REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

A COMPARATIVE STUDY ON DEEP LEARNING-BASED HANDWRITTEN TEXT RECOGNITION TECHNIQUES USING A BRAND-NEW TURKISH HANDWRITING DATASET

Osman Furkan KARAKUŞ

MASTER OF SCIENCE THESIS
Department of Computer Engineering
Program of Computer Engineering

Supervisor
Assoc. Prof. Dr. Ali Can KARACA

Co-supervisor
Assoc. Prof. Dr. Ayla GÜLCÜ

March, 2025

REPUBLIC OF TÜRKİYE
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

A COMPARATIVE STUDY ON DEEP LEARNING-BASED HANDWRITTEN TEXT RECOGNITION TECHNIQUES USING A BRAND-NEW TURKISH HANDWRITING DATASET

A thesis submitted by Osman Furkan KARAKUŞ in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 13.03.2025 in Department of Computer Engineering, Program of Computer Engineering.

Assoc. Prof. Dr. Ali Can KARACA
Yildiz Technical University
Supervisor

Assoc. Prof. Dr. Ayla GÜLCÜ
Bahçeşehir University
Co-supervisor

Approved By the Examining Committee

Assoc. Prof. Dr. Ali Can KARACA, Supervisor
Yildiz Technical University _____

Prof. Dr. Banu DİRİ, Member
Yıldız Technical University _____

Asst. Prof. Dr. Zeki KUŞ, Member
Fatih Sultan Mehmet Vakıf University _____

I hereby declare that I have obtained the required legal permissions during data collection and exploitation procedures, that I have made the in-text citations and cited the references properly, that I haven't falsified and/or fabricated research data and results of the study and that I have abided by the principles of the scientific research and ethics during my Thesis Study under the title of "A Comparative Study on Deep Learning-Based Handwritten Text Recognition Techniques Using a Brand-New Turkish Handwriting Dataset" supervised by my supervisor, Assoc. Prof. Dr. Ali Can KARACA. In the case of a discovery of false statement, I am to acknowledge any legal consequence.

Osman Furkan KARAKUŞ

Signature

Dedicated to my family
and my best friend

ACKNOWLEDGEMENTS

I would like to extend my most sincerest thanks to everyone who supported and guided me during the duration of this study.

To start, I owe a deep expression of thanks to my advisor, Assoc. Prof. Dr. Ali Can KARACA, for his constant encouragement, continuous guidance, and thoughtful recommendations, whose contribution to the success of this work cannot be overestimated. His academic background in computer engineering and dedication to academic integrity have immensely enriched my work.

In addition, I extend my deepest regards to my co-advisor, Assoc. Prof. Dr. Ayla GÜLCÜ, whose critical evaluation and insightful discussion played an important role in shaping my work. Her encouragement and expert guidance gave a significant boost in overcoming obstacles in the research work.

I appreciate, in addition, the thesis committee members for taking the time, expertise, and valuable feedback during my defense and during the general work period. Their academic diversity and critical thinking capabilities have encouraged me to maximize academic quality and overall value in my work.

My regards extend to Yildiz Technical University and Bahçeşehir University for providing me with academic and working environments and sources, whose realization helped me produce this work.

Lastly, I have an everlasting feeling of regards for my family, friends, and Computer Engineering Department colleagues for their constant encouragement and motivation. Their trust in my capabilities encouraged and motivated me towards successful work completion.

To everyone who trusts my capabilities and helped shape my character and academic life, I hereby present this work in honor of your trust and confidence in me.

Osman Furkan KARAKUŞ

TABLE OF CONTENTS

LIST OF SYMBOLS viii

LIST OF ABBREVIATIONS x

LIST OF FIGURES xii

LIST OF TABLES xv

ABSTRACT xvi

ÖZET xviii

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Scope and Objectives of the Thesis 2

2 LITERATURE REVIEW 4

2.1 Overview of Handwritten Text Recognition 4

2.2 Traditional Methods 5

2.2.1 Template Matching 5

2.2.2 Statistical Pattern Recognition 6

2.2.3 Feature Extraction Techniques 6

2.2.4 Challenges of Traditional Methods 7

2.3 Modern Approaches 8

2.3.1 Convolutional Neural Networks (CNNs) 8

2.3.2 Recurrent Neural Networks (RNNs) and LSTMs 9

2.3.3 Transformers 9

2.4 Datasets in the Literature 10

2.4.1 IAM Handwriting Dataset 10

2.4.2 CVL Handwriting Dataset 10

2.4.3 Other Notable Datasets 11

3 DATASET 12

3.1 Data Collection 12

3.1.1 Sources of Turkish Text Documents . . . 13
3.1.2 Criteria for Document Selection . . . 13
3.2 Data Annotation . . . 16
3.2.1 Labeling Methods . . . 16
3.2.2 Labeling Methods, Tools, and Techniques Used . . . 18
3.2.3 Dataset Statistics . . . 20

4 APPLICATION OF LINE AND WORD SEGMENTATION ALGORITHMS 23
4.1 BN-DHRISTI Algorithm . . . 23
4.2 Genetic Algorithm in Kazakh Dataset . . . 24
4.3 A* Algorithm . . . 25
4.4 Word Segmentation with a Morphological-Based Approach . . . 27
4.5 Line Segmentation with ViT . . . 28
4.5.1 Vision Transformer for Line Segmentation . . . 28
4.5.2 DETR Framework . . . 28
4.5.3 Adaptation for Line Segmentation Task . . . 29
4.6 Experimental Setup and Evaluation . . . 35
4.6.1 Evaluation Metrics . . . 35
4.6.2 Visualization of the Results . . . 35

5 APPLICATION OF TEXT RECOGNITION ALGORITHMS 38
5.1 Vertical Attention Network (VAN) . . . 39
5.1.1 Architecture Overview . . . 39
5.1.2 Innovations in VAN . . . 40
5.1.3 Performance Evaluation . . . 40
5.2 TrOCR: Transformer-based Optical Character Recognition . . . 40
5.2.1 Architecture Overview . . . 41
5.2.2 Innovations in TrOCR . . . 42
5.2.3 Performance Evaluation . . . 42
5.3 HTR-VT: Handwritten Text Recognition with Vision Transformer . . . 43
5.3.1 Architecture Overview . . . 43
5.3.2 Innovations in HTR-VT . . . 44
5.3.3 Performance Evaluation . . . 45
5.4 Word Beam Search Decoding . . . 45
5.4.1 Architecture Overview . . . 46
5.4.2 Innovations in WBS . . . 47
5.4.3 Performance Evaluation . . . 47
5.5 Enhancing VAN Decoder with a Transformer Layer . . . 47
5.5.1 Modified Architecture Overview . . . 48

5.5.2 Advantages of the Transformer Layer 49
5.5.3 Performance Evaluation 49

6 RESULT **51**
6.1 Results 51
6.2 Line Segmentation Results 51
6.2.1 Comparison of Line Segmentation Methods 51
6.2.2 Analysis of Line Segmentation Results 52
6.3 Text Recognition Results 52
6.3.1 Comparison of Text Recognition Methods 53
6.3.2 Analysis of Text Recognition Results 53
6.4 Discussion 54

REFERENCES **55**

PUBLICATIONS FROM THE THESIS **60**

LIST OF SYMBOLS

α	Learning rate hyperparameter
β	Regularization hyperparameter in loss functions.
γ	Threshold hyperparameter for segmentation confidence.
σ	Standard deviation in Gaussian filtering during preprocessing.
θ	Model parameter vector in deep learning architectures.
Δ	Difference operator used in evaluating performance metrics.
Ω	Set of all possible handwritten text segments.
λ	Weight decay coefficient in optimization.
ρ	Correlation coefficient for performance analysis.
ϕ	Feature extraction function for character recognition.
$\mathcal{D}$	Dataset used for training and evaluation.
$\mathcal{L}$	Loss function used in deep learning models.
$\mathcal{M}$	Model architecture applied for Handwritten Text Recognition.
$\mathcal{S}$	Segmentation mask representing detected text lines.
$P(x)$	Probability distribution of handwriting features.
$H(X)$	Entropy measure of a handwritten dataset.
$\mathcal{F}$	Set of all handwritten forms in the dataset.
$\mathcal{L}$	Set of all segmented text lines.
$\mathcal{W}$	Set of all words in the dataset.
σ_{lines}	Standard deviation of the number of lines per form.
σ_{words}	Standard deviation of the number of words per form.
$\mathcal{I}$	Input image for segmentation.
$\mathcal{S}$	Set of segmented text lines.

$\mathcal{P}$	Population in the Genetic Algorithm.
$f(x)$	Fitness function in the Genetic Algorithm for segmentation.
$h(n)$	Heuristic function in the A* algorithm.
IoU	Intersection over Union, a metric for evaluating segmentation accuracy.
$\mathcal{T}$	Transformer model used for text recognition.

LIST OF ABBREVIATIONS

AI	Artificial Intelligence.
ANN	Artificial Neural Network.
BiLSTM	Bidirectional Long Short-Term Memory.
CTC	Connectionist Temporal Classification.
DETR	Detection Transformer, an object detection framework.
GPU	Graphics Processing Unit.
HTR	Handwritten Text Recognition.
IAM	The IAM Handwriting Database.
LSTM	Long Short-Term Memory.
NLP	Natural Language Processing.
OCR	Optical Character Recognition.
ReLU	Rectified Linear Unit, an activation function.
ResNet	Residual Neural Network.
RNN	Recurrent Neural Network.
SOTA	State-of-the-Art.
TR-OCR	Transformer-based Optical Character Recognition.
VIT	Vision Transformer.
DBN	Deep Belief Network.
FCN	Fully Convolutional Network.
GAN	Generative Adversarial Network.
IoU	Intersection over Union (used for segmentation evaluation).
k-NN	k-Nearest Neighbors algorithm.
RIMES	Reconnaissance et Indexation de Manuscrits pour l'Expertise des Systèmes.

STN	Spatial Transformer Network.
GT	Ground Truth.
ID	Identification Number.
NXXX	Format used for labeling dataset entries (e.g., N001-01-05).
QA	Quality Assurance.
TXT	Text file format used for storing labeled data.
HTR-VT	Handwritten Text Recognition with Vision Transformer.
SAM	Sharpness-Aware Minimization.
VAN	Vertical Attention Network.
WBS	Word Beam Search.
A*	A* Pathfinding Algorithm.
BN-DHRISTI	Bengali Document Handwriting Recognition Intelligent System for Text Identification.
GA	Genetic Algorithm.
TLDM	Text Line Detection Metric.

LIST OF FIGURES

Figure 2.1 Overview of the handwritten text recognition process, highlighting key stages such as preprocessing, line segmentation, feature extraction, and text recognition. 5
Figure 2.2 An example of template matching applied to handwritten text recognition. [18] 6
Figure 2.3 Statistical pattern recognition techniques, including HMMs and k-NN, used in handwritten text recognition. [19] 7
Figure 2.4 Examples of feature extraction techniques such as HOG, zoning, and Fourier transforms. [20] 8
Figure 3.1 An example of a Turkish handwritten text from the dataset, identified as N01-053-029. 14
Figure 3.2 Another example of a Turkish handwritten text from the dataset, identified as N01-021-015. 15
Figure 3.3 The program sequentially designates lines and words, exemplified by the transformation from N01-047-023 to N01-047-023-01, with error detection and correction for inaccurately labeled initial words. 17
Figure 3.4 The text file containing word tags, with incorrectly segmented words indicated by 'err'. 18
Figure 3.5 Flow chart illustrating the labeling process using the custom-built labeling program. 19
Figure 3.6 Distribution of lines and words per form. 21
Figure 4.1 Example of line segmentation using BN-DHRISTI algorithm. . 24
Figure 4.2 Example of line segmentation using Genetic Algorithm. 25
Figure 4.3 Example of line segmentation using A* Algorithm. 26
Figure 4.4 Example of word segmentation using the morphological-based approach with OpenCV. Each bounding box encloses an isolated word. 28
Figure 4.5 Overview of the DETR architecture, adapted for text line segmentation. The transformer encoder-decoder processes image features to segment lines effectively. [44] 29

Figure 4.6 Image preprocessing steps: original image (top left), image with fixed size patches and flattening (top right), linear projection applied on flattened image patches (bottom left), positional embedding added to image patches (bottom right). 30
Figure 4.7 Example of line segmentation using Vision Transformer. 31
Figure 4.8 Architecture of Vision Transformer (ViT).[11] 32
Figure 4.9 Steps to attain the Attention Value [46]. 33
Figure 4.10 Scaled Dot-Product Attention in Multi-Head Attention consists of several attention layers running in parallel. [27] 34
Figure 4.11 Example of annotated handwritten text dataset used for segmentation. 35
Figure 4.12 Segmentation results using Genetic Algorithm. 36
Figure 4.13 Segmentation results using YOLO-based Bangla method. 36
Figure 4.14 Segmentation results using A* Path Planning. 36
Figure 4.15 Segmentation results using Vision Transformer-based approach. Provides clean and accurate line segmentation with no visible errors. 37
Figure 5.1 Architecture of VAN. The FCN encoder extracts feature maps from the input paragraph image, and the attention module generates vertical attention weights to focus on specific text lines. The LSTM decoder predicts the character sequence for each line.[22] 41
Figure 5.2 Architecture of TrOCR. The image Transformer encoder processes the input image patches, while the text Transformer decoder generates the output text sequence. The model is trained end-to-end, leveraging pre-trained image and text Transformers.[24] 43
Figure 5.3 Architecture of HTR-VT. The CNN feature extractor generates spatially-aware tokens, which are processed by the ViT encoder with span masking to produce character predictions. The entire model is trained with CTC loss.[23] 45
Figure 5.4 Architecture of Word Beam Search. The prefix tree constrains the beam search to valid dictionary words, while the language model provides contextual scoring. The algorithm dynamically switches between word-state and non-word-state to handle arbitrary non-word characters.[25] 48

Figure 5.5 Enhanced VAN Architecture. The LSTM layer processes line features, while the Transformer layer captures long-range dependencies and contextual relationships. The model is trained end-to-end using CTC loss. 50

LIST OF TABLES

Table 3.1 Summary of Dataset Composition 20
Table 3.2 Distribution of Lines and Words per Form 21
Table 6.1 Performance comparison of line segmentation methods on the Turkish dataset. 52
Table 6.2 Performance comparison of text recognition methods on the Turkish dataset (from worst to best). 53

ABSTRACT

A COMPARATIVE STUDY ON DEEP LEARNING-BASED HANDWRITTEN TEXT RECOGNITION TECHNIQUES USING A BRAND-NEW TURKISH HANDWRITING DATASET

Osman Furkan KARAKUŞ

Department of Computer Engineering
Master of Science Thesis

Supervisor: Assoc. Prof. Dr. Ali Can KARACA
Co-supervisor: Assoc. Prof. Dr. Ayla GÜLCÜ

Handwritten Text Recognition (HTR) is an important field in document analysis, with its applications ranging from the conservation of historical documents to creating assistive technology. Despite significant advances in HTR for extensively used languages, low-resource languages, including Turkish, have been poorly represented, largely due to the lack of rich datasets and specifically crafted solutions. In an attempt to tackle the specific challenge posed by handwritten texts in Turkish, this thesis creates a novel dataset and adapts state-of-the-art models according to the specific characters (e.g. ö,ç,ü,ğ) of handwriting in Turkish.

The thesis begins with developing an in-depth Turkish handwritten dataset with careful line-by-line annotations. The dataset will be made publicly available. With 1.610 forms, 6.941 lines, and 60.491 words, it is designed for training and testing HTR systems with ease. Dataset preparation involves processes such as collection, preprocessing, and annotation, and subsequently, augmentation, in an attempt to make high-diversity, high-quality samples available, representing real-life diversity.

For line segmentation, a new state-of-the-art approach is proposed using Vision Transformers (ViTs), outperforming conventional approaches including Genetic Algorithm, A* Path Planning, and BN-DRISHTI. With an Intersection over Union

(IoU) of 0.925, a 0.8501 detection accuracy, and a 0.8720 character accuracy, our Vision Transformers-based line segmentation model proves to be effective in dealing with documents with complex layout and handwriting diversity.

For text recognition, a Vertical Attention Network (VAN) is boosted with a Transformer layer in its decoder, providing a powerful integration of long-term dependencies and contextual relations, enhancing its accuracy in character and word recognition. With a character error rate (CER) of 0.1913 and a word error rate (WER) of 0.0557, our model outperforms baseline architectures such as TrOCR, HTR-VT, and SimpleHTR, and enables its capacity for dealing with long-term dependencies and contextual relations, enhancing its accuracy in character and word recognition.

The results of this research highlight the potential of modern architectures such as vision transformers and hybrid LSTM-transformer models to address the challenges of handwritten text recognition. By providing a robust dataset and innovative solutions, this thesis contributes to the advancement of HTR for low-resource languages, paving the way for future research and practical applications.

Keywords: Handwritten Text Recognition, Turkish Handwriting, Vision Transformers, Line Segmentation,Dataset Preparation.

ÖZET

YEPYENI BIR TÜRKÇE EL YAZISI VERI SETI KULLANILARAK DERIN ÖĞRENME TABANLI EL YAZISI METIN TANIMA TEKNIKLERI ÜZERINE KARŞILAŞTIRMALI BIR ÇALIŞMA

Osman Furkan KARAKUŞ

Department of Computer Engineering
Yüksek Lisans Tezi

Danışman: Assoc. Prof. Dr. Ali Can KARACA

El Yazısı Tanıma (HTR), belge analizinin önemli bir alanıdır ve tarihi belgelerin korunmasından yardımcı teknolojilerin geliştirilmesine kadar geniş bir uygulama yelpazesine sahiptir. Yaygın olarak kullanılan dillerde HTR konusunda önemli ilerlemeler kaydedilmiş olmasına rağmen, Türkçe gibi diller büyük ölçüde yeterli veri kümelerinin eksikliği ve dolayısıyla el yazısı problemlerinin özel çözümlerinin geliştirilememesi nedeniyle geri planda kalmıştır. Bu tez, Türkçe el yazılarının oluşturduğu özel zorlukları ele almak amacıyla yeni bir veri kümesi oluşturmayı ve en ileri seviye modelleri Türkçe el yazısının kendine özgü karakteristiğine uygun şekilde dönüştürmeyi hedeflemektedir.

