ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Αναγνώριση Πινακίδων Κυκλοφορίας Αυτοκινήτων από Φωτογραφίες με Μεθόδους Μηχανικής Μάθησης και Βαθειάς Μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΑΣΤΑΣΙΟΥ Γ. ΣΕΛΑΛΜΑΖΙΔΗ

Επιβλέπων : Κωνσταντίνος Γουλιάνας
Αναπληρωτής Καθηγητής, Αλεξάνδρειο ΤΕΙ Θεσσαλονίκης

Θεσσαλονίκη, Ιούνιος 2017

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΑΛΕΞΑΝΔΡΕΙΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΕΥΦΥΕΙΣ ΤΕΧΝΟΛΟΓΙΕΣ ΔΙΑΔΙΚΤΥΟΥ - WEB INTELLIGENCE

Αναγνώριση Πινακίδων Κυκλοφορίας Αυτοκινήτων από Φωτογραφίες με Μεθόδους Μηχανικής Μάθησης και Βαθειάς Μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

ΑΝΑΣΤΑΣΙΟΥ Γ. ΣΕΛΑΛΜΑΖΙΔΗ

Επιβλέπων : Κωνσταντίνος Γουλιάνας
Αναπληρωτής Καθηγητής, Αλεξάνδρειο ΤΕΙ Θεσσαλονίκης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή στις Choose a date.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Κωνσταντίνος Γουλιάνας Αναπληρωτής Καθηγητής Α.Τ.Ε.Ι.Θ.	Χρήστος Ηλιούδης Καθηγητής Α.Τ.Ε.Ι.Θ.	Ευστάθιος Αντωνίου Αναπληρωτής Καθηγητής Α.Τ.Ε.Ι.Θ.

Θεσσαλονίκη, Ιούνιος 2017

(Υπογραφή)

.....

Αναστάσιος Γ. Σελαλμαζίδης

Μηχανικός Πληροφορικής Α.Τ.Ε.Ι.Θ.

Περίληψη

Στην παρούσα διπλωματική εργασία γίνεται μια επισκόπηση σε βασικές μεθόδους επεξεργασίας εικόνας, οι οποίες θα χρησιμοποιηθούν για τη δημιουργία αλγορίθμου για τον εντοπισμό πινακίδων κυκλοφορίας αυτοκινήτων και την εξαγωγή των χαρακτήρων από αυτές. Στη συνέχεια αναλύονται δημοφιλείς μέθοδοι μηχανικής μάθησης και βαθειάς μάθησης. Αναφέρονται επίσης τα προβλήματα με τα δημόσια διαθέσιμα σύνολα δεδομένων και για τον λόγο αυτό δημιουργείται νέο σύνολο δεδομένων από φωτογραφίες αυτοκινήτων με εμφανείς τις πινακίδες κυκλοφορίας, πάνω στο οποίο γίνεται η εκπαίδευση και αξιολόγηση των διαφόρων μοντέλων μηχανικής μάθησης.

Τα αποτελέσματα έδειξαν πως οι κλασικές μέθοδοι μηχανικής μάθησης υπερίσχυαν των μεθόδων βαθειάς μάθησης. Συγκεκριμένα η μέθοδος SVM με πυρήνα RBF απέδωσε στα πειράματα ακρίβεια 99,4%. Όσο αφορά τον αλγόριθμο εντοπισμού της πινακίδας τα αποτελέσματα δεν ήταν τα αναμενόμενα. Αν και η επεξεργασία ήταν πολύ γρήγορη, η επιτυχία που είχε ήταν μόλις 57,42%.

Λέξεις Κλειδιά: Αυτόματη Αναγνώριση Πινακίδων Κυκλοφορίας, Μηχανική Μάθηση, Επεξεργασία εικόνας, Υπολογιστική Όραση, Δίκτυα Βαθειάς Μάθησης

Η σελίδα αυτή είναι σκόπιμα λευκή.

Abstract

This diploma thesis provides an overview of basic image processing methods, which will be used to create an algorithm for license plate identification and character segmentation. Furthermore, it analyzes popular machine learning and deep learning methods. The problems with publicly available datasets are also reported and for that reason, a new dataset of car photos is created where the license plates are clearly visible. The models are trained and evaluated on this new dataset.

