AUTOMATED TEXT LINE SEGMENTATION FOR OTTOMAN MANUSCRIPT TRANSCRIPTION

by
SEÇİLAY KUTAL

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University
July 2025

AUTOMATED TEXT LINE SEGMENTATION FOR OTTOMAN MANUSCRIPT TRANSCRIPTION

Approved by

PROF. AYŞE BERRİN YANIKOĞLU ....
(Thesis Supervisor)

PROF. ERCHAN APTOULA ....

PROF. MİNE ELİF KARSLIGİL ....
(Yıldız Technical University)

Date of approval: July 7, 2025

ABSTRACT

AUTOMATED TEXT LINE SEGMENTATION
FOR OTTOMAN MANUSCRIPT TRANSCRIPTION

SEÇİLAY KUTAL

COMPUTER SCIENCE AND ENGINEERING M.A. THESIS, JULY 2025

Thesis Supervisor: Prof. AYŞE BERRİN YANIKOĞLU

Keywords: Ottoman, Manuscript, Text Line, Segmentation, Computer Vision.

Text line segmentation is essential for analyzing historical manuscripts. This task becomes challenging with documents handwritten documents, especially Ottoman manuscripts, which often contain complex writing styles and overlapping lines. This thesis presents various deep learning-based approaches for automatic text line segmentation in Ottoman manuscripts. The U-Net-based approach relies on binary segmentation followed by a connected component post-processing. The YOLO-based methods include a single-stage (instance segmentation) and a two-stage (combining object detection and segmentation) approach. Models initially trained on Arabic datasets, are tested on a 40-page Ottoman manuscript dataset containing both straight and angled text lines, using various segmentation metrics. The YOLO approaches, which showed promising results, were further evaluated for their effect on OCR performance using an Ottoman dataset with 199 text lines. The best performance was obtained with the YOLO OBB & Segmentation method, achieving a precision score of 99.3% on straight lines and 95% on angled lines at a 75% IoU threshold. Furthermore, this approach yielded 13.1% CER and 34.7% WER on the OCR evaluation dataset. With this best-performing method, an automatically segmented Ottoman text line segmentation dataset is also generated, resulting in a precision score of 96.5% evaluated on a 20-page subset. The dataset collected in this work, consisting of 40 manually labeled and 110 automatically labeled pages, is publicly shared to contribute to research on line segmentation of Ottoman manuscripts.

ÖZET

OSMANLICA EL YAZMASI BELGELERİN TRANSKRİPSİYONU İÇİN OTOMATİK SATIR BÖLÜTLEMESİ

SEÇİLAY KUTAL

PROGRAM ADI YÜKSEK LİSANS TEZİ, TEMMUZ 2025

Tez Danışmanı: Prof. Dr. AYŞE BERRİN YANIKOĞLU

Anahtar Kelimeler: Osmanlıca, El Yazması, Metin Satırı, Bölütleme, Bilgisayarla Görü.

Metin satırı bölütlemesi, tarihi el yazmalarının analizinde için kritik bir adımdır. Bu işlem, el yazısı belgelerde, özellikle karmaşık yazı stilleri ve üst üste binen satırlar içeren Osmanlıca el yazmalarında zorlu hale gelmektedir. Bu tez, Osmanlıca el yazmalarında otomatik metin satırı bölütlemesi için çeşitli derin öğrenme tabanlı yaklaşımlar sunmaktadır. U-Net tabanlı yaklaşım, ikili segmentasyon ve ardından uygulanan bağlantılı bileşen işlemlerine dayanır. YOLO tabanlı yöntemler ise tek aşamalı (bölütleme) ve iki aşamalı (nesne tespiti ve bölütlemeyi) yaklaşımları kapsamaktadır. Başlangıçta Arapça veri seti üzerinde geliştirilen yöntemler, çeşitli bölütleme metrikleri kullanılarak 40 sayfadan oluşan ve düz ile açılı metin satırları içeren Osmanlıca el yazması veri seti üzerinde değerlendirilmiştir. Yüksek başarı gösteren YOLO yaklaşımları, ayrıca 199 satırlık bir Osmanlıca veri setiyle karakter tanıma performansı açısından da değerlendirilmiştir. En iyi sonuç, %75 IoU eşiğinde düz satırlarda %99.3 ve açılı satırlarda %95 kesinlik skoru ile YOLO OBB & Segmentasyon yöntemiyle elde edilmiştir. Yöntem ayrıca, karakter tanıma veri setinde %13.1 CER ve %34.7 WER elde etmiştir. En iyi performansı gösteren bu yöntemle, otomatik olarak Osmanlıca metin satırı bölütlemesi veri kümesi de oluşturulmuş ve 20 sayfalık bir alt kümede %96,5 hassasiyet puanı elde edilmiştir. Bu çalışmada toplanan, 40 adet manuel olarak etiketlenmiş ve 110 adet otomatik olarak etiketlenmiş sayfadan oluşan veri kümesi, Osmanlı el yazmalarının satır segmentasyonu araştırmalarına katkıda bulunmak üzere kamuoyuyla paylaşılmıştır.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my thesis advisor, Prof. Berrin Yanıkoğlu, Despite long periods without communication on my part, she always welcomed me back with open arms whenever I knocked on her door. I am fully aware that not every advisor would show such patience and understanding, and for that, I am truly thankful.

One of the most formative moments in this journey was attending Prof. Erchan Aptoula's Computer Vision class on the very first day of graduate school, which immediately gave me the sense that I had found where I truly belonged. Later, having the opportunity to work with him as a teaching assistant for a full year further deepened my interest in the field and shaped my academic perspective in a lasting way.

This journey would have been much harder without the support and friendship of my fellow graduate friends Zeynep, Egemen, Hasan, and Mekan. Sharing the challenges and frustrations of graduate school with them turned difficult moments into shared memories and created bonds I will always cherish.

I am also sincerely grateful to the members of VPALab, who made the lab a welcoming and comfortable environment.

I also want to acknowledge myself - for the determination it took to keep going. Balancing full-time work with graduate studies required sacrificing countless hours of personal time, and I am proud of the perseverance that made this achievement possible.

Throughout my academic and professional life, I have been incredibly fortunate to have a constant source of support and encouragement. Since our undergraduate days, Semih has stood by me through every challenge and setback, always believing in my potential even when I struggled to believe in myself. His unwavering encouragement and presence gave me the strength to overcome toughest moments.

Finally, I owe my deepest thanks to my parents, Seçil and Engin, who have stood by me through every step of my life with unconditional love and constant support. Their sacrifices and encouragement have carried me through every obstacle, inspiring me to keep moving forward. I am forever grateful for their presence in my life and the foundation they have provided for all my achievements.

Dedicated to
my loved ones...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATONS

ASCII	American Standard Code for Information Interchange	25, 52
BCE	Binary Cross Entropy	21
BiLSTM	Bi-directional Long Short-Term Memory	3, 6
C2PSA	Cross Stage Partial with Spatial Attention	18
CC	Connected Component	4
CER	Character Error Rate	3, 31, 44, 45
CLAHE	Contrast-Limited Adaptive Histogram Equalization	28
CNN	Convolutional Neural Networks	3, 16, 30
CTC	Connectionist Temporal Classification	6
Faster R-CNN	Faster Region-Convolutional Neural Network	5
FN	False Negative	29
FP	False Positive	29
GAN	Generative Adversarial Network	6
HMM	Hidden Markov Models	3
HPP	Horizontal Projection Profile	4
HTR	Handwritten Text Recognition	8
ICDAR	International Conference on Document Analysis and Recognition	6, 7
ICFHR	International Conference on Frontiers of Handwriting Recognition	6
ID	Identifier	13, 23, 25
IoU	Intersection over Union	18, 28, 29, 34, 41, 42, 44, 45, 46

LSTM	Long Short-Term Memory	3
mAP	Mean Average Precision	3, 44
OBB	Oriented Bounding Box	15, 26, 27, 41, 42, 44, 46, 47, 52
OCR	Optical Character Recognition	1, 2, 3, 8, 11, 28, 30, 47
ReLU	Rectified Linear Unit	16, 17, 20
RGB	Red Green Blue	22
RU-net	Recurrent U-Net	5
SPPF	Pyramid Pooling – Fast	18
TP	True Positive	29
WER	Word Error Rate	3, 31, 44, 45
YOLO	You Only Look Once	15, 17, 18, 25, 46, 47, 54

1. INTRODUCTION

The digitization of historical documents plays a crucial role in advancing academic research. As historical records gain increasing importance in contemporary science, they provide valuable insights about past civilizations. Ottoman Turkish is a language which was widely used in the Ottoman Empire between 13th and 20th centuries. Despite no longer being actively in use, documents in Ottoman Turkish can be valuable for learning the mysteries of the Ottoman Empire and tracing the historical roots of modern Türkiye. As such, they constitute essential resources for historical research (Ekinci, 2008; Kolodziejczyk, 2022).

Ottoman Turkish is a Turkish language that was developed under the influence of Arabic and Persian languages. Given the linguistic heritage of Ottoman Turkish, the translation of Ottoman documents into modern Turkish is a crucial step in their analysis. A prerequisite for facilitating this translation is digitization through Optical Character Recognition (OCR). Although OCR performs well on printed texts due to their consistent formatting and preserved image quality, the process becomes more complex for handwritten documents. The challenges arise from inconsistencies in handwriting, complex features of Ottoman script - such as diacritical marks and long-tailed forms of characters - and problems such as overlapping text and irregular alignment of lines. To navigate these difficulties effectively, a multistage approach is recommended, beginning with text line segmentation.

This study investigates text line segmentation as a solution to the challenges introduced by Ottoman manuscripts in the OCR process. The problem was approached through three distinct methods that utilize two different model architectures. Model training was conducted on publicly available Arabic datasets while testing was performed on a manually labeled Ottoman dataset, due to the lack of Ottoman datasets for segmentation tasks.

2. RELATED WORK

The preservation and analysis of Ottoman documents are of great importance for historical research, linguistic studies, and cultural heritage (İlhan, 1991). These documents, spanning several centuries, contain valuable information about the administrative, legal, literary and social aspects of the Ottoman Empire. However, they are difficult to access and study due to their cursive nature, linguistic complexity and the use of Ottoman Turkish, which uses a modified Arabic alphabet.

This section presents a comprehensive review of the literature in two primary domains. First, it examines studies on Ottoman recognition systems, with a particular focus on OCR and text recognition. It highlights the significance of these processes and the unique challenges posed by Ottoman script, such as its complex features and lack of annotated datasets. It further outlines existing methodologies, recent developments, and current limitations in the recognition of handwritten Ottoman texts, thus establishing a foundation for addressing these challenges in subsequent research.

Second, the section explores research on text line segmentation beyond Ottoman documents, with particular attention to languages that use Arabic script. This broader review helps to simulate the challenges encountered in Ottoman text segmentation, given the structural and orthographic similarities between Ottoman and Arabic handwriting.

By synthesizing insights from both domains, this review situates the present work within the broader context of historical document processing and establishes a foundation for the proposed methods.

2.1 Ottoman Recognition Systems

Manual transcription and analysis methods are time-consuming, driving the need for automatic recognition systems to support large-scale digitization. Researchers are increasingly adopting OCR and text recognition technologies to convert scanned documents into machine-readable text. This subsection reviews existing OCR and text recognition research on Ottoman documents, highlighting key methodologies, common challenges, and recent progress.

In a study using Hidden Markov Models (HMM), Onat, Yildiz & Gündüz (2006) developed an OCR system to recognize Ottoman handwritten characters. Despite its innovative approach, the system only achieved a recognition accuracy of 65%, highlighting the difficulties associated with processing Ottoman script.

Bilgin Tasdemir (2023) tackled the lack of publicly available Ottoman datasets by creating a recognition system trained on synthetic data. The system utilized a hybrid Convolutional Neural Networks (CNN) and Bi-directional Long Short-Term Memory (BiLSTM) architecture, enhanced with transfer learning to improve performance on real-world data. The system, trained on printed texts, achieved a Character Error Rate (CER) of 0.11 on the synthetic dataset and 0.16 on the real dataset.

In a study on Ottoman document retrieval, Ataer & Duygulu (2006) proposed a three-step approach comprising text line segmentation, word segmentation, and a hierarchical matching method to identify similar word images. The study focused on relatively clean printed documents, achieving 100% accuracy in text line segmentation, an 82% average word extraction rate, and a 85% Mean Average Precision (mAP) value for word matching on the evaluated dataset.

Dolek & Kurt (2021), in their OCR tool developed for printed Ottoman documents in Naksh font, once again highlighted the lack of Ottoman datasets in the literature. To address this challenge, they utilized a dataset comprising both original and synthetically generated documents. The system, built using a Long Short-Term Memory (LSTM) architecture, achieved 88.64% raw, 95.92% normalized, 97.18% combined character recognition accuracy rates, and 58% word recognition accuracy.

In contrast to traditional multi-stage methods, Tasdemir, Tandoğan, Akansu, Kızılırmak, Sen, Akcan, Kuru & Yanikoglu (2024) introduced a single-stage transcription system for Ottoman documents built on a CNN-BiLSTM architecture. Evaluation on a dataset comprising 6,800 text line images yielded a CER of 6.59% and a Word Error Rate (WER) of 28.46%.

2.2 Handwritten Text Line Segmentation

The problem of character recognition presents unique challenges in manuscripts compared to printed documents, primarily due to the inherent complexities of handwritten text. In manuscripts, closely spaced lines, overlapping characters, inconsistent line alignment, and writer-dependent variations in character shapes introduce significant difficulties. These challenges are particularly pronounced in Ottoman and Arabic scripts, where additional factors, such as context-dependent character shapes, connected letters, and characters with diacritical marks and elongated tails, further complicate recognition. At this stage, text line segmentation becomes a crucial preprocessing step in end-to-end text and character recognition systems. Effective segmentation can minimize recognition errors and significantly enhance system performance. This subsection reviews studies on text line segmentation across different languages, highlighting the significance of the problem and its impact on text recognition systems.

In their study on Arabic handwritten text line segmentation, Meziani, Bouchakour, Ghribi, Yahiaoui, Latrache & Abbas (2021) proposed a hybrid technique combining Horizontal Projection Profile (HPP), Connected Component (CC) analysis, and skeleton-based segmentation. The process begins by approximating line positions using HPP, after which CCs are used to separate non-adjacent lines. For overlapping or closely spaced lines, skeleton analysis is applied to detect intersection points and refine the segmentation. The method achieved an F-measure of 85% with a 90% MS2 (Daldali & Souhar, 2019) threshold. Although the approach demonstrated strong performance in distinguishing close and touching lines, it struggled in scenarios involving lines with multiple skew directions, limiting its overall robustness.

In their study on offline Uyghur handwritten text line segmentation - a script similar to Ottoman (Figure 2.1) - Suleyman et al. (2021) introduced an algorithm based on a projection-based adaptive threshold selection method. Unlike conventional techniques, their approach is insensitive to text line length, making it adaptable to various handwritten scripts. The proposed method was tested on Uyghur and Polish datasets, encompassing 1,474 text lines, where it achieved 97.70% recall on the Uyghur dataset and 63.23% recall on the Polish dataset. Despite its effectiveness in certain cases, the single-projection approach resulted in poor performance for skewed text lines, sometimes leading to segmentation failures. Additionally, closely spaced lines in handwritten texts introduced further challenges by disrupting peak extraction in the projection profile, negatively affecting segmentation accuracy.

