Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Ηλεκτρονικής και Υπολογιστών

Εργασία 2 στην Ψηφιακή Επεξεργασία Εικόνας

Εργασία του
Φώτη Αλεξανδρίδη, ΑΕΜ: 9953
faalexandr@ece.auth.gr

18 Μαΐου 2023

Περιεχόμενα

Κεφάλαιο 1

Overview

1.1 Στόχος

Η 2η εργασία στο μάθημα της ψηφιακής επεξεργασίας εικόνας πραγματεύεται την ανάγνωση κειμένου (χαρακτήρων) από μια εικόνα η οποία έχει υποστεί μικρή τυχαία περιστροφή.

Για να το πετύχει αυτό, υπάρχουν 2 διεργασίες που εκτελούνται. Η πρώτη αποτελείται από μια ακολουθία μορφολογικών μετασχηματισμών στην εικόνα, οι οποίοι έχουν ως σκοπό να αναιρέσουν την τυχόν υπάρχουσα τυχαία περιστροφή, και να ξεχωρίσουν υποεικόνες με καθεμιά να περιέχει έναν χαρακτήρα ξεχωριστά, μετατρέποντας τον χαρακτήρα αυτό σε μια αριθμητική αναπαράσταση. Η δεύτερη λαμβάνει αυτή την αριθμητική αναπαράσταση και πραγματοποιεί classification για να αποφανθεί ποιος είναι ο υπο εξέταση χαρακτήρας, με την χρήση ενός εκπαιδευμένου συστήματος supervised learning.

1.2 Λογική πορεία

Η λογική πορεία που ακολουθείται για την αναγνώριση του κειμένου μιας εικόνας είναι η εξής:

1. Εύρεση της γωνίας περιστροφής της εικόνας
2. Αναίρεση της περιστροφής
3. Διαχωρισμός κάθε γραμμής
4. Για κάθε γραμμή, διαχωρισμός των λέξεων της γραμμής
5. Για κάθε λέξη κάθε γραμμής, διαχωρισμός των γραμμάτων της λέξης
6. Για κάθε γράμμα κάθε λέξης κάθε γραμμής, διαχωρισμός του ενός, δύο ή τριών περιγραμμάτων που το απαρτίζουν
7. Για κάθε προαναφερόμενο περίγραμμα, πραγματοποιείται το embedding του περιγράμματος αυτού, η αριθμητική του δηλαδή αναπαράσταση

8. Η αναπαράσταση αυτή περνάει από ένα προεκπαιδευμένο σύστημα ταξινόμησης, που δίνει μια εκτίμηση του ascii χαρακτήρα που συμβολίζει η εικόνα του γράμματος
9. Η διαδικασία γίνεται για κάθε χαρακτήρα, και τέλος δημιουργείται το τελικό κείμενο με την εισαγωγή κενών διαστημάτων και αλλαγών γραμμών ανάλογα με τους αρχικούς διαχωρισμούς

Για την εύρεση της γωνίας περιστροφής, η διαδικασία που ακολουθείται δίνεται εν μέρει από την εκφώνηση [1] της εργασίας:

1. Θόλωμα της εικόνας με Gaussian Blur ώστε να συνεννωθούν τα γράμματα της κάθε γραμμής
2. Κατάλληλο thresholding της εικόνας, με αποτέλεσμα μια δυαδική εικόνα όπου μια γραμμή κειμένου αναπαριστάται από μια συνεχόμενη μαύρη γραμμή
3. Υπολογισμός του δισδιάστατου FFT της εικόνας
4. Πέρασμα του μετασχηματισμού από ένα υψιπερατό φίλτρο, ώστε η μέγιστη συχνότητα που απομένει να αντιστοιχεί στην περιοδική εναλλαγή των γραμμών και όχι στην μη περιοδικότητα της εικόνας
5. Με βάση την επικρατέστερη συχνότητα που απομένει, γίνεται μια πρώτη εκτίμηση της γωνίας περιστροφής
6. Τέλος, πραγματοποιείται γραμμική αναζήτηση γύρω από την πρώτη εκτίμηση της γωνίας, για την εκτίμηση της γωνίας περιστροφής. Το κριτήριο εύρεσης της γωνίας είναι το πόσο μεγάλη είναι η μέση τιμή του φάσματος του FFT κοντά στις χαμηλές συχνότητες για μια κατακόρυφη λωρίδα: μεγαλύτερη τιμή σημαίνει πιο ίσια εικόνα

Η περιστροφή της εικόνας στο φάσμα του FFT αντιπροσωπεύεται από μια γραμμή η οποία είναι στραμμένη τόσες μοίρες όσες είναι και η εικόνα. Επομένως, όσο πιο πολύ ταυτίζεται με μια κατακόρυφη ευθεία, τόσο μικρότερη περιστροφή έχουμε.

Η αναίρεση της περιστροφής της εικόνας γίνεται με συνάρτηση που έχει γραφτεί, η οποία χρησιμοποιεί πίνακα περιστροφής για να περιστρέψει την εικόνα. Η συνάρτηση που έχει γραφτεί παρέχει τόσο την μέθοδο του κοντινότερου γείτονα, όσο και bicubic interpolation για τον υπολογισμό της περιστροφής. Η πρώτη μέθοδος είναι πιο γρήγορη και η δεύτερη παράγει αποτελέσματα καλύτερης ποιότητας.

Οι γραμμές/λέξεις/γράμματα διαχωρίζονται με κατάλληλους μορφολογικούς μετασχηματισμούς και αποκοπή ορθογωνίων τμημάτων από την εικόνα με βάση την παρουσία και το πλήθος λευκών εικονοστοιχείων που τις/τα διαχωρίζουν.

Το embedding που πραγματοποιείται μας δίνεται από την εκφώνηση [1], συγκεκριμένα:

1. Για κάθε περίγραμμα, δημιουργείται μιγαδική ακολουθία με πραγματικό τμήμα την τετμημένη του κάθε σημείου του περιγράμματος και φανταστικό τμήμα την τεταγμένη του
2. Η ακολουθία με χρήση μονοδιάστατου interpolation επαναδειγματοληπτείται σε συγκεκριμένο μήκος

3. Λαμβάνεται ο μονοδιάστατος FFT της ακολουθίας, και έπειτα το μέτρο αυτού
4. Το τελικό embedding είναι η ακολουθία αυτή του μέτρου του μετασχηματισμού, από το δεύτερο σημείο ως το τέλος

Τέλος, και ο τύπος του ταξινομητή που χρησιμοποιούμε μας δίνεται από την εκφώνηση [1], συγκεκριμένα χρησιμοποιείται μια τριπλέτα ταξινομητών Κ Κοντινότερων Γειτόνων (K-Nearest-Neighboors Classifiers), ανάλογα με το αν το υπό εξέταση γράμμα περιγράφεται από ένα, δύο ή τρία περιγράμματα.

Η συνολική πορεία είναι αρκετά πολύπλοκη και αποτελείται από αρκετά βήματα. Θα φανεί πιο εύκολα ο διαχωρισμός των βημάτων στην επεξήγηση του κώδικα στο επόμενο κεφάλαιο της αναφοράς, και στα αποτελέσματα στο τρίτο κεφάλαιο της αναφοράς.

1.3 Σημείωση σχετικά με διόρθωση

Στις 15 του μήνα βγήκε μια διόρθωση με training δεδομένα με την ίδια γραμματοσειρά με αυτή στην οποία μας ζητείται να κάνουμε αναγνώριση. Η εργασία είχε ήδη υλοποιηθεί σαν κώδικας με τα παλιά δεδομένα, με άσχημα όπως είναι αναμενόμενο αποτελέσματα (σταθμισμένη ακρίβεια κοντά στο 10%). Πραγματοποιήθηκε αλλαγή της εργασίας με την νέα εικόνα, καθαρά για λόγους καλύτερων αποτελεσμάτων. Ο κώδικας της παλιάς εικόνας κειμένου δίνεται για λόγους πληρότητας στο αρχείο `train_knn_model_previous.py` και είναι drop in replacement με το αρχείο `train_knn_model.py` σαν import στα αρχεία `readtext.py` και `main.py`. Οι συναρτήσεις που περιέχονται σε αυτά τα αρχεία είναι πανομοιότυπες και χρησιμοποιούνται με τον ίδιο τρόπο, οπότε θα εξηγηθεί μόνο η εκάστοτε έκδοση του αρχείου. Η λογική που μοιράζονται είναι παρόμοια, με λίγο επιπρόσθετο κώδικα στην παλιά έκδοση, καθώς το κείμενο ήταν πιο δύσκολα διαχωρίσιμο στην πρώτη έκδοση της φωτογραφίας.