The results showed that the classical machine learning methods overcome deep learning methods. Specifically the SVM method with RBF kernel, yielded 99.4% accuracy. The results of the license plate identification algorithm where not as expected. Although the processing was very fast we gained an accuracy of only 57.52%.

Keywords: Automatic License Plate Recognition, Machine Learning, Image Processing, Computer Vision, Deep Learning

Στην οικογένειά μου... !!!

Πίνακας περιεχομένων

Πίνακας εικόνων

Πίνακας σχημάτων

Πίνακας πινάκων

1

Εισαγωγή

1.1 *Επεξεργασία εικόνας και αναγνώριση προτύπων*

Η ραγδαία εξέλιξη της τεχνολογίας που βιώνουμε τις τελευταίες δεκαετίες και η ανάπτυξη όλο και πιο γρήγορων υπολογιστικών συστημάτων, αλλά και η δημιουργία νέων μεθοδολογιών και αλγορίθμων στην επιστήμη των υπολογιστών, έχουν βοηθήσει στην εξέλιξη όλων των επιστημών. Από την πρόγνωση του καιρού και των σεισμών έως και τα ειδικά εφέ στις ταινίες του κινηματογράφου, τίποτα δεν θα ήταν το ίδιο χωρίς την τεχνολογία και την χρήση των ηλεκτρονικών υπολογιστών.

Αν λέγαμε σε κάποιον πριν από 20 χρόνια πως θα μπορεί να έχει στην τσέπη μια συσκευή που θα συνδυάζει έναν υπερσύγχρονο ηλεκτρονικό υπολογιστή, μια φωτογραφική μηχανή, όλες (σχεδόν) τις εγκυκλοπαίδειες του κόσμου, ένα σύστημα πλοήγησης, ένα ξυπνητήρι, μια παιχνιδομηχανή και ένα τηλέφωνο, το πιο πιθανό είναι πως θα μας περνούσε για τρελό και δεν θα μας ξαναμιλούσε.

Σήμερα όμως όλα τα παραπάνω έχουν γίνει πραγματικότητα και είναι ένα κομμάτι της καθημερινότητας μας. Σχεδόν όλες οι ηλεκτρονικές συσκευές είναι συνδεδεμένες στο διαδίκτυο με αποτέλεσμα όχι μόνο να μπορούν να ελεγχθούν από οποιαδήποτε γωνιά του κόσμου αλλά έτσι αποκτούν και μια δόση ευφυΐας.

Το ψυγείο μπορεί να μας προτείνει συνταγές βάση των τροφίμων που έχουμε αγοράσει. Μπορεί να μάθει τις προτιμήσεις μας ανάλογα με τα ψώνια που κάνουμε και να μας

ενημερώνει όταν τελειώνει κάποιο προϊόν. Μπορεί να μας προειδοποιεί όταν κοντεύουν να λήξουν κάποια προϊόντα αλλά και να παραγγείλει νέα χωρίς να χρειαστεί να πάμε για ψώνια. Κάποια έξυπνα σπίτια αναγνωρίζουν τους κατοίκους τους και προσαρμόζονται σε αυτούς. Για παράδειγμα καθώς μπαίνει το παιδί στο δωμάτιό του ανοίγει το ραδιόφωνο παίζοντας την αγαπημένη του μουσική. Η τηλεόραση ανοίγει καθώς μπαίνει ο μπαμπάς στο σαλόνι και δείχνει τον ποδοσφαιρικό αγώνα της αγαπημένης του ομάδας. Με την φωτογραφική μηχανή της κινητής μας συσκευής μπορούμε να τραβήξουμε φωτογραφία κάποιο λουλούδι, κάποιο ζώο ή οποιοδήποτε αντικείμενο και να το αναγνωρίσει.

Όλα τα παραπάνω και ακόμα πολλά παραδείγματα έχουν έναν κοινό παρονομαστή, την αναγνώριση προτύπων, η οποία επιτυγχάνεται με μεθόδους μηχανικής μάθησης σε συνδυασμό με μεθόδους επεξεργασίας εικόνας.

Το κυριότερο συστατικό για την αναγνώριση προτύπων είναι τα δεδομένα με τα οποία θα γίνει η εκπαίδευση των μοντέλων. Αν τα δεδομένα δεν είναι αντιπροσωπευτικά του προβλήματος τότε και τα αποτελέσματα δεν θα είναι ικανοποιητικά.

