



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΙΧΝΕΥΣΗ ΔΙΑΤΑΡΑΧΩΝ
ΓΡΑΦΗΣ ΚΑΙ ΟΡΘΟΓΡΑΦΙΑΣ ΜΕ ΧΡΗΣΗ
ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

ΒΙΟΛΕΝΤΗΣ ΑΝΤΩΝΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κολομβάτσος Κωνσταντίνος
Αναπληρωτής Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ

Ζυγούρης Νικόλαος
Επίκουρος Καθηγητής

Λαμία 5 Σεπτεμβρίου 2025



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΙΧΝΕΥΣΗ ΔΙΑΤΑΡΑΧΩΝ
ΓΡΑΦΗΣ ΚΑΙ ΟΡΘΟΓΡΑΦΙΑΣ ΜΕ ΧΡΗΣΗ
ΤΕΧΝΗΤΗΣ ΝΟΗΜΟΣΥΝΗΣ

ΒΙΟΛΕΝΤΗΣ ΑΝΤΩΝΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Κολομβάτσος Κωνσταντίνος
Αναπληρωτής Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ
(εφόσον υπάρχει)

Ζυγούρης Νικόλαος
Επίκουρος Καθηγητής

Λαμία 5 Σεπτεμβρίου 2025



UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

WRITING AND SPELLING
DISORDER DETECTION USING ARTIFICIAL
INTELLIGENCE

VIOLENTIS ANTONIOS

FINAL THESIS

ADVISOR

Kolomvatsos Konstantinos
Associate Professor

CO ADVISOR

Zygouris Nikolaos
Assistant Professor

Lamia September 5th 2025

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις ⁽¹⁾, που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάστηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.
2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.
3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια
4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία:/...../20.....

Ο – Η Δηλ.

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ABSTRACT

Producing handwritten text is a complicated cognitive process that involves a wide range of the human brain's components. However, due to limited access to specialists and limited knowledge on specific learning disabilities (SLDs), children who are unable to fully develop such skills often go undiagnosed and/or unnoticed. In addition, the orthographic complexity of the Greek language and its opaque spelling nature present additional challenges, particularly when compared to languages with transparent orthographies such as Finnish or Italian. Despite this, there currently exists no accessible means to detect such disorders online in Greece without the involvement of trained professionals. This thesis proposes methodologies that leverage a custom-built deep learning Optical Character Recognition (OCR) model to identify potential writing disorders – specifically writing and spelling disorders – in handwritten Greek text.

The motivation behind this project is first outlined, followed by a detailed overview of SLDs, their defining characteristics, and techniques used to identify them. Next, the fundamentals of deep learning and deep neural networks are introduced, along with their evolution, core components, and their applicability to complex problem-solving tasks such as image analysis. The basics of OCR are then presented, and the architecture of the developed model is analysed, explaining the reasoning behind the design choices and techniques adopted to detect potential writing and spelling disorders on handwritten text. Finally, the model's performance is evaluated using data from two control groups, with the results presented and analyzed.

Table of Contents

ABSTRACT	0
CHAPTER 1	5
MOTIVATION.....	5
1.1 INTRODUCTION	5
1.2 LEARNING DISABILITIES AND ACADEMIC PROGRESS	5
1.3 FLAWS IN MODERN LLM-BASED OCR	6
CHAPTER 2	8
WRITING AND SPELLING DISORDERS	8
2.1 INTRODUCTION TO SLDS	8
2.2 WRITING AND SPELLING SLDS	9
2.2.1 DYSGRAPHIA / WRITING DISORDERS	9
2.2.2 DYSORTHOGRAPHY / SPELLING DISORDERS	9
2.3 WRITING & SPELLING DISORDER SYMPTOMS	10
2.3.1 WRITING DISORDER SYMPTOMS	10
2.3.2 SPELLING DISORDER SYMPTOMS	11
2.4 COGNITIVE DEFICITS	13
2.5 SLD DETECTION SCREENINGS	14
2.5.1 WORLDWIDE SCREENING TOOLS	14
2.5.2 SCREENING TOOLS IN GREECE	14
2.6 DIAGNOSIS	15
CHAPTER 3	16
DEEP LEARNING	16
3.1 INTRODUCTION TO AI	16
3.1.1 ARTIFICIAL INTELLIGENCE	16
3.1.2 MACHINE LEARNING	16
3.1.3 DEEP LEARNING	17
3.2 NEURAL NETWORK FUNDAMENTALS	18
3.2.1 PERCEPTRONS	18
3.2.2 ACTIVATION FUNCTIONS	19
3.2.3 LOSS FUNCTIONS AND OPTIMIZATIONS	20
3.2.4 BACKPROPAGATION	21
3.3 COMPUTER VISION	22
3.3.1 CNNs	23
3.3.2 TRANSFORMERS	25
3.4 TRAINING DEEP LEARNING MODELS	25
3.4.1 DATASET PREPARATION AND AUGMENTATIONS	26
3.4.2 OVERFITTING, UNDERFITTING, AND REGULARIZATION	27

3.5 DEEP LEARNING FRAMEWORKS AND TOOLS.....	28
3.5.1 TENSORFLOW AND PYTORCH	28
3.5.2 LIBRARIES	28
CHAPTER 4	30
WRITING AND SPELLING DISORDER DETECTION	30
4.1 INTRODUCTION TO OCR - NEED FOR DATA	30
4.2 DATA COLLECTION	31
4.3 DATASET PRE-PROCESSING	32
4.3.1 DATASET FORMAT.....	32
4.3.2 AUGMENTATIONS AND DATA LOADING	33
4.3.3 TRANSFORMS.....	34
4.3.4 TRAIN-TEST DATA SPLIT	34
4.4 MODEL ARCHITECTURE.....	35
4.4.1 FULL MODEL PIPELINE	35
4.4.2 DYNAMIC AUGMENTATIONS.....	37
4.4.3 MODEL ARCHITECTURE ANALYSIS	38
4.5 MODEL PERFORMANCE.....	41
4.6 INPUT IMAGE PREPROCESSING AND SEGMENTATIONS	41
4.6.1 INPUT IMAGE PREPROCESSING	41
4.6.2 NOTEBOOK LINE SEGMENTATION	42
4.6.3 SINGLE-CHARACTER SEGMENTATION.....	45
4.6.4 WORD SEGMENTATION	47
4.6.5 CHARACTER PREDICTION	48
4.7 WRITING AND SPELLING DISORDER DETECTION	49
4.7.1 SPELLING DISORDER DETECTION.....	49
4.7.2 WRITING DISORDER DETECTION	50
4.8 EXPERIMENTS AND OCR COMPARISON	52
4.8.1 WORD SEGMENTATION EXPERIMENTS.....	52
4.8.2 CHARACTER SEGMENTATION EXPERIMENTS	53
4.8.3 CHARACTER RECOGNITION EXPERIMENTS	53
CHAPTER 5	55
TESTS AND USE CASES.....	55
5.1 TEST STRUCTURE	55
5.2 TEST RESULTS	55
CHAPTER 6	57
CONCLUSION, LIMITATIONS, AND LATER WORKS.....	57
6.1 CONCLUSION AND RESULTS	57
6.2 LIMITATIONS AND LATER WORKS.....	57

REFERENCES	59
-------------------------	-----------

CHAPTER 1

Motivation

1.1 Introduction

This chapter presents the motivation underlying the project and explains the reasoning for developing a custom AI system designed for single-character image analysis and pattern recognition. Specifically, it highlights the academic challenges faced by individuals with learning difficulties, the necessity for an identification mechanism tailored to the Greek language, as well as the importance of a single character analysis optical character recognition (OCR) algorithm for handwritten text – particularly in the context of spelling and writing disorder detection.

1.2 Learning Disabilities and Academic Progress

Identifying spelling and writing disorders in Greek text is not just a technical challenge, but rather a critical step toward inclusive education and individualized learning. Specific learning disabilities (SLDs) such as dysgraphia and dysorthography can significantly hinder a student’s academic progress as well as their self-confidence and self-esteem [1]. Yet, they often go undiagnosed due to delayed assessments, insufficient knowledge of SLDs, or limited access to specialists [41]. It’s not uncommon for a Greek child to receive their diagnosis years after they are qualified to, or even not receive one at all, resulting in academic setbacks and emotional strain [41].

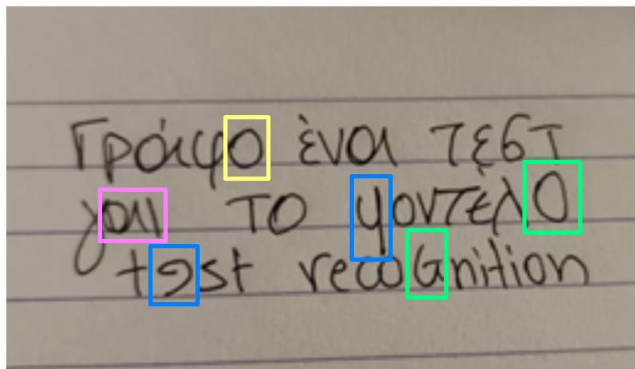
Additionally, spelling and writing disorder detection is a greatly underdeveloped field, and based on currently available resources, there appears to be no existing means of automatic self-assessment for writing and spelling disorders in Greece. This absence makes early detection inaccessible for many individuals and, as a result, countless students and adults continue to struggle unnoticed [41].

The Greek language presents unique phonological, morphological, and orthographic complexities, while also maintaining a high level of phonetic correspondence [2]. For example, the redundancy of vowel sounds such as “ι”, “η”, and “υ” poses complications and results in phonologically acceptable misspellings [3]. A 2009 study by Protopapas & Vlahou shows that the Greek language’s consistency is around 95% in the feedforward direction (reading) and around 80% in the feedback direction (spelling) – showing its asymmetrical nature of orthographic processing [4]. Specifically, decoding written text is considered quite reliable; however, accurate spelling demands more nuanced phonological awareness and lexical knowledge.

Therefore, the development of an AI-powered OCR model tailored for Greek handwriting is proposed as a means to create an accessible, efficient, trustworthy, and scalable tool that could be used by educators, clinicians, or even students themselves to identify writing SLDs early. This project was ultimately designed to bridge the gap between technology and special education and to propose a variety of ways that AI can be a useful approach to solving such problems.

1.3 Flaws in Modern LLM-based OCR

Optical Character Recognition is a deeply researched field that has roots stretching back over a century. Since then, multiple algorithms that aim to improve the accuracy of the OCR process have been proposed, and in recent years, there have been great advancements made due to the introduction of Deep Learning [5]. In particular, OCR on handwritten text is a notoriously complicated task due to connected components and inconsistent letters that often require context to be read correctly. As a result, modern OCR models with high accuracy tend to make predictions based on context and are usually trained on entire words rather than single letters. Thus, a phenomenon referred to as “hallucination” can often be observed when a model attempts to understand something that it is unable to fully transcribe [6]. Particularly, such models often tend to automatically correct text based on context cues and probabilities, making the transcription process inaccurate and often creating distortions in the original transcript. Such behavior is excellently captured by Rangasrinivasan et al. [7], and an experiment we conducted on the GPT-4o model can be found in Figure 1.1. The original text in that figure includes characters written in the wrong order, random uppercases and in reverse – all of which the model takes the liberty to self-correct. Subsequently, using such approaches to detect writing and spelling disorders poses a great risk of incorrectly written words being automatically corrected in the transcription process.



GPT-o4 Prediction

Γράφω ένα τεστ για
το μοντέλο test
recognition

Fig. 1.1. Common example of an OCR model distorting an original transcript, where:
Yellow = spelling error, Pink = wrong order, Blue = inverted letter, and Green = uppercase letter.
Prompt: Transcribe this handwritten sentence.

In the case of spelling and writing disorder detection, it is first necessary to assume that large parts of the handwritten text being handled contain words that are spelled incorrectly and/or cannot easily be deciphered. This ultimately creates a need for a custom-built model used for character-level analysis instead of typical word detection. Although there has been significant research conducted on single character segmentation over the past years, it is currently considered near impossible to handle single characters due to the hundreds of different writing styles that combine letters, making it challenging for the computer to understand where they need to be split. Moreover, the Greek language contains characters that often look similar in handwritten text and are thus nearly impossible to decipher unless context is present. Such example can be found in Figure 1.2, where the writer on the left wrote the letter “v” similar to the letter “u” in the word “αρνίεσαι”. If someone were to see the same letter in a different context, they would most likely read it according to the context of the sentence. Additionally, the writer on the right in the same figure did the opposite and wrote the letter “u” similar to the letter “v”. Finally, Figure 1.3 shows another example of context-based transcription where the

writer on the right connected the letter “τ” with the letter “η” and thus resulted in the letter “η” looking similar to the letter “ι”.

Sequentially, there is a need for an optimal OCR algorithm tailored to Greek text in order to properly capture every word letter-by-letter and provide proper feedback based on spelling and writing. This thesis aims to tackle the single character image processing problem and propose methodologies and algorithms that can effectively detect potential disorders in handwritten Greek text.

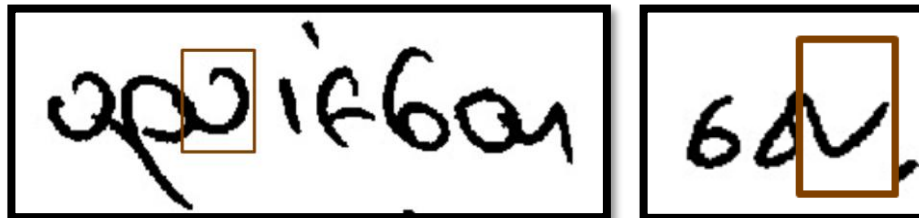


Fig. 1.2. Example of a need for context-based letter classification.



Fig. 1.3. Example of different writing styles with connected letters.

CHAPTER 2

Writing and Spelling Disorders

2.1 Introduction to SLDs

Specific Learning Disorder (SLD) is a biologically based neurodevelopmental condition characterized by atypical cognitive processing, which underlies observable behavioral manifestations [12]. It markedly interferes with the acquisition and effective use of core academic skills – such as reading, writing, and mathematics – despite the presence of normal or above-normal intelligence, sufficient educational instruction, and access to appropriate learning environments [8, 9, 12]. Additionally, studies have shown that the prevalence rates of SLDs range between 5-15% internationally [10] and 8.5-9.5% specifically in Greece [11].

As per the DSM-5 [12], there are three main types of specific learning disorders, and they can be classified as Dyslexia (reading disorder), Dysgraphia (disorder of writing expression), and Dyscalculia (disorder of calculation). Additionally, Dysorthography can be considered part of the SLD umbrella, suggesting a disorder in spelling [11]. It is worth noting that all of them begin with the prefix “dys”, which denotes a developmental origin, implying that the individual was born with the condition, as opposed to the prefix “a”, which typically refers to an acquired deficit resulting from injury or illness [13]. For example, “agraphia” and “dysgraphia” imply different meanings. A table of SLD symptoms, as stated by the DSM-5 [12], can be found in Table 2.1.

It is worth mentioning that specific learning disabilities can typically be reliably diagnosed from 3rd grade and up (8-9 years of age), when academic demands increase and persistent patterns of difficulty become more evident [12]. Earlier signs may still be observed, though a formal diagnosis is often delayed until this stage to ensure developmental variability is accounted for [9]. Additionally, while certain studies suggest minimal gender disparity in reading-related SLDs, several others tend to be more prevalent among boys [14].

Dyslexia often co-occurs with other neurological conditions [15]. For example, research by Döhla et al. shows that dysgraphia appears to be closely related to Dyslexia [16], with the prevalence of developmental writing disorders in children with dyslexia being around 70% [17]. Additionally, findings in the works of Butterworth (2003) and Wilson et al. (2014) show that 40% of learners with dyslexia pose difficulties with calculations [18, 19].

Inaccurate or slow and effortful word reading.
Difficulty understanding the meaning of what is read.
Difficulty with written expression.
Difficulty with spelling.
Difficulties with mathematical reasoning.
Difficulties mastering number sense, number facts, or calculation.

Table 2.1. Specific learning disorders symptoms, as stated by DSM-5 [12].

2.2 Writing and Spelling SLDs

The production of written text is a complicated cognitive process that activates a wide network of brain regions as a product of thousands of years of human evolution. To illustrate the neurological components involved, Figure 2.1 presents a simplified diagram. At schools, a large proportion of the average elementary school academic curriculum focuses on the acquisition of writing skills, while research shows that around 50% of a typical school day consists of activities related to the process of writing [20]. It could be said that one of the ultimate goals of education is to enable students to read and write effortlessly and almost automatically. However, the acquisition of writing is a skill that many students are unable to obtain, resulting in academic struggles and difficulties in catching up with students of typical development [41].

In the following two sections, a brief overview is presented of the two writing-related specific learning disorders that were the focus of this project, including their definitions, characteristics, and typical identification methods.

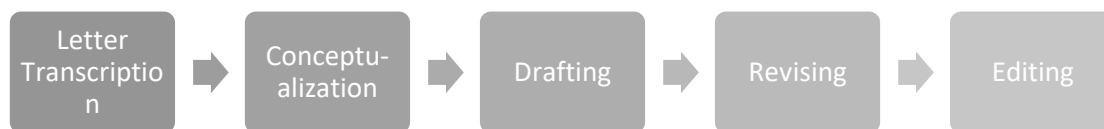


Fig. 2.1. A typical writing process that is often performed unconsciously.

2.2.1 Dysgraphia / Writing Disorders

Dysgraphia is a neurological specific learning disability that manifests as persistent challenges in developing proficient writing skills [16]. As mentioned in Chapter 2.1, it is closely related to dyslexia, and studies show that the prevalence of developmental dyslexia is around 17%, opposed to that of developmental writing disorders being 7-15% among children worldwide [21, 22]. Additionally, Chung et al. [23] state that children who fail to attain writing automaticity by third grade (9-10 years of age) are at significantly higher risk of encountering challenges with higher-order writing tasks, as the child's executive cognitive resources may remain occupied by the physical act of forming letters.

Case studies have shown that the cerebellum plays an important role in dysgraphia, and specifically in the coordination of writing [24]. In addition, writing expression involves distinct cognitive mechanisms, which may be variably impaired in individuals with dysgraphia. For example, accurate spelling of dictated words is heavily sustained within the working memory, and tasks involving novel or pseudowords depend on sublexical spelling processes that apply phoneme-grapheme rules [23].

2.2.2 Dysorthography / Spelling Disorders

According to DSM-5 [12], Dysorthography is considered a subtype of specific learning disorder with impairment in written expression. It is a specific type of writing impairment marked by frequent spelling mistakes such as letter omissions and reversals, and it is often linked to dyslexia, though it can also manifest independently [12, 2, 25]. As of the 2013 version of the DSM-5 [12], it can be diagnosed as "spelling acquisition disorder" [26].

The prevalence of spelling disorders is influenced by linguistic characteristics, the degree of orthographic transparency within a given language, and specific diagnostic frameworks employed [2]. Particularly, the Greek language's distinct morphological complexity and linguistic evolution introduce acquisition challenges in spelling, which are not encountered in more structurally uniform languages such as Finnish [2].

2.3 Writing & Spelling Disorder Symptoms

A comprehensive understanding of the symptomatic manifestations of writing and spelling disorders is essential for accurate identification and for providing effective support to individuals affected by them. The following two sections introduce key behavioral and cognitive indicators that often accompany these disorders. Additionally, real-world examples from handwritten text produced by individuals diagnosed with writing and spelling disorders are showcased. By examining the symptoms in depth, we can gain a greater understanding of the nuances of written expression and orthographic processing and provide a preface of how they can be detected.

2.3.1 Writing Disorder Symptoms

Though there are several core indicators of writing disorders, they often go undetected at early stages due to the continuous development of handwriting during the early years of schooling. Additionally, isolated cases of dysgraphia may remain unrecognized well into adolescence or early adulthood [23].

Another noteworthy complication in detecting writing disorder symptoms is their variability in the nature of the writing task and the environment. Children may exhibit inconsistencies and disorganization when composing an independent story but show relatively stable handwriting when copying from a board or a notebook [23].

Chung et al. [23] illustrate different warning signs split into three age groups, as published by the United States National Center for Learning Disabilities. Specifically:

- Preschool-aged children may:
 - Hold writing tools or position their bodies awkwardly while writing
 - Become easily fatigued during writing activities
 - Avoid writing and drawing tasks
 - Produce malformed, reversed, upside-down, or unevenly spaced letters
 - Struggle to keep their writing within the designated lines or margins
- Elementary school-aged children often show:
 - Illegible handwriting
 - Frequent switching between cursive and print writing styles
 - Challenges in written comprehension, finding words, and completing sentences
- Teenagers and young adults may experience:
 - Difficulty organizing their thoughts in written form
 - Problems with written grammar and sentence structure that are not evident in their spoken language

Authentic writing disorder examples were provided by my professor in a learning difficulties course [27], and they can be found in Fig. 2.2. The ground truth is also provided to show a comparison with typical development writing. In both paragraphs (b) and (c), it can be observed that the sizing and spacing of the letters are inconsistent, and the principle of maintaining a straight line is not satisfied. Additionally, various characters are missing, and the order of some is problematic (e.g., καρταρται instead of κατάρτι). Finally, the Greek language accent point tonos is often omitted in paragraph (b) and not used at all in paragraph (c).

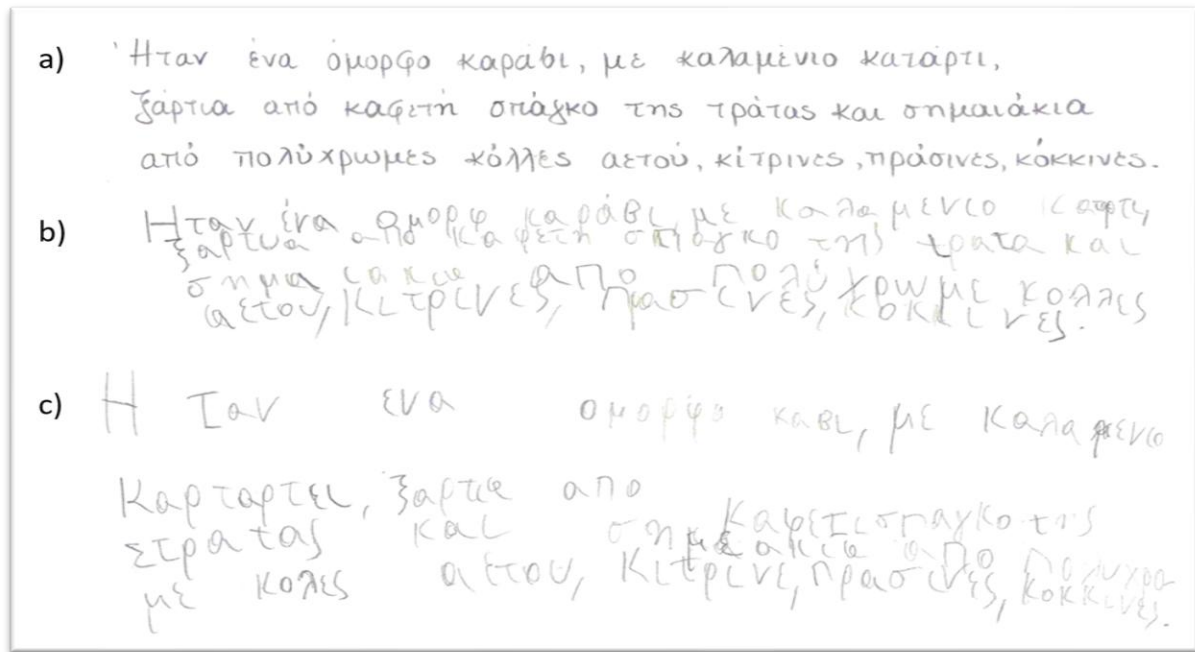


Fig. 2.2. Writing disorder examples. Paragraph (a) is the ground truth and paragraphs (b) and (c) were written by two children diagnosed with writing disorder.

2.3.2 Spelling Disorder Symptoms

Children who struggle with spelling disorders tend to have difficulties in key writing-related skills. Specifically, those include mapping sounds and word parts to their written forms, using correct spelling based on historical usage, following grammar-based spelling rules, and recognising individual words within a sentence [26].

Spelling errors manifested from a spelling disorder are often classified as two distinct types based on the linguistic level of the errors. In particular, they are referred to as historic and grammatical [26]. Historic spelling errors refer to historically determined spellings that do not follow specific phonological rules, such as the “α” in the word “παιδί”. Grammatical spelling errors affect inflectional morphology and denote spelling errors in tense, gender, case, and/or singular/plural forms [26]. In the same word example “παιδί”, the “ι” is considered a grammatical spelling error. More examples of historical and grammatical spelling can be found in Table 2.2.