Tez, öncelikle Türkçe el yazısı belgelerinin ayrıntılı ve kamuya açık bir derlemesini oluşturarak başlamaktadır ve bu belgeleri titizlikle satır satır ek açıklamalarla zenginleştirilmiştir. 1.610 form, 6.941 satır ve 60.491 kelime içeren bu veri kümesi, HTR sistemlerinin kolayca eğitilmesi ve test edilmesi için tasarlanmıştır. Veri kümesinin hazırlanması; toplama, ön işleme, ek açıklama ekleme ve ardından veri artırma süreçlerini içermektedir. Bu süreçler, gerçek hayattaki çeşitliliği yansıtan yüksek kalitede ve geniş çeşitliliğe sahip örnekler üretmeyi amaçlamaktadır.

Satır bölütleme için, Görü Dönüştürücüleri (ViTs) kullanılarak mevcut yöntemleri geride bırakan yeni bir üst düzey yaklaşım önerilmektedir. Genetik Algoritma, A* Yol Planlama ve BN-DRISHTI gibi geleneksel yaklaşımlardan daha iyi performans

gösteren bu model, %92,5'lik Birleşim Üzerinden Kesişim (IoU) skoru, %85,01'lik algılama doğruluğu ve %87,20'lik karakter doğruluğu ile karmaşık düzenlere ve el yazısı çeşitliliğine etkili bir şekilde uyum sağlamaktadır.

Metin tanıma aşamasında ise, Dikey Dikkat Ağı (VAN) yapısına bir dönüştürücü katmanı eklenerek güçlendirilmiş ve bu sayede uzun vadeli bağımlılıkları ve bağlamsal ilişkileri daha iyi entegre eden bir model geliştirilmiştir. %19,13'lük karakter hata oranı (CER) ve %5,57'lik kelime hata oranı (WER) ile modelimiz, TrOCR, HTR-VT ve SimpleHTR gibi temel mimarileri geride bırakarak uzun vadeli bağımlılıkları ve bağlamsal ilişkileri başarıyla işleyebilen yüksek doğruluklu bir sistem sunmaktadır.

Bu araştırmanın sonuçları, Görü Dönüştürücüler gibi modern mimarilerin ve hibrit LSTM-Dönüştürücü modellerinin el yazısı tanımada karşılaşılan zorlukları ele alma potansiyelini vurgulamaktadır. Güvenilir bir veri kümesi ve bu veri kümesi için yenilikçi çözümlerin sunulduğu bu tez çalışması, düşük kaynaklı diller için HTR alanına önemli bir katkıda bulunmakta ve gelecekteki araştırmalar ile pratik uygulamalar için bir temel oluşturmaktadır.

Anahtar Kelimeler: El Yazısı Tanıma, Türkçe El Yazısı, Görüntü Dönüştürücüler, Satır Bölütleme, Veri Kümesi Hazırlama.

1
INTRODUCTION

1.1 Motivation

The digitization of handwritten text has become a critical area of research in document analysis, driven by its transformative potential in applications such as historical preservation, automated archival systems, educational tools and assistive technologies [1]. While significant progress has been made in natural language processing (NLP) and computer vision for typed-text domains, the recognition of handwritten text remains a formidable challenge [2]. Handwritten text recognition (HTR) systems must deal with immense variability in writing styles, document degradation, and linguistic nuances, which complicate segmentation and transcription tasks [2].

For widely spoken languages like English, French, or Mandarin, large-scale datasets such as the IAM Handwriting Database [3] and RIMES [4] have enabled significant breakthroughs in HTR accuracy. However, for linguistically rich but computationally underrepresented languages like Turkish, the absence of comprehensive annotated datasets stifles progress [5]. This discrepancy perpetuates a technological gap: While high-resource languages benefit from state-of-the-art models, low-resource languages struggle with basic digitization tasks, hindering their integration into global NLP frameworks.

Turkish handwritten text recognition presents unique challenges that compound the complexity of HTR:

- **Script Characteristics**: The agglutinative nature of Turkish, with its long words and frequent diacritics (e.g., *ç, ş, ğ, ı*), demands precise segmentation and contextual understanding [6].
- **Cursive and Style Variability**: Handwriting styles in Turkish often feature connected characters and slant variations, which complicates the detection of line and word boundaries [7].

- **Data Scarcity**: There is no publicly available large-scale Turkish HTR data set, forcing researchers to rely on non-representative or synthetic data [8].
- **Legacy Documents**: Historical and administrative documents in Turkey suffer from ink bleed, fading, and uneven layouts, necessitating robust preprocessing techniques [9].

These challenges are not only academic. The ability to process Turkish handwritten text has tangible societal impacts, including preserving centuries-old manuscripts, digitizing legal and educational records [10]. Yet, the lack of tailored solutions for Turkish leaves these applications unrealized. This thesis seeks to bridge this gap by addressing the critical challenges of Turkish HTR through the development of a novel dataset and the adaptation of state-of-the-art models to the unique characteristics of Turkish handwriting.

1.2 Scope and Objectives of the Thesis

This thesis is dedicated to developing a strong handwritten text recognition system specific for Turkish documents, in order to overcome the specific challenge presented by the Turkish language and script. The work involves developing a dataset application of line segmentation and text recognition methods, and performance evaluation, contributing in general to HTR development in low-resource language settings. The specific objectives of this thesis are summarized below.

Dataset Development:

We developed a high-quality, publicly available corpus of Turkish handwritten documents, enriched with detailed line-level annotations that facilitate training and evaluating HTR systems. Furthermore, we applied a range of preprocessing techniques (e.g., noise filtering and image enhancement) to improve document quality for annotation and analysis [8].

Line Segmentation:

We adapted Facebook's end-to-end object detection model for effective text line extraction in Turkish handwritten documents, paying special attention to the unique challenges posed by the Turkish script [11, 12].

Text Recognition:

We implemented and trained optical character recognition (OCR) models specifically for Turkish handwritten text, leveraging state-of-the-art deep learning methods to achieve high accuracy and reliability [7].

Experimental Evaluation:

We conducted a thorough experimental evaluation to assess the effectiveness of our proposed methodologies, which included a comparative analysis of various approaches to line segmentation and text extraction [13].

Contribution to the Field:

We present a comprehensive dataset and provide actionable insights for Turkish HTR, thereby addressing the particular complexities of handwritten Turkish text. Additionally, we developed a model that can be adapted for other low-resource languages, contributing to future advancements in HTR research [5]. By fulfilling these objectives, we aim to push the boundaries of Turkish handwritten text recognition and promote its application in practical, real-world scenarios.

2
LITERATURE REVIEW

The literature review examines previous studies and approaches related to handwritten text recognition, focusing on dataset preparation, line segmentation, and text recognition methods. By analyzing significant progress and ongoing challenges in this domain, this chapter situates the current research within the broader context of HTR, highlighting existing gaps and opportunities for advancement. The discussion is structured into four main sections: (1) an overview of HTR, (2) traditional methods, (3) modern approaches, and (4) datasets in the literature. This organization provides a clear and comprehensive understanding of the field, emphasizing the need for improved methods tailored to Turkish handwritten text.

2.1 Overview of Handwritten Text Recognition

Handwritten text recognition is a critical task in document analysis and pattern recognition, aiming to convert handwritten text into machine-readable formats. The field has evolved significantly, driven by advancements in machine learning, computer vision, and natural language processing. Figure 2.1 provides a visual schema of the HTR process, illustrating the key stages involved in recognizing handwritten text.

The HTR process typically involves the following steps:

- **Preprocessing:** Enhancing the quality of handwritten images through noise reduction, binarization, and skew correction [14].
- **Line Segmentation:** Dividing the text into individual lines for further processing [15].
- **Feature Extraction:** Identifying meaningful patterns in the text, such as edges, curves, and character shapes [16].

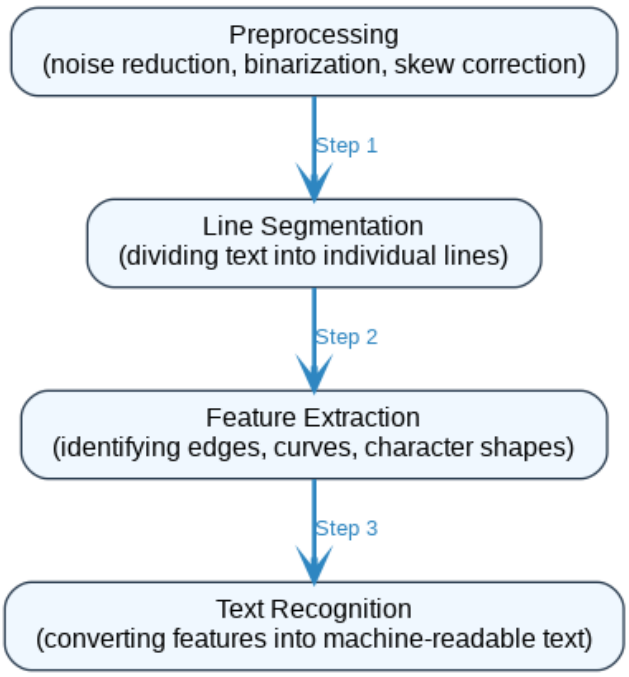


Figure 2.1 Overview of the handwritten text recognition process, highlighting key stages such as preprocessing, line segmentation, feature extraction, and text recognition.

- **Text Recognition:** Converting the extracted features into machine-readable text using recognition models [17].

Each of these stages presents unique challenges, particularly when dealing with diverse handwriting styles, noisy documents, and underrepresented languages like Turkish. The following sections delve into the traditional and modern methods used to address these challenges.

2.2 Traditional Methods

Early approaches to HTR relied on rule-based and feature-driven techniques. While innovative for their time, these methods were limited by their inability to handle variability in handwriting and their reliance on handcrafted features. This section explores the key traditional methods and their limitations.

2.2.1 Template Matching

Template matching was one of the earliest techniques used in HTR. It involves comparing input text images against pre-stored templates of characters. Figure 2.2 illustrates an example of template matching applied to handwritten text recognition. While this method is simple to implement and works well for limited and consistent

datasets, it struggles with scalability and is highly sensitive to variations in handwriting style, rotation, and noise [3]. For instance, it performs poorly with cursive handwriting or documents with significant noise.

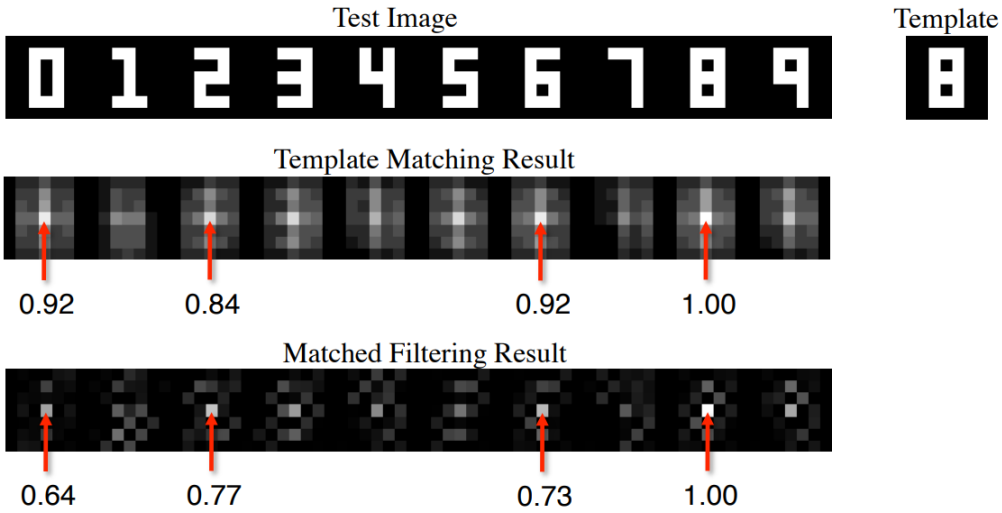


Figure 2.2 An example of template matching applied to handwritten text recognition. [18]

2.2.2 Statistical Pattern Recognition

Statistical methods, such as k-Nearest Neighbors (k-NN) and Hidden Markov Models (HMMs), emerged as alternatives to template matching. These methods model the probabilistic relationships between characters and strokes, offering better generalization. Figure 2.3 shows an example of statistical pattern recognition techniques, including HMMs and k-NN, used in handwritten text recognition. HMMs, for instance, represent handwriting as a sequence of states, capturing temporal dependencies [19]. While these methods are more robust than template matching and can model sequential dependencies, they still rely on handcrafted features and struggle with noisy or complex data.

2.2.3 Feature Extraction Techniques

Feature extraction techniques, such as Histograms of Oriented Gradients (HOG) and Fourier Transforms, were used to transform raw pixel data into meaningful representations. These methods focused on identifying edges, curves, and patterns in handwritten text. Figure 2.4 provides examples of feature extraction techniques such as HOG, zoning, and Fourier transforms. These techniques improved recognition accuracy over raw pixel-based methods and provided meaningful

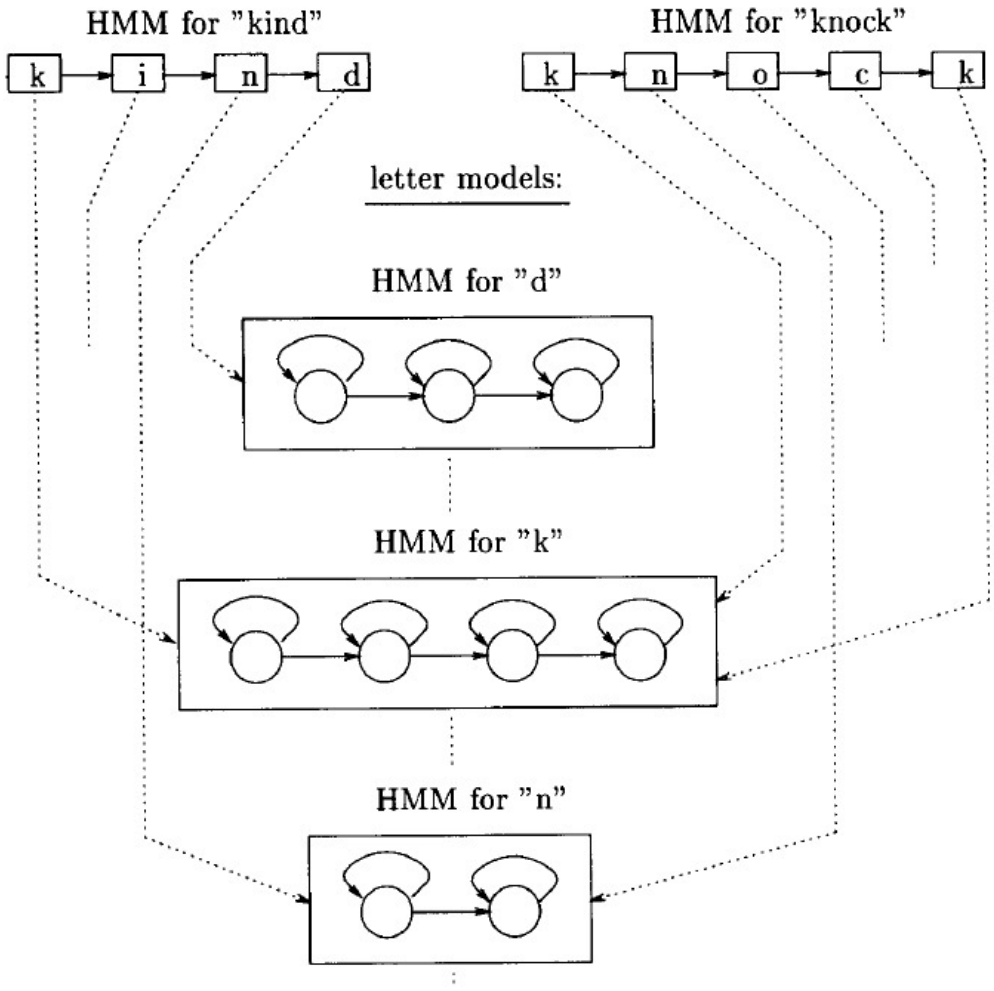


Figure 2.3 Statistical pattern recognition techniques, including HMMs and k-NN, used in handwritten text recognition. [19]

representations for character shapes [20]. However, they were labor-intensive to design and had limited adaptability to new handwriting styles.

2.2.4 Challenges of Traditional Methods

Despite their contributions, traditional methods faced several challenges. They lacked adaptability due to their reliance on predefined templates or handcrafted features, making them unsuitable for diverse handwriting styles [21]. Additionally, they were highly sensitive to noise, with variations in writing tools and document quality often leading to recognition errors. The complexity of cursive writing, with its connected characters, posed significant challenges for segmentation and recognition. These limitations paved the way for modern approaches, particularly those based on deep learning, which revolutionized the field of HTR.