Πέρα όμως από τα δεδομένα, παίζουν σημαντικό ρόλο και οι μέθοδοι μηχανικής μάθησης. Οι σύγχρονες μέθοδοι μηχανικής απαιτούν μεγάλο πλήθος δεδομένων ώστε να αποδώσουν ικανοποιητικά αποτελέσματα αλλά και μεγάλη επεξεργαστική ισχύ.

1.2 *Αντικείμενο διπλωματικής*

Η παρούσα διπλωματική έρχεται να δώσει λύση στο πρόβλημα της αναζήτησης των δεδομένων και την επιλογή της κατάλληλης μεθόδου για τον εντοπισμό και την αναγνώριση πινακίδων κυκλοφορίας αυτοκινήτων.

Η εύρεση κατάλληλων δεδομένων για την εκπαίδευση ενός μοντέλου είναι μεγάλη πρόκληση. Η αξία των ίδιων των δεδομένων πολλές φορές είναι ανεκτίμητη σε σχέση τη μεθοδολογία και με τον αλγόριθμο που γίνεται η εκπαίδευση. Για τον λόγο αυτό τείνουν τα σύνολα δεδομένων να μένουν ιδιωτικά και να μη αναρτώνται δημόσια.

Όσο αφορά την επιλογή των κατάλληλης μεθόδου για τον εντοπισμό πινακίδων κυκλοφορίας σε φωτογραφίες, τις περισσότερες φορές η πρώτη μέθοδος που επιλέγεται είναι η χρήση μεθόδων μηχανικής μάθησης. Αυτό απαιτεί την εκπαίδευση ενός ή περισσότερων μοντέλων η οποία είναι μια χρονοβόρα διαδικασία. Πέρα από τον παράγοντα του χρόνου, όπως αναφέρθηκε προηγουμένως, η εκπαίδευση απαιτεί τα κατάλληλα δεδομένα τόσο ποιοτικά όσο και ποσοτικά.

1.2.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Δημιουργήθηκε νέα συλλογή δεδομένων από φωτογραφίες πινακίδων κυκλοφορίας αυτοκινήτων.
2. Υλοποιήθηκε αλγόριθμος για τον εντοπισμό της πινακίδας κυκλοφορίας και την εξαγωγή των χαρακτήρων της από εικόνες, βίντεο αλλά και ζωντανά από κάμερα, που βασίζεται σε βασικές μεθόδους επεξεργασίας εικόνας και όχι σε μεθόδους μηχανικής μάθησης.
3. Συγκρίθηκαν 9 μέθοδοι μηχανικής μάθησης για την αναγνώριση τόσο των αριθμών αλλά και των γραμμάτων της πινακίδας κυκλοφορίας.

1.3 *Οργάνωση κειμένου*

Στο Κεφάλαιο 2 γίνεται μια παρουσίαση βασικών μεθόδων επεξεργασίας εικόνας που θα χρησιμοποιηθούν για την δημιουργία του αλγορίθμου εντοπισμού της πινακίδας κυκλοφορίας από φωτογραφίες καθώς και των μεθόδων μηχανικής μάθησης και βαθειάς μάθησης. Στο κεφάλαιο 3 γίνεται μια εισαγωγή στα Συστήματα Αυτόματης Αναγνώρισης Πινακίδων Κυκλοφορίας και στα χαρακτηριστικά τους. Στο Κεφάλαιο 4 συγκρίνονται κάποια δημόσια διαθέσιμα σύνολα δεδομένων από πινακίδες κυκλοφορίας και αναφέρονται τα προβλήματά τους. Στο Κεφάλαιο 5 αναλύεται ο αλγόριθμος εντοπισμού της πινακίδας κυκλοφορίας και της εξαγωγής των χαρακτήρων από αυτήν. Στο Κεφάλαιο 6 παρουσιάζεται ο τρόπος δημιουργίας της δικής μας συλλογής δεδομένων και εκπαιδεύονται τα μοντέλα για την αναγνώριση των χαρακτήρων (αριθμών και γραμμάτων) της πινακίδας. Στο Κεφάλαιο 7 παρουσιάζονται τα αποτελέσματα της εκπαίδευσης των μοντέλων μέσω διαγραμμάτων και πινάκων. Στο Κεφάλαιο 8 αναφέρονται τεχνικές λεπτομέρειες και τα εργαλεία και οι βιβλιοθήκες που χρησιμοποιήθηκαν στην εργασία. Τέλος, στο Κεφάλαιο 9 συνοψίζονται τα αποτελέσματα και παρουσιάζονται τα συμπεράσματα καθώς και κάποιες σχετικές εργασίες και μελλοντικές επεκτάσεις της έρευνας.