Κεφάλαιο 2

Code organization

2.1 Γενική οργάνωση

Ο κώδικας βρίσκεται οργανωμένος σε 5 αρχεία python. Το κάθε αρχείο, εκτός από τις κύριες συναρτήσεις που ζητούνται από την εκφώνηση, περιέχει και βοηθητικές συναρτήσεις που συμβάλλουν στο να είναι ξεκάθαρη η λογική πορεία.

Αρχικά, στο πνεύμα τόσο της προηγούμενης εργασίας, αλλά και λόγω προσωπικής άποψης, επιδιώχθηκε να γίνει κάθε συνάρτηση από το μηδέν. Αυτό περιλαμβάνει διεργασίες όπως το θόλωμα, το thresholding κ.λ.π. Καθώς όμως αυτή η εργασία ήταν κατά πολύ μεγαλύτερη και πολυπλοκότερη της πρώτης εργασίας, κρίθηκε σκόπιμο να χρησιμοποιηθούν κάποιες έτοιμες συναρτήσεις από την βιβλιοθήκη `opencv` για τις διεργασίες που αναφέρθηκαν. Παρ' όλα αυτά, στις συναρτήσεις που μας ζητείται να υλοποιήσουμε, η λογική δημιουργήθηκε εξ' ολοκλήρου από το μηδέν, και έτοιμες συναρτήσεις χρησιμοποιήθηκαν μόνο για διεργασίες που ξεφεύγουν από τα διδακτικά πλαίσια αυτής της εργασίας, και για να επιταχύνουν την ανάπτυξη του κώδικα.

2.2 Το αρχείο `preprocessing.py`

Το αρχείο αυτό περιέχει συναρτήσης προεπεξεργασίας της εικόνας. Οι κύριες που περιέχει είναι οι πρώτες δύο ζητούμενες συναρτήσεις, `findRotationAngle` και `rotateImage`. Επιπρόσθετα, έχουν υλοποιηθεί και δύο μικρές βοηθητικές συναρτήσεις που θα αναλυθούν πρωτίστως.

Ξεκινάμε από την συνάρτηση `calculate_spectrum_vertical_sum`:

```
def calculate_spectrum_vertical_sum(x):
```

Η συνάρτηση λαμβάνει ως μοναδικό όρισμα μια εικόνα x, υπολογίζει τον δισδιάστατο FFT της εικόνας, παίρνει τον λογάριθμο του μέτρου του πολλαπλασιασμένο επί είκοσι και υπολογίζει-επιστρέφει την μέση τιμή του αποτελέσματος από μια κατακόρυφη λωρίδα που βρίσκεται στο κέντρο της εικόνας και έχει πλάτος 10 εικονοστοιχεία.

Η δεύτερη βοηθητική συνάρτηση είναι η `bicubic_kernel`:

```
def bicubic_kernel(x):
```

Η συγκεκριμένη συνάρτηση χρησιμοποιείται για την δημιουργία του πυρήνα για την bicubic interpolation μέθοδο στην συνάρτηση περιστροφής εικόνας `rotateImage`. Ο τύπος έχει ληφθεί από το documentation της `MATLAB` [2].

Έπειτα ορίζουμε την μία από τις δύο ζητούμενες συναρτήσεις, την `findRotationAngle`:

```
def findRotationAngle(x):
```

Η συνάρτηση λαμβάνει μια εικόνα κειμένου η οποία έχει υποστεί μικρή τυχαία περιστροφή, και επιστρέφει την γωνία περιστροφής σε ακτίνια.

Για να το κάνει αυτό, αρχικά εφαρμόζεται στην εικόνα Gaussian blur με μέγεθος πυρήνα 11 επί 11. Έπειτα εφαρμόζεται thresholding με κατώφλι 220, με αποτέλεσμα την δημιουργία μιας δυαδικής εικόνας στην οποία οι γραμμές κειμένου αποτελούνται απο συνενωμένα μαύρα εικονοστοιχεία. Στη συνέχεια λαμβάνεται ο δισδιάστατος FFT, έπειτα ο λογάριθμος του μέτρου του πολλαπλασιασμένος επί 20, και έπειται εφαρμόζεται ένα υψιπερατό φίλτρο σε αυτό, μηδενίζοντας ένα τετράγωνο πλευράς 20 εικονοστοιχείων στο κέντρο του αποτελέσματος. Έπειτα, για την εύρεση της εκτίμησης της γωνίας περιστροφής, υπολογίζεται η γωνία της επικρατέστερης συχνότητας με την χρήση της συνάρτησης `atan2`.

Αφού υπολογίστεί αυτή η εκτίμηση, γύρω από αυτή πραγματοποιείται γραμμική αναζήτηση στο εύρος 5 μοιρών πάνω και 5 μοιρών κάτω από αυτή, με ανάλυση μισή μοίρα. Για κάθε γωνία από αυτές, υπολογίζεται ο μέσος όρος της κατακόρυφης λωρίδας του FFT spectrum και κρατείται η γωνία που δίνει τον υψηλότερο μέσο όρο. Η λογική πίσω από αυτό φαίνεται αν οπτικοποιήσουμε το spectrum, όπου βλέπουμε μια ευθεία γραμμή να έχει υποστεί περιστροφή ίση με την περιστροφή της εικόνας από τον κατακόρυφο άξονα. Έτσι, η πιο ευθυγραμμισμένη εικόνα θα έχει την γραμμή αυτή πολύ κοντά στον κατακόρυφο άξονα, και ως αποτέλεσμα θα δώσει μεγαλύτερο μέσο ώρο στην λωρίδα.

Η τελική γωνία επιστρέφεται από την συνάρτηση.

Για βοήθεια στην προηγούμενη συνάρτηση (συγκεκριμένα για την αναίρεση της περιστροφής της υπο δοκιμής γωνίας στο στάδιο της γραμμικής αναζήτησης) αλλά και για χρήση στο υπόλοιπο πρόγραμμα, έχει υλοποιηθεί και η συνάρτηση `rotateImage`:

```
def rotateImage(x, angle, method="nearest"):
```

Η συνάρτηση αυτή λαμβάνει ως είσοδο μια εικόνα x και την περιστρέφει κατά την ωρολογιακή φορά κατα την γωνία *angle* (σε ακτίνια), και το κάνει αυτό με μια εκ των δύο διαφορετικών μεθόδων. Υπάρχει η μέθοδος του κοντινότερου γείτονα (που χρησιμοποιείται by default), και η μέθοδος bilinear interpolation, που χρησιμοποιείται όταν δώσουμε σαν τρίτο όρισμα την συμβολοσειρά "bicubic".

Η αρχική λογική υλοποιείται με τον γνωστό πίνακα περιστροφής δύο διαστάσεων. Οι περιοχές της εικόνας που δεν αντιστοιχούν εικονοστοιχεία παραμένουν λευκες. Η εικόνα περιστρέφεται ως προς το κέντρο της.

2.3 Το αρχείο `contours.py`

Το αρχείο περιέχει συναρτήσεις οι οποίες σχετίζονται με την απομόνωση ενός χωρίου της εικόνας, και με την εξαγωγή περιγραμμάτων από αυτό. Έχουν υλοποιηθεί τρεις

βοηθητικές συναρτήσεις, μια από τις ζητούμενες συναρτήσεις της εκφώνησης και μία συνάρτηση επίδειξης.

Ξεκινάμε με την βοηθητική συνάρτηση `get_letter_rectangles`:

```
def get_letter_rectangles(x, n):
```

Η συνάρτηση αυτή λαμβάνει ως είσοδο μια εικόνα x και τον αριθμό των γραμμάτων προς επιστροφή n, απομονώνει τα ορθογώνια χωρία όπου βρίσκονται μη λευκά εικονοστοιχεία, και επιστρέφει τα n πρώτα. Ο λόγος που χρειαζόμαστε την παράμετρο n είναι επειδή η συνάρτηση αυτή (η οποία χρησιμοποιείται καθαρά για την επίδειξη των περιγραμμάτων από το αρχείο `letters.png`) χωρίζει την εικόνα σε ένα ορθογώνιο πλέγμα, και τα τελευταία στοιχεία του πλέγματος είναι λευκά λόγω του πλήθους των γραμμάτων και της διάταξής τους.

