Ψηφιακή Επεξεργασία Εικόνας

Εργασία 2-Οπτική αναγνώριση χαρακτήρων

Μιχάλης Δαδόπουλος ΑΕΜ:9989

Στην παρούσα εργασία περιγράφετε η διαδικασία που ακολούθησα και ο κώδικας που εκτέλεσα για την αυτόματη, οπτική αναγνώριση χαρακτήρων.

Αρχικά διαβάζοντας την εκφώνηση της εργασίας αναζήτησα βιβλιοθήκες για να με βοηθήσουν να υλοποιήσω τους κατάλληλους μετασχηματισμούς στην εικόνα.Η πιο χρήσιμη βιβλιοθήκη που βρήκα είναι η cv2 της opencv με πολλές χρήσιμες συναρτήσεις.

Αποφάσισα να δουλέψω την εργασία σε jupyter notebook γιατί με βοηθάει να οργανώσω και να τεστάρω καλύτερα τον κώδικα μου αλλά καθώς η συνάρτηση cv2.imshow δεν δουλεύει σε jupyter αποφάσισα να δουλέψω στο google colab όπου υπάρχει παραλλαγή της συνάρτησης cv2_imshow.Αργότερα μετέφερα τον κώδικα μου σε αρχεία python στο VS ,ωστόσο αυτό είναι το λινκ από το colab που δούλευα όπου τρέχει κανονικά ο κώδικας εφόσον ανεβούν στον φάκελο τα κατάλληλα αρχεία εικόνων και κειμένων:
https://colab.research.google.com/drive/1Dd7BhozTjiquLs_QKwzIwrJW8hRvwrRG .

Το πρώτο κομμάτι της εργασίας δεν ήταν ιδιαίτερα απαιτητικό εφόσον είχα βρει τις κατάλληλες συναρτήσεις και τις είχα κατανοήσει και καθώς βρήκα παρόμοια παραδείγματα για την υλοποίηση των συναρτήσεων αναζήτησης γωνίας και περιστροφής της περιεστρεμμένης εικόνας

Ένα από τα δυσκολότερα κομμάτια της εργασίας ήταν η αναγνώριση σωστά των λέξεων της κάθε γραμμής με το ίδιο τρόπο που διαβάζονται από ένα αρχείο κειμένου ώστε να γίνει αντιστοίχιση 1-1 των περιγραφέων των γραμμάτων που υπολογίζονται με το αντιστοίστιχο χαρακτήρα του γράμματος. Η δυσκολία οφειλόταν στο γεγονός ότι έπρεπε να βρω ένα κατώφλι για το κενό μεταξύ των λέξεων τέτοιο ώστε να συμπεριλαμβάνεται και οι χαρακτήρες “.,()” μαζί με τη λέξη το οποίο

έπρεπε να είναι μικρότερο από το κενό μεταξύ των λέξεων. Μετά από διάφορες δοκιμές κατέληξα στις κατάλληλες τιμές. Μια ιδιαίτερη δυσκολία που αντιμετώπισα αργότερα ήταν να γενικεύσω αυτά τα όρια για κάθε εικόνα. Έτσι μετά από αρκετές δοκιμές κατέληξα με έναν συντελεστή που προκύπτει με βάση το μέγεθος των εικόνων και δίνεται σαν παράμετρος στις συναρτήσεις που έχουν κάποιο κατώφλι το οποίο πολλαπλασιάζεται με αυτόν τον συντελεστή. Δουλεύει για όλες τις εικόνες εκτός από την text1_rotated.png όπου γράφω σε σχόλια που χρειάζεται αλλαγή σε ένα μόνο κατώφλι.

Μια άλλη δυσκολία που αντιμετώπισα είναι με τα περιγράμματα των γραμμάτων .Για την εικόνα text1.png υπήρχαν ιδιαίτερες δυσκολίες λόγων των υπογραμμισμένων λέξεων και κάποια γράμματα επέστρεφαν σαν ένα από την συνάρτηση segment_characters λόγω πολύ μικρής απόστασης μεταξύ τους ή λόγω της υπογράμμισης. Έπρεπε λοιπόν να τσεκάρω πόσα περιγράμματα επέστρεφαν από την συνάρτηση getcontour και να τα διασχίσω όλα και μέσω της ιεραρχίας τους που επιστρεφόταν ώστε να ανακτήσω όλους τους περιγραφείς για την τάξη στην οποία ανήκει το κάθε γράμμα. Επίσης με τον ορισμό ενός κατωφλιού για το μέγεθος των περιγραμμάτων και την λογική ότι η getcontour επιστρέφει τα περιγράμματα από δεξιά προς τα αριστερά και κάτω προς τα πάνω της εικόνας κατάφερα να εξαμολύνω μικρά σκουπίδια όπως η γραμμή υπογράμμισης και να κρατήσω πχ την τελεία που έχουν τα γράμματα i,j.