Words	Historic	Grammatic
το σκυλί	το σκυλί	το σκυλί
η λέξη	η λέξη	η λέξη
εσύ φεύγεις	εσύ φεύγεις	εσύ φεύγεις
αυτό είναι	αυτό είναι	αυτό είναι

Table 2.2. Examples of Historic and Grammatic spelling, where red denotes the respective spelling type.

According to Mazade (2011) [28], spelling errors found in learning disabilities can be grouped into two systems: form-based typologies and deficit-based origins, and they can be classified into seven and six distinct categories respectively [26].

I. Typological Classification of Errors (Form-Oriented)

These reflect on structural characteristics of the errors themselves.

1. Errors of Spatial Letter Placement
 - a. Visual confusability manifested from similar shapes or orientations (e.g., β -> δ, ε -> 3, ρ -> 9).
 - b. Omission and/or additions of letters or syllables (e.g., τρ -> ρτ, ππ -> π).
 - c. Letter reversals (e.g., στρατός -> σαρτός).
 - d. Word fusion (e.g., το σπίτι -> τοσπίτι).
2. Phonological Errors
 - a. Voicing confusion (e.g., δέμα -> τέμα).
 - b. Simplification of consonant clusters (e.g., όμορφο -> όμορο).
 - c. Assimilation (e.g., παγώνω -> παγώγω).
 - d. Insertion of letters or syllables (e.g., κατάρτι -> κατατάρτι).
 - e. Vowel distortion (e.g., κίτρινο -> κίτρινι)
3. Errors in Morphological Encoding

Incorrect use of suffixes or stems (e.g., σκυλί -> σκυλί, τρώω -> τρώο)
4. Errors in Representing the same Phoneme with Alternative Graphemes

E.g., πιστεύω -> πιστέβω.
5. Errors in Grammar

Despite knowing grammatical rules, students may struggle to apply them (e.g., της -> τις).
6. Lexical Substitution Errors

The original word is replaced by a semantically unrelated one (e.g., σκύλος -> γάτα).
7. Homophonic Word Errors

Words that sound similar are interchanged (e.g., μίλα -> μήλα, σήκω -> σύκο).

II. Classification Based on Underlying Deficit (Origin-Oriented)

These refer to cognitive processes that fail during spelling.

1. Phonological Deficit Errors

Difficulty in phonological-letter matching (often co-occurring with dyslexia).
Includes omissions, substitutions, additions, or letter reordering (Examples from I. b and c).
2. Morphological Errors without Phonological Impact

Errors are confined to the visual structure of the word
(e.g. κόκκινο -> κώκινο, ξάρτια -> κσάρτια).
3. Semantic Errors

Incorrect word use that alters the meaning (e.g. θήρα -> θύρα).
4. Grammatical Errors

Errors in tense, gender, case, forms (e.g. εσύ φεύγεις -> εσύ φεύγει, οι λέξεις -> η λέξεις).
5. Stress

Misplacing accent affects phonology and sometimes semantics (e.g. διπλά -> δίπλα).
6. Other Errors

Includes punctuation, letter orientation, and capitalization issues
(e.g. το σπίτι μου -> το σΠίτ μού, νερό -> νΕ9ο)

Similar to the previous section, an authentic example of written text produced by a child diagnosed with a spelling disorder can be found in Figure 2.3. Accent marks (tonos) are often omitted, and several spelling mistakes such as “κόλες” instead of “κόλλες” can be found. The ground truth of the particular passage is the same as that of passage (a) in Figure 2.2. Similarly, the image was provided by my professor.

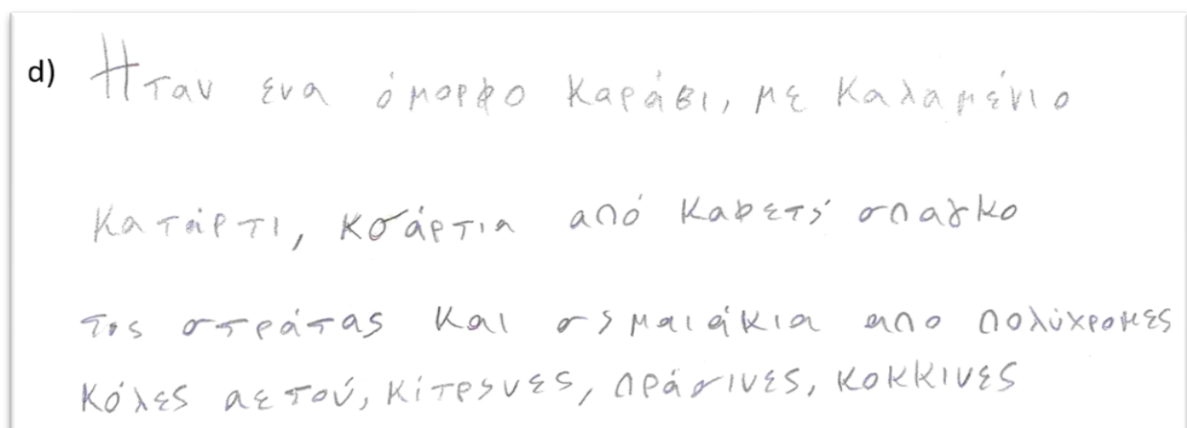


Fig. 2.3. Spelling disorder example.

2.4 Cognitive Deficits

Individuals with writing and spelling disorders often exhibit cognitive deficits that intervene with their ability to process, retain, and reproduce written language [134, 136]. These skills are not always immediately observable in surface-level writing errors, but can significantly impact the acquisition and execution of writing-related skills [134, 135].

One frequently observed deficit is in auditory and visual working memory, which refers to the ability of containing information for a short period of time [136, 137]. Auditory working memory deficits affect the individual's ability to retain verbal information, such as remembering multi-step instructions or holding a sentence structure while writing [134]. On the other hand, visual working memory deficits may hinder the individual's ability to recall letter shapes, grammatical or word patterns, or spatial arrangements on the page, resulting in disorganized or inconsistent written text [136].

Another common deficit involves phonological processing, which refers to the ability to recognize and manipulate the sound structure of a language [139]. Individuals with phonological processing difficulties often struggle to segment words into phonemes, blend sounds, or accurately match sounds to corresponding graphemes [138]. Instead, they may rely on inconsistent or incorrect sound-symbol associations, which may lead to spelling errors and impair both language decoding and encoding skills [140, 141].

Additionally, visual-spatial processing disorders can interfere with an individual's ability to perceive and organize visual information [136, 137]. This may manifest in difficulties with letter formation, alignment, spacing, and handwriting legibility [136]. Individuals with such deficits may also struggle to navigate the layout of a page or follow directional cues, which may result in deviations from a notebook's lines [137].

Finally, inhibitory control is a key concept worth highlighting. In essence, it refers to the executive function that allows individuals to suppress instinctive reactions and emotions in order to stay focused in a designated task [149]. Further research shows that cognitive deficits in writing disorder often extend into executive functions and orthographic knowledge. Core executive functions, such as planning, attention control, inhibitory processes, and organization, play a vital role in written expression, and deficits in these areas may result in difficulties organizing ideas, error control, and maintaining attention [150, 151]. For instance, individuals with reduced inhibitory control may know the correct spelling of a word, but still produce errors manifested by difficulty in suppressing incorrect phonological representations.

2.5 SLD Detection Screenings

Although the field of writing and spelling disorders remains significantly under-researched, considerable efforts have been made globally to develop screening tools that analyze behavioral indicators and linguistic patterns associated with learning disabilities. Such tools and algorithms have undergone substantial advancements in recent years, largely due to the rise of artificial intelligence and deep learning technologies, which have enabled more precise pattern recognition and classification capabilities. This section examines the progress that has been made in the past years to detect learning disorders both worldwide and exclusively in Greece.

2.5.1 Worldwide Screening Tools

Numerous screening tools aimed at detecting SLDs have been developed globally. However, for the purpose of this overview, only a selection of the most widely used and contemporary approaches will be examined. It is important to note that all tools presented are intended for identifying potential risk indicators of SLDs and do not serve as diagnostic tools.

1. Shaywitz DyslexiaScreen, 2016

Developed by Sally Shaywitz et al. in the United States, it is used as a brief screener tool with emphasis on phonological, linguistic, and academic performance and is used to help identify students at risk for developmental dyslexia [29]. Case studies conducted by Burns et al. (2022) found a decision accuracy of 45% using a sample of 115 students [30].

2. DIBELS Next, 2010

DIBELS Next (Dynamic Indicators of Basic Early Literacy Skills) is a series of brief assessments designed to monitor the development of early literacy skills in students from Kindergarten through Grade 6 (from 5 until ~12 years of age) [31]. It was developed at the University of Oregon, and it is primarily used in the United States. Its main purpose is to identify children who may be at risk for reading difficulties. In the same study by Burns et al. (2022), DIBELS Next scored a decision accuracy of 78% [30].

3. Devi et al.'s Machine Learning based Online Rehabilitation Tool for SLD Detection, 2019

Introduced by Devi et al. [32], this web-based screening tool uses a modern machine learning algorithm approach – specifically, decision trees – to analyze student data. It is used to identify indicators for dyslexia, dysgraphia, and dyscalculia, and it scored an accuracy of 91.86% in the authors' tests for LD identification.

4. WISC-V, 2014

WISC (Wechsler Intelligence Scale for Children) can be considered the most widely used screening tool worldwide for SLD identification [33]. It utilizes different indicators that are used to detect patterns related to Dyslexia, Dysgraphia, and Dyscalculia. Additionally, it supports over 20 languages, including English and Greek. Its strength lies in its ability to examine the child's verbal comprehension, abstract thinking, problem solving, processing speed, working memory, and attention [34]. A case study by Raiford et al. reported a reliability coefficient ranging from .91 to .96 – indicating its high accuracy and consistency [35].

2.5.2 Screening Tools in Greece

As of 2025, a few screening tools tailored towards SLD detection in children of different age groups have been created in Greece. All of them include game-like exercises that test spelling skills, though none of them have a means to check writing and ultimately identify writing and spelling disorders manifested in the production of handwritten text. The following screening applications are often used

in Greece as an SLD indicator, and by no means a diagnostic tool. Their characteristics, contents, and the age groups they are tailored for are briefly analysed below.

1. eMADys, 2000

Tailored primarily for first-year gymnasium (12-13 years of age) students, it focuses on cognitive indicators that may signal specific learning disorders rather than evaluating academic knowledge. It includes a variety of tests such as spelling quizzes and memory exercises [36].

2. Askisi-SD, 2025

An accessible web-based screening tool designed to assess cognitive and orthographic deficits in Greek primary school students (6-12 years of age) [37]. It emphasizes on spelling disorders and aims to detect them early. Specifically, it includes tasks that evaluate spelling processes, visual and auditory working memory, and response times. In a recent study involving 264 children, the authors scored a Spearman-Brown coefficient of 0.78 [2], showcasing its reliability and consistency.

2.6 Diagnosis

The diagnosis of writing and spelling disorders presents challenges both in Greece and internationally due to the complex linguistic, cognitive, and motor factors involved. According to ICD-10 [38], in order for one to be eligible for a dysorthography diagnosis, the individual must have shown clinical features, indicators, and manifestations from the early years of schooling, and in order to receive a standalone diagnosis, the symptoms must occur without coexisting reading difficulties. In addition, diagnosis can be established once the individual consistently fails to learn spelling, despite receiving targeted support at school [39, 26]. In Greece, the ACS Athens Educational & Diagnostic Testing Center [39] follows a series of evaluation methods, as shown in Table 2.3.

Once the first evaluation procedure is completed, a comprehensive five-step assessment process begins. Specifically, it starts with a referral, followed by an appointment arrangement and then an intake interview to collect background information. A series of assessments is then conducted over several days, and a final meeting is carried out where the results and tailored recommendations are shared with the guardian.

Collection of educational and clinical background
Interview with the student's guardian to gather family history
Consultations with classroom teachers to explore academic and behavioral concerns
Direct behavioral observations conducted both during classroom activities and the evaluation process
Examination of prior psychoeducational assessments and reports

Table 2.3. Evaluation methods for SLD diagnosis.

CHAPTER 3

Deep Learning

3.1 Introduction to AI

The concept of Artificial Intelligence (AI) began to gain popularity in 1950 when Alan Turing posed the question of whether machines can think in his “Computing Machinery and Intelligence” paper [42]. Since then, extensive research has taken place in the AI industry [43], and as of 2025, large language models (LLMs) are widely used to complete tasks that one would consider impossible for a computer to handle a few years ago.

According to a global survey conducted by McKinsey [43], the number of users actively engaging with AI tools in their daily lives has shown a sharp increase, reaching 378.8 million – or 3.9% of the global population – as of 2025. This number is estimated to increase by 92.4% by 2030 [44]. Additionally, the same survey showcases the immense funding for AI, which has reached a total of 91 billion dollars in 2025 alone.

Despite its growing popularity, many individuals still struggle to fully understand the true meaning of AI – how it functions and what differentiates it from Deep Learning. This chapter aims to provide useful insights into AI and Deep Learning, including neural network fundamentals, key architectures, the training processes, and commonly used frameworks, with an emphasis added on those utilized throughout this thesis.

3.1.1 Artificial Intelligence

Artificial Intelligence is considered a broad domain that encompasses both machine learning and deep learning [45], and its history dates back to the 20th century. Accordingly, it is worthwhile to begin with a brief historical overview. Although several definitions have been proposed over the years [45, 46], the High-Level Expert Group on Artificial Intelligence (AI HLEG) of the European Commission (EC) provides a clear task-based definition that beautifully captures the essential qualities of AI. Specifically, they define it as: “Systems that display intelligent behavior by analysing their environment and taking actions – with some degree of autonomy – to achieve specific goals.” [47, 46]. In order to further showcase the correlation between artificial intelligence, machine learning, and deep learning, a visualization can be found in Figure 3.1.

The first roots of AI were set in 1956 at a workshop at Dartmouth College, where four pioneering researchers conducted a summer study session aimed at investigating the possibilities of making machines “simulate aspects of human intelligence” [48]. Following this, the field of artificial intelligence began to blossom, creating hundreds of new possibilities for automation, data, natural language understanding, and more. An iconic moment in AI history worth mentioning is the chess match in 1997 between IBM’s Deep Blue and the world champion Garry Kasparov, which recorded the first-ever computer win over a world champion [49].

3.1.2 Machine Learning

Although machine learning is frequently discussed within the context of AI, the two terms are often used interchangeably, which contributes to conceptual ambiguity and confusion [50, 51]. Janiesch et

al. define machine learning as the ability of systems to acquire knowledge from specialized training datasets in order to enable the automatic development of models to perform designated tasks [52]. Specifically, it refers to the process of allowing computers to autonomously learn important patterns and connections through observed data for the purpose of making predictions or creating decisions [53].

Some representative machine learning problems worth mentioning include classification, regression, and clustering. Classification can be considered the act of categorizing data into predefined classes [53]. For example, a singular handwritten digit may be categorized as part of a class from a designated set of classes 0 to 9 (making up for 10 distinct classes). Following, regression refers to the prediction of continuous values from a series of input data [52]. One common real-world application may be considered house price prediction, where future property values are estimated based on several factors such as location and size [54, 55]. Finally, clustering is a learning approach used to partition data points based on shared characteristics [56]. It is considered an unsupervised technique, denoting the ability to analyse unlabeled data without the need for human intervention [57]. A frequently observed application is customer analytics, where businesses group customers into specific categories based on purchasing behaviours [56]. Additionally, it may also be found in the medical field, where patient data is analysed to discover hidden patterns [58].

3.1.3 Deep Learning

Deep learning is a subset of machine learning, which itself is a subfield of artificial intelligence [5]. Its aim is to create computational models comprised of large layers of neurons with the purpose of learning representations of data with multiple levels of abstraction, and using that to make accurate decisions [4, 45]. Additionally, it is exceptionally effective in environments involving high data complexity and the availability of large-scale datasets [45].

Its origin lies in the desire to create computer systems that mimic the structure of the human brain, and the first recorded milestone dates back to 1873 when Alexander Bain introduced the term “neural groupings” [59]. Following, the foundations of deep learning were deeply set in the 20th century, with revolutionary concepts like perceptrons (1958) and backpropagation (1974) being introduced [60, 61]. Its growth accelerated in the early 2000s, with a key achievement being the release of OpenAI’s large language model ChatGPT in 2022 [62].

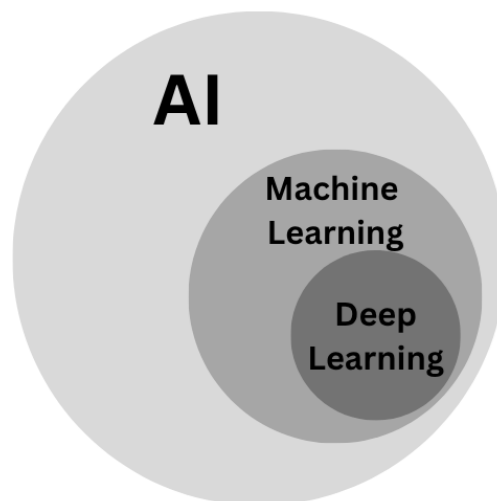


Fig. 3.1. The correlation between AI, Machine Learning and Deep Learning.

3.2 Neural Network Fundamentals

Artificial neural networks (ANNs) are inspired by the human brain's natural neurons, and they can be considered a core element of deep learning, enabling systems to learn complicated patterns from large-scale datasets efficiently [63, 45]. Their history goes back to 1943, when Warren McCulloch and Walter Pitts proposed a mathematical model of the nervous system that produced an output signal based on an input through artificial neurons [64]. This chapter aims to provide a comprehensive overview of the fundamental components of a neural network, beginning by showcasing the most basic type – perceptrons, as well as their predecessor – multilayer perceptrons. Following, the introduction of non-linearity to networks through activation functions is presented. Finally, the key element during the learning process – backpropagation – as well as loss functions and optimization techniques are briefly analysed.

3.2.1 Perceptrons

For centuries, humans have made attempts to understand how the human brain functions, with the first roots found in the works of the Greek philosopher Aristotle in 300 BC [59]. Following to the 20th century, the psychologist Frank Rosenblatt proposed one of the first recorded algorithms to mimic biological learning through a computer by adjusting values called “weights” the same way that the human brain reinforces synapses [60]. This algorithm was named “perceptron” and is now considered the foundation of modern neural networks [60, 63]. A side-by-side comparison of a human brain's synapse and a low-level ANN architecture can be found in Figure 3.2 (a) and (b), where the term weighted sum refers to the mathematical sum of neuron inputs and their respective weights in order to determine importance in characteristics. In addition, activation functions can be considered the threshold that allows information to flow onto a following neuron [65], and more information about them can be found in section 3.2.2.

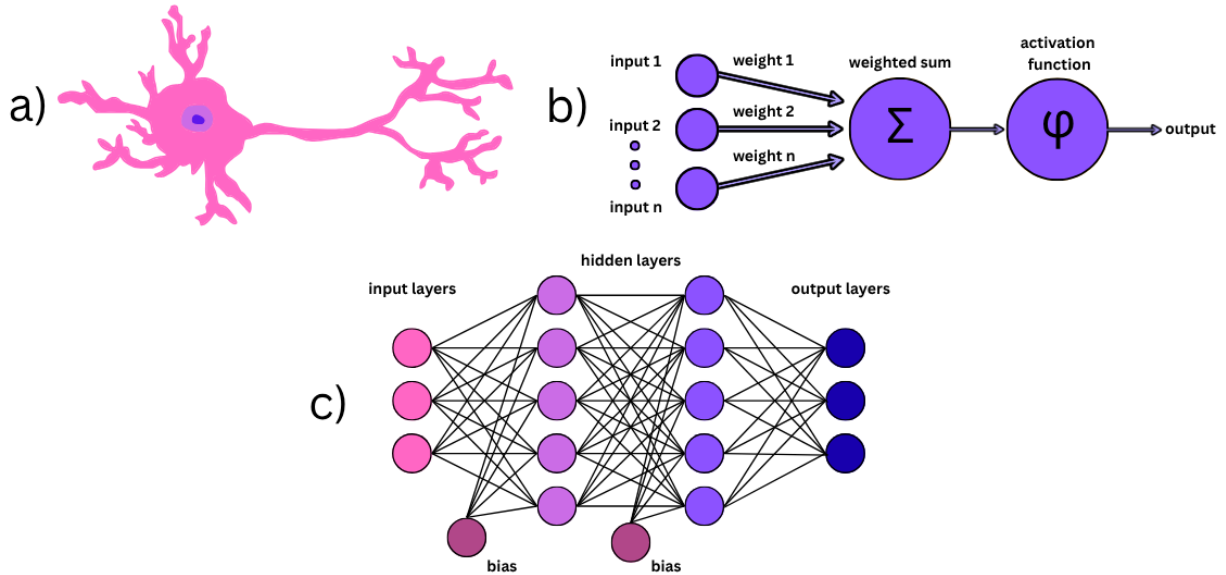


Fig. 3.2. A side-by-side comparison of (a) the human brain's synapse, (b) a low-level ANN architecture, and (c) a typical MLP architecture.

Having received an impact from perceptrons, the most frequently used neural network architecture nowadays has come to be the multilayer perceptron (MLP), or the “feed-forward” architecture [66]. If we consider a network with one or more input layers, we can define hidden layers as those used for intermediate computation and no direct connection to the input-output environment [66]. MLPs

typically consist of one or more input layers, a series of hidden layers, and an output layer, and a visual representation can be found in Figure 3.2 (c). Additionally, the term bias refers to a constant added to a neuron's weighted output, and it is used to promote learning when the weighted sum of a neuron is insufficient to pass onto a following neuron [45]. The non-linear nature of MLPs makes them excel in solving complex problems, opposed to single perceptrons, which were limited to simple binary classifications (0 or 1) [60].

3.2.2 Activation Functions

Activation functions are an essential component that introduces the non-linearity mentioned at the start of this chapter [45]. Neural networks need to be able to learn complex patterns and non-linear relationships in data. This means that rather than a straight-line relationship between input and output, different layers share characteristics in order to capture complicated characteristics in real-world data [66]. Essentially, activation functions receive an input in_i and, based on a threshold ϑ , they decide whether they should pass their information to other neurons or not according to their importance [65].

Several activation functions are commonly used and serve different properties based on data and problem structure. Following, the two activation functions used for this project – relu and softmax – are briefly analysed, and both are visualized in Figure 3.3.

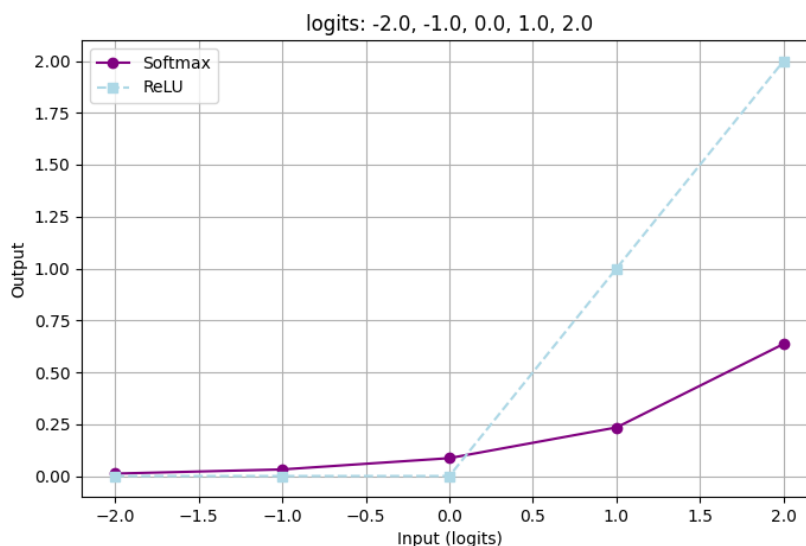


Fig 3.3. Visual representation of the ReLU and softmax activation functions.

- **Rectified Linear Unit (ReLU)** is widely considered a modern, cutting-edge activation function due to its simplicity and high performance [65]. Essentially, it only allows positive inputs to pass information to the next neurons. As seen in Equation (1), the function is very simple and easy to implement in a computer, since it only requires one computationally inexpensive condition [66].

$$f_{\text{ReLU}}(z) = \begin{cases} 1, & z > 0 \\ 0, & z < 0 \end{cases} \quad (1)$$

- **Softmax** is another commonly used activation function, and it can often be found in multi-class classification tasks. It is used to transform raw outputs of the neural network into a singular vector of probabilities, where each value represents a likelihood of an input belonging to a specific class, with the sum of all probabilities adding to 1 [67].