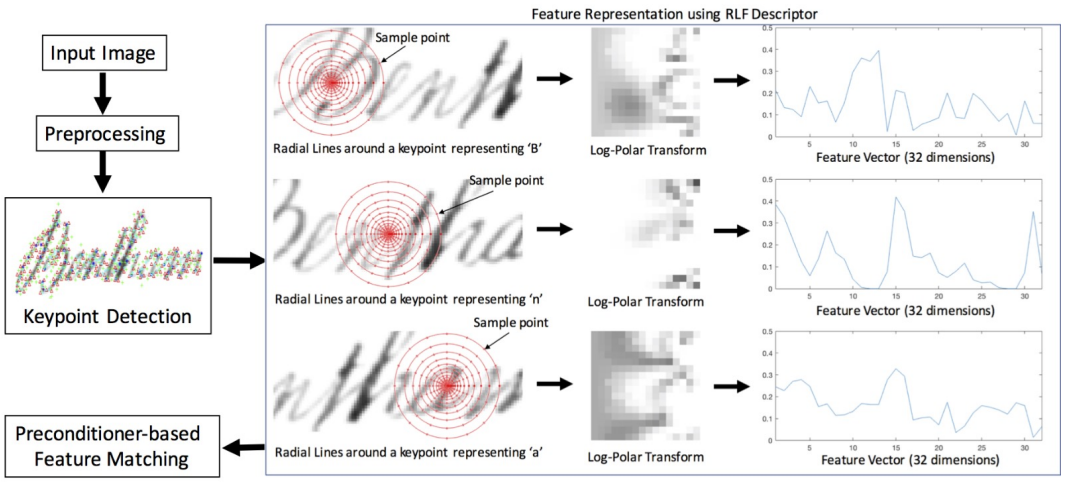


Figure 2.4 Examples of feature extraction techniques such as HOG, zoning, and Fourier transforms. [20]

2.3 Modern Approaches

The advent of deep learning has transformed HTR [22–25], enabling the development of models that automatically learn features and handle complex handwriting variations. This section explores the key modern approaches, including Convolutional Neural Networks [17] (CNNs), Recurrent Neural Networks [26] (RNNs), and Transformers [27].

2.3.1 Convolutional Neural Networks (CNNs)

CNNs have become the backbone of modern HTR systems due to their ability to automatically extract spatial features from handwritten text images. They excel at capturing intricate patterns and are robust to variations in handwriting style and noise [28].

Strengths:

- Automatic feature extraction.
- Robustness to handwriting variations.
- Scalability to large datasets and diverse scripts.

Challenges:

- High data requirements for training.
- Computational complexity in designing and tuning architectures.

2.3.2 Recurrent Neural Networks (RNNs) and LSTMs

RNNs [26], particularly Long Short-Term Memory [29] (LSTM) networks, are widely used for modeling sequential dependencies in handwriting. They are effective for recognizing connected characters and capturing contextual information [30].

Strengths:

- Effective for sequential data modeling.
- High accuracy in recognizing cursive and connected handwriting.

Challenges:

- High computational cost.
- Dependence on large labeled datasets.

2.3.3 Transformers

Transformers have recently gained popularity in HTR due to their ability to model long-range dependencies using self-attention mechanisms. They offer parallelization and flexibility, making them suitable for diverse handwriting styles and scripts [27].

Strengths:

- Parallelization of input sequences.
- Effective modeling of long-range dependencies.
- Flexibility for multilingual and diverse handwriting styles.

Challenges:

- High data and computational requirements.
- Complexity in training and fine-tuning.

Despite the significant improvements seen in state-of-the-art handwritten text recognition algorithms, their performance is most determined by diversity and

accuracy in datasets utilized. In order for deep learning algorithms to produce successful training, it is critical that datasets not only be thorough and accurately annotated but also represent a variety of handwriting, language, and types of documents. As a result, datasets become a critical consideration in model performance evaluation and comparative model studies in HTR. The following section will have a thorough discussion of several high-profile datasets that have become well-known in HTR studies, with an objective of explaining their contribution to the field.

2.4 Datasets in the Literature

Datasets play a crucial role in the development and evaluation of HTR systems. This section discusses some of the most widely used datasets, their characteristics, and their contributions to HTR research.

2.4.1 IAM Handwriting Dataset

The IAM Handwriting Dataset [3] is a benchmark dataset for English handwritten text recognition. It includes samples from 657 writers, with annotations at the line, word, and character levels.

Key Features:

- Diverse handwriting styles.
- Detailed annotations for line, word, and character segmentation.

Challenges:

- Variations in handwriting styles and document quality.

2.4.2 CVL Handwriting Dataset

The CVL Handwriting Dataset [31] focuses on English and German handwriting. It includes high-resolution images and annotations for line and word segmentation.

Key Features:

- Multilingual content (English and German).
- High-quality scanned images.

Challenges:

- Limited to Latin scripts.

2.4.3 Other Notable Datasets

Other datasets, such as RIMES [4] and HKR [32], have contributed to HTR research by addressing specific languages and scripts. However, the lack of a comprehensive Turkish handwritten text dataset remains a significant gap in the literature.

This chapter has provided a comprehensive review of the literature on handwritten text recognition, covering traditional methods, modern approaches, and key datasets. While significant progress has been made, challenges remain, particularly for underrepresented languages like Turkish. The next chapter focuses on the development of a novel Turkish handwritten text dataset and the adaptation of state-of-the-art methods for line segmentation and text recognition.

3
DATASET

HTR systems are strongly dependent on the quality and size of the datasets they are trained and tested on. Proper preparation of a dataset will involve some key steps: data collection, preprocessing, annotation, and augmentation. Each of these steps is crucial for a dataset that will reflect variability and diversity in real-life handwritten documents.

Below, we suggest the methodology for developing the novel dataset of Turkish handwritten texts. The preparation of the dataset starts with diverse handwritten samples gathered from various sources, reflecting variability in handwriting styles and document quality. Data gathering is supported by preprocessing techniques that enhance the quality of the images and prepare them for further processing.

Annotation is a crucial step in the preparation of the dataset, and it creates the ground truth required for training and testing the HTR models. Manual data annotation is done in the process, so the accuracy and consistency of the annotations are ensured, as the data is gathered at the line and word levels. Data augmentation techniques artificially enlarge the dataset to alleviate the issue of small data and improve the stability of the models learned from it.

The dataset prepared will bridge the existing gap in the resources for Turkish handwriting text and be a beneficial resource for further research in this subject area. By documenting every stage of the process of dataset preparation, this chapter allows not only the reproducibility of the research but also presents the best way of preparing high-quality datasets for handwritten text recognition.

3.1 Data Collection

The success of handwritten text recognition systems hugely depends on the quality and quantity of datasets on which they are trained and on which their performances are evaluated. In this work, we have collected a dataset of 14 individual literary

works, each chosen to provide a broad range in genre, author, form, and number of words. Further, we describe our process for data collection and storage and direct our interest to the composition of the dataset and the characteristics of it.

3.1.1 Sources of Turkish Text Documents

The dataset contains a corpus of 14 individual literary works, each of which was assigned a special Document ID. The literary works contained in this dataset reflect a wide variety of genres, including Novel, Fiction, Psychology, Spirituality, and History. The corpus contains clear author identifications, which reflect the specific authors of these literary works. Interestingly, the dataset has a total of 1,610 forms, 6,941 lines, and 60,491 words, with an average of about 49.7 words per text, which reflects the granularity and depth of each work. The total word count of all these literary works provides an insight into the overall length and complexity of the dataset.

The dataset was developed in cooperation with nearly 200 contributors who were involved in the collection, annotation, and validation of the handwritten texts. Their diligent effort ensured the accuracy and consistency of the dataset, making it a valuable asset for both training and evaluation of handwritten text recognition systems. The large-scale environment and heterogeneous data distribution of the dataset provide a solid foundation for the advancement of research in handwritten text recognition, particularly for low-resource languages like Turkish [8].

3.1.2 Criteria for Document Selection

The data collection process involved several stages to ensure the robustness and diversity of the dataset. Initially, literary works were selected based on their genre and historical significance. Each text was then transcribed and annotated to facilitate subsequent recognition tasks. Special attention was given to preserving the original formatting and stylistic elements of the handwritten texts, which are crucial for accurate text recognition [33].

To illustrate the diversity and quality of the collected handwritten texts, we provide two examples from the dataset. Figure 3.1 shows a sample text identified as N01-053-029. This image highlights the complexity of handwritten Turkish text, capturing various stylistic nuances and formatting challenges. Similarly, Figure 3.2 presents another sample text, identified as N01-021-015, further demonstrating the variability in handwriting styles and the meticulous attention given to preserving the authenticity of each document.

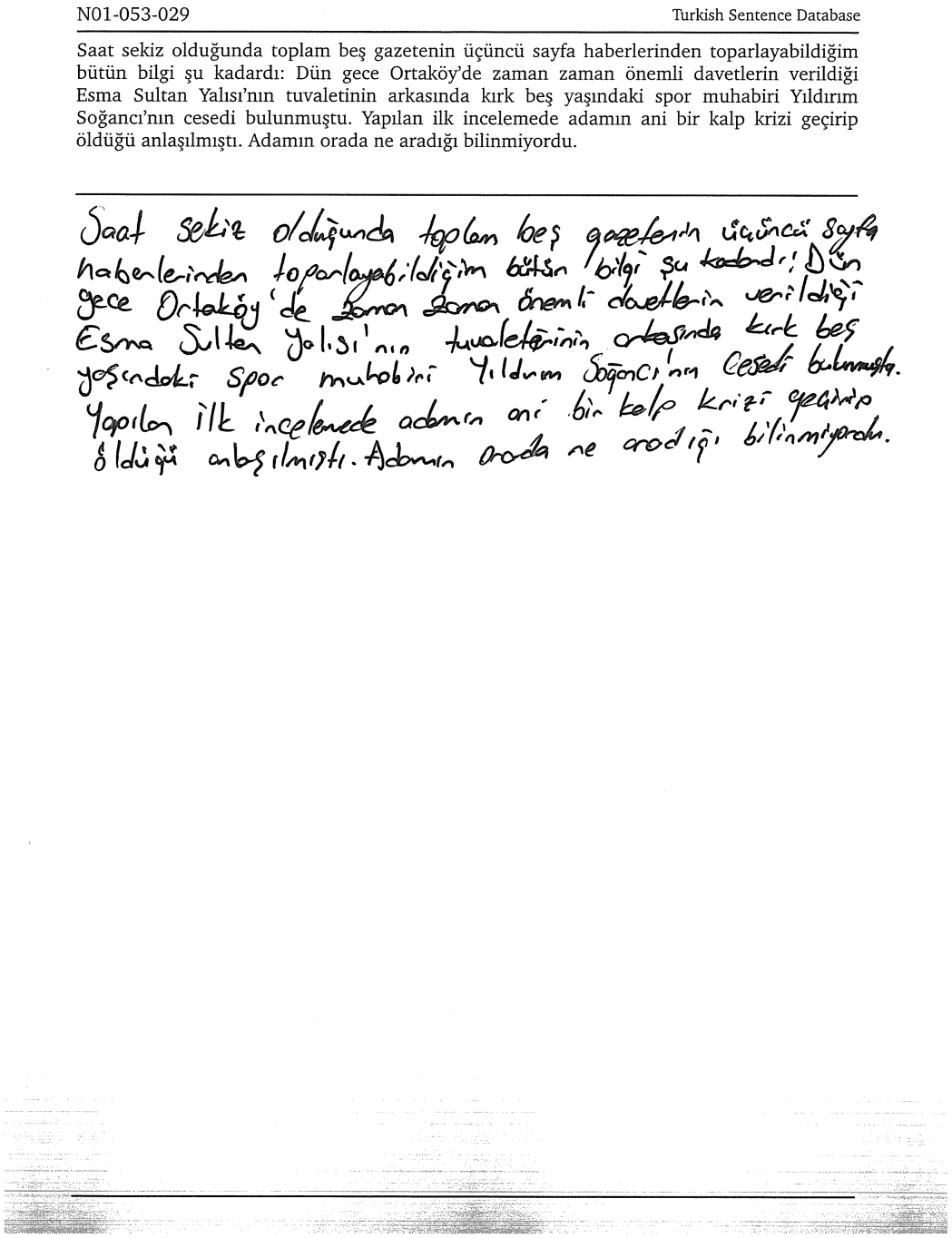


Figure 3.1 An example of a Turkish handwritten text from the dataset, identified as N01-053-029.

To maintain a high standard of data quality, all collected texts underwent a rigorous verification process. This included manual proofreading and validation by multiple annotators to ensure consistency and accuracy in the annotations. Additionally, the dataset was stored in a structured format, with metadata detailing the genre,

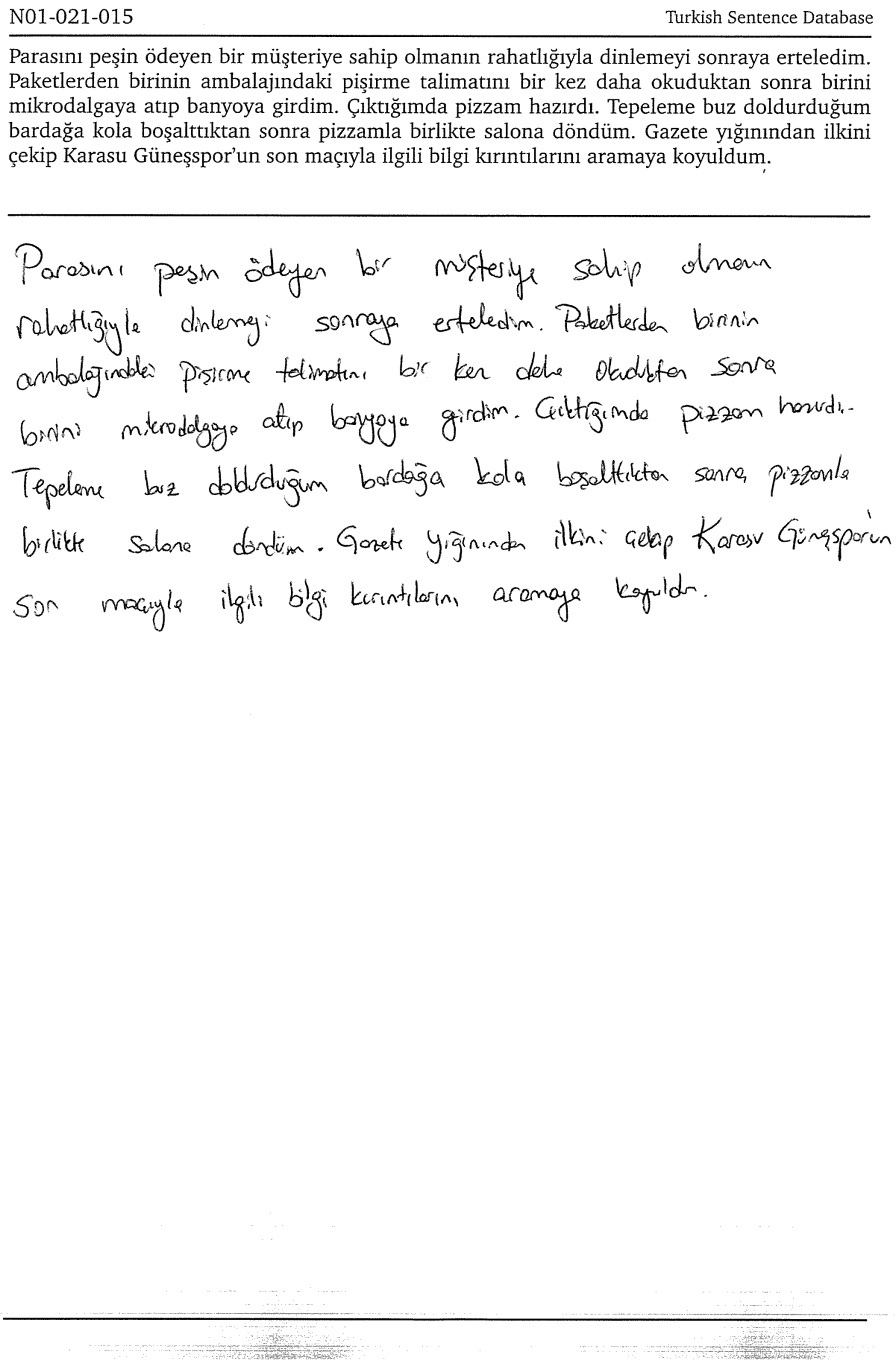


Figure 3.2 Another example of a Turkish handwritten text from the dataset, identified as N01-021-015.

author, word count, and text form for each literary work. This structured storage approach not only facilitates efficient data retrieval and management but also supports reproducibility and transparency in research.

3.2 Data Annotation

Accurate data annotation is an important phase in designing effective handwritten text recognition (HTR) systems. The accuracy in annotations reflects directly in the performance of trained models over the dataset, and high accuracy and uniformity in the annotation process is therefore a must. For this purpose, a careful annotation process was utilized for marking a recently developed dataset for handwritten texts in Turkish.

The data annotation involved a sequence of phases, including handwritten text segmentation in terms of words and lines, accuracy checking for segmentation, and individual label assignment for each word. Manual checking and label assignment for each section were performed via a group of expert annotators, employing high accuracy and uniformity for each stage. Not only does such a meticulous approach preserve the language structure of the content, but high confidence in compiled datasets for training and testing HTR systems is guaranteed too [34].

To ease the annotation process, a custom labeler tool was developed specifically for effective checking and label assignment for each word. The tool played a key role in regulating the flow of annotation, and efficient checking and label assignment for each word could be performed in a convenient manner. Error checking and fixing features were included in the tool, and efficiency and accuracy for the annotations increased.

In this section, we detail an in-depth analysis of the techniques, tools, and processes involved in data annotation in this work. We introduce the custom label scheme, segmentation validation criteria, and methods for uniformity and accuracy in labeling, in addition to providing a description for storing the final labels in a specific format and techniques for effective retrieval and analysis of the data. In this thorough annotation scheme, we strive to build a powerful and reliable dataset for supporting high-end handwriting text recognition studies.

3.2.1 Labeling Methods

First, we identified excerpts from 14 different books. We used the NXXX-XX-XX format for labeling. To simplify using numbers, the 'N001' in '001' refers to the book or literary work with ID 001. The '01' after the hyphen represents the number used to distinguish the handwriting of the person/author from whom we received the handwritten text. The remaining part represents the number used to distinguish the excerpt taken from the book or literary work whose ID is given at the beginning. In our effort to ensure the precise and consistent labeling of a handwritten Turkish text

dataset, a dedicated team of ten individuals was employed. Each team member was assigned the responsibility of labeling individual words within the dataset. These labeled words played a crucial role in the automated construction of text lines, thus preserving the original linguistic structure of the content [34].