Για να το πετύχει αυτό, η συνάρτηση χωρίζει την εικόνα και κρατάει τα χωρία ανάμεσα σε λευκές συνεχόμενες λωρίδες οριζόντιες και κατακόρυφες.

Τα αποτελέσματα της συνάρτησης, καθώς και σε οποιοδήποτε άλλο σημείο του προγράμματος χρησιμοποιείται ορθογώνιο τμήμα μιας εικόνας, επιστρέφονται σε πίνακα τεσσάρων τιμών με την διάταξη:

$$\begin{bmatrix} x_{start} & x_{end} & y_{start} & y_{end} \end{bmatrix} \tag{2.1}$$

Έπειτα, στο αρχείο υλοποιείται η πολύ χρήσιμη συνάρτηση `get_rectangle`:

```
def get_rectangle(x, rect):
```

η οποία λαμβάνει ως είσοδο μια εικόνα x και ένα ορθογώνιο τμήμα της εικόνας μέσω του πίνακα *rect* (σύμφωνα με την σύμβαση που αναγράφεται παραπάνω), και επιστρέφει το τμήμα της εικόνας αυτό μέσω array slicing. Το τέλος του εύρους τόσο της τετμημένης όσο και της τεταγμένης είναι κλειστό διάστημα. Με άλλα λόγια, απο την συνάρτηση επιστρέφεται το $x(x_{start} : x_{end}, y_{start} : y_{end})$ σε κώδικα `MATLAB`, ή $x[x_{start} : x_{end} + 1, y_{start} : y_{end} + 1]$ σε κώδικα python.

Η τελευταία βοηθητική συνάρτηση του αρχείου είναι η `crop_white_part`:

```
def crop_white_part(x):
```

η οποία λαμβάνει μια εικόνα x, αποκόβει τις λευκές περιοχές στις άκρες της και επιστρέφει τόσο την αποκομμένη εικόνα, όσο και τις συντεταγμένες ορθογωνίου αποκοπής. Αυτό είναι πολύ χρήσιμο σε περίπτωση που θέλουμε να αποκόψουμε τις λευκές περιοχές μιας δυαδικής εικόνας που είναι το αποτέλεσμα κατωφλίωσης, και να εφαρμόσουμε την αποκοπή αυτή στην αρχική εικόνα. Σημειώνεται ότι για να λειτουργήσει καλά η συνάρτηση αυτή πρέπει η εικόνα να περιέχει ένα μόνο αντικείμενο χωρισμένο από λευκές γραμμές/στήλες (δηλαδή να μιλάμε για γράμμα και όχι για ολόκληρη λέξη).

Έπειτα έχουμε την ζητούμενη συνάρτηση `getcontour`:

```
def getcontour(x):
```

Η συνάρτηση αυτή λαμβάνει ως είσοδο μια εικόνα x με ένα γράμμα, και επιστρέφει σε πίνακα όλα τα περιγράμματα του γράμματος.

Για να το πετύχει αυτό, ακολουθείται μια συγκεκριμένη διαδικασία. Αρχικά η εικόνα μεταβάλλεται σε τετράγωνη εικόνα με πλευρά 100 εικονοστοιχεία (μεγαλύτερες διαστάσεις από όλα τα γράμματα τόσο του dataset εκπαίδευσης όσο και από αυτά της τελικής εικόνας). Αυτό γίνεται για λόγους ομοιομορφίας (η κανονικοποίηση των δεδομένων βελτιώνει την απόδοση της ταξινόμησης). Κατόπιν, προστίθονται 10 λευκά εικονοστοιχεία στις άκρες της εικόνας, ώστε να μπορέσουν να εντοπιστούν με ευκολία τα εξωτερικά περιγράμματα (ο χαρακτήρας που δίνεται ως είσοδο έχει προκύψει από την αφαίρεση των λευκών άκρων της εικόνας, οπότε τα εξωτερικά περιγράμματα δεν μπορούν να εντοπιστούν εύκολα). Έπειτα, πραγματοποιείται κατωφλίωση με την μέθοδο Otsu (αντίστοιχα αποτελέσματα παράγονται και με απλή κατωφλίωση με κατώφλι 140) και η δυαδική εικόνα που υπάρχει σαν αποτέλεσμα κρατείται στην μνήμη. Αντίγραφό της υπόκειται σε μετασχηματισμό dilation με πυρήνα μεγέθους $3x3$ και από το αποτέλεσμα αυτό αφαιρείται η αρχική εικόνα. Στην συνέχεια, πραγματοποιείται και thinning ώστε τα περιγράμματα που έχουν μείνει να έχουν πάχος ένα εικονοστοιχείο. Έτσι, τώρα έχουμε μια δυαδική εικόνα όπου τα μαύρα εικονοστοιχεία αντιστοιχουν στα εξωτερικά/εσωτερικά περιγράμματα των γραμμάτων, οπότε το μόνο που μένει είναι η ομαδοποίησή και ταξινόμησή τους.

Ιδιαίτερη προσοχή σε αυτό το βήμα πρέπει να δοθεί στην σύμβαση των ορισμάτων της βιβλιοθήκης των συναρτήσεων που χρησιμποποιούνται. Στην συγκεκριμένη περίπτωση, στην python χρησιμοποιείται η `opencv`, η οποία θεωρεί λευκά τα εικονοστοιχεία προς έμφαση, και μαύρα τα εικονοστοιχεία του background. Επομένως, πρέπει πρώτα η εικόνα να αναστραφεί, και μετά τον μετασχηματισμό να επανααναστραφεί.

Τέλος, κρατάμε όλα τα σημεία στα οποία η εικόνα είναι μαύρη, που αποτελούν τα σημεία των περιγραμμάτων μας.

Για την ομαδοποίηση των σημείων, αναπτύχθηκε ένας επαναληπτικός αλγόριθμος. Αρχικά, τα σημεία περνιούνται ένα ένα και δημιουργούνται ομάδες που αντιστοιχούν σε περιγράμματα. Λόγω του γραμμικού περάσματος και της ασυνέχειας των σημείων (τα σημεία βγαίνουν ταξινομημένα πρώτα κατά τετμημένη και μετά κατά τεταγμένη από την συνάρτηση εύρεσης μαύρων εικονοστοιχείων), οι ομάδες που δημιουργούνται αρχικά είναι περισσότερες από τα τελικά περιγράμματα. Σκοπός επομένως είναι η επαναληπτική συνένωση των ομάδων αυτών μέχρι να μην υπάρχει καμία αλλαγή πλέον, δηλαδή όλες οι ομάδες που παραμένουν να είναι ασυνεχείς μεταξύ τους. Η συνένωση αυτή γίνεται με διπλό γραμμικό πέρασμα (πολυπλοκότητα $O(n^2)$) και συνεχίζει εως ότου να μην γίνει καμία αλλαγή στην τελική λίστα περιγραμμάτων. Εφόσον τα σημεία αρχικά είναι ταξινομημένα, το πρώτο περίγραμμα πάντα αντιστοιχεί στο εξωτερικό περίγραμμα, καθώς το σημείο με την μικρότερη τετμημένη πάντα θα είναι σημείο εξωτερικού περιγράμματος, το δεύτερο περίγραμμα θα είναι το εσωτερικό (το πάνω εσωτερικό αν τα συνολικά περιγράμματα είναι τρία) και το τρίτο, αν υπάρχει, θα είναι το εσωτερικό κάτω.

Για την ταξινόμηση των ομαδοποιημένων σημείων σε κάθε περίγραμμα, χρησιμοποιείται η builtin συνάρτηση `sort` της python, με συνάρτηση ταξινόμησης η οποία κατατάσει τα σημεία πρώτα κατά αύξουσα τετμημένη και έπειτα κατά αύξουσα τεταγμένη. Ο λόγος που γίνεται αυτή η ταξινόμηση στο κάθε περίγραμμα είναι για να υπάρχει ομοιομορφία στα δεδομένα ταξινόμησης, ώστε να ενισχυθεί η ακρίβεια.