Έπειτα έκανα δοκιμές αυτών των αρχικών συναρτήσεων για τις εικόνες τυπώνοντας απαιτούμενες τιμές ή τμημα της εικόνας για να δω που έχει θέμα ο αλγόριθμος.Στην αρχή έκανα δοκιμές με α την text1.png ωστόσο το είχε αρκετά μικρά σκουπίδια η εικόνα και το αρχείο κειμένου που είχε δεν αντιστοιχούσε σωστά οπότε έκανα κάποιες αλλαγές σε αυτό.Όπως:

Στην 5 σειρά της τελευταίας παραγράφου άλλαξα το κείμενο σε document, although όπως στην εικόνα γιατί ήταν documentjalthough .Στην 3 παράγραφο στην 1 σειρά και στην τελευταια παράγραφο 4 σειρά έβαλα κόμμα μετά το TeachText στο κείμενο καθώς στην εικόνα είχε. Στην τελευταία παράγραφο στην 1 και 2 σειρά μείωσα το κενό μεταξύ του Teach Text καθώς στην εικόνα ήταν μία λέξη.

Μετά από τις δοκιμές υλοποίησα τις συναρτήσεις για την παραγωγή της συλλογής των δεδομένων που θα χρησιμοποιούνταν για την εκπαίδευση

όπου διάβαζε ταυτόχρονα χαρακτήρες(εξάγοντας τους περιγραφείς των γραμμάτων) από την εικόνα και το κείμενο σε αναλογία 1-1 και τα πρόσθετε σαν δείγμα στην αντίστοιχη συλλογή δεδομένων ανάλογα την κλάση του κάθε γράμματος. Όσο έτρεχα αυτήν την συνάρτηση βρήκα και άλλα λάθη κάνοντας τις κατάλληλες προσαρμογές. Υλοποίησα επίσης την συνάρτηση read_text όπου με παρόμοιο σκεπτικό διαβάζει μια εικόνα και αναγνωρίζει τα γράμματα κάνοντας πρόβλεψη με τον αντίστοιχο ταξινομητή με βάση τους περιγραφείς του γράμματος που εξάγωνταν για κάθε γράμμα που αναγνωριζόταν. Στο τέλος επιστρέφει το κείμενο που κατάφερε να προβλέψει το πρόγραμμα δομημένο σε γραμμές και λέξεις.

Ένα άλλο πρόβλημα που αντιμετώπισα ήταν στην εκπαίδευση των ταξινομητών KNN για τα γράμματα κλάσεων 2 και 3 καθώς το μέγεθος των δειγμάτων δεν επέτρεπε να εκπαιδευτεί ο ταξινομητής. Οπότε αυτό που έκανα ήταν να ενώσω τους περιγραφείς του κάθε γράμματος σε ένα μεγάλο διάνυσμα ,με άλλα λόγια μείωση διάστασης.

Έπειτα εφόσον όλες οι συναρτήσεις λειτουργούσαν υλοποίησα την τελική λειτουργία του προγράμματος που απαιτούνταν και άρχισα να τεστάρω διάφορα αποτελέσματα.

Στην επεξεργασία της εικόνας δεν χρησιμοποίησα opening για καθαρισμό σκουπιδιών γιατί μου αλλοίωνε την εικόνα παραπάνω από όσο έπρεπε.

Στην main συνάρτηση του προγράμματος ορίζω σε ποια εικόνα θέλω να εκπαιδευτεί το σύστημα κα την ακρίβεια του στο τεστ σετ αυτής της εικόνας και σε ποια εικόνα να προβλέψει όλο το κείμενο αλλά και την ακρίβεια των ταξινομητών σε αυτήν.Σε κάθε περίπτωση τυπώνονται η ακρίβεια του κάθε ταξινομητή και ο confusion matrix.

