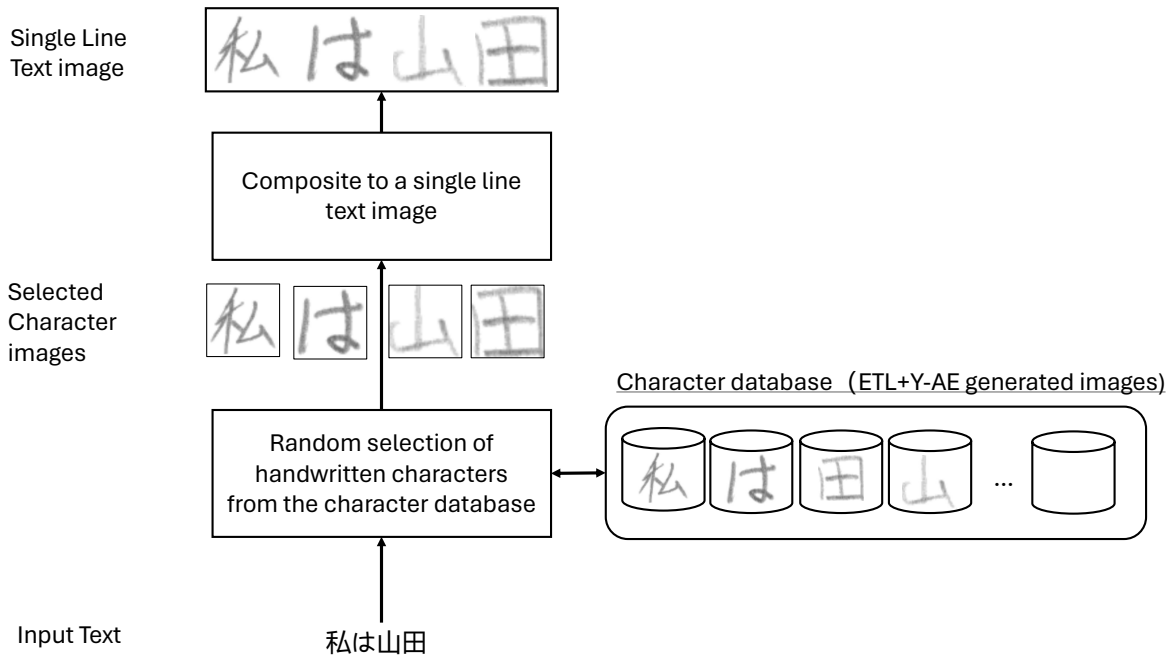


Figure 8.2: Single line image synthesis

8.3 Experiment

8.3.1 Pre-Training of TrOCR

TrOCR is a model known for benefiting from pre-training. For this thesis, pre-training data was created by asking 200 individuals to write sentences randomly chosen from Wikipedia, formatted on A4 paper. The specifics of this pre-training dataset are detailed in Table 8.1. This dataset includes 25,471 text lines and 239,169 characters, with 4,470 different character types. The validation set comprises 2,422 text lines, with 1,967 character types and a total of 22,724 characters. In this thesis, we fine-tuned a Japanese pre-trained model using the TrOCR English pre-trained model available through Microsoft Huggingface ¹.

Table 8.1: The statistics of pre-training handwritten text line images

Dataset	Number of Data	Number of Character Types	Number of Characters
Training	25,471	4,470	239,169
Evaluation	2,422	1,967	22,724

¹<https://huggingface.co/microsoft/trocr-base-stage1>

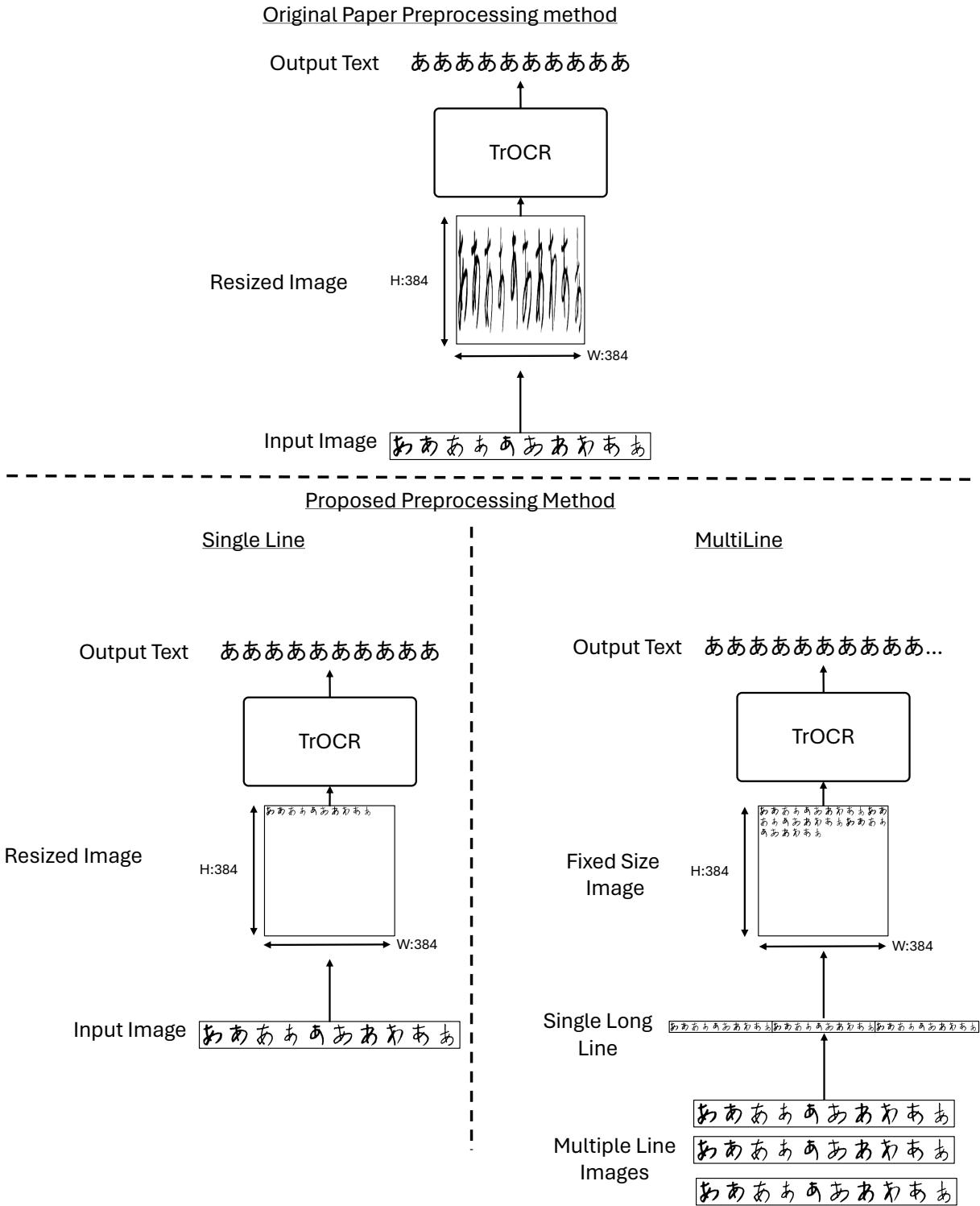


Figure 8.3: TrOCR model original paper preprocess method and this thesis proposed method

8.3.2 Single Line Image Generation with Y-AE Generated Images

As discussed in Chapter 6, Y-AE was used to create a total of 37,199,046 Kanji images. In this experiment, alongside ETL9, images generated by Y-AE served as a character image

database for producing both single and multiline text images, as described in Section 8.2. These images were then utilized to train the TrOCR model. The TrOCR model is trained with the default settings of huggingface’s Seq2SeqTrainingArguments² with a modification of batch size to 8.

8.4 Result

8.4.1 Pretraining Results

Our experiment revealed distinct differences in the performance of the two TrOCR training strategies. Figure 8.4 demonstrates an overview of the trends of losses. Initiating training from the ground up led to rapid stabilization of the training loss, suggesting an early ceiling in the learning process from the dataset. Remarkably, this method yielded a Character Error Rate (CER) of 100%, showcasing the model’s complete failure to correctly identify characters when trained from scratch, raising questions about the feasibility of starting training anew for sophisticated character recognition challenges. On the other hand, there was a gradual reduction in loss, suggesting continued improvement, when training began with the pre-trained model microsoft/trocr-base-stage1. This improvement aligns with the CER trends shown in Figure 8.5, where we see the CER methodically dropping to 6.54%. Such a decrease in CER highlights the fine-tuned model’s growing precision in recognizing characters, especially those in Japanese text, showcasing the advantages of building upon a pretrained model for complex languages. Based on these findings, we chose to use the fine-tuned model for further testing on the ETL and Y-AE generated image sets.

8.4.2 Single-line Training Result

Hiragana and Katakana Training Result

In Chapter 7, the character-by-character classifier improved the accuracy by approximately 47.4% maximum. In this chapter, we adapted a line-by-line classifier with Hiragana and Katakana characters by synthesis Hiragana and Katakana single line text images and examined the character error rates (CERs). Table 8.2 provides a detailed analysis of the CERs for single-line Hiragana and Katakana text recognition, highlighting the influence of data augmentation (DA) and the integration of Y-AE generated images (GIs). The table illustrates that all models maintain CERs below 10%, affirming the effectiveness of the models in handling the validation data, which includes both ETL and ETL+GIs with randomly generated images.

An important finding is the uniform CER of 7.25% in models (2) and (4), where DA was applied, regardless of the dataset composition (ETL-only or ETL+GIs). This

²https://huggingface.co/docs/transformers/main_classes/trainer

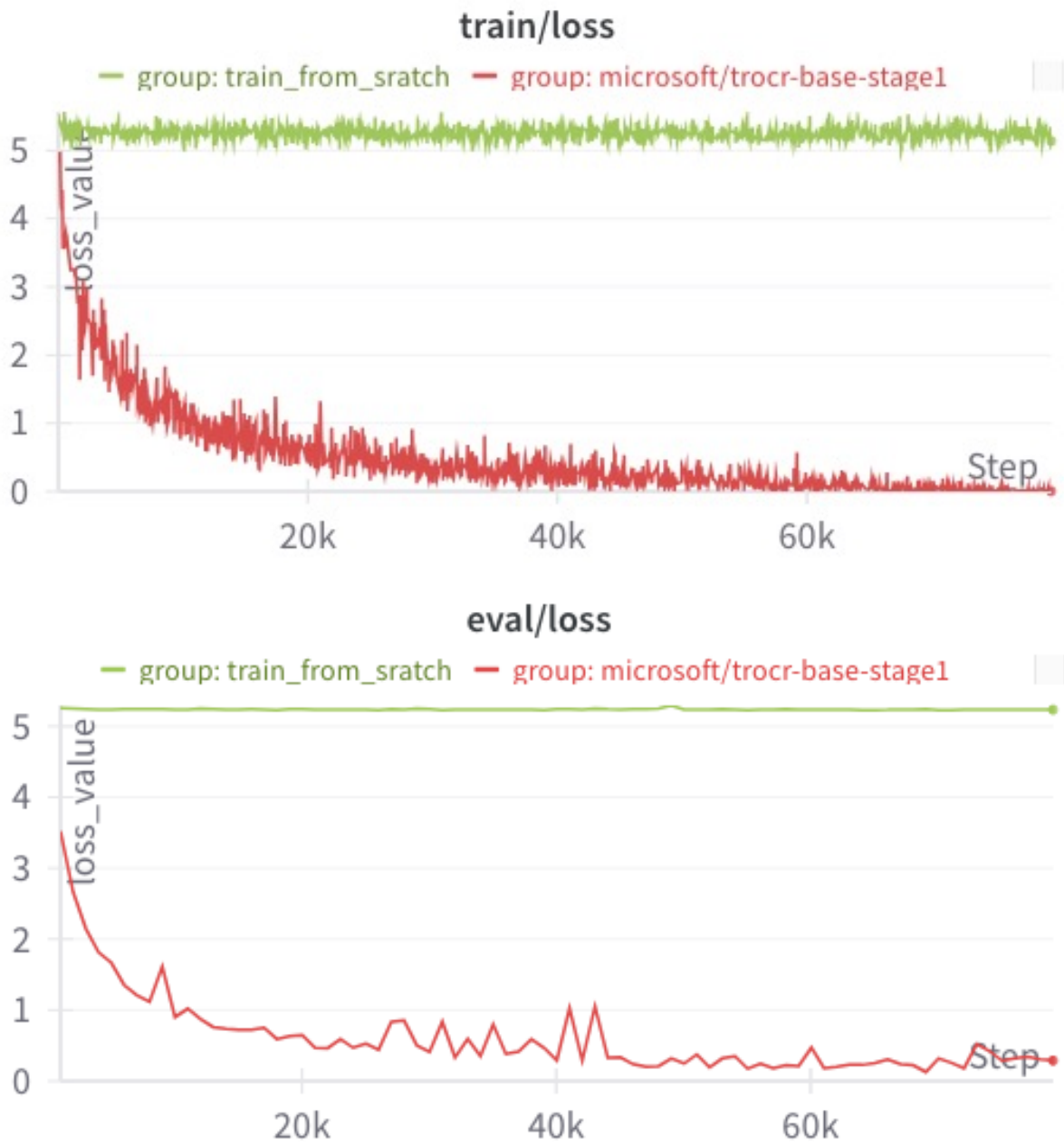


Figure 8.4: TrOCR model pre-training losses graph

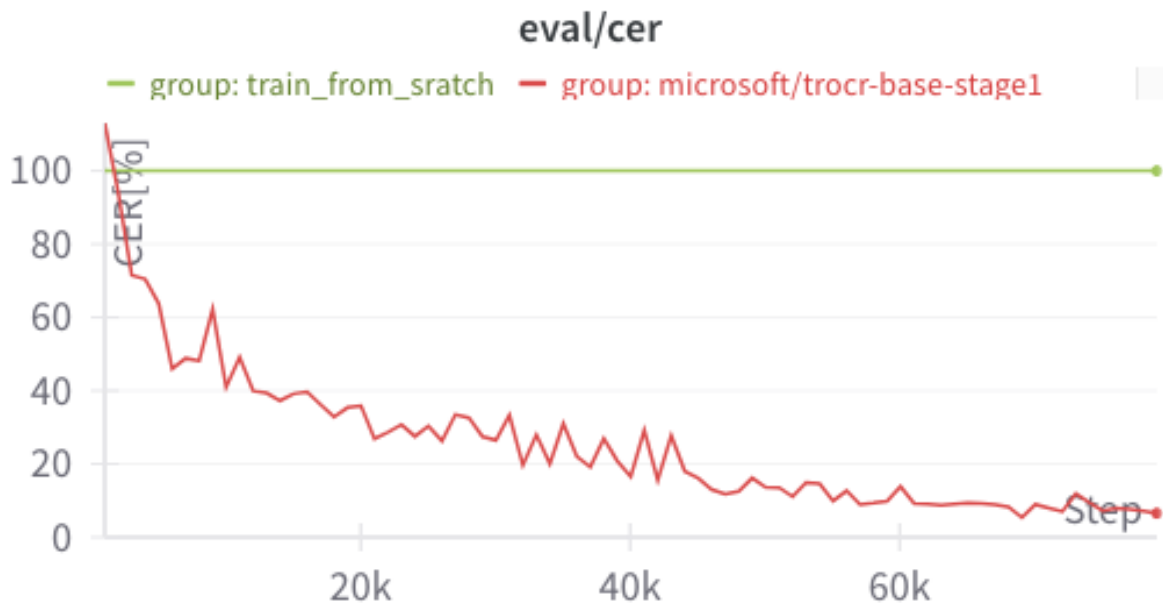


Figure 8.5: TrOCR model pre-trained model character error rates graph

Table 8.2: Character error rates of single-line Hiragana and Katakana text images with randomly generated(RG) image, \checkmark : DA is applied, \times : DA is not applied

Model no.	Dataset description	Validation Data	DA	CERs
(1)	ETL only (baseline)	RG with ETL only	\times	4.03
(2)	ETL only	RG with ETL only	\checkmark	7.25
(3)	ETL + GIs	RG with ETL+GIs	\times	4.32
(4)	ETL + GIs	RG with ETL+GIs	\checkmark	7.25

uniformity suggests that the complexities added by DA, through varied character styles and forms, create a more challenging recognition task. Models without DA, (1) and (3), show lower CERs, 4.03% and 4.32% respectively, with the baseline model (1), trained just on ETL data without DA, achieving the lowest CER. This result indicates a strong ability to recognize characters in simpler scenarios. The minor increase in error rate for Model (3), incorporating GIs, suggests that while synthetic data adds complexity, it does not drastically affect character recognition capability. This outcome is encouraging, indicating that models can still effectively recognize characters with the addition of GIs. It also hints at the potential benefit of combining GIs with more realistic, image-based DA techniques to possibly reduce CERs further.

Overall, these results highlight the trade-off between adding diversity through augmentation and preserving model accuracy. Although DA adds complexity, making it harder for models to recognize validation images, incorporating GIs during training shows promise. These findings suggest that strategic data augmentation, especially those mimicking real

image conditions, could enhance OCR models’ ability to recognize a wide array of Hiragana and Katakana characters.

Single-line Kanji Training Result

Table 8.3 compares the CERs for single-line Kanji text recognition, showing results from various models and datasets. In the ETL-only dataset (Models (1) and (2)), the CERs are observed at 15.53% without data augmentation (DA) and slightly higher at 17.09% with DA. This indicates that DA in this scenario does not significantly improve recognition accuracy. A notable outcome is the substantial increase in CER to 99.41% for Model (3), which combines the ETL dataset with Y-AE generated images (GIs) and does not employ DA. This dramatic increase in error rate is indicative of an actual loss explosion in the model, suggesting that the inclusion of Y-AE generated images complicates the recognition process to a great extent. This loss explosion could be due to a potential mismatch in data distribution between the real ETL dataset and the synthetic GIs, possibly leading to overfitting or ineffective training which illustrated in Figure 8.7 train losses and evaluation losses. The complexity of Kanji characters, coupled with potentially flawed or non-diverse synthetic data, might have exacerbated the training challenges.