In a previous chapter (3.2.1), a multi-class problem in the form of digit classification from 0 to 9 was mentioned, and softmax is an ideal candidate for extracting probabilistic results. If we assume a well-trained neural network that takes a picture of the number 9 as an input, the vector produced by softmax would look something like Equation (2). In this case, the class with the maximum probability (0.9875) would be chosen for the final classification.

$$v = \begin{pmatrix} 0.0001 \\ 0.0003 \\ 0.0002 \\ 0.0005 \\ 0.0008 \\ 0.0012 \\ 0.0023 \\ 0.0015 \\ 0.0056 \\ 0.9875 \end{pmatrix} \quad (2) \quad f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^K e^{x_j}} \quad (3)$$

If we assume that:

x_i : the output of the previous layer in the network (logit) for the i -th class.

K : the total number of classes.

e^{x_i} : the exponential score of the logit.

$\sum_{j=1}^K e^{x_j}$: the sum of exponentials for all classes.

Then, the softmax equation can be defined as shown in Equation (3). The exponentials are used to amplify the differences between scores by enlarging large scores and shrinking the small ones. In addition, the sum is used to normalise the exponentials and ensures that all the probabilities add up to 1 [67]. The reader may return to the example of Figure 3.3 now that the formula has been analysed. The y-axis denotes the output of the softmax function, and it can be observed that the higher the input logit is, the higher the output probability.

3.2.3 Loss Functions and Optimizations

Loss functions can be defined as a way to inform a machine how close the combination of weights and biases is to finding the optimal solution to a problem [68]. While several loss functions have been proposed over the years, the ones we experimented with for this project, cross-entropy loss and mean squared error (MSE), will be briefly analysed. It should be noted that loss functions contain every weight and bias of a neural network, and calculating them requires a great number of computations depending on the size of the network [68].

- **Cross-Entropy Loss**, often referred to as Log Loss, is a loss function used to measure the difference between two probability distributions p and q , where p is the predicted and q is the true distribution [69]. In other words, it is used to minimize the error between the predicted and actual weights in a neural network, and a lower cross-entropy score denotes better performance.

There are two widely used cross-entropy formulas depending on the classification problem: binary cross-entropy (BCE) used for binary classification, and categorical cross-entropy (CCE) used for problems involving multiple classes [70].

If we consider:

y_i : the ground truth label for instance i (0 or 1).

\hat{p}_i : the predicted probability that the label is 1.

Then, the binary cross entropy (BSE) formula for a single sample can be defined as seen in Equation (4), where \log is used to penalize predictions far from the ground truth [70]. For example, a ground truth label of $y_i = 1$ and a predicted probability of $\hat{p}_i = 0.05$ would result in a significantly high loss due to the logarithmic scaling.

$$L(y_i, \hat{p}_i) = - [y_i \times \log(\hat{p}_i) + (1 - y_i) \times \log(1 - \hat{p}_i)] \quad (4)$$

Additionally, if we consider that C is the number of classes in a network, then the categorical cross-entropy (CCE) formula for a single sample may be defined as shown in Equation (5) [70].

$$L(y_i, \hat{p}_i) = - \sum_{j=1}^C y_{i,j} \times \log(\hat{p}_{i,j}) \quad (5)$$

- **Mean Squared Error (MSE)**, or L2 loss, is a widely used loss function that calculates the average of the squared differences between the predicted values and the ground truth [69, 70]. It is considered a smooth gradient-based optimization, albeit with a downside in its poor performance on datasets with outliers, meaning that they need to be handled accordingly in the data preprocessing stage [70].

If we consider:

θ : the model parameters.

n : the total amount of data.

y_i : the actual value for the i -th sample.

\hat{y}_i : the predicted value for the i -th sample.

Then, the mean squared error formula is defined as shown in Equation (6). Squaring the differences works similarly to the logarithmic error amplification introduced in Equations (4) and (5), though MSE may place more emphasis on mistakes when the error is too big. Additionally, just like BCE and CCE, MSE needs to be minimized to produce better results.

$$MSE(\theta) = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (6)$$

3.2.4 Backpropagation

Bryson and Ho (1969) were the pioneers who introduced what is considered the backbone of modern neural networks – backpropagation [71, 45]. This algorithm refers to the process of fine-tuning weights and biases in a neural network based on an error rate obtained in a previous iteration [71]. Despite the original concept dating back to 1969, Werbos laid its theoretical foundation in 1974, and it only began to gain popularity in 1986 when McClelland and Williams applied it to MLPs, showcasing its potential to solve complex problems [72, 73, 74].

In order to thoroughly understand the reason back propagation is necessary, it is important to introduce the concept of forward propagation first. Forward propagation refers to the process of inputs being passed through the network in order to generate an output [45]. The output is then compared to the ground truth through the use of a loss function (e.g., BSE or MSE), and a numerical representation of the loss is produced [45].

Back propagation uses the aforementioned error to calculate what are considered the “gradients” of the loss, meaning the representation of the amount a network weight contributed to the final loss [45]. Finally, the weights of every layer are adjusted through the gradients using a loss minimization function, and the scale of these adjustments is managed by the “learning rate” of the network [45,

75]. This reinforcement process continues for every pass through the training dataset (epoch), and the model's performance is gradually increased; often until an equilibrium is found [45]. An updated illustration of Figure 3.2 (c) with the inclusion of back propagation can be seen in Figure 3.4.

Throughout the back propagation process in neural networks, one needs to be cautious of two common training problems from occurring: the vanishing and the exploding gradient [76]. The vanishing gradient refers to the progressive shrinkage of gradients, which results in earlier layers receiving minimal to no updates during training [76]. On the other hand, the exploding gradient describes the opposite scenario where gradients receive progressively larger values, resulting in instability and ultimately causing the convergence to fail due to the entrapment in a local minima [76]. Two common ways to tackle those problems can be found in the regularization techniques section analysed in section 3.4.2.

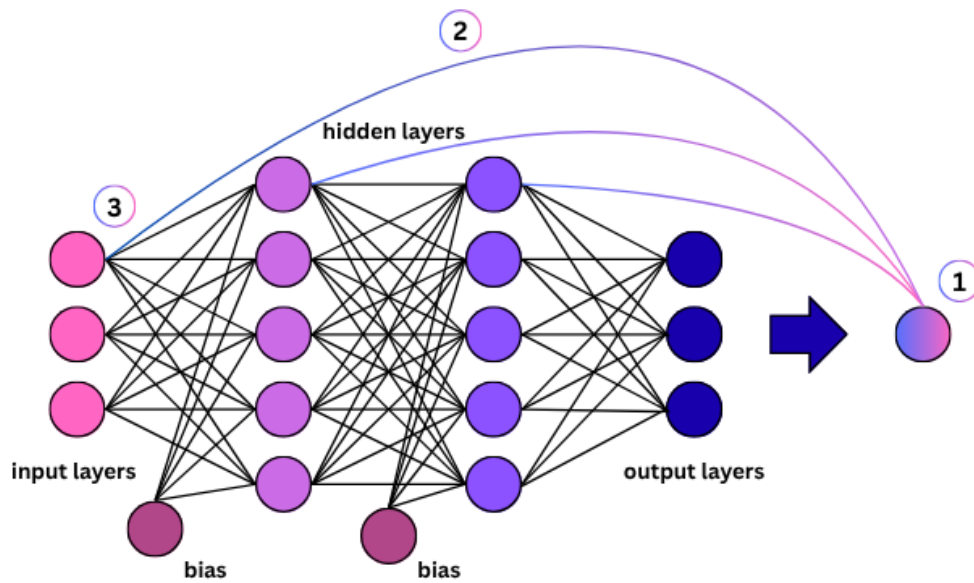


Fig. 3.4. The back propagation algorithm where:
(1) the error is calculated, (2) the error is sent back to each neuron, and
(3) the gradient of error is computed with respect to each individual weight.

3.3 Computer Vision

Enabling computers to perceive and understand the real world is a difficult challenge in artificial intelligence, commonly referred to as computer vision (CV) [77]. Although its applications are countless, one of the most impactful research areas is optical character recognition, which enables machines to read and interpret written text [78].

Considering the main topic of this thesis is OCR applied to spelling and writing disorder detection, it is essential to examine the deep learning architectures that enable robust and accurate handwritten text analysis. Among these, convolutional neural networks (CNNs) and transformer-based models have grown to become fundamental components in modern OCR systems and feature extraction. This section provides an overview of the relevance of these architectures, as well as how they function and what they consist of.

3.3.1 CNNs

Influenced by the backpropagation algorithm, what is now considered the backbone of computer vision – convolutional neural networks – was introduced in 1989 by Yann LeCun et al. [79]. Based on the convolution operation on a matrix referred to as a kernel, this architecture excels in extracting complicated features from images by generating feature maps [5]. These maps provide a numerical representation of patterns such as edges and textures, and the accumulation of multiple feature maps across convolutional layers forms a convolutional neural network [5].

In order to understand how convolutional neural networks process data, it is first necessary to introduce two important image representation concepts – grayscale and RGB image formats. Grayscale images are represented by a set of pixels depending on their resolution, and each pixel contains a value from 0 to 255 [80]. Coloured images following the “red green blue” (RGB) format comprise three channels representing the intensity of each respective colour on each pixel [80]. A pixel-level visualization of both the grayscale and RGB formats can be seen in Figure 3.5.

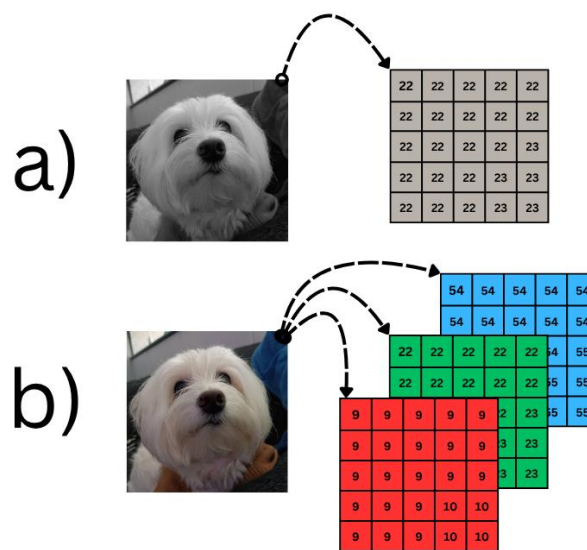


Fig. 3.5. Visualization of the upper-right 5x5 pixel region extracted from a 2698x2698 image shown in: (a) grayscale, and (b) RGB format.

The aforementioned convolution operation is known as a kernel-based transformation. Essentially, a kernel refers to a small square matrix that constantly slides over image pixels in order to extract information from them. This is calculated through a dot product operation between its own values and the pixel values it overlaps with at each position [5]. This results in a “feature map” that contains important visual characteristics, and depending on the parameters, may be smaller than the input [5].

Unlike traditional non-linear approaches – also referred to as artificial neural networks – CNNs allow networks to represent an image in a 3D format that includes height, width, and channels [5]. For grayscale images, the number of channels is one, while for RGB it is represented by three – one for each colour intensity [80]. If we assume two distinct images containing a zebra and a panda respectively, a 1D representation of all their information would ultimately lead to uncertainty in key features, such as colour placement and texture. The 3D format enables networks to distinguish features and visual patterns – such as the zebra’s stripes and the panda’s fur – through a thorough pixel analysis across every dimension and colour channel [81]. Additionally, each convolutional layer applies multiple kernels that extract various features from localized regions of the image, and as they accumulate, the areas of interest grow, allowing deeper layers to capture more abstract patterns.

It should be noted that kernel sizes may vary, and their application is dependent on the image's resolution [81]. For instance, a kernel of 3x3 pixels is applicable to a 7x7 image, though one of 4x4 is not. The general formula for applying kernels is shown in Equation (7), where stride refers to the number of pixels the kernel moves at each step, and padding is the number of pixels added around the border of the image [82].

$$Output\ Size = \frac{Input\ Size + 2 \times Padding - Kernel\ Size}{Stride} \quad (7)$$

As the number of layers begins to grow, the computational cost as well as the risk of overfitting rise [82]. Particularly, complicated models are prone to learning noise or meaningless patterns. To help control this, a mechanism referred to as pooling is commonly used after convolutional layers [82, 87]. Its goal is to reduce the dimensionality of feature maps – also known as down-sampling – while maintaining key characteristics and patterns. Two methods are frequently used to achieve this: average and max pooling, both of which preserve important information while discarding irrelevant details [87].

- **Average pooling** refers to the process of extracting the average from each position of a kernel [83], as shown in Figure 3.6.

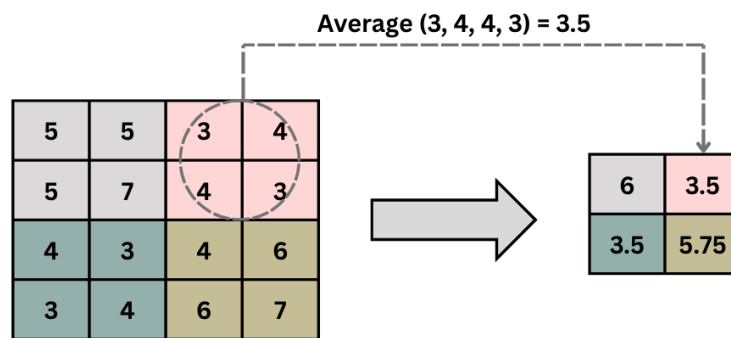


Fig. 3.6. Average pooling example for a 2x2 kernel with a stride of 2.

- **Max pooling** involves selecting the maximum value within a group of features [84], as illustrated in Figure 3.7.

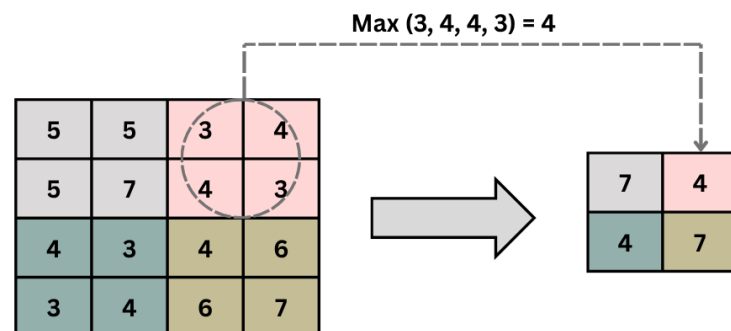


Fig. 3.7. Max pooling example for a 2x2 kernel with a stride of 2.

At the end of the convolutional and pooling layers, the last feature maps generated need to be transformed in a way that can be later utilized for the final classification. This is achieved by fully connected layers, also referred to as dense layers, whose goal is to convert a given input into a 1D vector, which will then be passed into an activation function [87]. In essence, every neuron in a fully

connected layer is connected to each one of the previous layers, forming a “dense” network of connections [5]. A visualization of a typical CNN architecture can be found in Figure 3.8.

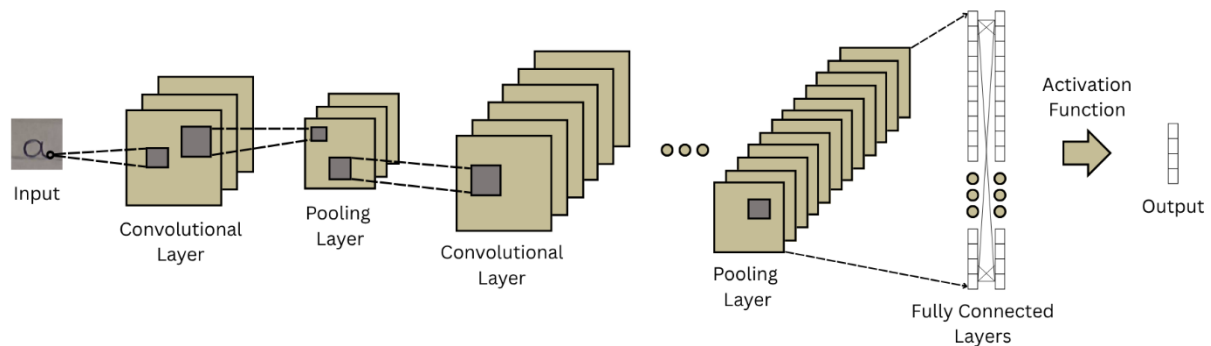


Fig. 3.8. Illustration of a convolutional neural network.

3.3.2 Transformers

While architectures such as recurrent neural networks (RNNs) and long short-term memory (LSTMs) have historically been used for sequence modeling in OCR tasks, a new architecture referred to as transformers has gained attention for its ability to capture long-range dependencies and contextual relationships [85]. Originally introduced by Vaswani et al. in 2017, this architecture uses a “self-attention” mechanism that allows models to calculate the importance of different output elements dynamically [85]. This importance is represented by weights, and a greater weight denotes greater importance [85]. Additionally, unlike RNNs and LSTMs, which use sequential processing, transformers enable parallel processing – allowing the model to process entire sequences at once instead of little by little. This parallelization not only makes the training process faster, but also enables the model to understand more ambiguous patterns – something that is essential for OCR, where noise and unnecessary information are present [86].

Transformer-based models utilize the aforementioned self-attention mechanism through multi-head attention, allowing them to focus on multiple parts of an input sequence simultaneously [85]. In essence, the input is split into multiple heads which independently learn to focus on different patterns and data relationships – each capturing different contextual cues [85]. Although it has showcased exceptional results in tasks such as machine translation and text generation [88, 89], the field of computer vision has also flourished due to its ability to extract complex features, enabling more accurate image segmentation and object identification [90].

3.4 Training Deep Learning Models

To effectively train a deep learning model, it is essential to utilize a custom dataset specifically designed to address the target problem. Just as humans acquire knowledge through living experiences in the form of observation and mistakes, deep learning models heavily rely on collections of data in the form of text or images to develop a comprehensive understanding of patterns and connections. This chapter first goes through an overview of a typical dataset preparation process, as well as the concept of data augmentation. Then, the two main data-related errors during the training process, overfitting and underfitting, along with techniques to avoid them are presented.

3.4.1 Dataset Preparation and Augmentations

A typical data preparation process may comprise four distinct steps, depending on the type of data. Specifically, they can be characterized as data collection, preprocessing, splitting, and formatting.

- **Data collection** can be described as the process of gathering raw data from various sources. This data can be found on websites or created by users themselves to tailor to specific tasks. The most commonly used dataset websites in the machine learning community seem to be Kaggle [91], UCI [92], Google Dataset Search [93], and the US government's open data portal [94] – each providing thousands of free-to-use datasets with a wide range of applications.
- **Preprocessing** is the practice of cleaning, reformatting, and normalizing data in order to improve usability and quality [96, 98]. It is considered a time-consuming yet necessary task that 60% of data scientists spend their time on, according to a survey published by Forbes [95]. Depending on the structure of the dataset, preprocessing could include the handling of outliers and missing values, inconsistencies, noise reduction, and class imbalances [96].

Additionally, a concept referred to as “data augmentation” is one worth setting our attention to before proceeding to the following chapter. Data augmentation refers to the process of applying one or more transformations to existing data in order to generate more samples [97]. It is often used on images and time series data (e.g., dates, temperatures) and excels in settings where data is limited yet necessary for a model's training process [97]. Figure 3.9 illustrates ten different commonly used image transformations – few of which were utilized in the creation of this thesis' proposed OCR model.

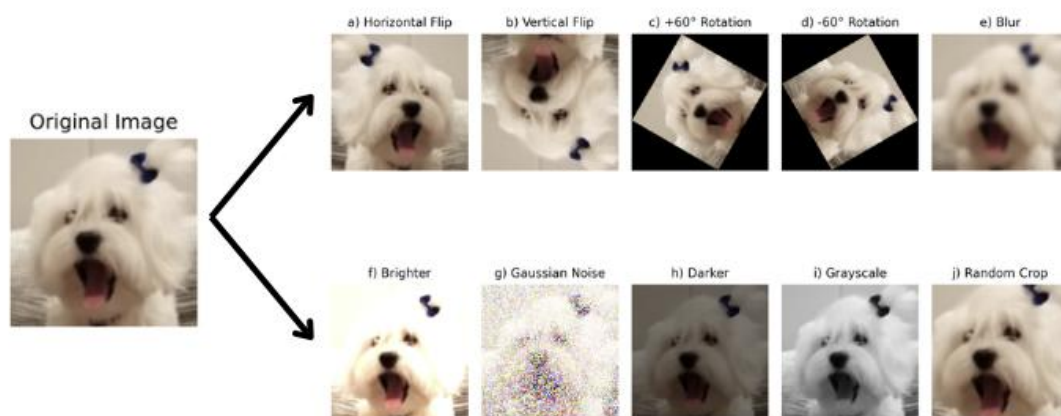


Fig 3.9. Image augmentation examples.

- **Splitting** refers to the process of dividing a given dataset into two separate ones: one for training and one for testing. Specifically, the training dataset is used by the neural network to learn key patterns, and the validation dataset is used to evaluate how well the model performs on unseen data [96]. Two commonly used dataset splits for testing and validation are 80-20 and 70-30, where the numbers represent the percentage allocated to the respective subset [99, 100]. Those percentages are selected to help avoid the phenomenon of learning the test data too much, or struggling to learn them, and thus being unable to generalise to unseen instances [99].
- **Formatting** is an important step that involves the transformation of accumulated raw data into a machine-readable format tailored to the framework being utilized [96, 98]. If we consider a “dog” dataset with a “Breed” column, there may be hundreds of breed names in

the form of strings (e.g., Maltese, Yorkshire terrier). Machine learning models typically require numerical input, and thus each value of the aforementioned dataset column may need to undergo an encoding process that transforms strings to a numerical representation.

3.4.2 Overfitting, Underfitting, and Regularization

Two common problems are prone to occurring in machine learning models and deep neural networks: overfitting and underfitting [87]. Overfitting refers to a model's overperformance in the training dataset and underwhelming accuracy in the validation/testing dataset, where the model struggles to generalise on unseen data [96, 101]. On the other hand, underfitting can be found in settings where a model's architecture is too simple and, as a result, the model will generalise; though both the training and validation performance will be low [96, 101]. An illustration of overfitting and underfitting examples can be found in Figure 3.10, where the purple overfitted lines overlap with the training points and thus stray from potential future data. In contrast, training or data-related issues have caused the blue underfitting line to deviate greatly from the optimal one.

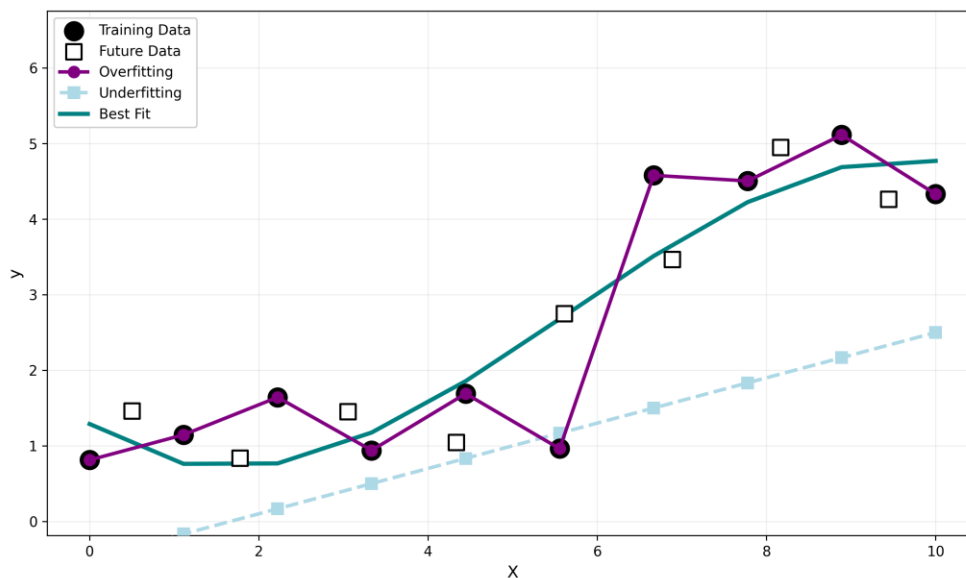


Figure 3.10. Overfitting and underfitting example.