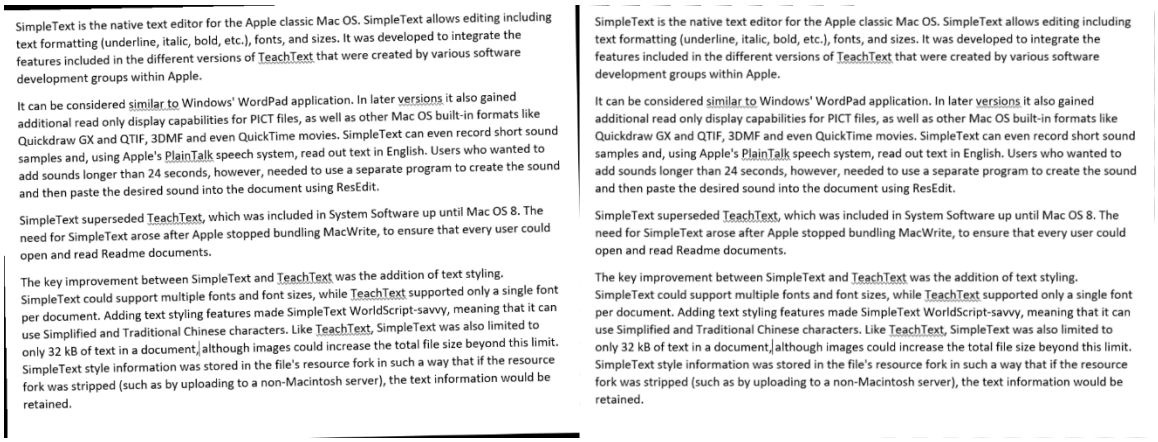
Επιπλέον σημείωση: τα αρχεία text1.txt text2.txt έχουν κωδικοποίηση utf-16 ενώ τα αρχεία text1_v2.txt,text1_v3.txt utf-8 οπότε στην main όταν ορίζονται οι εικόνες εισόδου πρέπει να ρυθμίζετε αντίσοιχα η παράμετρος encoding στην συνάρτηση create_datasets που είναι για αυτόν τον λόγο.

Παραδείγματα λειτουργίας του κώδικα

Παράδειγμα λειτουργίας της συνάρτησης findRotationAngle και rotateImage για τις 2 περιεστρεμμένες εικόνες.

Για την text1_rot.png :

```
The best angle is:  -1.0999999999998735
```



SimpleText is the native text editor for the Apple classic Mac OS. SimpleText allows editing including text formatting (underline, italic, bold, etc.), fonts, and sizes. It was developed to integrate the features included in the different versions of TeachText that were created by various software development groups within Apple.

It can be considered similar to Windows' WordPad application. In later versions it also gained additional read only display capabilities for PICT files, as well as other Mac OS built-in formats like Quickdraw GX and QTIF, 3DMF and even QuickTime movies. SimpleText can even record short sound samples and, using Apple's PlainTalk speech system, read out text in English. Users who wanted to add sounds longer than 24 seconds, however, needed to use a separate program to create the sound and then paste the desired sound into the document using ResEdit.

SimpleText superseded TeachText, which was included in System Software up until Mac OS 8. The need for SimpleText arose after Apple stopped bundling MacWrite, to ensure that every user could open and read Readme documents.

The key improvement between SimpleText and TeachText was the addition of text styling. SimpleText could support multiple fonts and font sizes, while TeachText supported only a single font per document. Adding text styling features made SimpleText WorldScript-savvy, meaning that it can use Simplified and Traditional Chinese characters. Like TeachText, SimpleText was also limited to only 32 kB of text in a document, although images could increase the total file size beyond this limit. SimpleText style information was stored in the file's resource fork in such a way that if the resource fork was stripped (such as by uploading to a non-Macintosh server), the text information would be retained.

SimpleText is the native text editor for the Apple classic Mac OS. SimpleText allows editing including text formatting (underline, italic, bold, etc.), fonts, and sizes. It was developed to integrate the features included in the different versions of TeachText that were created by various software development groups within Apple.