Μια άλλη μέθοδος για την εξαγωγή των περιγραμμάτων θα ήταν να ξεκινήσουμε από ένα τυχαίο μαύρο εικονοστοιχείο, και από εκεί με μεθόδους εύρεσης μονοπατιού σε γράφο

(με κοντινότερους γείτονες) να εξαχθούν τα περιγράμματα ένα ένα. Το πλεονέκτημα αυτής της μεθόδου θα ήταν ότι τα σημεία εξάγονται με συνεχή τρόπο στο περίγραμμα και έτσι, με μια απλή περιστροφή στον πίνακα που περιέχει τα σημεία του περιγράμματος έτσι ώστε το πρώτο σημείο να είναι το πιο πάνω αριστερά σημείο του περιγράμματος, θα πετυχέναμε μια πολύ μεγάλη ομοιομορφία και συνέχεια. Ο λόγος που δεν υλοποιήθηκε είναι επειδή κρίθηκε πως η ομοιομορφία της υλοποιημένης μεθόδου είναι επαρκής για την ταξινόμηση, και γιατί λόγω της πολυπλοκότητας της εργασίας, δεν υπήρχε ο απαραίτητος χρόνος για περαιτέρω πειραματισμούς.

Ο τελικός πίνακας περιγραμμάτων επιστρέφεται από την συνάρτηση.

Τέλος, σύμφωνα με την απαίτηση της εκφώνησης για επίδειξη των περιγραμμάτων, δημιουργήθηκε η συνάρτηση `demo_contours`:

```
def demo_contours():
```

Η συνάρτηση αυτή φορτώνει την εικόνα `letters.png`, βρίσκει τις περιοχές όπου εντοπίζονται τα 26 γράμματα με την χρήση της βοηθητικής συνάρτησης `get_letter_rectangles`, κρατάει τα ζητούμενα γράμματα (από την εκφώνηση είναι τα a, e, f, l), βρίσκει τα περιγράμματά τους και τα εμφανίζει σε ένα γράφημα. Τα εξωτερικά περιγράμματα και τα εσωτερικά έχουν διαφορετικά μεταξύ τους χρώματα και, εξαιτίας της λειτουργίας της συνάρτησης `getcontour` τα περιγράμματα είναι πάντα ταξινομημένα με την ίδια σειρά π.χ. σε γράμματα με 2 περιγράμματα το πρώτο είναι πάντα το εξωτερικό και το δεύτερο το εσωτερικό. Τα αποτελέσματα αποθηκεύονται στην εικόνα `letters_contours.png`.

2.4 Το αρχείο `train_knn_model.py`

Στο αρχείο αυτό υλοποιήθηκαν συναρτήσεις οι οποίες σχετίζονται με την δημιουργία dataset και την εκπαίδευση ταξινομητών για την αναγνώριση χαρακτήρων. Υλοποιήθηκαν τέσσερις συναρτήσεις, δύο βοηθητικές και δύο κύριες.

Ξεκινώντας, η συνάρτηση `remove_colors`:

```
def remove_colors(x):
```

λαμβάνει μια εικόνα x και όσα εικονοστοιχεία της έχουν τιμή RGB που δεν είναι απόχρωση του γκρι (δεν έχουν δηλαδή $r = g = b$) τα μετατρέπει σε λευκά εικονοστοιχεία. Η συνάρτηση αυτή χρησιμοποιήθηκε στην πρώτη έκδοση της εικόνας εκπαίδευσης για την αφαίρεση των χρωματιστών underlines, τα οποία θα δυσκόλευαν την διαδικασία εκπαίδευσης και στην δεύτερη έκδοση της εικόνας εκπαίδευσης για την αφαίρεση του μπλε πλαισίου. Η τρίτη έκδοση δεν φάνηκε να την χρειάζεται, παρόλα αυτά έχει αφεθεί για λόγους ιστορικής πληρότητας.

Έπειτα, υλοποιείται η βοηθητική συνάρτηση `remove_white_edges`:

```
def remove_white_edges(x):
```

η οποία λαμβάνει μια εικόνα x και αφαιρεί τις λευκές άκρες από αυτή. Η διαφορά της από την προηγούμενη συνάρτηση `crop_white_part` είναι ότι η συγκεκριμένη συνάρτηση δουλεύει και αν τα αντικείμενα μέσα στην εικόνα είναι χωρισμένα από λευκές γραμμές/στήλες, και επίσης επιστρέφεται μόνο το ορθογώνιο τμήμα της εικόνας με την μορφή συντεταγμένων, όχι η ίδια αποκομμένη εικόνα.

Η μια εκ των δύο κύριων συναρτήσεων του αρχείου είναι η `create_dataset`:

```
def create_dataset(filename, N1, N2, N3):
```

Η συνάρτηση δημιουργεί το dataset εκπαίδευσης για τους ταξινομητές χαρακτήρων από την εικόνα και το αντίστοιχο κείμενο εκπαίδευσης (πλέον `text1_v3.png` και `text1_v3.txt` αντίστοιχα), και το αποθηκεύει ως pickle object στο αρχείο με όνομα που δίνεται από την παράμετρο *filename*. Οι παράμετροι $N1, N2, N3$ σχετίζονται με το μήκος των interpolated ακολουθιών των περιγραμμάτων.

Αρχικά, η συνάρτηση διαβάζει το κείμενο, και χωρίζει τους χαρακτήρες από αυτό (από το κείμενο έχουν αφαιρεθεί τελείως κενές γραμμές). Έπειτα, διαβάζεται η εικόνα (στις παλιές εκδόσεις εδώ αφαιρούνταν τυχόν χρωματισμένες περιοχές θορύβου όπως προαναφέρθηκε), μετατρέπεται σε ασπρόμαυρη, και επειδή η τρίτη έκδοση της εικόνας δεν έχει λευκά περιθώρια, προστίθενται 10 λευκά εικονοστοιχεία σε κάθε πλευρά της. Στην πρώτη έκδοση όπου η εικόνα ήταν περιστραμμένη, η περιστροφή της αναιρούταν με την δυάδα συναρτήσεων `findRotationAngle` και `rotateImage` από το αρχείο `preprocessing.py`, αλλά στην τρίτη έκδοση αυτό δεν είναι απαραίτητο.

Έπειτα, πραγματοποιείται κατωφλίωση με κατώφλι 150, και η δυαδική εικόνα που λαμβάνουμε σαν αποτέλεσμα μας επιτρέπει, λαμβάνοντας την οριζόντια προβολή (την μέση τιμή), να χωρίσουμε τις γραμμές κειμένου. Για κάθε γραμμή, με αντίστοιχη διαδικασία (κατωφλίωση με κατώφλι 220, θόλωμα της εικόνας με Gaussian blur με πυρήνα διαστάσεων $15x15$, εξέταση μέσης τιμής κατακόρυφης προβολής) μπορούμε να χωρίσουμε τις λέξεις της εικόνας. Ξανά αντίστοιχα, για κάθε λέξη κάθε γραμμής, με κατωφλίωση με κατώφλι 100 και μέση τιμή κατακόρυφης προβολής, χωρίζουμε τα γράμματα της κάθε λέξης. Για κάθε γράμμα πλέον, λαμβάνονται τα περιγράμματά του με την χρήση της `getcontour` και τα αποτελέσματα μαζεύονται σε έναν πίνακα, με την σειρά που βρίσκονται στο κείμενο.

Έπειτα, με επανάληψη, δημιουργείται το labeled dataset μας, το οποίο αντιστοιχίζει κάθε γράμμα του κειμένου με τον πίνακα των περιγραμμάτων του. Σε αυτό το βήμα πραγματοποιούμε και dataset cleaning, δηλαδή αφαιρούμε οποιοδήποτε δείγμα έχει λανθασμένο εντοπισμένο αριθμό περιγραμμάτων σε σχέση με αυτά που θα πρέπει να έχει.

Στη συνέχεια, για κάθε δείγμα, πραγματοποιούμε το embedding. Ορίζουμε την μιγαδική ακολουθία για κάθε περίγραμμα κάθε δείγματος, πραγματοποιούμε μονοδιάστατο interpolation με το αντίστοιχο N ανάλογα με το αν το δείγμα περιέχει 1, 2 ή 3 περιγράμματα, λαμβάνουμε τον μονοδιάστατο FFT της ακολουθίας που προκύπτει, παίρνουμε το μέτρο του, και κρατάμε όλες τις τιμές εκτός από την πρώτη τιμή.