Table 8.3: Character error rates of single line Kanji text images with RG image (DA only applies on ETL or Y-AE based single line synthesis), ✓: DA is applied, ✗: DA is not applied

Model no.	Dataset description	Validation Data	DA	CERs
(1)	ETL only (baseline)	RG with ETL only	✗	15.53
(2)	ETL only	RG with ETL only	✓	17.09
(3)	ETL + GIs	RG with ETL+GIs	✗	99.41
(4)	ETL + GIs	RG with ETL+GIs	✓	17.19
(5)	Pre-training Dataset and ETL	Pre-training validation dataset	✗	8.61
(6)	Pre-training Dataset and ETL	Pre-training validation dataset	✓	9.93
(7)	Pre-training Dataset with ETL and GIs	Pre-training validation dataset	✗	N/A
(8)	Pre-training Dataset with ETL and GIs	Pre-training validation dataset	✓	8.57

When DA is applied in Model (4) (ETL + GIs), the CER is reduced to 17.19%, aligning it more closely with the results seen in the ETL-only dataset with DA. This improvement could be attributed to the regularization effect of DA. By introducing variability in the training images, DA might have mitigated the overfitting on specific characteristics of the

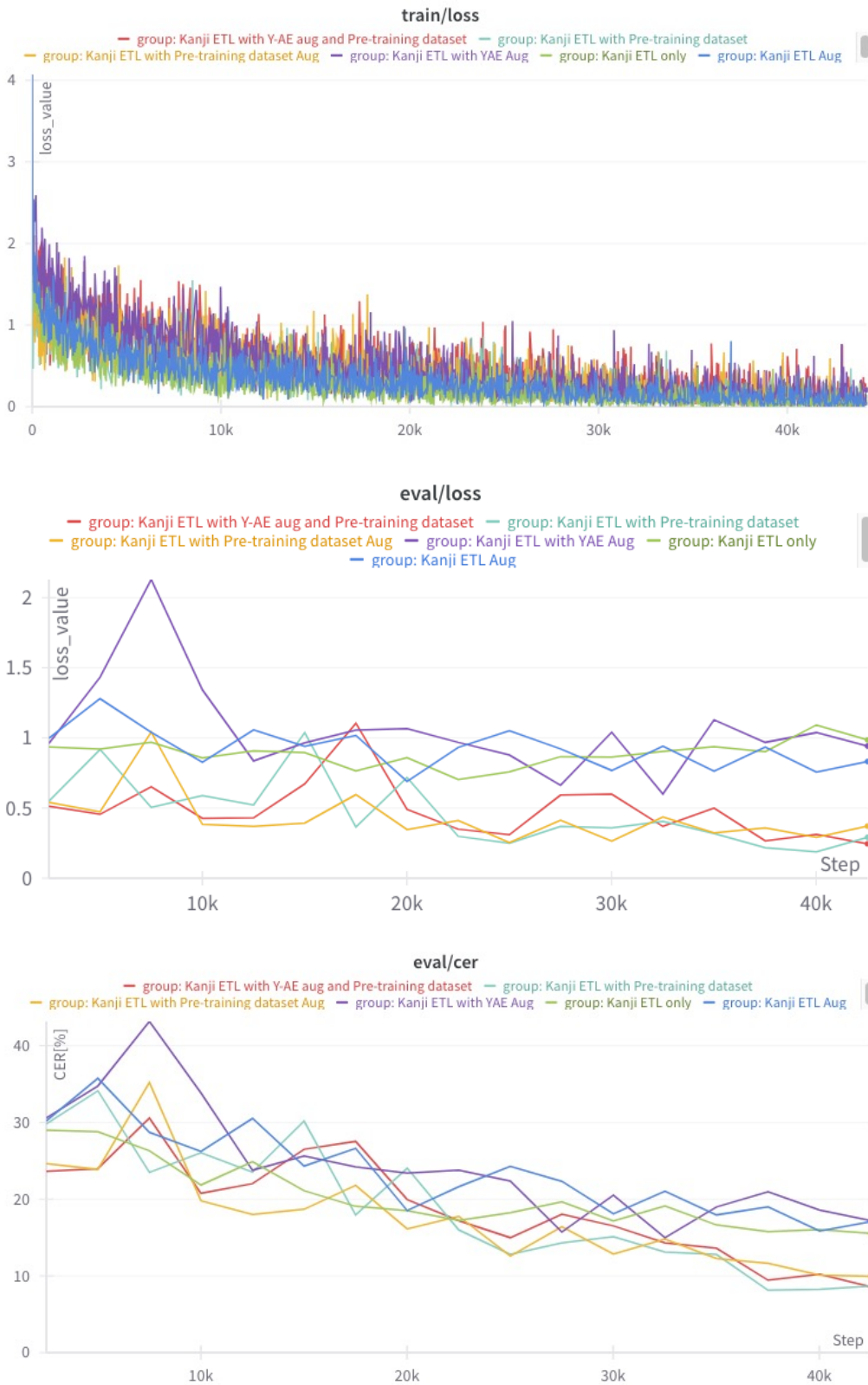


Figure 8.6: TrOCR model Kanji single line training, eval losses and eval cers graphs without loss explosion

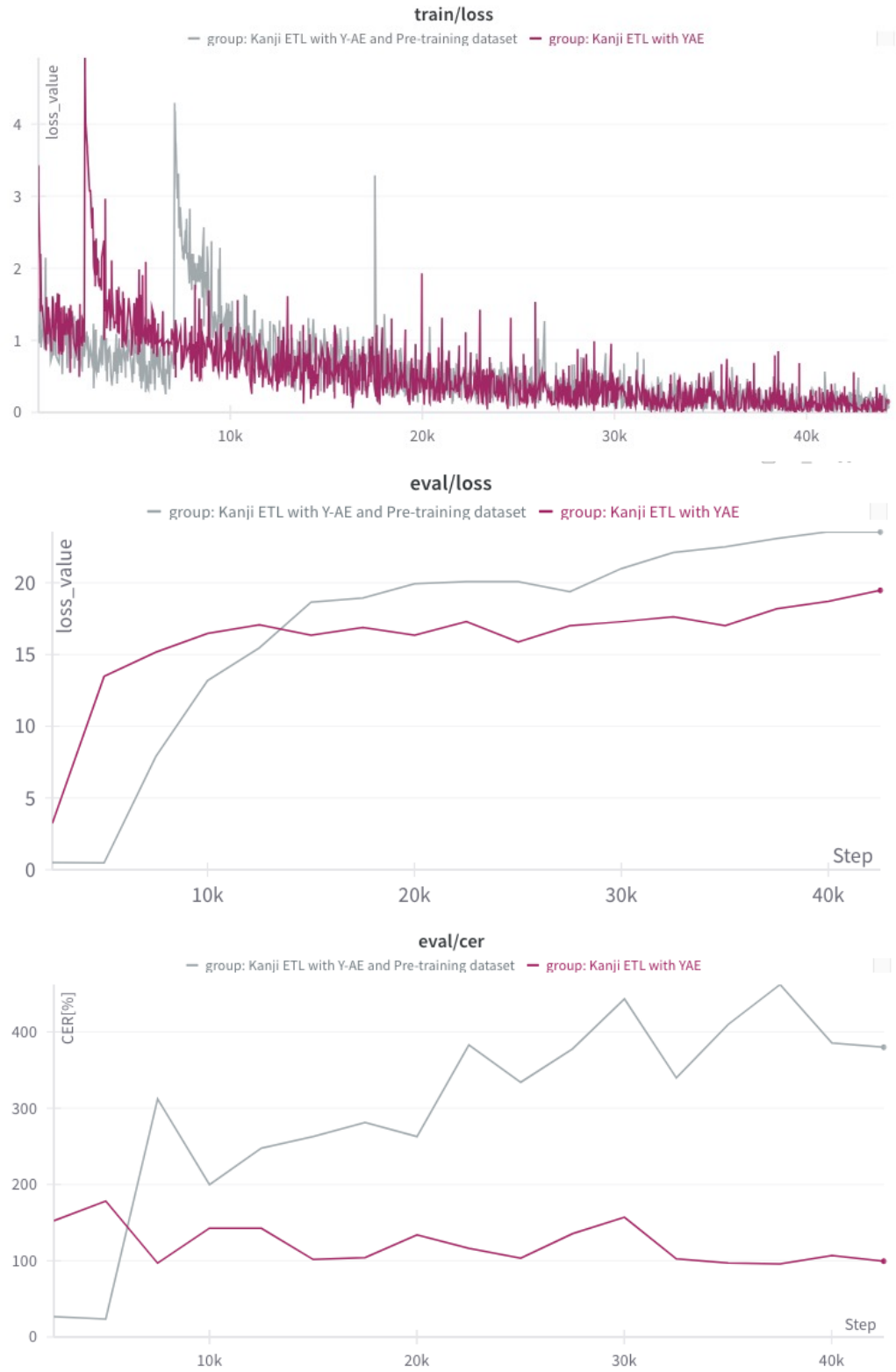


Figure 8.7: TrOCR model Kanji single line training, eval losses and eval cers graphs with loss explosion

training data, particularly the synthetic GIs. This could explain why the loss explosion was avoided in Model (4), as DA provided a more generalized training experience for the model. This is evident from the stabilized training and evaluation losses and CERs shown in Figure 8.6.

In contrast, Models (5) and (6), which utilize a pre-training dataset alongside ETL, show much lower CERs. This pre-training dataset comprises real handwritten single-line text images, contributing to CERs of 8.61% without DA and 9.93% with DA. The presence of genuine handwritten text in the pre-training dataset evidently provides a more authentic and challenging training environment, leading to significantly better model performance and lower error rates. Notably, Models (7) and (8), which leverage both the pre-training dataset and Y-AE generated images alongside ETL, are particularly insightful. Model (7) experiences a loss explosion, as indicated by its missing CER value, reaffirming the complications introduced by the integration of Y-AE GIs. On the other hand, Model (8), with DA applied, shows a particularly low CER of 8.57%. This indicates that while synthetic GIs can lead to detrimental effects such as loss explosion, as seen in Model (7), the strategic application of DA and the inclusion of a diverse pre-training dataset can significantly mitigate these issues and enhance the model’s performance.

These findings underscore the crucial impact of training dataset composition on the effectiveness of character recognition models. The severe loss explosion seen with the inclusion of Y-AE GIs demonstrates the complexities involved in integrating synthetic data into OCR models. Conversely, the real-world handwritten images in the pre-training dataset enhance model performance, highlighting the importance of high-quality, realistic training data in OCR applications.

8.4.3 Multiple-lines Training Result

Table 8.4 shows the comparison of CERs for multiple-line Kanji text recognition. For Models (1) and (2), which utilize the ETL dataset exclusively, CERs are recorded at 10.46% and 12.45%, respectively. Intriguingly, the implementation of Data Augmentation (DA) in Model (2) leads to an elevated CER, contradicting the anticipated performance boost. This reversal implies that the addition of variability through DA might not uniformly benefit complex recognition tasks, as illustrated in Figure 8.8. Here, while the training losses remain stable, the evaluation losses exhibit an unexpected and gradual rise, hinting at a discrepancy between the model’s performance during training and its ability to generalize to new data.

Referring to the insights from Table 8.3, which highlighted a significant loss explosion in single-line Kanji text recognition models, a similar trend is evident in the context of multiple-line text recognition as shown in Table 8.4, particularly noticeable in Models (3) and (4). The absence of CERs for these models, indicated by a (“N/A”) in the table, signals a profound disruption in the training process attributed to a loss explosion. This outcome, graphically illustrated in Figure 8.9, emphasizes the complexities introduced by

Table 8.4: Character error rates of multiple-lines (Includes Kanji text images by using randomly generated(RG) from single line text images, \checkmark : DA is applied, \times : DA is not applied

Model no.	Dataset description	Validation Data	DA	CERs
(1)	ETL only (baseline)	RG with ETL only	\times	10.46
(2)	ETL only	RG with ETL only	\checkmark	12.45
(3)	ETL + GIs	RG with ETL + GIs	\times	N/A
(4)	ETL + GIs	RG with ETL + GIs	\checkmark	N/A
(5)	Pre-training Dataset	RG with pre-training validation dataset	N/A	24.88

integrating synthetic Y-AE generated images (GIs) with multiple lines of text. The visual representation of training and evaluation losses, especially the noticeable and gradual increase in evaluation losses, further confirms the occurrence of an evaluation phase loss explosion. This not only echoes the complications observed in single-line recognition but also escalates them, highlighting the increased complexities and unpredictabilities when managing broader text arrangements. The combination of synthetic data with real text scenarios seems to push the models beyond their learning thresholds, resulting in unpredictable and unstable training outcomes. The loss explosion in multiple-line text recognition models underlines the necessity of a careful and thoughtful integration of GIs data as well as in training a single-line recognition model. Contrarily, Model (5) in the multiple-line text recognition scenario, which uses a pre-training dataset comprising real-world, handwritten text images, presents a CER of 24.88%. This high error rate reflects the inherent difficulty in recognizing diverse and complex real-world text layouts, a challenge significantly different from that presented by single-line text images.

These insights emphasize the critical role of dataset composition in the performance of OCR models, especially in complex tasks like multiple-line text recognition. The integration of synthetic data, such as Y-AE GIs, needs to be managed with precision to avoid destabilizing the model’s training process. At the same time, the higher error rates with real-world text scenarios in the pre-training dataset highlight the importance of including diverse and high-quality training data for effective OCR application, demonstrating the potential of advanced OCR technologies like TrOCR in addressing these challenges.

8.5 Conclusion

The incorporation of Y-AE generated data, detailed in Chapter 6, into the training process of TrOCR has been a pivotal focus of this chapter. While the Y-AE generated images have shown potential in enhancing the diversity of training datasets, their integration has also presented challenges, particularly in the context of multiple-line text recognition.



Figure 8.8: TrOCR model multiple line training, eval losses and eval cers graphs without loss explosion

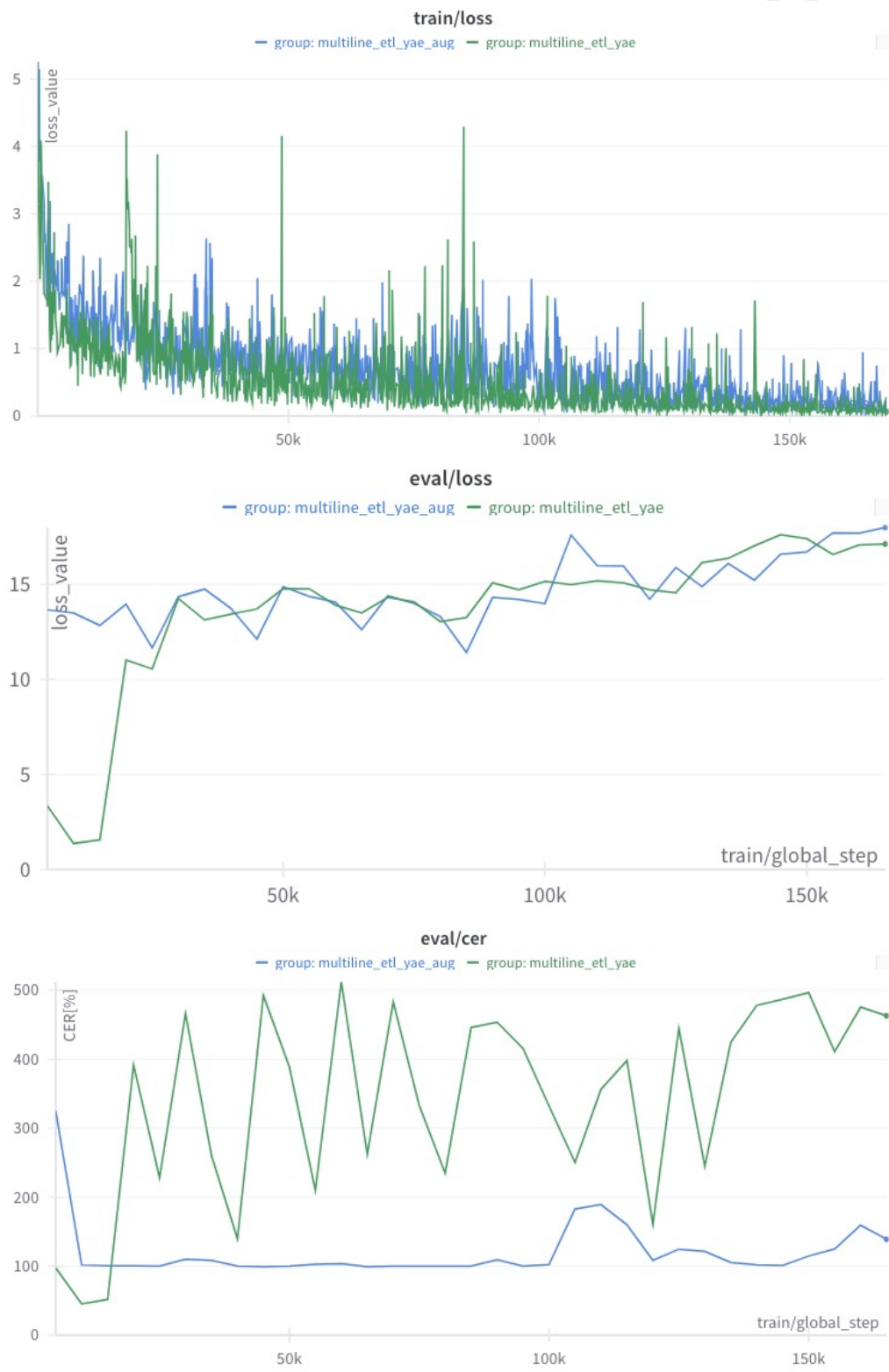


Figure 8.9: TrOCR model multiple line training, eval losses and eval cers graphs with loss explosion

The occurrence of loss explosions in models trained with Y-AE generated images underscores the complexities involved in synthesizing and utilizing synthetic data effectively. The experiments conducted in this chapter have revealed significant insights into the performance of TrOCR in various scenarios. The fine-tuning of the TrOCR model from an advanced stage model, as opposed to training from scratch, has proven to be more effective, especially in the context of Japanese text recognition. This approach has been instrumental in achieving lower Character Error Rates (CERs), highlighting the model's adaptability and learning capacity.

In single-line text recognition, TrOCR has shown excellent results, particularly with Hiragana and Katakana characters. The model's ability to handle a wide range of character styles and patterns, even in the presence of data augmentation, is a clear indicator of its robustness. However, the integration of Y-AE generated images for Kanji character recognition has been more challenging, with significant increases in CERs observed in certain models. This outcome points to the need for a more nuanced approach to incorporating synthetic data into the training process. The multiple-lines text recognition experiments have further demonstrated the complexities of working with diverse and realistic text layouts. While the pre-training dataset comprising real-world handwritten texts has resulted in higher CERs, it has also provided a more authentic and challenging training environment, essential for developing effective OCR models.

In conclusion, the exploration of TrOCR in this chapter has underscored the importance of sophisticated training methodologies, careful data integration, and the utilization of advanced Transformer models in OCR. The insights gained from this research provide a valuable foundation for further advancements in OCR technology, with potential applications in various fields requiring efficient and accurate text recognition.

8.6 Summary

The capabilities of TrOCR, a transformative OCR system utilizing Transformer technology, are highlighted in this chapter. Its architectural design effectively addresses the challenges in recognizing both single-line and multiple-line texts. Experiments and results demonstrate that while TrOCR excels in handling diverse textual formats, the integration of synthetic data such as Y-AE generated images requires careful consideration. The model shows promise in adapting to complex multi-line text scenarios, especially when trained with diverse and high-quality datasets. This exploration of TrOCR's performance underscores the importance of strategic dataset composition and training methodologies in advancing OCR technologies, particularly for applications requiring nuanced text recognition capabilities.