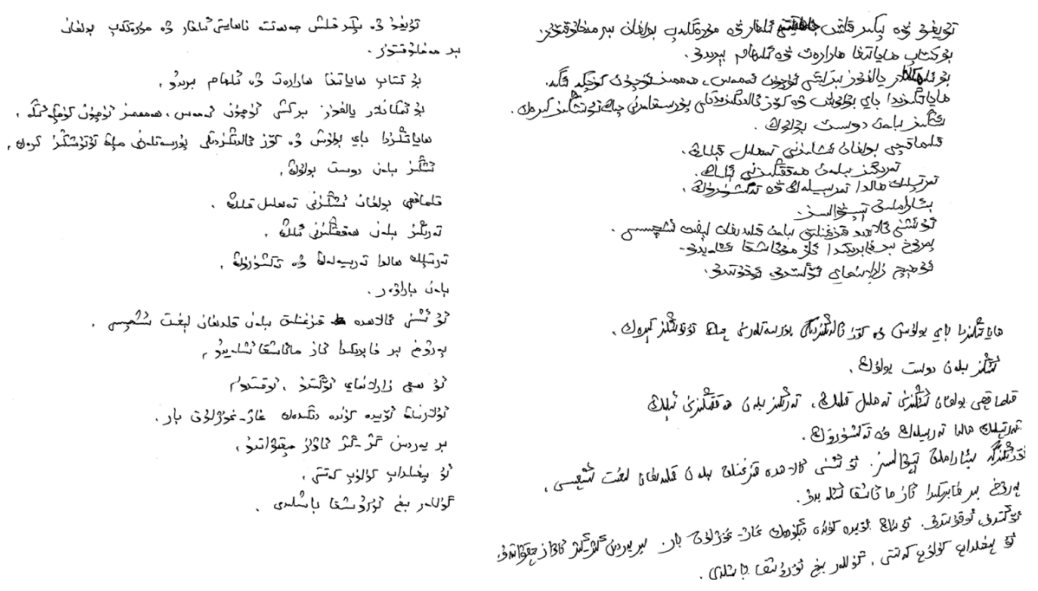


Figure 2.1 Sample images from the dataset used in the study by Suleyman et al. (2021)

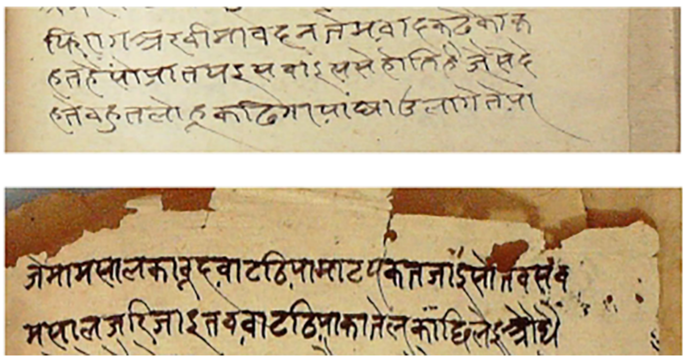


Figure 2.2 Sample images from the dataset used in the study by Jindal & Ghosh (2023)

Jindal & Ghosh (2023) addressed the challenge of text line segmentation in ancient handwritten documents, marking the first study in the literature to focus on Devanagari script. The dataset used was outdated and partially corrupted, and the characteristics of Devanagari script resemble the complexities encountered in Ottoman manuscripts (Figure 2.2). They proposed a Faster Region-Convolutional Neural Network (Faster R-CNN)-based approach for text line segmentation. Their method achieved an F-measure of 99.98% on a Devanagari document dataset, which they introduced in their study. They also noted that the model struggles with skewed and overlapping lines, as it fails to produce separate bounding boxes in these more complex layout scenarios.

Neche, Belaid & Kacem-Echi (2019) proposed two deep learning-based approaches for Arabic text line and word segmentation. For text line segmentation, they utilized the Recurrent U-Net (RU-net) architecture, which performs pixel-level segmentation to distinguish between text lines and background pixels. Their method relied on x-height labeling ((Renton, Chatelain, Adam, Kermorvant & Paquet, 2017)), followed by a post-processing step for baseline extraction. For word segmentation, they introduced a method based on BiLSTM followed by Connectionist Temporal Classification (CTC). Their experiments on the Arabic KHATT dataset (Mahmoud, Ahmad, Alshayeb, Al-Khatib, Parvez, Fink, Märgner & Abed, 2012) yielded an F-measure of 96.71% for text line segmentation and 80.1% for word segmentation.

Demir, Özşeker & Özkaya (2021) addressed the segmentation of handwritten text lines in documents with complex page layouts, including curved, multi-skewed, and multi-directional lines. They proposed a Generative Adversarial Network (GAN)-based architecture, employing U-Net (Ronneberger, Fischer & Brox, 2015b) as the generator and PatchGAN (Isola, Zhu, Zhou & Efros, 2017) as the discriminator. The model was trained and evaluated on the Challenging Text Line Dataset, an Arabic dataset labeled using line masks. A subset of 24 images was used for training via random cropping, and the model was tested on 4 held-out images. The system achieved 83% precision, 88% recall, and an F-measure of 85%. The authors noted segmentation inconsistencies, particularly in skewed lines, and recommended future improvements by modifying the loss function to include segmentation-specific terms and exploring alternative generators within the proposed architecture.

Barakat, Droby, Alaasam, Madi, Rabaev, Shammes & El-Sana (2021) proposed an unsupervised deep learning approach for the segmentation of handwritten text lines, leveraging both the text lines and the inter-line spaces. The method embeds image patches in an unsupervised manner, requiring no manual annotations. The model learns to distinguish text from the spaces based on differences in the number of foreground pixels between the two regions. Evaluated on Arabic datasets - VML-AHTE, International Conference on Document Analysis and Recognition (ICDAR) 2017 (Simistira, Bouillon, Seuret, Würsch, Alberti, Ingold & Liwicki, 2017), and International Conference on Frontiers of Handwriting Recognition (ICFHR) 2010 (Gatos, Stamatopoulos & Louloudis, 2010) - the method demonstrated robust performance in segmenting lines of uniform height, even in complex scenes. However, it exhibited reduced performance when faced with variations in line height or irregular text arrangements.

Adıgüzel, Şahin & Kalpaklı (2012) proposed a method for text line segmentation based on projection profiles. Using this approach, they achieved a segmentation accuracy of 96% on a 20-page Ottoman manuscript dataset characterized by straight text lines and relatively wide interline spacing.

3. DATASETS

Given the limited availability of Ottoman datasets in the literature, alternative datasets in other languages were considered for model development. Due to the linguistic and structural similarities between Ottoman Turkish and Arabic, the Arabic dataset discussed in Section 3.1 was selected for training purposes. The performance of the developed systems was then evaluated using the Ottoman dataset introduced in Section 3.2.

3.1 RASM2019 Dataset

RASM2019 dataset (British Library & Keinan-Schoonbaert, 2019), released in 2019 for the ICDAR Competition on the Recognition of Historical Arabic Scientific Manuscripts, serves as a benchmark for layout analysis and text recognition in historical document processing.

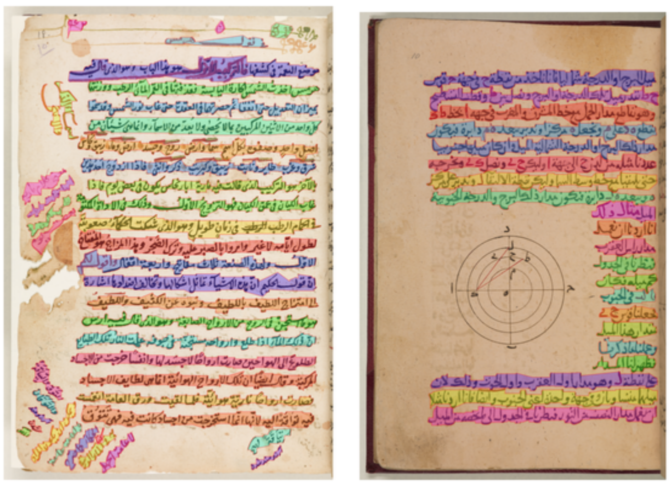


Figure 3.1 Sample images from the RASM dataset (British Library & Keinan-Schoonbaert, 2019) (with each text line highlighted in a different color)

The dataset contains 120 high-resolution TIFF images, drawn from Arabic scientific manuscripts spanning the 10th to 19th centuries. Each image is accompanied by an XML file providing detailed ground truth annotations, including text block boundaries and text line transcriptions, which make the dataset suitable for layout analysis, OCR, and Handwritten Text Recognition (HTR) tasks (Figure 3.1).

An analysis of the number of text lines per page in the dataset revealed that the majority of pages contain between 15 and 25 lines (Figure 3.2). However, pages with a significantly higher number of lines were also observed, representing outliers in the distribution.

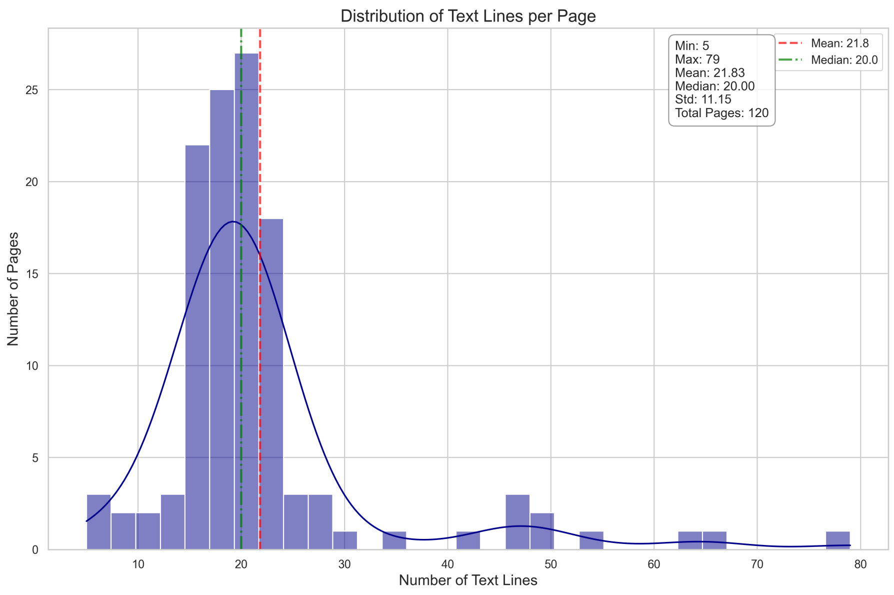


Figure 3.2 Distribution of text lines per page for RASM (British Library & Keinan-Schoonbaert, 2019) dataset

Bounding boxes were generated for each polygon in the dataset, and the corresponding widths and heights were analyzed (Figure 3.3). The results indicate that polygon widths fall between 3000 and 4000 pixels, with corresponding text lines averaging around 500 pixels in height. Additionally, a secondary cluster of text lines with widths below 1000 pixels was also identified in the graph, characterized by greater variability in terms of height. Despite these differences, the majority of text lines exhibit heights below 1000 pixels. These findings indicate that the text lines in the dataset demonstrate a heterogeneous distribution in terms of both size and orientation.

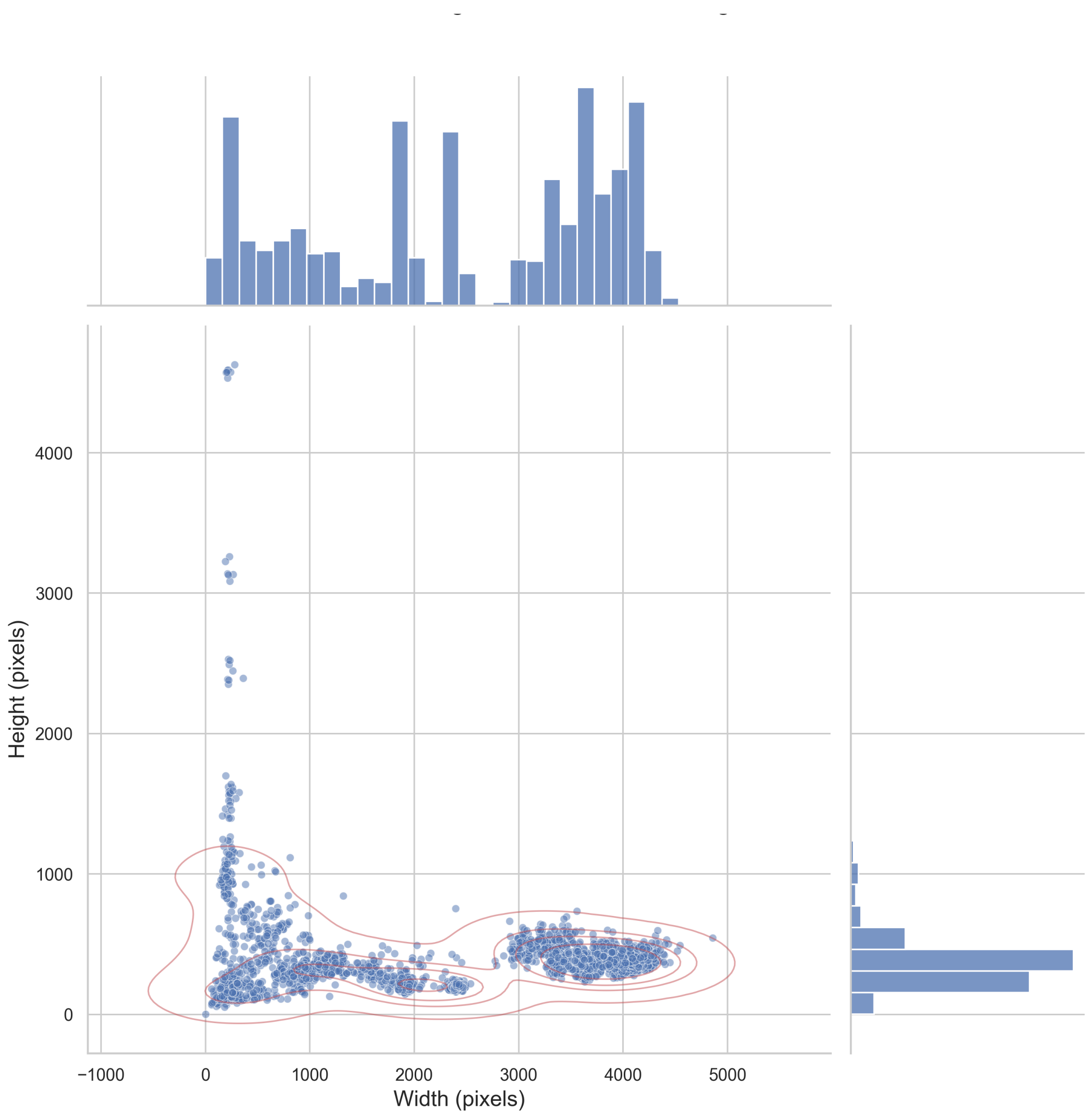


Figure 3.3 Joint distribution of polygon widths and heights for RASM (British Library & Keinan-Schoonbaert, 2019) dataset

3.2 Asir Efendi Dataset

For this study, a custom dataset - referred to as the Asir Efendi Dataset - was created and manually labeled at the pixel level to facilitate the identification of individual text lines (Section 3.2.1). The resulting dataset is publicly shared on GitHub [1].

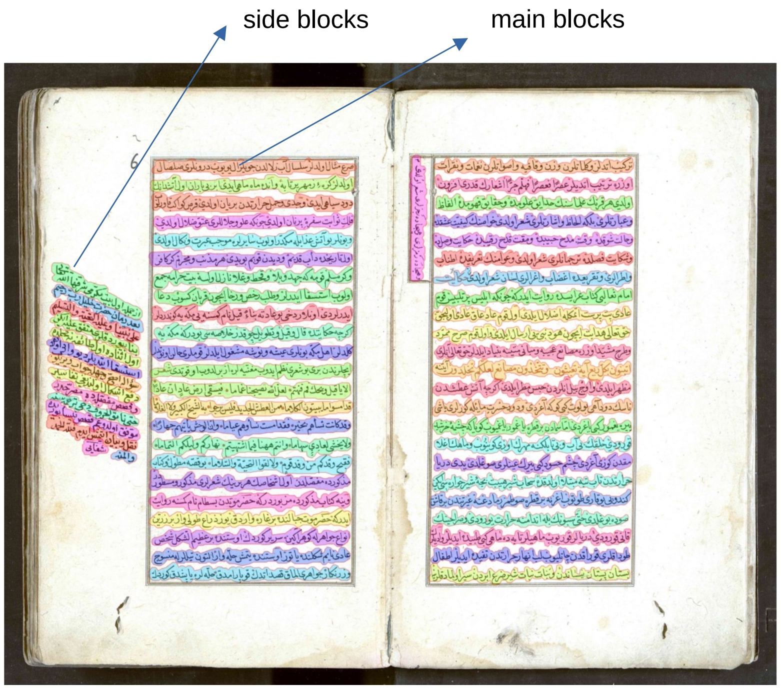


Figure 3.4 Sample image from the Asir Efendi Dataset (with each text line highlighted in a different color)

The dataset includes 40 images of digitized Ottoman manuscripts, with each image containing two adjacent manuscript pages. Text lines within the images are evaluated in two categories: main blocks, which consist of straight, consistently aligned lines, and side blocks, which contain text lines with varying orientations and misalignments across the manuscript pages (Figure 3.4).