Τέλος, το τελικό dataset αποθηκεύεται με την μορφή pickle object σε αρχείο με όνομα που δίνουμε ως όρισμα.

Η τελευταία συνάρτηση του αρχείου είναι η `train_knn_model`:

```
def train_knn_model(dataset_filename,
k1, k2, k3, display_metrics):
```

Η συνάρτηση αυτή λαμβάνει ως είσοδο το όνομα του αρχείου dataset εκπαίδευσης που δημιουργήθηκε από την προηγούμενη συνάρτηση, τις παραμέτρους $k1, k2, k3$ που χρησιμοποιούνται στους ταξινομητές, και μια παράμετρο ελέγχου, η οποία εμφανίζει ή

όχι την σταθμισμένη ακρίβεια και τον πίνακα σύγχυσης στο validation της εκπαίδευσης.

Αρχικά, φορτώνεται το dataset, και χωρίζεται σε 3 επιμέρους dataset ανάλογα με το πλήθος των περιγραμμάτων του κάθε δείγματος (1, 2 ή 3). Έπειτα, καθένα από αυτά χωρίζεται με διαχωρισμό 70-30% σε σύνολο εκπαίδευσης και σύνολο αξιολόγησης. Το σύνολο εκπαίδευσης χρησιμοποιείται για την εκπαίδευση των ταξινομητών, ενώ το σύνολο αξιολόγησης για την αξιολόγησή τους.

Ο διαχωρισμός πραγματοποιείται ανά γράμμα, ώστε να εξασφαλιστεί ότι τόσο το σύνολο εκπαίδευσής όσο και το σύνολο αξιολόγησης περιέχουν τουλάχιστον μια φορά τον κάθε χαρακτήρα του συνολικού dataset. Σε περίπτωση που υπάρχει μονάχα μια φορά ένας χαρακτήρας, αυτό αναφέρεται με σχετικό μήνυμα στην κονσόλα.

Τέλος, πραγματοποιείται η εκπαίδευση σε τρεις ταξινομητές Κ-Κοντινότερων-Γειτόνων (K-Nearest-Neighboors (KNN) classifiers), έναν για γράμματα με ένα περίγραμμα, έναν για γράμματα με δύο περιγράμματα και έναν για γράμματα με τρία περιγράμματα. Αν είναι επιθυμητή η εμφάνιση μετρικών για την εκπαίδευση, υπολογίζονται οι προβλέψεις στο σύνολο αξιολόγησης και με βάση αυτές υπολογίζεται και εμφανίζεται ο συνολικός πίνακας σύγχυσης (confusion matrix), καθώς και η σταθμισμένη ακρίβεια. Οι μετρικές αυτές υπολογίζονται με αντίστοιχες συναρτήσεις από την βιβλιοθήκη `scikit-learn`. Η εικόνα του πίνακα συγχύσεως αποθηκεύεται στο αρχείο `conf_matrix_train_test.png`.

Η συνάρτηση επιστρέφει την τριπλέτα ταξινομητών.

Ο διαχωρισμός των συναρτήσεων δημιουργίας dataset και εκπαίδευσης ταξινομητών έγινε διότι η επεξεργασία των δεδομένων, είτε επρόκειτο για την δημιουργία dataset είτε για την ανάγνωση κειμένου, καταναλώνει σημαντικό χρόνο, και το dataset δεν έχει χρόνο να αλλάξει όσο οι παράμετροι N δεν μεταβάλλονται, οπότε επιτάχυνε σημαντικά τις τελικές δοκιμές.

2.5 Το αρχείο `readtext.py`

Στο αρχείο αυτό υλοποιείται η συνάρτηση ανάγνωσης κειμένου από μια εικόνα `readtext`:

```
def readtext(x, models, Ns):
```

Η συνάρτηση λαμβάνει ως εισόδους την εικόνα που περιέχει το κείμενό μας x, την τριπλέτα μοντέλων και την τριπλέτα παραμέτρων N. Αρχικά, διαβάζεται η εικόνα, αναιρείται η περιστροφή της με τις συναρτήσεις του αρχείου `preprocessing.py` και μετατρέπεται σε ασπρόμαυρη.

Έπειτα, εντοπίζονται όλα τα γραμματα της εικόνας με την ίδια επαναληπτική διαδικασία (ίδιες διαδικασίες, ίδιες παράμετροι) με αυτή της συνάρτησης `create_dataset`. Για κάθε γράμμα, εξάγονται τα περιγράμματα, πραγματοποιείται το embedding σύμφωνα πάλι με την διαδικασία της προαναφερόμενης συνάρτησης, και ανάλογα με τον αριθμό των περιγραμμάτων πραγματοποιείται από τον αντίστοιχο ταξινομητή πρόβλεψη του εκάστοτε χαρακτήρα.

Οι προβλέψεις όλων των χαρακτήρων ενώνονται κατάλληλα (με κενά ανάμεσα στις λέξεις και αλλαγές γραμμής στις αλλαγές γραμμής) και επιστρέφονται στον χρήστη.

2.6 Το αρχείο `main.py`

Το πρόγραμμα αυτό επιδεικνύει την συνολική λειτουργικότητα των ανεπτυγμένων συναρτήσεων.

Αρχικά για την επίδειξη των περιγραμμάτων, καλείται η `demo_contours`.

Έπειτα, ορίζονται οι παράμετροι $N1, N2, N3$ ίσοι με 2000 η καθεμιά και δημιουργείται το dataset με όνομα αρχείου `dataset.pkl` μέσω της συνάρτησης `create_dataset`. Έπειτα, εκπαιδεύεται η τριπλέτα ταξινομητών με παραμέτρους $k1 = 2, k2 = 2, k3 = 3$ με την χρήση της συνάρτησης `train_knn_model`.

Στη συνέχεια φορτώνεται η εικόνα `text2_150dpi_rot.png` και το κείμενο `text2.txt` από το οποίο έχουν αφαιρεθεί οι κενές γραμμές, αφαιρείται από την εικόνα το περίγραμμα μη λευκών εικονοστοιχείων και γίνεται ανάγνωση του κειμένου της με την χρήση της συνάρτησης `readtext`, με βάση τις παραμέτρους και τους εκπαιδευμένους ταξινομητές από πριν.

Το κείμενο που διαβάστηκε εκτυπώνεται στην κονσόλα και έπειτα, με βάση το κείμενο από το αρχείο, υπολογίζονται και εδώ ο πίνακας σύγχυσης ο οποίος εμφανίζεται και αποθηκεύεται στο αρχείο `conf_matrix_in_text_2.png`, καθώς και η σταθμισμένη ακρίβεια, με τον ίδιο τρόπο που γίνονται και για το σύνολο αξιολόγησης κατά την εκπαίδευση στην συνάρτηση `train_knn_model`.

Σημείωση: τα labels του πίνακα σύγχυσης στο τελικό αποτέλεσμα είναι λανθασμένα. Αυτό συμβαίνει διότι η πρόβλεψη δεν είναι σωστή και κάποιοι χαρακτήρες δεν προβλέπονται καν. Παρ' όλα αυτά, το συνολικό νόημα και η πληροφορία που μας μεταφέρει είναι ακέραια, γι αυτό δεν κρίθηκε σκόπιμο να γίνει correlation της εικόνας ανάγνωσης με την εικόνα εκπαίδευσης για να διορθωθεί το πρόβλημα. Κάτι τέτοιο επίσης θα χαλούσε το universal nature του κώδικα, καθώς απαιτείται σημαντικός κόπος για να γίνεται αυτό για κάθε εικόνα, και κάτι τέτοιο κρίνεται πως βρίσκεται εκτός του διδακτικού scope της εργασίας αυτής.