Chapter 9

Summary and Future Works

This thesis undertakes an extensive examination of OCR and HTR, charting their progression from conventional techniques to contemporary methods based on Deep Learning.

Chapter 1 provides an overview of the thesis’s research background, delves into related works by other researchers, and outlines the research objectives and scope. Additionally, it highlights the significant contributions made by this thesis. The chapter concludes by detailing the structure and organization of the thesis.

Chapter 2 of the thesis delves into the historical evolution and current state of Text Image Recognition, with a particular emphasis on OCR and HTR. It traces the journey of textual representation from ancient manuscripts through the digital age, highlighting the significant role of OCR in revolutionizing data processing. The chapter outlines the origins of OCR, its evolution from early pattern-matching techniques to advanced deep learning methods, and discusses the unique challenges and advancements in HTR. It covers both offline and online HTR, highlighting their applications and the impact of deep learning in enhancing accuracy. The chapter concludes by summarizing the transformative journey of text image recognition and its profound impact on various sectors in the digital era.

Chapter 3 of the text explores deep learning, a subset of machine learning and AI, drawing inspiration from the human brain’s structure. It traces the historical development of neural networks, highlighting contributions by Geoffrey Hinton, Yann LeCun, and Yoshua Bengio. The chapter outlines deep neural networks’ structure and functioning, focusing on their layers, computation, and training processes, including backpropagation and optimization techniques like SGD, Adam, and RMSprop. It emphasizes CNNs for their role in computer vision and introduces the Transformer model architecture, pivotal in sequence-to-sequence tasks, alongside ViT for image classification. The chapter concludes by summarizing deep learning’s impact and potential in various AI domains, noting its role in advancing CNNs and transformers.

Chapter 4 provides an insightful overview of advancements in Text and Character Detection in OCR, tracing the evolution from traditional methods to sophisticated deep learning models. It emphasizes the challenges of character localization in diverse contexts and highlights key developments such as the DBNet for complex scene text detection,

DBNet++ with enhanced features for varied text sizes and backgrounds, and the CRAFT model, which specializes in individual character segmentation. The chapter underscores the significant contributions of these models in advancing OCR technology and their broad applicability across different domains.

Chapter 5 of the thesis explores the application of deep learning in image generation, focusing on autoencoders and the advanced Y-AE architecture. It begins by introducing autoencoders, a type of neural network for unsupervised learning developed by Geoffrey E. Hinton, primarily used for dimensionality reduction, feature learning, and denoising. The chapter explains the structure of an autoencoder, comprising an encoder to compress data into a latent space and a decoder for reconstructing the input from this compressed representation. The training process aims to minimize loss, usually the mean squared error, to capture essential data features and ignore noise. Autoencoders are notably effective in image processing and anomaly detection. The chapter then delves into the Y-AE, highlighting its unique dual-branch design optimized for different loss functions, which is particularly effective for style transfer and image reconstruction. This chapter provides a comprehensive understanding of autoencoders and their significant role in image reconstruction, leading to the Y-AE's innovative approach to precise image generation.

Chapter 6 present a novel approach to character generation for OCR systems using an enhanced Y-AE model integrated with AdaIN. This advanced model is pivotal in generating a wide array of character images from a limited amount of training data, significantly enriching OCR training datasets. The chapter meticulously outlines the model's architecture, its adaptation from previous research, and the innovative application of the AdaIN layer. The model demonstrates remarkable versatility in producing diverse character styles, which is crucial for robust character recognition in OCR systems. However, the utility of the generated images varies, necessitating the implementation of sophisticated filtering methods, such as MSE-based and classifier-based approaches, to exclude non-beneficial images. The effectiveness of this enhanced Y-AE model is validated through comprehensive evaluation experiments, revealing a substantial improvement in character recognition accuracy when combined with traditional data augmentation techniques. This significant advancement is not just in the model's ability to produce varied and realistic character styles, but also in its potential to transform OCR technology, making it more efficient and accurate in handling diverse character sets. The chapter's findings underscore the potential of advanced machine learning techniques in overcoming the limitations of traditional OCR systems, marking a notable contribution to the field.

Chapter 7 of the thesis presents a significant enhancement to the CRAFT model for OCR. It introduces a line segmentation branch to improve single-line text detection in multi-line documents. This enhancement allows for more effective handling of complex text layouts, especially those with narrow line spacings. Key modifications include refined region and affinity scores, tailored for various character shapes and sizes. A novel post-processing method combines line segmentation with character region outputs, optimizing images for OCR processing. The model's improved accuracy in text detection

is demonstrated through comprehensive evaluations using a unique dataset of font and handwritten texts, establishing it as a robust solution for advanced OCR applications.

Chapter 8 delves into TrOCR, a state-of-the-art OCR system employing Transformer models, particularly effective for single-line text recognition. The chapter explores TrOCR’s sophisticated architecture, which includes a vision Transformer for feature extraction and a sequence Transformer for decoding text. This dual-component setup allows TrOCR to process both single and multiple lines of text adeptly. The model’s training involves unique strategies like label smoothing in loss calculation, enhancing its text recognition accuracy. Experiments demonstrate TrOCR’s proficiency with single-line texts and reveal the complexities of integrating Y-AE generated images for multiple-line text recognition. This integration, while enriching the training dataset, presents challenges, indicating the need for cautious synthetic data utilization. The chapter highlights TrOCR’s adaptability to complex text layouts and the importance of quality training data in developing effective OCR models, underscoring its potential in diverse OCR applications.

Future work can expand on the current study’s findings in OCR and HTR by refining Transformer models, especially TrOCR, for better multi-line text recognition accuracy. This may include advanced methods for integrating Y-AE generated images with a more proper data augmentation method that can synthesis realistic text image, into model training. Exploring the fusion of OCR with other AI fields, like natural language processing and semantic understanding, could lead to the creation of more context-aware and intelligent text recognition systems. Additionally, enhancing decoder-only Transformer models like Decoder-only TrOCR (DTrOCR) [105] or Kosmos-2.5 [106], to improve linguistic information integration with image data, is another promising direction. This approach aims to enhance accuracy, versatility, and computational efficiency in diverse language and text style applications.

Acknowledgment

I was born in Kuala Lumpur, the capital of Malaysia, a multi-racial country. When I was a child, I dreamed of becoming a scientist, but I gave up that dream once because I could not study well from elementary school to junior high school, and I lived my life being carried along by the people around me. However, after being exposed to Japanese anime and learning about Japanese culture, I choose to target to study abroad in Japan in the third year of junior high school and began to pursue my childhood dream of becoming a scientist once again. From there, my checkered journey began, and I began to study hard. It was around that time that I began to seriously study the Japanese language on my own, and when my aunt, who was living in the U.S., happened to return to Malaysia, we worked together to persuade my parents to allow me to study in Japan, which is why I am here in Japan today. I would like to express my deep and sincere gratitude to my aunt for persuading me to do so, and to my parents for allowing me to study in Japan, and for their constant support regarding my study in Japan.

I received a scholarship from the Japan Student Services Organization (JASSO) in 2014 to enroll in the Department of Mechatronics Engineering at the University of Yamanashi as an undergraduate student. As an undergraduate student, I studied mechanical, electrical and electronic, and information engineering, and I had to study various general education subjects in addition to my specialty, and at first I was worried that I would not do well. However, thanks to the Japanese friends I met and the opportunity to study with them, I was able to successfully earn credits and graduate from the undergraduate program. From the moment I decided to study in Japan, I wanted to build a robot with artificial intelligence. At that time, I was able to study algorithms more deeply in Professor Nishizaki's C language class, and I was very interested in his research on speech recognition on his official website. I remember vividly as if it was yesterday that I asked him directly what I needed to study in order to create an artificial intelligence robot, outside of class. I was in Nishizaki lab as a temporary member through curriculum, but I worked hard and became a member of Nishizaki lab and started to study in earnest for research in the field of AI. I have continued on to the master's program with a scholarship from the Ito International Foundation. With the understanding and support of Professor Nishizaki, I also continued my research on deep learning as well as speech recognition and noise reduction.

I successfully completed my master's degree in 2020 and went on to a doctoral program,

as well as starting a venture company with an acquaintance to produce AI-OCR software, which gave me the experience of running a company. Thanks to my research life in the laboratory, I was able to maximize my passion and talent in developing AI-OCR software in the venture company and also do the research about OCR. In addition to my own research, I was also able to get involved in a smart agriculture project that the University of Yamanashi is working on as a representative research institute, where I was able to do the research on AI robots that I wanted to do. I would like to express my sincere gratitude to Dr. Xiaoyang Mao, Professor, Director and Vice President of the University of Yamanashi, Dr. Koji Makino, Associate Professor of the University of Yamanashi Graduate School of Engineering, and Dr. Nishizaki, Professor of the University of Yamanashi Graduate School of Engineering, for their guidance and support in this process.

However, it was not smooth sailing, as it was very difficult to simultaneously develop software, run a company, and study for a doctoral course. Things did not go well at the company, and I once fell ill emotionally. At that time, Professor Nishizaki kindly listened to me even though he was very busy. Also, Assistant Professor Shinji Nishitani of the Counseling Support Office helped me a lot with counseling. I believe that it is thanks to these doctors that I am now back on my feet. I am deeply grateful for that. Although after that, I resigned from the company, and I am very grateful to the colleagues who worked with me.

This thesis signifies the culmination of my extensive research journey throughout my doctoral studies at the University of Yamanashi's Integrated Graduate School of Medicine, Engineering, and Agricultural Science, System Integration Engineering Course.

I extend my profoundest gratitude to Professor Hiromitsu Nishizaki, who has been an exceptional mentor and guide since my initial undergraduate year. His mentorship in the Nishizaki Laboratory has been a cornerstone of my academic progression, instilling in me the principles of rigorous research and professional excellence. His expert guidance on research methodologies and adept skills in professional communication have been instrumental in my evolution as a scholar.

I express my deep indebtedness to Professor Kazuho Ito, Professor Yoshimi Suzuki, Associate Professor Toshiya Kitamura, Professor Kenji Ozawa, and Professor Ryutarou Ohbuchi for their invaluable contributions as sub-examiners. Their critical insights and feedback have immensely contributed to the refinement and success of my doctoral thesis.

Special acknowledgments are due to my collaborative partners, Mr. Hideaki Yajima and Mr. Tomoki Kitagawa, for their steadfast support and collaboration in our mutual research pursuits. Their companionship and intellectual exchange have been pillars of my research journey.

My heartfelt thanks also go out to my seniors, contemporaries, and juniors in the laboratory. Their camaraderie, shared wisdom, and engaging discussions have immensely enriched my academic life. The collective experiences of challenges faced have been instrumental in shaping my research perspective.

As I reflect upon the remarkable seven years spent at the Nishizaki Laboratory, I

recognize it as a period of immense personal and professional growth. This journey, spanning from my undergraduate years through to my doctoral studies, has been a time of significant learning and discovery. The knowledge and experiences I have amassed during this period have been invaluable, and I am eternally grateful for the opportunities and growth it has afforded me. This chapter of my life has been a testament to the power of dedicated mentorship, collaborative research, and the relentless pursuit of academic excellence.

Last but not least, I would like to express my sincere gratitude again to all those who supported me during my study abroad in Japan, to my professors and teachers for their professional guidance, and to my friends and colleagues who gave me advice in difficult times. Life is not always goes well, but I will continue to remember those who have helped me, and I will do my best to live my life to the fullest, doing the best I can with what I have decided to do.

謝辞

私は多民族国家であるマレーシアの首都、クアラルンプールで生まれました。小さい頃は科学者になることを夢見ていましたが、小学校の頃から中学校まであまり勉強できなかった私は一度その夢を諦めたことがあり、周りに合わせ流される人生を歩んできました。しかし、日本のアニメに触れ、日本の文化などについて知り、中学3年生から日本留学を目指し始め、小さい頃の夢であった科学者になることをもう一度目指し始めました。そこから、私の波瀾万丈な旅が始まり、必死に勉強し始めました。その頃から日本語についても本格的に独学し始めて、アメリカに住んでいた叔母さんがたまたまマレーシアに帰国したときに、一緒に親を説得したお陰で、日本への留学を許されたことで、今こうして日本で留学できました。そのことについて一緒に説得してくださった叔母、また日本留学を許してくださった両親に、私の留學生活について絶え間ないサポートをしてくださったことについて、深く、深く感謝の意を表します。

私は2014年に独立行政法人日本学生支援機構から奨学金を受け、学部生として山梨大学のメカトロニクス工学科に入学しました。学部生のときは機械、電気電子、情報の3分野について勉強しており、専門以外にも様々な一般教養科目も勉強する必要があり、最初の頃はうまくやっていたのではないかと心配をしていました。しかし、日本人の友達に出会い、彼らと一緒に勉強できたお陰で無事単位を取ることができ、学部を卒業することができました。私は日本留学を決めたときから、人工知能搭載のロボットを作りたいと思いました。当時授業で西崎先生のC言語の授業でより深くアルゴリズムの勉強ができ、西崎先生のホームページで音声認識について研究をされているとても興味がありました。そこで、西崎先生に人工知能を作るためにどういった勉強必要なのか、授業以外に直接西崎先生に問い合わせみたが昨日のように鮮明に覚えています。西崎先生は丁寧に私にどういった勉強を教えてください、授業の一環で仮の形で西崎研究室にいましたが、勉強を頑張ることで西崎研究室の一員になり、AIの分野の研究に本格的に勉強し始めました。2018年に無事学士を取得し、伊藤国際交流財団から奨学金を受け引き続き修士課程に進学しました。また、西崎先生のご理解とご支援を得て、私は引き続き音声認識や音声の雑音除去の他に、深層学習についても研究し続けていました。

私は2020年に無事修士号を取得し、博士課程に進学することともに、AIOCRソフトウェアを作るベンチャー企業を知り合いとともに起業し、会社を経営する経験を得ることができました。研究室での研究生活のお陰で、ベンチャー企業でAI-OCRソフトウェアを開発と研究をする上で自分の持てる情熱と才能を最大限に発揮することができました。また自分の研究の他にも、山梨大学が代表研究機関として研究しているスマート農業プロジェクトにも関わることができ、自分がやりたかったAIロボットの研究をすることも

きました。その際にご指導をしてくださった、山梨大学の理事・副学長、茅 暁陽先生、山梨大学大学院総合研究部工学域准教授牧野浩二先生、山梨大学大学院総合研究部工学西崎先生に心より感謝申し上げます。

しかし、ソフトウェアの開発と会社経営、そして博士課程の勉強をすべて同時にこなすのはとても難しく、決して順風満帆ではありませんでした。会社でうまく行かず、私は一度心を病んでしまいました。このとき、西崎先生はとても忙しいにも関わらず、親身に私の話を聞いてくださいました。また、カウンセリングサポート室の西谷晋二先生にもカウンセリングでいろいろ相談に乗ってもらいました。今私が立ち直れたことは、先生方のおかげだと考えています。そのことについて深く感謝いたします。その後、私は会社を退職いたしました。一緒に仕事をしてくださった仲間にとっても感謝します。

本論文は、山梨大学大学院医工学農学総合研究科システム統合工学コース博士課程における私の研究の集大成である。

ここに至るまでの学問の旅路において、山梨大学大学院総合研究部工学域の西崎博光教授には、私が学部生として山梨大学に入学し当初から、卓越した指導者であり導き手として深い影響を与えていただきました。西崎研究室での先生のご指導は、私の学術的な基盤を築く上で不可欠であり、厳格な研究の姿勢と専門家としての卓越性を身に付けさせていただきました。特に、研究方法論や専門的なコミュニケーションにおける先生の精密なアプローチは、私の学者としての成長に大きく貢献しました。

この場を借りて、山梨大学大学院総合研究部工学域教授鈴木良弥先生、山梨大学大学院総合研究部工学域教授小澤賢司先生、山梨大学大学院総合研究部工学域教授大淵竜太郎先生、山梨大学大学院総合研究部工学域准教授北村敏也先生、山梨大学大学院総合研究部生命環境学域教授伊藤一帆先生には、博士論文の副査として頂いたご指導に心から感謝申し上げます。

また、本論文の共同研究者である矢島英明氏と北川智樹氏に対しては、共同での研究に対する揺るぎない支援と協力に対し、特別な感謝の気持ちを表します。彼らとの友情と知的交流は、私の研究旅程における重要な柱でありました。

研究室の先輩、同期、後輩たちにも心から感謝しています。彼らの仲間意識、知恵の共有、魅力的なディスカッションは、私の学問人生を大いに豊かにしてくれました。困難に直面した経験の積み重ねが、私の研究観の形成に役立っています。西崎研で過ごした素晴らしい7年間を振り返りながら、私はこの7年間を、研究者としての私自身の原点に立ち返ろうと思います。

最後になりますが、留学生活を支えてくださった方々、専門的な知識をご指導いただいた先生方、そして困難な時に助言をくださった友人や仕事仲間の皆様に、心より感謝の意を表します。人生は何もかも順調なことばかりではありませんが、私はこれからも助けてくださった人のことを忘れずに、自分自身が決めたことを一生懸命生きていきたいと思っています。

References

- [1] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, “Real-time scene text detection with differentiable binarization,” Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-20), pp.11474–11481, 2020.
- [2] M. Liao, Z. Zou, Z. Wan, C. Yao, and X. Bai, “Real-time scene text detection with differentiable binarization and adaptive scale fusion,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.45, no.01, pp.919–931, 2023.
- [3] M. Patacchiola, P. Fox-Roberts, and E. Rosten, “Y-autoencoders: disentangling latent representations via sequential-encoding,” Pattern Recognition Letters, vol.140, pp.59–65, 2020.
- [4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning,” 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp.265–283, 2016.
- [5] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” Advances in Neural Information Processing Systems, ed. H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Curran Associates, Inc., 2019.
- [6] S. Che, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, and K. Skadron, “A performance study of general-purpose applications on graphics processors using cuda,” Journal of Parallel and Distributed Computing, vol.68, no.10, pp.1370–1380, 2008. General-Purpose Processing using Graphics Processing Units.
- [7] Tegaki.ai, “Tegaki ai handwriting recognition.” <https://www.tegaki.ai/>, 2023. Online Accessed: 2023.11.19.
- [8] NTT East, “Rpa ai ocr service.” https://business.ntt-east.co.jp/service/rpa_aiocr/, 2023. Online Accessed: 2023.11.19.