The analysis of the number of text lines per page (or image) in the dataset showed that most pages contained approximately 46 lines (Figure 3.5). Each page in the dataset represents an image that scans two manuscript pages; thus, each individual manuscript page contains about 23 lines. Additionally, the presence of side blocks on some pages contributed to an increase in the number of lines detected per page.

Bounding boxes were generated for each polygon in the dataset, and their corresponding widths and heights were analyzed (Figure 3.7). The results showed that

[1] https://github.com/seccily/ottoman-text-line-segmentation-dataset

the polygon widths formed two main distinct clusters of polygon widths: one between 300 and 400 pixels and another between 600 and 800 pixels. In both cases, the text line height was consistently maintained between 50 and 100 pixels.

To evaluate the effect of the developed models on OCR, an additional dataset - comprising the remaining pages of the Asır Efendi collection - was included in the study. This dataset contains rectangular crops of straight text lines extracted from the main blocks on each page, along with their corresponding transcriptions (Figure 3.6).

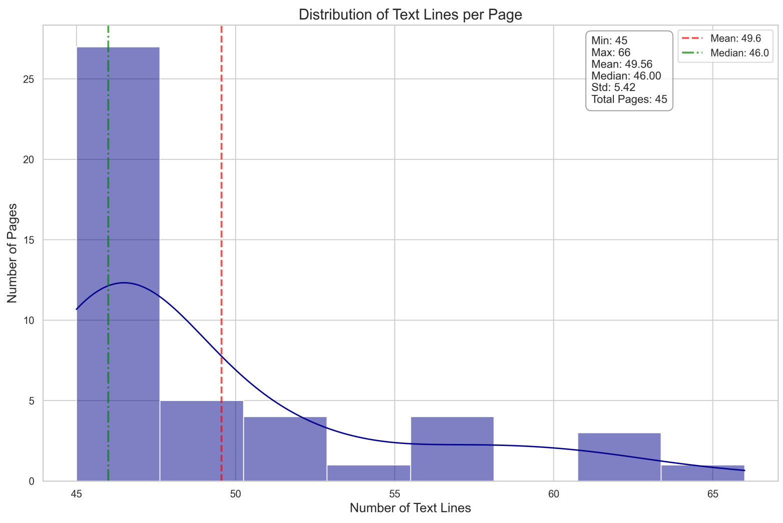
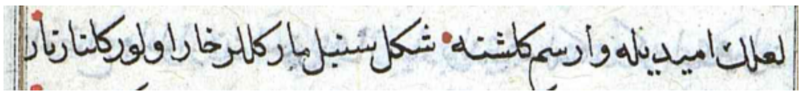
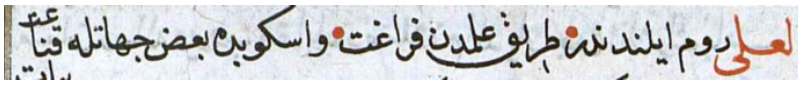


Figure 3.5 Distribution of text lines per page for Asir Efendi dataset



(a) Transcription: *la'lün ümmidiyle varsam gülşene şekl-i sünbül mar güller har olur gülnar nar*



(b) Transcription: *la'li rumili'ndendür tarik-i 'ilmden feragat ve üsküb'de ba'zı cihatla kana'at*

Figure 3.6 Transcription sample from the Asir Efendi dataset for OCR evaluation

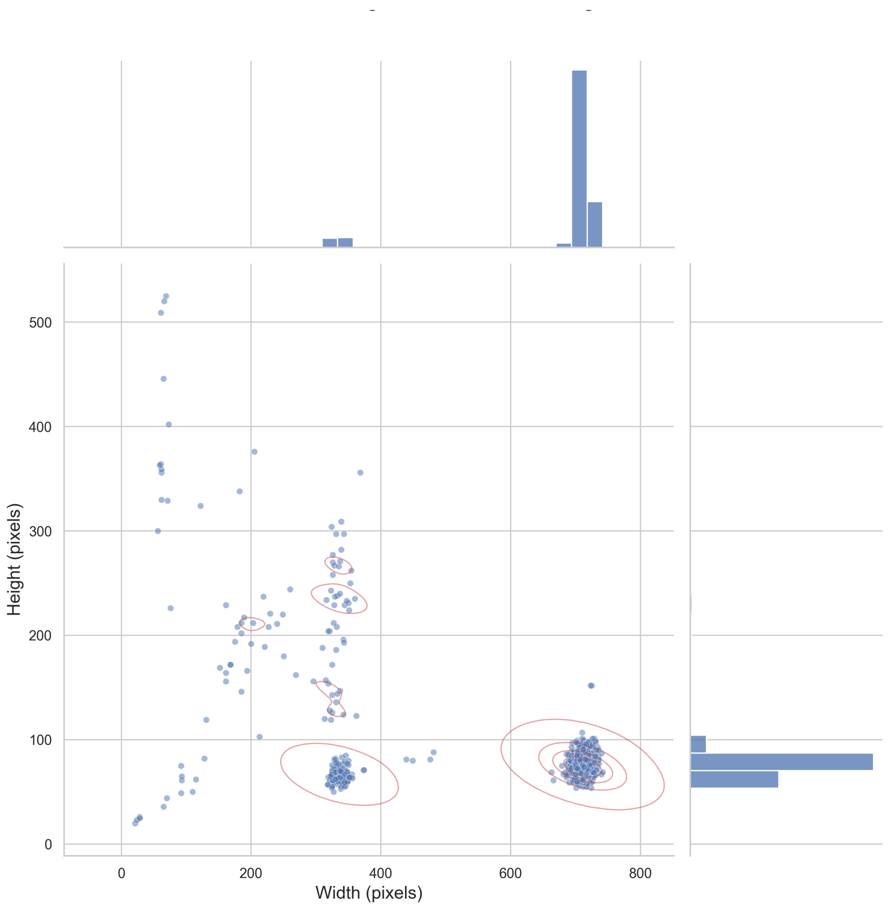


Figure 3.7 Joint distribution of polygon widths and heights for Asir Efendi dataset

3.2.1 Segmentation Labeling

The Label Studio (Tkachenko, Malyuk, Holmanyuk & Liubimov, 2020) tool, an open-source and user-friendly annotation tool that supports labeling of audio, image, text, video, and time series data for a variety of tasks, was used for the annotation process.

Label Studio supports both brush and polygon annotation methods for semantic segmentation tasks. In the polygon annotation method, the boundaries of the area to be segmented are defined by marking data points to form a closed shape. In contrast, the brush annotation method allows the user to paint the area directly. In terms of labeling efficiency, the polygon method is less effective, as it requires marking a large number of points, particularly when segmenting irregularly shaped objects. In this study, the brush tool was selected due to its efficiency, enabling a faster annotation process compared to the polygon-based method (Figure 3.8).

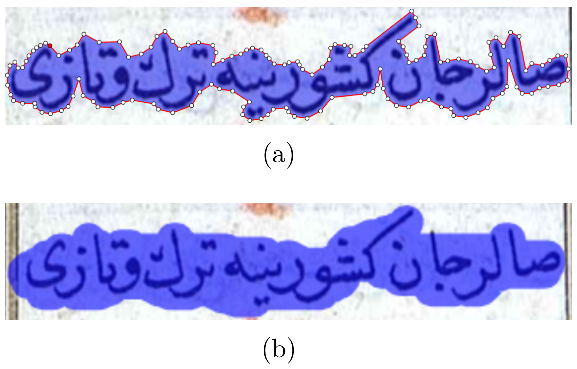
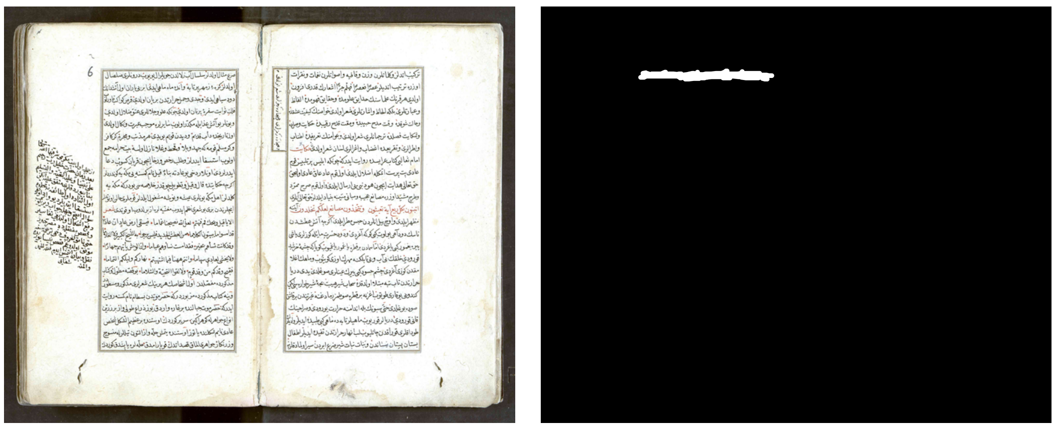


Figure 3.8 Polygon (a) and brush (b) labeling samples

As a result of the labeling process, the exported labels consisted of multiple binary masks, each representing an individual line of text within an image using a pixel value of 1 (Figure 3.9). These labels were subsequently converted into two formats to accommodate different model input requirements: a single binary mask per image and a polygon-based format. All exported masks were combined to produce a single binary mask for each image. For the polygon format, each individual binary mask was first converted into a polygon using the OpenCV library (Bradski, 2000), and the resulting polygon coordinates, along with class Identifier (ID)s, were compiled into a single `.txt` file for each image.



(a) Input image (b) Mask label for the first line

Figure 3.9 Exported annotation sample for the Asir Efendi dataset

4. METHODOLOGY

The methods explored in this study leverage object detection, semantic segmentation, and instance segmentation tasks of computer vision to address the problem of text line segmentation.

Object detection is defined as the task of identifying the location and class of objects within an image. Each object is represented by a corresponding bounding box that encloses it (Zou, Chen, Shi, Guo & Ye, 2023). Semantic segmentation refers to the pixel-wise classification of an image, where all objects of the same class are categorized together. In contrast, instance segmentation not only classifies but also differentiates between individual objects within the same category (Minaee, Boykov, Porikli, Plaza, Kehtarnavaz & Terzopoulos, 2021).

In this study, three segmentation approaches were developed based on the U-Net (Ronneberger et al., 2015a) and You Only Look Once (YOLO) (Redmon et al., 2015) architectures. The U-Net-based method performs semantic segmentation through binary segmentation to distinguish text lines from the background, followed by a connected component post-processing step to separate individual text lines (Section 4.2). In contrast, YOLO-based methods utilize instance segmentation (Section 4.3); one further integrates Oriented Bounding Box (OBB) object detection step to improve localization accuracy (Section 4.4). Among these three approaches, the final approach demonstrated the highest overall performance.

4.1 Background

4.1.1 U-Net

U-Net (Ronneberger et al., 2015a) is a CNN architecture originally developed for biomedical image segmentation. The model employs a training strategy based on data augmentation to make more efficient use of limited datasets.

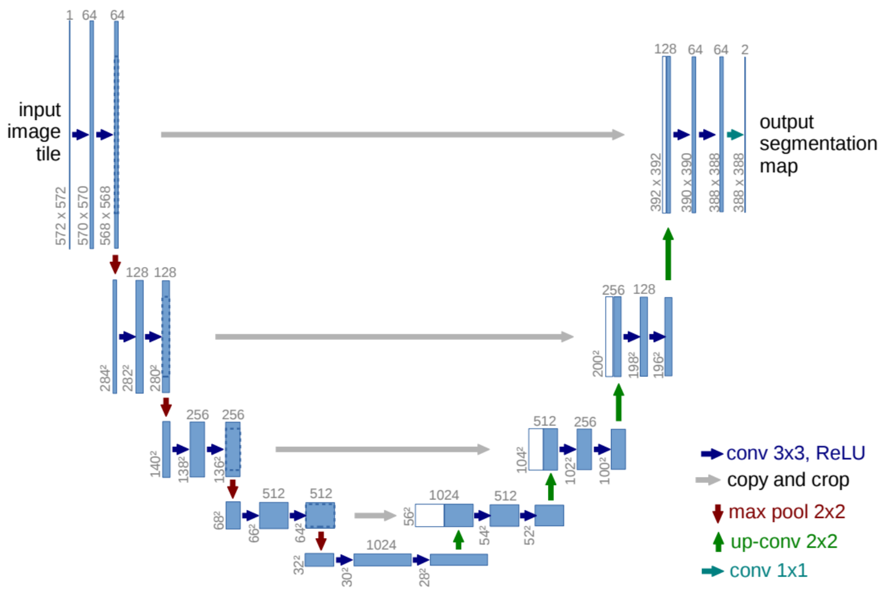


Figure 4.1 Original U-Net architecture (Ronneberger et al., 2015a)

The network comprises two distinct parts (Figure 4.1): a contracting path, which captures semantic context through downsampling operations, and an expanding path, which performs precise localization via upsampling symmetric to the contracting path. This dual-path structure allows U-Net to achieve high accuracy in pixel-wise segmentation tasks.

The U-Net model consists of 23 convolutional layers in total. In the contracting path, each block is composed of a 3×3 convolution without padding, a Rectified Linear Unit (ReLU) activation, and a 2×2 max pooling operation with a stride of 2. This structure progressively reduces spatial resolution while doubling the number of feature channels at each step. In the expanding path, each layer begins with a 2×2 up-convolution, followed by concatenation with the corresponding feature map from the symmetrical layer in the contracting path. This is followed by a 3×3 convolution and a ReLU activation. Finally, the final output layer uses a 1×1 convolution to map the feature representation to the number of target classes.

While the U-Net architecture was initially designed for biomedical segmentation tasks, it has significantly influenced subsequent research across various domains, such as underwater imaging (Nezla, Mithun Haridas & Supriya, 2021), remote sensing (Wang & Miao, 2022), and other computer vision applications (Shaukat, Farooq, Tu, Xiao & Ali, 2022; Zaib, ur Rehman & Khursheed, 2024; Zhang & Zhang, 2023).

4.1.2 You Only Look Once (YOLO)

YOLO (Redmon et al., 2015) is a deep learning-based object detection framework that formulates object recognition as a regression problem. Rather than treating detection as a pipeline of separate stages, YOLO predicts bounding boxes and their corresponding class probabilities directly in a single pass through the network. This architecture enables the system to achieve exceptionally fast inference speeds, while maintaining competitive accuracy , making it suitable for real-time applications.

The YOLO architecture is composed of 24 convolutional layers followed by 2 fully connected layers (Figure 4.2). The convolutional layers are used to extract spatial and semantic features from the input image, while the fully connected layers are responsible for estimating bounding box coordinates and class probabilities.

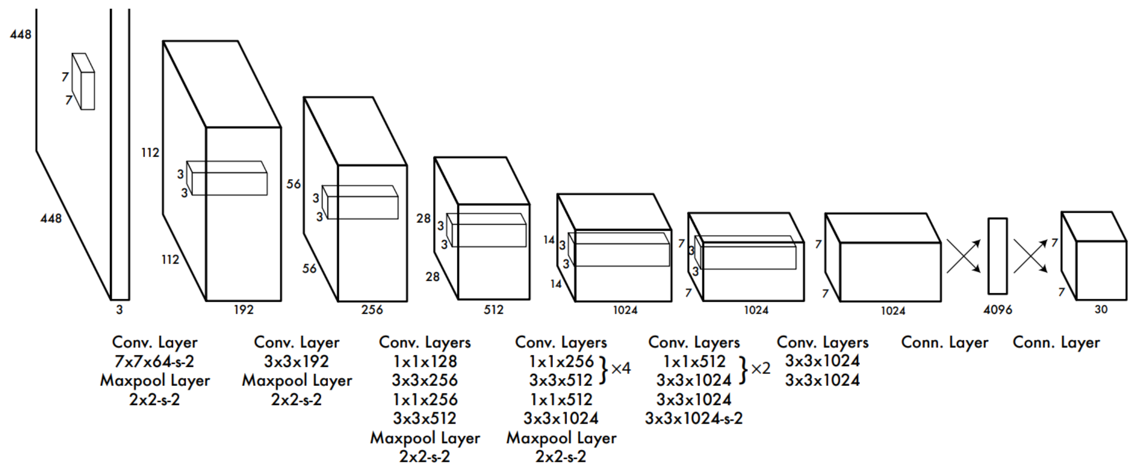


Figure 4.2 Original YOLO architecture (Redmon et al., 2015)

The system divides the input image into an S × S grid, with each grid cell responsible for detecting objects whose center point falls within the cell. Each cell predicts bounding boxes, class labels, and confidence scores. Each bounding box is defined by five parameters: the x and y coordinates of the center of the box, width and height of the box, and a confidence score, which represents the Intersection over Union (IoU) between the predicted and ground truth bounding boxes.