Over the years, several regularization techniques have been proposed and are widely used to tackle the overfitting problem in deep neural networks [102]. Two commonly found techniques include dropout layers and batch normalization, and we believe they are worth presenting in order to provide a greater understanding of why we used them in the model presented in the following chapter.

- **Dropout layers** are an elegant approach to solving the overfitting problem, and their general idea lies in randomly deactivating neurons temporarily from a neural network during the training process [103]. This technique allows the network to combine different architectures and discover the ideal one while excluding potentially biased neurons with a probability p [103].
- **Batch normalization** is a regularization technique used to improve the training process of a neural network by allowing it to reach an equilibrium state faster and improving its stability [104]. In order to fully understand how it works, typical normalization in neural networks needs to be mentioned first.

Normalization refers to the process of scaling data to ensure that every input feature in a neural network contributes to the learning process equally [104]. A frequently used scaling technique is one where input features are transformed to hold a mean of zero and a standard deviation of one [87]. This process is independent of the model and is conducted before training as part of preprocessing [87].

Batch normalization is a neural network layer placed between two connected hidden layers in order to normalise the output of the first one before it gets passed onto the next [104]. It is a technique used during the training process for small batches of data to solve the “internal covariate shift” problem, where the distribution of layer inputs varies as the model keeps learning [104]. Additionally, it allows the network to utilize higher learning rates while mitigating common neural network problems such as the exploding or vanishing gradient mentioned in Chapter 3.2.4 [104].

3.5 Deep Learning Frameworks and Tools

Over the years, several frameworks and tools have been developed in order to assist machine learning and the creation of deep neural networks. This chapter goes through a brief showcase of the two most commonly used machine learning frameworks, as well as what was selected for this project. Following, a few fundamental Python libraries used throughout the entire project are mentioned.

3.5.1 TensorFlow and PyTorch

As of 2025, there are two main frameworks that are widely used to create neural networks: TensorFlow and PyTorch. Both frameworks are commonly paired with the Python programming language, which, as of July 2025, holds a TIOBE index of 26.98% making it the current most popular programming language [105].

- **TensorFlow** was created by Google in 2015 with the goal of providing the ability to execute machine learning algorithms on various platforms [106]. Its TensorFlow 2.0 update in 2019 brought various quality-of-life improvements, such as an easier data loading feature and more “Pythonic” network definitions, and is now considered a very beginner-friendly framework [106, 107].
- **PyTorch** was created by Meta’s AI Research lab (FAIR) in 2016 as an open-source machine learning framework [108]. Its core characteristics lie in its “Pythonic” nature and high flexibility, making it especially prevalent in research in academia. Additionally, a comparative analysis published by Leapcell in July 2025 showcases that around 85% of deep learning research papers use PyTorch for experiments and AI model creation [109]. This shows a sharp increase since the 2021 research, which showcased that 70% of authors on the website “Papers with Code” used PyTorch to implement their machine learning works [110].

For this project, the deep learning model proposed was developed with PyTorch due to its customizable and reliable nature.

3.5.2 Libraries

Although several Python libraries were utilized throughout the entire project, the main ones were OpenCV, Matplotlib, and those incorporated in PyTorch, though several others such as numpy and skimage were also used for image calculations.

OpenCV [111] is a commonly used library that handles image processing. It provides a wide range of built-in transforms with over 2500 implemented algorithms, such as grayscale conversions and binarizations, and it was the backbone of almost every text-related task we handled. In particular, image preprocessing techniques such as automatic thresholding, as well as the entire notebook line, word, and letter detection process that accompanied the custom-made OCR model was enabled by its extensive range of tools.

In addition, Matplotlib [112] was frequently used to produce most of the example plots showcased throughout every chapter of this thesis, as it provides great customization tools for data visualization. For example, Figure 3.10 showcases a custom regression algorithm that was implemented and plotted through Matplotlib.

Finally, several PyTorch [108] libraries were used to create the proposed deep learning model, with the most notable ones being “transforms” and “DataLoader” – both of which were used to handle our custom datasets. A detailed overview of them can be found in the following chapter.

CHAPTER 4

Writing and Spelling Disorder Detection

4.1 Introduction to OCR - Need for Data

Optical Character Recognition (OCR) can be considered the process of extracting any text from images into a machine-readable format [147]. Its concept dates back to the 1910s where Emanuel Goldberg designed a machine capable of reading characters and converting them into telegraph code [113], and it has seen a great rise since then – especially with the recent introduction of deep learning [5]. The procedure typically consists of 3 main stages, and a low-level pipeline can be found in Figure 4.1.

The first step is image preprocessing. Its main role is to clean up the input image and make it more legible by utilizing transforms such as grayscale conversion, noise removal, binarization and resizing. The second step, and probably most complex one, is segmentation. Segmentation is a process that divides an image into meaningful regions and typically isolates rows of text, blocks of words, and single letters. Particularly, single letter segmentation in handwritten text can be considered an exceptionally difficult task due to the tendency of two or more characters being written in one stroke, or them being too close to each other – ultimately making it hard for the computer to understand where it would be best to split them. The final step is to utilize a robust model, designed for handwritten text and most importantly, character recognition.

For this project, various open-source OCR models that are widely used today – such as Google’s Tesseract [114] – were tested, but the results were unsatisfactory. Therefore, creating a custom-made model from scratch, tailored to single Greek character recognition, would be most beneficial. Thus, it was necessary to solve not only the word and character segmentation problem, but also the character recognition problem. This also means that the text predicted by the model needs to be close to 100% accurate in order to make correct spelling classifications for each word used in the spelling disorder detection stage.

However, this also created a need for handwritten Greek letters to be used as training data for the model. We searched for data in websites like Kaggle and Hugging Face, but since there were no major public datasets for Greek letters available, we decided to create our own and thus asked a group of volunteers to write us Greek letters that could be used for the training process of the model. Additionally, the National Centre for Scientific Research Demokritos provided us a small dataset of Greek passages from the ICDAR2012 Writer Identification competition [115]. The dataset included 2 short passages from 100 different volunteers, each containing around 35 characters, and was used to boost our original dataset.

This chapter outlines the procedures for data collection, preprocessing, and processing. Following this, it introduces the proposed model, detailing the different approaches and algorithms employed in order to complete the OCR process and detect potential disorders in handwritten text.

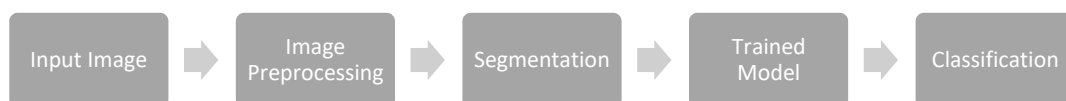


Fig. 4.1. Low-level OCR process.

4.2 Data Collection

For the initial dataset, approximately 100 individuals aged from 18 to 50 were asked to write the Greek alphabet in both lowercase and uppercase, along with all tonos and dialytic variants, following a provided format (Figure 4.2). Each participant completed a consent form including their name, signature, and date, explicitly stating their permission for their data to be used in this project.

The data collection process was conducted both in person and online. Participants who contributed in person used a pen to complete the form, which was subsequently photographed. Those who participated remotely submitted a digitally signed consent form along with photographs of their handwritten samples.

Each character was separated by a large horizontal space and a vertical notebook line, making it easier to crop individual characters that exceed the bottom notebook line, such as ρ (ro) and ξ (ksi). This format avoided interference from adjacent lines and prevented character overlap, thereby improving segmentation accuracy. Two samples of the raw data are shown in Figure 4.2. Additionally, a sample of the ICDAR2012 Writer Identification competition dataset is shown in Figure 4.3.

To enrich the dataset, participants were also asked to write selected double-character combinations that are frequently produced in a single stroke, such as αι, γγ, γκ, ει and ου. These samples were intended to provide insight into how the same characters may be written differently when other letters are present.

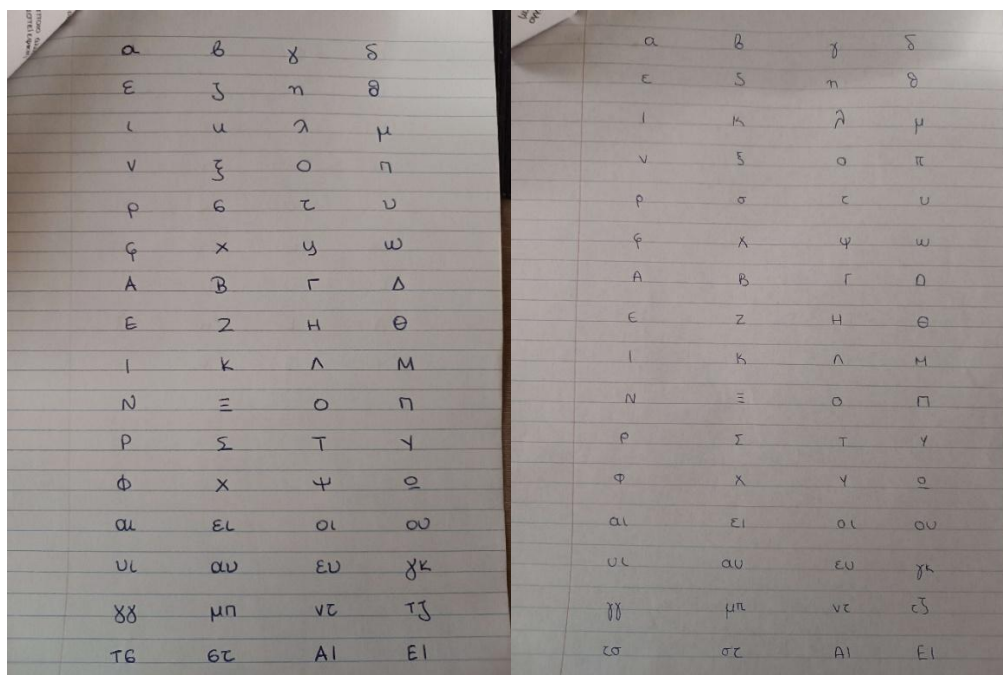


Fig. 4.2. Sample of the raw data received from the participants.

Η λευτεριά δεν έχω σκατό. Μίτσε βρίσκεσαι εμείς γης ετούτη -
οι γης ετούτη βρίσκεσαι ποτόχα ο αγώνας για τη λευτεριά.
Αγωνισόμαστε για τα αίματά, και γι' αυτό ο σύντροφός
έγινε να είναι ζώο.

Fig. 4.3. Sample of the ICDAR2012 Writer Identification competition dataset.

4.3 Dataset Pre-Processing

The next step following data collection was to extract and process each character individually in order to teach the OCR model how to recognize Greek characters and their potential variations. Specifically, preprocessing involved character cropping, folder separation, image augmentations, class renaming, and train-test splits.

4.3.1 Dataset Format

Upon having cropped each letter individually, various folders were created for letters that represent classes, and would ultimately be predicted by the OCR model afterwards. The format used to iterate through every letter was created as shown in Figure 4.3, where each letter folder contained around 500 image files of its corresponding handwritten character.

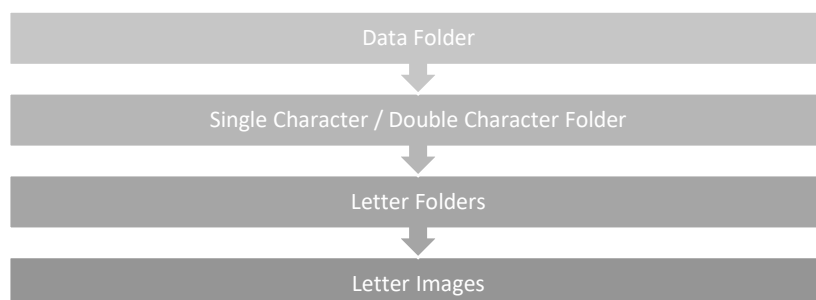


Fig. 4.3. Data file pipeline used for this project.

Character image cropping was accomplished in three different ways to add diversity, which will accompany the dynamic augmentations implemented later and help boost the training performance. Specifically, we experimented with:

- i. Tightly cropping both the x and y axes around each letter.
- ii. Tightly cropping the x-axis and including both horizontal notebook lines above and below the letter.
- iii. Tightly cropping the x-axis and cropping the upper y-axis to a point above the letter and below the notebook line above the character.

Additionally, both approaches we experimented with included the bottom horizontal notebook lines in order to give our model more context about character placement, especially when it comes to characters that exceed that bottom line, and thus making the training process more efficient. Some samples of the letters we cropped and passed onto the final dataset can be found in Figure 4.4.

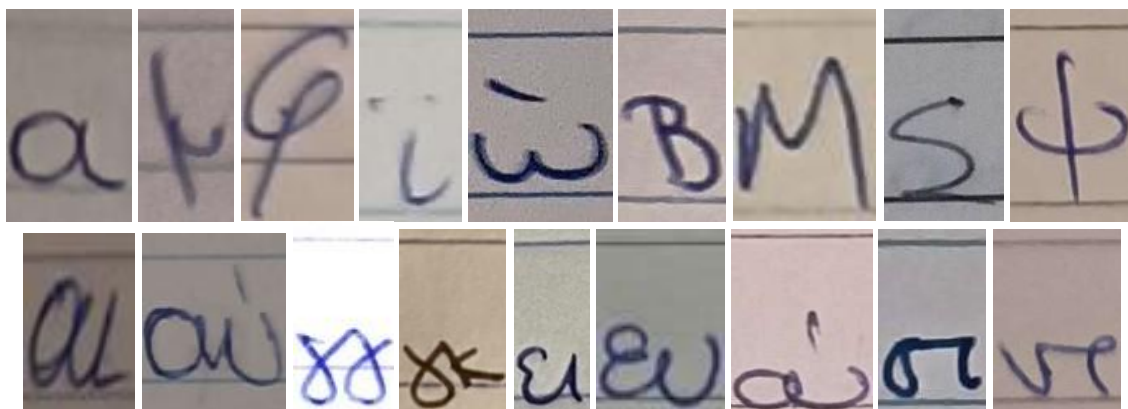


Fig. 4.4. Cropped images part of the final dataset.

4.3.2 Augmentations and Data Loading

For a deep learning OCR model used for a project as big as this one, having a mere ~500 images for each class would be insufficient and limit the training process greatly. To solve this problem, we created augmented versions of each image, both statically to pass the dataset to the model, and dynamically as the model was training.

After having collected all characters and labeled them in their corresponding folders, we proceeded to create a script that loads the data with its respective classes and creates augmentations for each image. Specifically, we created 1,500 augmentations of every letter input with the following attributes to make up for the need for more data and variations:

- i. Random $(-7, 7)$ degree rotation, accounting for different angles at which pictures can be taken.
- ii. Random $(1, 1.5)$ contrast added, accounting for the different lighting in which pictures can be taken in.
- iii. Random y-axis extension, accounting for various ways that a letter could be captured in the single-letter segmentation process.

In addition to the rotation augmentation, we made sure that no black gaps were left on the new augmented images by extending the nearest edge pixels outward – keeping the background consistent and essentially removing the noise in the corners. This was done by utilizing OpenCV's BORDER_REPLICATE function, and it worked exceptionally well in our case, where the rotations were of relatively small size.

Because of these augmentations, every letter folder in the final dataset contained around 600,000 images, making it ideal to feed to the model. A few examples of the static augmentations that were applied can be found in Figure 4.5.

It should be noted that we also applied a large variety of other dynamic augmentations while the model was training to reinforce the learning process and ultimately increase the model's accuracy by ~13%. Further analysis of the dynamic segmentations we used can be found in section 4.4.2.

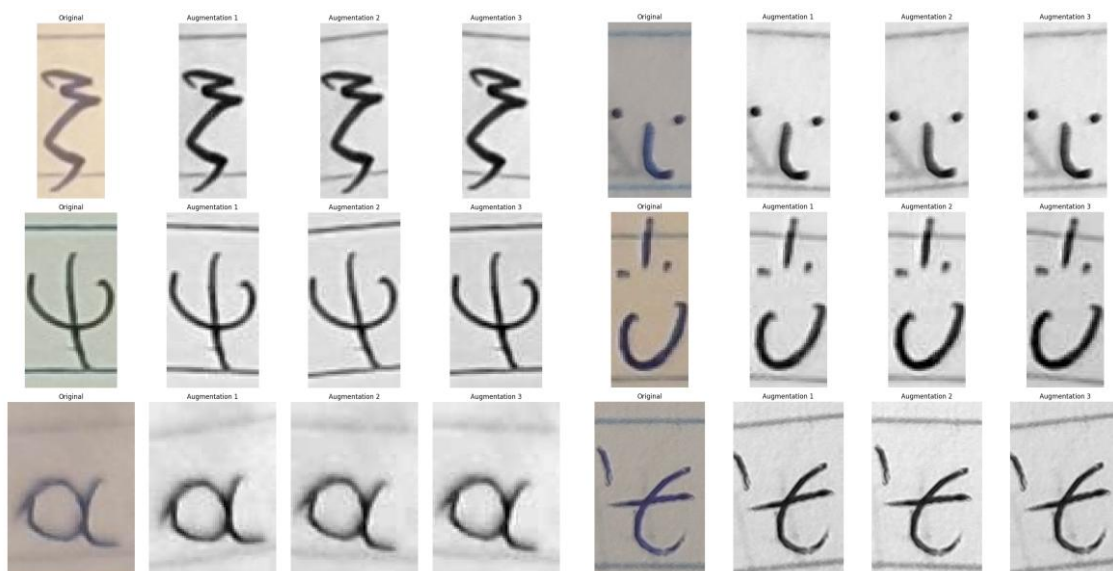


Fig. 4.5. Original Images and their respective static augmentations.

4.3.3 Transforms

Upon having completed the static augmentation creation and image loading, we proceeded to apply a series of transformations to each image of the dataset so as to maximize stability and efficiency during the training process. Specifically, we:

- Resized all input images to a fixed 512x78 pixel size, where 512 is the height and 78 is the width of every image respectively. We experimented with a variety of pixel sizes, but found that this was the one that captured every necessary data component while keeping the size relatively small – making the training process shorter and more efficient.
- Converted the images into PyTorch tensors that also scale each image's pixel values from [0, 255] to [0.0, 1.0].
- Normalized the image tensors with a mean of 0.5 and a standard deviation of 0.5 for each channel. Essentially, this centers the data around zero and stretches it into the range [-1, 1] in order to promote better learning dynamics and ultimately allow the model to train faster and more stably.

4.3.4 Train-Test Data Split

In order to pass the dataset to the OCR model for training, there is a need to split the data into training and testing data. To do this, we iterated through each data folder individually, and performed a dynamic split to account for potential disorder in the number of data that each folder contains. After experimenting with the train-split ratio, we found that 0.2 (80% training - 20% testing) worked best for our model and dataset.

It should be mentioned that every augmentation of each image was part of either the training or testing dataset, which do not overlap with each other, something that would make the training process problematic as the model would include the same data in both the training and testing datasets otherwise. In other words, the training dataset is learning new data, as opposed to classifying data that it has already seen and learned.

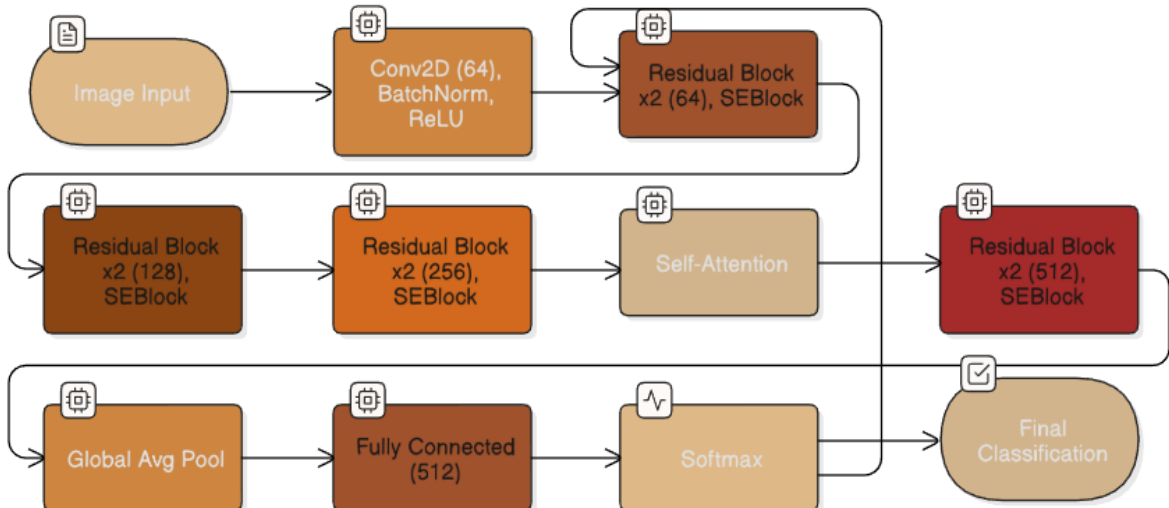


Fig. 4.6. The proposed OCR model's architecture.

4.4 Model Architecture

Having completed every necessary data preparation step, the next task was to design and train the handwritten text recognition model. We experimented with a variety of deep convolutional neural network architectures; however, we found that an upgraded Res-Net architecture worked best for this problem. Essentially, we borrowed a Res-Net-style architecture [116], but applied some upgrades tailored for handwriting recognition that include Squeeze-and-Excitation blocks (SEBlocks), Multi-Head Attention layers, and the addition of a final residual block after the attention-enhanced features are processed, followed by global average pooling and a fully connected layer to output the class logits. The final architecture can be seen in Figure 4.6. The following sections analyze the model's entire training pipeline, its architecture, and the dynamic augmentations applied throughout the model's training process.

4.4.1 Full Model Pipeline

After the augmentations have been created and the dataset loading and train-test split are complete, the model is set to begin its training process. In the training loop, the following specifications and techniques were used in order to control overfitting and instability.

- **Loss Function**

CrossEntropyLoss loss function combined with Class Weighting and Label Smoothing.

- *CrossEntropyLoss* measures the difference between the probabilities that the model predicted and the actual class label [69], making it ideal for the proposed multi-class classification-based model nearing 70 classes.
- *Class Weighting* balances the learning process for classes with fewer samples by giving them higher weights [117]. It was used to avoid biasing towards frequent characters, and we found it performed well, since several characters such as “ç” were underrepresented.
- *Label Smoothing* improves generalization and reduces overfitting. It essentially “softens” the class labels, preventing the model from being over-confident [118].

- **Optimizer**

AdamW optimizer combined with a ReduceLROnPlateau scheduler.

- *AdamW* [120] combines the Adam optimizer [119] with a decoupled weight decay, and it was used to achieve better generalization and stable convergence. Both Adam and AdamW were tested, and we found that AdamW converged better, likely due to the large dataset size.
- *ReduceLROnPlateau* controls the validation accuracy by reducing the model's learning rate by a 0.2 factor if the accuracy stops increasing for 3 epochs [121]. Additionally, a minimum learning rate of 1e-4 was set to avoid it becoming too low.

- **Gradient Control**

Gradient clipping was used to limit the gradients across all model parameters to a maximum of 1.0 [122]. Combined with the AdamW optimizer, it handles gradients properly and prevents the exploding gradient problem from occurring. Essentially, there is no fear of the gradients computed through the backpropagation process becoming too large and leading to unstable updates to the model's weights.

- **Early Stopping**

An early stopping mechanism was used to complete the training process when the validation set's accuracy starts degrading or does not improve after a set number of epochs [123]. This was done as a countermeasure to the model memorizing the training data too well, including the noise in them, and ultimately resulting in overfitting and poor generalization. Additionally, avoiding overtraining meant a more resource-efficient training process.

- **Accuracy**

A custom accuracy metric was created to “soften” the penalty for misclassifying uppercase and lowercase letters that are often written the same way on handwritten text. As seen in equation (1), it consists of a division between the sum of correct classifications over the total amount of classifications, with a lighter penalty for misclassifications of characters included in Table 4.1. The letter check was done by first mapping the uppercase letters to their respective lowercase alternative using a bidirectional dictionary (e.g., O to o and o to O) and then creating a custom CaseAwareLoss class, which was used to update the accuracy after each iteration.

If we assume:

N : the total number of samples.

y_i : the true label for the i -th sample.

\hat{y}_i : the predicted label for the i -th sample.

δ_{y_i, \hat{y}_i} : 1 if the prediction is correct, 0 otherwise.

$S(y_i)$: the pair of the label similar to y_i from the bidirectional dictionary.

$\delta_{\hat{y}_i \in S(y_i)}$: 1 if the prediction is similar to the true label, 0 otherwise.