- [9] J.S. Denker, W.R. Gardner, H.P. Graf, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, H.S. Baird, and I. Guyon, “Neural network recognizer for hand-written zip code digits,” *Proceedings of the Advances in Neural Information Processing Systems 1 (NIPS 1988)*, pp.323–331, 1988.
- [10] A. Buslaev, V.I. Iglovikov, E. Khvedchenya, A. Parinov, M. Druzhinin, and A.A. Kalinin, “Albumentations: Fast and flexible image augmentations,” *Information*, vol.11, no.2, 2020.
- [11] U. Marti and H. Bunke, “The IAM-database: An english sentence database for off-line handwriting recognition,” *International Journal on Document Analysis and Recognition*, vol.5, pp.39–46, 2002.
- [12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol.86, no.11, pp.2278–2324, 1998.
- [13] G. Cohen, S. Afshar, J.C. Tapson, and A. van Schaik, “EMNIST: Extending MNIST to handwritten letters,” *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, pp.2921–2926, 2017.
- [14] C.L. Liu, F. Yin, D.H. Wang, and Q.F. Wang, “CASIA online and offline chinese handwriting databases,” *Proceedings of the 2011 International Conference on Document Analysis and Recognition (ICDAR)*, pp.37–41, 2011.
- [15] Y. Zhu, Z. Xie, L. Jin, X. Chen, Y. Huang, and M. Zhang, “SCUT-EPT: New dataset and benchmark for offline chinese text recognition in examination paper,” *IEEE Access*, vol.7, pp.370–382, 2019.
- [16] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, and H. Hu, “SimMIM: A simple framework for masked image modeling,” *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.9643–9653, 2022.
- [17] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, “Masked autoencoders are scalable vision learners,” *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.15979–15988, 2022.
- [18] C. Wei, H. Fan, S. Xie, C.Y. Wu, A. Yuille, and C. Feichtenhofer, “Masked feature prediction for self-supervised visual pre-training,” *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.14648–14658, 2022.
- [19] E.D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q.V. Le, “AutoAugment: Learning augmentation strategies from data,” *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.113–123, 2019.

- [20] B. Moysset and R. Messina, “Manifold mixup improves text recognition with ctc loss,” Proceedings of the 2019 International Conference on Document Analysis and Recognition (ICDAR), pp.799–804, 2019.
- [21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” Advances in Neural Information Processing Systems 27 (NIPS 2014), vol.27, 2014.
- [22] J.Y. Zhu, T. Park, P. Isola, and A.A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” 2017 IEEE International Conference on Computer Vision (ICCV), pp.2242–2251, 2017.
- [23] B. Chang, Q. Zhang, S. Pan, and L. Meng, “Generating handwritten chinese characters using cyclegan,” pp.199–207, 2018.
- [24] W. Kong and B. Xu, “Handwritten chinese character generation via conditional neural generative models,” Proceedings of the 31st Conference on Neural Information Processing Systems NIPS, 2017.
- [25] I. Csiszar, “ I -Divergence Geometry of Probability Distributions and Minimization Problems,” The Annals of Probability, vol.3, no.1, pp.146 – 158, 1975.
- [26] S. Kullback and R.A. Leibler, “On Information and Sufficiency,” The Annals of Mathematical Statistics, vol.22, no.1, pp.79 – 86, 1951.
- [27] L.A. Gatys, A.S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- [28] Y. LeCun and Y. Bengio, Convolutional Networks for Images, Speech, and Time Series, p.255–258, MIT Press, Cambridge, MA, USA, 1998.
- [29] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48, ICML’16, p.1349–1357, JMLR.org, 2016.
- [30] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- [31] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” Proceedings of the 32nd International Conference on Machine Learning, ed. F. Bach and D. Blei, Proceedings of Machine Learning Research, vol.37, Lille, France, pp.448–456, PMLR, 2015.

- [32] T. Kitagawa, C.S. Leow, and H. Nishizaki, “Handwritten character generation using Y-autoencoder for character recognition model training,” Proceedings of the 13th Language Resources and Evaluation Conference (LREC 2022), pp.7344–7351, 2022.
- [33] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, “East: An efficient and accurate scene text detector,” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.2642–2651, 2017.
- [34] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, “Textsnake: A flexible representation for detecting text of arbitrary shapes,” Proceedings of the European Conference on Computer Vision (ECCV), pp.20–36, 2018.
- [35] Y. Liu, H. Chen, C. Shen, T. He, L. Jin, and L. Wang, “Abcnet: Real-time scene text spotting with adaptive bezier-curve network,” IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020.
- [36] J. Ye, Z. Chen, J. Liu, and B. Du, “Textfusenet: Scene text detection with richer fused features,” Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, pp.516–522, 2020.
- [37] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character region awareness for text detection,” Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.9357–9366, 2019.
- [38] Y. Baek, S. Shin, J. Baek, S. Park, J. Lee, D. Nam, and H. Lee, “Character region attention for text spotting,” Computer Vision – ECCV 2020, pp.504–521, Springer International Publishing, 2020.
- [39] M. Javed, P. Nagabhushan, and B.B. Chaudhuri, “A review on document image analysis techniques directly in the compressed domain,” Artificial Intelligence Review, vol.50, no.4, pp.539–568, 2018.
- [40] X. Wu, T. Ma, X. Du, Z. Hu, J. Yang, and L. He, “Drfn: A unified framework for complex document layout analysis,” Information Processing Management, vol.60, no.3, p.103339, 2023.
- [41] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, “Layoutlm: Pre-training of text and layout for document image understanding,” pp.1192–1200, 2020.
- [42] Y. Xu, Y. Xu, T. Lv, L. Cui, F. Wei, G. Wang, Y. Lu, D. Florencio, C. Zhang, W. Che, M. Zhang, and L. Zhou, “LayoutLMv2: Multi-modal pre-training for visually-rich document understanding,” Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), ed. C. Zong, F. Xia, W. Li, and R. Navigli, Online, pp.2579–2591, Association for Computational Linguistics, 2021.

- [43] Z. Shen, R. Zhang, M. Dell, B.C.G. Lee, J. Carlson, and W. Li, “Layoutparser: A unified toolkit for deep learning based document image analysis,” Document Analysis and Recognition – ICDAR 2021, pp.131–146, 2021.
- [44] S. Long, S. Qin, D. Panteleev, A. Bissacco, Y. Fujii, and M. Raptis, “Towards end-to-end unified scene text detection and layout analysis,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.1049–1059, 2022.
- [45] Y. Huang, T. Lv, L. Cui, Y. Lu, and F. Wei, “Layoutlmv3: Pre-training for document ai with unified text and image masking,” pp.4083–4091, 2022.
- [46] I. Sutskever, O. Vinyals, and Q.V. Le, “Sequence to sequence learning with neural networks,” Advances in neural information processing systems, pp.3104–3112, 2014.
- [47] A. Aberdam, R. Litman, S. Tsiper, O. Anshel, R. Slossberg, S. Mazor, R. Manmatha, and P. Perona, “Sequence-to-sequence contrastive learning for text recognition,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.15302–15312, 2021.
- [48] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2015.
- [49] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks,” Proceedings of the 23rd International Conference on Machine Learning, ICML ’06, New York, NY, USA, p.369–376, Association for Computing Machinery, 2006.
- [50] H. Li, P. Wang, C. Shen, and G. Zhang, “Show, attend and read: A simple and strong baseline for irregular text recognition,” Proceedings of the AAAI Conference on Artificial Intelligence, vol.33, pp.8610–8617, 2019.
- [51] F. Min, S. Zhu, and Y. Wang, “Offline handwritten chinese character recognition based on improved googlenet,” Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Pattern Recognition, AIPR 2020, New York, NY, USA, p.42–46, Association for Computing Machinery, 2020.
- [52] A. Graves and J. Schmidhuber, “Offline handwriting recognition with multidimensional recurrent neural networks,” Advances in Neural Information Processing Systems, ed. D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Curran Associates, Inc., 2009.

- [53] N.T. Ly, C.T. Nguyen, K.C. Nguyen, and M. Nakagawa, “Deep convolutional recurrent network for segmentation-free offline handwritten japanese text recognition,” 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), pp.5–9, 2017.
- [54] R. Ingle, Y. Fujii, T. Deselaers, J. Baccash, and A. Popat, “A scalable handwritten text recognition system,” 2019 International Conference on Document Analysis and Recognition (ICDAR), pp.17–24, 2019.
- [55] D. Coquenot, C. Chatelain, and T. Paquet, “Faster dan: Multi-target queries with document positional encoding for end-to-end handwritten document recognition,” International Conference on Document Analysis and Recognition (ICDAR), Lecture Notes in Computer Science, vol.14190, pp.182–199, 2023.
- [56] D. Coquenot, C. Chatelain, and T. Paquet, “Dan: a segmentation-free document attention network for handwritten document recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol.45, pp.8227–8243, 2023.
- [57] G.M. de Buy Wenniger, L. Schomaker, and A. Way, “No padding please: Efficient neural handwriting recognition,” 2019 International Conference on Document Analysis and Recognition (ICDAR), Los Alamitos, CA, USA, pp.355–362, IEEE Computer Society, 2019.
- [58] Jaida AI, “EasyOCR,” 2020. <https://github.com/JaidedAI/EasyOCR> [Online Accessed: 2023.4.1].
- [59] Z. Kuang, H. Sun, Z. Li, X. Yue, T.H. Lin, J. Chen, H. Wei, Y. Zhu, T. Gao, W. Zhang, K. Chen, W. Zhang, and D. Lin, “Mmocr: A comprehensive toolbox for text detection, recognition and understanding,” Proceedings of the 29th ACM International Conference on Multimedia, MM ’21, New York, NY, USA, p.3791–3794, Association for Computing Machinery, 2021.
- [60] Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang, and H. Wang, “PP-OCR: A practical ultra lightweight OCR system,” CoRR, vol.abs/2009.09941, 2020.
- [61] Y. Du, C. Li, R. Guo, C. Cui, W. Liu, J. Zhou, B. Lu, Y. Yang, Q. Liu, X. Hu, D. Yu, and Y. Ma, “Pp-ocrv2: Bag of tricks for ultra lightweight ocr system,” ArXiv, vol.abs/2109.03144, 2021.
- [62] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.770–778, 2016.

- [63] C.Y. Lee and S. Osindero, “Recursive recurrent nets with attention modeling for ocr in the wild,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
- [64] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), pp.1–14, 2015.
- [65] M. Li, T. Lv, L. Cui, Y. Lu, D.A.F. Florêncio, C. Zhang, Z. Li, and F. Wei, “Trocr: Transformer-based optical character recognition with pre-trained models,” AAAI Conference on Artificial Intelligence, 2021.
- [66] D.H. Diaz, S. Qin, R.R. Ingle, Y. Fujii, and A. Bissacco, “Rethinking text line recognition models,” CoRR, vol.abs/2104.07787, 2021.
- [67] 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,” Proceedings of the International Conference on Learning Representations (ICLR 2021), pp.1–21, 2021.
- [68] G. Tauschek, “Reading machine,” 1935. US Patent 2,026,330.
- [69] H. of Computers, “Optical character recognition.” <https://history-computer.com/optical-character-recognition/>. Online Accessed: 2023.11.20.
- [70] K. Technologies, “Kurzweil technologies.” <https://www.kurzweiltech.com/kcp.html>, 2000. Online Accessed: 2023.11.20.
- [71] R. Bales, “Apple ii explained: Everything you need to know.” History-Computer.com, 2023. Last updated July 31, 2023.
- [72] Adobe, “Who created pdf.” <https://blog.adobe.com/en/publish/2015/06/18/who-created-pdf>, 2015. Online Accessed: [2023.11.20].
- [73] D. Kalina and R. Golovanov, “Application of template matching for optical character recognition,” 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIconRus), pp.2213–2217, 2019.
- [74] J.C. Rodríguez-Rodríguez, G.S. de Blasio, C.R. García, and A. Quesada-Arencibia, “A very high-speed validation scheme based on template matching for segmented character expiration codes on beverage cans,” Sensors, vol.20, no.11, 2020.
- [75] D. Marr and E. Hildreth, “Theory of edge detection,” Proceedings of the Royal Society of London. Series B, Biological Sciences, vol.207, no.1167, pp.187–217, 1980.

- [76] M. Okamoto and K. Yamamoto, “On-line handwriting character recognition method with directional features and direction-change features,” Proceedings of the Fourth International Conference on Document Analysis and Recognition, pp.926–930 vol.2, 1997.
- [77] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, “Learning internal representations by error propagation,” in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, ed. D.E. Rumelhart and J.L. McClelland, pp.318–362, MIT Press, Cambridge, MA, 1986.
- [78] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol.9, no.8, pp.1735–1780, 1997.
- [79] H.T. Nguyen, C.T. Nguyen, and M. Nakagawa, “Online japanese handwriting recognizers using recurrent neural networks,” 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp.435–440, 2018.
- [80] F. Yang, F. Bao, and G. Gao, “Online handwritten mongolian character recognition using cma-mohr and coordinate processing,” 2020 International Conference on Asian Language Processing (IALP), pp.30–33, 2020.
- [81] H.G. E., “Boltzmann machines, constrained satisfaction network that learn,” *CMU-CS*, pp.84–119, 1984.
- [82] O. Matan, H. Baird, J. Bromley, C. Burges, J. Denker, L. Jackel, Y. Le Cun, E. Pednault, W. Satterfield, C. Stenard, and T. Thompson, “Reading handwritten digits: a zip code recognition system,” *Computer*, vol.25, no.7, pp.59–63, 1992.
- [83] Y. Bengio, Y. LeCun, and D. Henderson, “Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden markov models,” *Advances in Neural Information Processing Systems*, ed. J. Cowan, G. Tesauro, and J. Alspector, Morgan-Kaufmann, 1993.
- [84] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol.323, pp.533–536, 1986.
- [85] H.E. Robbins, “A stochastic approximation method,” *Annals of Mathematical Statistics*, vol.22, pp.400–407, 1951.
- [86] D.P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” Proceedings of the 3rd International Conference on Learning Representations (ICLR), pp.1–15, 2015.
- [87] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol.12, no.61, pp.2121–2159, 2011.

- [88] G. Hinton, “Csc321: Introduction to neural networks and machine learning.” University of Toronto, n.d. Lecture 6 Slides.
- [89] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS’17, Red Hook, NY, USA, p.6000–6010, Curran Associates Inc., 2017.
- [90] F. Xu, C. Chen, Z. Shang, Y. Peng, and X. Li, “A crnn-based method for chinese ship license plate recognition,” IET Image Processing, pp.1–14, 2023.
- [91] D. Wang, Y. Tian, W. Geng, L. Zhao, and C. Gong, “Lpr-net: Recognizing chinese license plate in complex environments,” Pattern Recognition Letters, vol.130, pp.148–156, 2020. Image/Video Understanding and Analysis (IUVA).
- [92] N. Otsu, “A threshold selection method from gray-level histograms,” IEEE Transactions on Systems, Man, and Cybernetics, vol.9, no.1, pp.62–66, 1979.
- [93] W. Niblack, An introduction to digital image processing, Strandberg Publishing Company, 1985.
- [94] K. Khurshid, I. Siddiqi, C. Faure, and N. Vincent, “Comparison of niblack inspired binarization methods for ancient documents,” pp.1–10, 2009.
- [95] OpenCV, “Open computer vision library.” <https://opencv.org/>, 2023. Online Accessed: 2023.11.20.
- [96] R. Fisher *et al.*, “Morphology.” <https://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm>, 2023. Online Accessed: 2023.11.20.
- [97] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, pp.234–241, 2015.
- [98] G. Hinton and R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” Science (New York, N.Y.), vol.313, pp.504–7, 2006.
- [99] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), pp.1510–1519, 2017.
- [100] J. Gu and J.C. Ye, “Adain-based tunable cyclegan for efficient unsupervised low-dose ct denoising,” IEEE Transactions on Computational Imaging, vol.7, pp.73–85, 2021.

- [101] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- [102] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee, “Character region awareness for text detection –supplementary material–,” Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp.1–2, 2019. Online Accessed: 2023.1.21].
- [103] National Institute of Advanced Industrial Science and Technology, “ETL Character Database,” 2014. <http://etlcdb.db.aist.go.jp/> [Online Accessed: 2023.4.1].
- [104] K. Maekawa, M. Yamazaki, T. Ogiso, T. Maruyama, H. Ogura, W. Kashino, H. Koiso, M. Yamaguchi, M. Tanaka, and Y. Den, “Balanced corpus of contemporary written japanese,” Language Resources and Evaluation, vol.48, no.2, pp.345–371, 2014.
- [105] M. Fujitake, “Dtrocr: Decoder-only transformer for optical character recognition,” 2023.
- [106] T. Lv, Y. Huang, J. Chen, L. Cui, S. Ma, Y. Chang, S. Huang, W. Wang, L. Dong, W. Luo, S. Wu, G. Wang, C. Zhang, and F. Wei, “Kosmos-2.5: A multimodal literate model,” 2023.

Relationship between publications and this thesis

The proposed text image recognition and detection relationship from chapter 2 Text Image Recognition, chapter 4 Text Detection, and chapter 7 Single-line Text Detection In Multiple-lines Text Images.

1. Chee Siang Leow, Hideaki Yajima, Tomoki Kitagawa, and Hiromitsu Nishizaki, “Single-line Text Detection in Multi-line Text with Narrow Spacing for Line-based Character Recognition,” *IEICE Transaction on Information & Systems*, Vol.E106-D, No.12, pp.2097-2106, 2023, DOI:10.1587/transinf.2023EDP7070.

The proposed character image generation relationship in chapter 2 Text Image Recognition, chapter 5 Image Generation Using Deep Learning, chapter 6 Character Generation with Y-Autoencoder.

1. Tomoki Kitagawa, Chee Siang Leow, Hiromitsu Nishizaki, “Handwritten Character Generation using Y-Autoencoder for Character Recognition Model Training,” *Proceedings of the Language Resources and Evaluation Conference (LREC 2022)*, pp. 7344-7351, 2022.
2. Chee Siang Leow, Tomoki Kitagawa, Hideaki Yajima, Hiromitsu Nishizaki, “Data Augmentation With Automatically Generated Images for Character Classifier Model Training,” *Proceedings of the 2023 IEEE 12th Global Conference on Consumer Electronics (GCCE 2023)*, pp. 856-860, 2023.

Publications

Journals

1. 西崎博光, レオ チーシャン, 牧野浩二, “小型コンピュータにおける深層学習アプリケーションの動作検証”, 電気学会論文誌 C, Vol.138, No.9, pp.1108-1115, 2018.9, DOI:10.1541/ieejeiss.138.1108
2. Tatsuyoshi Amemiya, Chee Siang Leow, Prawit Buayai, Koji Makino, Xiaoyang Mao, Hiromitsu Nishizaki, “Appropriate grape color estimation based on metric learning for judging harvest timing,” The Visual Computer, pp.4083-4094, 2022, DOI: 10.1007/s00371-022-02666-0
3. Chee Siang Leow, Hideaki Yajima, Tomoki Kitagawa, and Hiromitsu Nishizaki, “Single-line Text Detection in Multi-line Text with Narrow Spacing for Line-based Character Recognition,” IEICE Transaction on Information & Systems, Vol.E106-D, No.12, pp.2097-2106, 2023, DOI:10.1587/transinf.2023EDP7070.