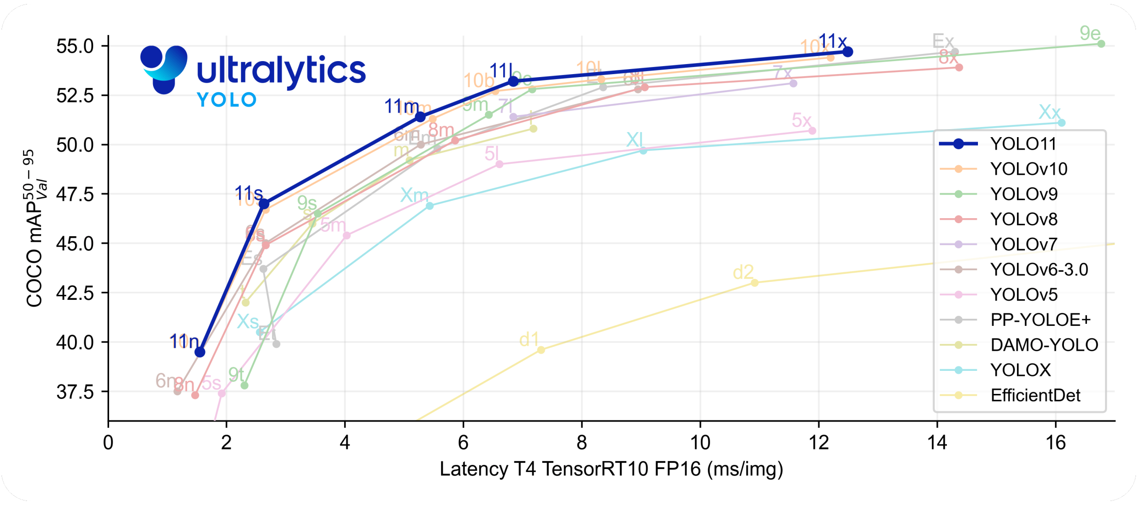


Figure 4.3 Benchmarks for various YOLO versions (Jocher et al., 2023)

The ongoing developments in artificial intelligence and computer vision have led to the evolution of multiple versions of the YOLO architecture over time from YOLOv1 to YOLOv11 (Khanam & Hussain, 2024), as illustrated in Figure 4.3. In this research, the most recent and state-of-the-art variant, YOLOv11 (Jocher et al., 2023), was selected to implement the proposed segmentation approaches.

Since the introduction of YOLOv1, the YOLO family has undergone continuous development, with each version introducing significant improvements. YOLOv11 continues this trend, introducing enhancements that improve detection speed, accuracy, and computational performance (Sapkota, Qureshi, Calero, Badjugar, Nepal, Poulose, Zeno, Vaddevolu, Khan, Shoman, Yan & Manoj, 2024).

One of the YOLOv11's key innovations is the C3k2 block, which replaces the C2f block used in earlier versions. This modification enhances the feature extraction process, making it both faster and more efficient. In addition, the integration of the Cross Stage Partial with Spatial Attention (C2PSA) block, placed after the Spatial Pyramid Pooling – Fast (SPPF) block—a component already present in prior versions—improves spatial attention in the generated feature maps. These combined innovations represent a substantial improvement over previous versions, positioning YOLOv11 as a state-of-the-art solution for object detection.

Certain models in the YOLO family, such as YOLO11-seg and YOLO11-pose (Jocher et al., 2023), are designed not only for object detection, but also for tasks such as pose estimation, segmentation, oriented bounding box detection, and image classification.

These task-specific capabilities are enabled by architectural differences in the model's head. For instance, in the case of segmentation, an additional head is incorporated into the model to perform mask prediction alongside standard object detection.

4.2 U-Net Approach

In this approach, the U-Net architecture was utilized to leverage its ability to preserve spatial details across layers. The primary objective is to segment text lines as a single class (i.e., binary segmentation), followed by connected component analysis on the resulting mask to separate individual lines.

As illustrated in Figure 4.4, the U-Net consists of a contracting path with five encoder block, a bridge connection, and an expanding path with five decoder blocks.

Each encoder block comprises a convolution block followed by a 2D max pooling layer with a 2×2 kernel. The convolution block includes two repetitions of a sequence containing a 2D convolution layer (without activation), batch normalization, and a LeakyReLU activation layers. Padding is applied in all convolution layers to maintain input dimensions, ensuring that downsampling occurs only during max pooling. The bridge connection consists of a convolution block with a filter size of 1024. Each decoder block includes a 2×2 transposed convolution layer with a stride of 2, followed by concatenation with the corresponding encoder feature map, and a convolution block identical to those in the encoder. Finally, the output is passed through a 1×1 convolution layer with a softmax activation function, mapping the result to the number of output classes.

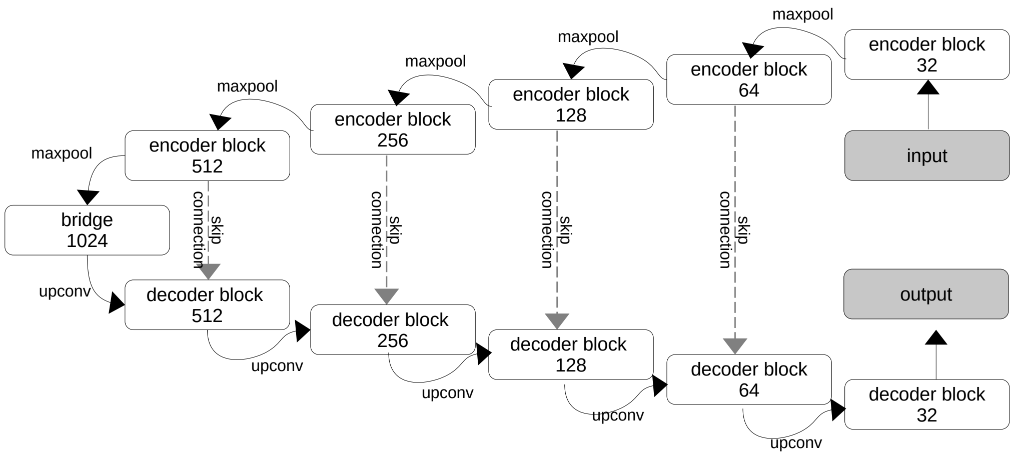


Figure 4.4 Implemented architecture of the U-Net approach

4.2.1 Loss Functions

The proposed segmentation architecture was implemented and evaluated using four different loss functions to analyze their impact on performance:

1.1 **Combined Loss:** To optimize both pixel-level classification and segmentation accuracy, a weighted combination of Dice Loss and Binary Cross Entropy (BCE) was used. Each component was assigned a weight of 0.5 to ensure equal weight distribution for segmentation and classification tasks in the combination.

Dice Loss, as proposed by Milletari, Navab & Ahmadi (2016), evaluates segmentation performance by maximizing the Dice coefficient, which quantifies the similarity between the predicted segmentation and the ground truth. The formulation is provided in Equation (4.1), where p_i represents the prediction and g_i the ground truth value.

$$DiceCoef = \frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2} \tag{4.1}$$

BCE, on the other hand, measures the discrepancy between predicted probabilities and ground truth labels. It is defined in Equation (4.2), where y is the ground truth label and p_i is the predicted probability of the class being positive.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i) \tag{4.2}$$

1.2 **Focal Loss :** Focal loss is designed to down-weight easy examples during training, allowing the model to focus more on hard-to-classify instances (Lin, Goyal, Girshick, He & Dollár, 2017). In the context of this study, it was employed to mitigate class imbalance between background (easy) and foreground (text line - hard) pixels, encouraging the model to pay greater attention to the foreground regions. Focal loss can be defined as in Equation (4.3), where p_t is the probability of the true class estimated by the model and γ is the focusing parameter that controls how much the loss focuses on hard examples.

$$FocalLoss(p_t) = -(1-p_t)^{\gamma}\log(p_t) \tag{4.3}$$

1.3 **Foreground-Weighted Binary Cross Entropy:** This variant of BCE applies higher weights to foreground (text line) pixels in order to address the imbalance between background and foreground regions. The goal is to increase the model's sensitivity to text line areas.

1.4 **Boundary-Weighted Binary Cross Entropy:** In this approach, the loss function gives higher importance to pixels located at the boundaries between text lines and background areas. This encourages the model to produce more accurate segmentation along the edges of text lines, where misclassifications are more likely to occur.

4.2.2 Data Preparation

The input of the U-Net model consists of page images and their corresponding binary masks, where each mask represents the text lines as foreground. To reduce the input dimensionality, the original Red Green Blue (RGB)-formatted page images were first converted to grayscale using the luminance method, which calculates a weighted sum of the RGB channels, with respective weights of $[0.2989, 0.5870, 0.1140]$ (International Telecommunication Union, 1995). Both page images and masks were then resized to 1024×1024 pixels, to match the model's input layer. As a last step, both the images and their corresponding binary masks were normalized to the $[0, 1]$ range to standardize the input data.

As part of the experimental setup, an optional data augmentation technique was implemented. This augmentation involved random horizontal (left/right) and vertical (up/down) flipping, each applied with a probability of 0.5. The objective of this augmentation was to improve the model's robustness by encouraging it to learn orientation-invariant features, thus allowing accurate segmentation of text lines regardless of the writing direction in the input images.

4.2.3 Connected Component Labeling Post-Processing

The segmentation masks produced by the model effectively separate text line regions from the background but do not distinguish individual text lines as separate instances. To address this limitation, a post-processing procedure must be applied to convert the binary segmentation results into instance-level segmentations.

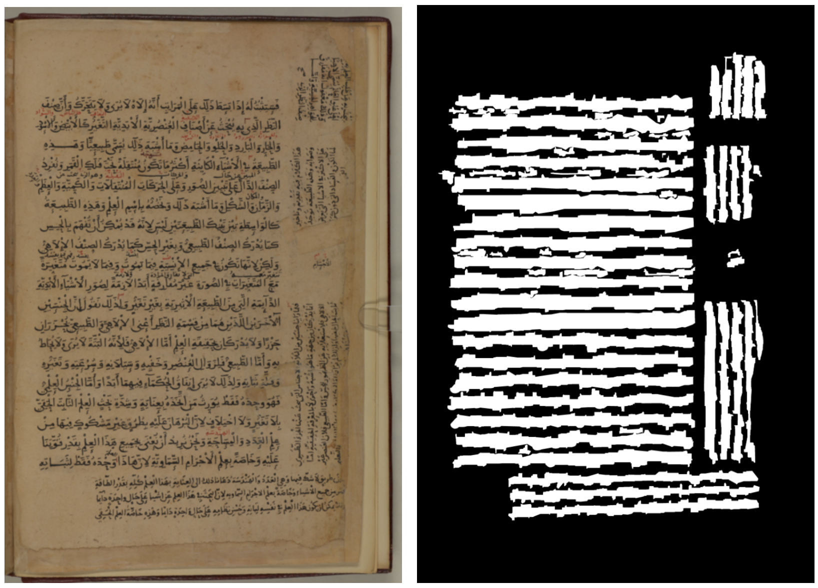

(a) Input image (b) Corresponding mask

Figure 4.5 Sample input pair for U-Net segmentation approach

Connected component labeling is a commonly preferred method in fields such as computer vision and pattern recognition (Abdollahi & Pradhan, 2021; Majanga & Viriri, 2021). The method involves labeling pixels by grouping them based on their connectivity or neighborhood.

The connectivity between two pixels can be defined in two ways (He et al., 2017):

- **4-connected:** Only the horizontal and vertical neighbors of a pixel are considered.
- **8-connected:** The horizontal and vertical neighbors, as well as the diagonal neighbors, are included.

In a binary image, such as the output of the developed U-Net model, background pixels are represented by 0, while foreground pixels - in other words, object pixels - are represented by 1. Connected component labeling, assigns a unique ID to each group of connected foreground pixels and labels them as separate instances (Figure 4.6).

For this process, a connected component labeling algorithm was applied using the maximum connectivity allowed by the image's dimensionality. Specifically, since the input image is two-dimensional, the algorithm applied an 8-connectivity criterion.

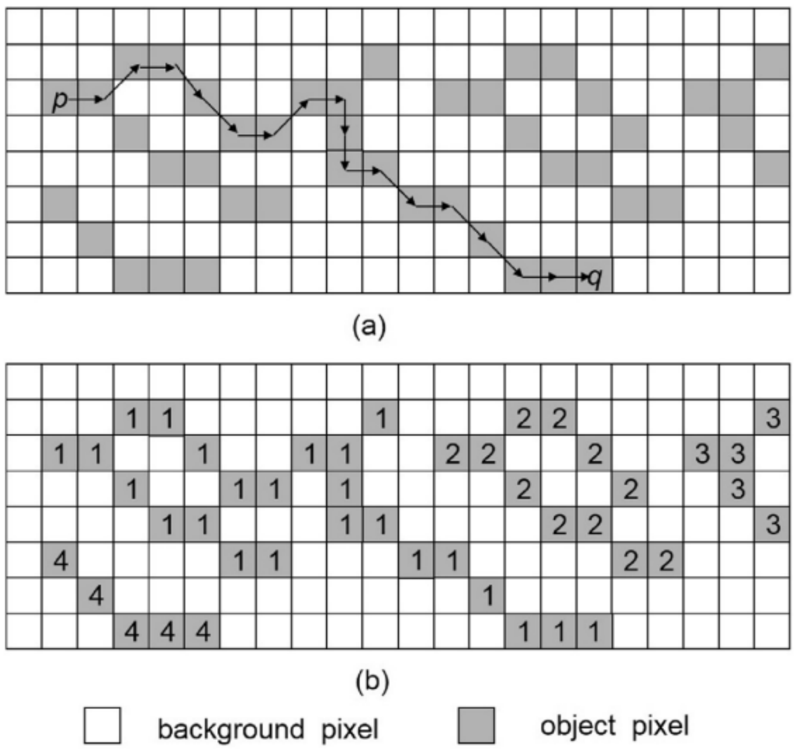


Figure 4.6 Example of 8-connected component labeling. (a) The path illustrating connectivity between pixels *p* and *q*. (b) Resulting labeling of the connected pixels in the given image (a) (He et al., 2017)

4.3 YOLO with Instance Segmentation Approach

The objective of this approach is to perform instance-level segmentation by identifying and separating all text lines directly from the input image in a single pass. This eliminates the need for connected component labeling, which is used as a post-processing step in the U-Net-based approach (Section 4.2). This strategy aims to resolve problems associated with insufficient line separation created by the connected component analysis approach, especially in cases where text lines are closely aligned or overlapped.

The YOLOv11-seg model features a dedicated segmentation head, providing enhanced capabilities compared to the standard YOLO model. This head is responsible for generating both segmentation masks and bounding boxes for detected instances. However, the bounding boxes generated at this stage lack orientation, which may lead to overlaps between them and interfere with the accurate segmentation of text regions.

Training the YOLOv11-seg model requires labels in an American Standard Code for Information Interchange (ASCII)-formatted file, provided alongside the input images. Each of these files includes class IDs and polygon coordinates for the instances present in the corresponding image.