α : the lighter reward for similar uppercase/lowercase predictions, where $0 < \alpha < 1$.

Then the accuracy function is denoted as:

$$\text{acc} = \frac{1}{N} \sum_{i=1}^N [\delta_{y_i, \hat{y}_i} + \alpha \cdot \delta_{\hat{y}_i \in S(y_i)}] \quad (6)$$

- **Confusion Matrix**

After the training process is completed, a function that plots a confusion matrix is called in order to visualize the characters misclassified by the model. The final confusion matrix can be found in Figure 4.7. Essentially, a “perfect” confusion matrix can be considered one that matches each letter of the x-axis (true letter) to the exact same letter of the y-axis (letter predicted by the model), creating a diagonal line of deep blue.

A lot of Greek letters can have their lowercase and uppercase variants written in similar styles, making it almost impossible to distinguish them unless there is context included – even for humans. A list of similar letter pairs can be found in Table 4.1. In addition, since our model reads characters one by one rather than entire words, it would be nearly impossible to make such distinctions. A result of this problem could be that words read by our model include both uppercase and lowercase letters, making dictionary searches a difficult task (e.g. ελΙΚόπτερΟ instead of ελικόπτερο). However, we found that converting the letters in that list into lowercase when read by the OCR model was a quick and elegant fix that would ultimately increase performance and accuracy and would not interfere much with the uppercase-lowercase inconsistency analysis in the case of writing disorder detection.

E – ε	Θ – θ	I – ι	K – κ	O – ο
Π – π	P – ρ	T – τ	X – χ	Ψ – ψ

Table 4.1. Common Greek letters of which their handwritten uppercase and lowercase variants look similar.

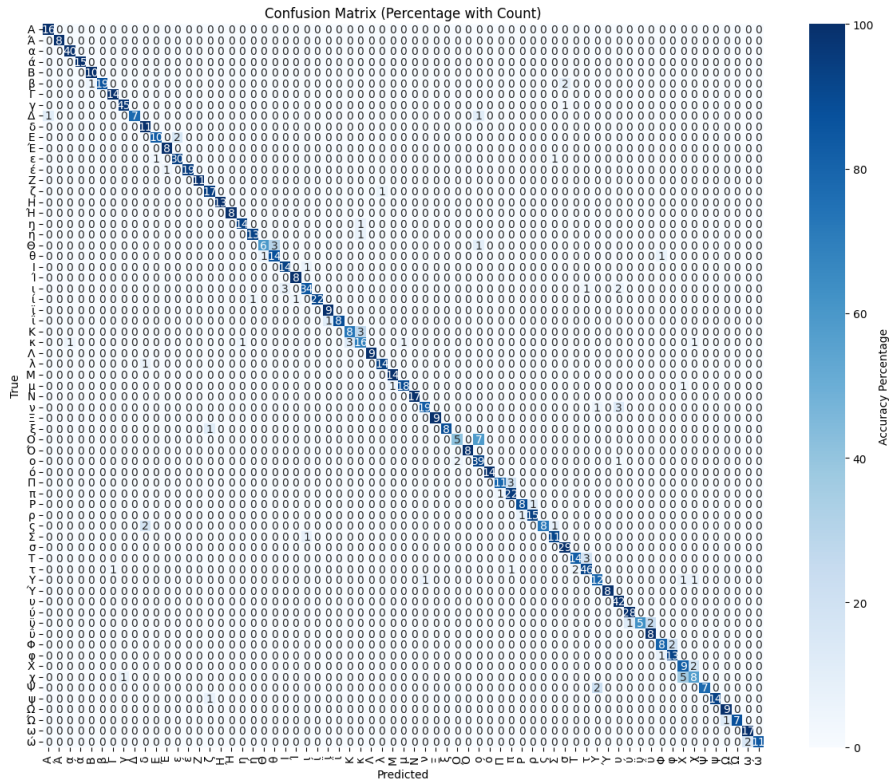


Fig. 4.7. The proposed model's Confusion Matrix.

4.4.2 Dynamic Augmentations

Dynamic augmentations are an essential element when it comes to building a robust model that focuses on important characteristics and features. If we were to only use the static augmentations mentioned in section 4.3.2, the model would suffer from learning pure noise after a specific point, hindering its classification ability greatly because of poor training. This occurs since the augmented images all consist of the same characteristics of the respective original image, including noise, resulting in the model focusing on said characteristics and overfitting to specific character shapes or line styles.

Each image of the current batch that the model is processing has a 50% chance of being transformed by each of the following techniques. They were ultimately designed to add variation by emulating realistic imperfections that can be found in actual real-world writing and help the model focus on important feature extraction without changing the original image too much. Examples and comparisons of each augmentation can be found in Figure 4.8, and the transforms applied are as follows:

- Random Affine**
 Mimics slight rotations, shifts, and shears to reinforce the static augmentations. The rotations were set to ± 7 degrees, the translation to $\pm 5\%$, the shear to $\pm 5\%$ and the scaling between 95% and 105%. This augmentation in particular was found to be very useful since it formatted the data in a way that would look similar to a realistic word segmentation. For example, a small part of the letter α is cut off in the 4th row of Figure 4.7 – though the letter remains distinguishable.
- Random Perspective**
 Simulates a warped or skewed page or lens distortion. The distortion scale was set to 0.3% and the probability of application was set to 40%.

- **Gaussian Blur**
Mimics blur from poor resolution, motion, or camera shake.
The kernel size was set to 3, and the sigma range from 0.1 to 1.0.
- **Color Jitter & Gaussian Noise**
Combined both Color Jittering and Gaussian Noise through a lambda function in order to make contrast adjustments and stimulate scanner noise or pen smudges.

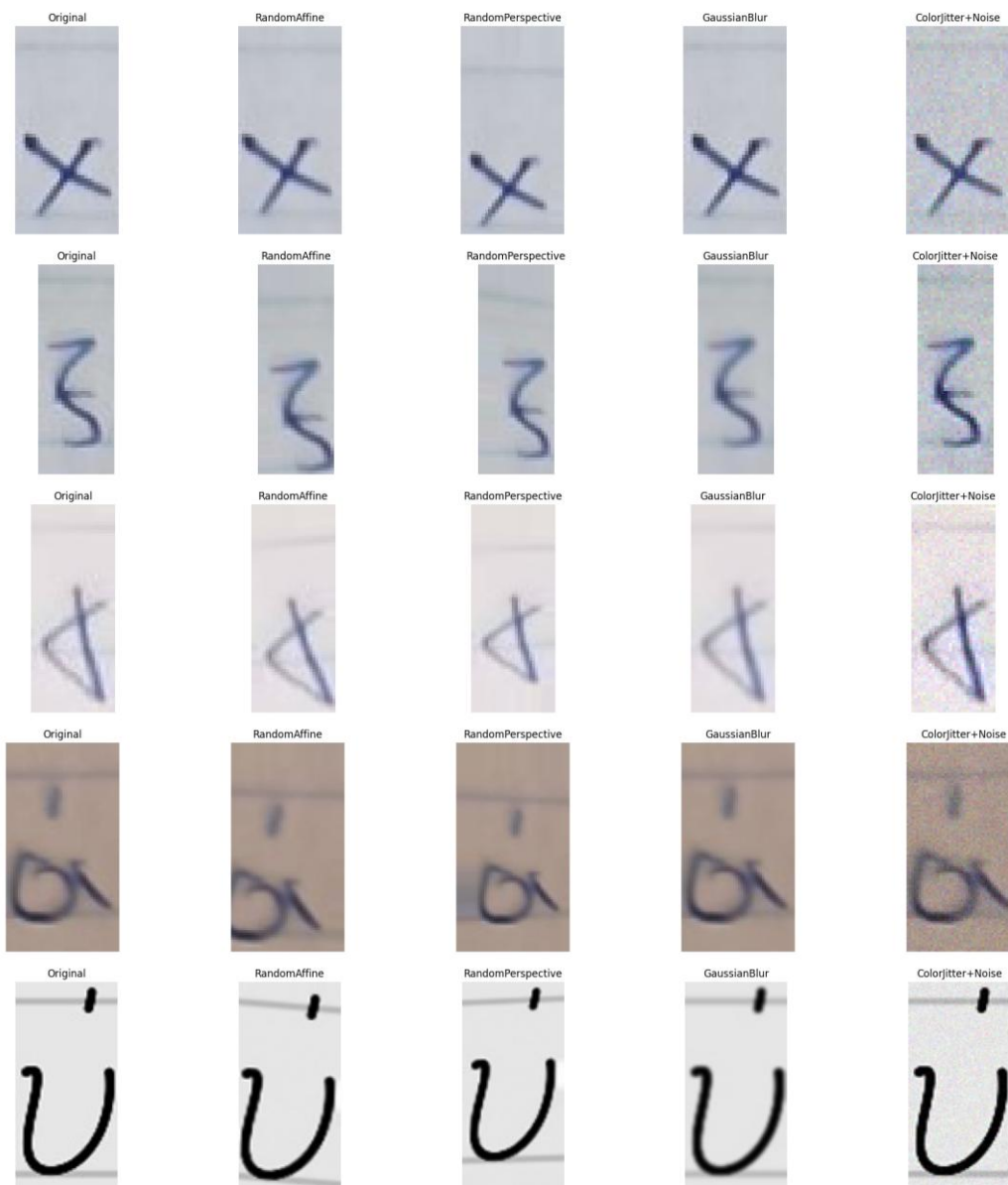


Fig. 4.8. Dynamic augmentations. Left to right: Original, Random Affine, Random Perspective, Gaussian Blur and Color Jitter + Gaussian Noise.

4.4.3 Model Architecture Analysis

The model's architecture is essentially split into five large components and was heavily inspired by Res-Net, though tweaked and upgraded to form a modern, robust, and accurate handwritten text classification model. Specifically, the model's components can be described as follows:

- **1. Convolutional Layer, Batch Normalization, and ReLU**

Used for the initial feature extraction of the input images.

- The Conv2D layer extracts low-level features such as edges and texture, taking the 1x512x78 (1 being the grayscale colour channel) input image and creating 64 output channels. In addition to this, a stride of 2 is applied, reducing the image's size to half and passing it to the following layers in the form of 64x256x39.
- BatchNorm2D is used to normalize activations and improve gradient flow [104].
- ReLU is used to introduce non-linearity to the entire network, allowing the model to learn complex patterns [65].

- **2. Residual Blocks**

Residual Blocks help the model preserve identity information while expanding its capability to learn different variations in stroke and character shapes [124]. Each residual block has two Convolutional layers and uses Batch Normalization and ReLU. Additionally, a skip connection is included to support the model's refinement process rather than the creation of constant replacements. In other words, the addition of a skip connection means that the network will retain the original stroke shapes and other features of the data even if it learns new ones, making it ideal for characters that can often be found written in multiple ways, such as ψ , π , and Ω . Each of the 4 layers' filter size becomes progressively bigger, scaling to 512, and reducing the image's resolution by half each time. The progression can be described as shown in Table 4.2.

Layer	Filters	Resolution
Layer1	64	256x39
Layer2	128	128x20
Layer3	256	64x10
Layer4	512	32x5

Table 4.2. Residual Block analysis per layer.

- **3. SE Blocks**

SE Blocks are based on the Squeeze-and-Excitation (SE) mechanism that was introduced by Hu et al. [125], and we thought they would be a perfect fit for improving our network's quality of feature representations. An SE Block operates in two main stages:

- Squeeze (Global information Embedding)
The spatial dimensions of the input feature map are reduced using global average pooling. Each channel is condensed into a single scalar, capturing global spatial information and summarizing the channel-wise statistics.
- Excitation (Adaptive Recalibration)
The squeezed vector is passed through a small fully connected bottleneck network, which consists of a two-layer MLP (Multilayer Perceptron) with a ReLU and Sigmoid activation function. Essentially, this network learns non-linear channel-wise dependencies and outputs a set of weights in the range [0,1] which are then used to rescale the original feature maps via channel-wise multiplication.

In other words, the addition of SE Blocks allows the network to capture important handwriting patterns, enhances its sensitivity to diagnostic features (upper and lowercases), and ultimately learns what features to focus on by suppressing irrelevant or misleading information.

- **4. Multihead Attention**

The Multihead Attention mechanism was introduced by Vaswani et al. in the “Attention is All You Need” paper [85], revolutionizing the way that models process information. It enables the network to attend to spatially distributed features such as letter strokes and line alignment across the entire feature map. In simple terms, it is a mechanism that allows the model to focus on different parts of each input image simultaneously. Specifically, it was placed between Layers 3 and 4 of the Residual Blocks mentioned above to allow the model to capture long-range dependencies while the input image’s resolution was still sufficiently high (64x10 as opposed to 32x5) and included contextually rich features.

- **5. Global Pool and Fully Connected Layer**

The Global Average Pool (GAP) layer ensures robustness to variations in writing style and letter positions, while the Fully Connected layer interprets the GAP layer’s output and makes the final class predictions. Specifically:

- Global Average Pooling (GAP) is used to reduce each feature map to a single scalar by computing the average of all spatial elements. It focuses on the presence of features rather than their exact position, making it perfect for learning different handwritings. If we assume B is the Batch Size, C is the number of feature maps, and H, W are the Height and Width of the feature maps respectively. For an input tensor of shape $[B, C, H, W]$, the GAP layer will output a tensor of shape $[B, C, 1, 1]$, which we then flatten to $[B, C]$. This output is then sent to the fully connected layer.
- The Fully Connected (FC) layer acts as a classifier that associates image features with specific class labels. The layer takes as input the compact feature representation of the GAP layer and focuses on high-level semantics (using a 512-dimensional vector).

After the entire five-step process is complete, a `log_softmax` function is applied through `CrossEntropyLoss` in the training loop to make class predictions and select the one with the highest logit in the validation process.

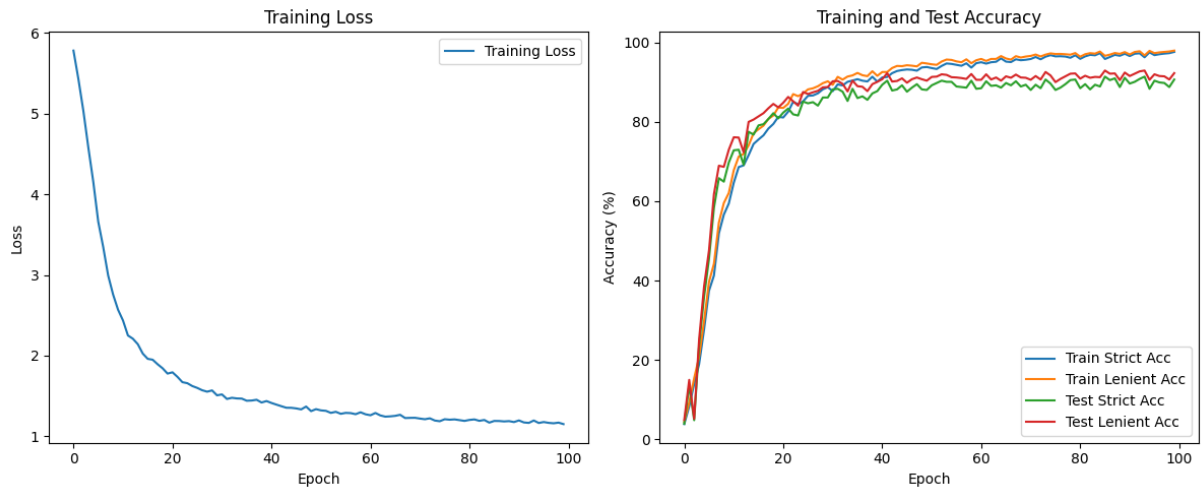


Fig 4.9. The proposed model’s loss and accuracy charts.

4.5 Model Performance

The training process took approximately 3 hours to complete on a system equipped with an RTX 4090 GPU provided by my professor. It was trained for around 600 epochs, and each one took roughly 16 seconds to complete with a batch size of 16. The final accuracies were 91.16% and 95.13% for the normal and custom accuracies respectively, which we considered sufficient to carry out our experiments. A chart of the model's loss and accuracy can be found in Figure 4.9, and it can be observed that the model required just a few epochs to reach the 90% accuracy range. However, we decided to keep training it for a few more epochs to allow it to learn dynamic augmentations and ultimately become more robust and accurate when it comes to classifying unseen data.

4.6 Input Image Preprocessing and Segmentations

Having created a model that can recognize Greek handwritten characters, the next step was to complete the input image preprocessing and segmentation steps, so that we could utilize the model. The following four sections go into depth about both of these processes and contain the methodologies and heuristics adopted to make the text recognition robust and reliable. It should be noted that this image preprocessing differs from the one already analysed in Chapter 4.3, since we are no longer dealing with the model's dataset, but rather real hand-written blocks of text. Additionally, the proposed segmentation process consists of 3 parts: the notebook line, single character, and word segmentation, respectively. A comprehensive pipeline of the entire OCR process proposed is shown in Figure 4.10.

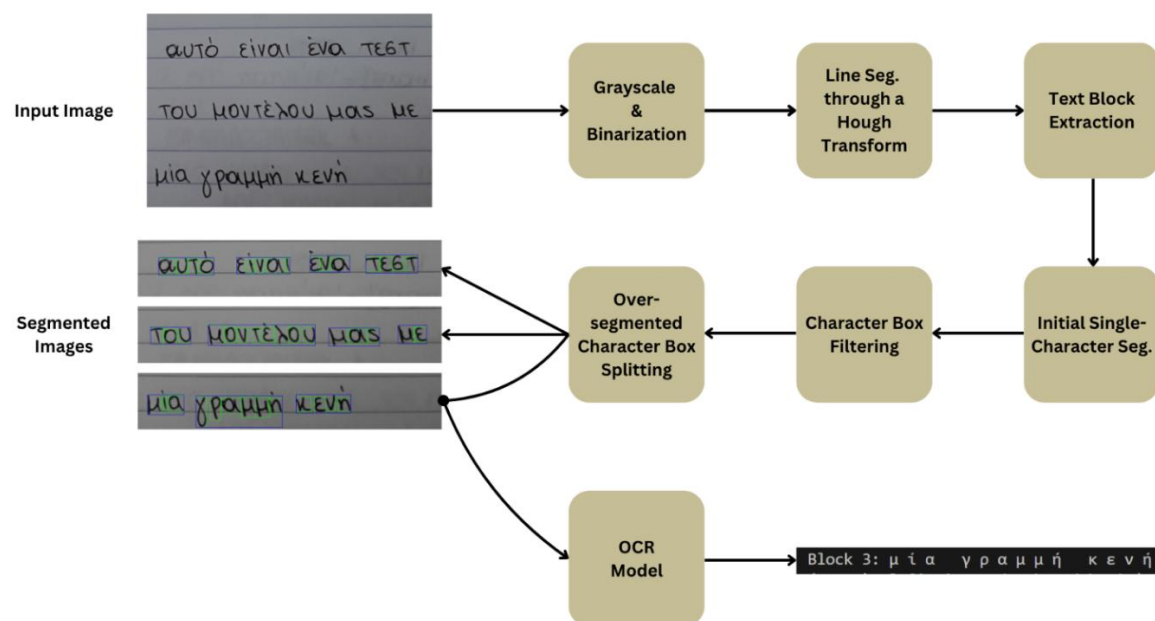


Fig. 4.10. The proposed OCR pipeline.

4.6.1 Input Image Preprocessing

The input image preprocessing consists of a quick two-step process, which aims to output a clearer, noise-free image that makes it easier to handle the segmentation processes analyzed in the following sub-chapters. Its key components can be described as follows, and a visual illustration is shown in Figure 4.11.

- **Grayscale conversion**

The original RGB image is opened and then converted to a single-channel image through the `ImageOps.grayscale` function provided by the Pillow library. As a result, the data dimensions are reduced and each pixel holds a value from the $[0, 255]$ range, where 0 is black, and 255 is white, respectively. This is an essential step to create a “cleaner” image because of the dimensionality reduction that ultimately allows us to work with just black and white colours, as opposed to three channels for each colour.

- **Binarization**

A threshold binarization is manually applied, where a constant threshold X is chosen in order to reduce background noise. Every pixel with intensity less than or equal to X is set to black (0), and every pixel greater than X is set to white (255). This allows the handwritten text with thick black pixels to remain the way it is, and every background noise to be removed.

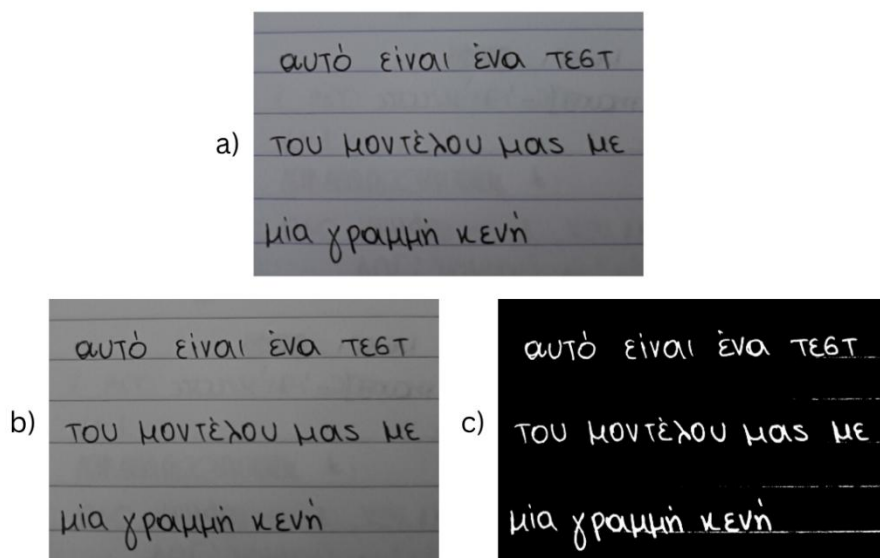


Fig. 4.11. Input image (a) shown in:
(b) Grayscale conversion, and (c) Binarization.

4.6.2 Notebook Line Segmentation

Having the ability to understand the coordinates of each notebook line given an input image is something that would be of great help in the long run. For starters, it gives us a general idea of what the boundaries of each block of text are, meaning that we are able to extract said blocks of text with ease. Additionally, understanding those boundaries can ultimately give us an idea of whether the text is written by a person of typical development or not. Further analysis of the latter can be found in section 4.7.

After careful research and experiments, we found that the Hough transform by Duda & Hart (1972) worked best for our input images. Essentially, the Hough transform is a computational technique used to detect concurrent curves by transforming points in image space into a parameter space, where patterns become easier to identify through a voting mechanism [126]. It matches each edge point (x, y) in image space to a sinusoidal curve in parameter space defined by $\rho = x\cos\theta + y\sin\theta$, and then accumulates votes using a 2D array indexed by ρ and ϑ , where each cell represents a candidate line. The more votes a cell receives, the more likely it is that it corresponds to a real line in the input image. This approach was especially helpful due to its robust nature in noisy environments, even when parts of the lines are missing as a result of the preprocessing process analyzed in section 4.6.1.

In the developed code, we particularly used the *hough_line* and *hough_line_peaks* functions provided by the *skimage.transform* library in Python. Specifically:

- *hough_line* builds the parameter space where lines manifest as peaks. It takes as input a binarized image with inverted pixel values (where the text is white and the background black) and uses angle values ϑ in a 180-degree field ranged from $(-\frac{\pi}{2}, \frac{\pi}{2})$.
- *hough_line_peaks* identifies local maxima in the accumulator array produced by the *hough_line* function. It first scans the 2D accumulator array for peaks above a threshold. Then, a minimum distance (in our case, 20) is enforced between peaks to avoid duplicate lines. The most prominent (ρ, ϑ) pairs are returned and stored in an array that we later use to manipulate line positions.

A visual representation of the Hough transform space, as well as its result can be found in Figure 4.12. Notice how it is able to recognize even the last line, which is comparatively thinner and misses a few pixels due to noise. Additionally, the final array produced by this example's transform contains 6 lists, one for each line, with the following values, which denote the respective (x1, x2) coordinate pairs of each line, as shown in Table 4.3.