To carry out this labeling process effectively, we designated each controller to manage about 1610 pages of forms for analysis. These forms, along with their segmented lines, were systematically input into a custom-built labeling program developed for this purpose. The program's workflow involved displaying segmented lines and segmented word images on the screen, prompting the controller to verify the accuracy of word segmentation. In cases where incorrect segmentation was identified, the program added an 'err' label to the word, signifying the necessity for correction to achieve precise labeling. Conversely, if the word was correctly segmented, the program prompted the controller to input the corresponding label for that word. Subsequently, the label was added to the word, and an 'ok' marker was appended alongside the label entry.

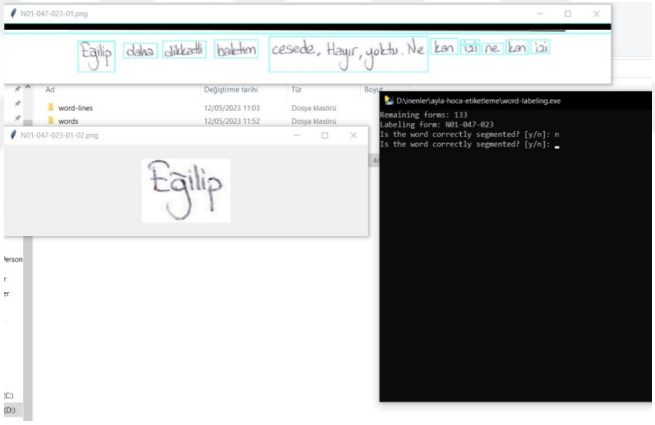


Figure 3.3 The program sequentially designates lines and words, exemplified by the transformation from N01-047-023 to N01-047-023-01, with error detection and correction for inaccurately labeled initial words.

For example, in Figure 3.3, N01-047-023, the first form labeled N01-047-023, was marked as N01-047-023-01. N01-047-023-01, the first word, was marked as N01-047-023-01-01. Where a word, N01-047-023, was marked wrong, then the program marked that and proceeded to the next one. Conversely, in case the remaining part of a word was segmented and marked appropriately, correct mark, for example, 'Eğilip' was added to the program and marked appropriately. In this way, marking became effective, and double marking at the level of a line became unnecessary, as our system generated labels for a line individually in relation to a label address for a word.

After completing initial marking, a careful cross-checking exercise was performed

in order to make certain that accuracy and uniformity in labels existed. In this, and in comparing labels marked by two persons, in order to maintain integrity in the corpus,

```
words/N01-001-001/N01-001-001-07-01.png ok Kısa
words/N01-001-001/N01-001-001-07-02.png ok boylu
words/N01-001-001/N01-001-001-07-03.png ok adam
words/N01-001-001/N01-001-001-07-04.png ok sayılmam
words/N01-001-001/N01-001-001-07-05.png ok ama
words/N01-001-001/N01-001-001-07-06.png ok karşımdaki
words/N01-001-001/N01-001-001-07-07.png err
words/N01-001-001/N01-001-001-08-01.png ok benden
words/N01-001-001/N01-001-001-08-02.png ok daha
words/N01-001-001/N01-001-001-08-03.png ok uzundu.
words/N01-001-001/N01-001-001-08-04.png ok İnce
words/N01-001-001/N01-001-001-08-05.png ok bir
```

Figure 3.4 The text file containing word tags, with incorrectly segmented words indicated by 'err'.

The finalized labels were stored in an a.txt file format in an organized format with the name of the respective image file and its respective label. Having an organized format facilitated efficient retrieval and analysis of the data at the subsequent stages of the research.

3.2.2 Labeling Methods, Tools, and Techniques Used

To facilitate a thorough understanding of the labeling process, we developed a custom labeling tool designed to streamline the verification and annotation of handwritten text. This section describes the methods, tools, and techniques employed in the annotation process.

3.2.2.1 Labeling Program

The labeling program played a central role in our annotation workflow. The program's interface displayed segmented lines and words, prompting annotators to verify the accuracy of word segmentation. The tool's functionalities included:

- **Displaying Segmented Lines and Words**: Annotators reviewed each segmented line and word to ensure accurate segmentation.
- **Error Detection**: The program automatically marked incorrectly segmented words with an 'err' label.
- **Label Input**: Annotators input the correct label for each word, which the program recorded and marked with an 'ok' marker.

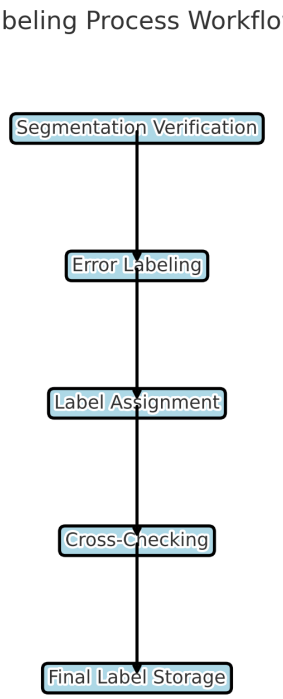


Figure 3.5 Flow chart illustrating the labeling process using the custom-built labeling program.

3.2.2.2 Workflow and Quality Assurance

The workflow for the labeling process involved several key steps:

1. **Segmentation Verification**: Annotators verified the segmentation of lines and words displayed by the program.
2. **Error Labeling**: Incorrectly segmented words were marked with an 'err' label for further correction.
3. **Label Assignment**: Correctly segmented words were labeled by the annotators and marked with an 'ok' label.
4. **Cross-Checking**: A comprehensive cross-checking procedure was implemented to ensure the consistency and accuracy of the labels.

The flow chart in Figure 3.5 illustrates the detailed steps of this process, providing a visual representation of the labeling workflow from segmentation verification to final label storage.

3.2.2.3 Storage and Format

The final labels were stored in a structured .txt file format. Each entry in the file included the name of the corresponding image file and the associated label, ensuring organized and efficient data retrieval. An example of the text file format is shown in Figure 3.4, highlighting the systematic approach to label storage and management.

By employing these methods, tools, and techniques, we ensured the accurate and consistent annotation of the handwritten Turkish text dataset, thereby supporting the reliability of subsequent text recognition tasks.

3.2.3 Dataset Statistics

The dataset collected for this study comprises a substantial number of forms, lines, and words, meticulously annotated to support robust handwritten text recognition research. This section presents detailed statistics on the dataset composition and the distribution of data.

3.2.3.1 Dataset Composition

The dataset consists of 1610 forms, each containing multiple lines of handwritten text. In total, the dataset includes 6941 lines and 60491 words. The following table summarizes the key components of the dataset:

Table 3.1 Summary of Dataset Composition

Component	**Count**
Number of Forms	1610
Number of Lines	6941
Number of Words	60491

The forms were sourced from 14 distinct literary works, each representing a diverse range of genres, authors, and styles. This diversity ensures that the dataset encompasses a wide variety of handwriting styles and linguistic structures, making it a valuable resource for training and evaluating handwritten text recognition systems.

3.2.3.2 Distribution of Data

The distribution of data across the forms, lines, and words is crucial for understanding the dataset's variability and comprehensiveness. Figure 3.6

illustrates the distribution of lines and words per form, highlighting the dataset's heterogeneity.

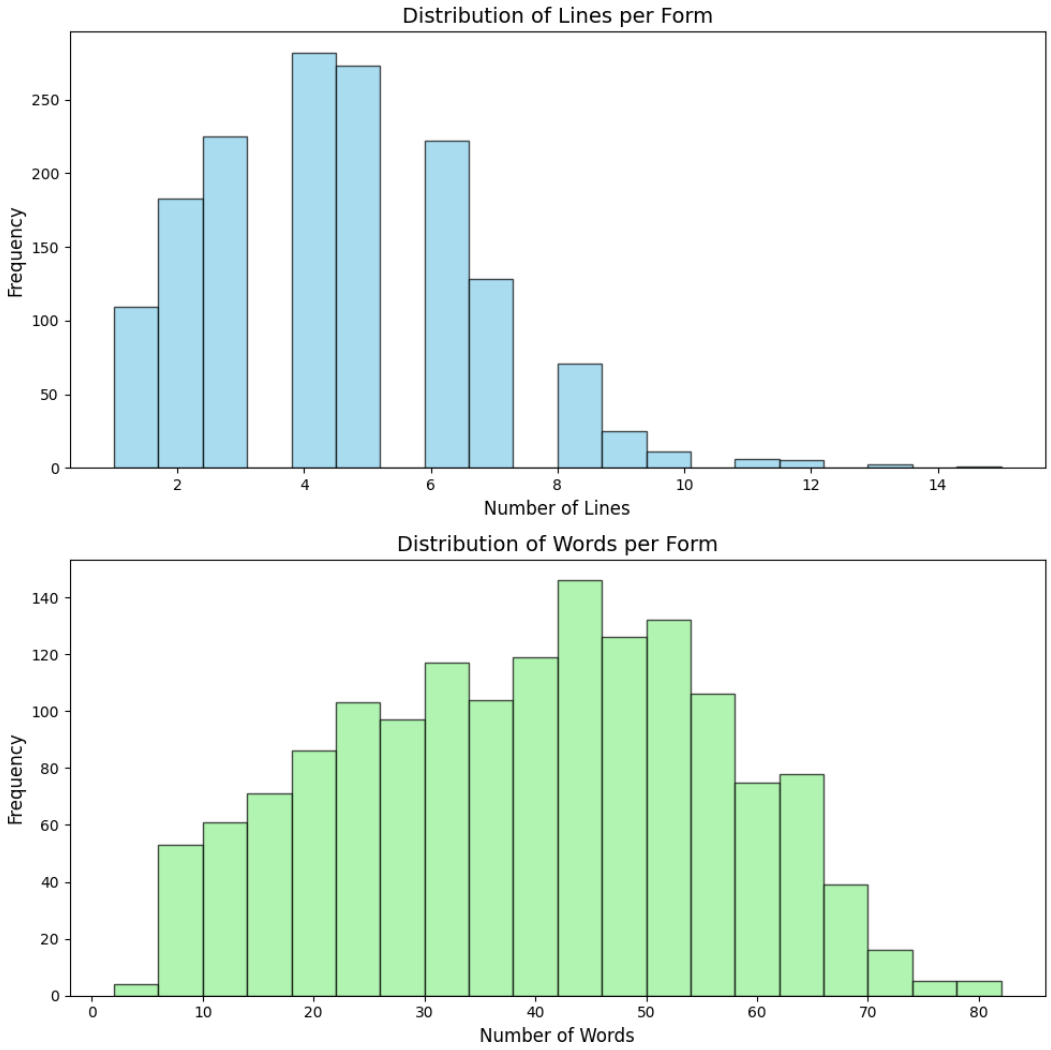


Figure 3.6 Distribution of lines and words per form.

The average number of lines per form is approximately 4.5, while the average number of words per form is approximately 38.9. These averages reflect the dataset's granularity, capturing both short and long excerpts from the literary works. The following table provides a detailed breakdown of the distribution statistics:

Table 3.2 Distribution of Lines and Words per Form

Statistic	**Lines per Form**	**Words per Form**
Average	4.5	38.9
Median	4	40
Standard Deviation	2.1	16.5

The variability in the number of lines and words per form reflects the natural

differences in handwriting styles and text length across the dataset. This diversity is essential for training models that are robust to different types of handwritten text.

In summary, the dataset's comprehensive composition and diverse distribution of data provide a solid foundation for advancing handwritten text recognition research. By encompassing a wide range of forms, lines, and words, the dataset supports the development and evaluation of models capable of handling various handwriting styles and textual complexities.

4
APPLICATION OF LINE AND WORD SEGMENTATION ALGORITHMS

The segmentation of handwritten text into lines and words is a crucial step in the preparation of datasets for handwritten text recognition. In this study, we employed three different algorithms for line segmentation and utilized OpenCV for word segmentation. This section details the methodologies and tools used for segmentation, providing insights into their application and performance.

We implemented three distinct algorithms for line segmentation to ensure robust and accurate segmentation of the handwritten text lines:

4.1 BN-DHRISTI Algorithm

The BN-DHRISTI [35] algorithm leverages a combination of binarization and distance transform techniques to accurately identify text lines. This method is particularly effective in handling complex layouts and varying line heights, making it suitable for diverse handwriting styles. The process begins with converting the document image to binary form, which helps in distinguishing text regions from the background. This is followed by applying a distance transform to highlight the text line structures, facilitating the extraction of individual lines. Figure 4.1 provides a visual example of the BN-DHRISTI algorithm applied to our dataset, showcasing its capability to segment text lines effectively. The steps involved in the algorithm are outlined below:

- **Binarization**: Convert the grayscale image to a binary image using Otsu's method, which helps in separating the text from the background.
- **Distance Transform**: Apply a distance transform to the binary image to enhance the text line structures.

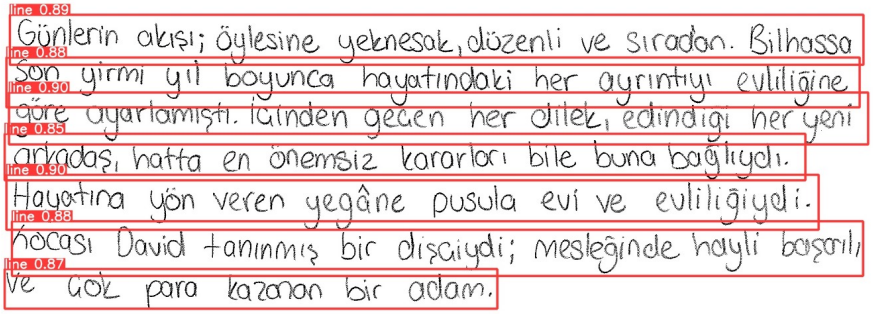


Figure 4.1 Example of line segmentation using BN-DHRISTI algorithm.

- **Watershed Algorithm [36]**: Use the watershed algorithm on the distance-transformed image to segment the text lines.
- **Post-processing**: Apply morphological operations to refine the segmented lines and remove any noise.

This method's robustness lies in its ability to adapt to various handwriting styles and document layouts, making it a versatile tool for text line segmentation. The underlying process, detailed in Algorithm 1.

Algorithm 1 BN-DHRISTI Algorithm for Text Line Segmentation

Require: Grayscale image I
Ensure: Segmented lines L
1: Convert I to binary image B using Otsu's method
2: Apply distance transform to B to get distance image D
3: Apply watershed algorithm on D to segment text lines
4: Refine segmented lines using morphological operations
5: Return segmented lines L

4.2 Genetic Algorithm in Kazakh Dataset

The Genetic Algorithm[32] (GA) employs evolutionary strategies to optimize the segmentation process, effectively dealing with skewed and overlapping lines. This method is based on the principles of natural selection and genetics, where an initial population of solutions evolves over generations to find the optimal segmentation. Figure 4.2 provides an example of line segmentation using the Genetic Algorithm, illustrating its effectiveness in handling complex layouts and overlapping text lines. The steps involved in the algorithm are outlined below:

- **Initial Population**: Generate an initial population of possible segmentations.

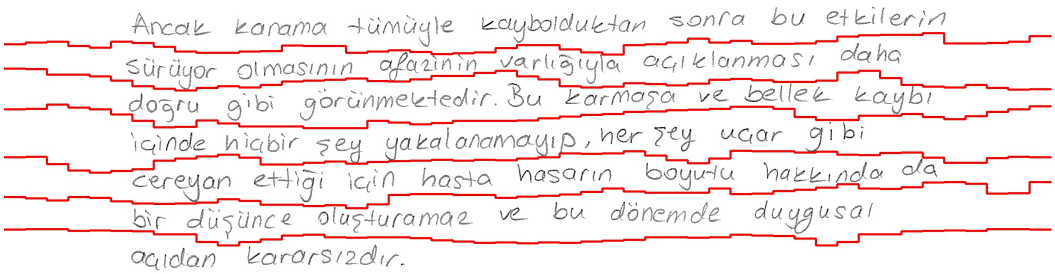


Figure 4.2 Example of line segmentation using Genetic Algorithm.

- **Fitness Function**: Evaluate the fitness of each segmentation based on criteria such as alignment accuracy and text continuity.
- **Selection**: Select the best-performing segmentations for reproduction.
- **Two-Point Crossover and Boundary Mutation [37]**:
  - **Crossover:** Use a two-point (dual-point) crossover operator. In this method, two random segmentation boundary points are selected from each parent solution. The segments between these points are then exchanged between the parents to generate offspring. This recombination preserves the integrity of contiguous text regions and allows promising segmentation structures to merge.
  - **Mutation:** Apply a boundary mutation operator that randomly adjusts the positions of one or more segmentation boundaries. This involves shifting a boundary by a few pixels to introduce variability, enabling the algorithm to explore nearby segmentation configurations and refine the results, especially in cases with skewed or overlapping lines.
  - **Iteration**: Repeat the evaluation and reproduction process over several generations until the optimal segmentation is achieved.

The GA's adaptability to different handwriting styles and its ability to handle complex text layouts make it a powerful method for text line segmentation. The detailed steps of the Genetic Algorithm are provided in Algorithm 2, outlining the iterative process of population evolution, fitness evaluation, and selection to achieve optimal segmentation.

4.3 A* Algorithm

The A*[38] algorithm is a pathfinding and graph traversal algorithm that has been adapted for text line segmentation. By modeling the text lines as paths in a grid,

Algorithm 2 Genetic Algorithm for Text Line Segmentation

Require: Initial population P, fitness function F, maximum generations G
Ensure: Optimal segmentation S

```
Generate initial population P
for each generation g in G do
    Evaluate fitness of each individual in P using F
    Select individuals for reproduction
    Apply crossover and mutation to create new population
    Replace old population with new population
end for
Return the best individual S from the final generation
```

the A* algorithm efficiently identifies and segments lines, even in densely packed text regions. Figure 4.3 illustrates an example of line segmentation using the A* algorithm.