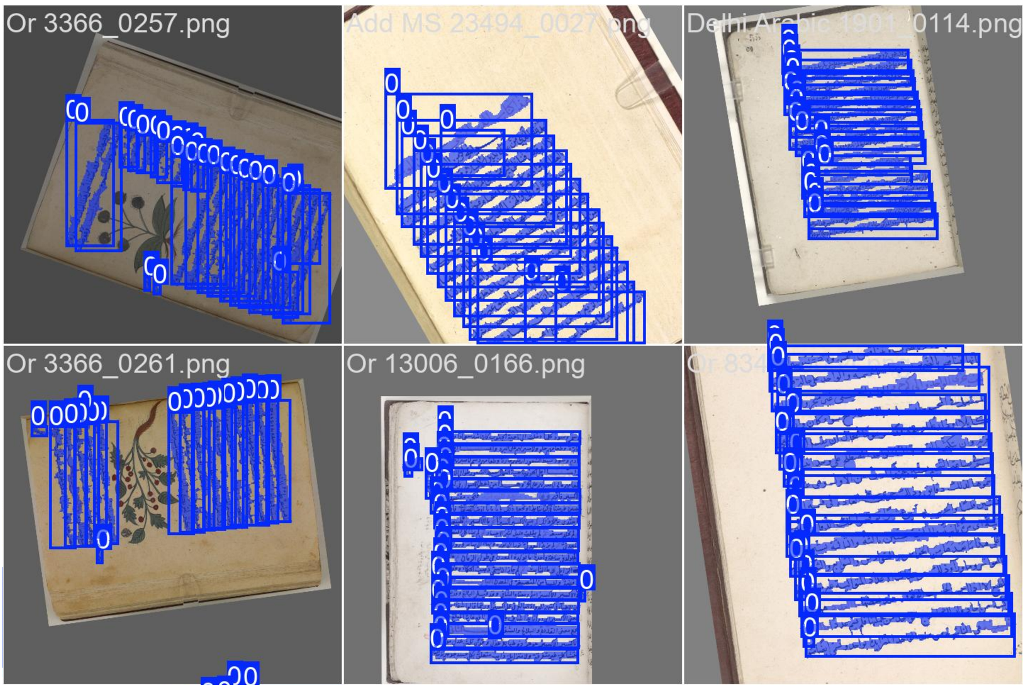


Figure 4.7 Sample images from the train batch of the training for YOLO segmentation approach

4.4 YOLO with OBB & Segmentation Approach

In this approach, a two-stage system is designed to distribute the complexity of the problem across separate models, thereby simplifying the task for each model. In the proposed system, the first stage involves detecting text line areas as oriented bounding boxes. In the second stage, the segmentation model tries to segment the text lines within these detected bounding boxes, or predefined regions.

4.4.1 Data Preparation

To implement the proposed approach, OBBs must first be identified for each text line, and an object detection dataset must be constructed using these labels. Based on this dataset, the bounding boxes are then cropped from each image to create a corresponding segmentation dataset from the resulting image regions.

To create oriented bounding boxes, the dataset described in Section 4.3 was utilized. For each polygon in the dataset, the minimum-area enclosing rectangle was computed from the set of 2D points defining the polygon. This operation yields the smallest rectangle - by area - that fully encloses the specified region of polygon, allowing for orientation alignment with the shape.

An initial dataset was created using the specified bounding box areas (Figure 4.8a). However, it was anticipated that the model might encounter difficulties in distinguishing the boundaries of OBBs, particularly for letters with straight bar structures, since the polygon boundaries in the RASM dataset terminated at the text boundaries (Figure 4.9a). Additionally, it was observed that parts of some characters, especially tails, were omitted in several instances (Figure 4.9b).

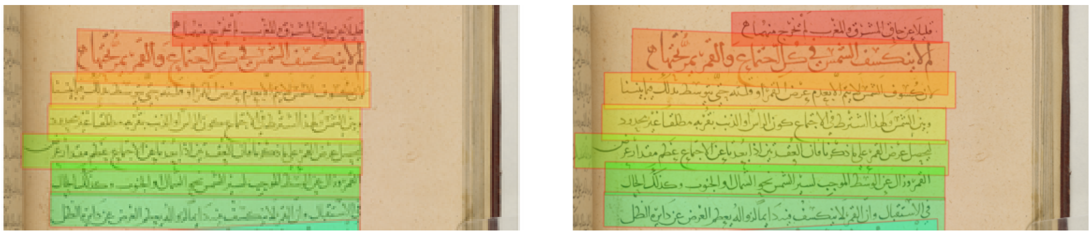
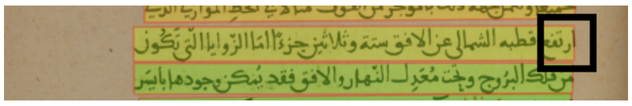
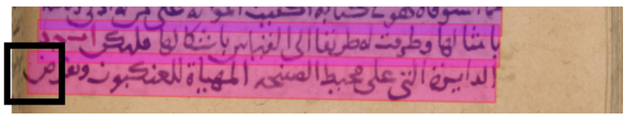

(a) Initial Version (b) Padded Version

Figure 4.8 Sample annotation crops for YOLO OBB model

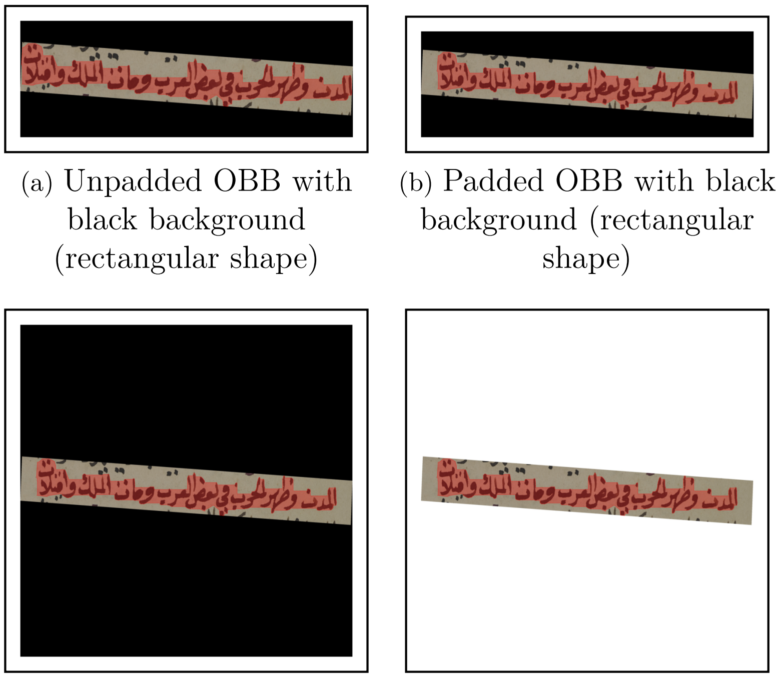


(a) Character positioned at the edge of the OBB boundary



(b) Tail of a character excluded from the OBB

Figure 4.9 Sample annotation issues observed with unpadded OBBs



(a) Unpadded OBB with black background (rectangular shape)

(b) Padded OBB with black background (rectangular shape)

(c) Padded OBB with black background (square shape)

(d) Padded OBB with white background (square shape)

Figure 4.10 Examples of input images prepared in different shapes for the segmentation model. The variations include fill color (black or white) and final image shape (rectangle or square), applied to both padded and unpadded OBB regions. For visibility, edges are highlighted with a black frame.

To mitigate these issues and ensure reliable model detections, an alternative dataset version was created by horizontally expanding the bounding boxes by a specified ratio (Figure 4.8b).

The segmentation dataset was created using the two previously described datasets prepared for OBBs. Initially, the regions defined by the OBBs were cropped. These cropped areas were then padded using both white and black backgrounds to generate square and rectangular image shapes (Figure 4.10). Throughout this process, the orientation of the OBBs was preserved to avoid any loss of information.

4.5 Evaluation Methods

The evaluation of the methods was carried out in two stages: segmentation performance (Section 4.5.1) and the impact of the segmentation on the accuracy of the OCR (Section 4.5.2).

4.5.1 Segmentation Evaluation

Segmentation evaluation of the models is done via a comprehensive metrics calculation framework, which is capable of calculating instance-level and pixel-level metrics.

The evaluator performs the evaluation by comparing ground truth instances with predicted instances. However, to enable this comparison, the corresponding ground truth and predicted instances must first be matched. For this purpose, the Hungarian algorithm (Kuhn, 1955) was employed to maximize the IoU scores. Instance-level and pixel-level metrics are then calculated based on the matched instance pairs.

For instance-level metrics, 50% and 75% IoU thresholds are used to determine correctly matched pairs, as in other studies (Fizaine, Bard, Paindavoine, Robin, Bouyé, Lefèvre & Vinter, 2024). At this stage, the evaluation aligns with object detection metrics rather than pixel-level accuracy. The instance-level metrics include the number of matched text lines, the number of false positive detections, recall, and precision.

In the calculation of pixel-level metrics, binary masks containing only the text pixels of the matched instance pairs are used. The binary masks for the text lines are obtained via Contrast-Limited Adaptive Histogram Equalization (CLAHE) (Pizer, Amburn, Austin, Cromartie, Geselowitz, Greer, ter Haar Romeny, Zimmerman & Zuiderveld, 1987) based contrast enhancement on the lightness channel of the LAB color space to enable effective separation of brightness from color for better text pixel detection. Then, adaptive Gaussian thresholding (Bradley & Roth, 2007) is applied to detect text regions. Morphological operations are used to reduce noise and improve mask quality. As a final step, the resulting mask is refined by intersecting it with ground truth annotations to precisely localize text pixels. This approach aims to eliminate the influence of background regions and potential labeling errors on the pixel-level evaluation. The calculated pixel-level metrics include IoU, precision, recall, and F1-score.

The evaluation metrics based on ground truth G and predictions P are formulated as follows:

$$IoU = \frac{|G \cap P|}{|G \cup P|} = \frac{\sum_{i,j} G_{i,j} \cdot P_{i,j}}{\sum_{i,j} \max(G_{i,j}, P_{i,j})} \tag{4.4}$$

$$Precision = \frac{TP}{TP + FP} = \frac{\sum_{i,j} G_{i,j} \cdot P_{i,j}}{\sum_{i,j} P_{i,j}} \tag{4.5}$$

$$Recall = \frac{TP}{TP + FN} = \frac{\sum_{i,j} G_{i,j} \cdot P_{i,j}}{\sum_{i,j} G_{i,j}} \tag{4.6}$$

$$F1 - Score = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{4.7}$$

Here, True Positive (TP) denotes pixels that are correctly identified as text lines, False Positive (FP) denotes pixels that are incorrectly identified as text lines, and False Negative (FN) denotes pixels that are incorrectly classified as background. The term (i, j) indicates the coordinates of the respective pixel. For instance-level metrics, instance counts are used as the evaluation criterion instead of pixel counts.

In the given formulas, G and P represent binary masks when used in pixel-level metrics, where $G_{i,j}$ and $P_{i,j}$ indicate whether the pixel at location (i, j) belongs to a text region in the ground truth or the prediction, respectively. In contrast, for instance-level metrics, G and P denote the sets of ground truth and predicted text line regions.

4.5.2 OCR Evaluation

For OCR evaluation, the Akis Ottoman transcription system by Akcan, Taşdemir, Kızılırmak, Kuru, Öncel & Yanıkoğlu (2022) was utilized.

In the study, the researchers used 13 different printed Ottoman documents to develop their system. Prior to transcription, they applied a deep learning-based segmentation method proposed by Kodym & Hradiš (2021) to detect text lines. The layout analysis system developed by Kodym & Hradiš (2021) is designed to detect the text line polygons grouped into text blocks within documents. In their study, the authors also introduced the PERO Layout dataset, which comprises printed and handwritten documents in multiple languages, including Arabic and Russian. The proposed architecture for text line detection combines a CNN-based model, ParseNet, with a post-processing stage that estimates text line polygons. In the post-processing stage, polygon estimation is performed using detected baselines along with the model's ascender and descender detection channels. The system achieved a top precision of 91.4% on the cBAD 2019 dataset Diem, Kleber, Sablatnig & Gatos (2019) using adaptive scaling and line merging enhancements.

To conduct the OCR test, the developed segmentation methods were used to extract text lines from all transcribed pages of the Asir Efendi dataset. Each segmented line was masked and padded into a rectangular shape. The resulting images were saved along with the page number and the left/right page position. These line images were then processed using the OCR model developed for Akis, and the predictions were stored in a dataframe along with the corresponding page and position information.

To match the OCR outputs with their corresponding reference lines, a matching algorithm was employed. This algorithm operates on a page-by-page basis, comparing each OCR-generated line with its respective reference using Levenshtein distance.

The Levenshtein distance is a widely used metric that calculates the difference between two strings Zhang, Hu & Bian (2017). It is defined as the minimum number of changes on the character level to transform one string into the other. The algorithm generates a matrix for each pair of strings, wherein the value in the bottom-right cell represents the final distance between the two strings. This matrix is constructed based on Equation (4.8), where i and j represent the positions of the first and the second strings, respectively.

$$LD(i,j) = \begin{cases} 0, & i = 0,\ j = 0 \\ j, & i = 0,\ j > 0 \\ i, & i > 0,\ j = 0 \\ min, & i > 0,\ j > 0 \end{cases} \tag{4.8}$$

A total of 199 prediction–reference pairs, identified based on Levenshtein distance, were randomly selected to calculate CER and WER scores as the OCR evaluation metrics.

CER and WER are evaluation metrics based on Levenshtein distance and are used to measure character- and word-level errors made by a model. Both metrics are computed using the same formula, as shown in Equation (4.9); the distinction lies in that CER operates at the character level, while WER is calculated over words.

$$WER\ or\ CER = \frac{S + D + I}{N} \tag{4.9}$$

Here, S represents substitution, D represents deletion, I represents insertion, and N represents the number of words (for WER) or characters (for CER) in the reference.

5. EXPERIMENTS AND RESULTS

This section provides a comparative analysis of the results achieved by the proposed methods. Throughout the experiments, models in all methods were trained until they achieved optimal convergence on their respective training and validation sets. Since each method expects different input formats, their training behavior and convergence were assessed separately, based on their respective datasets. To measure how well these models perform on Ottoman dataset - which is outside of their training domain - all of them were tested on the Asir Efendi dataset. All methods were tested on the Asir Efendi dataset to assess their performance on the Ottoman dataset.

In the remainder of this section, the results of each approach will be discussed in detail (Sections 5.1-5.3). Since the methods were developed sequentially - each building on the challenges identified in the previous one - the results are presented in the order in which they were obtained. After examining each method individually, the best outcomes from all approaches are compared (Section 5.4).

The data generated using the methods described in Section 4.2.2 was divided into 90% for training and 10% for validation during the training of the models.

To prevent overfitting, all models were trained using an early stopping strategy, which stopped the training if no improvement was observed on the validation set for 15 consecutive epochs. The model weights from the epoch with the best validation performance were then restored for the final evaluation.

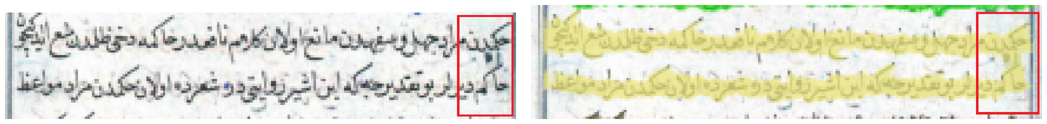
Except for the focal loss model, all U-Net approaches generally achieved high performance in binary segmentation. In the case of the focal loss model, although both training and validation losses were lower than those of the other models, the corresponding accuracy values were also comparatively lower. One issue across all U-Net approaches was the inability to separate individual lines, which was primarily due to the limitations of the connected component post-processing applied after binary segmentation. As shown in Table 5.1, the best performance among the models was achieved using the combined loss function, which resulted in a training accuracy of 95.49% and a validation accuracy of 95.69%. The boundary-weighted BCE loss function followed closely, with a training accuracy of 95.48%, and a validation accuracy of 95.28%.

Table 5.1 Train Results for U-Net Approaches

Loss Type	**Epochs**	**Train Loss**	**Train Accuracy**	**Val Loss**	**Val Accuracy**
Combined Loss	1193	0.1256	**0.9549**	0.1199	**0.9569**
Focal Loss	55	**0.0528**	0.8555	**0.0471**	0.8666
Foreground Weighted BCE	500	0.1839	0.9435	0.1686	0.9473
Boundary Weighted BCE	1264	0.1629	0.9548	0.1707	0.9528

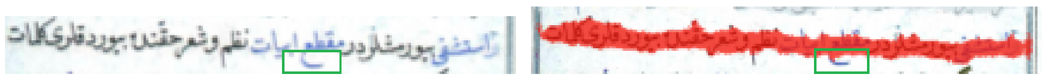
The examination of the outputs of the model trained with the combined loss function showed that the model often treated closely spaced or nested text lines as adjacent. As a result, these lines were merged and interpreted as a single instance during the connected component post-processing stage (Figure 5.1). Furthermore, the model struggled to detect the tails of long-tailed characters in some lines as the foreground

(Figure 5.2). These observations indicated that simply distinguishing between foreground (text lines) and background was not sufficient for accurate training. To address this, focal loss was introduced to help the model pay greater attention to harder examples, specifically the boundaries between text lines.