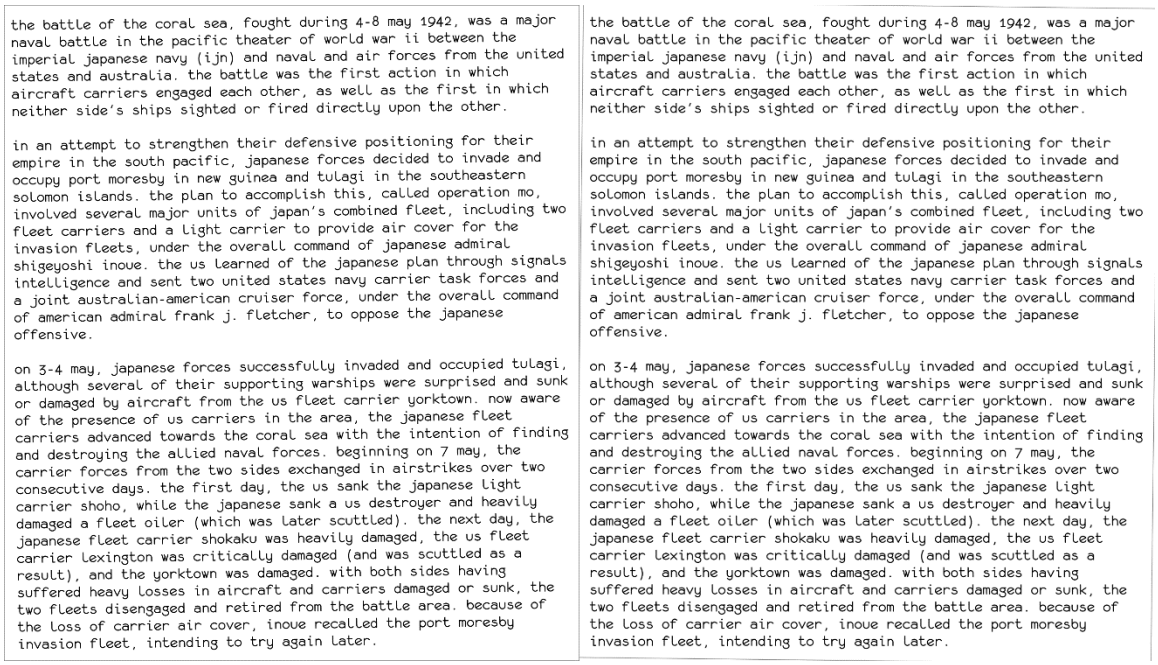
It can be considered similar to Windows' WordPad application. In later versions it also gained additional read only display capabilities for PICT files, as well as other Mac OS built-in formats like Quickdraw GX and QTIF, 3DMF and even QuickTime movies. SimpleText can even record short sound samples and, using Apple's PlainTalk speech system, read out text in English. Users who wanted to add sounds longer than 24 seconds, however, needed to use a separate program to create the sound and then paste the desired sound into the document using ResEdit.

SimpleText superseded TeachText, which was included in System Software up until Mac OS 8. The need for SimpleText arose after Apple stopped bundling MacWrite, to ensure that every user could open and read Readme documents.

The key improvement between SimpleText and TeachText was the addition of text styling. SimpleText could support multiple fonts and font sizes, while TeachText supported only a single font per document. Adding text styling features made SimpleText WorldScript-savvy, meaning that it can use Simplified and Traditional Chinese characters. Like TeachText, SimpleText was also limited to only 32 kB of text in a document, although images could increase the total file size beyond this limit. SimpleText style information was stored in the file's resource fork in such a way that if the resource fork was stripped (such as by uploading to a non-Macintosh server), the text information would be retained.

Για την text2_150dpi_rot.png:

```
The best angle is:  -0.6499999999998671
```



the battle of the coral sea, fought during 4-8 may 1942, was a major naval battle in the pacific theater of world war ii between the imperial japanese navy (ijn) and naval and air forces from the united states and australia. the battle was the first action in which aircraft carriers engaged each other, as well as the first in which neither side's ships sighted or fired directly upon the other.

in an attempt to strengthen their defensive positioning for their empire in the south pacific, japanese forces decided to invade and occupy port moresby in new guinea and tulagi in the southeastern solomon islands. the plan to accomplish this, called operation mo, involved several major units of japan's combined fleet, including two fleet carriers and a light carrier to provide air cover for the invasion fleets, under the overall command of japanese admiral shigeyoshi inoue. the us learned of the japanese plan through signals intelligence and sent two united states navy carrier task forces and a joint australian-american cruiser force, under the overall command of american admiral frank j. fletcher, to oppose the japanese offensive.