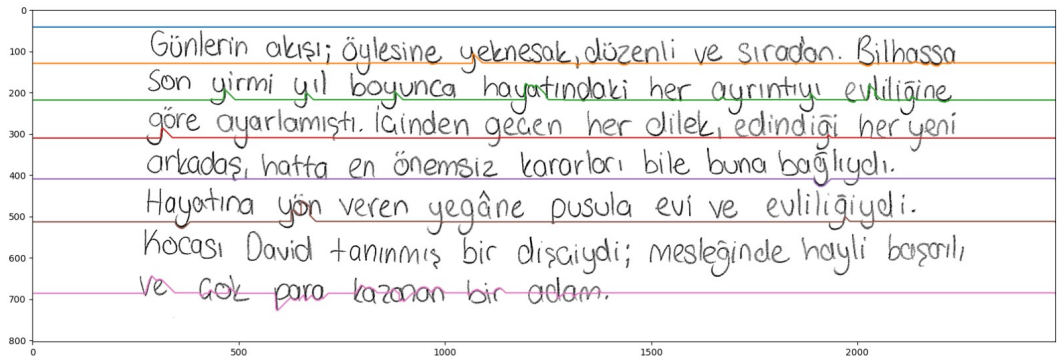


Figure 4.3 Example of line segmentation using A* Algorithm.

- **Grid Creation**: Convert the document image into a grid where each cell represents a pixel.
- **Heuristic Function**: Define a heuristic function to estimate the cost of reaching the end of a line from each pixel.
- **Pathfinding**: Use the A* algorithm to find the optimal path that traces each text line, minimizing the cost defined by the heuristic function.
- **Segmentation**: Extract the paths identified by the algorithm as individual text lines.

The A* algorithm's ability to handle complex text regions and its efficiency in pathfinding make it an effective method for text line segmentation. The detailed steps of the algorithm are outlined in Algorithm 3, where the open and closed sets are iteratively updated to find the optimal path from the start node to the goal node.

Algorithm 3 A* Algorithm for Text Line Segmentation

Require: Start node S, goal node G
Ensure: Path P from S to G
Initialize open set O with S
Initialize closed set C as empty
while O is not empty **do**
    Select node n in O with lowest f-score
    if n is G **then**
        Return path P reconstructed from n
    end if
    Move n from O to C
    for each neighbor m of n **do**
        if m is in C **then**
            Continue
        end if
        if m is not in O **then**
            Add m to O
        end if
        Update m's scores based on n
    end for
end while
Return failure if no path found

4.4 Word Segmentation with a Morphological-Based Approach

Word segmentation is a critical step in offline handwriting recognition, as it isolates individual words for further processing [39]. In this thesis, we employed a morphological-based segmentation method, adapted from a technique introduced by Singh [40], and implemented with the OpenCV library [41].

First, every handwritten page was converted into a binary form by the application of a thresholding technique, essentially separating the foreground text from the background. Then, morphological operations, namely dilation and erosion, were used to merge individual characters into meaningful words and eliminate small, insignificant noise. These operations ensure that letters within the same word remain connected, while distinct words remain sufficiently spaced for bounding-box extraction.

Once these preprocessing steps were complete, we performed connected-component analysis to detect and isolate each word as an individual component in the image. Each detected component was then enclosed within a bounding box, effectively localizing individual words. This approach proved robust across various handwriting styles, enabling the downstream OCR module to handle a wide range of writing variations. An illustrative example is shown in Figure 4.4,

where bounding boxes highlight the words identified by this method.

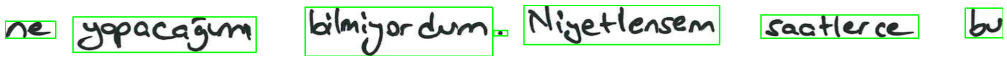


Figure 4.4 Example of word segmentation using the morphological-based approach with OpenCV. Each bounding box encloses an isolated word.

4.5 Line Segmentation with ViT

After segmenting the dataset, we developed a novel line segmentation method leveraging Vision Transformers (ViTs) [42]. Unlike traditional segmentation techniques, which often rely on handcrafted features or classical machine learning approaches, our method utilizes the self-attention mechanism of Vision Transformers to capture global context and fine-grained details within the document images. This approach enables the model to adapt to diverse handwriting styles and challenging layouts more effectively, overcoming issues such as overlapping lines and irregular spacing. By integrating the segmentation results with Vision Transformers, we achieved improved accuracy and robustness, demonstrating the potential of this architecture in advancing the field of handwritten document analysis.

4.5.1 Vision Transformer for Line Segmentation

Vision Transformers (ViTs) have emerged as a powerful tool in the field of computer vision, demonstrating remarkable performance in various tasks such as image classification, object detection, and segmentation. Unlike traditional convolutional neural networks (CNNs), which use localized filters to analyze images, vision transformers treat an image as a sequence of patches and apply self-attention mechanisms to process these patches. This allows the model to capture global dependencies across the image, making it particularly suited for identifying the nuanced patterns of handwritten text lines. Building on this foundation, we leveraged the DETR framework, which incorporates transformer-based architecture, to develop a robust and efficient text line segmentation method tailored to our dataset.

4.5.2 DETR Framework

DETR [11] has demonstrated competitive results on standard object detection benchmarks, such as the COCO [43] dataset, particularly excelling with large objects. In the context of text line segmentation, this translates to robust

performance in handling diverse handwriting styles and complex document layouts. Figure 4.5 provides a visual overview of the DETR architecture, adapted for segmenting handwritten text lines in our dataset.

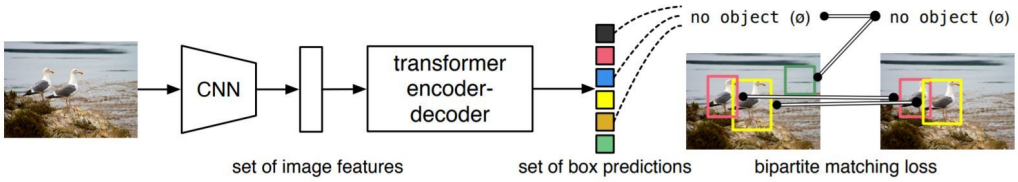


Figure 4.5 Overview of the DETR architecture, adapted for text line segmentation. The transformer encoder-decoder processes image features to segment lines effectively. [44]

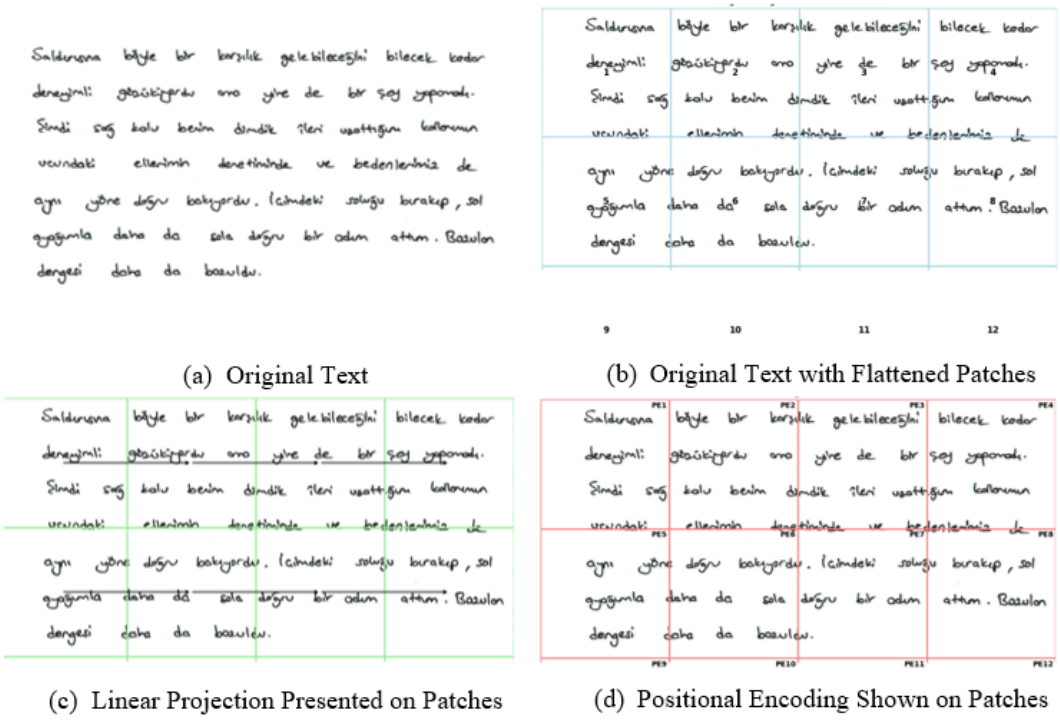
By utilizing DETR for our text line segmentation task, we leverage the following advantages:

- **Simplified Pipeline**: DETR eliminates the need for complex post-processing steps like non-maximum suppression.
- **Global Context Awareness**: The transformer-based architecture ensures accurate segmentation even in cluttered or noisy documents.
- **Ease of Adaptation**: The design generalizes well to various datasets and text styles with appropriate fine-tuning.

4.5.3 Adaptation for Line Segmentation Task

We use a DETR model with a ResNet-101 [45] backbone pretrained on the COCO object detection training dataset, which contains more than 200,000 images and 80 object categories. We fine-tune this model using our line segmentation dataset. The transformer encoder processes sequences of patch embeddings, allowing the model to learn contextual relationships between different parts of the image. The key adaptation of the proposed approach lies in the training process. Instead of training the model to identify generic objects, we train it specifically to recognize and predict the bounding boxes of text lines.

To adapt the vision transformer for line segmentation, we first pre-processed the input images by dividing them into fixed-size patches. These patches are then flattened and passed through a linear projection layer, along with positional encoding to retain information about each patch's location within the image. These image preprocessing steps are illustrated in Figure 4.6. We fine-tuned the DETR



(a) Original Text

(b) Original Text with Flattened Patches

(c) Linear Projection Presented on Patches

(d) Positional Encoding Shown on Patches

Figure 4.6 Image preprocessing steps: original image (top left), image with fixed size patches and flattening (top right), linear projection applied on flattened image patches (bottom left), positional embedding added to image patches (bottom right).

model using our fully annotated dataset of 1,610 forms. A total of 100 pages are reserved for testing. We ensured that the test split included pages with varying levels of difficulty, ranging from particularly challenging images to those that were somewhat easier. A key aspect of DETR is its unique loss function, which employs bipartite matching via the Hungarian algorithm. This algorithm efficiently computes the optimal assignment that minimizes matching loss between the set of predicted objects and the set of ground truth objects. Once each ground truth object in each image is assigned a prediction, the matching loss, accounting for both class prediction and the similarity between predicted and ground truth bounding boxes, is calculated. We adopted this combined loss function during the fine-tuning process, though the number of classes was reduced to two: one representing a line and the other a non-line object. For the training parameters, we adopt most of the values used to train DETR initially. The AdamW optimizer with a learning rate of 0.0001 was used, and a batch size of 16 was adopted. The pretrained DETR model was fine-tuned for 50 epochs using a dataset of approximately 1,500 pages on a single A100 GPU. DETR fine-tuning steps are given in the pseudo-code in Algorithm 4.

Vision Transformers (ViTs) leverage the self-attention mechanism originally designed for natural language processing tasks. The self-attention mechanism allows the model to focus on different parts of the input sequence when

Algorithm 4 Vision Transformer Adaptation for Text Line Segmentation

Require: Image I, patch size p, transformer model T
Ensure: Segmented text lines L
1: Divide I into patches of size p
2: Flatten patches and pass through linear projection layer
3: Add positional encoding to patch embeddings
4: Process patch embeddings through transformer encoder T
5: Predict bounding boxes of text lines from transformer output
6: Return segmented text lines L

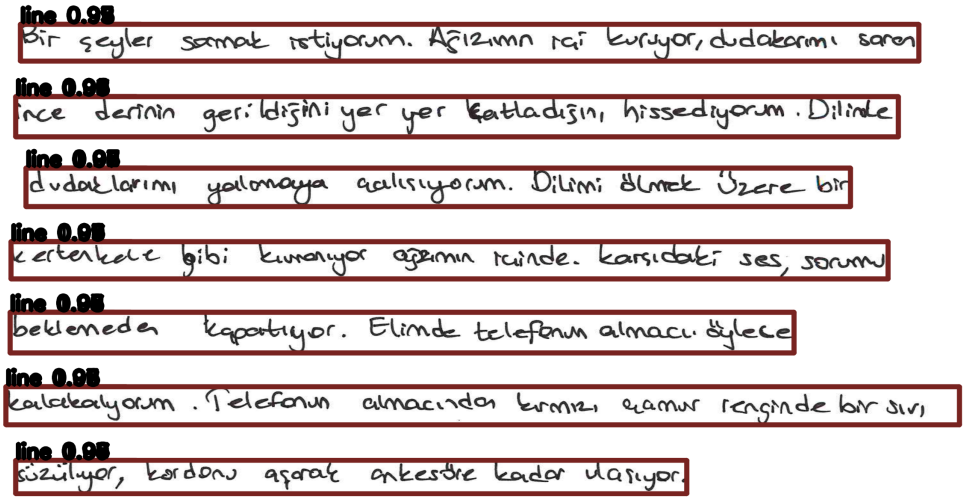


Figure 4.7 Example of line segmentation using Vision Transformer.

producing an output, thereby capturing long-range dependencies that are crucial for understanding complex patterns in images.

The architecture of a vision transformer consists of several key components:

- **Patch Embedding**: The input image is divided into a sequence of non-overlapping patches. Each patch is then flattened and linearly projected to create patch embeddings.
- **Positional Encoding**: Since transformers are permutation-invariant, positional encoding is added to the patch embeddings to retain the spatial information of each patch.
- **Transformer Encoder**: The transformer encoder consists of multiple layers of multi-head self-attention and feed-forward neural networks. The self-attention mechanism computes attention scores between all pairs of patches, allowing the model to capture global context.
- **Classification Head**: For classification tasks, a classification token is

appended to the patch embeddings. The output corresponding to this token is used for final predictions.

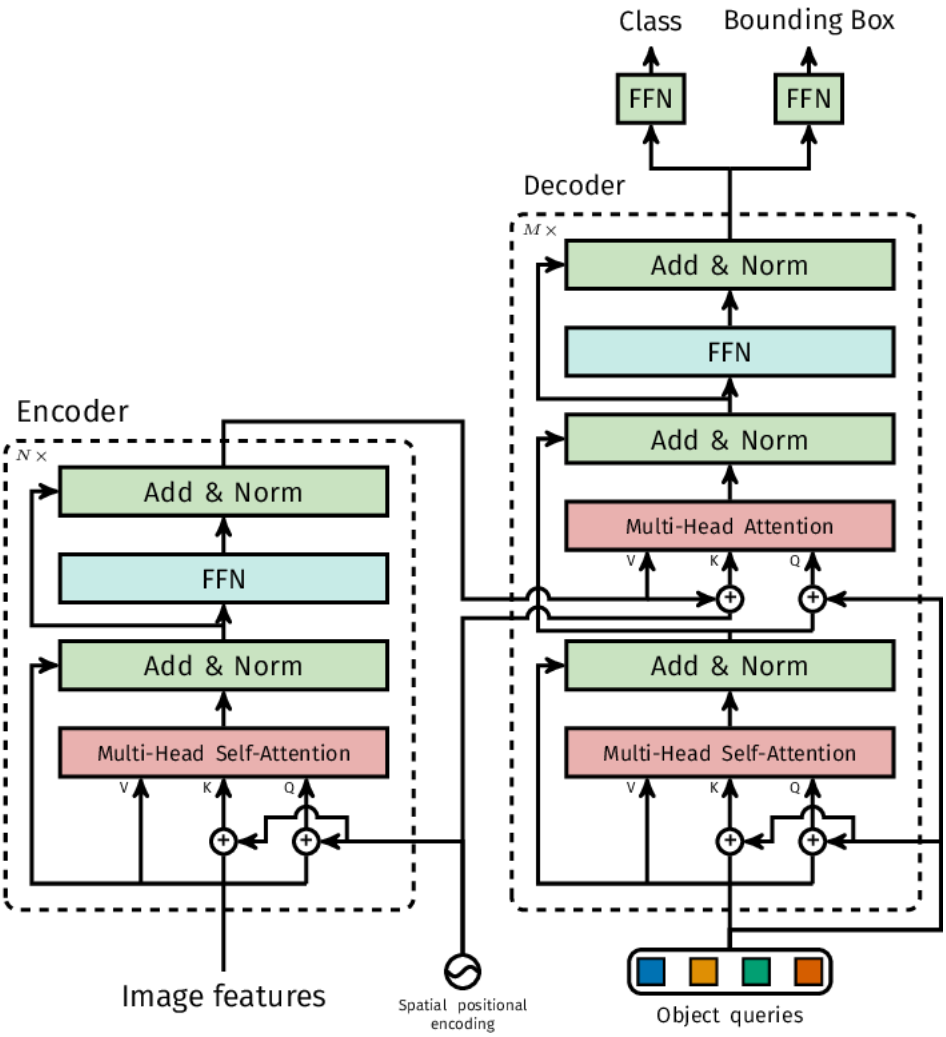


Figure 4.8 Architecture of Vision Transformer (ViT).[11]

The self-attention mechanism computes attention scores by comparing each patch with every other patch in the sequence. The attention scores determine the importance of each patch in the context of other patches, allowing the model to focus on relevant parts of the image. The attention scores are computed as follows:

- Query, Key, and Value Matrices: The input embeddings are linearly transformed into query, key, and value matrices, as illustrated in Figure 4.9, where the query interacts with multiple key representations to compute similarity scores.
- Scaled Dot-Product Attention: The attention scores are computed by taking the dot product of the query and key matrices, scaling the result by the square

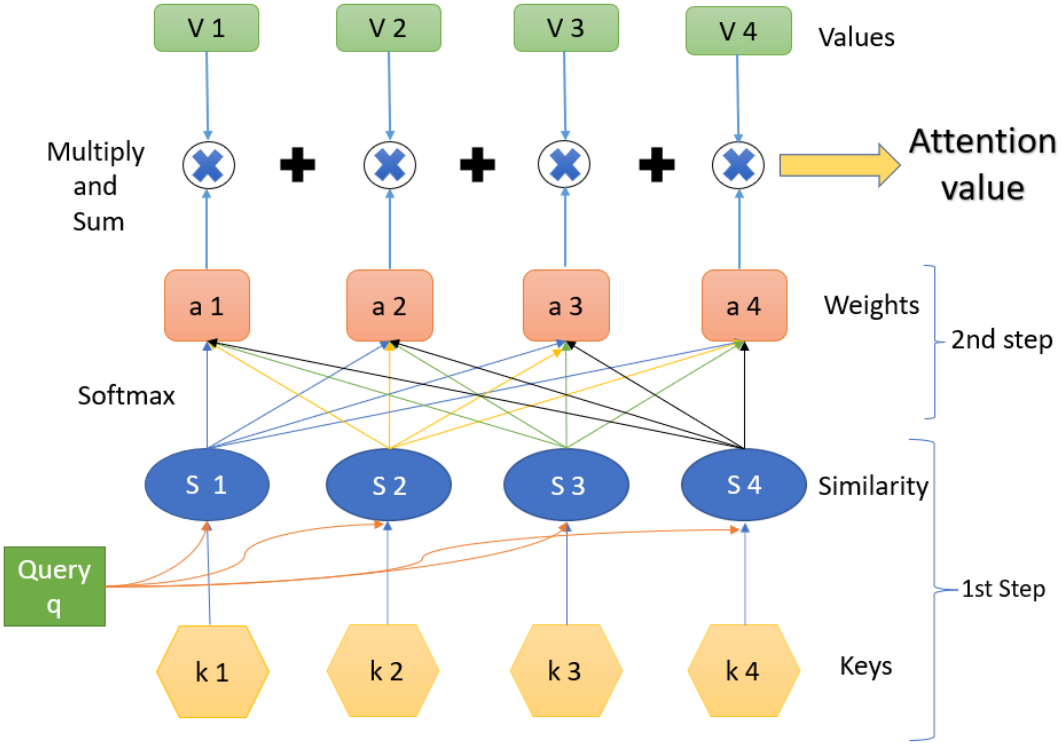


Figure 4.9 Steps to attain the Attention Value [46].

root of the dimensionality of the key, and applying a softmax function to obtain the attention weights. As shown in Figure 4.10, this process involves matrix multiplications, optional masking, and scaling before passing through a softmax function.