International Conference Presentations (Peer-Reviewed)

1. Chee Siang Leow, Hiromitsu Nishizaki, “A Task Manual Creation Support System Using Automatic Speech Recognition,” Proceedings of the 2018 IEEE 7th Global Conference on Consumer Electronics (GCCE), pp.259-262, 2018, DOI: 10.1109/GCCE.2018.8574796
2. Akifumi Yamamoto, Christian Bilgera, Maki Sawano, Haruka Matsukura, Naoki Sawada, Chee-Siang Leow, Hiromitsu Nishizaki, and Hiroshi Ishida, “Application of Sequence Input and Output Long Short-Term Memory Neural Networks for Autonomous Gas Source Localization in an Outdoor Environment,” Proceedings of the 2019 ISOCS/IEEE International Symposium on Olfaction and Electronic Nose (ISOEN), pp.1–3, 2019, DOI: 10.1109/ISOEN.2019.8823160
3. Yuta Sano, Chee Siang Leow, Soichiro Iida, Takehito Utsuro, Junichi Hoshino, Akio Kobayashi, and Hiromitsu Nishizaki, “Spoken Dialog Training System for Customer

- Service Improvement,” Proceedings of the 12th Asia-Pacific Signal and Information Processing Association Annual Summit and Conference 2020 (APSIPA ASC), pp.403-408, 2020.
4. Chee Siang Leow, Tomoaki Hayakawa, Hiromitsu Nishizaki, and Norihde Kitaoka, “Development of a Low-Latency and Real-Time Automatic Speech Recognition System,” Proceedings of the 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE 2020), pp.925-928, 2020, DOI: 10.1109/GCCE50665.2020.9291818
 5. Yu Wang, Chee Siang Leow, Hiromitsu Nishizaki, Akio Kobayashi, and Takehito Utsuro, “ExKaldi: A Python-Based Extension Tool of Kaldi,” Proceedings of the 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE 2020), pp.929-932, 2020, DOI: 10.1109/GCCE50665.2020.9291717
 6. Yu Wang, Chee Siang Leow, Akio Kobayashi, Takehito Utsuro, Hiromitsu Nishizaki, “ExKaldi-RT: A Real-Time Automatic Speech Recognition Extension Toolkit of Kaldi,” Proceedings of the 2021 IEEE 10th Global Conference on Consumer Electronics (GCCE 2021), pp. 320-324, 2021, DOI: 10.1109/GCCE53005.2021.9621992
 7. Tatsuyoshi Amemiya, Kodai Akiyama, Chee Siang Leow, Prawit Buayai, Koji Makino, Xiaoyang Mao, and Hiromitsu Nishizaki, “Development of a Support System for Judging the Appropriate Timing for Grape Harvesting,” Proceedings of the 2021 International Conference on Cyberworlds (CW2021), pp.194-200, 2021, DOI: 10.1109/CW52790.2021.00040
 8. Prawit Buayai, Kabin Yok-In, Daisuke Inoue, Chee Siang Leow, Hiromitsu Nishizaki, Koji Makino and Xiaoyang Mao, “End-to-End Inflorescence Measurement for Supporting Table Grape Trimming with Augmented Reality,” Proceedings of the 2021 International Conference on Cyberworlds (CW2021), pp.101-108, 2021, DOI: 10.1109/CW52790.2021.00022
 9. Tomoaki Hayakawa, Chee Siang Leow, Akio Kobayashi, Takehito Utsuro, and Hiromitsu Nishizaki, “Language and Speaker-Independent Feature Transformation for End-to-End Multilingual Speech Recognition,” Proceedings of INTERSPEECH 2021, pp.2431-2435, 2021, DOI:10.21437/Interspeech.2021-390
 10. Yuto Nonaka, Chee Siang Leow, Akio Kobayashi, Takehito Utsuro, and Hiromitsu Nishizaki, “Voice Activity Detection for Live Speech of Baseball Game Based on Tandem Connection with Speech/Noise Separation Model,” Proceedings of INTERSPEECH2021, pp.351-355, 2021, DOI:10.21437/Interspeech.2021-792
 11. Tomoki Kitagawa, Chee Siang Leow, Hiromitsu Nishizaki, “Handwritten Character Generation using Y-Autoencoder for Character Recognition Model Training,” Proceedings of the Language Resources and Evaluation Conference (LREC 2022), pp. 7344-7351, 2022.

12. Chee Siang Leow, Tomoki Kitagawa, Hideaki Yajima, Hiromitsu Nishizaki, “Data Augmentation With Automatically Generated Images for Character Classifier Model Training,” Proceedings of the 2023 IEEE 12th Global Conference on Consumer Electronics (GCCE 2023), pp. 856-860, 2023.
13. Akihiro Dobashi, Chee Siang Leow, Hiromitsu Nishizaki, “Metric Learning Approach for End-To-End Multilingual Automatic Speech Recognition Model,” Proceedings of the 2023 IEEE 12th Global Conference on Consumer Electronics (GCCE 2023), pp. 447-451, 2023.
14. Yinghao He, Chee Siang Leow, Hiromitsu Nishizaki, “Image Remapping Data Augmentation Approach for Improving Fisheye Face Recognition,” Proceedings of the 2023 IEEE 12th Global Conference on Consumer Electronics (GCCE 2023), pp. 745-749, 2023.
15. Chee Siang Leow, Ryosuke Shimazu, Tomoki Kitagawa, Hideki Yajima, Prawit Buayai, Koji Makino, Xiaoyang Mao, Hiromitsu Nishizaki “Estimation of Non-Invasive Grape Ripeness and Sweetness From Images Captured by a General-Purpose Camera,” Proceedings of the 2023 IEEE International Workshop on Metrology for Agriculture and Forestry, pp.295-300, 2023.

International Conference Presentations (Non-Peer-Reviewed)

1. Akihiro Dobashi, Chee Siang Leow, Hiromitsu Nishizaki, “Frequency-Directional Attention Model for Multilingual Automatic Speech Recognition,” arXiv preprint, arXiv:2203.15473, pp.1–5, 2022.03.29.
2. Yu Wang, Chee Siang Leow, Akio Kobayashi, Takehito Utsuro, Hiromitsu Nishizaki, “ExKaldi-RT: A Real-Time Automatic Speech Recognition Extension Toolkit of Kaldi,” arXiv preprint, arXiv:2104.01384, 2021.4.

Domestic Presentations (Non-Peer-Reviewed)

1. 西崎博光, Leow Chee Siang, “技術伝承のための作業記録の作成・閲覧支援システムの開発”, 平成 29 年度山梨大学 COC 事業成果報告会, 2018 年 3 月 19 日.
2. 西崎博光, Leow Chee Siang, “映像と音声で記録した作業コンテンツを用いる技術伝承のための手順書作成支援”, やまなし産学官連携研究交流事業研究発表会, 2017 年 10 月 31 日.

3. Yu Wang, Hiromitsu Nishizaki , Akio Kobayashi , Takehito Utsuro, Chee Siang Leow, “Development and Evaluation of Kaldi Extension Tools with Python,” 情報処理学会研究報告, 音声言語情報処理, 2019-SLP-130(5), pp.1-5, 2019.12.
4. Chee Siang Leow , Hiromitsu Nishizaki , Akio Kobayashi , Takehito Utsuro, “Speech Recognition-based Evaluation of a Noise Reduction Method in Known-Noise Environment,” 情報処理学会研究報告, 音声言語情報処理, 2019-SLP-130(5), pp.1-6, 2019.12.
5. 山本晃史, Christian Bilgera, 澤野真樹, 松倉悠, 澤田直輝, Chee Siang Leow, 西崎博光, 石田寛, “深層学習を用いた屋外環境における自動ガス源探索—入力するセンサデータの長さについての検討—”, 2019 年電気学会センサ・マイクロマシン部門大会第 36 回「センサ・マイクロマシンと応用システム」シンポジウム論文集, 19am3-PS3-49, p.1-4, 2019.11.
6. 佐野祐太, レオチーシャン, 飯田宗一郎, 西崎博光, 星野准一, 宇津呂武仁, “接客訓練のための音声対話システムの試作”, 日本音響学会 2020 年春季研究発表会, 3-P-12, pp.1021-1022, 2019.3.
7. レオ チーシャン, 西崎博光, “既知の工場環境音を用いた深層学習に基づく工作機械雑音除去の検討”, 日本音響学会 2019 年秋季研究発表会, 1-P-2, pp.829-830, 2019.9.4.
8. レオ チーシャン, 早川友瑛, 西崎博光, 北岡教英, “Kaldi ベースの低遅延リアルタイム音声認識システムの開発と評価”, 日本音響学会 2020 年秋季研究発表会講演論文集, 2-P1-3, pp. 837–838, 2020.9.
9. 西尾瞳希, 飯田宗一郎 , 佐野祐太, Leow Chee Siang, 西崎博光, 宇津呂武仁, 星野准一, “話し方のトレーニングが可能な接客訓練 VR システム”, 情報処理学会研究報告, コンピュータビジョンとイメージメディア, Vol.2021-CVIM-224, No.9, pp.1-4, 2021.1.
10. 土橋晃弘, レオチーシャン, 西崎博光, “周波数軸注意機構を用いた特徴変換モデルに基づく複数言語音声認識”, 日本音響学会 2022 年春季研究発表会講演論文集, 2-3P-6, pp.1109-1112, 2022.3.10.
11. 北川智樹, レオ チーシャン, 西崎博光, “文字認識モデル訓練のための手書き文字生成”, 情報処理学会第 84 回全国大会講演論文集, 7Q-05, pp.2-275–2.276, 2022.3.5.
12. レオ チーシャン, 王 宇, 小林彰夫, 宇津呂武仁, 西崎博光, “Kaldi ベースのストリーミング音声認識システムの開発”, 日本音響学会 2021 年秋季研究発表会講演論文集, 1-3Q-4, pp.1033-1036, 2021.9.7.
13. 雨宮達佳, レオ チーシャン, ブアヤイ プラウィット, 牧野浩二, 茅 暁陽, 西崎博光, “画像の色空間を考慮したシャインマスカットブドウの色推定モデル”, 情報処理学会第 85 回全国大会講演論文集, 4Q-04, vol.2, pp.221-222, 2023.3.3.

14. 土橋晃弘, レオ チーシャン, 西崎博光, “End-to-End 複数言語音声認識モデル訓練における距離学習の効果”, 日本音響学会 2023 年秋季研究発表会講演論文集, 3-Q-3, pp.143, 2023.9.29.
15. 西崎博光, 雨宮達佳, レオ チーシャン, ブアヤイ プラウィット, 牧野浩二, 茅 暁陽, “シャインマスカット栽培支援ロボットのための色推定モデルを用いた収穫適期判定システム”, ロボティクス・メカトロニクス講演会講演論文集, 講演番号 2A1-B03, pp. 2A1-B03(1)-(4), 2023.6.30.
16. 牧野 浩二, 丹沢 勉, 柴山 航太郎, Bong Tze Yaw, Leow Chee Siang, 西崎 博光, “音と人感センサを利用した果実盗難防止のための通報装置の開発”, 第 24 回計測自動制御学会システムインテグレーション部門講演会, 1C4-08, 2023.12.14.
17. Bong Tze Yaw, Leow Chee Siang, 丹沢 勉, 牧野 浩二, 西崎 博光, “果実盗難通報装置のための小型マイコンで動作する不審音検出システム”, 第 24 回計測自動制御学会システムインテグレーション部門講演会, 1C4-09, 2023.12.14.

Appendix A

Y-AE Generated Kanji Statistics

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
一	7675	丁	5230	七	1380	万	18512
丈	17252	三	26376	上	22221	下	8749
不	55139	与	10205	丑	57416	且	32317
世	20165	丘	26972	丙	24434	両	665
並	8964	中	11623	串	16565	丸	5730
丹	47518	主	40905	乃	20742	久	22419
之	8099	乍	42660	乎	12829	乏	14218
乘	16271	乙	8370	九	12399	乞	18300
也	15721	乱	23015	乳	19929	乾	10877
龜	11318	了	12765	予	25131	争	18731
事	16410	二	3772	云	54740	互	30620
五	25312	井	7028	亘	10279	互	11961
些	24657	垂	26376	亡	5669	交	27309
亥	26559	亦	24806	亨	9543	享	21360
京	24221	亭	5757	亮	7720	人	15383
什	22727	仁	46424	仇	49601	今	49225
介	36257	仏	46751	仔	25707	仕	15595
他	38440	付	25679	仙	37238	代	25586
令	16576	以	28052	仮	18993	仰	3399
仲	1936	件	18843	任	1291	企	22277
伊	22496	伍	18163	伎	25609	伏	24285
伐	7559	休	127	会	31193	伝	2429
伯	27139	伴	15330	伶	27039	伸	21470
伺	20739	似	10356	伽	30478	佃	23932
但	21102	位	25961	低	24885	住	2092
佐	25020	佑	22919	体	20395	何	8820
余	24041	作	25214	佳	16713	併	11397
佼	26429	使	20251	侃	24460	例	9405
侍	12166	供	24412	儷	15013	侠	19094
倆	8953	俚	23139	侮	13874	侯	7211

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
侵	9938	侶	8007	便	26420	係	26473
促	10018	俄	7636	俊	9279	俗	9196
保	28109	信	13511	俣	9941	修	5751
俳	9698	俵	10200	俸	28403	俺	9428
倉	9376	個	10246	倍	9757	倒	7127
倅	24705	候	10885	借	5108	倣	24368
值	8986	倦	24327	倫	17439	倭	27312
俱	5017	儉	8316	偉	8408	偏	10379
停	6261	健	10352	偲	28437	側	7484
偵	7949	偶	11542	偽	6292	傍	7018
傑	11846	傘	10877	備	7292	催	10959
傭	10108	債	11471	傷	10827	傾	10174
僅	9612	働	9740	像	9831	僑	6434
僕	12197	僚	9349	僧	8818	僻	12061
儀	7506	億	10268	儒	10624	償	8458
優	8289	儲	7327	允	37102	元	17963
兄	39683	充	26306	兆	10156	兇	9524
先	10426	光	19142	克	24541	免	5639
兎	22414	兎	6978	党	6085	兜	27349
入	25691	全	117	八	11227	公	47133
六	28330	共	6	兵	26442	其	31330
具	22617	典	7763	兼	7900	内	34406
円	21146	冊	17543	再	1709	冒	4624
冗	31821	写	16662	冠	10115	冥	10910
富	16547	冬	36389	冴	17363	冶	19909
冷	21628	凄	6930	准	1598	凋	25832
凌	22718	凍	9179	凝	964	凡	20444
処	36597	凧	35604	凧	2961	凱	10533
凶	18840	凸	11790	凹	21723	出	45079
函	13389	刀	25027	刃	15971	分	38530

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
切	22123	刈	30766	刊	11361	刑	18767
列	14098	初	23206	判	23072	別	9457
利	25390	到	9109	制	13672	刷	26945
券	14266	刺	7883	刻	13598	剃	7895
則	30361	削	9658	前	29668	剖	16084
剛	5314	劍	7720	劑	7076	剥	26545
副	10587	剩	9772	割	7324	創	7964
劃	12299	劇	11300	劉	11694	力	1287
功	22780	加	48411	劣	12690	助	6527
努	25104	劫	23528	励	26255	勞	5343
効	9443	効	18609	勃	9056	勅	6799
勇	19919	勉	9915	勘	9580	務	9497
勝	23330	募	8049	勢	11841	勤	8786
勸	9096	勳	8158	勺	16543	勾	34965
勿	33319	匆	35952	勺	33228	包	21417
化	44894	北	31659	匙	29428	匝	26285
匠	24165	匡	24249	匪	30263	匹	33386
区	35047	医	25089	匿	26059	十	44
千	804	升	23591	午	18117	半	22578
卑	4595	卒	21369	卓	16460	協	11167
南	22795	单	12615	博	4557	占	12615
卦	26433	卯	14763	印	15965	危	28846
即	23495	却	20707	卵	9840	卸	9180
卿	9153	厄	40372	厘	9107	厚	28945
原	9016	厨	10082	厩	11434	厭	11864
巖	7824	去	56225	参	28712	又	7289
叉	8369	及	23076	友	39301	双	41015
反	49107	収	36212	叔	14856	取	24897
受	28657	叙	8033	叛	7737	叡	8660
叢	8943	古	32130	句	10202	叩	19487

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
只	25989	叫	24734	召	28170	可	36765
台	27935	叱	21932	史	19989	右	48671
叶	23192	号	41535	司	16175	吃	3731
各	1402	合	7838	吉	26069	吊	20637
同	20169	名	69	后	30548	吏	24159
吐	25520	向	229	君	26104	吟	23786
吠	30370	否	26578	含	23319	吸	916
吹	26907	吻	30661	吾	24552	吕	26051
呆	30438	呈	31121	呉	29780	告	26180
吞	13915	周	8154	呪	24940	味	21015
呼	24492	命	25321	咋	18106	和	18091
咲	9120	咳	10001	咽	29671	哀	10120
品	26536	哉	7093	員	4233	哨	14748
哩	27032	哲	7689	唄	6954	唆	8636
唇	9411	唐	10688	唾	27010	唯	16385
唱	7457	唾	6057	啄	25782	商	23219
問	7959	啓	10674	善	10784	喉	9768
喋	11716	喚	6572	喜	10360	喝	9147
喧	9699	喪	7944	喫	9223	喬	6112
喰	11125	營	3461	嗣	9848	嘆	10833
嘉	11449	嘗	9685	嘘	5585	嘩	7454
囑	10501	噲	6929	噌	6811	嚙	5152
器	11423	噴	11393	嘶	6883	嚇	6299
囊	5976	囚	23568	四	15286	回	6773
因	24066	团	51	困	16635	圉	2808
凶	4409	固	6809	国	18894	圃	28651
圈	1252	園	2919	土	11386	庄	20819
圭	20779	地	4	坂	23158	均	22151
坊	19887	坐	27827	坑	23751	坤	11205
坦	20835	坪	13206	垂	23598	型	25602