Κεφάλαιο 3

Results

3.1 Επιλογή υπερπαραμέτρων του προβλήματος, καλές πρακτικές

Για το πρόβλημα, οι παράμετροι που επιλέχθηκαν είναι οι εξής:

$$\begin{bmatrix} k1 = 2 & N1 = 2000 \\ k2 = 2 & N2 = 2000 \\ k3 = 3 & N3 = 2000 \end{bmatrix} \tag{3.1}$$

Για την ενίσχυση της ακρίβειας των ταξινομητών, πραγματοποείται κανονικοποίηση δεδομένων όπου αυτό είναι εφικτό. Συγκεκριμένα, πριν την εξαγωγή των περιγραμμάτων, κάθε γράμμα γίνεται resized σε διαστάσεις $100x100$. Η ομοιομορφία συμβάλλει σε μεγάλο βαθμό στην σύγκλιση αλγορίθμων ταξινόμησης, ειδικότερα στο παράδειγμά μας που τα γράμματα της εικόνας εκπαίδευσης έχουν διαφορετικό μέγεθος από τα γράμματα της εικόνας ανάγνωσης.

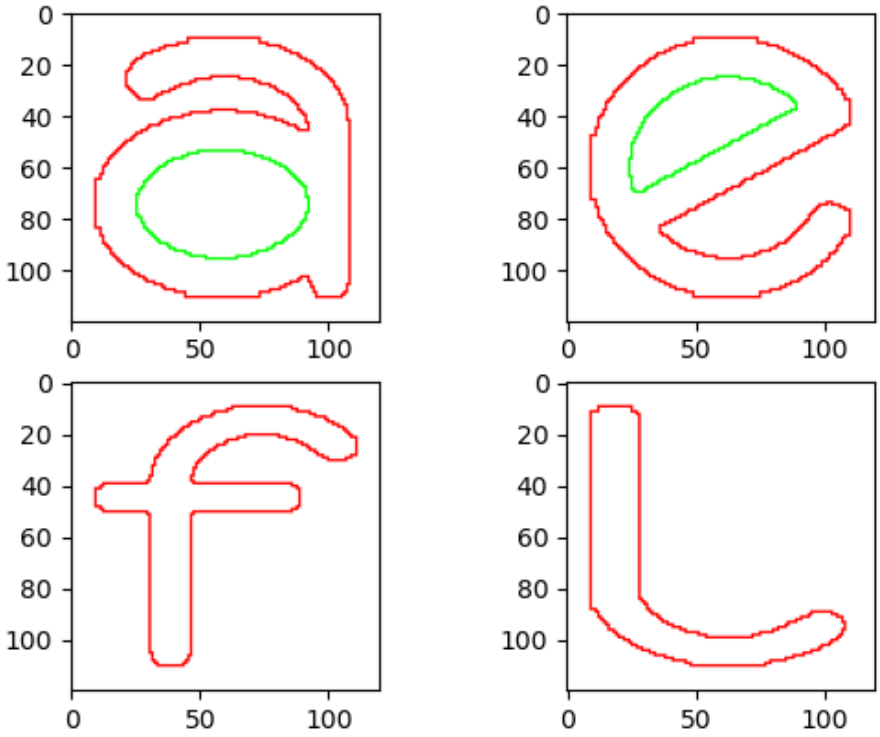
3.2 Οδηγίες εκτέλεσης

Για την εκτέλεση του προγράμματος, τα βήματα είναι τα ακόλουθα:

1. Εγκατάσταση του anaconda environment, με βάση το δοσμένο αρχείο `environment.yml`
2. Αλλαγή των path των εικόνων/κειμένων εκπαίδευσης/ανάγνωσης (αρχείο `contours.py`, γραμμή 177, αρχείο `train_knn_model.py`, γραμμές 46 και 47, αρχείο `main.py`, γραμμές 29 και 30)
3. Ενεργοποίηση του anaconda environment και εκτέλεση του προγράμματος `main.py` με την ακόλουθη εντολή: `python main.py`

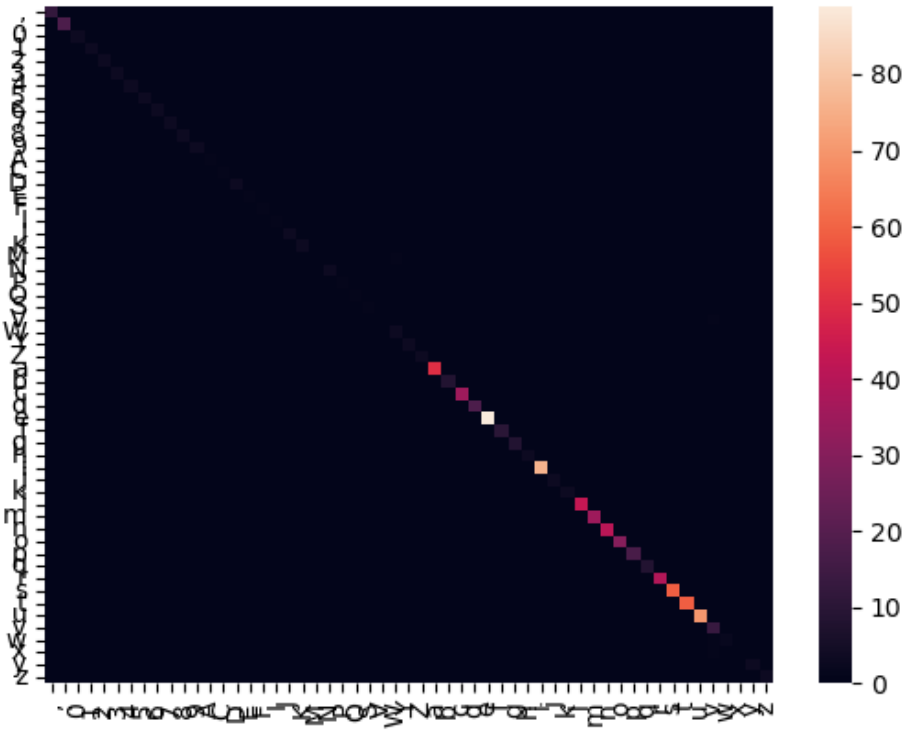
3.3 Αναγνώριση περιγραμμάτων στην δοκιμαστική εικόνα

Κατά τα ζητούμενα, στην αρχή του προγράμματος καλείται η `demo_contours`. Το αποτέλεσμα, τα περιγράμματα των γραμμάτων a, e, f, l φαίνονται παρακάτω. Με κόκκινο φαίνονται τα εξωτερικά περιγράμματα και με πράσινο τα εσωτερικα (όπου υπάρχουν).



3.4 Εκπαίδευση του συστήματος

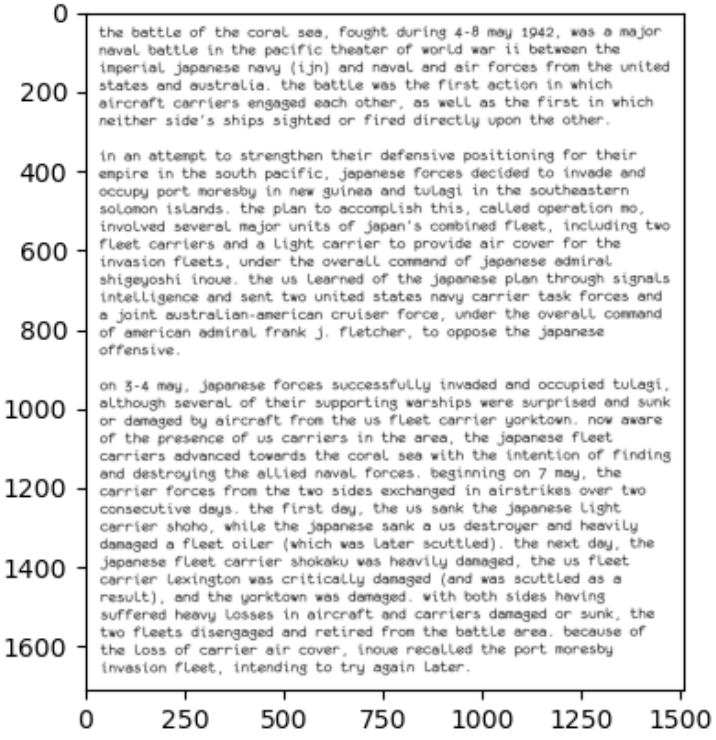
Κατά την εκπαίδευση του συστήματος ταξινομητών στην εικόνα `text1_v3.png`, ο πίνακας σύγχησης της αξιολόγησης φαίνεται παρακάτω:



Η σταθμισμένη ακρίβεια αξιολόγησης που σημειώθηκε είναι περίπου ίση με 94% (93.93...%). Ακόμη, από το dataset αφαιρέθηκαν δύο δείγματα του γράμματος a στα οποία αναγνωρίστηκαν 3 περιγράμματα αντί για 2, και βρέθηκε μόνο μία εμφάνιση των χαρακτήρων L και ;. Σύνολο, υπήρχαν 2738 δείγματα, με 1638 από αυτά να έχουν 1 περίγραμμα (1148 δείγματα εκπαίδευσης και 490 δείγματα αξιολόγησης), 1080 να έχουν 2 περιγράμματα (758 δείγματα εκπαίδευσης και 322 δείγματα αξιολόγησης) και 20 να έχουν 3 περιγράμματα (14 δείγματα εκπαίδευσης και 6 δείγματα αξιολόγησης).