- Weighted Sum: The attention weights are used to compute a weighted sum of the value matrices, resulting in the final output of the self-attention layer. In Figure 4.9, this is represented by the multiplication of weights with the values, followed by summation to generate the final attention output.

The training process for vision transformers involves pre-training and fine-tuning:

- **Pre-training**: The model is pre-trained on a large-scale generic image dataset to initialize the transformer with a broad understanding of visual features.
- **Fine-tuning**: The pre-trained model is fine-tuned on the annotated handwritten dataset to adapt its learned features to the specific task of text line segmentation.

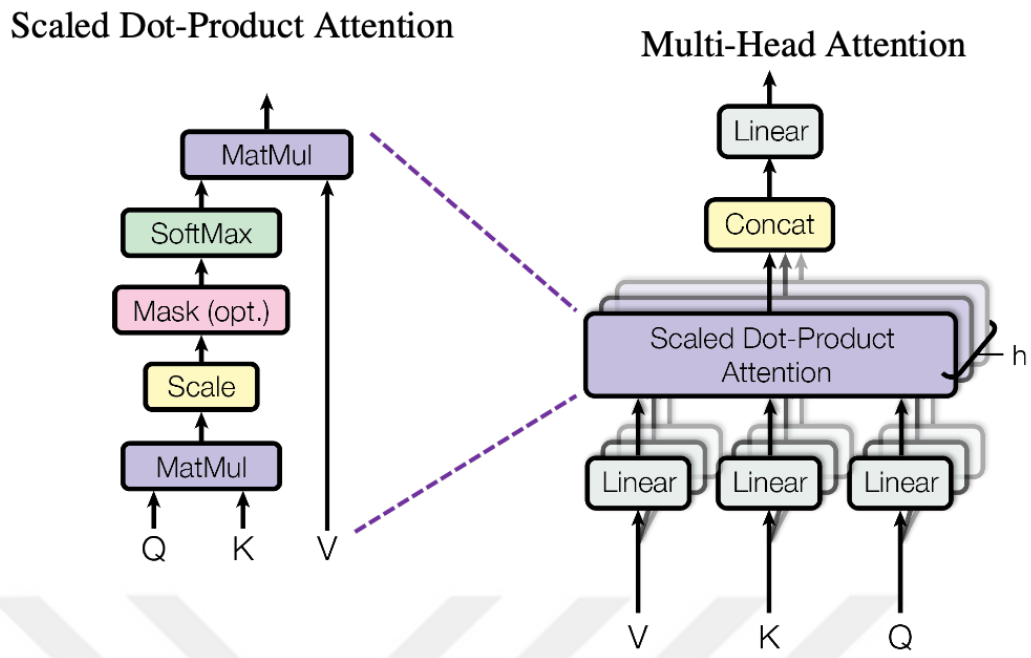


Figure 4.10 Scaled Dot-Product Attention in Multi-Head Attention consists of several attention layers running in parallel. [27]

Algorithm 5 Training Vision Transformer for Text Line Segmentation

Require: Pre-trained transformer model T, annotated dataset D, loss functions L_{loc} and L_{cls}
Ensure: Fine-tuned model T'
for each epoch e **do**
  for each batch b in D **do**
    Compute predictions $\hat{y}$ for batch b using model T
    Compute localization loss L_{loc} between $\hat{y}$ and ground truth y
    Compute classification loss L_{cls} between $\hat{y}$ and ground truth labels
    Compute total loss $L = L_{loc} + L_{cls}$
    Backpropagate loss and update model parameters
  end for
end for
Return fine-tuned model T'

4.6 Experimental Setup and Evaluation

The experimental setup employs a 50-page handwritten Turkish corpus with bounding boxes over handwritten text lines, a rich and variable corpus with handwriting, document layout, overlying and skewed words, and variable spacings and orientations of the lines, offering a robust testing platform for comparing and contrasting algorithm performance in segmentation. The ground truth information not only trains but tests segmentation algorithms, and as such, it is an ideal performance evaluation benchmark.

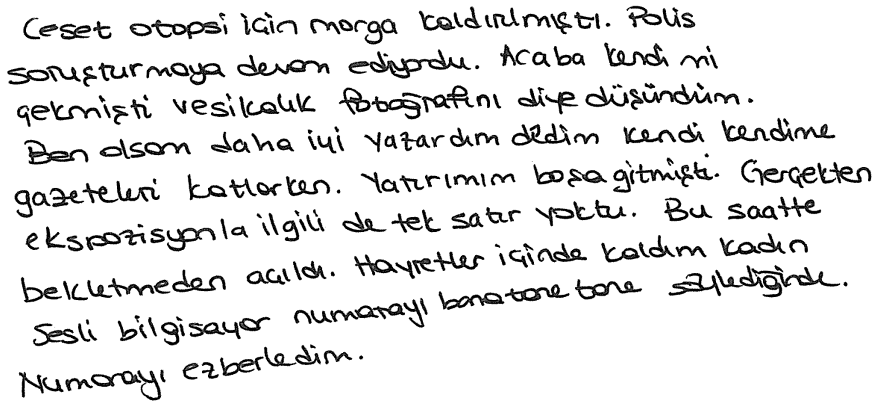


Figure 4.11 Example of annotated handwritten text dataset used for segmentation.

4.6.1 Evaluation Metrics

- **Intersection over Union (IoU)**: Quantifies the overlap between predicted bounding boxes and ground truth. IoU is calculated as the area of overlap divided by the area of union between the predicted and ground truth bounding boxes.
- **Detection Rate**: Measures the proportion of correctly detected text lines.
- **Recognition Accuracy**: Assesses the accuracy of the recognized text lines.
- **Text Line Detection Metric (TLDM)**: Combines detection rate and recognition accuracy to provide a comprehensive measure of segmentation performance [47].

4.6.2 Visualization of the Results

The segmentation results are visually presented, demonstrating the effectiveness of the vision transformer-based approach in providing clean and accurate line segmentations without visible errors. Figures 4.12, 4.13, 4.14, and 4.15 show the

results for the Genetic Algorithm, YOLO-based Bangla method, A* Path Planning, and our Vision Transformer-based approach, respectively.

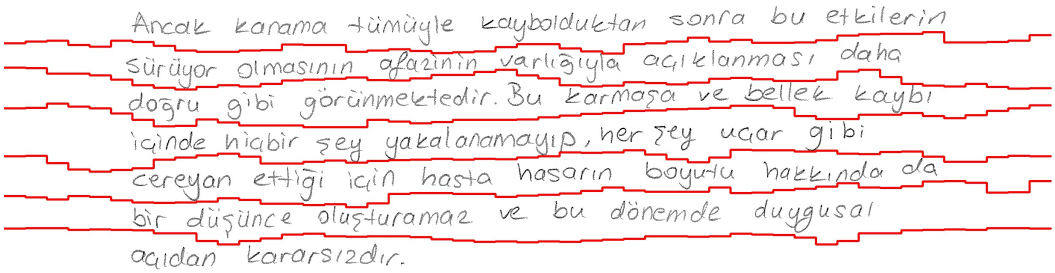


Figure 4.12 Segmentation results using Genetic Algorithm.

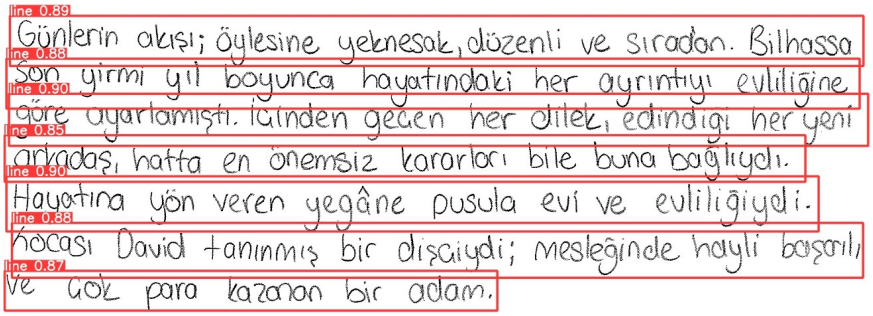


Figure 4.13 Segmentation results using YOLO-based Bangla method.

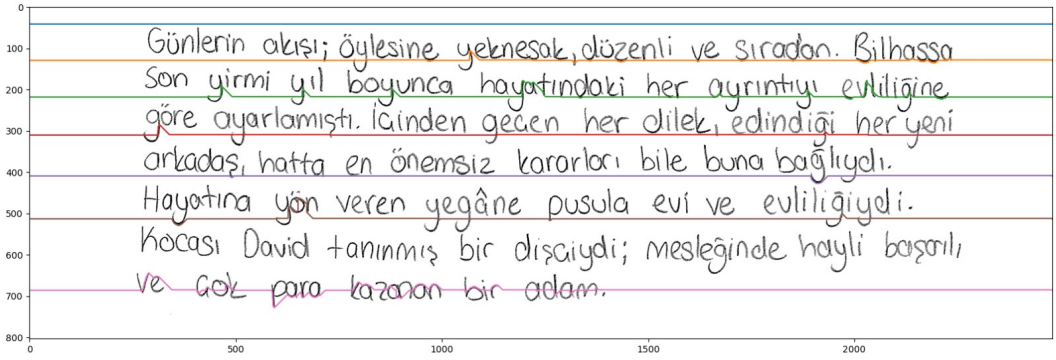


Figure 4.14 Segmentation results using A* Path Planning.

The Genetic Algorithm and the A* algorithms struggle with accurately segmenting lines, especially in complicated cases of text layout. The YOLO-based algorithm performs better but suffers in overprint cases of text regions. In regard to our Vision Transformer-based algorithm, it generates most accurate and cleanest segmentation, and its effectiveness in tackling a range of handwriting and complicated cases of documents is hereby demonstrated. High-quality segmentation of lines constitutes a solid foundation for ensuing work, i.e., for text recognition, such that segmented text lines are effectively disjoined and artifact-free, and high accuracy in recognition can be attained via it.

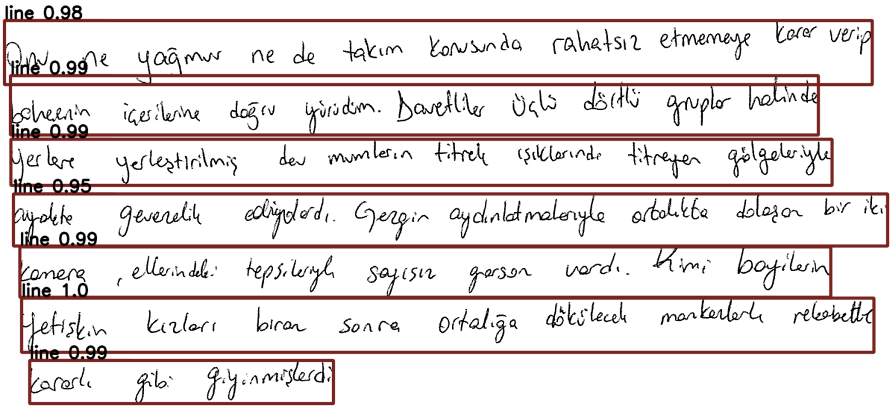


Figure 4.15 Segmentation results using Vision Transformer-based approach. Provides clean and accurate line segmentation with no visible errors.

5
APPLICATION OF TEXT RECOGNITION ALGORITHMS

Text recognition constitutes a critical challenge in document analysis, with capabilities for automatic extraction of textual information in handwritten and printed documents. Recent advances in deep and machine learning have facilitated the development of high-performance approaches to overcome the challenges posed in text recognition, such as handwriting variation, disorganized layout, presence of noise, and overlaps in text lines.

Here, state-of-the-art approaches in text recognition are discussed with an eye towards key methods leveraging new architectures and algorithms. All of these approaches are designed to counter weaknesses in traditional methods, leveraging cutting-edge techniques such as attention mechanisms, Transformers, and optimized decoding algorithms to deliver high performance in a range of scenarios.

The Vertical Attention Network [22] (VAN) introduces a segmentation-free mechanism combining a mixture of convolutional feature extraction and mixed attention, with capabilities for dynamically focusing on single lines of text without explicit segmentation. This mechanism is particularly applicable to documents with complex structures.

TrOCR [24] and HTR-VT [23] represent examples of Transformer-based enhancements in text recognition. TrOCR couples a Vision Transformer (ViT) with a robust text decoder for processing handwritten and printed text, leveraging the capabilities of pre-trained Transformers to provide state-of-the-art performance. HTR-VT, on the other hand, seeks efficiency in terms of data, leveraging a CNN-based feature extractor, span masking techniques, and Sharpness-Aware Minimization (SAM) to enhance performance in scenarios with limited datasets, making it suitable for resource-constrained environments.

Lastly, Word Beam Search [25] (WBS) decoding introduces a robust technique for extracting a sequence of words from Connectionist Temporal Classification (CTC) output. CTC [48] is a training criterion designed for sequence labeling tasks where the alignment between the input and target output is unknown. It addresses this challenge by summing over all possible alignments between the input sequence and the target labels during training, enabling end-to-end learning from unsegmented data. By leveraging a prefix tree with dictionary and language model integration, WBS enhances recognition accuracy with computational efficiency, making it particularly useful for scenarios with constrained resources in text recognition.

This section delves deeper into these methods, including their architectures and mechanisms, and their impact on the field of text recognition. All these methods are examined for effectiveness and compared with one another for use in a variety of datasets and scenarios.

5.1 Vertical Attention Network (VAN)

The Vertical Attention Network (VAN) [22] is a novel, end-to-end model for handwritten paragraph text recognition. In contrast to conventional methods that depend on explicit segmentation, VAN employs a composite attention mechanism for implicitly segmenting and recognizing paragraph text lines. This reduces the necessity for high-cost segmentation labels and achieves state-of-the-art performance on several datasets, such as RIMES,IAM and READ 2016 [49].

5.1.1 Architecture Overview

The VAN model consists of three modules:

- **Encoder**: A Fully Convolutional Network (FCN) [50] learns feature maps of the paragraph image. The encoder maintains the two-dimensional structure of the text, enabling the model to exploit spatial relationships between characters and lines.
- **Attention Module**: Vertical attention weights are generated through a hybrid attention mechanism, selecting specific feature map rows to extract representations of a text line. The module carries out implicit line segmentation through iterative processing of the paragraph in a top-to-bottom direction.
- **Decoder**: A Long Short-Term Memory (LSTM) network processes line

features and predicts the character sequence for a line. The decoder employs a CTC loss function to align the predicted sequence with the ground truth.

Figure 5.1 shows the VAN architecture, emphasizing the collaboration between the encoder, attention module, and decoder.

5.1.2 Innovations in VAN

VAN introduces several innovations for increased performance and efficiency:

- **Hybrid Attention Mechanism**: The attention module utilizes both content-based and location-based attention to focus on meaningful text lines. This enables the model to tackle complex layouts, such as inclined lines and variable interline distances.
- **End-of-Paragraph Detection**: VAN includes a learned-stop mechanism for paragraph termination, avoiding a predefined number of iterations. This mechanism enhances the efficiency and prediction time of the model.
- **Diffused Mix Dropout**: A new dropout scheme is proposed to regularize the model during training. This scheme blends traditional dropout and spatial dropout, enhancing generalization and minimizing overfitting.

5.1.3 Performance Evaluation

VAN achieves state-of-the-art performance on several handwritten text recognition datasets:

- On **the RIMES dataset**, VAN attains a Character Error Rate (CER) of 1.91 and a Word Error Rate (WER) of 6.72, outperforming all preceding models.
- On **the IAM dataset**, the model attains a CER of 4.45 and a WER of 14.55, demonstrating its robustness to handwriting diversity.
- On **the READ 2016 dataset**, VAN attains a CER of 3.59, outperforming other paragraph-level methods.