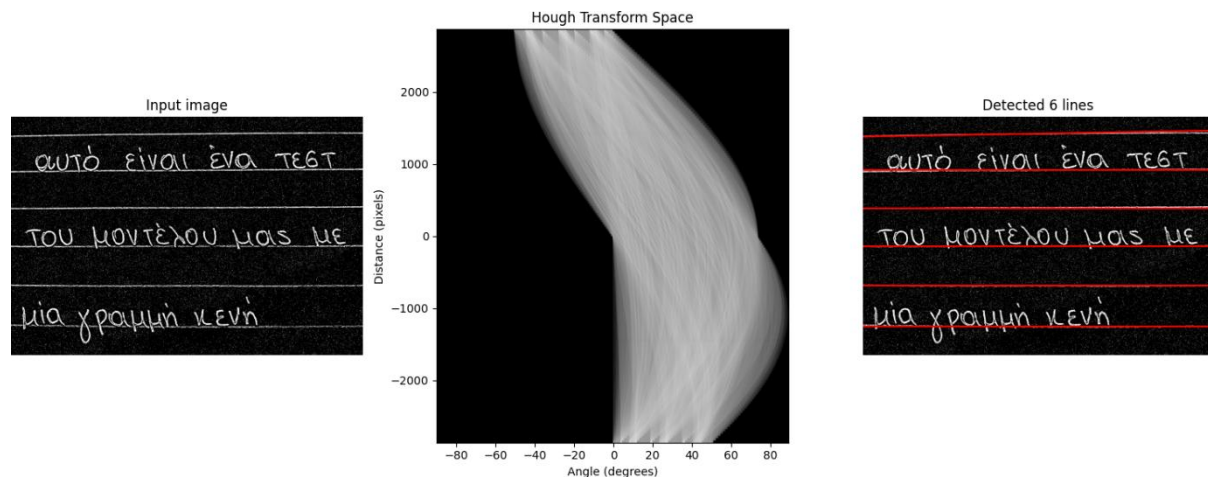


Fig. 4.12. Hough Transform example on realistic Input Image. Left to right: Input Image, Hough Transform Space, Detected lines coloured red.

Line	X1	X2
0	134.02	92.62
1	358.0	358.0
2	616.0	616.0
3	875.0	875.0
4	1137.0	1137.0
5	1413.0	1413.0

Table 4.3. (x1, x2) pairs for each line produced by the Hough Transform in Fig. 4.12's example.

Having knowledge of where each notebook line is allows us to generate areas of interest with ease. Specifically, we developed an algorithm that processes the two notebook lines surrounding blocks of text, adds padding between them, and then focuses on a shorter version of the area of interest. The two notebook lines of each area were found by utilizing the coordinates extracted from the Hough transform analysed in section 4.6.1.

The aforementioned padding was added in two distinct ways dynamically, and Figure 4.13 illustrates the regions of interest extracted from a sample input image.

- For the upper padding, the average height between two notebook lines was first calculated using Equation (7), where N is the number of lines found by the Hough transform, and $X1_i$ is the left-most value of the x-axis for the i -th line. Then, this value was multiplied by a very small number, as seen in Equation (8). The final upper padding was subtracted from each text block's upper bound in order to extend it upwards.

$$ALH = \frac{1}{N} \sum_{i=0}^{N-1} (X1_{i+1} - X1_i) \quad (7) \quad u_{pad} = ALH \times 0.03 \quad (8)$$

- For the lower padding, Equation (7) was utilized to find the average height between two notebook lines. This number was then multiplied by 0.7, as seen in Equation (9), and added to each block's lower bound to extend it downwards. This was done as a measure to capture letters that exceed the bottom notebook line, such as ξ and ρ .

$$l_{pad} = ALH \times 0.7 \quad (9)$$

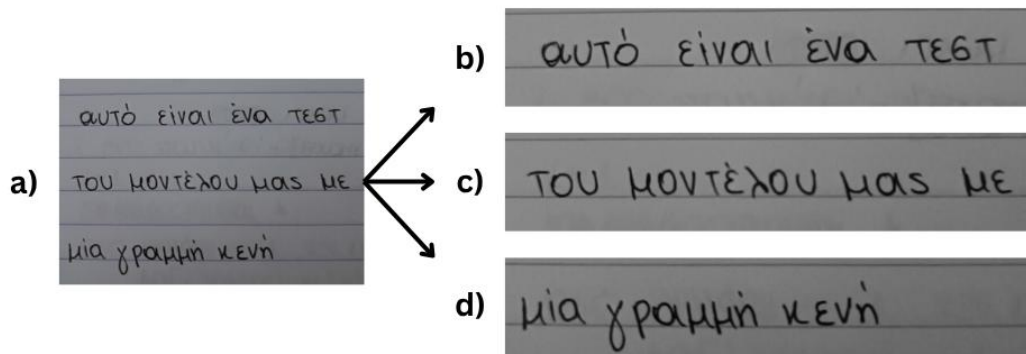


Fig. 4.13. Text block example, where (a) is the original image and (b), (c), and (d) are the extracted text blocks.

In addition to the heuristic line segmentation that worked exclusively for notebooks with lines, an implementation of the 2007 paper proposed by Arivazhagan et. al. [152] was created. The proposed algorithm uses Gaussian-based line modeling paired with probabilistic decision-making for ambiguous components, and it excels at segmenting lines on skewed text and overlapping components. The implementation was conducted in the C++ programming language, and its GitHub repository can be found in [154].

In essence, the algorithm begins by splitting an input image into distinct vertical sections referred to as “chunks”. Then, the chunks are examined based on their vertical projection profiles, and the local maxima (peaks) and local minima (valleys) are extracted in order to gain an understanding of where words and line gaps are. Afterwards, the extracted valleys are connected across chunks through probabilities, and a final refinement step is applied through Gaussian modeling or distance calculation, based on several criteria such as chunk placement and pixel presence.

Figure 4.14 shows an example of a chunk's histogram analysis, and Figure 4.15 shows two examples of the algorithm's performance with colour-coded lines. Its robust line-drawing logic displayed excellent results, especially in Figure 16's second input image, and the only major errors can be found in tonos and connected components handling, likely due to histogram misses.

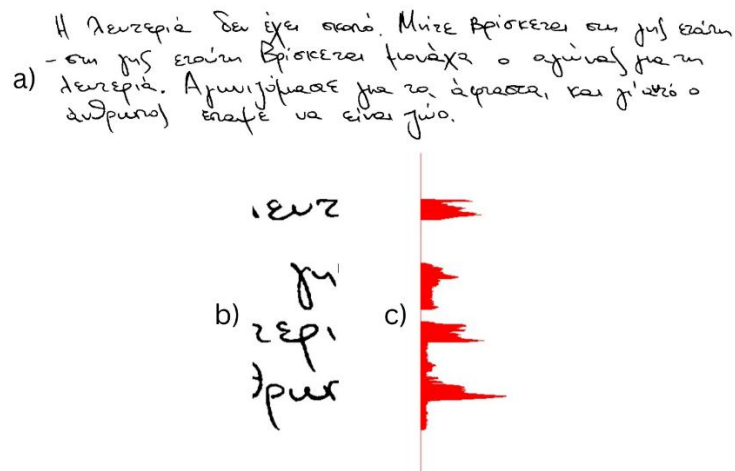


Fig. 4.14. Arivazhagan et. al.'s proposed chunk extraction, where: (a) is the input image, (b) is a chunk, and (c) is the chunk's vertical histogram.

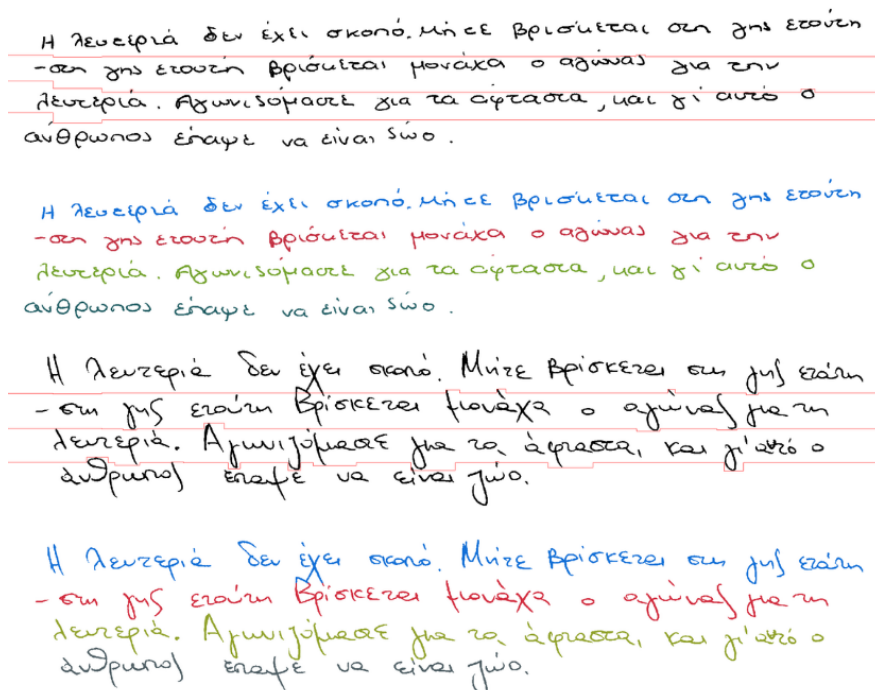


Fig. 4.15. Arivazhagan et. al.'s proposed algorithm tested on the ICDAR2012 Writer Identification competition dataset.

4.6.3 Single-Character Segmentation

As mentioned in previous sections, single character segmentation is a process that requires a lot of delicate calculations to produce favorable results. Although there have been several algorithms proposed for efficient single-character segmentation over the years, such as [127], most of them suffer from the complexity of connected component analysis. We experimented with different segmentation approaches, such as an algorithm proposed by Rajput et al., which performs peak width analysis [128], but found better results in a custom algorithm that applies OpenCV transforms and a series of heuristic approaches. This entire process was conducted prior to word segmentation as a heuristic strategy to better cluster individual letters into word units. Additionally, one of the goals of this single character segmentation was to isolate characters and discard small contours, such as tons and commas, for smoother character recognition.

First, an image of an entire passage is passed as an input. Binarization is then performed to isolate characters from the background using Otsu's method, which automatically determines the optimal threshold value that separates pixel intensity distributions into a foreground (letter) and a background [129]. After binarization, horizontal notebook lines are detected and removed to prevent them from interfering with the character-level segmentation process. This is achieved by utilizing a morphological operation with a long horizontal structuring element that isolates horizontal artifacts. The detected lines are then subtracted from the binarized image, producing an output of pure handwritten characters without the notebook lines. Then, a morphological dilation and erosion are applied with a rectangular kernel to remove noise and re-expand characters for better recognition. Finally, external contours are extracted and have their coordinates and areas stored by calculating their width and height product. The average of those areas is then calculated, and each individual contour is processed to remove false positives generated by noise or small artifacts like tonos, dialytics, and commas. This was accomplished by creating a threshold θ , calculated as shown in Equation (10), where CAVG is the average contour area. Contours with an area greater than ϑ are considered valid, and those with an area less than ϑ get discarded.

$$\theta = 0.2 \times CAVG \quad (10)$$

As a second measure for false positives – specifically tonos and dialytics – a position-based algorithm for contours was created. Specifically, overlapping bounding boxes on the y-axis were examined, and small rectangles directly above base characters were removed if three conditions were satisfied.

- i. The candidate's bottom edge is above the base character's top edge and within a vertical distance of H_{median} .
- ii. The candidate horizontally overlaps with the base character, allowing a small tolerance of around 5 pixels.
- iii. The candidate's height is less than 80% of the base character's height.

This approach ultimately handles edge cases where a tonos is too large and ends up being considered a character box. For example, Figure 4.16 displays an example where a tonos is almost the same height as the letter "ι", yet the tonos is the one that gets discarded. Additionally, this filtering process was a necessary step for the character recognition pipeline, since every character box was saved as an image, and unnecessary images would ultimately lead to inconsistent character recognition. The way tonos and dialytics were handled is analysed in the following section.

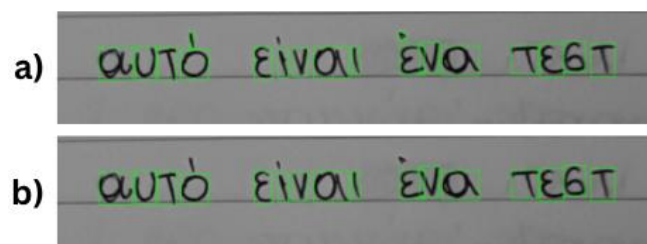


Fig. 4.16. Small artifact removal example, where: (a) is the threshold ϑ approach, and (b) includes both threshold ϑ and baseline β filtering.

A character segmentation approach as simple as this one suffers from connected components leading to oversegmentation. As a countermeasure, a custom splitting mechanism was implemented after the contour filtering process. An illustration of the result of this splitting algorithm is shown in Figure 4.17, where the two letters "μ" are slightly connected in (a), and split at an ideal splitting point in (b).

The algorithm begins by selecting candidate contours that might include two or more connected components based on two custom thresholds – average contour area and average contour width, both of which were calculated using Numpy’s median function. Those thresholds were then multiplied by distinct numbers to dynamically capture potentially problematic contour sizes, as shown in Equations (10) and (11). If a contour is bigger than those two thresholds, then it is considered a candidate for an over-segmented character.

$$\theta_area = median_area \times 1.5 \quad (10)$$

$$\theta_width = median_width \times 1.7 \quad (11)$$

Once a candidate has been created, the algorithm proceeds to create a column-wise projection in order to identify the thinnest connection point between the characters. This projection is then reduced to 40% of the contour’s central region to avoid edge noise and focus on the connected components. Finally, the minimum splitting point is selected as the column with the least pixel density, and the two new contours are created.

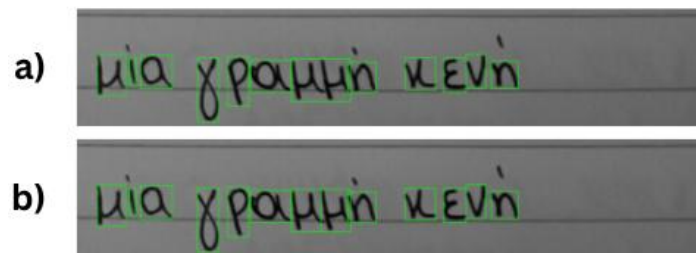


Fig. 4.17. Oversegmented contour split example, where: (a) is the original contour output, and (b) is the output of the connected component splitting algorithm.

4.6.4 Word Segmentation

Word segmentation in handwritten text is another difficult task, and although various segmentation techniques have been proposed over the years [130], none of them ensure 100% accuracy. Usually, single-character segmentation approaches complete this step prior to detecting the single-character contours, but this heuristic approach performs a dynamic contour clustering afterwards.

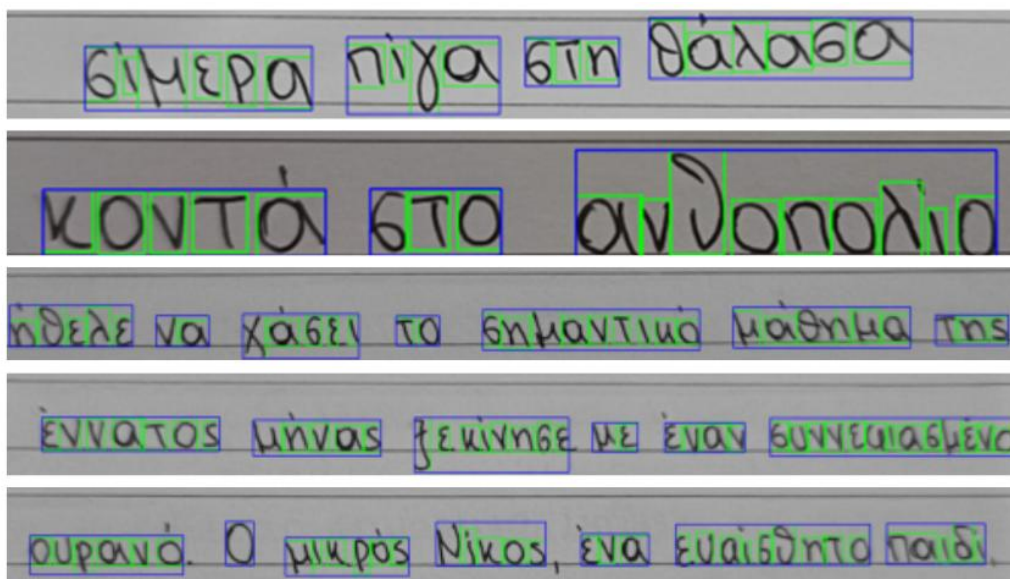


Fig. 4.18. Word segmentation performance examples, where blue denotes the words recognized.

Individual text blocks are given as input first, and their contours are sorted from left to right based on their x-coordinate to ensure correct reading order. After the average character width has been calculated, an acceptable gap between characters is defined as shown in Equation (12). Individual contours are then iterated through, and the horizontal gap between the current character and the one following it is calculated. If the gap is below the threshold, the character is added to the same word; otherwise, a new word begins. Figure 4.18 provides a visual illustration of how the algorithm performs. Notice how the space between the words “ευαίσθητο” and “παιδί” was captured in the final block, due to the dynamic threshold’s robustness.

$$\theta_{\text{width}} = \text{average_width} * 0.75 \quad (12)$$

4.6.5 Character Prediction

The character prediction process was simple, though there was a need to apply a series of heuristics to ensure robustness. The first step was to map the class indices into their respective class names (e.g., 1 -> A), after which the OCR model was loaded using PyTorch’s `load_state_dict` and `eval` functions.

Throughout the three-step segmentation process, each single character was assigned metadata as its corresponding image version was created. Specifically, the naming format captured the character’s block location, the word number it belonged to, and its numerical position within that word. This information allowed us to group letters together and separate words using the space indicator. An illustration of character-level image samples along with their metadata is shown in Figure 4.19.

As each character image was iterated through, their upper y-axis was extended towards the notebook line directly above them or the upper-most pixel of the block they belonged to. This approach allowed us to capture the potential tonos or dialytics below each character, and brought them to a form similar to the one the model was trained on. Following this, sizing transforms were applied and then sent to the model to perform the OCR, where the first output of the softmax activation function was selected as the final letter prediction. Finally, the letters showcased at the start of the chapter in Table 4.1 were transformed into their lowercase variant to boost consistency.

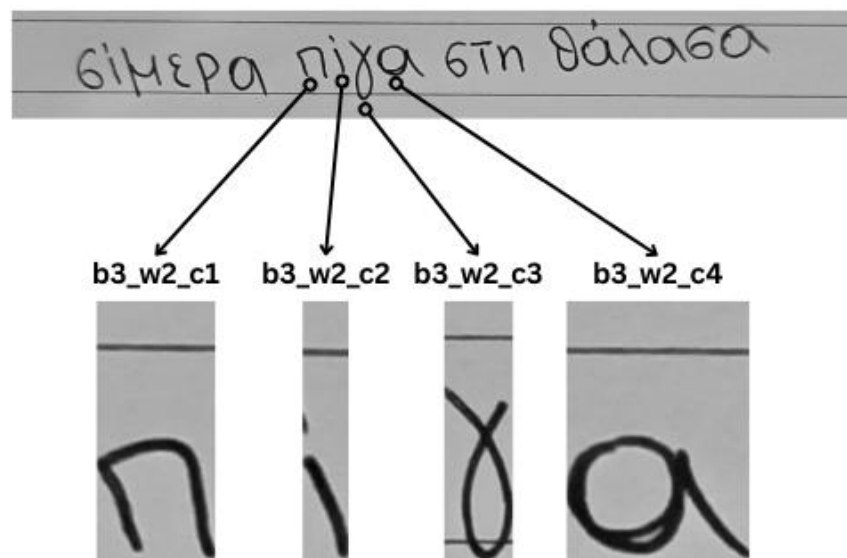


Fig. 4.19. Character-level cropping example, where:
b = block, *w* = word, *c* = character.

4.7 Writing and Spelling Disorder Detection

Having extracted precise letter position mappings and completed the letter recognition process, the next step was to create robust algorithms for writing and spelling disorder detection. For spelling disorder detection, words recognized by the OCR model were validated against a dictionary, and a Trie-based system was implemented to point out incorrect characters and suggest accurate replacements along with their respective error locations. In parallel, writing disorder detection focused on shape transforms and character box manipulation, complemented by checks for inconsistent use of capital letters and spaces.

4.7.1 Spelling Disorder Detection

For spelling disorder detection, it was necessary to utilize a dictionary that includes Greek words spelled correctly. The dictionary we decided to use contained a file with around one million words, including their frequencies. As a preprocessing measure, a script that reduced the dictionary size based on frequency was created. Specifically, words with a frequency score less than 2 were completely removed, and as a result, the dictionary's size was halved, storing 450.000 words.

For dictionary storing, both Python sets and TRIEs [131] were experimented with, but we decided to use a trie implementation as it is both quick and not as memory-intensive as a set. A brief analysis of both can be seen as follows:

- **Python Sets** use a hash-based structure to store strings [148], and every word is stored in full, regardless of similarity. A word's lookup time is $O(1)$ due to the utilization of hashes, though its memory usage is high due to the hash overhead [148].
- A **TRIE** has a tree-like structure where each node represents a letter [131], and in the case of the Greek language, 24. A typical word search takes $O(k)$ computational time, where k is the length of the word. It avoids duplication and unnecessary character storing, and thus can be considered a more memory-efficient approach. Additionally, its tree-based structure allows dynamic predictions and suggestions for incorrectly spelled words. For example, the input "σχολίο" would produce a prediction of the correctly spelled version "σχολείο" by utilizing the dictionary's frequency score and letter-node distribution.

A table of speed tests conducted for both the trie and set implementation can be seen in Table 4.4. Both implementations were exceptionally fast, with the set being a bit faster in every test; however, the tradeoff in terms of space was significant, and the difference in search speed was minimal.

Word	αυτός	φανταστικό	βιβλιογραφία	υπερδιπλασιάζεται	αποτελεσματική
Set	0.000006	0.000006	0.000008	0.000007	0.000005
Trie	0.000013	0.000015	0.000015	0.000015	0.000017

Table 4.4. Speed Tests between Python Set and Trie dictionary searches.

Once the dictionary lookup was implemented, a predefined error threshold needed to be established to flag text as potentially problematic when a certain number of misspelled words is exceeded. The final threshold we decided on for this project was $\vartheta = 10$ incorrectly spelled words for passages with around 70 characters. The testing process will be further analyzed in Chapter 5; however, it is worth noting that the words used in the final passage were carefully selected after continuous consultation with my professor.

Additionally, it was necessary to create a mechanism to handle mistakes made during the OCR process and prevent them from being classified as incorrectly spelled words. Specifically, if the model predicted a letter with a probability below 0.70, the entire word to which the letter belonged was excluded from the threshold-based classification. This approach ultimately handles inconsistencies in letter recognition and segmentation. If a word is guessed correctly and not found in the Trie, the counter for incorrectly spelled words is incremented by one, and if that number reaches ten, the user is informed accordingly.

In addition to the incorrectly spelled word check, a word suggestion system was implemented using edit distance calculations through the Levenshtein distance [132], accompanied by the Wagner-Fischer algorithm [133]. In essence, the system operates by traversing the Trie dictionary while dynamically tracking edit operations such as insertions, deletions, and substitutions. A depth-first search is used to explore paths, and the Levenshtein distance for each candidate word is calculated by measuring the difference between them through string similarity. Specifically, this is represented by a matrix that includes integers denoting how similar two words are. The smallest value is then selected, and in case of ties, the word with the highest frequency is chosen as the best match. This two-phase approach handles key features of the Greek language, such as tonos and diacritics, through the edit operations, and an example can be seen in Figure 4.20.

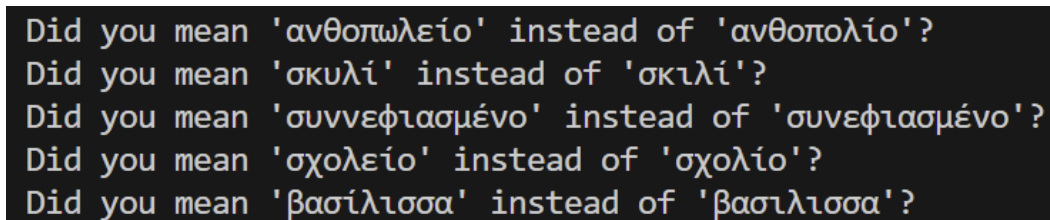


Fig. 4.20. Incorrectly spelled word check examples.

4.7.2 Writing Disorder Detection

Three main algorithms were implemented in order to detect key patterns commonly found in passages written by individuals diagnosed with writing disorder.