2

Θεωρητικό υπόβαθρο

2.1 *Βασικές Μέθοδοι Επεξεργασίας Εικόνας*

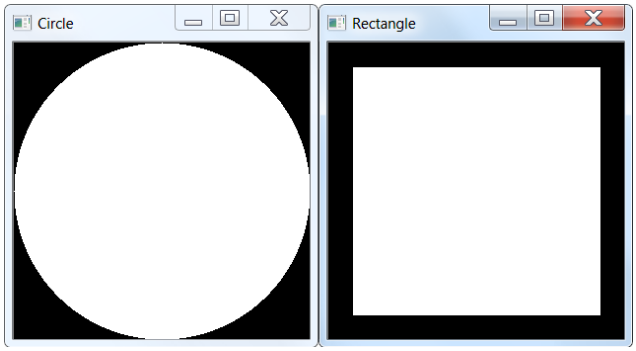
Για να γίνει κατανοητή η διαδικασία που θα ακολουθήσουμε στα παρακάτω κεφάλαια πρέπει πρώτα να γίνει μια εισαγωγή στις βασικές μεθόδους επεξεργασίας εικόνας που θα χρησιμοποιήσουμε. Οι μέθοδοι που θα καλύψουμε περιλαμβάνουν από απλές πράξεις όπως δυαδικές πράξεις μέχρι εντοπισμό περιγραμμάτων.

2.1.1 Δυαδικές πράξεις

Οι δυαδικές πράξεις είναι πολύ γνωστές στην επιστήμη των υπολογιστών. Παρόμοια και στην επεξεργασία εικόνας, οι δυαδικές πράξεις γίνονται σε δυαδικό επίπεδο και αναπαρίστανται ως εικόνες σε κλίμακα του γκρι. Συγκεκριμένα ένα πίξελ μπορεί να έχει την τιμή μηδέν οπότε είναι μαύρο, ή να έχει τιμή μεγαλύτερη του μηδέν οπότε έχει απόχρωση του γκρι. Για να γίνει καλύτερα κατανοητό ας δούμε τα παρακάτω παραδείγματα όπου παρουσιάζονται οι δυαδικές πράξεις:

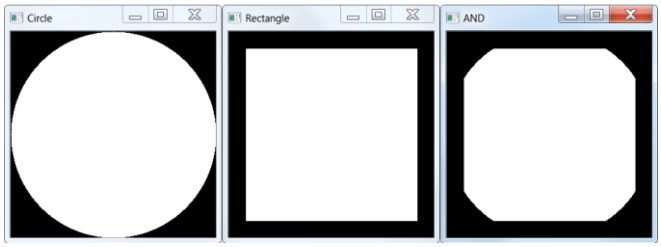
- ΚΑΙ (AND)
- Ή (OR)
- ΑΠΟΚΛΕΙΣΤΙΚΟ Ή (XOR ή Exclusive OR)
- ΌΧΙ (NOT)

Δημιουργούμε δύο εικόνες μεγέθους 300x300 πίξελ και ζωγραφίζουμε στην κάθε μία ένα τετράγωνο μεγέθους 250x250 και έναν κύκλο ακτίνας 150 πίξελ αντίστοιχα. Τα σχήματα τα τοποθετούμε στο κέντρο των εικόνων (εικόνα 1).