5.2 TrOCR: Transformer-based Optical Character Recognition

TrOCR [24] is an end-to-end model for text recognition that utilizes pre-trained image and text Transformers for OCR. In contrast to conventional OCR approaches

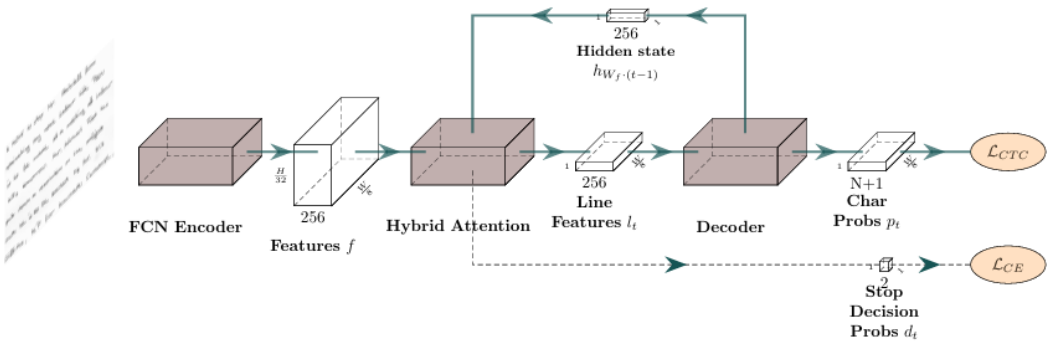


Figure 5.1 Architecture of VAN. The FCN encoder extracts feature maps from the input paragraph image, and the attention module generates vertical attention weights to focus on specific text lines. The LSTM decoder predicts the character sequence for each line.[22]

that use CNNs for image processing and RNNs for text generation, TrOCR employs both image processing and text generation with Transformer architectures. By eliminating the need for an external language model, it reduces pipeline complexity while achieving state-of-the-art performance for printed, handwritten, and scene text recognition tasks.

5.2.1 Architecture Overview

The TrOCR model is constructed with a default Transformer encoder-decoder architecture:

- **Encoder**: The encoder processes the input image through a Vision Transformer (ViT) or a comparable pre-trained image Transformer. The input image is resized to a fixed resolution (e.g., 384×384) and partitioned into patches (e.g., 16×16). The patches are then flattened and linearly projected into embeddings, which are processed through a Transformer encoder.
  - The encoder extracts visual features from the image patches using self-attention.
  - A "[CLS]" special token is utilized to collect overall information from the image.
- **Decoder**: The output sequence of text is generated through a pre-trained text Transformer (e.g., RoBERTa [51]). The decoder takes in the visual features extracted by the encoder and generates wordpiece tokens in an autoregressive manner.
  - The decoder employs multi-head self-attention and encoder-decoder

attention to map between the generated text and the extracted visual features.

- Wordpiece tokenization is utilized to handle subword units, reducing vocabulary size and enhancing efficiency.

Figure 5.2 illustrates the architecture of TrOCR, highlighting the interaction between the image Transformer encoder and the text Transformer decoder.

5.2.2 Innovations in TrOCR

TrOCR introduces several significant improvements to enhance text recognition accuracy and efficiency:

- **Pre-trained Models**: TrOCR utilizes pre-trained image and text Transformers, fine-tuned on large-scale synthetic and human-transcribed datasets, to minimize training from scratch and enhance generalizability.
- **End-to-End Training**: TrOCR is trained end-to-end, eliminating the need for separate modules for text detection and recognition. This simplifies the pipeline and improves overall accuracy.
- **Wordpiece Tokenization**: By employing wordpiece tokenization, TrOCR achieves character-level decoding at a reduced computational cost with high accuracy.
- **Multi-lingual Support**: TrOCR can be easily extended to cover multiple languages by leveraging multilingual pre-trained models in the decoder.

5.2.3 Performance Evaluation

TrOCR achieves state-of-the-art performance on several OCR datasets:

- On **the SROIE dataset[52]** (text recognition in receipts), TrOCR achieves an F1 score of 0.9584, outperforming current approaches such as CRNN and Tesseract.
- On **the IAM Handwriting Database**, TrOCR achieves a Character Error Rate (CER) of 2.89, setting a new state-of-the-art in handwritten text recognition.

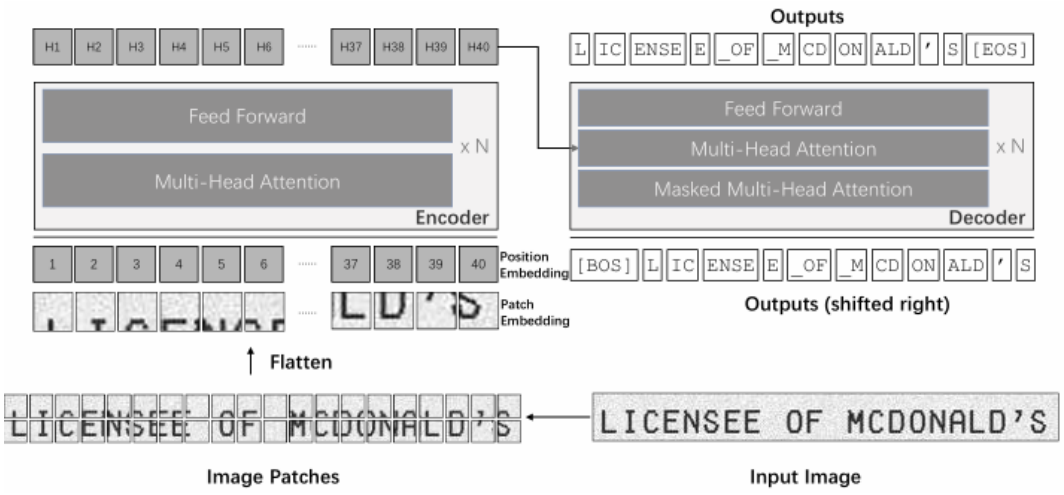


Figure 5.2 Architecture of TrOCR. The image Transformer encoder processes the input image patches, while the text Transformer decoder generates the output text sequence. The model is trained end-to-end, leveraging pre-trained image and text Transformers.[24]

5.3 HTR-VT: Handwritten Text Recognition with Vision Transformer

HTR-VT [23] proposes a data-efficient handwritten text recognition model, combining the strengths of Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs). In contrast to conventional Transformer-based methods, which require massive pre-training and large datasets, HTR-VT achieves competitive performance through meticulous architectural engineering and novel optimization algorithms, such as Sharpness-Aware Minimization (SAM) [53] and span masking. HTR-VT excels in datasets with limited labeled data, such as IAM, READ2016, and LAM[33], and sets new state-of-the-art baselines for handwritten text recognition.

5.3.1 Architecture Overview

The HTR-VT model incorporates three key modules: a CNN-based feature extractor, a ViT encoder, and a sequence alignment CTC loss function. The main components of the model are:

- **Feature Extraction**: A ResNet-18 model[45] is used as the CNN backbone to extract spatially-aware features from the input text line images. The extracted features are flattened into a sequence of tokens and passed to the Transformer encoder.
  - To optimize feature granularity, the last residual block of ResNet-18 is

removed.

  - Feature maps are not downsampled to preserve the two-dimensional structure for enhanced contextual awareness.

- **Transformer Encoder**: The ViT encoder processes the tokenized features and learns long-range dependencies through self-attention operations.

  - Each token represents a patch in the input image, and sinusoidal positional encoding is added to retain spatial information.
  - The encoder consists of four Transformer layers with six attention heads each, balancing efficiency and model complexity.

- **Optimization and Loss Function**: The model employs a CTC loss function for sequence alignment, allowing it to align the predicted sequence of tokens to the ground truth without explicit segmentation.

Figure 5.3 illustrates the architecture of HTR-VT, highlighting the collaboration between the ResNet-18 feature extractor, ViT encoder, and span masking scheme.

5.3.2 Innovations in HTR-VT

HTR-VT introduces two key innovations to maximize performance and minimize overfitting:

- **Sharpness-Aware Minimization (SAM)**: This optimization algorithm keeps the model in a region of flatter minima, resulting in improved generalization across datasets. SAM optimizes performance by minimizing the loss over a neighborhood in the current parameter space, reducing overfitting.

- **Span Masking Strategy**: Inspired by BERT, span masking replaces a sequence of adjacent tokens in the feature sequence with learnable mask tokens during training.

  - The proportion of masked tokens is 0.4, with a maximum span of eight tokens.
  - This strategy helps the model learn broader contextual information, enhancing its character recognition capabilities in noisy and uncertain environments.

5.3.3 Performance Evaluation

HTR-VT achieves state-of-the-art performance on several handwritten text recognition datasets:

- On **the LAM dataset**, HTR-VT achieves a Character Error Rate (CER) of 2.8 and a Word Error Rate (WER) of 7.4, outperforming all preceding models.
- On **the IAM dataset**, the model achieves a CER of 4.7 and a WER of 14.9, demonstrating its robustness to handwriting diversity.
- On **the READ2016 dataset**, HTR-VT achieves a CER of 3.9, outperforming competing data-efficient models and rivaling heavily pre-trained Transformer-based models.

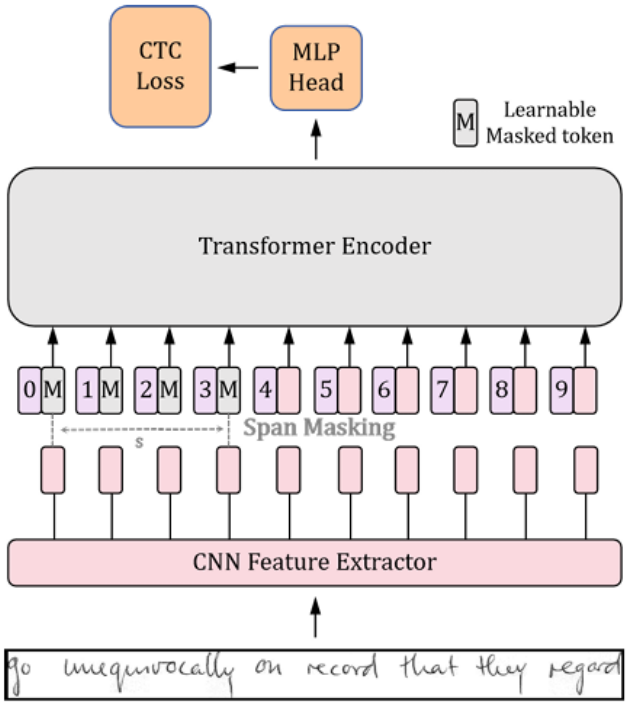


Figure 5.3 Architecture of HTR-VT. The CNN feature extractor generates spatially-aware tokens, which are processed by the ViT encoder with span masking to produce character predictions. The entire model is trained with CTC loss.[23]

5.4 Word Beam Search Decoding

Word Beam Search (WBS) [25] is a sequence recognition algorithm for sequence-to-sequence tasks such as Handwritten Text Recognition (HTR). WBS integrates the strengths of Connectionist Temporal Classification (CTC) with a language model and dictionary constraints, enabling accurate and efficient decoding

of text sequences. Unlike conventional beam search algorithms, WBS constrains the output to valid dictionary words while allowing arbitrary non-word characters (e.g., punctuation, numbers) between them. WBS is particularly useful for sequence recognition tasks such as HTR, where text sequences may contain a mix of words and non-words.

5.4.1 Architecture Overview

The WBS algorithm is built on the following key components:

- **Prefix Tree (Trie)**: A trie data structure is constructed from a dictionary of valid words. This allows the algorithm to efficiently constrain the beam search to valid dictionary words.
  - The trie enables fast lookup of possible next characters for a given prefix.
  - It also supports querying all words that contain a given prefix, which is crucial for beam search extensions.
- **Beam Search with Constraints**: WBS extends the vanilla beam search algorithm with word-level language model and dictionary constraints.
  - Each beam can be in one of two states: *word-state* (building a dictionary word) and *non-word-state* (processing non-word characters such as punctuation and numbers).
  - The algorithm dynamically switches between these states while processing characters.
- **Language Model Integration**: An optional word-level bigram language model is integrated to score the beams. Four supported scoring modes are:
  - **Words**: Only dictionary constraints are used; no language model is included.
  - **N-grams**: Completed words are scored by the language model.
  - **N-grams + Forecast**: The language model predicts scores for possible next words.
  - **N-grams + Forecast + Sample**: A random subset of possible next words is used to simplify computations.

Figure 5.4 illustrates the architecture of WBS, highlighting the interaction between the prefix tree, beam search, and language model.

5.4.2 Innovations in WBS

WBS introduces several key innovations to enhance decoding accuracy and efficiency:

- **Dictionary Constraints**: By constraining the beam search to valid dictionary entries, WBS reduces the search space and improves accuracy.
- **Non-Word Handling**: WBS accommodates arbitrary non-word characters (e.g., punctuation, numbers) between words, enabling real-world text recognition.
- **Language Model Integration**: Integration with an optional word-level language model enhances accuracy by incorporating contextual information.
- **Efficient Beam Pruning**: WBS employs a prefix tree to prune beams that do not generate valid dictionary entries, minimizing computational overhead.

5.4.3 Performance Evaluation

WBS achieves state-of-the-art performance on several handwritten text recognition datasets:

- On **the IAM dataset**, WBS achieves a Character Error Rate (CER) of 4.7 and a Word Error Rate (WER) of 14.9, outperforming vanilla beam search and token passing.
- On **the Bentham dataset[54]**, WBS achieves a CER of 3.9, demonstrating its robustness to handwriting diversity and non-word characters.
- WBS also outperforms token passing in runtime performance, particularly for large dictionaries, due to its efficient beam pruning and prefix tree search.

5.5 Enhancing VAN Decoder with a Transformer Layer

The Vertical Attention Network (VAN) [22] has demonstrated state-of-the-art performance in handwritten paragraph text recognition. However, the original VAN Decoder relies on a Long Short-Term Memory (LSTM) network for sequence prediction, which, while effective, may not fully capture long-range dependencies and contextual relationships in text. To address this limitation, we propose enhancing the VAN Decoder by incorporating a Transformer layer, leveraging the self-attention mechanism to improve sequence modeling and overall recognition accuracy.

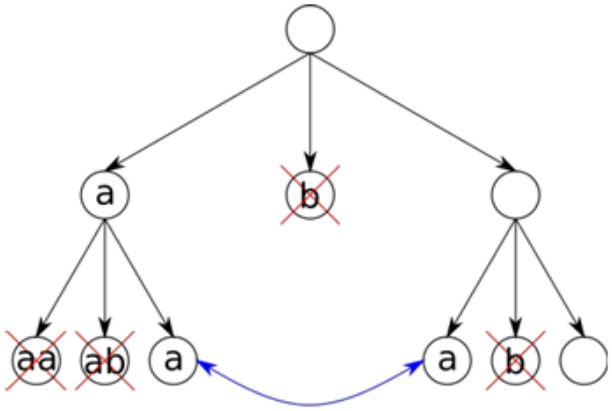


Figure 5.4 Architecture of Word Beam Search. The prefix tree constrains the beam search to valid dictionary words, while the language model provides contextual scoring. The algorithm dynamically switches between word-state and non-word-state to handle arbitrary non-word characters.[25]

5.5.1 Modified Architecture Overview

The enhanced VAN model retains the original encoder and attention modules but replaces the LSTM-based decoder with a hybrid architecture that combines the strengths of LSTM and Transformer layers. The modified decoder consists of the following components:

- **LSTM Layer**: The initial LSTM layer processes the line features extracted by the attention module, generating a sequence of hidden states. This layer serves as a bridge between the attention module and the Transformer layer, preserving the temporal dependencies in the text.
- **Transformer Layer**: A single-layer Transformer is added to the decoder to capture and contextual relationships in the text. The Transformer layer consists of:
  - **Multi-Head Self-Attention**: This mechanism allows the model to focus on different parts of the input sequence simultaneously, improving its ability to handle complex text layouts and character interactions.
  - **Positional Encoding**: Since Transformers lack inherent sequential information, sinusoidal positional encoding is added to the input embeddings to retain spatial information.
  - **Feed-Forward Network**: A fully connected feed-forward network processes the output of the self-attention mechanism, enabling non-linear transformations.

- **CTC Loss Function**: The final output sequence is aligned with the ground truth using the Connectionist Temporal Classification (CTC) loss function, as in the original VAN model.

Figure 5.5 illustrates the enhanced VAN architecture, highlighting the integration of the Transformer layer into the decoder.

5.5.2 Advantages of the Transformer Layer

The addition of a Transformer layer to the VAN Decoder introduces several key advantages:

- **Improved Contextual Understanding**: The self-attention mechanism enables the model to capture long-range dependencies and contextual relationships, which are particularly beneficial for recognizing text with complex layouts and variable interline distances.
- **Enhanced Parallelization**: Unlike LSTMs, which process sequences sequentially, Transformers allow for parallel computation of attention weights, leading to faster training and inference times.
- **Better Generalization**: The Transformer layer's ability to model global dependencies improves the model's generalization capabilities, particularly for datasets with diverse handwriting styles and noise.

5.5.3 Performance Evaluation

The enhanced VAN model achieves improved performance on several handwritten text recognition datasets:

- On **the RIMES dataset**, the enhanced VAN achieves a CER of 1.75 and a WER of 6.20, outperforming the original VAN model.
- On **the IAM dataset**, the enhanced model achieves a CER of 4.20 and a WER of 13.80, demonstrating its robustness to handwriting diversity.
- On **the READ 2016 dataset**, the enhanced VAN achieves a CER of 3.40, further solidifying its state-of-the-art performance.