The first approach detects irregularities in word alignment. Specifically, individual words inside each block of text had their vertical positions analyzed and character bottoms calculated. The median of character bottoms was then used to estimate the baseline of each word, and the distance between those baselines and the bottom notebook line was calculated. It should be noted that descending letters that could skew the calculations, such as “ρ”, “μ”, and “ξ”, were excluded from the calculations. The filtering process was achieved by setting a descending threshold θ_{desc} , and it ensured that only characters with typical alignment contributed to the baseline calculation.

If we consider:

median_top: the median of the top y-coordinates of all characters in a word.

median_height: the median height of all characters.

Then, the threshold used to identify deep descending characters is defined as shown in Equation (13), where 10% is added to *median_height* in order to allow for a more natural variation in letter size. Any character whose bottom falls below the threshold is considered a descender and is removed from the word baseline calculation.

$$\theta_{desc} = median_top + median_height * 1.1 \quad (13)$$

After the individual word baselines are calculated, the deviation from the bottom notebook line is computed, and if it exceeds threshold θ_{base} , the word is flagged as problematic. The calculation of θ_{base} is defined as 10% of the height between two notebook lines (ALH), as shown in Equation (14). This approach appeared to perform well on text that deviated both above and below the notebook's baseline, and two examples can be found in Figure 4.21.

$$\theta_{base} = 0.10 \times ALH \quad (14)$$

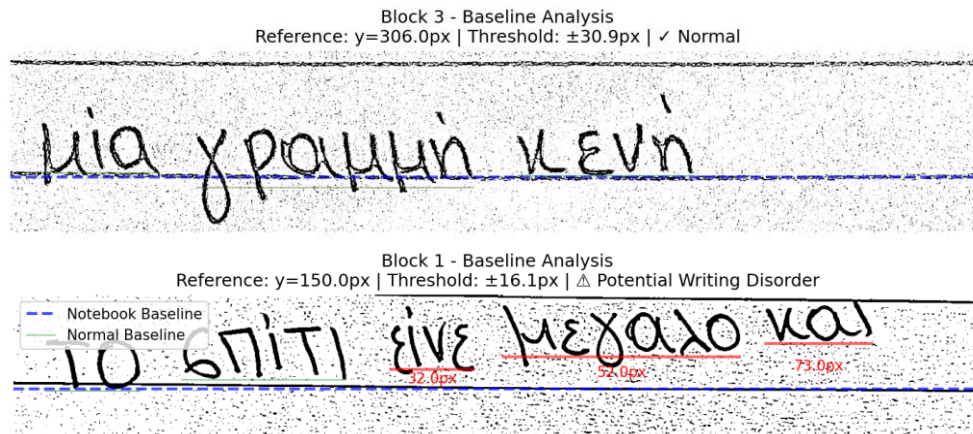


Fig. 4.21. Examples of the word alignment irregularity algorithm, where: green = normal, and red = problematic.

The second algorithm handled inconsistent capital letter usage in words. This was achieved by a string analysis for recognized words, which marked them as problematic when any character other than the first one was written in its uppercase variant, as shown in Figure 4.18. While this algorithm was easy to implement, it was exceptionally difficult to achieve great results due to the setback of needing to transform several of the OCR model's misclassified letters into their lowercase variants. As a result, only 12 letters of the Greek alphabet could be accurately checked for inconsistent capitalization, and a list of those letters is shown in Table 4.5.

Α	Γ	Δ	Ζ	Η	Λ
Μ	Ν	Ξ	Σ	Υ	Ω

Table 4.5. Letters that were tested for the inconsistent uppercase usage algorithm.

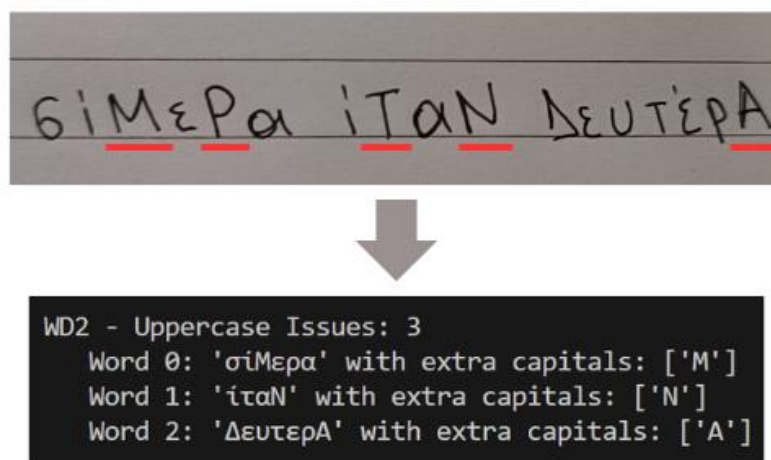


Fig. 4.22. Example of the capitalization inconsistency algorithm.

The third algorithm was designed to detect irregularity in spaces between words. The algorithm takes a block as an input and begins by calculating each word's left-most and right-most horizontal coordinate. These coordinates are then stored as tuples indexed by word position and sorted based on their horizontal placement to ensure sequential comparison. Then, the horizontal gap between each word pair is calculated, and a dynamic threshold θ_{gap} is calculated as shown in Equation (15), where an anomaly is detected if any word gap is above 180% of the median, or if the bounding boxes generated for words are too big. Figure 4.23 illustrates an example of the algorithm's performance on words with inconsistent spacing.

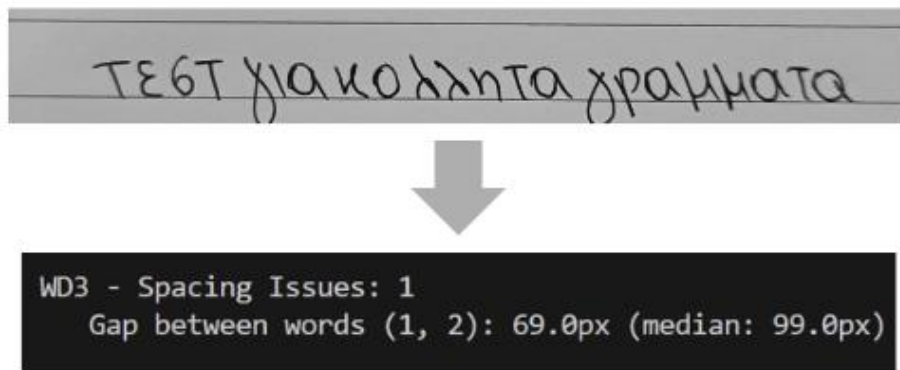


Fig. 4.23. Example of the spacing irregularity algorithm.

4.8 Experiments and OCR Comparison

For our personal experiments, we decided to work with different character and word segmentation techniques, as well as two OCR models to compare our own model and algorithms with. Specifically, we experimented with different CV2 image analysis techniques, character segmentation methodologies proposed by different researchers, Google's Tesseract, the word-level detection model CRAFT [141], and our proposed methods and model. Additionally, all tests were conducted on data we received from participants who wrote the passages introduced in Chapter 5, as the format needed to match the structural assumptions of the proposed segmentation pipeline.

Before proceeding with the experiments, there is a need to introduce the CRAFT model, as well as why we decided to use it for our testing. CRAFT is a robust word segmentation algorithm that generates heatmaps for character regions and affinities, making it ideal for curved or distorted text [141]. It was originally designed for real-world image character segmentation, and is not considered ideal for handwritten text. Additionally, it is trained on the SynthText [142], IC13 [143], and IC17 [144] datasets, which support multilanguage scripts, though Greek was not one of them. However, we still decided to test its segmentation results since Greek shares similar characters to its supported scripts.

4.8.1 Word Segmentation Experiments

Word segmentation accuracy was evaluated using three algorithms: CRAFT, Tesseract OCR, and the proposed approach. To assess performance, two metrics were utilized – precision and word error rate (WER) – to effectively capture segmentation correctness and errors.

- **Precision** is a commonly used accuracy metric for evaluating classification accuracy. It is calculated as the ratio of correctly segmented words to the total number of predicted segments, as shown in Equation (15).

If we assume:

TP : The number of true positives (correctly classified words).

FP : The number of false positives (misclassified words).

Then, the precision formula is defined as follows:

$$Precision = \frac{TP}{TP+FP} \quad (15)$$

- **Word error rate (WER)** is a metric commonly used in speech recognition and transcription accuracy evaluation [144], though we thought it would be ideal for word segmentation.

If we consider:

S : The number of wrongly segmented words.

D : The number of missed words.

I : The number of oversegmented words.

N : The total words in the ground truth.

Then, the WER can be defined as shown in Equation (16).

$$WER = \frac{S+D+I}{N} \quad (16)$$

Table 4.6 illustrates the word segmentation results, where CRAFT struggled with oversegmentation, while our proposed approach tended to undersegment words in blocks with inconsistent spacing. Additionally, Tesseract OCR scored a WER of 1.064 – or approximately 106% error rate – due to its letter and word hallucinations. Its precision score is remarkably low, though the WER metric enabled more representative results.

4.8.2 Character Segmentation Experiments

Character segmentation accuracy was calculated through the same metrics as the ones discussed in the previous section, though the WER metric was changed to CER – or character error rate – using the same calculations, but for characters instead of words. CRAFT performed exceptionally well due to the lack of connected components in the images used for testing, and only struggled with oversegmentation. However, Tesseract OCR showed underwhelming results that involved hallucinations and missing letters, reaching a CER score of 1.79 – or nearly 180% error rate.

4.8.3 Character Recognition Experiments

Character recognition accuracy was also evaluated using the precision and CER metrics. The proposed OCR system performed relatively well, and its accuracy was close to the model’s final validation accuracy. Additionally, most of the errors included misclassified letters which looked similar (e.g., O and 0), and character segmentation faults.

Word Segmentation	Tesseract OCR	CRAFT	Proposed Approach	Character Segmentation	Tesseract OCR	CRAFT	Proposed Approach
Precision	38.70%	78.43%	98.38%	Precision	13.72%	98.30%	99.21%
WER	106.45%	29.03%	4.83%	CER	179.19%	1.33%	0.73%

Tables 4.6 and 4.7. Word segmentation and single-character segmentation experiments.

OCR Accuracy	Tesseract OCR	i2OCR	Convertio*	OCR.Space*	2OCR**	Proposed Approach
Precision	12.97%	11.95%	x	x	94.51%	94.7%
CER	87.06%	88.82%	x	x	5.49%	4.92%

Table 4.8. OCR accuracy experiments.

**Convertio and OCR.Space could not produce any reasonable output for the given inputs.*

***Though 2OCR performed exceptionally well, it automatically corrected wrongly spelled words.*

CHAPTER 5

Tests and Use Cases

5.1 Test Structure

In order to test our model and algorithms, we conducted a writing test on 15 Greek university students. The passages we requested the students to write were read aloud and contained a series of words that are often used by my professor to flag potential writing and spelling disorders. Additionally, the participants were asked to keep one line empty and to avoid connecting characters too much in order to prevent letters from being stuck together and thus becoming unable to be recognized by the OCR model. The ground truth for those passages is shown in Figure 5.1.

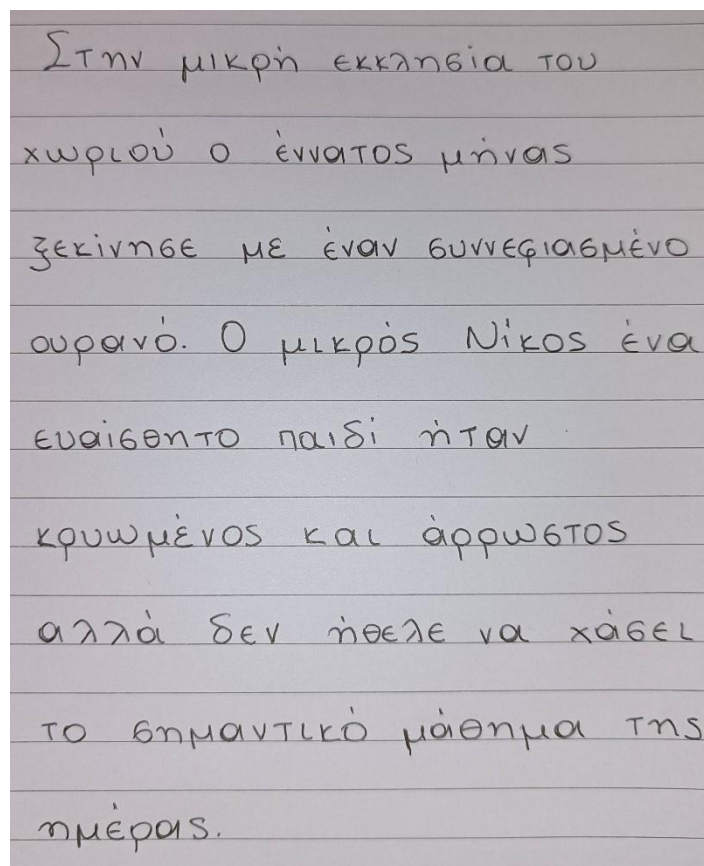


Fig. 5.1. The ground truth of the text recited to the participants.

5.2 Test Results

Upon having received the raw data from the participants, each image was passed to the algorithm and had it generate the OCR predictions, as well as the writing and spelling disorder classifications. Though the majority of the words were written correctly by the participants, those which were not were successfully captured by the model and flagged as problematic. Additionally, the three-step writing disorder detection performed exceptionally well, and was able to analyze words exceeding the bottom notebook baseline, as well as understand spacing and capitalization inconsistencies.

Before continuing to the test results, it is necessary to introduce the recall and F1-Score metric, which were used to evaluate the results alongside the precision metric.

- **Recall**, also referred to as sensitivity, calculates the model’s ability to find all of the positive instances, and its formula can be found in Equation (1).

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)} \quad (1)$$

- **F1-Score** is a commonly used metric that balances precision and recall by creating a harmonic mean between them, as shown in Equation (2).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (2)$$

Table 5.1 shows the performance of each important element of the proposed approach, and the metrics used for our tests – precision and F1-Score – were based on different criteria: line deviation detection, potential disorder classification, and overall OCR accuracy. It is worth noting that the binary classification for potential writing and/or spelling disorder was carefully validated by my professor to obtain the ground truth.

Table 5.2 presents the results of experiments evaluating various modern models on their ability to process misspelled words. Specifically, it measures how often models read and classify words exactly the way they are written, without attempting to automatically correct them. All of the metrics use a character-based evaluation, and the models selected for the experiments were 2OCR, GPT-5, Gemini 2.5 Flash, Copilot, and our proposed model. Finally, the prompt given to the LLMs was “Transcribe this Greek passage and give me your opinions on it.”.

	Line Deviation	Spelling Disorder Classification	Writing Disorder Classification	Overall OCR Accuracy
Precision	98.27%	91.31%	98.4%	94.7%
F1-Score	97.88%	90.14%	98.09%	93.52%

Table 5.1. Performance metrics extracted from the control group data.

Incorrect Spelling	2OCR	GPT-5	Gemini 2.5 Flash	Copilot	Proposed Approach
Precision	46.15%	80.76%	15.38%	0%	96.15%
F1-Score	48.32%	82.84%	17.42%	0%	98.03%
CER	54.21%	16.72%	83.21%	x	4.92%

Table 5.2. Performance of models on incorrectly spelled word classification.

GPT-5 showed surprisingly positive results compared to its previous GPT-o4 model, but still made a few automatic corrections that distorted the original written text given as input. Meanwhile, 2OCR achieved close to 50% precision, Gemini 2.5 Flash self-corrected almost every incorrectly written word, and Copilot consistently corrected every recognized word without exception.

CHAPTER 6

Conclusion, Limitations, and Later Works

6.1 Conclusion and Results

In this thesis we proposed our take on the Greek OCR problem, as well as the recognition of potential writing difficulties in handwritten text. We believe that making an accessible tool for everyone to use is an essential step in breaking the wall of misunderstanding and limited awareness surrounding SLDs related to writing. Furthermore, we believe that a quick, intuitive and non-intrusive self-assessment mechanism could ultimately encourage more people to seek appropriate support and engage in targeted practice if needed. AI-powered medical assessments require a lot of delicate attention, though we believe that such approaches are often able to make for a solid estimation [155, 156], and are encouraged to be used as such.

To the best of our knowledge, this is one of the first studies conducted to attempt to detect potential writing disorders on handwritten text, and we worked hard to incorporate modern image analysis techniques. We thoroughly believe that our approach is by no means perfect, but rather a stepping stool that could be used for further optimization techniques that would ultimately result in better accuracy – both on the spelling, and writing disorder classification.

6.2 Limitations and Later Works

It is worth revising on the main limitations of the proposed approach. Those include:

- The lack of data used for the model’s training process. Even though 200 is a decent amount of different handwriting styles for a project of a scale as big as this one, it is highly likely that various handwriting styles were not included, which may result in potential misclassifications by the proposed OCR model.
- Training our model on letters instead of words can also be considered a major setback. Large models that handle handwritten text recognition often train on entire words and produce exceptional results, albeit with hallucinations. A different approach could consist of a hybrid LLM trained on both words and letters. Additionally, an approach involving the transcript mapping problem on single-character level is currently being developed by me and Dr. Gatos Vasilis at the National Centre for Scientific Research Demokritos.
- Having to leave a notebook line empty is a heuristic that worked for us and is necessary to run the algorithms we developed. It eventually improved the OCR model’s accuracy and potential writing disorder detection, but we believe that it may not be necessary to use such approach on a more robust OCR model and pipeline.
- Needing to leave a tiny bit of space between letters and not connecting them is probably the biggest limitation of this project. We found it is extremely difficult to properly segment connected handwritten letters, and as mentioned in section 4.6.3, even though multiple techniques have been proposed over the years, none of them ensure that the final segmentation is always correct.

While we do believe our approach for writing learning disability detection works sufficiently when specific conditions are met, our proposed approach is still apt to making false classifications, as shown in our experiment results. As mentioned above, this thesis was not only attempting to solve the OCR problem itself, but also the writing and spelling disorder detection problem, which implies that the letter recognition worked perfectly in the first place. We also believe that more heuristics and more complicated approaches would ultimately increase the performance of our proposed algorithms.

REFERENCES

- [1] Aremu, E. S., & Adewunmi, A. T. (2023). *Understanding the nature of dysgraphia: Exploring its meaning, symptoms, impacts and advancing management strategies. Journal of Studies in Humanities*, 13(1), 54–66.
- [2] Zygouris, N. C., Toki, E. I., Vlachos, F., Styliaras, S. K., & Tziritas, N. (2025). *The implementation of the Askisi-SD neuropsychological web-based screener: A battery of tasks for screening cognitive and spelling deficits of children. Education Sciences*, 15(4), 452.
- [3] Protopapas, A., Fakou, A., Drakopoulou, S., Skaloumbakas, C., & Kyriakou, A. (2012). *What do spelling errors tell us? Classification and analysis of errors made by Greek schoolchildren with and without dyslexia. Reading and Writing*, 26(5).
- [4] Protopapas, A., & Vlahou, E. L. (2009). *A comparative quantitative analysis of Greek orthographic transparency. Behavior Research Methods*, 41, 991–1008.
- [5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning. Nature*, 521, 436–444.
- [6] He, Z., Zhang, C., Wu, Z., Chen, Z., Zhan, Y., Li, Y., Zhang, Z., Wang, X., & Qiu, M. (2025). *Seeing is believing? Mitigating OCR hallucinations in multimodal large language models. arXiv preprint. <https://arxiv.org/abs/2506.20168>*
- [7] Rangasrinivasan, S., Suresh, M. S. S., Olszewski, A., Setlur, S., Jayaraman, B., & Govindaraju, V. (2025). *AI-enhanced child handwriting analysis: A framework for the early screening of dyslexia and dysgraphia. SN Computer Science*, 6, 399.
- [8] Caravolas, M. (2004). *Spelling development in alphabetic writing systems: A cross-linguistic perspective. European Psychologist*, 9(1), 3–14.
- [9] American Psychiatric Association. *Diagnostic and statistical manual of mental disorders*. 5th ed. Washington, DC: American Psychiatric Association, 2013.
- [10] Gerber, P. J. (2012). *The impact of learning disabilities on adulthood: review of the evidenced-based literature for research and practice in adult education. Journal of Learning Disabilities*, 45, 31–46.
- [11] Protopapas, A., Fakou, A., Drakopoulou, S., Skaloumbakas, C., & Kyriakou, A. (2013). *What do spelling errors tell us? Classification and analysis of errors made by Greek schoolchildren with and without dyslexia. Reading and Writing*, 26(5), 615–646.
- [12] American Psychiatric Association. (2013). *Diagnostic and statistical manual of mental disorders* (5th ed.). Arlington, VA: American Psychiatric Publishing.
- [13] Henderson, V. W. (2009). *Alexia and agraphia. In Handbook of Clinical Neurology* (Vol. 95, pp. 583–601). Elsevier. [https://doi.org/10.1016/S0072-9752\(08\)02137-4](https://doi.org/10.1016/S0072-9752(08)02137-4)
- [14] Muges, T., Arthi, J., & Dutt, S. (2023). *Specific learning disabilities: Revisiting the theories, research and rehabilitation in the current perspective. International Journal of Creative Research Thoughts*, 11(2), 456–470.