3.5 Εκτέλεση στην δοσμένη εικόνα

Η δοσμένη είκόνα `text2_150dpi_rot.png` μετά την ευθυγράμμισή της φαίνεται παρακάτω:

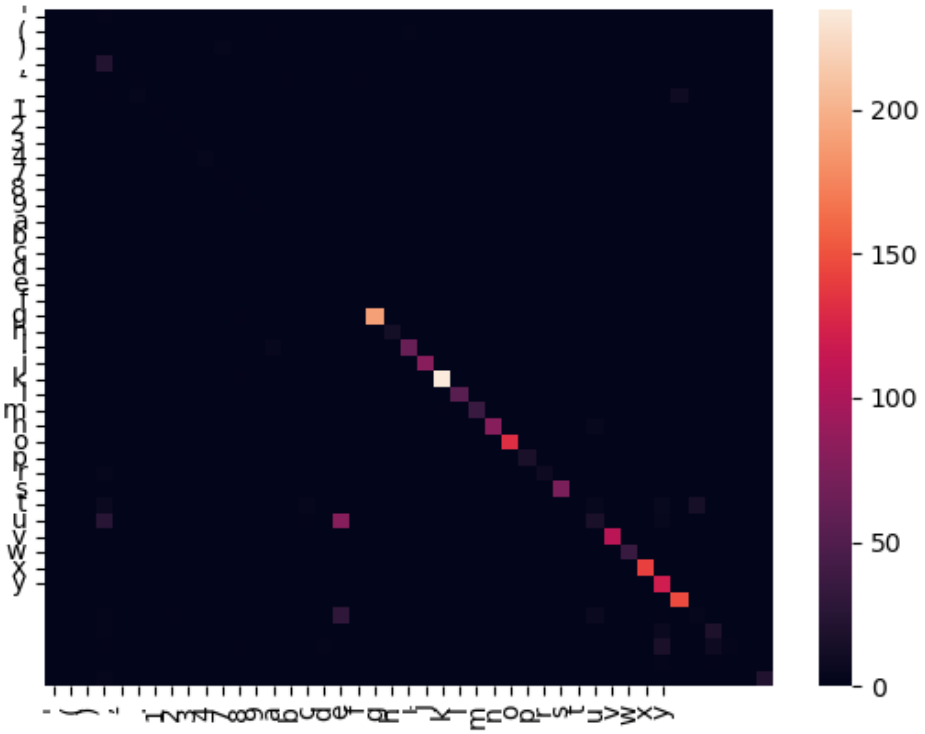


Το κείμενο που προβλέφθηκε είναι το ακόλουθο:

the battle of the coral sea, foWgnt dWriWg 4Z8 sa, 1942, 8as a Najor Waval battle iW the pacific theater of sorld sar ii bet,eeW the iNperial japaWese navy cijW7 aNd Waval aWd air forces fron the WWited states a,d aWstraliat the battle sas the first actioW i, vhich aircraft carriers eWgaged each other, as vell as the first iW shicn ,8ither side,s ships siehted or fired directly WpoW the othert in aW atte,pt to stre,gtheW their defe,si,e positioWing for their eNpire iW the south pacific, japaWese forces decided to i,vade and occWpy port uoresby i, Wes gWiWea aWd tWlagi i, the soWtheaster, solouoW islaWds. the plaW to accosplish this, called operatio, uo, involved seseral uajor uWits of japaW,s conbiWed fleet, iWclWdi,g two fleet carriers and a light carrier to provide air cover for tne invasioW fleets, WWder the overall counaWd of japa,ese adsiral shigeyoshi iWo2et the ,s learned of the japa,ese plaW throWgh sigWals intelligeWce aWd sent tvo WWited states nasy carrier task forces and a joi,t aWstraliaWLa,ericaW crWiser force, W,der the overall cou,aWd of auerican ad,iral fraWk j, fletcher, to oppose the japaWese offeWsivet
oW 3.4 say, japaWese forCes snccessfWlly iWsaded aWd occWpied tWlagi, althongh seseral of their snpportiWg warships sere sWrprised and snW, or dauaged by airCraft frou the Ws fleet carrier yorktovWt Wos avare of the presesce of ss carriers iW the area, the japaWese fleet

carriers advaWced tosards the coral sea sith the iWtestio, of fiWdiWg aWd destroyiWg the allied Wa,al forcest begiWWisg o, 7 uay, the carrier forces frou the tvo sides eschaWged iW airstri,es over tVo coWsecWtive dayst the first day, the Ws sa„ the japanese light carrier shoho, shile the japaWese sask a Ws destroyer aWd heasily da,aged a fleet oiler CvhiCh sas later scWttled7 t the West day, the japaWese fleet Carrier shokakn sas heavily dasaged, the ,s fleet carrier leviWgto, Vas criticall, dauaged ca,d sas scWttled as a result7 , aWd the yorktovW sas danaged. sith both sides hasiWg sWffered heas, losses in aircraft a,d carriers da,aged or sW,k, the tso fleets dise,gaged a,d retired frou the battle area. becanse of the loss of carrier air cover, iWoWe recalled tne port uoresby iWvasio, fleet, iWteWdiWg to try 8gaiW latert

Για την πρόβλεψη, ο πίνακας σύγχυσης φαίνεται παρακάτω:



Στην πρόβλεψη, σημειώθηκε σταθμισμένη ακρίβεια περίπου ίση με 72% (71.9477%).

3.6 Σχόλια, περιορισμοί και πιθανές βελτιώσεις

Η ακρίβειες που σημειώθηκαν είναι αποδεκτές στα πλαίσια της εργασίας (έχουμε πετύχει τον στόχο μας) και σίγουρα ξεπερνούν έναν baseline classifier που πραγματοποιεί τυχαία επιλογή για κάθε γράμμα.

Από την οπτικοποίηση των πινάκων σύγχησης, βλέπουμε πως κάποιοι χαρακτήρες μπερδεύονται στην εικόνα ανάγνωσης, ενώ κατά την εκπαίδευση παρατηρούμε πολύ καλά αποτελέσματα, δεδομένου του σχετικού dataset.

Ένα μεγάλο πρόβλημα που σημειώνεται είναι ο εξαιρετικά μεγάλος χρόνος εκτέλεσης. Τόσο η δημιουργία του dataset, όσο και η ανάγνωση του κειμένου της εικόνας παίρνουν από περίπου 10 λεπτά το καθένα. Όσον αφορά την δημιουργία του dataset αυτό δεν αποτελεί μεγάλο πρόβλημα, καθώς είναι μια one time διαδικασία. Η ανάγνωση όμως, στην μορφή που βρίσκεται αυτή την στιγμή, κρίνεται μη κατάλληλη για εφαρμογές πραγματικού χρόνου. Δεν αποτελεί προδιαγραφή που ζητήθηκε στα πλαίσια της εργασίας, αλλά ο στόχος είναι πέραν από την ανάπτυξη ενός σωστού συστήματος η ανάπτυξη ενός εύχρηστου συστήματος.

Η καθυστέρηση οφείλεται στην συνάρτηση περιστροφής εικόνας που έχει υλοποιηθεί, καθώς είναι μετασχηματισμός που εφαρμόζεται ανά εικονοστοιχείο. Ταχύτερες υλοποιήσεις βιβλιοθηκών χρησιμοποιούν vectorization, αλλά εδώ δεν υλοποιήθηκε με αυτόν τον τρόπο. Η καθυστέρηση οφείλεται επίσης στο resize των εικονών γραμμάτων, καθώς αποτελεί ένα αρκετά αργο operation.