(a) Original image (b) Segmented image

Figure 5.1 Example of nested text lines that are merged by the U-Net approach with combined loss

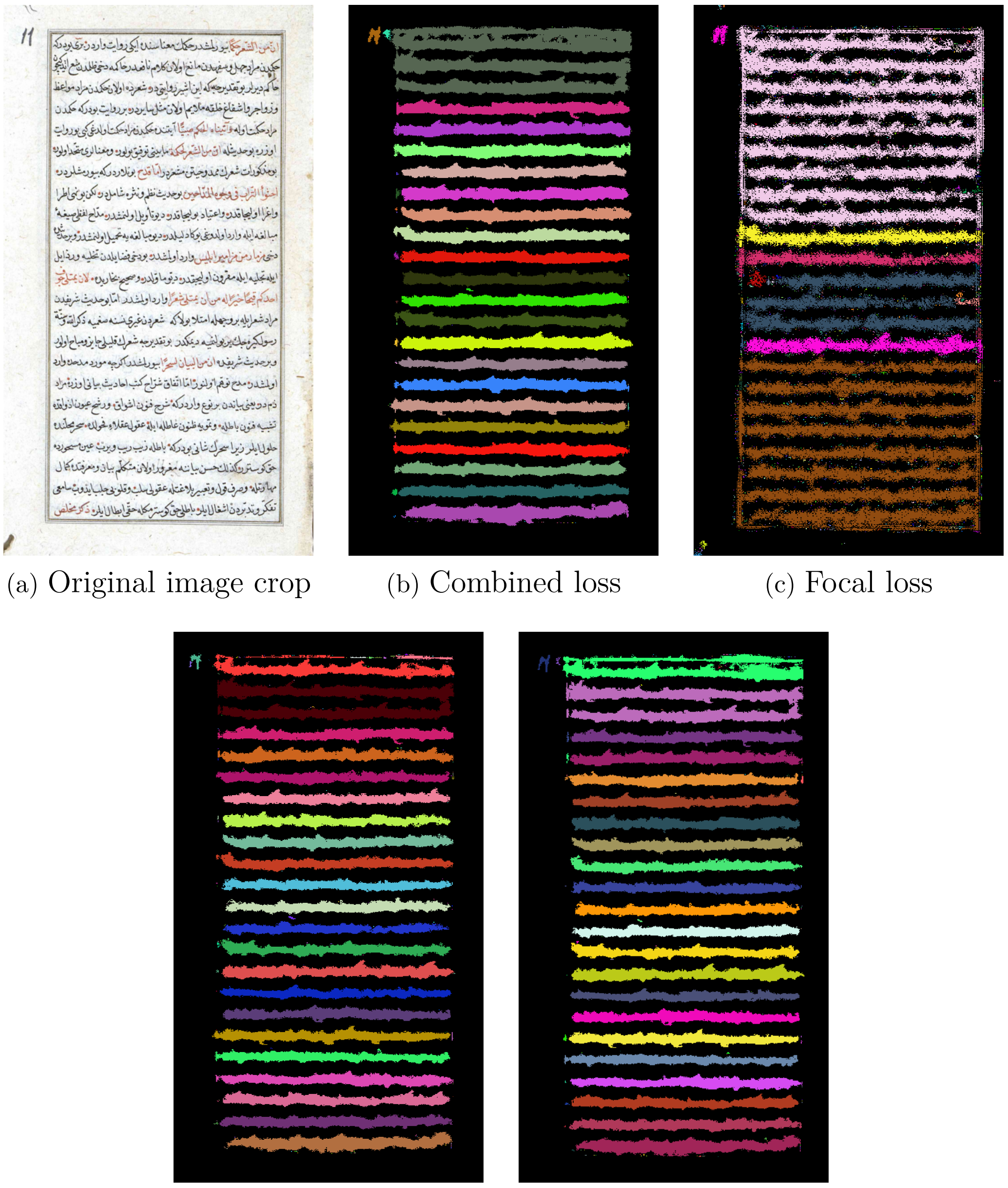


(a) Original image (b) Segmented image

Figure 5.2 Example of a segmented text line where the U-Net approach with combined loss misses the tail of a character

The implementation of focal loss led to the generation of noisy mask outputs. This noise introduced challenges during the connected component post-processing stage, particularly in separating individual text lines. As a result, the output showed that multiple lines were often merged and detected as a single instance (Figure 5.3c). The outputs obtained using focal loss supported the assumption that it might be more effective to assign greater weight directly to line boundaries or foreground regions, rather than focusing on model errors or harder examples. Based on this insight, two alternative loss functions were implemented: a foreground-weighted binary cross-entropy (BCE) loss and a boundary-weighted BCE loss. However when foreground-weighted BCE and boundary-weighted BCE loss functions were applied, approaches still struggled to separate rows during the connected component post-processing stage.

Table 5.2 presents the performance of the U-Net-based approaches on the angled lines subset of the Asir Efendi dataset, evaluated at an IoU threshold of 75%. The focal loss variant was excluded from the comparison due to the model generating an excessive number of output instances - an outcome linked to the high level of noise in the model's predictions. Among the tested approaches, the boundary-weighted BCE approach achieved the highest Intersection over Union (IoU) score at 51.8%. However, this result indicates that only about half of the predicted regions overlapped with the actual text areas, highlighting a limitation in segmentation accuracy. These results supported the assumption that relying solely on connected component analysis after binary segmentation was insufficient to separate text lines accurately. As a result, the study shifted from binary segmentation to instance segmentation approaches (Section 4.3).



(a) Original image crop (b) Combined loss (c) Focal loss

(d) Foreground-weighted BCE (e) Boundary-weighted BCE

Figure 5.3 Original image and predicted masks for straight lines by U-Net binary approach with different loss functions

Table 5.2 Test results @ IoU 0.75 for the U-Net approaches on the Asir Efendi subset of angled lines

	Pixel-Level Metrics				**Instance-Level Metrics**					
Loss Function	**IoU**	**Precision**	**Recall**	**F1-Score**	**Total GT**	**Matched Text Lines**	**Predicted Text Lines**	**False Positives**	**Instance Recall**	**Instance Precision**
Combined Loss	0.36	0.38	0.36	0.37	794	298	5279	**4981**	0.375	0.056
Boundary Weighted BCE	**0.518**	**0.565**	**0.518**	**0.54**		**446**	**6751**	6305	**0.562**	**0.066**
Foreground Weighted BCE	0.446	0.488	0.446	0.466		386	8120	7734	0.486	0.048

5.2 Results for the YOLO Instance Segmentation Approach

In the YOLO instance segmentation approach, a total of 11 model variations were trained and evaluated. These variations include differences in augmentation techniques and pre-trained models. Detailed configurations of each model are presented in Table 5.3.

The details of the augmentation parameters given in the table can be explained as follows:

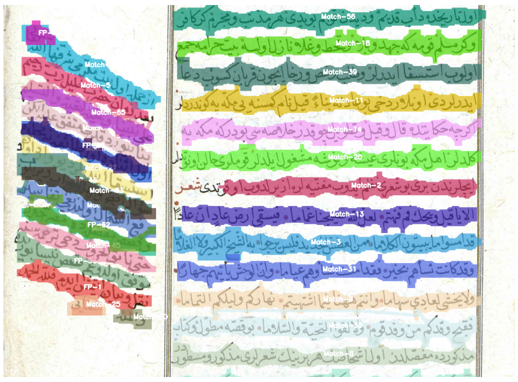
- **mosaic:** Combines four training images into one, with a specified probability value between 0 and 1.
- **close mosaic:** Disables mosaic augmentation for a specified number of epochs towards the end of the training process.
- **overlap mask:** Specifies whether object masks are merged into a single composite mask or kept separate. When overlaps occur, the smaller mask is placed on top of the larger one.
- **mask ratio:** Specifies the downsampling ratio applied to segmentation masks, which determines the resolution of the masks used during training.
- **degrees:** Rotates the image randomly at a given value between 0-180 degrees.
- **translate:** Shifts the image horizontally and vertically, using a magnitude value between 0 and 1. The displacement can be either positive or negative.
- **fliplr:** Performs horizontal flip by reflecting the image along the x-axis with a specified probability value between 0 and 1.

The data generated using the methods described in Section 4.4.1 was divided into 90% for training and 10% for validation during the training of the models. To ensure consistency in experimental conditions, the *random_state* parameter used during the U-Net data split was preserved. This ensured that models were evaluated under the same conditions, allowing a fair comparison of their performance.

Model number 9 yielded the highest performance in this approach. On the Asir Efendi dataset, it was notably effective in segmenting straight text lines. However, the model had difficulty with slanted lines, often missing them or segmenting them incompletely (Figure 5.4).

Table 5.3 Configurations for the Model Variations of YOLO Instance Segmentation Approach

ID	**Pretrained Model**	**epochs**	**Batch**	**imgsz**	**mosaic**	**close mosaic**	**overlap mask**	**mask ratio**	**degrees**	**translate**	**fliplr**
1	yolo11s seg	100	32	640	1	50	FALSE	4	0	0	0
2	yolo11m seg	100	16	1024	1	0	FALSE	4	0	0	0
3	yolo11m seg	100	16	1024	1	0	FALSE	4	0	0	0
4	model-3	100	16	1024	1	10	TRUE	4	0	0.1	0.5
5	model-3	200	16	1024	1	0	FALSE	4	0	0.1	0
6	model-3	200	16	1024	1	0	FALSE	4	0	0	0
7	yolo11m seg	200	16	1024	1	0	FALSE	4	45	0.2	0
8	yolo11m seg	200	16	1024	1	0	FALSE	4	45	0.2	0
9	yolo11m seg	200	16	1024	0	0	TRUE	1	90	0.2	0
10	yolo11m seg	200	16	1024	0	0	FALSE	4	90	0.1	0
11	model-10	200	16	1024	0	0	FALSE	4	90	0.1	0


(a)

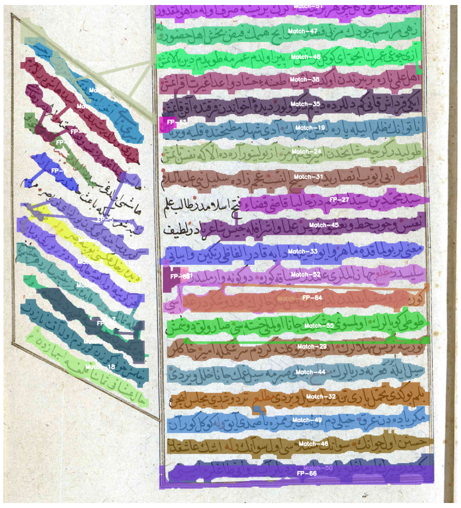

(b)

Figure 5.4 Sample output crops produced by the best-performing model (Model-9) of the YOLO segmentation approach

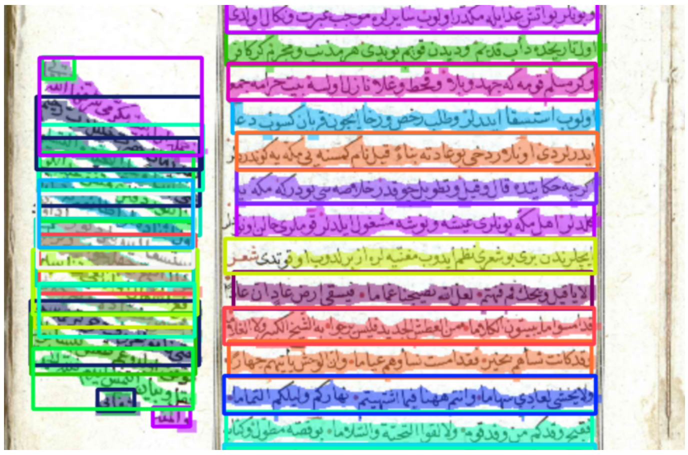


Figure 5.5 Sample output crop from the YOLO segmentation approach, illustrating how bounding boxes were used to localize text lines

A detailed examination of the model outputs revealed that the bounding boxes generated by the model during segmentation were consistently horizontally aligned, regardless of the slope of the text lines (Figure 5.5). This limitation caused the model to produce multiple overlapping bounding boxes as the angle of the lines increased, making it difficult to distinguish individual text lines within a given region accurately. To address this issue, a two-step system - which first detects lines using oriented bounding boxes, then applies segmentation - was investigated (Section 5.3).

Table 5.4 Test results for the YOLO segmentation approach experiments @ IoU 75% on the Asir Efendi subset of angled lines

	Pixel-level Metrics				**Instance-level Metrics**					
Model ID	**IoU**	**Precision**	**Recall**	**F1-score**	**Total GT**	**Matched Text Lines**	**Predicted Text Lines**	**False Positive**	**Instance Recall**	**Instance Precision**
1	0.809	0.896	0.818	0.855	794	947	714	233	0.899	0.754
2	0.821	0.911	0.824	0.865		909	722	187	0.909	0.794
3	0.803	0.905	0.805	0.852		978	716	262	0.902	0.732
4	0.715	0.835	0.717	0.771		1072	660	412	0.831	0.616
5	0.776	0.906	0.777	0.836		1102	716	386	0.902	0.650
6	0.777	0.863	0.779	0.818		829	680	149	0.856	0.820
7	0.743	0.884	0.745	0.808		822	703	119	0.885	0.855
8	0.878	0.962	0.880	0.919		922	**765**	157	**0.963**	0.830
9	**0.890**	0.956	**0.895**	**0.924**		937	762	175	0.960	0.813
10	0.812	0.934	0.816	0.870		847	743	104	0.936	0.877
11	0.847	**0.958**	0.849	0.900		841	761	**80**	0.958	**0.905**

5.3 Results for the YOLO OBB & Segmentation Approach

Since the YOLO OBB & Segmentation approach is composed of two distinct stages, its evaluation is presented in two parts (Sections 5.3.1-5.3.2), with the overall performance of the full pipeline discussed separately (Section 5.3.3).

The data generated using the methods described in Section 4.4.1 was divided into 90% for training and 10% for validation during the training of both the OBB and segmentation models. At this stage, the OBB dataset was first split according to the specified ratio. Then, the segmentation dataset was constructed using the pages allocated for training in the OBB split, ensuring that the segmentation model used the same train-validation division as the OBB model. To maintain consistency across experiments, the same train-validation split used in the YOLO segmentation (Section 5.2) approach was applied.

5.3.1 OBB Model Experiments

The performance of the OBB models developed within the YOLO pipeline was evaluated based on both their training performances and their impacts on the subsequent segmentation model. A total of 16 model variations were tested, each differing in datasets, augmentation techniques, and the choice of pre-trained models. The detailed configuration of each model is presented in Table 5.5.

The models were developed using the two versions of the data set, as described in Section 4.4.1. For clarity, these versions will be referred to in this section as the *initial* and the *padded* versions.

Among the augmentation parameters listed in the table, the *translate* and *degrees* parameters are consistent with those described in Section 4.3. The remaining parameters are explained below:

- **scale:** Resizes the image by a factor of 1 plus or minus the specified value.

The models were evaluated using their validation performance, as shown in Table 5.6. In the table, *(B)* represents the performance of the model on object detection task. The *mAP50* and *mAP50-95* expressions given in the table represent the mean average precision value obtained at the 50% IoU threshold and at values varying between 50% - 95% IoU thresholds, respectively. The *fitness* parameter is a weighted combination of the metrics. The weights for the metrics *precision*, *recall*, *mAP50* and *mAP50-95* are; 0.0, 0.0, 0.1, 0.9, respectively.

The best performing models according to dataset versions were model-2 with 99.2% precision score for the initial dataset and model-13 with 99.9% precision score for the padded dataset. Model-13 was selected to be used as the OBB model in the final pipeline due to the higher performance obtained with the *padded* data set in the overall.