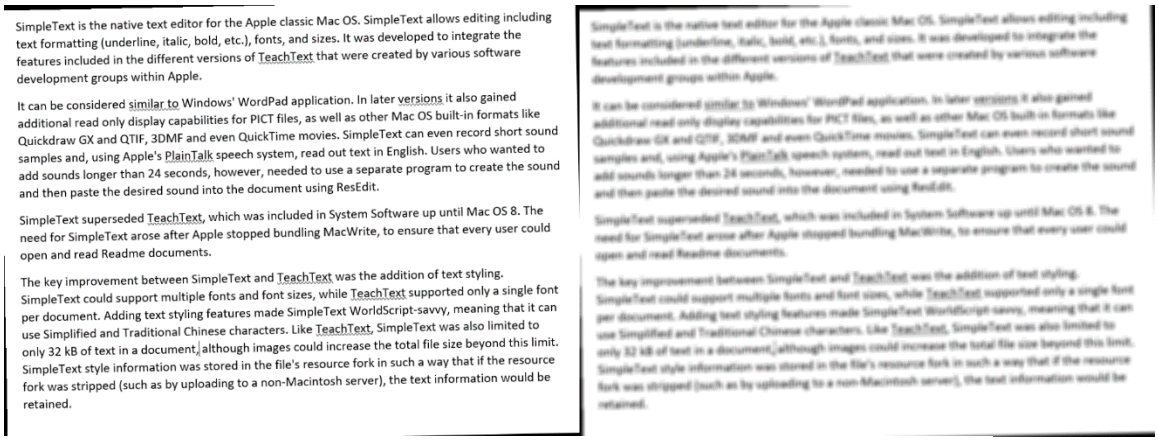
on 3-4 may, japanese forces successfully invaded and occupied tulagi, although several of their supporting warships were surprised and sunk or damaged by aircraft from the us fleet carrier yorktown. now aware of the presence of us carriers in the area, the japanese fleet carriers advanced towards the coral sea with the intention of finding and destroying the allied naval forces. beginning on 7 may, the carrier forces from the two sides exchanged in airstrikes over two consecutive days. the first day, the us sank the japanese light carrier shoho, while the japanese sank a us destroyer and heavily damaged a fleet oiler (which was later scuttled). the next day, the japanese fleet carrier shokaku was heavily damaged, the us fleet carrier lexington was critically damaged (and was scuttled as a result), and the yorktown was damaged. with both sides having suffered heavy losses in aircraft and carriers damaged or sunk, the two fleets disengaged and retired from the battle area. because of the loss of carrier air cover, inoue recalled the port moresby invasion fleet, intending to try again later.

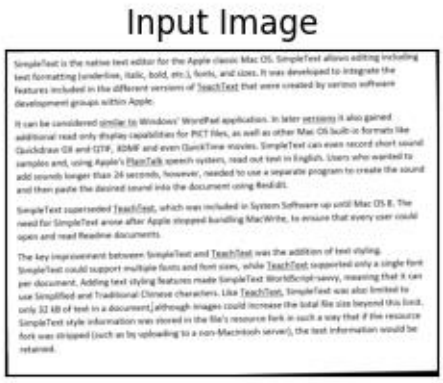
the battle of the coral sea, fought during 4-8 may 1942, was a major naval battle in the pacific theater of world war ii between the imperial japanese navy (ijn) and naval and air forces from the united states and australia. the battle was the first action in which aircraft carriers engaged each other, as well as the first in which neither side's ships sighted or fired directly upon the other.

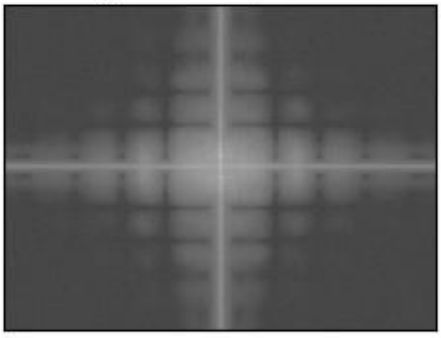
in an attempt to strengthen their defensive positioning for their empire in the south pacific, japanese forces decided to invade and occupy port moresby in new guinea and tulagi in the southeastern solomon islands. the plan to accomplish this, called operation mo, involved several major units of japan's combined fleet, including two fleet carriers and a light carrier to provide air cover for the invasion fleets, under the overall command of japanese admiral shigeyoshi inoue. the us learned of the japanese plan through signals intelligence and sent two united states navy carrier task forces and a joint australian-american cruiser force, under the overall command of american admiral frank j. fletcher, to oppose the japanese offensive.

on 3-4 may, japanese forces successfully invaded and occupied tulagi, although several of their supporting warships were surprised and sunk or damaged by aircraft from the us fleet carrier yorktown. now aware of the presence of us carriers in the area, the japanese fleet carriers advanced towards the coral sea with the intention of finding and destroying the allied naval forces. beginning on 7 may, the carrier forces from the two sides exchanged in airstrikes over two consecutive days. the first day, the us sank the japanese light carrier shoho, while the japanese sank a us destroyer and heavily damaged a fleet oiler (which was later scuttled). the next day, the japanese fleet carrier shokaku was heavily damaged, the us fleet carrier lexington was critically damaged (and was scuttled as a result), and the yorktown was damaged. with both sides having suffered heavy losses in aircraft and carriers damaged or sunk, the two fleets disengaged and retired from the battle area. because of the loss of carrier air cover, inoue recalled the port moresby invasion fleet, intending to try again later.