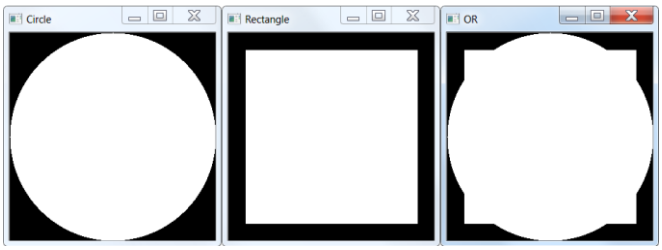
Εικόνα 1: Ασπρόμαυρες εικόνες με κύκλο και τετράγωνο

Όταν εφαρμόσουμε τον τελεστή "ΚΑΙ" στις δύο εικόνες αυτό θα έχει ως αποτέλεσμα τη δημιουργία μιας νέας εικόνας (εικόνα 2), όπου για κάθε πίξελ στις συντεταγμένες (x,y), αν η τιμή του είναι μεγαλύτερη του μηδενός και στις δύο εικόνες, τότε και το πίξελ της νέας εικόνας έχει τιμή μεγαλύτερη του μηδενός, θα είναι δηλαδή λευκό. Κρατάμε δηλαδή στην νέα εικόνα κοινές περιοχές με τιμή μεγαλύτερη του μηδενός.



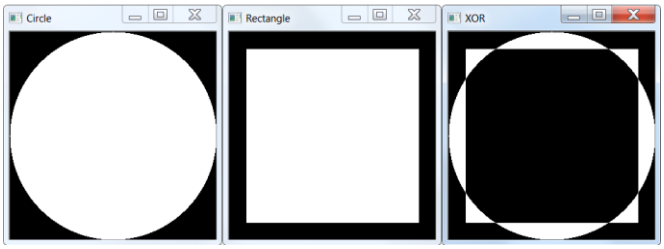
Εικόνα 2: Δυαδικός τελεστής ΚΑΙ (AND)

Αντίστοιχα με τον τελεστής Ή (OR) κρατάμε περιοχές που έχουν τιμές μεγαλύτερες του μηδενός ή στην μία εικόνα ή στην άλλη ή και στις δύο (εικόνα 3).



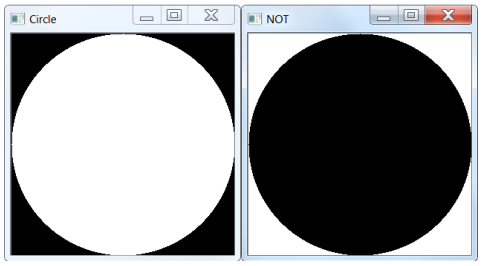
Εικόνα 3: Δυαδικός τελεστής Ή (OR)

Ο τελεστής Αποκλειστικό Ή είναι ιδιαίτερος καθώς κρατάει μόνο τις περιοχές που οι τιμές είναι μεγαλύτερες του μηδενός ή στην μία εικόνα ή στην άλλη, αλλά όχι και στις δύο (εικόνα 4).



Εικόνα 4: Δυαδικός τελεστής Αποκλειστικό Ή (XOR)

Τέλος ο τελεστής ΟΧΙ (NOT) εφαρμόζεται σε μία μόνο εικόνα και αντιστρέφει τα bit (εικόνα 5)



Εικόνα 5: Δυαδικός τελεστής ΟΧΙ (NOT)

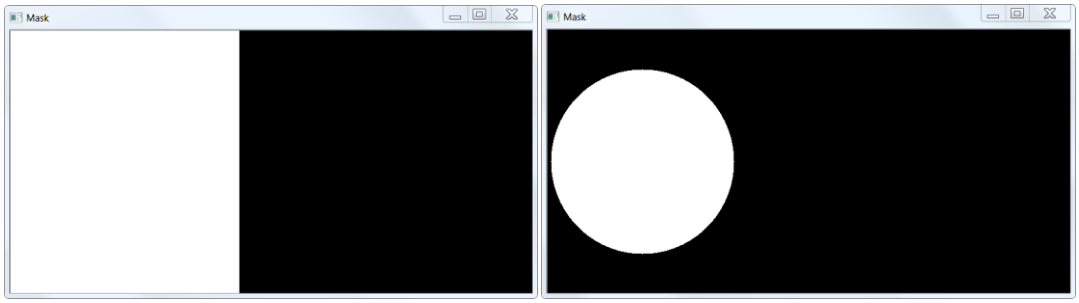
2.1.2 Μάσκες

Οι μάσκες σε συνδυασμό με τους δυαδικούς τελεστές μάς επιτρέπουν να απομονώσουμε σε εικόνες, περιοχές που μας ενδιαφέρουν (Regions of Interest ή ROI). Οι περιοχές αυτές μπορούν να έχουν οποιοδήποτε σχήμα επιθυμούμε.