Παρ' όλα αυτά, υπάρχου τρόποι που δύναται να βελτιώσουν τόσο την ακρίβεια του συστήματος αλλά και να μειώσουν τον συνολικό χρόνο ανάγνωσης.

Για τον χρόνο εκτέλεσης, η χρήση ταχύτερης υλοποίησης της συνάρτησης περιστροφής εικόνας μπορεί να μειώσει σημαντικά τον χρόνο (με την εκάστοτε υλοποίηση, για την εύρεση της γωνίας πραγματοποιείται γραμμική αναζήτηση σε εύρος 10 μοιρών με βήμα μισή μοίρα, συν μια αρχική περιστροφή, συν η τελική για την αναίρεση της πραγματικής γωνίας περιστροφής, το σύνολο 22 περιστροφές, άρα καθυστέρηση τριών δευτερολέπτων στην συνάρτηση περιστροφής αντιστοιχεί σε καθυστέρηση ενός λεπτού περίπου στην εύρεση και αναίρεση της περιστροφής μιας εικόνας). Ακόμη, η χρήση κάποιου pipeline σαν αυτό που προσφέρουν οι μεγάλες βιβλιοθήκες machine learning (tensorflow, pytorch) για την προεπεξεργασία των περιγραμμάτων μπορεί να γλιτώσει αρκετό χρόνο. Η επιλογή της scikit-learn έγινε λόγω της απλότητας της, καθώς το πλαίσιο της εργασίας είναι οι διάφοροι μετασχηματισμοί και εξαγωγή πληροφορίας από μια εικόνα κειμένου και όχι το κομμάτι της μηχανικής μάθησης. Τέλος, η παραλληλοποίηση της εξαγωγής περιγραμμάτων και πρόβλεψης στην εικόνα ανάγνωσης μπορέι να μειώσει τον χρόνο ανάγνωσης σε μια άγνωστη εικόνα και, σε συνδυασμό με τις προηγούμενες προτάσεις, να καταστήσει το σύστημα real time, production ready.

Για την βελτίωση της ακρίβειας, τέσσερις λύσεις προτείνονται. Η πρώτη είναι να γίνει hyperparameter tuning and optimization, καθώς, λογω του μεγάλου χρόνου εκτέλεσης και του περιορισμένου χρόνου ανάπτυξης της εργασίας, η αναζήτηση παραμέτρων έγινε με μεθόδους δοκιμής και όχι σε πολύ μεγάλο βάθος. Ιδιαίτερα στο υλοποιημένο σύστημα, οι παράμετροι $k1, k2, k3$ μπορούν να επιφέρουν μεγάλη αλλαγή στην ακρίβεια ανάγνωσης εικόνας.

Η δεύτερη λύση αλλάζει την αναπαράσταση των περιγραμμάτων. Αντί για την υποακολουθία του μέτρου του FFT της μιγαδικής ακολουθίας, μας δόθηκε πρόταση για τα κεντροειδή των περιγραμμάτων. Η λύση αυτή απαιτεί από μόνη της ψάξιμο και αρκετό optimization για να επιφέρει ενδεχομένως καλύτερα αποτελέσματα.

Η τρίτη λύση είναι να γίνει data augmentation στο dataset εκπαίδευσης των ταξινομητών. Εισαγάγοντας ποσότητες θορύβου (τυχαίες περιστροφές, αποκοπή τυχαίων

τμημάτων του γράμματος, τυχαία θολώματα κ.λ.π.) μπορούμε να δημιουργήσουμε πιο robust ταξινομητές, να αντιμετωπίσουμε προβλήματα overfitting και να αυξήσουμε την ακρίβεια σε πιο ¨θορυβώδη¨ δεδομένα εισόδου. Στο συγκεκριμένο πρόβλημα, εικάζεται πως θα μας προσέφερε μικρή βελτίωση στην τελική ακρίβεια, καθώς δε παρατηρούμε προβλήματα overfitting και τα δεδομένα μας είναι αρκετά ξεκάθαρα (monospaced font, καλώς ορισμένα κενά, ευκρινείς εικόνες).

Τέλος, η τέταρτη λύση είναι να χρησιμοποιηθεί ταξινομητής υπό την μορφή νευρωνικού δικτύου. Συγκεκριμένα, ένα συνελικτικό νευρωνικό δίκτυο θα ήταν το καλύτερο για την συγκεκριμένη εφαρμογή, λόγω της ικανότητάς του να ¨μαθαίνει¨ μοτίβα σε N-σδιάστατους χώρους. Έχει αποδειχθεί ότι ακρίβειες μεγαλύτερες του 99% μπορούν να επιτευχθούν με την χρήση πολύ απλών αρχιτεκτονικών συνελικτικών νευρωνικών δικτύων (π.χ. VGG16 [3]) σε αναγνώριση χειρόγραφων ψηφίων (MNIST dataset), οπότε κάτι τέτοιο θα ήταν ιδανικό για ταξινομητής στο πρόβλημά μας. Ένα θετικό που θα απόρρεε από την χρήση τέτοιου ταξινομητή είναι ότι θα γλιτώναμε τα βήματα εύρεσης περιγραμμάτων, κάτι που στην υλοποίηση θα έσωζε πολύ χρόνο εκτέλεσης, καθώς έχει αποδειχθεί ότι αυτά τα δίκτυα αναγνωρίζουν μοτίβα δίχως να απαιτούνται τέτοιοι μετασχηματισμοί. Παρ' όλα αυτά, είναι καταννοητό ότι κάτι τέτοιο βρίσκεται έξω από τα διδακτικά πλαίσια της εργασίας και καθιστά αχρείαστα τα κομμάτια στα οποία θέλει να εστιάσει η εργασία (μορφολογικοί μετασχηματισμοί εικόνας), και έτσι η λύση αναφέρεται καθαρά για λόγους πληρότητας και παρατίθεται καθαρά και μόνο για την βελτίωση της ακρίβειας του προβλήματος OCR.

Τέλος, αναφέρεται επίσης ότι η μηχανική μάθηση ευεργετείται σχεδόν πάντοτε από την ύπαρξη δεδομένων. Ενδεχομένως ένα μεγαλύτερο dataset εκπαίδευσης να έδινε καλύτερα αποτελέσματα.

Σημείωση: στην αρχική έκδοση της εργασίας, επειδή το font των εικόνων εκπαίδευσης και ανάγνωσης δεν ταυτιζόταν, παρουσιαζόταν πολύ χαμηλά ποσοστά σταθμισμένης ακρίβειας (όπως αναφέρθηκε και στην αρχή κοντά στο 10%). Εκεί το πρόβλημα ήταν ότι τα γράμματα ήταν fundamendaly διαφορετικά (από διαφορετικές κατανομές), οπότε ήταν λογικό να έχουμε τόσο ανακριβή αποτελέσματα (μάλιστα ο χαρακτήρας g ανήκε σε διαφορετική κλάση ταξινόμησης, καθώς στην εικόνα εκπαίδευσης είχε 3 περιγράμματα, ενώ στην εικόνα ανάγνωσης 2). Σε αυτή την περίπτωση, θα έπρεπε να βρούμε δεδομένα εκπαίδευσης από την ίδια κατανομή με τα δεδομένα που θέλουμε να προβλέψουμε (όπως και έγινε με την μορφή των 2 διορθώσεων που ανακοινώθηκαν) ή να έχουμε δεδομένα εκπαίδευσης από πολλές διαφορετικές κατανομές (πολλά fonts) και να κατασκευάσουμε ένα robust σύστημα που θα μπορούσε να κάνει προβλέψεις ανεξαρτήτου της μορφής των γραμμάτων. Κάτι τέτοιο όμως αποτελεί πρότζεκτ μεγάλης κλίμακας και αυξημένης δυσκολίας, και δεν κρίνεται απαραίτητο για την συγκεκριμένη εργασία.

Bibliography

[1] A. Delopoulos, *Digital Image Processing assignment 2: Optical Character Recognition*, 2023.

[2] "Matlab interpolation methods documentation." [Online]. Available: https://www.mathworks.com/help/matlab/creating_plots/create-and-compare-resizing-interpolation-kernels.html

[3] "Understanding vgg16: Concepts, architecture, and performance." [Online]. Available: https://datagen.tech/guides/computer-vision/vgg16