Table 5.5 Configurations for OBB Model Variations of YOLO OBB & Segmentation Approach

ID	**Data Version**	**Pretrained Model**	**epochs**	**translate**	**scale**	**degrees**
1	initial	yolo11m obb	200	0.1	0.5	90
2		yolo11s obb	200	0	0.1	90
3		yolo11l obb	200	0	0.1	90
4	padded	yolo11l obb	200	0	0.1	90
5		yolo11m obb	200	0.1	0.5	90
6		model-5	50	0.1	0.5	0
7		model-5	50	0.1	0.5	90
8		model-5	20	0.1	0.5	0
9		model-5	100	0.1	0.5	0
10		yolo11m obb	100	0.1	0.5	0
11		yolo11m obb	100	0.1	0.5	0
12		yolo11m obb	200	0.1	0.5	0
13		model-12	200	0.1	0.5	0
14		yolo11x obb	50	0.1	0.5	90
15		model-14	50	0.1	0.5	90
16		model-15	50	0.1	0.5	0

Table 5.6 Validation Results for OBB Models of the YOLO OBB & Segmentation Approach

ID	Data Version	precision (B)	recall (B)	mAP50 (B)	mAP50-95 (B)	fitness
1	initial	0.981	0.987	0.986	0.796	0.815
2		0.992	0.973	0.987	0.790	0.810
3		0.974	0.984	0.985	0.813	0.830
4	padded	0.963	0.981	0.977	0.796	0.814
5		0.981	0.981	0.990	0.782	0.803
6		0.981	0.995	0.992	0.825	0.842
7		0.982	0.977	0.992	0.820	0.837
8		0.981	0.995	0.992	0.825	0.842
9		0.989	0.988	0.989	0.836	0.851
10		0.979	0.981	0.993	0.793	0.813
11		0.979	0.981	0.993	0.793	0.813
12		0.992	0.976	0.994	0.819	0.837
13		**0.999**	**1.000**	**0.995**	**0.915**	**0.923**
14		0.966	0.925	0.976	0.694	0.723
15		0.977	0.963	0.987	0.733	0.759
16		0.989	0.981	0.992	0.817	0.835

5.3.2 Segmentation Model Experiments

For the segmentation part of the pipeline, four different models trained and evaluated. The models were developed using the three of the four dataset versions described in Section 4.4.1. Detailed configurations of each model are presented in Table 5.7. Augmentation parameters listed in the table, are consistent with those described in Sections 4.3 and 5.3.1.

Table 5.7 Configurations for Segmentation Model Variations of YOLO OBB & Segmentation Approach

ID	Data Version	Pretrained Model	epochs	batch	degrees
1	padded-obb, black rectangle background	yolo11m seg	100	16	90
2	padded-obb, white square background	yolo11m seg	100	16	0
3	padded-obb, black square background	yolo11m seg	100	16	0
4		yolo11l seg	200	32	0

The models were evaluated using their validation performance, as shown in Table 5.8. In the table, while *(B)* represents the performance of the model on object detection task like on the OBB model (Section 5.3.1); *(M)* represents the performance of the model on segmentation task. The *fitness* parameter is a summation of the weighted combination of the metrics for object detection and segmentation tasks. The weights are the same used in the OBB model, which are 0.0, 0.0, 0.1, 0.9; for the metrics *precision*, *recall*, *mAP50* and *mAP50-95*, respectively.

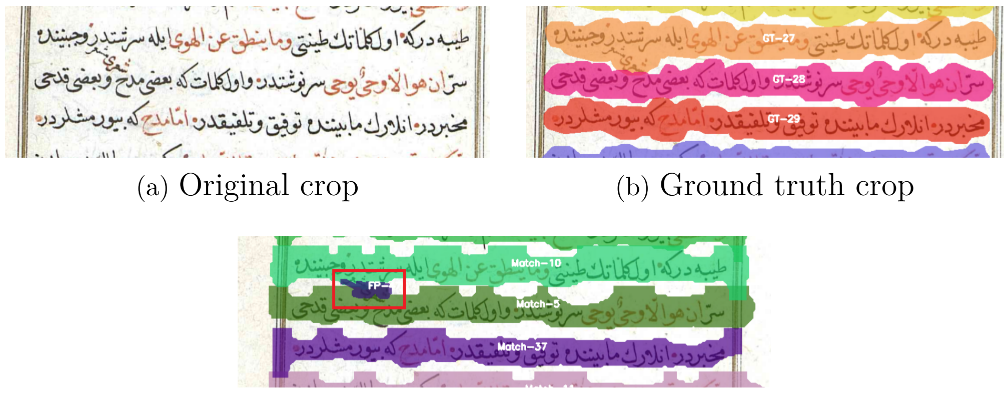
The best performing model was model-4 with precision scores of 1 for detection task and 99.6% for segmentation task. The best-performing model was also evaluated for OCR performance using the procedure described in Section 4.5.2. On the subset containing 199 lines, the method achieved a CER of 13.9% and a WER of 36.6%.

Table 5.8 Validation Results for Segmentation Models of the YOLO OBB & Segmentation Approach

ID	**Data Version**	**(B)**				**(M)**				**fitness**
		precision	**recall**	**mAP 50**	**mAP 50-95**	**precision**	**recall**	**mAP 50**	**mAP 50-95**	
1	padded-obb, black rectangle background	0.983	0.996	0.985	0.898	0.979	0.992	0.981	0.700	1.634
2	padded-obb, white square background	0.983	1.000	0.994	0.935	0.979	0.996	0.994	0.707	1.677
3	padded-obb, black square background	0.989	0.984	0.994	0.915	0.989	0.984	0.994	0.675	1.629
4	padded-obb, black square background	**1.000**	**1.000**	**0.995**	**0.960**	**0.996**	**0.996**	**0.994**	**0.708**	**1.701**

5.3.3 Pipeline Experiments

To create the final pipeline, the best performing models in OBB and segmentation experiments - model-13 for OBB and model-4 for segmentation - were used. The final pipeline was evaluated on the Asır Efendi dataset, and the results were recorded for comparison with the other methods. The approach achieved a precision score of 99.3% at the 75% IoU threshold on the subset containing only straight lines. On the mixed dataset, which included both angled and straight lines, it achieved a precision score of 95%. The pipeline was also evaluated for OCR performance using the procedure described in Section 4.5.2. On the subset containing 199 lines, the method achieved a CER of 13.1% and a WER of 34.8%.



(a) Original crop

(b) Ground truth crop

(c) Prediction crop

Figure 5.6 Output sample with a note between the lines (highlighted with red rectangle) produced by the YOLO OBB & Segmentation Approach

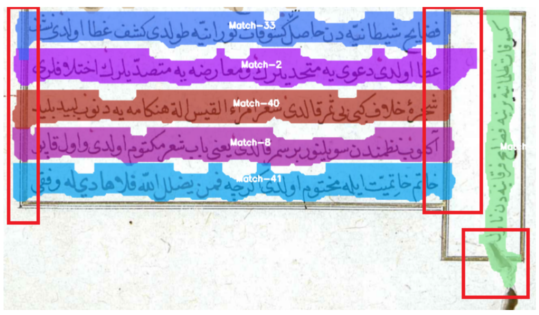


Figure 5.7 Output sample with noises (highlighted with red rectangles) in the polygon boundaries produced by the YOLO OBB & Segmentation Approach

The relatively small difference in performance between the 50% and 75% IoU thresholds suggests that the model is not highly sensitive to variations in the IoU value.

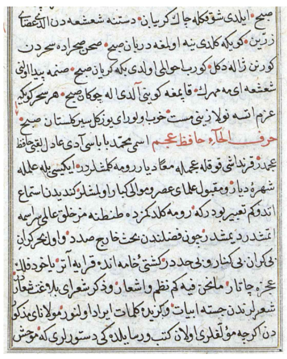
A common issue observed in the outputs is that the approach sometimes detects notes written between lines as separate text lines (Figure 5.6). This likely results from inconsistencies in the training data, where such notes have been labeled separately. In the test dataset, however, these annotations are typically merged with the corresponding line and labeled as a single instance. As a result, these predictions of the approach are considered false positives, which negatively affects performance. However, this inconsistency can be mitigated through refined dataset preparation and a consistent annotation practices between the train and test data.

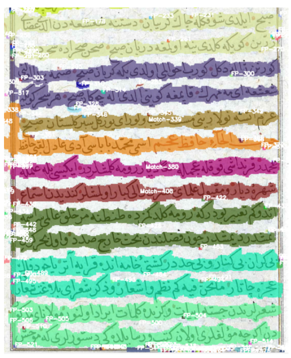
Another issue arises from the page boundary lines and the wear visible on the documents caused by the age of the documents. These elements introduce visual noise, leading the segmentation model to include these regions within the text line polygon boundaries (Figure 5.7).

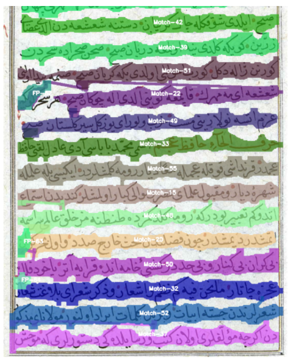
5.4 Comparative Results

An analysis of the performance of the methods at the thresholds 50% and 75% IoU on subsets that contain angled (Tables 5.11 -5.13) and straight lines (Tables 5.10-5.12) reveals that the YOLO OBB & Segmentation approach outperforms the others.

In the evaluation performed on straight text lines, different behaviors were observed between the approaches (Figure 5.8). The U-Net approach frequently grouped several text lines in a single instance, while the YOLO segmentation model produced multiple detections within a single text line. The YOLO OBB & Segmentation approach was able to detect all text lines as distinct instances, although it tended to generate polygon regions that exceeded the actual boundaries of the text lines.


(a) Original Crop


(b) U-Net Combined Loss


(c) YOLO Segmentation

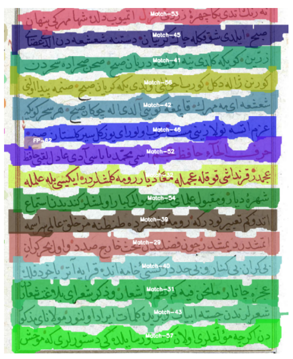
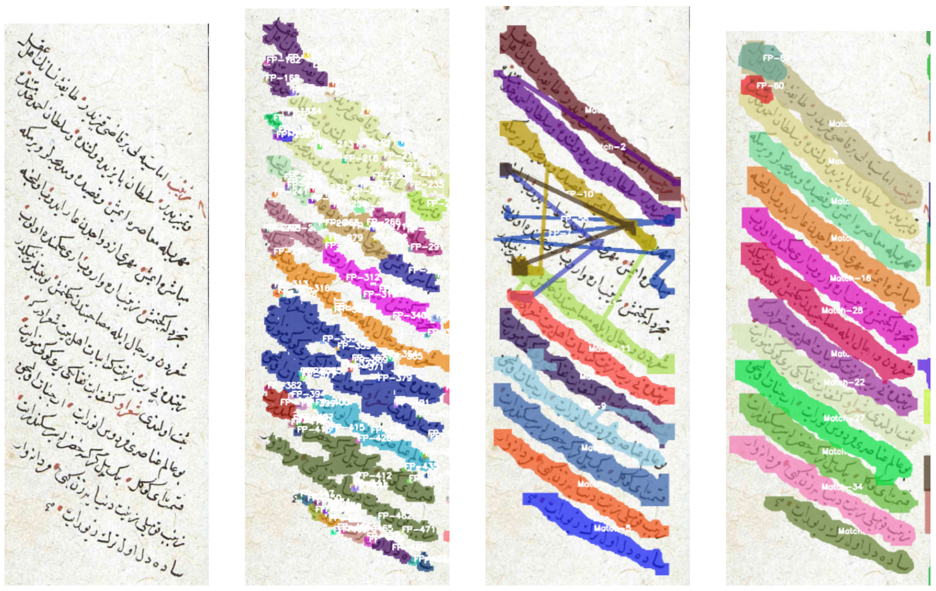

(d) YOLO OBB & Segmentation

Figure 5.8 Sample outputs for the straight lines produced by best-performing approaches

In general, all methods demonstrated reduced performance on the subset with angled lines. This difference in performance is likely due to the imbalance in the training set, which consisted mainly of straight lines, even though augmentation was applied.

In a test conducted on angled text lines, the U-Net approach was able to generate masks that covered the overall text regions but failed to separate individual lines, which is a similar behaviour it showed on straight lines. The YOLO segmentation method, while unable to detect all lines, produced a fragmented output by breaking the detected lines into multiple parts. However, it still correctly identified and separated more lines than the U-Net model. The YOLO OBB & Segmentation approach, despite detecting small segments at the beginning of some of the text lines, demonstrated superior performance by correctly identifying all of the text lines. However, it also introduced some false-positive detections.



(a) Original Crop (b) U-Net Combined Loss (c) YOLO Segmentation (d) YOLO OBB & Segmentation

Figure 5.9 Sample outputs for the angled lines produced by best-performing approaches

An examination of the OCR results presented in Table 5.9 reveals that the YOLO OBB & Segmentation approach, which adopts a multi-stage strategy, outperforms the YOLO segmentation method.

Table 5.9 CER and WER Comparison of YOLO-Based Approaches and Manual Cropping

Method	**CER**	**WER**
YOLO Segmentation	0.139	0.365
YOLO OBB & Segmentation	0.131	0.347
Manual Crop	**0.109**	**0.309**

Table 5.10 Comparative Results for Straight Lines @ IoU 0.5

Approach		**Pixel-Level Metrics**				**Instance-Level Metrics**					
Model	**Loss Type**	**IoU**	**Precision**	**Recall**	**F1-Score**	**Total GT**	**Matched Text Lines**	**Predicted Text Lines**	**False Positive**	**Instance Recall**	**Instance Precision**
YOLO Instance Segmentation	Standard	0.839	0.968	0.859	0.91	1152	1146	1208	**62**	0.995	**0.949**
YOLO OBB & Segmentation	Standard	**0.923**	**0.993**	**0.925**	**0.958**		**1148**	**1214**	66	**0.997**	0.946
U-Net Binary Segmentation	Combined	0.556	0.586	0.589	0.585		716	7388	6672	0.622	0.097
	Foreground Weighted BCE	0.641	0.7	0.66	0.678		833	11030	10197	0.723	0.076
	Boundary Weighted BCE	0.758	0.826	0.779	0.8		980	9708	8728	0.851	0.1

Table 5.11 Comparative Results for Straight & Angled Lines @ IoU 0.5

Approach		**Pixel-Level Metrics**				**Instance-Level Metrics**					
Model	**Loss Type**	**IoU**	**Precision**	**Recall**	**F1-Score**	**Total GT**	**Matched Text Lines**	**Predicted Text Lines**	**False Positive**	**Instance Recall**	**Instance Precision**
YOLO Instance Segmentation	Standard	0.752	0.863	0.774	0.815	794	703	785	82	0.885	0.896
YOLO OBB & Segmentation	Standard	**0.886**	**0.955**	**0.892**	**0.922**		**763**	**843**	**80**	**0.961**	**0.905**
U-Net Binary Segmentation	Combined	0.35	0.37	0.371	0.369		308	5288	4980	0.388	0.058
	Foreground Weighted BCE	0.45	0.495	0.464	0.478		405	8122	7717	0.51	0.05
	Boundary Weighted BCE	0.521	0.569	0.539	0.552		466	6746	6280	0.587	0.069

Table 5.12 Comparative Results for Straight Lines @ IoU 0.75

Approach		**Pixel-Level Metrics**				**Instance-Level Metrics**					
Model	**Loss Type**	**IoU**	**Precision**	**Recall**	**F1-Score**	**Total GT**	**Matched Text Lines**	**Predicted Text Lines**	**False Positive**	**Instance Recall**	**Instance Precision**
YOLO Instance Segmentation	Standard	0.853	0.987	0.856	0.916	1152	1140	1224	84	0.99	0.931
YOLO OBB & Segmentation	Standard	**0.923**	**0.993**	**0.926**	**0.958**		**1148**	**1214**	**66**	**0.997**	**0.947**
U-Net Binary Segmentation	Combined	0.564	0.596	0.564	0.58		686	7386	6700	0.595	0.093
	Foreground Weighted BCE	0.639	0.7	0.639	0.668		806	11028	10222	0.7	0.073
	Boundary Weighted BCE	0.764	0.834	0.764	0.797		961	9709	8748	0.834	0.099

Table 5.13 Comparative Results for Straight & Angled Lines @ IoU 0.75

Approach		**Pixel-Level Metrics**				**Instance-Level Metrics**					
Model	**Loss Type**	**IoU**	**Precision**	**Recall**	**F1-Score**	**Total GT**	**Matched Text Lines**	**Predicted Text Lines**	**False Positive**	**Instance Recall**	**Instance Precision**
YOLO Instance Segmentation	Standard	0.812	0.934	0.816	0.87	794	743	847	104	0.936	0.877
YOLO OBB & Segmentation	Standard	**0.883**	**0.95**	**0.888**	**0.918**		**758**	**843**	**85**	**0.955**	**0.899**
U-Net Binary Segmentation	Combined	0.36	0.38	0.36	0.37		298	5279	4981	0.375	0.056
	Foreground Weighted BCE	0.446	0.488	0.446	0.466		386	8120	7734	0.486	0.048
	Boundary Weighted BCE	0.518	0.565	0.518	0.54		446	6751	6305	0.562	0.066

5.5 Auto Labeling and Dataset Generation

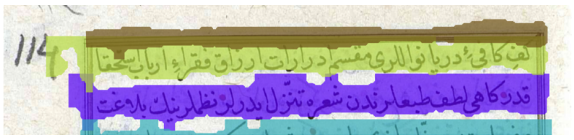
In addition to the model contributions of the study, an Ottoman text line segmentation dataset generated using the developed methods. Utilizing the YOLO OBB & Segmentation approach, which demonstrated the best overall performance. The resulting dataset is publicly shared and available on GitHub [1].