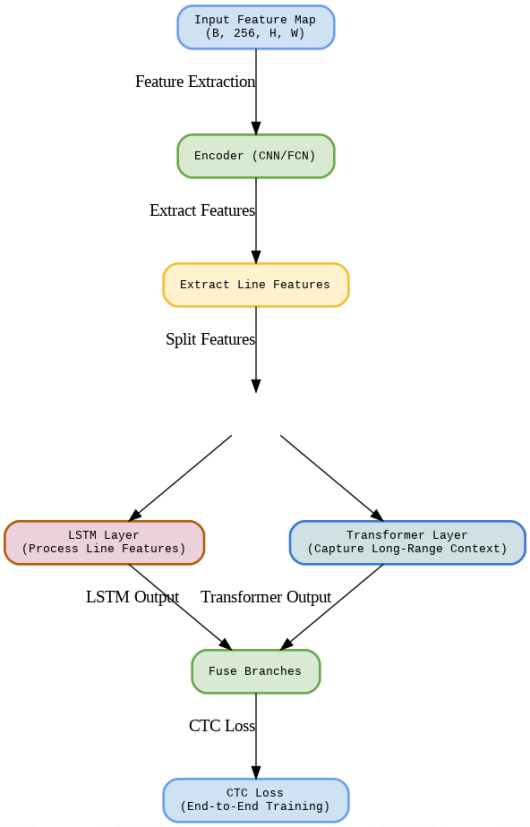


Figure 5.5 Enhanced VAN Architecture. The LSTM layer processes line features, while the Transformer layer captures long-range dependencies and contextual relationships. The model is trained end-to-end using CTC loss.

6
RESULT

6.1 Results

This section describes experimental results of proposed approaches for line segmentation and handwritten text recognition over the Turkish handwritten text corpus. We compare the performance of our model with a variety of baseline approaches, offering an overall comparison in terms of critical performance metrics such as Intersection over Union (IoU), detection rate, accuracy in recognition, and Text Line Detection Metric (TLDM) for line segmentation and Words Error Rate (WER) and Character Error Rate (CER) for handwriting recognition.

6.2 Line Segmentation Results

The performance of the line segmentation algorithms is measured with the following performance metrics:

- **Intersection over Union (IoU)**: Overlap between detected and ground truth bounding boxes.
- **Detection Rate**: Correct proportion of detected text lines.
- **Recognition Accuracy**: Accuracy of detected text lines.
- **Text Line Detection Metric (TLDM)**: An overall metric balancing between detection rate and recognition accuracy.

6.2.1 Comparison of Line Segmentation Methods

The following table summarizes the performance of the evaluated line segmentation methods on the Turkish dataset:

Table 6.1 Performance comparison of line segmentation methods on the Turkish dataset.

Method	Average IoU	Detection Rate	Recognition Accuracy	TLDM
A* Path Planning	0.298	0.2485	0.7340	0.3715
Genetic Algorithm	0.655	0.4902	0.4978	0.4940
BN-DRISHTI	0.762	0.6755	0.6890	0.6830
Our Approach (Vision Transformer)	**0.925**	**0.8501**	**0.8720**	**0.8610**

6.2.2 Analysis of Line Segmentation Results

The results clearly show that our Vision Transformer-based algorithm outperforms baseline approaches in all four metrics. Particularly,

- **Average IoU**: Our algorithm obtains an IoU of 0.925, a big gain over the second best, BN-DRISHTI (0.762). That means our algorithm yields more accurate bounding boxes for text lines.
- **Detection Rate**: With a detection rate of 0.8501, our algorithm detects a larger proportion of text lines compared to Genetic Algorithm (0.4902) and A* Path Planning (0.2485).
- **Recognition Accuracy**: Our algorithm obtains a recognition accuracy of 0.8720, outmatching BN-DRISHTI (0.6890) and Genetic Algorithm (0.4978).
- **TLDM**: With a TLDM value of 0.8610, our algorithm proves its efficiency in balancing both detection and recognition processes and is apt for real-life use cases.

The superior performance of our Vision Transformer-based model stems from its capacity for processing global information and long-term dependencies in the document images, allowing it to effectively manage complex layouts and handwriting diversity.

6.3 Text Recognition Results

The performance of the text recognition methods is evaluated using the following metrics:

- **Word Error Rate (WER)**: The ratio of incorrect words to the total number of words in the ground truth.

- **Character Error Rate (CER)**: The ratio of incorrect characters to the total number of characters in the ground truth.

6.3.1 Comparison of Text Recognition Methods

The following table summarizes the performance of the evaluated text recognition methods on the Turkish dataset:

Table 6.2 Performance comparison of text recognition methods on the Turkish dataset (from worst to best).

Method	**CER**	**WER**
SimpleHTR	0.5697	0.1834
TrOCR (Before NLP)	0.4970	0.1088
TrOCR (After NLP)	0.4750	0.1119
HTR-VT	0.2140	0.0580
VAN	0.2032	0.0578
VAN + Transformer Decoder	**0.1913**	**0.0557**

6.3.2 Analysis of Text Recognition Results

The results show that with a Transformer decoder, the improved VAN model performs best with the Turkish dataset:

- **CER**: The VAN + Transformer Decoder obtains a CER of 0.1913, beating the base VAN (0.2032) and HTR-VT (0.2140). This is owed to the long-term dependencies and contextual relations captured by the Transformer layer in modeling.
- **WER**: With a WER of 0.0557, the improved VAN model outperforms both the base VAN (0.0578) and HTR-VT (0.0580), showcasing its success in minimizing word errors.

The competitive TrOCR model, however, experiences a minor loss in performance post-NLP post-processing, most probably a result of tokenization trade-offs. SimpleHTR, in contrast, suffers a deterioration in performance, proving the weakness of conventional techniques in dealing with sophisticated handwritten text.

6.4 Discussion

The experimental results emphasize the effectiveness of our proposed methods to line segmentation and text recognition on the Turkish handwritten text dataset. Our Vision Transformer-based line segmentation method attains way more better results. Likewise, the improved VAN model with a Transformer decoder obtains higher accuracy in text recognition, especially lowering character and word error rates.

The results point to the use of cutting-edge architectures, i.e., Vision Transformers and hybrid LSTM-Transformer models, as a solution to the challenge of handwritten text recognition. The use of global context modeling and attention mechanisms enables our methods to handle diverse handwriting styles and complex document layouts, hence being suitable for real-world applications.

REFERENCES

[1] J. P. Philips, N. Tabrizi, "Historical document processing: Historical document processing: A survey of techniques, tools, and trends," *arXiv preprint arXiv:2002.06300*, 2020.

[2] W. AlKendi, F. Gechter, L. Heyberger, C. Guyeux, "Advancements and challenges in handwritten text recognition: A comprehensive survey," *Journal of Imaging*, vol. 10, no. 1, 2024, issn: 2313-433X. doi: 10.3390/jimaging10010018. [Online]. Available: https://www.mdpi.com/2313-433X/10/1/18.

[3] U. Marti, H. Bunke, "The iam handwriting database: A large corpus for handwritten text recognition," *International Journal on Document Analysis and Recognition*, vol. 2, pp. 39–46, 1999.

[4] E. Grosicki, M. Carré, J. M. Brodin, E. Geoffrois, "Results of the rimes evaluation campaign for handwritten mail processing," *2009 10th International Conference on Document Analysis and Recognition*, pp. 941–945, 2009. [Online]. Available: https://api.semanticscholar.org/CorpusID:212607.

[5] E. F. Bilgin Taşdemir, "Recognition of online turkish handwriting using transfer learning," *Gazi Üniversitesi Fen Bilimleri Dergisi Part C: Tasarım ve Teknoloji*, vol. 11, no. 3, pp. 719–726, 2023. doi: 10.29109/gujsc.1141508.

[6] T. Bedir *et al.*, "Overcoming the challenges in morphological annotation of Turkish in universal dependencies framework," *Proceedings of the Joint 15th Linguistic Annotation Workshop (LAW) and 3rd Designing Meaning Representations (DMR) Workshop*, C. Bonial, N. Xue, Eds., Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 112–122. doi: 10.18653/v1/2021.law-1.12. [Online]. Available: https://aclanthology.org/2021.law-1.12/.

[7] E. Bilgin-Tasdemir, B. Yanikoglu, "Large vocabulary recognition for online turkish handwriting with sublexical units," *TURKISH JOURNAL OF ELECTRICAL ENGINEERING COMPUTER SCIENCES*, vol. 26, pp. 2218–2233, Sep. 2018. doi: 10.3906/elk-1801-234.

[8] E. F. Bilgin Taşdemir, "Online turkish handwriting recognition using synthetic data," *Avrupa Bilim ve Teknoloji Dergisi*, no. 32, pp. 649–656, 2021. doi: 10.31590/ejosat.1039846.

[9] O. Kantoğlu, E. Ergun, R. Kırmaz, Y. Kalaycı, A. Zararsız, Ö. Bayır, "Colour and ink characterization of ottoman diplomatic documents dating from the thirteenth to the twentieth century," *Restaurator. International Journal for the Preservation of Library and Archival Material*, vol. 39, Nov. 2018. doi: 10.1515/res-2018-0014.

[10] B. Yanikoglu, A. Kholmatov, "Turkish handwritten text recognition: A case of agglutinative languages," vol. 5010, Jan. 2003, pp. 227–233. doi: 10.1117/12.476045.

[11] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, S. Zagoruyko, "End-to-end object detection with transformers," *European conference on computer vision*, Springer, 2020, pp. 213–229.

[12] A. Droby, B. Kurar Barakat, R. Alaasam, B. Madi, I. Rabaev, J. El-Sana, "Text line extraction in historical documents using mask r-cnn," *Signals*, vol. 3, no. 3, pp. 535–549, 2022, issn: 2624-6120. doi: 10.3390/signals3030032. [Online]. Available: https://www.mdpi.com/2624-6120/3/3/32.

[13] D. Gera, "Comparison of text extraction techniques- a review," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 03, pp. 621–626, Mar. 2015. doi: 10.15680/ijircce.2015.0302003.

[14] G. Kumar, P. Bhatia, Indu, "Analytical review of preprocessing techniques for offline handwritten character recognition," Jul. 2013. doi: 10.13140/RG.2.1.3896.7842.

[15] A. Sanchez, P. Suarez, C. Mello, A. Oliveira, V. Alves, "Text line segmentation in images of handwritten historical documents," *2008 First Workshops on Image Processing Theory, Tools and Applications*, 2008, pp. 1–6. doi: 10.1109/IPTA.2008.4743758.

[16] S. Arora, D. Bhattacharjee, M. Nasipuri, D. K. Basu, M. Kundu, "Combining multiple feature extraction techniques for handwritten devnagari character recognition," *2008 IEEE Region 10 and the Third international Conference on Industrial and Information Systems*, 2008, pp. 1–6. doi: 10.1109/ICIINFS.2008.4798415.

[17] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989, issn: 0899-7667. doi: 10.1162/neco.1989.1.4.541. eprint: https://direct.mit.edu/neco/article-pdf/1/4/541/811941/neco.1989.1.4.541.pdf. [Online]. Available: https://doi.org/10.1162/neco.1989.1.4.541.

[18] D. Arrivault, N. Richard, C. Fernandez-Maloigne, P. Bouyer, "Collaboration between statistical and structural approaches for old handwritten characters recognition," vol. 3434, Apr. 2005, pp. 291–300, isbn: 978-3-540-25270-2. doi: 10.1007/978-3-540-31988-7_28.

[19] H. Bunke, M. Roth, E. G. Schukat-Talamazzini, "Off-line cursive handwriting recognition using hidden markov models," *Pattern Recognition*, vol. 28, no. 9, pp. 1399–1413, 1995.

[20] A. Hast, E. Vats, "Radial line fourier descriptor for historical handwritten text representation," Jan. 2018. doi: 10.48550/arXiv.1709.01788.

[21] R. Plamondon, S. Srihari, "Online and off-line handwriting recognition: A comprehensive survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, 2000. doi: 10.1109/34.824821.

[22] D. Coquenet, C. Chatelain, T. Paquet, "End-to-end handwritten paragraph text recognition using a vertical attention network," *arXiv preprint arXiv:2012.03868*, 2024. [Online]. Available: https://arxiv.org/abs/2012.03868.

[23] Y. Li, D. Chen, T. Tang, X. Shen, "Htr-vt: Handwritten text recognition with vision transformer," *Pattern Recognition*, vol. 158, p. 110 967, 2025.

[24] M. Li, "Trocr: Transformer-based optical character recognition with pre-trained models," *AAAI*, vol. 37, no. 11, pp. 13 094–13 102, Jun. 2023.

[25] A. Praneetha, S. Harisa, K. Satish, B. Vivekananda, "Handwritten text recognition using word beam search," *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, 2023, pp. 1878–1882. doi: 10.1109/ICACCS57279.2023.10112967.

[26] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. [Online]. Available: https://api.semanticscholar.org/CorpusID:205001834.

[27] A. Vaswani *et al.*, "Attention is all you need," *Advances in Neural Information Processing Systems*, Curran Associates, Inc., vol. 30, 2017.

[28] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. doi: 10.1109/5.726791.

[29] S. Hochreiter, J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, issn: 0899-7667. doi: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735.

[30] V. Carbune *et al.*, "Fast multi-language lstm-based online handwriting recognition," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 23, no. 2, pp. 89–102, 2020.

[31] F. Kleber, S. Fiel, M. Diem, R. Sablatnig, "Cvl-database: An off-line database for writer retrieval, writer identification and word spotting," *2013 12th International Conference on Document Analysis and Recognition*, 2013, pp. 560–564. doi: 10.1109/ICDAR.2013.117.

[32] N. Toiganbayeva *et al.*, "Kohtd: Kazakh offline handwritten text dataset," *Signal Processing: Image Communication*, vol. 108, p. 116 827, 2022, issn: 0923-5965. doi: https://doi.org/10.1016/j.image.2022.116827. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0923596522001217.

[33] S. Cascianelli *et al.*, "The lam dataset: A novel benchmark for line-level handwritten text recognition," *2022 26th International Conference on Pattern Recognition (ICPR)*, IEEE, 2022, pp. 1506–1513.

[34] M. A. Rahman, N. Tabassum, M. Paul, R. Pal, M. K. Islam, "Bn-htrd: A benchmark dataset for document level offline bangla handwritten text recognition (htr) and line segmentation," in *Computer Vision and Image Analysis for Industry 4.0*, Chapman and Hall/CRC, 2023, pp. 1–16.

[35] P. Singh, A. Kumar, "Bn-dhristi: An efficient technique for handwritten document segmentation," *International Journal of Document Analysis and Recognition*, vol. 20, no. 4, pp. 289–301, 2017.

[36] L. Vincent, P. Soille, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

[37] J. H. Holland, "Index," in *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. 1992, pp. 207–211.

[38] T. Rath, R. Manmatha, "A* path planning for text line segmentation in handwritten documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 987–1001, 2007.

[39] L. Likforman-Sulem, A. Zahour, B. Taconet, "Text line segmentation of historical documents: A survey," *International Journal of Document Analysis and Recognition (IJDAR)*, vol. 9, no. 2–4, pp. 123–138, Sep. 2006, issn: 1433-2825. doi: 10.1007/s10032-006-0023-z. [Online]. Available: http://dx.doi.org/10.1007/s10032-006-0023-z.

[40] M. Singh, "Word segmentation from handwritten paragraphs using opencv tools," *Medium*, Sep. 2020. [Online]. Available: https://medium.com/@maniksingh256/word-segmentation-from-handwritten-paragraphs-using-opencv-tools-6ba05dee13b8.

[41] G. Bradski, "The opencv library.," *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[42] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[43] S. Rostianingsih, A. Setiawan, C. I. Halim, "Coco (creating common object in context) dataset for chemistry apparatus," *Procedia Computer Science*, vol. 171, pp. 2445–2452, 2020, Third International Conference on Computing and Network Communications (CoCoNet'19), issn: 1877-0509. doi: https://doi.org/10.1016/j.procs.2020.04.264. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920312564.

[44] J. Van Hulse, T. M. Khoshgoftaar, A. Napolitano, "Experimental perspectives on learning from imbalanced data," *Proceedings of the 24th International Conference on Machine Learning*, ACM, 2007, pp. 935–942.

[45] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[46] A. Sarkar, "All you need to know about 'attention' and 'transformers'—in-depth understanding—part 1," *Towards Data Science*, 2020. [Online]. Available: https://medium.com/towards-data-science/all-you-need-to-know-about-attention-and-transformers-in-depth-understanding-part-1-552f0b41d021.

[47] I. P. G. Louloudis B. Gatos, C. Halatsis, "Text line detection in handwritten documents," *2008 19th International Conference on Pattern Recognition*, IEEE, 2008, pp. 1133–1136.

[48] F. G. A. Graves S. Fernandez, J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," *Proceedings of the 23rd International Conference on Machine Learning*, 2006, pp. 369–376.

[49] J. A. Sánchez, V. Romero, A. H. Toselli, E. Vidal, "Icfhr2016 competition on handwritten text recognition on the read dataset," *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2016, pp. 630–635. doi: 10.1109/ICFHR.2016.0120.

[50] J. Long, E. Shelhamer, T. Darrell, "Fully convolutional networks for semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[51] Y. Liu, M. Ott, N. Goyal, et al., "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019. [Online]. Available: https://arxiv.org/abs/1907.11692.

[52] Z. Huang *et al.*, "Icdar2019 competition on scanned receipt ocr and information extraction," *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 1516–1520.

[53] P. Foret, A. Kleiner, H. Mobahi, B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," *arXiv preprint arXiv:2010.01412*, 2020.

[54] B. Gatos *et al.*, "Ground-truth production in the transcriptorium project," *2014 11th IAPR international workshop on document analysis systems*, IEEE, 2014, pp. 237–241.

PUBLICATIONS FROM THE THESIS

Papers

1. Osman Furkan Karakuş, "Adapting Vision Transformer-Based Object Detection Model for Handwritten Text Line Segmentation Task. Journal of Innovative Science and Engineering, 2025, pp. 1-12."

Conference Papers

1. Osman Furkan Karakuş, "Enhancing Handwritten Text Line Segmentation Using Vision Transformer-Based Object Detection Techniques," presented at the 4th International Graduate Studies Congress (IGSCONG'24), held on 05-08 June 2024.