- [15] Ijeoma, J.-A., & Ugwu, C. J. (2019). *Dyslexia, neurodevelopmental conditions and comorbidity: A rule rather than an exception*. Archives in Neurology & Neuroscience, Iris Publishers.
- [16] Döhla, D., & Heim, S. (2015). *Developmental dyslexia and dysgraphia: What can we learn from the one about the other?* *Frontiers in Psychology*, 6, Article 2045. <https://doi.org/10.3389/fpsyg.2015.02045>
- [17] Nicolson, R. I., Fawcett, A. J., & Dean, P. (2005). *Developmental dyslexia, learning and the cerebellum*. *Journal of Neural Transmission*. Supplement, (69), 19–36. <https://pubmed.ncbi.nlm.nih.gov/16355601>
- [18] Butterworth, B. (2003). Developmental dyscalculia. *Dyslexia*, 9(2), 67–78. <https://doi.org/10.1002/dys.237>
- [19] Wilson, A. J., Andrews, T., Struthers, H., Rowe, V., Bogdanovic, R., & Waldie, K. E. (2014). Dyscalculia and dyslexia in adults: Cognitive bases of comorbidity. *Cognitive Neuropsychology*, 31(5–6), 527–560. <https://doi.org/10.1080/02643294.2014.934219>
- [20] Rosenblumm, S., Weiss, P. L., & Parush, S. (2004). *Handwriting evaluation for developmental dysgraphia: Process versus product*. *Reading and Writing: An Interdisciplinary Journal*, 17, 433–458.
- [21] Shaywitz, S. E., & Shaywitz, B. A. (2005). Dyslexia (specific reading disability). *Biological Psychiatry*, 57(11), 1301–1309. <https://doi.org/10.1016/j.biopsych.2005.01.043>
- [22] Hawke, J. L., Wadsworth, S. J., Olson, R. K., & DeFries, J. C. (2009). Etiology of reading difficulties and writing difficulties: Same or different? *Learning and Individual Differences*, 19(2), 113–117. <https://doi.org/10.1016/j.lindif.2008.05.001>
- [23] Chung, P. J., Patel, D. R., & Nizami, I. (2020). Disorder of written expression and dysgraphia: Definition, diagnosis, and management. *Translational Pediatrics*, 9(Suppl 1), S46–S54. <https://doi.org/10.21037/tp.2019.11.01>
- [24] Gubbay SS, de Klerk NH. A study and review of developmental dysgraphia in relation to acquired dysgraphia. *Brain Dev* 1995;17:1-8. 10.1016/0387-7604(94)00110-J
- [25] Wanzek, J., Vaughn, S., Roberts, G., & Fletcher, J. M. (2018). *Current evidence on the effects of intensive early reading interventions*. *Journal of Learning Disabilities*, Nov/Dec Issue.
- [26] Καρατζάς, Α., & Βάμβουκα, Ι. (2022). *Ειδική διαταραχή μάθησης με έλλειμμα στην ορθογραφία (δυσορθογραφία) και συμπεριληπτική διδακτική προσέγγισή της*. In Ν. Κιοσσογλου & Μ. Καλογερά (Eds.), *Outlining Learning Difficulties*. Kallipos.
- [27] Zygouris, N. (2023). *Introduction to Learning Difficulties: Causes and Interventions Withing the Classroom Context*. CS_U_134.
- [28] Mazade, A. (2011). *Written Language Disorders: Dyslexia and Dysorthography*. Paris: ESF Publishers.
- [29] Shaywitz, S. E. (2016). *Shaywitz DyslexiaScreen*. San Antonio, TX: Pearson Assessments.
- [30] Burns, M. K., VanDerHeyden, A. M., Duesenberg-Marshall, M. D., Romero, M. E., Stevens, M. A., Izumi, J. T., & McCollom, E. M. (2022). Decision accuracy of commonly used dyslexia screeners among students who are

- potentially at-risk for reading difficulties. *Learning Disability Quarterly*, 46(4), 1–11. <https://doi.org/10.1177/07319487221096684>
- [31] Powell-Smith, K. A., Good, R. H., & Atkins, T. (2010). *DIBELS Next Oral Reading Fluency Readability Study (Technical Report No. 7)*. Dynamic Measurement Group. <https://studylib.net/doc/18602694/dibels-next-oral-reading-fluency-readability-study-technical>
- [32] Devi, A., Prakash, S. S., & Abinaya, D. (2019). Online rehabilitation tool for identifying learning disabilities. *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, 1–5. IEEE. <https://doi.org/10.1109/ICSCAN.2019.8878838>
<https://doi.org/10.3390/educsci15040452>
- [33] Weiss, L. G., Saklofske, D. H., Holdnack, J. A., & Prifitera, A. (Eds.). (2016). *WISC-V assessment and interpretation: Scientist-practitioner perspectives*. Elsevier Academic Press. <https://doi.org/10.1016/B978-0-12-404697-9.00001-7>
- [34] Wechsler, D. (2014). *Wechsler Intelligence Scale for Children–Fifth Edition (WISC-V)*. Pearson. <https://doi.org/10.1037/t79359-000>
- [35] Raiford, S. E., Drozdick, L. W., Zhang, O., & Zhou, X. (2015). *WISC-V technical and interpretive manual*. Pearson.
- [36] ILSP / Athena RC. (2001). *eMADys: Software production for identification learning weaknesses*. <https://www.ilsp.gr/en/projects/emadys-software-production-for-identification-learning-weaknesses/>
- [37] Zygouris, N. C., Toki, E. I., Vlachos, F., Styliaras, S. K., & Tziritas, N. (2025). *The Implementation of the Askisi-SD Neuropsychological Web-Based Screener: A Battery of Tasks for Screening Cognitive and Spelling Deficits of Children*. *Education Sciences*, 15(4), 452.
- [38] World Health Organization (WHO). (1993). *The ICD-10 classification of mental and behavioural disorders*. World Health Organization.
- [39] Département de l'Instruction Publique. (2008). *Les clefs de l'école: Magazine d'information du Département de l'Instruction Publique*. Service Ecoles-Médias. <https://edudoc.ch/record/31673/files/2008.pdf>
- [40] ACS Athens. (n.d.). *Educational & diagnostic testing center*. [ACS Athens | Modeling education for the 21st Century](https://www.acsathens.gr/en/Modeling%20education%20for%20the%2021st%20Century)
- [41] Kostaras, A., Lekka, E., Pilafas, G., & Louka, P. (2024). *Breaking barriers: Strategies for early dyslexia identification in the Greek educational system*. *International Journal of Research and Review*, 11(3). <https://doi.org/10.52403/ijrr.20240348>
- [42] Turing, A. M. (1950). *Computing machinery and intelligence*. *Mind*, 59(236), 433–460.
- [43] McKinsey & Company. (2024). *The state of AI: Global survey*. <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai>
- [44] Naik, S. (2025, July 14). *Eye-opening AI statistics (2025): Market size, usage & more*. Resourcera. <https://resourcera.com/data/artificial-intelligence/ai-statistics/>
- [45] Kelleher, J. D. (2019). *Deep Learning*. MIT Press. [Google Books](https://books.google.com/books?id=8v3tDwAAQAAJ)
- [46] Sheikh, H., Prins, C., & Schrijvers, E. (2023). *Artificial Intelligence: Definition and Background*. In *Research for Policy* (pp. 15–41). Springer.

- [47] High-Level Expert Group on Artificial Intelligence. (2019). *A definition of AI: Main capabilities and scientific disciplines*. European Commission. https://ec.europa.eu/newsroom/dae/document.cfm?doc_id=56341
- [48] McCarthy, J., Minsky, M., Rochester, N., & Shannon, C. E. (1955). *A proposal for the Dartmouth summer research project on artificial intelligence*. Stanford University. <https://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html>
- [49] Goodman, D. & Keene, R. (1997). *Man versus machine: Kasparov versus Deep Blue*. H3 Publications.
- [50] Ågerfalk, P. J. (2020). Artificial intelligence as digital agency. *European Journal of Information Systems*, 29(1), 1–8. <https://doi.org/10.1080/0960085X.2020.1721947>
- [51] Köhl, N., Schemmer, M., Satzger, G., & others. (2022). Artificial intelligence and machine learning. *Electronic Markets*, 32(4), 2235–2244. <https://doi.org/10.1007/s12525-022-00598-0>
- [52] Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(4), 685–695. <https://doi.org/10.1007/s12525-021-00475-2>
- [53] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- [54] Huang, J.-C., Ko, K.-M., Shu, M.-H., & Hsu, B.-M. (2020). Application and comparison of several machine learning algorithms and their integration models in regression problems. *Neural Computing and Applications*, 32, 5461–5469. <https://doi.org/10.1007/s00521-019-04644-5>
- [55] Manasa, J., Gupta, R., & Narahari, N. S. (2020). *Machine Learning based Predicting House Prices using Regression Techniques*. Proceedings of the International Conference on Inventive Material Science Applications (ICIMIA), 624–630. <https://doi.org/10.1109/ICIMIA48430.2020.9074952>
- [56] Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2, 160. <https://doi.org/10.1007/s42979-021-00592-x>
- [57] Han J, Pei J, Kamber M. Data mining: concepts and techniques. Amsterdam: Elsevier; 2011.
- [58] Komaru, Y., Yoshida, T., Hamasaki, Y., Nangaku, M., & Doi, K. (2020). Hierarchical Clustering Analysis for Predicting 1-Year Mortality After Starting Hemodialysis. *Kidney international reports*, 5(8), 1188–1195. <https://doi.org/10.1016/j.ekir.2020.05.007>
- [59] Wang, H. (2017). *On the origin of deep learning*. arXiv. <https://arxiv.org/abs/1702.07800>
- [60] Rosenblatt, F. (1958). *The perceptron: A probabilistic model for information storage and organization in the brain*. *Psychological Review*, 65(6), 386–408. <https://doi.org/10.1037/h0042519>
- [61] Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences* (Doctoral dissertation, Harvard University).
- [62] OpenAI. (2022). Introducing ChatGPT. <https://openai.com/blog/chatgpt>
- [63] Gershenon, C. (2003). *Artificial neural networks for beginners*. arXiv. <https://arxiv.org/pdf/cs/0308031>

- [64] McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133. <https://doi.org/10.1007/BF02478259>
- [65] Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2021). *Activation functions in deep learning: A comprehensive survey and benchmark*. *Neural Computation & Applications*, 33, 13569–13598. <https://doi.org/10.1007/s00521-020-04744-5>
- [66] Popescu, M.-C., Balas, V. E., Perescu-Popescu, L., & Mastorakis, N. E. (2009). *Multilayer perceptron and neural networks*. *WSEAS Transactions on Circuits and Systems*, 8(7).
- [67] Janiesch, C., Zschech, P., & Heinrich, K. (2021). *Machine learning and deep learning*. *Electronic Markets*, 31(3), 685–695. <https://doi.org/10.1007/s12525-021-00475-2>
- [68] Lopez Davila, A. (n.d.). *Neural networks: The backpropagation algorithm*. College of William and Mary. Retrieved from <https://cklixx.people.wm.edu/teaching/math400/Annette-paper.pdf>
- [69] Goodman, R., & Miller, J. W. (1991). Objective functions for neural network classifier design. In *Proceedings of the 1991 IEEE International Symposium on Information Theory* (pp. 136). IEEE. <https://core.ac.uk/download/pdf/216269118.pdf>
- [70] Terven, J., Cordova-Esparza, D.-M., Romero-González, J.-A., Ramírez-Pedraza, A., & Chávez-Urbiola, E. A. (2025). A comprehensive survey of loss functions and metrics in deep learning. *Artificial Intelligence Review*, 58, Article 195. <https://doi.org/10.1007/s10462-025-11198-7>
- [71] Bryson, A. E., & Ho, Y. C. (1969). *Applied Optimal Control: Optimization, Estimation, and Control*. Waltham, MA: Blaisdell Publishing.
- [72] Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences* (Doctoral dissertation, Harvard University).
- [73] Rumelhart, D. E., & McClelland, J. L. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations*. MIT Press
- [74] Hecht-Nielsen, R. (1989, June). *Theory of the backpropagation neural network*. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (pp. 593–605). IEEE. <https://doi.org/10.1109/IJCNN.1989.118638>
- [75] Gabrié, M., Ganguli, S., Lucibello, C., & Zecchina, R. (2023). Neural networks: From the perceptron to deep nets. In M. Mézard, A. Montanari, & G. Parisi (Eds.), *Spin glass theory and far beyond: Replica symmetry breaking after 40 years* (Chapter 24). arXiv. <https://arxiv.org/abs/2304.06636>
- [76] Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166. <https://doi.org/10.1109/72.279181>
- [77] Cernadas, E. (2024). *Applications of computer vision, 2nd edition*. *Electronics*, 13(18), 3779. <https://doi.org/10.3390/electronics13183779>

- [78] Tappert, C. C., Suen, C. Y., & Wakahara, T. (1990). *The state of the art in online handwriting recognition*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8), 787–808. <https://doi.org/10.1109/34.57669>
- [79] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). *Backpropagation applied to handwritten zip code recognition*. *Neural Computation*, 1(4), 541–551. <https://doi.org/10.1162/neco.1989.1.4.541>
- [80] Gonzalez, R. C., & Woods, R. E. (2018). *Digital image processing* (4th ed.). Pearson.
- [81] O'Shea, K., & Nash, R. (2015). *An introduction to convolutional neural networks*. arXiv. <https://doi.org/10.48550/arXiv.1511.08458>
- [82] Gholamalinezhad, H., & Khosravi, H. (2020). Pooling methods in deep neural networks, a review. *Neural Computation*, 32(10), 2520–2553. https://doi.org/10.1162/neco_a_01384
- [83] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1990). Handwritten digit recognition with a back-propagation network. *Advances in Neural Information Processing Systems*, 2, 396–404.
- [84] Ranzato, M. A., Boureau, Y.-L., & LeCun, Y. (2008). *Sparse feature learning for deep belief networks*. *Advances in Neural Information Processing Systems*, 20, 1185–1192.
- [85] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention is all you need*. *Advances in Neural Information Processing Systems*, 30.
- [86] Dong, R., & Smith, D. A. (2018). *Multi-input attention for unsupervised OCR correction*. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), 2363–2372. <https://aclanthology.org/P18-1220.pdf>
- [87] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. <http://www.deeplearningbook.org>
- [88] Sun, Z., Huang, S., Wei, H.-R., Dai, X., & Chen, J. (2019). *Generating diverse translation by manipulating multi-head attention*. arXiv preprint arXiv:1911.09333.
- [89] OpenAI. (2024). *GPT-4 Technical Report*. arXiv preprint.
- [90] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020). *An image is worth 16x16 words: Transformers for image recognition at scale*. arXiv preprint arXiv:2009.07485.
- [91] Kaggle Kaggle. (n.d.). *Datasets*. Retrieved August 1, 2025, from <https://www.kaggle.com/datasets>
- [92] UCI Machine Learning Repository Dua, D., & Graff, C. (2019). *UCI Machine Learning Repository*. University of California, Irvine, School of Information and Computer Sciences. Retrieved August 1, 2025, from <https://archive.ics.uci.edu/ml/index.php>
- [93] Google Dataset Search Google. (n.d.). *Dataset Search*. Retrieved August 1, 2025, from <https://datasetsearch.research.google.com>
- [94] U.S. Government Open Data Portal U.S. Government. (n.d.). *Data.gov*. Retrieved August 1, 2025, from <https://www.data.gov>

- [95] Press, G. (2016, March 23). *Cleaning big data: Most time-consuming, least enjoyable data science task, survey says*. Forbes.
<https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/>
- [96] El Morr, C., Jammal, M., Ali-Hassan, H., & El-Hallak, W. (2022). *Data preprocessing*. In *Machine learning for practical decision making* (pp. 117–163). Springer.
https://doi.org/10.1007/978-3-031-16990-8_4
- [97] Wang, J., & Perez, L. (2017). *The effectiveness of data augmentation in image classification using deep learning* (arXiv:1712.04621). arXiv.
<https://arxiv.org/pdf/1712.04621>
- [98] Jamshed, H., Khan, M. S. A., Khurram, M., Inayatullah, S., & Athar, S. (2019). *Data preprocessing: A preliminary step for web data mining*. *3C Tecnología: Glosas de innovación aplicadas a la pyme*, 8(1), 206–221.
- [99] Gholamy, A., Kreinovich, V., & Kosheleva, O. (2018). *Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation* (UTEP-CS-18-09). University of Texas at El Paso.
https://scholarworks.utep.edu/cs_techrep/1209/
- [100] Joseph, V. R., & Vakayil, A. (2022). *Optimal ratio for data splitting*. arXiv.
<https://arxiv.org/pdf/2202.03326>
- [101] Aliferis, C., & Simon, G. (2024). *Overfitting, underfitting and general model overconfidence and under-performance: Pitfalls and best practices in machine learning and AI*. In *Artificial Intelligence and Machine Learning in Health Care and Medical Sciences* (pp. 477–524). Springer.
https://doi.org/10.1007/978-3-031-39355-6_10
- [102] Chen, G., Chen, P., Shi, Y., Hsieh, C.-Y., Liao, B., & Zhang, S. (2019). *Rethinking the usage of batch normalization and dropout in the training of deep neural networks*. arXiv.
<https://doi.org/10.48550/arXiv.1905.05928>
- [103] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A simple way to prevent neural networks from overfitting*. *Journal of Machine Learning Research*, 15(1), 1929–1958.
<https://www.jmlr.org/papers/volume15/srivastava14a>
- [104] Sergey Ioffe and Christian Szegedy. (2015) *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. arXiv. <https://doi.org/10.48550/arXiv.1502.03167>
- [105] Jansen, P. (2025, July). *TIOBE Index for July 2025*. TIOBE Software. Retrieved from <https://www.tiobe.com/tiobe-index/>
- [106] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, Ł., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2016). *TensorFlow: Large-scale machine learning on heterogeneous distributed systems* (arXiv:1603.04467). arXiv.
<https://doi.org/10.48550/arXiv.1603.04467>

- [107] Rao, K. G., & Arunesh, A. (2021). *Comparison of TensorFlow and PyTorch in convolutional neural network-based applications*. International Journal of Computer Science and Technology, 12(2), 45–51.
- [108] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library* (arXiv:1912.01703). arXiv. <https://doi.org/10.48550/arXiv.1912.01703>
- [109] Hayes, D. (2025, June 16). *TensorFlow vs PyTorch: A Comparative Analysis for 2025*. Leapcell. https://leapcell.io/blog/tensorflow-vs-pytorch-a-comparative-analysis-for-2025?detect_lang=0
- [110] O'Connor, R. (2021, December 14). *PyTorch vs TensorFlow in 2023*. AssemblyAI. <https://www.assemblyai.com/blog/pytorch-vs-tensorflow-in-2023>
- [111] Bradski, G. (2000). *The OpenCV Library*. Dr. Dobb's Journal of Software Tools.
- [112] Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- [113] Incode. (2025). *The history of optical character recognition (OCR)*.
- [114] Smith, R. (2007). *An overview of the Tesseract OCR engine*.
- [115] Louloudis, G., Gatos, B., & Stamatopoulos, N. (2012). *ICFHR 2012 Competition on Writer Identification Challenge 1: Latin/Greek Documents. 13th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 829–834. https://users.iit.demokritos.gr/~bgat/ICFHR_2012_Louloudis.pdf
- [116] He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep residual learning for image recognition*. arXiv preprint. arXiv:1512.03385.
- [117] Buda, M., Maki, A., & Mazurowski, M. A. (2018). A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks, 106, 249–259. <https://doi.org/10.1016/j.neunet.2018.07.011>
- [118] Müller, R., Kornblith, S., & Hinton, G. (2019). *When does label smoothing help?* Advances in Neural Information Processing Systems, 32, 4694–4703.
- [119] Kingma, D. P., & Ba, J. (2015). *Adam: A method for stochastic optimization*. In International Conference on Learning Representations (ICLR). <https://arxiv.org/abs/1412.6980>
- [120] Loshchilov, I., & Hutter, F. (2019). *Decoupled weight decay regularization*. In International Conference on Learning Representations (ICLR). <https://arxiv.org/abs/1711.05101>
- [121] Al-Kababji, A., Bensaali, F., & Dakua, S. P. (2022). *Scheduling techniques for liver segmentation: ReduceLROnPlateau vs OneCycleLR*. In Intelligent Systems and Pattern Recognition (pp. 204–212). Springer. https://doi.org/10.1007/978-3-031-08277-1_17
- [122] Zhang, J., He, T., Sra, S., & Jadbabaie, A. (2020). *Why gradient clipping accelerates training: A theoretical justification for adaptivity*. arXiv. <https://doi.org/10.48550/arXiv.1905.11881>

- [123] Prechelt, L. (1998). *Early stopping — but when?* In G. B. Orr & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade* (pp. 55–69). Springer. https://doi.org/10.1007/3-540-49430-8_3
- [124] He, K., Zhang, X., Ren, S., & Sun, J. (2016). *Deep residual learning for image recognition*. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). <https://doi.org/10.1109/CVPR.2016.90>
- [125] Hu, J., Shen, L., & Sun, G. (2018). *Squeeze-and-excitation networks*. *CVPR*.
- [126] Duda, R. O., & Hart, P. E. (1972). *Use of the Hough transformation to detect lines and curves in pictures*. *Communications of the ACM*, 15(1), 11–15.
- [127] Dave, N. (2015). *Segmentation methods for handwritten character recognition*. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 8(4), 155–164.
- [128] Rajput, V., Jayanthi, N., & Indu, S. (2019). *An efficient character segmentation algorithm for connected handwritten documents*. In S. Sundaram & G. Harit (Eds.), *Document Analysis and Recognition* (pp. 97–105). *Communications in Computer and Information Science*, Vol. 1020. Springer.
- [129] Otsu, N. (1979). *A threshold selection method from gray-level histograms*. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62–66. <https://doi.org/10.1109/TSMC.1979.4310076>
- [130] Louloudis, G., Gatos, B., Pratikakis, I., & Halatsis, C. (2009). *Line and word segmentation of handwritten documents*. *International Journal on Document Analysis and Recognition*, 12, 85–98.
- [131] Bodon, F., & Rónyai, L. (2003). *Trie: An alternative data structure for data mining algorithms*. *Mathematical and Computer Modelling*, 38(7–8), 739–751.
- [132] Levenshtein, V. I. (1966). *Binary codes capable of correcting deletions, insertions, and reversals*. *Soviet Physics Doklady*, 10(8), 707–710.
- [133] Wagner, R. A., & Fischer, M. J. (1974). *The string-to-string correction problem*. *Journal of the ACM*, 21(1), 168–173. <https://doi.org/10.1145/321796.321811>
- [134] Fletcher, J. M., Lyon, G. R., Fuchs, L. S., & Barnes, M. A. (2019). *Learning disabilities: From identification to intervention* (2nd ed.). The Guilford Press.
- [135] Whittaker, M. M., & Burns, M. K. (2024). *Evaluation for specific learning disabilities: Allowable methods of identification and their implications*. *Developmental Psychology*.
- [136] Crouch, A. L., & Jakubecy, J. J. (2007). *Dysgraphia: How it affects a student's performance and what can be done about it*. *Teaching Exceptional Children Plus*, 3(3), Article 3.
- [137] Rangasrinivasan, S., Suresh, M. S., Olszewski, A., Setlur, S., Jayaraman, B., & Govindaraju, V. (2025). *AI-enhanced child handwriting analysis: A framework for the early screening of dyslexia and dysgraphia*. *SN Computer Science*, 6, Article 399. <https://doi.org/10.1007/s42979-025-03927-0>
- [138] Snowling, M. J. (2000). *Dyslexia*. Blackwell Publishing.

- [139] Ehri, L. C. (2000). *Learning to read and learning to spell: Two sides of a coin*. *Topics in Language Disorders*, 20(3), 19–36.
<https://doi.org/10.1097/00011363-200020030-00005>
- [140] Wagner, R. K., Torgesen, J. K., & Rashotte, C. A. (1994). *Development of reading-related phonological processing abilities: New evidence of bidirectional causality from a latent variable longitudinal study*. *Developmental Psychology*, 30(1), 73–87. <https://doi.org/10.1037/0012-1649.30.1.73>
- [141] Treiman, R. (1993). *Beginning to spell: A study of first-grade children*. Oxford University Press.
- [141] Baek, Y., Lee, B., Han, D., Yun, S., & Lee, H. (2019). *Character region awareness for text detection*. *arXiv preprint*. arXiv:1904.01941.
- [142] Gupta, A., Vedaldi, A., & Zisserman, A. (2016). *Synthetic data for text localisation in natural images*. *arXiv preprint*. arXiv:1604.06646.
- [143] Karatzas, D., Shafait, F., Uchida, S., Iwamura, M., Gomez i Bigorda, L., Mestre, S. R., ... & De Las Heras, L. P. (2013). *ICDAR 2013 robust reading competition*. In *12th International Conference on Document Analysis and Recognition* (pp. 1484–1493). IEEE.
- [144] Nayef, N., Yin, X. C., Chanda, A., Karatzas, D., Luo, J., Matas, J., ... & Uchida, S. (2017). *ICDAR2017 robust reading challenge on multi-lingual scene text detection and script identification – RRC-MLT*. In *14th IAPR International Conference on Document Analysis and Recognition* (pp. 1454–1459). IEEE.
- [144] Speechmatics. (n.d.). *Accuracy benchmarking: How to calculate Word Error Rate*. Speechmatics Docs. <https://docs.speechmatics.com/speech-to-text/accuracy-benchmarking>
- [145] Brown, L., et al. (2024). *AI-augmented screening for dyscalculia: Early detection techniques*. *Educational Psychology Review*, 18(2), 120–135.
- [146] Singh, G., et al. (2022). *Artificial intelligence enabled hybrid machine learning application for dyslexia detection using optimized multiclass support vector machine and personalized interactive and assistive tools using adaptive reinforcement*. In *International.
- [147] Govindan, V. K., & Shivaprasad, A. P. (1990). *Character Recognition – A review*. *Pattern Recognition*, 23(7), 671–683. [https://doi.org/10.1016/0031-3203\(90\)90091-X](https://doi.org/10.1016/0031-3203(90)90091-X)
- [148] Dumka, P. (2022). *Understanding sets with the help of Python*. *International Journal of Novel Research and Development (IJNRD)*.
- [149] Tiego, J., Testa, R., Bellgrove, M. A., Pantelis, C., & Whittle, S. (2018). *A hierarchical model of inhibitory control*. *Frontiers in psychology*, 9, Article 1339. <https://doi.org/10.3389/fpsyg.2018.01339>
- [150] Berninger, V. W., & Richards, T. L. (2010). *Brain literacy for educators and psychologists*. Academic Press.
- [151] Smith-Spark, J. H., & Fisk, J. E. (2007). *Working memory deficits in developmental dyslexia: Role of executive processes*. *Brain and Cognition*, 65(1), 1–8. <https://doi.org/10.1016/j.bandc.2006.05.007>

- [152] Arivazhagan, M., Srinivasan, H., & Srihari, S. N. (2007). *A statistical approach to line segmentation in handwritten documents*. Center of Excellence for Document Analysis and Recognition (CEDAR), University at Buffalo, State University of New York.
<https://cedar.buffalo.edu/~srihari/papers/SPIE-2007-lineSeg.pdf>
- [153] Violentis, A. (2025). writing_LD_detection. GitHub.
https://github.com/juundayo/writing_LD_detection
- [154] Violentis, A. (2025). TextLineSegmentation. GitHub.
<https://github.com/juundayo/TextLineSegmentation>
- [155] Brown, L., et al. (2024). *AI-augmented screening for dyscalculia: Early detection techniques*. *Educational Psychology Review*, 18(2), 120–135.
- [156] Singh, G., et al. (2022). *Artificial intelligence enabled hybrid machine learning application for dyslexia detection using optimized multiclass support vector machine and personalized interactive and assistive tools using adaptive reinforcement*. In *International*