The resulting dataset was saved in YOLO format. In this format, each image has a corresponding ASCII-formatted file with the same name. Within each label file, a separate line is written for every instance in the image. Each line contains the class ID followed by the x and y coordinates of the polygon points representing the instance. The format of each line is structured as follows:

- *class-id x1 y1 x2 y2 ... xn yn*

During the dataset creation, a 50% IoU threshold and a 90% overlap threshold were used to reduce noise by eliminating duplicate detections of the same text line area. These detected regions can be further filtered based on size and count, or segmentation errors can be mitigated by merging closely aligned lines.

A subset of 20 images was randomly selected to evaluate the instance segmentation performance of the resulting dataset. Out of 1086 lines, the model successfully segmented 1066, meaning the predicted masks visually enclose the text line. However, the model missed 20 lines, typically in visually complex regions such as multi-column formatted layouts (Figure 5.11). Additionally, 39 false positives were observed, where non-text or non-Arabic elements, such as borders or page numbers presented in the page, were incorrectly identified as text lines (Figure 5.10). Despite those limitations, the dataset maintains high performance with a precision score of 96.5% and a recall of 98.1%.


(a) Border identified as a text line

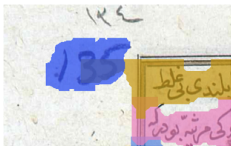

(b) Page number identified as a text line

Figure 5.10 Sample crops from the resulting dataset with false positive detections

[1] https://github.com/seccily/ottoman-text-line-segmentation-dataset

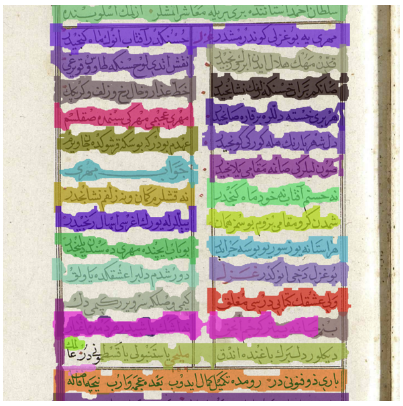
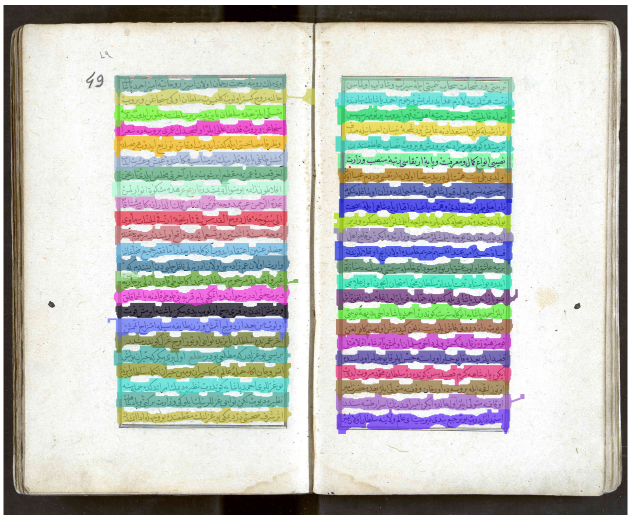


Figure 5.11 A sample crop from the resulting dataset with multi-column formatting


(a) Sample with straight lines

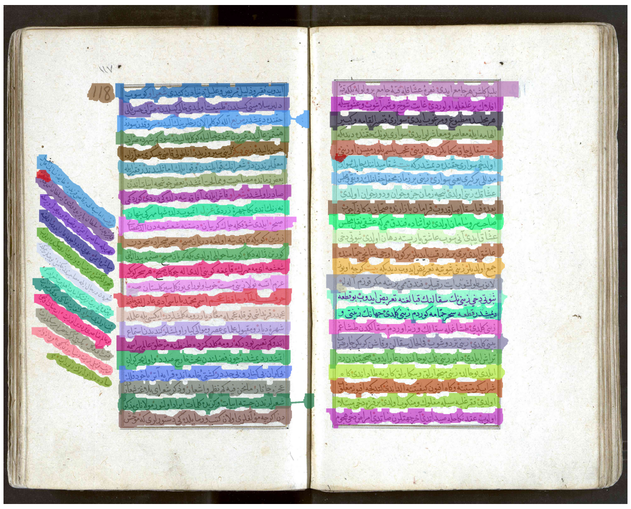

(b) Sample with angled lines

Figure 5.12 Samples from the resulting dataset

6. CONCLUSION AND FUTURE WORK

In this work, different deep learning-based approaches are evaluated to solve the line segmentation problem in Ottoman manuscripts. Alongside typical challenges - such as line overlap and inconsistent handwriting - Ottoman/Arabic scripts bring additional complexity, including long-tailed characters and intra-word character connections. To address these challenges, three methods were developed based on the U-Net and YOLO architectures.

The method developed using the U-Net architecture relies on binary segmentation followed by connected component post-processing. This approach was evaluated using five different loss functions. However, the results indicated that the connected component analysis applied after binary segmentation was not very effective in separating individual text lines.

Using the YOLO architecture, two approaches were proposed. In the first approach, instance segmentation is used instead of binary segmentation, to avoid the need for connected component post-processing. Although this approach performed well on straight lines, despite overlaps, it failed to properly detect angled lines on the side of the pages, often merging multiple lines into a single prediction.

The best-performing approach in the study was the OBB & Segmentation method that was developed using the YOLO architecture. This approach addressed the complexity of the segmentation task through a two-stage pipeline. It achieved a precision score of 99.3% on the subset containing only straight lines at the 75% IoU threshold, and 95% precision on the mixed subset containing both straight and angled lines.

In addition to model development, this study contributed to the field of Ottoman manuscript text line segmentation by generating a dataset using the best-performing approach. The resulting dataset consists of 110 pages automatically labeled and 40 pages manually labeled images. The performance of the automatically labeled dataset was evaluated on a subset of 20 images, yielding a precision score of 96.5% .

As future work, methods that were originally developed using Arabic datasets and tested on Ottoman data due to the lack of resources can be trained directly on Ottoman datasets. Additionally, increasing dataset variation could support the development of more robust and generalizable models.

BIBLIOGRAPHY

Abdollahi, A. & Pradhan, B. (2021). Integrated technique of segmentation and classification methods with connected components analysis for road extraction from orthophoto images. *Expert Systems with Applications*, *176*, 114908.

Adıgüzel, H., Şahin, P. D., & Kalpaklı, M. (2012). Line segmentation of ottoman documents. In *2012 20th Signal Processing and Communications Applications Conference (SIU)*, (pp. 1–4). IEEE.

Akcan, A., Taşdemir, E. F. B., Kızılırmak, F., Kuru, M., Öncel, F., & Yanıkoğlu, B. (2022). Akis: transcription software of ottoman-turkish texts using deep learning. *Journal of the Ottoman and Turkish Studies Association*, *9*(2), 43–48.

Ataer, E. & Duygulu, P. (2006). Retrieval of ottoman documents. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, (pp. 155–162).

Barakat, B. K., Droby, A., Alaasam, R., Madi, B., Rabaev, I., Shammes, R., & El-Sana, J. (2021). Unsupervised deep learning for text line segmentation. In *2020 25th International Conference on Pattern Recognition (ICPR)*, (pp. 2304–2311).

Bilgin Tasdemir, E. F. (2023). Printed ottoman text recognition using synthetic data and data augmentation. *International Journal on Document Analysis and Recognition (IJDAR)*, *26*(3), 273–287.

Bradley, D. & Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of graphics tools*, *12*(2), 13–21.

Bradski, G. (2000). The opencv library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, *25*(11), 120–123.

British Library & Keinan-Schoonbaert, A. (2019). Ground truth transcriptions for training ocr of historical Arabic handwritten texts. Dataset comprising 120 digitised images from historical Arabic scientific manuscripts (10th-19th century).

Daldali, M. & Souhar, A. (2019). Handwritten arabic documents segmentation into text lines using seam carving. *IJIMAI*, *5*(5), 89–96.

Demir, A. A., Özşeker, I., & Özkaya, U. (2021). Text line segmentation in handwritten documents with generative adversarial networks. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, (pp. 1–5).

Diem, M., Kleber, F., Sablatnig, R., & Gatos, B. (2019). cbad: Icdar2019 competition on baseline detection. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, (pp. 1494–1498).

Dolek, I. & Kurt, A. (2021). Ottoman ocr: Printed naskh font. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*, (pp. 1–5). IEEE.

Ekinci, M. U. (2008). Reflections of the first muslim immigration to america in ottoman documents. In A. D. Balgamış & K. H. Karpat (Eds.), *Turkish Migration to the United States: From Ottoman Times to the Present* (pp. 45–56). University of Wisconsin Press.

Fizaine, F. C., Bard, P., Paindavoine, M., Robin, C., Bouyé, E., Lefèvre, R., & Vinter, A. (2024). Historical text line segmentation using deep learning algorithms: Mask-rcnn against u-net networks. *Journal of Imaging*, *10*(3), 65.

Gatos, B., Stamatopoulos, N., & Louloudis, G. (2010). Icfhr 2010 handwriting segmentation contest. In *2010 12th International Conference on Frontiers in Handwriting Recognition*, (pp. 737–742).

He, L., Ren, X., Gao, Q., Zhao, X., Yao, B., & Chao, Y. (2017). The connected-component labeling problem: A review of state-of-the-art algorithms. *Pattern Recognition*, *70*, 25–43.

International Telecommunication Union (1995). Recommendation ITU-R BT.601-5: Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios. Technical report, ITU. Accessed: 2025-07-15.

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 1125–1134).

Jindal, A. & Ghosh, R. (2023). Text line segmentation in indian ancient handwritten documents using faster r-cnn. *Multimedia Tools and Applications*, *82*(7), 10703–10722.

Jocher, G., Qiu, J., & Chaurasia, A. (2023). Ultralytics yolo.

Khanam, R. & Hussain, M. (2024). Yolov11: An overview of the key architectural enhancements.

Kodym, O. & Hradiš, M. (2021). Page layout analysis system for unconstrained historic documents.

Kolodziejczyk, D. (2022). *Ottoman-Polish Diplomatic Relations (15th-18th Century): An Annotated Edition of'Ahdnames and Other Documents*, volume 18. Brill.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, *2*(1-2), 83–97.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, (pp. 2980–2988).

Mahmoud, S. A., Ahmad, I., Alshayeb, M., Al-Khatib, W. G., Parvez, M. T., Fink, G. A., Märgner, V., & Abed, H. E. (2012). Khatt: Arabic offline handwritten text database. In *2012 International Conference on Frontiers in Handwriting Recognition*, (pp. 449–454).

Majanga, V. & Viriri, S. (2021). Dental images' segmentation using threshold connected component analysis. *Computational Intelligence and Neuroscience*, *2021*(1), 2921508.

Meziani, F., Bouchakour, L., Ghribi, K., Yahiaoui, M., Latrache, H., & Abbas, M. (2021). Arabic handwritten text to line segmentation. In *2021 International Conference on Information Systems and Advanced Technologies (ICISAT)*, (pp. 1–5).

Milletari, F., Navab, N., & Ahmadi, S.-A. (2016). V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, (pp. 565–571). Ieee.

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2021). Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, *44*(7), 3523–3542.

Neche, C., Belaid, A., & Kacem-Echi, A. (2019). Arabic handwritten documents segmentation into text-lines and words using deep learning. In *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*, volume 6, (pp. 19–24).

Nezla, N. A., Mithun Haridas, T., & Supriya, M. (2021). Semantic segmentation of underwater images using unet architecture based deep convolutional encoder decoder model. In *2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, (pp. 28–33).

Onat, A., Yildiz, F., & Gündüz, M. (2006). Ottoman script recognition using hidden markov model. *IEEE Transaction on Engineering Computing Technology*, *14*, 71–73.

Pizer, S. M., Amburn, E. P., Austin, J. D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J. B., & Zuiderveld, K. (1987). Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, *39*(3), 355–368.

Redmon, J., Divvala, S. K., Girshick, R. B., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, *abs/1506.02640*.

Renton, G., Chatelain, C., Adam, S., Kermorvant, C., & Paquet, T. (2017). Handwritten text line segmentation using fully convolutional network. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 5, (pp. 5–9). IEEE.

Ronneberger, O., Fischer, P., & Brox, T. (2015a). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, *abs/1505.04597*.

Ronneberger, O., Fischer, P., & Brox, T. (2015b). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, (pp. 234–241). Springer.

Sapkota, R., Qureshi, R., Calero, M. F., Badjugar, C., Nepal, U., Poulose, A., Zeno, P., Vaddevolu, U. B. P., Khan, S., Shoman, M., Yan, H., & Manoj, K. (2024). Yolo11 to its genesis: A decadal and comprehensive review of the you only look once (yolo) series.

Shaukat, Z., Farooq, Q. u. A., Tu, S., Xiao, C., & Ali, S. (2022). A state-of-the-art technique to perform cloud-based semantic segmentation using deep learning 3d u-net architecture. *BMC bioinformatics*, *23*(1), 251.

Simistira, F., Bouillon, M., Seuret, M., Würsch, M., Alberti, M., Ingold, R., & Liwicki, M. (2017). Icdar2017 competition on layout analysis for challenging medieval manuscripts. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, (pp. 1361–1370).

Suleyman, E., Hamdulla, A., Tuerxun, P., & Moydin, K. (2021). An adaptive threshold algorithm for offline uyghur handwritten text line segmentation. *Wireless Networks*, *27*, 3483–3495.

Tasdemir, E. F. B., Tandoğan, Z., Akansu, S. D., Kızılırmak, F., Sen, M. U., Akcan, A., Kuru, M., & Yanikoglu, B. (2024). Automatic transcription of ottoman documents using deep learning. In Sfikas, G. & Retsinas, G. (Eds.), *Document Analysis Systems*, (pp. 422–435)., Cham. Springer Nature Switzerland.

Tkachenko, M., Malyuk, M., Holmanyuk, A., & Liubimov, N. (2020). Label Studio: Data labeling software.

Wang, H. & Miao, F. (2022). Building extraction from remote sensing images using deep residual u-net. *European Journal of Remote Sensing*, *55*(1), 71–85.

Zaib, S., ur Rehman, M., & Khursheed, F. (2024). Optimizing road extraction with residual u-net: Enhanced training. *Journal of Computing & Biomedical Informatics*, *7*(01), 529–536.

Zhang, S., Hu, Y., & Bian, G. (2017). Research on string similarity algorithm based on levenshtein distance. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, (pp. 2247–2251). IEEE.

Zhang, S. & Zhang, C. (2023). Modified u-net for plant diseased leaf image segmentation. *Computers and Electronics in Agriculture*, *204*, 107511.

Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*, *111*(3), 257–276.

İlhan, M. M. (1991). The ottoman archives and their importance for historical studies: With special reference to arab provinces. *BELLETEN*, *55*(213), 415–472.

APPENDIX A

Publications

The following publication is based on the work conducted during this thesis:

- Kutal, S., Aptoula, E. & Yanıkoğlu, B. Text Line Segmentation of Ottoman Manuscripts. *Presented at Signal Processing and Communications Applications Conference (SIU)*, 2025.