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
垢	7363	垣	7354	埋	16459	城	16923
埜	23456	域	3262	埠	11210	埴	8235
執	4268	培	17874	基	9902	埼	3731
堀	26899	堂	10699	堅	6921	堆	5840
墮	9894	堤	3193	堪	8405	堰	3901
報	8373	場	22995	堵	8414	堺	7424
塀	6148	畧	11652	塊	6324	塑	9937
塔	7306	塗	8832	塘	5699	塙	6587
塚	1431	塞	10585	塩	7261	填	8108
塵	8650	塾	11524	境	6279	墓	8449
增	4512	墜	8361	墨	11595	墳	9701
墾	10686	壁	9892	壇	5132	壤	4591
壕	10096	士	6875	壬	33269	壯	4627
声	26680	壳	27169	壳	24994	壺	14113
变	16807	夏	6348	夕	5	外	38155
夙	8661	多	61	夜	26113	夢	5954
大	18462	天	26038	太	32242	夫	42549
央	30945	失	51972	夷	23384	奄	21919
奇	20574	奈	22070	奉	23540	奏	26082
契	10907	奔	9415	套	8523	奧	1123
獎	10100	奪	10332	奮	10282	女	17371
奴	18336	好	2441	如	14792	妃	23791
妄	1712	妊	16439	妓	21300	妖	22805
妙	26091	妥	18422	妨	28651	妬	12601
妹	25667	妻	24528	妾	14436	姉	25851
始	19248	姐	21084	姑	19736	姓	12042
委	27611	姥	8677	姦	5108	姪	5583
姬	25141	始	6741	姻	23821	姿	9564
威	9660	娃	8461	娘	22360	娠	6078
媿	19547	媿	9440	娼	15777	婁	7535

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
婆	9546	婚	9960	婦	10598	婿	8545
媒	7786	媛	6498	嫁	7811	嫉	3511
嫌	6335	嫡	4675	嬉	10890	孃	3751
孀	5506	嬰	7795	子	5988	孔	44120
字	1	存	14379	孜	17018	孝	2589
孟	13682	季	20639	孤	7619	学	16771
孫	8216	宅	5686	宇	12830	守	2354
安	2464	宋	28292	完	25225	宍	27325
宏	28225	宕	21628	宗	14225	官	21117
宙	19527	定	30148	宛	6852	宜	18981
宝	23392	実	27626	客	21478	宣	27515
室	18647	宥	7380	宮	5957	宰	10543
害	4811	宴	10676	宵	1136	家	8243
容	2986	宿	27140	寂	8745	寄	6434
寅	20360	密	5514	富	6788	寒	29843
寓	8703	寬	6779	寢	4529	察	8740
寡	9103	寧	8681	審	7579	寮	4480
寵	6818	寸	5520	寺	2889	对	22461
寿	30861	封	4958	專	14075	射	8043
将	6965	尉	9818	尊	6740	尋	10739
導	7824	小	25165	少	46068	尖	25745
尚	8041	尤	42815	堯	24606	就	8637
尺	31474	尻	27108	尼	55774	尽	7691
尾	31600	尿	24590	局	25975	居	5182
屈	13869	届	12155	屋	23902	屍	9456
屑	14222	展	6688	属	9462	屠	1718
層	10545	履	11253	屯	11212	山	21268
岐	28055	岡	8815	岨	26874	岩	18729
岬	21477	岱	28263	岳	16931	岸	19471
峠	10157	峡	9644	峨	10796	峯	15965

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
峰	26268	島	8910	峻	27641	崇	5916
崎	7364	崖	9247	崩	7444	嵐	3671
嵩	8283	嵯	7916	嶋	10730	嶺	4712
巖	4458	川	27465	州	6866	巡	14688
巢	8374	左	45139	巧	34871	巨	41712
差	10193	己	1292	巳	1620	巴	26207
巷	8047	卷	19060	巽	10985	巾	22517
市	37961	布	21723	帆	24584	希	23953
帖	20627	帝	8295	帥	9080	師	2390
席	6052	帶	9142	婦	9709	帳	8063
常	8188	帽	6367	幅	8041	幌	8004
幕	9488	幡	5923	幣	8076	干	9067
平	24448	年	142	幸	21997	幹	5217
幻	44552	幼	42200	幽	5207	幾	6736
庁	21221	広	53771	庄	23012	庇	27799
床	24319	序	15571	底	18452	庖	24757
店	30118	庚	25202	府	19505	度	20042
座	9608	庫	4766	庭	4893	庵	16652
庶	8749	康	9479	庸	3468	廢	9429
廉	9362	廊	3875	廓	9438	廟	6274
廠	5864	延	10008	廷	19667	建	11670
廻	6117	廻	5917	廿	21610	弁	19394
弄	25260	弊	6394	式	1449	式	6474
弓	7742	弔	31689	引	17302	弗	9768
弘	33818	弛	25704	弟	6725	弥	19805
弦	16984	弧	8311	弱	10257	張	10554
強	18082	弼	4955	彈	6999	彊	9582
当	13125	形	24542	彦	8000	彩	10926
彪	30797	彪	11172	彬	20269	彰	9916
影	12038	役	25913	彼	26096	往	14681

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
征	24209	徑	23899	待	26164	律	27824
後	19964	徐	10435	徒	10001	從	5031
得	10473	御	9612	復	8534	循	9995
微	4786	德	6461	徵	3971	徹	10460
徽	2414	心	33494	必	9218	忌	28814
忍	29637	志	23348	忘	4946	忙	26469
応	11664	忠	8754	快	25373	念	17204
忽	15347	怒	7622	怖	12777	怜	22153
思	11428	怠	7246	急	24078	性	15205
怨	10469	怪	14207	怯	29105	恋	14284
恐	3454	恒	8269	恕	9961	恢	9353
恥	21759	恨	8415	恩	5716	恭	8865
息	9254	恰	6133	惠	4480	悉	22792
悌	8105	悔	7040	悟	10247	悠	22314
患	8656	悅	7073	惱	24622	惡	29529
悲	11954	悶	4601	悼	23516	情	9469
惇	15828	惑	4041	惚	7012	惜	8204
惟	25867	惚	7453	慘	10677	惰	8447
想	12101	惹	8434	愁	8725	愈	7671
愉	10962	意	2025	愚	8344	愛	4491
感	6078	慈	9884	態	8020	慌	7326
慎	7605	慕	7232	慢	9867	慣	6965
慧	7935	慨	4618	慮	8209	慰	11583
慶	6855	慾	5420	憂	8329	憎	8526
憐	9764	憤	8556	懂	10003	憩	4725
憲	6642	憶	8401	憾	7670	懇	10403
懷	6364	懲	5767	懸	8310	戊	37109
戎	17925	成	1570	我	12028	戒	25128
或	9940	戚	5126	戟	7771	戟	10614
戲	3764	戴	10797	戶	42937	戾	23683

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
房	18069	所	14753	扇	6962	扉	10546
手	20263	才	13692	打	26115	扌	16176
托	7389	扮	20727	扱	15669	扶	20568
批	4932	承	11503	技	6500	抄	28115
把	21578	抑	16093	投	21493	抗	21767
折	21069	拔	22956	扌	25727	披	9736
抱	10320	抵	10588	抹	11415	押	14089
抽	8148	担	5951	拍	6470	拐	16274
拒	11711	拓	10788	拘	13402	拙	8449
招	11464	扌	14256	扌	8198	扌	22005
括	4257	拭	6625	拳	9797	搽	8464
拷	2588	拾	12238	持	16432	指	15036
按	5577	挑	3762	拳	8554	挟	6211
挨	7988	挫	4753	振	8279	挺	3391
挽	8757	插	8029	捉	17394	捌	22584
捕	12724	抄	19719	搜	5027	捧	11345
捨	3995	据	3863	捲	15024	捷	20461
捺	12151	捻	24394	掃	6923	授	8225
掌	9121	排	20132	掘	4097	掛	8146
掠	22883	採	8918	探	4627	接	5960
控	6007	推	6285	掩	21398	措	6443
掬	7511	揭	6899	搵	8807	搔	15089
掬	6130	描	12853	提	5884	搵	7905
揚	9006	換	6255	握	9108	揮	5194
援	8477	搖	9407	損	5022	搬	3320
搭	2929	携	8452	搾	5883	搵	5936
摘	3046	摩	7959	摸	5023	摺	6572
擊	11948	撒	7339	撚	5421	撞	5431
撤	8121	撫	6197	播	4492	撮	8679
撰	6301	撲	4929	攪	6543	擁	3300

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
操	4735	擢	3738	擦	6757	擬	4880
擾	4477	支	32944	改	24378	攻	18821
放	17905	政	17414	故	19241	敏	3993
救	4169	敗	9563	教	17687	敢	7807
散	9110	敦	9836	敬	8150	數	9990
整	10159	敵	9209	敷	8956	文	17745
齊	11604	斌	4931	齋	6635	斐	11813
斑	1619	斗	12821	料	7578	斜	4566
幹	9290	斤	36525	斥	19963	斧	16774
斬	9058	斷	8289	斯	7216	新	9846
方	42212	於	16732	施	5380	旅	5178
旋	7419	族	5762	旗	7190	既	1206
日	17065	旦	29203	旧	42187	旨	26129
早	7	旬	11891	旭	4175	旺	8555
昂	15855	昆	22526	昇	12206	昌	20549
明	26797	昏	26553	易	11760	昔	18452
星	27142	映	19234	春	13893	昧	6078
昨	17268	昭	27987	是	7646	昼	24904
時	5410	晃	28762	晋	24915	晒	28744
晦	9211	晚	5788	普	5984	景	8544
晴	14265	晶	9846	智	6298	曉	6416
暇	6942	暑	5770	暖	10091	暗	10146
暢	9908	曆	8020	暫	11948	暮	8372
暴	11963	曇	9463	曙	10896	曜	8850
曝	10171	曲	3795	曳	14864	更	30260
書	4065	曹	8942	曾	5957	曾	5858
替	9827	最	10439	月	20062	有	1248
朋	28847	服	23283	朔	16237	朕	23925
朗	3127	望	11297	朝	11890	期	22923
木	23785	未	22340	未	50847	本	46454

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
札	27524	朱	20754	朴	17379	机	19342
朽	23110	杉	22034	李	25680	杏	23669
材	17080	村	19537	杓	28801	杖	25949
杜	22121	束	7398	条	14237	柰	31601
来	6223	杭	22266	杯	12228	東	22570
杵	12142	杷	21187	松	21662	板	23644
枇	19613	析	6322	枕	3230	林	20631
枚	16549	果	16877	枝	14426	梓	14784
枢	10666	枯	4856	架	5020	柁	7987
柄	2549	柁	7179	柏	6017	某	10089
柑	6406	染	10436	柔	7788	栢	7670
柚	3303	柱	19314	柳	2919	柴	24511
柵	3499	查	26313	柁	7099	柿	6878
姆	6440	析	6177	柴	15782	栓	9696
栖	19028	栗	18997	校	7873	栢	11466
株	2757	梅	19781	核	5874	根	7240
格	7957	栽	1609	柁	5424	桂	15712
桃	7100	案	9333	桐	13511	桑	29881
桓	15180	桔	4460	桧	8437	榭	8855
栈	6062	桧	28839	桶	6397	梁	14086
梅	4757	梓	8065	梗	8144	梢	6091
梧	8887	梨	29258	梯	8424	械	4678
梱	18769	梃	21625	棒	16097	棄	8415
棉	4984	棋	8586	棒	7305	棚	7781
棟	3699	森	26167	棲	9244	棺	9074
椀	5722	椅	5805	椋	10124	植	10858
椎	7531	梃	6990	椋	16532	檢	7250
椴	6207	椿	6327	楊	8907	楓	6275
椿	7508	楚	9363	楠	4835	檣	8196
業	11224	楯	4461	楸	9882	極	8639

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
楼	5118	桀	5961	概	7762	桫	8016
榎	9706	榔	8130	榛	10006	構	6772
槌	7808	槍	7482	樣	9745	槓	6273
楓	8679	槽	5765	槌	8616	樗	8988
標	7803	樟	7277	模	7697	榷	6311
橫	6680	櫟	9668	樵	10939	樹	5404
樺	8635	樽	2140	橋	4390	橘	6317
機	4708	橡	8625	檀	10522	檀	6951
檣	7624	櫓	8073	櫛	6267	欄	7253
鬱	8543	欠	34304	次	4	欣	19305
欧	8335	欲	9365	欺	7743	欽	7299
款	9252	歌	9137	歎	10215	歛	11037
止	54926	正	62602	此	5635	武	1623
步	26427	歪	10116	齒	26186	歲	11320
歷	7035	死	18137	殆	9141	殉	9467
殊	6325	殘	9802	殖	8606	毆	7691
段	19886	殺	8158	殼	5771	殿	5512
毅	7437	母	20938	每	8401	毒	24703
比	21315	毘	8495	毛	28839	氏	39581
民	22050	氣	7480	水	19491	氷	30356
永	19459	汜	38365	汀	29411	汁	10055
求	23639	汎	21279	汐	3890	汗	18938
污	26196	汝	22086	江	24657	池	77
汰	26997	汲	11884	決	17877	汽	4211
沃	26251	沈	25590	沌	26702	沓	22409
冲	21887	沙	25136	没	20801	沢	27976
沫	18746	河	12510	沸	4714	油	13699
治	3318	沼	18654	沿	14544	況	3563
泉	28273	泊	14941	泌	3256	法	21264
泡	13747	波	18455	泣	19516	泥	8657

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
注	16466	泰	26864	泳	23737	洋	11130
洗	20641	洛	7393	洞	7145	津	4598
洩	5271	洪	6787	洲	4456	活	19537
派	26591	流	2078	淨	4021	淺	24725
浜	26824	浦	28967	浩	22509	浪	24277
湮	18294	浮	8948	浴	9329	海	17692
浸	6987	消	8709	涌	4431	淚	27820
濤	17942	澆	26641	涯	9759	液	4347
涼	9523	淀	4451	淋	6444	淑	7018
淘	13943	淡	4689	淫	7680	深	25042
淳	7326	淵	8713	混	9329	添	6404
清	5975	渴	9606	濟	5482	涉	4575
涉	5529	溪	8372	渚	473	減	8363
渠	9706	渡	8603	渥	10890	渦	4612
溫	11232	測	6001	港	1238	湊	10209
湖	2185	湘	7651	湛	7225	湧	6042
湯	3906	灣	5734	濕	7798	滿	5285
澆	4047	源	8446	準	7894	溜	4843
溝	9020	溢	4536	溶	5498	溺	4312
滅	1063	滋	7614	滑	6266	淹	9495
滯	10301	滴	3538	漁	7341	漂	5678
漆	4824	漉	10035	漏	6572	演	7441
漕	9263	漠	9182	漠	11611	漣	6413
漫	8776	漬	8306	漸	3589	灌	8862
潔	11022	潛	7031	滂	7363	潤	4939
潮	3501	漬	6328	澄	9832	澗	4858
澱	5098	激	3485	濁	7404	濃	6286
濠	6537	濡	8998	濫	3929	灌	8528
瀕	1055	澣	2295	瀦	2095	瀧	1748
瀨	1033	灘	565	火	23489	灯	17890

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
灰	1	灸	25880	灼	28759	災	8547
炉	7209	炊	18515	炎	12988	炭	16200
点	20703	為	13304	烈	14112	烏	13945
烹	10470	焰	6982	焚	8661	無	4912
焦	9366	然	10516	燒	1954	煉	5969
煎	8995	煙	5052	煤	10792	照	5004
煩	5184	煮	9683	煽	5224	熊	8891
熔	6109	熟	3597	熱	9566	燃	6970
燈	8534	燐	7792	燕	8708	燥	3419
燦	4972	燭	8784	爆	7739	爪	43954
爵	4300	父	42874	爺	10291	爽	8518
爾	7371	片	38193	版	28080	牌	7006
牒	8798	牙	34718	牛	9699	牝	7521
牟	22886	牡	20061	牢	18260	牧	13245
物	25115	牲	7650	特	9998	牽	23804
犀	8179	犧	5345	犬	21573	犯	21028
状	3353	狂	25870	狐	6275	狗	24159
狙	14000	狽	24035	狩	7598	独	21408
狹	5394	狸	19367	狼	22214	狙	13381
猛	27018	獺	8756	猪	2327	猫	7493
猷	9580	猶	10059	獸	5485	猿	9109
獄	9529	獅	4985	獸	9797	獲	6980
玄	24657	率	6382	玉	28828	王	24065
玖	17944	玩	14083	玲	4813	珂	8161
珊	6339	珍	7801	珠	4340	珪	19561
班	1669	現	6700	球	27242	理	15700
琉	6371	琢	5957	琳	7330	琴	10956
琵琶	11446	琶	8966	瑚	7285	瑛	8915
瑞	2982	瑠	6217	瑳	7785	璃	3113
環	7765	璽	11223	瓜	3243	瓢	9602

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
瓦	25832	瓶	17110	甌	6365	甘	22971
甚	8157	甜	10612	生	46716	產	9279
甥	8516	用	44090	甫	23006	田	31204
由	21731	甲	16599	申	18608	男	25377
町	8028	画	28794	界	25474	畏	24116
畑	11233	畔	2833	留	10425	畜	4372
畝	8961	畠	26163	畢	16595	略	7766
畦	11129	番	10763	異	8108	疊	4397
畷	2353	畿	9463	疋	60840	疎	7043
疏	8076	疑	6465	疫	6253	疲	25438
疹	29240	疾	7693	病	2998	症	9194
痔	13420	痕	10439	痘	10177	痛	6462
痢	11413	瘦	6867	痢	2137	療	7508
癌	4767	癒	3161	癬	5425	癩	16731
登	10531	白	27201	百	830	的	11233
皆	7456	皇	26510	皐	16868	皮	7826
皿	30066	盃	8595	盆	9413	盈	5345
益	6450	盜	4501	盛	8682	盟	12070
監	9593	盤	10235	目	44425	盲	17885
直	27728	相	19314	盾	5865	省	12878
眉	5454	看	20321	梟	23595	真	1487
眠	21308	眺	5280	眼	6052	着	15999
睡	6529	督	11330	睦	6553	瞥	5911
瞬	7650	瞭	2987	瞳	6364	矛	21899
矢	24393	知	25496	矧	7017	矩	29018
短	21954	矯	7231	石	54012	砂	24507
研	7475	碎	7607	砥	16255	砦	29547
砧	5359	砲	23330	破	4248	斫	27489
砣	29259	硝	7618	硫	10235	硬	5885
硯	6963	砧	6200	碁	8287	碇	9570

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
碍	10130	碑	7256	碓	4603	碇	6622
碗	5412	碧	9581	碩	8255	確	7962
磁	8092	磐	5113	磨	1919	磯	3950
礁	8405	礎	7144	示	34343	礼	28090
社	13403	祁	24384	祇	8323	祈	15807
祉	15657	祐	7962	祖	20627	祝	23557
神	12045	祢	5087	祥	2663	票	9954
祭	24997	禱	21842	禁	9602	禄	10476
禅	2634	禍	6466	禎	9388	福	8320
禦	10392	禰	8939	禽	7385	禾	32438
秃	31121	秀	26006	私	24223	秋	20845
科	16652	秒	14551	秘	2089	租	6285
秤	28156	秦	11461	秩	15588	称	1462
移	7685	稀	6862	程	9755	税	5174
稔	2191	稗	5521	稚	6359	稜	8060
種	6897	稻	6008	稼	6513	稽	6649
稿	4167	穀	9251	穗	9311	穆	7915
積	7823	穎	6648	穩	2851	穉	11563
穰	4406	穫	8630	穴	20936	究	23645
空	28235	穿	2127	突	19990	窃	5758
窄	28636	窞	7370	窓	8436	窟	6883
窪	10710	窮	10325	窞	9295	窺	9629
竈	11949	立	49746	竜	8097	章	13062
竣	2267	童	24318	豎	9659	端	5500
競	2540	竹	1627	竺	29031	竿	8115
笈	27245	笑	9854	笛	8895	笠	13748
筍	4577	符	18721	第	12831	筐	17272
筆	28359	筭	7357	等	12679	筋	9438
筏	9403	筑	11293	筒	8057	答	8384
策	8536	篋	7780	箇	6877	箔	10492