Επίσης παρακάτω ακολουθούν παραδείγματα θόλωσης εικόνας και του φάσματος μεγέθους του DFT







Αποτελέσματα ακρίβειας ταξινομητών

<u>Εκπαίδευση στην εικόνα text1.png ,accuracy και weighted accuracy στο test σετ για την κάθε κλάση</u> :

-Accuracy: 1:0.96,2:0.969.3:1

-Weighted accuracy: 1:0.94,2:0.96,3:1

Αξιολόγηση στην εικόνα text2.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.116,2:0.02.3:1

-Weighted accuracy: 1:0.09,2:0.4,3:1

Αξιολόγηση στην εικόνα text1_v2.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.07,2:0.0037.3:0.5

-Weighted accuracy: 1:0.03,2:0.0,3:0.25

Αξιολόγηση στην εικόνα text1_v3.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.07,2:0.0037.3:0.5

-Weighted accuracy: 1:0.06,2:0.0,3:0.25

Αξιολόγηση στην ίδια την εικόνα text1.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.97,2:0.98.3:0.95

-Weighted accuracy: 1:0.96,2:0.0,3:0.93

Εκπαίδευση στην εικόνα text1 v3.png ,accuracy και weighted accuracy στο test σετ για την κάθε κλάση :

-Accuracy: 1:0.987,2:0.95.3:1

-Weighted accuracy: 1:0.99,2:0.96,3:1

Αξιολόγηση στην εικόνα text2_150dpi_rot.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.14,2:0.4.3:1.0

-Weighted accuracy: 1:0.05,2:0.25, 3:1.0

Αξιολόγηση στην εικόνα text1.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.018,2:0.17.3:0.04

-Weighted accuracy: 1:0.00,2:0.04, 3:0.0

Αξιολόγηση στην εικόνα text1_v2.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.93,2:0.92.3:1.0

-Weighted accuracy: 1:0.94,2:0.94, 3:1.0

Εκπαίδευση στην εικόνα text2_150dpi_rot.png ,accuracy και weighted accuracy στο test σετ για την κάθε κλάση :

-Accuracy: 1:0.97,2:0.99.3:1

-Weighted accuracy: 1:0.97,2:0.99,3:1

Αξιολόγηση στην εικόνα text2_150dpi_rot.png με N=[50,25,25](ίδια αποτελέσματα για N=[80,50,50]):

-Accuracy: 1:0.98,2:0.99.3:1.0

-Weighted accuracy: 1:0.98,2:0.99, 3:1.0

Οι καλύτερες τιμές για το μήκος των περιγραφέων που βρήκα είναι N=[50,25,25] και N=[80,40,40] ,ωστόσο τα αποτελέσματα είναι ίδια και για τα δύο, ακόμα και αν τα μεγαλώσω.

Γενικά τα αποτελέσματα όταν οι ταξινομητές αξιολογούνται στην ίδια εικόνα(ακόμα και στην text2) που έχουν εκπαιδευτεί (είτε στο τεστ σετ είτε σε όλη)η ακρίβεια τους είναι πολύ κοντά στο 1.(>0.95).Για εκπαίδευση στην εικόνα text1_v3 και αξιολόγηση στην text1_v2 η ακρίβεια είναι πάλι καλή (0.92-0.94) γιατί οι εικόνες είναι παρόμοιες. Η ακρίβεια των ταξινομητών αν εκπαιδευτούν στην εικόνα text1_v3 και αξιολογηθούν στην text2_150dpi_rot.png είναι πολύ κακή(0.05-0.14 τάξη 1 ,0.25-0.4 τάξη 2 και 100 τάξη 3).

Παρατήρησα επίσης ότι πολλές φορές όταν έτρεχα τον κώδικα στο VS κάποιες φορές δεν εμφάνιζε κάποιον confusion matrix αλλά το εμφάνιζε πάντα στο colab.