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
箕	11286	算	10109	管	10087	箏	5221
箭	6762	箱	8069	箸	7422	節	8677
範	12134	篇	5655	築	9958	篠	10202
篤	9381	箆	10630	簡	10808	箴	9653
簾	9419	簿	8721	籍	8491	米	15502
粿	5194	秆	4802	粿	9014	粉	8334
粹	1093	耗	26604	粒	21624	粕	9825
粗	3371	粘	23061	肅	10063	粟	6564
粥	10160	粧	7369	精	6358	糊	3887
糲	7454	糖	7154	糞	5016	糟	5392
糠	5039	糧	7918	糸	50	系	24656
糾	5724	紀	23922	約	30778	紅	21092
紋	18254	納	6774	紐	21702	純	2044
紗	19433	紘	24085	紙	10227	級	11917
紛	28395	素	10443	紡	9396	索	9579
紫	7890	紬	9156	累	19435	細	7696
紳	11353	紹	6998	紺	10744	終	26198
絃	16549	組	26324	絳	8556	結	7642
絞	7863	絡	8572	絢	7548	給	3172
統	7925	繪	21666	絕	8604	絹	3680
繼	4420	綜	9868	綬	9040	維	8823
綱	3599	網	4445	綴	6292	綻	8032
綾	6021	綿	7885	緊	8986	緋	11644
綵	4287	綠	10059	緒	4452	線	7907
締	8547	編	7141	緩	11219	緬	4086
緯	6382	練	9303	緣	11076	繩	9827
縛	11204	縞	4994	縱	1111	縫	10228
縮	10064	績	8591	繁	8657	織	916
繫	4398	繡	6323	織	4205	繕	10612
繭	11108	繰	8310	纂	6558	缶	24556

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
罪	8136	罅	11034	置	10203	罰	9488
署	6250	罵	10219	罷	9327	羅	3537
羊	30	美	30232	群	11386	羨	10328
義	4333	羽	26638	翁	10495	翌	11995
習	30025	翠	8311	翫	9324	翰	5716
翻	2342	翼	8971	耀	2681	老	25226
考	9	者	4239	而	23379	耐	7205
耕	7569	耗	28132	耳	3	耶	8060
耽	9638	聖	11215	聞	7522	聰	5132
聯	6790	聽	5041	職	4675	聾	9584
肇	6765	肉	947	肋	22157	肌	25743
肖	12604	肘	20189	肝	30355	股	13011
肢	16341	肥	11525	肩	14107	肪	8586
肯	17771	肱	27862	育	9011	肴	19647
肺	21395	胃	13529	胆	7506	背	18692
胎	7390	胞	6880	胡	7705	胤	10173
胴	6117	胸	7600	能	10043	脂	6190
脅	1075	脆	25084	脇	27306	脈	1707
脊	749	脚	9127	脫	6150	腦	7628
脹	11259	腎	9044	腐	4238	腔	8539
腕	8570	腫	11187	腰	7208	腸	10939
腹	10255	腺	9466	腿	10500	膏	8917
膚	5850	膜	9924	膝	9949	膨	9518
膳	12087	膿	9588	臆	10262	臟	4703
臣	25330	臥	9994	臨	7037	臭	7719
至	2259	致	16663	白	23114	興	7255
舌	1	舍	12395	舖	2122	舛	9327
舜	5491	舞	7968	舟	24008	航	9471
般	7863	舵	16352	舶	19268	舷	9195
船	27620	艇	4136	艮	23971	良	25465

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
色	7535	艷	6997	芋	25212	芙	31142
芝	23545	芥	20569	芦	14789	芭	28429
芯	26650	花	25781	芳	19170	芸	26671
芹	21640	芽	18572	苻	7559	苑	27105
苓	23213	苔	11394	苗	15914	苛	23765
若	13490	苦	21266	苧	22302	苦	22364
英	17764	茂	21372	茄	25126	茅	20295
荃	20380	茜	9565	茨	24787	茶	18995
茸	3864	草	16218	荆	1596	荏	7752
荒	7719	莊	5438	荷	10102	荻	13601
莞	17534	莫	10569	萊	24251	菅	11840
菊	9061	菌	8445	菓	7324	菖	26864
菜	6547	菟	6924	菩	19952	華	10806
菰	10792	菱	18320	荀	9637	萌	7693
菱	7013	萩	5725	萱	5892	落	27724
葉	17760	律	8457	著	5738	葛	7195
葡	6261	董	10955	葦	2419	葬	7423
葱	11299	葵	9508	葦	8633	蔣	7702
蒐	10390	蒔	10080	蒙	10934	蒜	10070
蒲	10491	蒸	5601	蒼	9738	蓄	8673
蓉	8386	蓋	9946	蓑	9729	蓬	8377
蓮	9781	蔀	9849	蔑	9443	蔓	9754
蔚	11670	蔦	8631	蔭	11809	藏	11063
蔽	10074	蕃	9240	蕉	8030	蕎	9312
蔕	8321	蕨	8715	蕩	6823	蕪	10441
薄	7900	藪	10398	薙	10527	薦	10787
薩	10276	薪	9654	薰	10606	藥	3618
藪	7472	薯	3631	藁	2771	藍	8931
藤	4859	藩	9272	藪	9292	藻	6883
蘇	10899	蘭	10128	虎	19395	虐	5616

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
虛	8400	虜	9574	虞	8957	虫	1314
虹	8486	虻	9082	蚊	8165	蚕	10987
蚤	6435	蛇	7723	蚤	25626	蚩	8966
蛎	15714	蛙	9379	蛤	7469	蛭	5860
蚤	9663	蛸	7738	蛾	3646	蜂	9102
蜘蛛	6570	蜜	8362	蟬	4428	蠟	9746
蝕	6476	蝦	7608	蝶	6868	蠅	8716
融	11951	螺	8711	蟹	10813	蟻	5293
血	26710	衆	5619	行	10168	術	7275
街	7105	衛	2398	衝	8354	衡	6942
衣	795	表	27239	衰	6838	衷	8654
衿	7788	袈	13784	袋	9097	袖	8322
被	20634	袴	14339	袷	17998	裁	7920
裂	11024	裝	3644	裏	9403	裕	5925
補	7131	裘	10620	裡	8463	裳	8322
裸	7578	裾	9726	複	7495	褐	9222
褒	9368	襖	5488	襟	5118	襲	9692
西	3154	要	22830	覆	8986	霸	7939
見	24978	規	8605	視	9968	覲	9170
覺	27623	覽	9954	親	8044	觀	10404
角	25396	解	5062	觸	7652	言	11634
訂	6839	計	17591	訊	9599	討	5707
訓	9348	託	15026	記	8299	訟	11247
訣	20746	訪	9995	設	9093	許	5190
訴	7143	診	8197	註	7062	証	9548
詐	7372	訛	10964	詔	6843	評	6277
詞	7193	詠	8989	詣	7964	試	10246
詩	10155	詫	5737	詮	5877	詰	2744
話	7089	該	9651	詳	7096	誇	6622
譽	7488	誌	8147	認	10440	誓	6505

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
誕	8646	誘	8600	語	8931	誠	9366
誤	7089	說	10682	讀	8697	誰	10956
誹	11983	誼	8739	調	10044	談	9398
請	9880	諫	6008	誨	7679	諒	11907
論	5934	謀	6748	諦	7191	諭	11182
諮	5080	諸	6712	諺	7483	諾	8782
謀	11242	謁	10359	謂	7467	膽	6931
謎	9623	謙	11874	講	10688	謝	9863
謠	3188	謬	9648	謹	11060	識	6830
譜	7163	警	6686	議	7924	讓	6235
讚	4578	讐	10552	谷	7713	豆	3836
豐	10453	豚	14464	象	3567	豪	6747
豹	24981	貌	5604	貝	26511	貞	6564
負	11527	財	8519	貢	697	貧	8253
貨	7278	販	22729	貫	7634	責	11165
貯	9992	貫	4695	貴	5192	買	6122
貸	5931	費	2113	貼	6618	貿	10271
賀	7098	賂	6883	賃	10431	賄	5546
資	5246	賤	6185	賑	8747	賓	9464
贊	9514	賜	9888	賞	9045	賠	9491
賢	8816	賦	6576	質	8376	賭	6844
購	10043	贈	8363	赤	26614	赦	6352
赫	9617	走	23236	赴	9968	起	8791
超	6756	越	8830	趣	8887	趨	7563
足	22242	距	9721	跡	6083	跨	7780
路	9045	跳	7228	踐	8726	踊	7573
蹄	8964	躡	8525	蹴	6657	身	20902
軀	26504	車	22910	軌	9916	軍	10188
軒	9084	軟	22443	轉	11762	軸	6409
輕	16549	較	8693	載	10287	輔	9988

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
輝	6166	輩	9778	輪	5483	輯	10812
輸	9042	輿	7779	轄	11453	轍	9216
轡	10091	辛	12451	辭	5963	辰	30783
辱	4243	農	11136	辺	32060	辻	3302
込	18828	込	18519	迂	26868	迄	25060
迅	24053	迎	16676	近	21545	迦	7326
迹	18127	迫	14047	迭	5235	述	11689
迷	16754	追	19055	退	20669	送	19603
逃	8797	逆	13648	透	13759	逐	20083
遁	13938	途	21728	逗	24045	這	15350
通	7444	逝	10275	速	8726	造	6358
逢	16366	連	7916	逮	9409	週	26771
進	24417	逸	5105	逼	9940	遁	9339
遂	5389	遲	3886	遇	9626	遊	9315
運	5118	遍	8859	過	3302	道	8469
達	2754	違	7949	遜	2869	遠	7632
遣	5744	遙	9709	適	4902	遭	7928
遮	5871	遵	6889	遷	7619	選	8775
遺	8317	遼	3811	避	8699	還	7482
邑	29322	那	23216	邦	24725	邪	12553
邱	4211	郁	8175	郊	8006	郎	4369
郡	2499	部	6189	郭	11116	郵	6674
鄉	9919	都	4640	鄭	10408	酉	17952
酋	3623	酌	6569	配	2805	酎	20942
酒	7750	醉	8098	酢	6090	酪	1585
酬	5812	醇	6068	酸	8067	醇	4979
醞	8413	醐	8341	醒	8638	醜	8539
醜	9026	醬	187	釀	4642	采	20294
采	21529	积	8931	里	21176	重	14281
野	20025	量	9680	金	27284	釘	20550

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
釜	10854	針	10896	鈞	5135	釰	7375
釧	20662	鈍	9562	鈞	5955	鈴	7299
鈷	6271	鉄	6816	鉛	8154	鉢	9747
鉦	7903	鉉	10933	銛	10332	銀	7252
銃	5307	銅	3660	銘	6850	鈔	10628
錢	8736	鋒	9502	鋤	8026	鋪	8865
銳	10741	鋌	7807	鑄	10888	鋸	11131
鋼	3506	鏄	6544	錐	10228	錘	7348
錠	7300	錦	7923	錨	10075	錫	7293
鍊	10842	錯	9251	録	8817	鍋	6092
鍍	7069	鏢	10098	鍛	6725	鍬	9739
鍵	8117	鍾	8730	鎌	7561	鎖	7584
鎗	9004	鎚	5237	鎮	10075	鎚	4243
鏡	8866	鐘	5916	鐙	8678	鑑	7313
鑼	8938	長	29150	門	16097	閃	14747
閉	3448	開	17561	閨	3607	閑	2896
間	6885	閨	4676	閣	655	閤	5120
閣	1889	閱	3905	閭	1247	闕	4161
阜	19388	阪	23333	防	24036	阻	6597
阿	12428	陀	26524	附	9836	降	3011
限	19344	陞	9335	院	7667	陣	5546
除	8489	陷	8717	陪	19302	陰	11150
陳	11224	陵	21706	陶	4306	險	11190
陽	23705	隅	7171	隆	28200	隈	11972
隊	8214	階	19824	隨	2498	隔	7642
隙	1153	際	9364	障	10960	隱	5450
隣	6546	隸	9313	隻	9357	隼	27094
雀	1409	雁	10636	雄	6479	雅	8714
集	19068	雇	10686	雌	9674	雜	4365
雛	8646	離	7881	難	3840	雨	28825

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
雪	27327	雫	30618	霽	10563	雲	13642
雷	10861	電	11217	需	11870	震	7898
靈	10215	霜	8447	霞	11981	霧	9028
青	21302	靖	3310	靜	10575	非	14179
面	21408	革	9351	鞞	2743	靴	9274
鞞	9943	鞍	5408	鞞	5269	鞞	8636
鞭	7337	韓	8235	韮	12086	音	10200
韻	10124	響	10246	頁	5192	頂	8456
頃	7408	項	1510	順	8197	須	10689
預	9019	頑	5690	頰	7374	頓	1396
領	7606	頰	8571	頰	2483	頭	2924
穎	10160	頻	3809	賴	5966	題	11219
額	6618	顎	10938	顏	4586	顛	9306
願	6793	顛	7096	類	7045	風	15691
飛	10157	食	21484	飢	10695	飯	8058
飲	14077	飴	9896	飼	7619	飽	7539
飾	5207	餅	435	養	10535	餌	7357
餐	4170	餓	3367	館	7386	饗	7088
首	18771	香	5416	馨	7718	馳	10332
馴	7915	駁	9350	馱	10149	馱	9040
馱	9375	馱	8675	駐	8157	駒	8537
駕	4476	駿	10353	騎	9495	騷	8805
駟	8667	駟	10366	騰	6625	驚	8398
骨	631	骸	10374	髓	8411	高	8879
髮	6350	髭	9153	鬼	10091	魁	10244
魂	6641	魅	6759	魚	14912	魯	6553
鮎	7623	鮎	10855	鮫	9808	鮭	10209
鮮	11382	鯉	5085	鯖	9102	鯛	7872
鯨	8849	鯨	10601	鰻	7489	鱈	10611
鰻	8664	鰻	10963	鱈	7743	鱈	10703

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₃	Frequency ₄
鱗	7576	鳥	2807	鳩	3949	鳳	8341
鳴	3405	鳶	8273	鴝	6084	鷗	1077
鴛	10308	鴨	4944	鳴	4870	鶯	4613
鴻	10397	鵝	3296	鵠	4078	鷓	2322
鷄	1254	鷺	1990	鷹	7392	鷺	8923
鹼	7505	鹿	10568	麓	9194	麗	8172
麟	3726	麥	7025	麴	8441	麵	9276
麻	8092	磨	5739	黃	12356	黍	12244
黑	26809	默	4368	黛	4116	鼎	12147
鼓	6531	鼠	11317	鼻	8204	齡	8587
龍	8189	-	-	-	-	-	-

Appendix B

Pre-Training Dataset

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
space	324	!	30	”	24	\$	1
%	4	'	1	(36)	34
*	4	+	2	,	13	-	145
.	4	/	1	0	217	1	146
2	113	3	121	4	102	5	101
6	84	7	63	8	107	9	85
:	6	i	5	=	2	い	2
?	43	A	3	B	1	C	4
D	2	E	1	F	2	G	2
H	3	I	6	K	1	L	1
M	1	N	2	O	4	P	2
R	3	S	4	T	1	U	1
W	2	X	2	[5		2
]	6	'	2	a	3	c	2
d	2	e	2	h	1	i	2
l	1	m	3	n	1	o	7
p	1	r	3	s	1	t	10
v	1	w	3	※	2	↑	1
→	2	∞	4	■	3	▼	1
▽	1	○	2	,	480	。	539
々	5	○	2	「	131	」	100
〒	136	あ	163	い	420	う	1
う	217	え	1	え	73	お	61
か	249	が	273	き	98	ぎ	7
く	125	ぐ	12	け	91	げ	20

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
こ	147	ご	15	さ	99	ざ	11
し	346	じ	26	す	198	ず	17
せ	54	ぜ	1	そ	120	ぞ	10
た	419	だ	139	ち	68	っ	261
つ	68	づ	7	て	329	で	337
と	323	ど	58	な	307	に	320
ぬ	6	ね	30	の	614	は	348
ば	31	ひ	13	び	4	び	1
ふ	7	ぶ	11	へ	20	べ	19
ほ	19	ぼ	4	ぼ	3	ま	193
み	40	む	14	め	48	も	180
ゃ	17	や	38	ゅ	2	ゆ	4
よ	19	よ	86	ら	160	り	140
る	296	れ	151	ろ	41	わ	40
ゐ	1	を	297	ん	143		1
ァ	2	ア	58	ィ	13	イ	140
ウ	1	ウ	171	エ	7	エ	27
オ	108	カ	141	ガ	64	キ	98
ギ	25	ク	113	グ	32	ケ	144
ゲ	5	コ	46	ゴ	23	サ	58
ザ	17	シ	269	ジ	34	ス	68
ズ	26	セ	19	ゼ	2	ソ	6
ゾ	3	タ	64	ダ	30	チ	126
ツ	34	ツ	27	ヅ	6	テ	27
デ	21	ト	77	ド	35	ナ	82
ニ	26	ヌ	5	ネ	12	ノ	40
ハ	26	バ	35	パ	9	ヒ	38
ビ	9	ピ	3	フ	33	ブ	20
プ	15	ヘ	3	ベ	8	ペ	4
ホ	18	ボ	16	ポ	2	マ	153
ミ	105	ム	25	メ	24	モ	33
ャ	9	ヤ	91	ユ	23	ユ	13
ヨ	79	ヨ	31	ラ	76	リ	54
ル	38	レ	20	ロ	24	ワ	57
ン	258	ヴ	2	ケ	1	・	57

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
一	110	一	45	丁	1	七	4
万	2	丈	1	三	26	上	32
下	23	不	9	丑	1	世	11
丘	4	兩	5	並	5		1
个	1	中	41	串	2	丸	6
主	13	乃	1	久	7	之	4
乏	1	乘	7	九	11	也	1
乱	3	龜	1	了	1	予	6
争	2	事	27	二	20	于	1
五	7	井	14	些	1	垂	3
亡	1	亢	1	交	3	京	16
亨	2	人	35	仁	3	仇	1
今	23	介	3	仕	5	他	6
付	5	仙	3	代	11	令	2
以	10	仮	1	仰	1	件	3
任	6	企	3	伊	5		1
伏	2	伐	1	休	2	会	34
伝	6	伸	1	似	1	位	2
低	5	住	5	佐	10	佑	2
体	12	何	11	余	1	作	5
	1	佼	1	使	12	來	3
例	8	侍	1	供	4	依	2
価	4	侵	2	侶	1	係	3
俑	1	俗	1	保	6	信	7
修	1	俯	1	俳	1	俺	2
倉	10	個	4	倒	2	借	3
值	2	倫	2	儉	3	偈	2
偉	2	停	1	健	1	側	3
傳	1	傍	1	備	4		1
傷	2	働	3	像	4	僕	10

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
僧	2	儀	1	償	1	儲	1
元	7	兄	2	兆	1	先	11
光	7	免	1	兕	7	党	3
入	21	全	8	八	8	公	10
六	5	共	5	兵	6	其	1
具	5	内	12	円	6	再	3
写	1	富	4	冬	2	冴	1
冷	1	凌	2	凍	1	処	3
凭	1	凸	1	凹	1	出	44
刀	3	分	29	切	8	刑	1
列	2	初	3	判	6	別	8
利	11	制	3	券	1	刻	1
前	27	剂	1	剥	1	剩	1
割	1	劃	1	劇	3	劉	2
劍	1	力	5	功	2	加	8
助	5	努	1	励	1	劳	1
効	3	勇	2	勉	2	勒	1
動	16	務	8	勝	6	募	1
勢	4	勤	3	包	1	匕	1
化	10	北	13	匝	1	匹	1
区	27	医	1	十	3	千	12
半	8	卓	1	南	29	单	3
博	3	卜	1	印	1	危	3
即	1	卵	1	厚	1	原	27
厨	1	厭	1	嚴	2	去	3
参	6	又	2	及	4	友	6
反	7	収	4	叔	1	取	9
受	9	叢	1	口	14	古	6
句	6	叩	1	召	1	可	7
台	10	史	1	右	3	号	1

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
司	2	各	1	合	24	吉	16
同	9	名	13	后	1	吏	1
向	7	君	4	吠	2	否	1
吹	2	吽	2	呂	2	吳	1
告	3	呪	1	味	11	呼	4
命	5	和	11	啞	1	咲	1
哀	1	品	6	哉	1	員	6
哲	2	峻	1	唇	1	唐	1
唯	1	唱	1	啄	1	商	3
問	14	啓	1	啜	1		1
善	1	喉	1	喜	4	營	4
嘆	1	嘲	1	嚙	1	器	4
嚇	1		1		2	四	7
回	7	因	2	団	4	困	1
困	3	凶	5	固	4	国	25
國	1	園	2	土	6	庄	2
在	5	圭	1	地	20	坂	4
均	1	坊	2	坑	1	型	2
埋	2	城	14	域	5	埴	1
執	2	基	6	埼	8	堀	2
堂	1	堤	1	堯	1	報	3
場	17	堺	1	塑	1	塚	1
塩	1	塵	1	塾	1	境	2
墓	2	增	5	墨	1	墳	2
墾	1	壁	3	壕	1	士	4
壬	3	壯	2	声	6	壳	4
変	8	夏	4	夕	1	外	10
多	9	夜	6	夢	2	大	62
天	10	太	12	夫	5	央	4
失	1	奈	8	奉	2	契	1

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
奕	1	奧	3	獎	1	奪	1
奮	2	女	15	奴	3	好	3
如	1	妙	2	妥	1	妻	1
妾	1	姊	1	始	4	委	2
姬	3	姻	1	姿	2	威	1
娑	1	娜	1	婀	1	婆	1
婚	5	婦	1	媛	5	嫁	1
嫌	1	子	31	孔	1	字	3
存	5	孝	1	孟	1	季	2
學	14	孫	4	宅	1	宇	7
守	3	安	11	宋	1	完	1
宗	4	官	4	定	14	寶	1
實	10	客	6	宣	1	室	2
宮	22	害	4	宴	1	宵	2
家	11	容	1	宿	5	寄	4
寅	1	富	9	寓	1	寬	2
寢	3	察	2	寫	1	寮	1
寸	3	寺	7	對	10	壽	1
射	4	將	3	尉	2	尊	3
尋	3	導	1	小	25	少	8
尤	3	就	1	尸	1	尼	1
尾	14	局	2	屁	1	居	5
屈	2	屆	1	屋	11	展	1
屠	1	層	3	屯	1	山	51
岐	4	岡	31	岩	7	岳	1
岸	2	峨	1	峯	1	島	37
崇	2	崎	18	崖	1	嵩	1
嵯	1	川	29	州	6	巡	1
巢	1	工	12	左	2	巨	3
差	3	己	1	已	1	卷	3

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
市	124	布	1	希	2	帝	2
師	3	席	1	婦	5	帳	1
帷	1	常	5	帽	1	幅	1
幡	1	幣	1	干	4	平	17
年	22	幸	3	幻	1	庁	1
広	12	庄	3	序	1	底	1
店	6	府	11	度	18	座	5
庫	3	庭	2	康	3	廢	1
廟	3	延	5	建	1	弁	2
式	18	弓	1	引	10	弘	2
弛	1	弥	2	張	4	強	8
当	6	形	13	彦	2	彩	2
彫	1	彭	1	彰	1	影	6
役	4	彼	18	待	2	律	2
後	19	從	2	得	5	御	6
復	4	循	1	德	6	微	1
心	8	必	17	志	4	忘	1
応	4	忠	2	快	1	念	1
怒	3	怖	3	怛	1	思	22
怠	1	急	4	性	11	恋	1
恐	2	恒	1	恭	2	惠	1
悟	3	悩	1	惡	7	悲	2
情	4	惧	2	慘	1	惰	1
想	4	愁	1	意	15	愛	15
感	6	慈	1	態	4	慢	1
慣	1	慮	1	憶	1	應	1
懲	1	懼	1	戍	3	戍	1
成	12	我	3	戰	3	戮	2
戰	1	戸	11	戾	3	房	2
所	16	手	29	才	1	打	9

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
扌	3	扌	2	批	1	承	1
抄	2	投	3	折	3	拔	4
披	1	抱	2	押	2	抽	1
担	3	拌	2	拐	1	拒	1
拓	2	招	1	抛	3	括	1
拭	1	拷	1	拾	1	拿	1
持	13	指	6	拳	5	挟	1
振	8	捕	1	捨	1	捷	1
掃	1	授	2	掌	1	排	2
掛	2	接	5	控	1	推	3
措	1	掬	1	揄	1	揆	1
揉	1	描	2	提	1	換	1
握	1	揮	1	擲	1	搖	1
損	4	摘	2	摩	4	擊	4
撥	1	撫	1	撮	1		1
攪	2	支	6	攸	1	改	1
攻	1	放	2	政	7	故	1
敏	2	救	1	教	13	敝	2
散	5	敬	3	數	10	整	5
敷	2	文	15	齊	1	齋	1
斗	1	料	5	斬	1	新	20
方	38	施	3	旅	6	族	2
日	49	旦	1	旧	2	旨	1
早	5	旭	3		1	昌	1
明	11	易	2	昔	2	星	3
映	4	春	6	昨	3	昼	2
時	12	眺	1		1	晚	1
景	7	晶	1	智	2	曉	1
暇	1	暖	1	暗	2	曆	1
暮	1	暴	1	曇	1	曙	1

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
曜	1	曠	1	曲	3	書	9
曾	1	替	4	最	8	月	13
有	6	服	1	望	5	朝	7
期	5	木	24	未	1	末	3
本	36	杉	1	李	1	材	3
村	10	東	2	条	3	来	13
杯	1	東	19	杵	4	松	11
析	1	枕	1	林	5	枚	2
果	4	枝	2	柄	1	柏	1
某	1	柘	1	柳	5	柴	1
查	2	柿	2	枋	3	柴	4
栗	2	校	5	株	12	根	10
格	2	栽	1	架	1	桁	2
桂	2	框	1	案	2	桐	1
桑	1	桜	3	榭	2	梁	1
梅	1	梨	4	械	1	梶	1
梲	2	棄	1	棒	2	森	4
棹	1	椀	1	植	2	檢	1
檣	1	檣	1	業	15	楯	2
極	2	楸	1	渠	7	構	1
樣	3	槌	1	標	3	模	3
榷	2	橫	7	樹	2	櫨	2
橋	10	橘	2	機	11		1
檄	1	檣	1	檣	1	櫓	5
次	7	欲	2	歌	4	飲	1
止	2	正	7	此	1	武	7
步	6	歪	1	齒	1	歲	2
歷	2	死	6	殊	1	殘	10
殖	1	毆	2	段	2	殺	9
殼	1	殿	2	毅	1	母	6

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
每	3	比	4	毛	4	毯	1
氏	4	民	3	氣	16	水	22
氷	2	永	2	汁	2	求	5
江	5	池	4	汲	1	決	5
沂	3	冲	3	沙	3	沢	11
油	5	治	7	沼	4	泄	2
泉	3	泊	2	法	14	波	5
泣	1	泥	1	注	3	泰	1
泳	1	洋	2	洗	1	津	12
洲	2	派	2	流	6	淨	3
淺	1	浜	5	浦	3	浩	1
浮	2	海	10	浸	1	消	1
涓	1	涛	1	涯	1	液	1
涼	1	淋	1	深	3	淵	2
混	1	清	9	濟	5	渚	1
減	2	渡	6	温	1	測	2
港	1	湊	1	湖	2	湘	3
湫	1	湯	2	滿	3	準	2
滝	2	滯	1	漂	1	漓	1
演	1	漠	1	漬	1	潁	1
潔	1	瀉	4	潮	2	澤	1
激	2	濤	1	瀟	3	灑	5
火	7	灯	2	灰	1	災	1
炊	1	炭	1	点	6	為	7
烈	3	烏	1	烹	1	焰	1
無	6	焦	1	然	4	燒	4
煎	1	煙	1	煥	1	照	3
煮	3	煽	1	熊	5	熙	1
熟	1	熱	2	熾	1	燕	1
燥	1	燭	1	燮	1	燿	1

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
爭	1	父	9	爺	1	片	1
版	1	牛	1	牡	1	牧	1
物	11	特	7	犬	2	犯	1
狀	1	狂	1	狐	1	狗	1
獨	1	狹	2	狼	1	貓	1
獻	1	猶	1	猿	1	獾	1
獸	1	獲	1	玄	4	率	2
玉	11	王	2		1	玲	2
珠	2	現	10	球	3	理	12
琵琶	1	琴	1	瑕	1	瑛	2
瑠	1	瑳	1	瑾	1	璃	1
璋	1	環	2	璽	2	瓜	1
瓶	1	甘	1	生	29	產	5
甥	1	甦	1	用	14	甫	1
甬	1	田	40	由	4	甲	2
男	10	町	78	画	4	界	6
畑	1	留	1	畝	2	番	4
異	3	疊	1	疏	2	疑	2
疣	1	疱	1	疵	1	病	7
症	1	痔	2	痛	2	瘍	1
瘡	1	療	1	癢	1	癒	1
癬	1	癩	1	癩	10	登	2
白	8	百	1	的	13	皇	3
皐	1	皮	3	盈	1	益	2
盜	1	盛	5	監	2	目	9
直	5	相	13	省	1	県	136
眞	1	真	6	眠	3	眼	2
着	3	睛	1	睡	1	督	1
睦	1	瞳	1	瞥	1	瞬	1
瞰	1	矢	3	知	15	矩	1

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
短	1	石	18	砂	2	研	1
碎	1	砲	1	硬	2	硯	1
碁	1	碇	1	碑	1	碓	1
碁	1	碓	7	磬	2	磨	3
磯	2	礁	1	示	3	社	22
祀	1	祐	2	神	12	祭	2
禁	1	祿	1	禍	1	福	19
禹	1	秀	1	私	16	秋	2
秒	2	秦	2	称	1	移	1
程	1	税	4	種	3	稿	2
穀	1	積	1	穫	1	穴	2
究	1	空	2	突	2	窓	3
窟	1	窺	1	立	11	竜	1
章	3	竣	1	端	1	笑	6
笛	1	笠	1	第	3	笹	3
筆	2	等	4	筋	2	筒	1
答	3	策	5	箇	1	算	4
管	3	箱	1	節	2	範	1
築	2	篠	1	篤	1	簡	3
籌	1	籍	2	米	3	粉	1
粹	3	粒	2	粕	1	粗	1
粘	2	肅	1	粟	4	粳	1
精	1	糧	1	糯	1	系	2
紀	4	紂	1	約	3	紅	3
	1	紋	2	納	4	純	3
紙	3	級	2	素	4	紬	1
細	5	紹	2	紺	1	終	2
組	5	経	2	結	8	絡	4
緋	1	給	2	絨	1	統	2
絳	1	絵	3	絶	2	続	4

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
維	3	綱	1	綺	1	綾	1
緊	1	紘	2	綠	5	緒	1
線	4	締	1	緩	1	練	1
繩	3	縞	1	縱	1	織	1
縲	1	纏	1	缶	1	罍	1
罪	5	置	5	罰	1	署	2
羅	1	羊	1	美	19	群	3
羨	1	義	6	羯	1	羽	4
習	3	翠	1	翰	1	翼	1
老	1	考	10	者	23	而	3
耕	5	耳	2	耿	1	聖	3
聘	1	聞	7	聰	1	聽	4
職	2	聽	1	聾	1	肉	4
肌	1	肝	1	肥	1	肩	2
育	3	胃	1	胆	2	背	5
胡	1	胸	1	能	9	脂	1
脅	1	脇	3	脫	1	腔	1
腕	3	腫	1	腰	1	腹	1
腺	1	膝	1	膾	1	膿	1
臆	3	臙	1	臣	2	臨	4
自	18	至	4	白	3	舍	2
舜	1	舞	4	舟	1	般	1
船	3	良	10	色	3	艸	1
芋	1	芝	1	芦	1	芭	2
花	3	芳	3	芸	2	芹	1
苑	1	苔	1	苗	1	若	7
苦	3	英	4	茂	1	茱	2
茨	1	茶	1	草	1	荒	1
莊	2	荷	1	苳	1	莎	1
菊	1	菌	1	菓	1	華	2

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
菱	1	菱	2	萱	1	落	9
葉	13	著	1	葛	1	葦	1
葬	1	蒲	1	蒼	1	蓄	1
葳	4	蕉	2	蕩	1	蕪	1
蘋	1	蕾	1	薙	1	薩	1
葉	2	薺	1	藏	2	藤	6
蘋	1	虎	1	蛛	1	蛇	1
蜀	1	蜘	1	融	1	蟹	2
	1	血	5	衆	1	行	32
術	7	街	1	衙	1	衛	3
衝	1	衣	2	表	14	袁	3
被	2	裁	3	裝	2	裏	1
裕	1	補	1	輛	1	裳	1
裴	1	製	1	複	2	檔	1
襲	1	西	13	要	14	覆	2
霸	1	見	31	規	3	視	4
覲	1	覺	1	覽	1	親	11
觀	4	角	3	解	7	觸	1
言	30	計	2	討	1	訓	2
託	1	記	4	訛	1	訟	1
訣	1	訪	4	設	3	許	1
詛	2	訴	2	証	2	詔	1
評	4	試	6	話	10	詳	1
譽	1	誌	1	認	5	誓	1
誘	1	語	6	誦	1	說	1
誦	2	誰	4	課	2	誹	1
調	7	談	1	請	1	諏	1
論	5	諫	1	諳	1	謀	1
膳	1	謗	1	謝	1	謳	1
識	3	警	3	議	3	護	2

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
讒	1	谷	10	豆	1	豐	3
豚	1	象	2	豹	1	貝	2
貞	3	負	1	貧	1	貨	1
販	1	貫	1	責	5	貯	1
貫	2	買	4	貸	1	費	3
賀	2	賃	1	資	5	賊	1
贊	1	賞	3	質	11	購	2
贖	1	赤	6	走	4	起	2
超	1	越	3	趙	1	趣	2
足	9	跡	2	路	4	踏	1
蹴	1	踏	1	躊	1	躍	1
身	12	軀	2	車	11	軍	8
軀	7	軀	2	較	1	載	1
輝	2	輪	4	輸	3	轆	1
辭	2	辯	1	農	2	辺	7
込	7	迎	1	近	9	返	4
迫	4	述	2	迷	1	追	7
退	2	送	4	逃	4	逆	1
透	1	途	1	通	8	逞	1
造	3	逢	1	連	9	週	1
進	9	遁	1	遇	1	遊	2
運	8	過	7	道	8	達	4
違	10	遠	2	遙	1	適	4
遭	2	選	2	遺	2	邁	1
邑	1	那	2	邦	1	邪	2
郎	9		1	郡	20	部	17
郭	2	郷	8	都	16	鄒	1
	2	配	7	酒	5	醉	1
酸	1	醜	1	里	3	重	9
野	27	量	3	金	20	釜	1

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
針	2	鐵	3	鉛	1	銀	2
銃	1	錢	1	鋒	1	鋤	1
鎡	1	銳	1	鑄	1	鋼	3
錐	1	錠	1	錦	2	鍊	3
錄	1	鍛	1	鎌	2	鎮	2
鎚	1	鏡	2	鐘	1	長	21
門	5	閉	1	開	11	閑	1
間	22	閑	13	闕	1		1
闕	1	關	1	阜	3	阪	6
防	3	阻	1	阿	1	陀	2
附	1	降	3	限	4	陛	1
院	4	陣	1	除	5	陰	2
陳	1	陸	2	險	3	陽	3
隅	4	隆	3	隈	2	隊	2
階	4	隨	2	隕	1	際	4
障	4	隣	1	隸	1	雄	2
雅	2	集	7	雇	2	雜	2
離	3	難	3	雨	5	雪	4
雫	1	雫	1	雲	5	電	6
霈	1	霞	1	露	1	青	6
靜	7	非	1	面	10	革	2
鞞	1	鞞	1	鞭	1	韃	1
音	8	韻	1	頃	1	項	4
須	4	預	1	頑	1	頓	1
頰	2	頰	1	頰	1	頭	9
頰	3	題	4	額	2	顎	1
顏	10	顛	1	類	1	風	4
飛	2	食	7	飯	2	飲	4
飼	1	飾	3	餅	1	養	3
餐	1	餡	1	館	3	首	4

Key ₁	Frequency ₁	Key ₂	Frequency ₂	Key ₃	Frequency ₃	Key ₄	Frequency ₄
馮	1	香	4	馬	9	馱	1
馱	1	馭	2	駐	2	駿	1
騎	1	騷	1	驚	1	高	22
髮	2	鬘	1	鬼	2	魏	2
魔	4	魚	2	魯	2	鮫	1
鮮	1	鱉	1	烏	3	鳩	1
鳳	1	鳴	1	鴨	1	鴻	1
鵠	1	鵬	1	鶉	1	鶴	2
鷹	1	鸚	2	鸞	1	鹿	10
麗	1	麥	1	麻	2	藟	1
黑	6	鼓	1	軒	1	齡	1
龍	2		1	龕	1	-	-