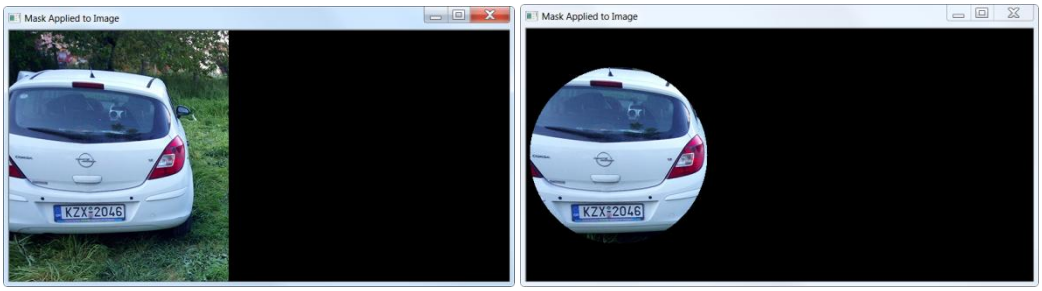
Ας υποθέσουμε πως σε μια εικόνα (εικόνα 6) με δύο οχήματα θέλουμε να κρατήσουμε μόνο το ένα από δύο. Θα μπορούσαμε απλά να κόψουμε (crop) την εικόνα ή να εφαρμόσουμε σε αυτήν μια μάσκα.

Εικόνα 6: Εικόνα με δύο οχήματα στην οποία θα εφαρμόσουμε μάσκες

Στην εικόνα 7 δημιουργήσαμε δύο περιοχές, ένα παραλληλόγραμμο και έναν κύκλο. Οι εικόνες αυτές, συγκεκριμένα οι άσπρες περιοχές, θα χρησιμοποιηθούν ως μάσκες στην εικόνα 6. Αυτό γίνεται δίνοντας στην μέθοδο δυαδικής πράξης *`AND`* ως πρώτο και δεύτερο όρισμα την εικόνα με τα οχήματα και ως τρίτο όρισμα την τιμή *`mask=mask`* όπου *mask* είναι μια από τις παρακάτω μάσκες. Στην εικόνα 8 βλέπουμε τα αποτελέσματα των μασκών.



Εικόνα 7: Μάσκες που θα εφαρμοστούν στην εικόνα ()



Εικόνα 8: Αποτέλεσμα εφαρμογής των μασκών της εικόνας () στην εικόνα ()

2.1.3 Πυρήνες

Οι πυρήνες δεν είναι τίποτα άλλα παρά πίνακες οι οποίοι ολισθαίνουν σε μια εικόνα αριστερά, δεξιά, πάνω και κάτω και εφαρμόζουν κάποια μαθηματική πράξη σε κάθε (x,y) συντεταγμένη της εικόνας. Η διαδικασία αυτή ονομάζεται Convolution και λειτουργεί ως φίλτρο πάνω στην εικόνα. Ως αποτέλεσμα μπορεί να έχουμε μια θολωμένη (blur) εικόνα , μια

πιο έντονη (sharpen) ή να εντοπιστούν ακμές (edges) σε αυτήν. Βέβαια αυτά είναι μερικά από τα αποτελέσματα[1]. Οι πυρήνες μπορούν να χρησιμοποιηθούν επίσης και για την εξαγωγή χαρακτηριστικών από εικόνες όπως στα Βαθιά Νευρωνικά Δίκτυα CNN (Convolutional Neural Networks).

Όπως αναφέραμε ο πυρήνας είναι ένα πίνακας που τοποθετείται πάνω σε κάθε πίξελ και εφαρμόζεται σε αυτό κάποια μαθηματική πράξη. Συγκεκριμένα ο πυρήνας τοποθετείται έτσι ώστε το κεντρικό σημείο του να βρίσκεται πάνω από κάποιο πίξελ. Καταλαβαίνουμε λοιπόν πως οι διαστάσεις, MxN, του πυρήνα πρέπει να είναι μονοί αριθμοί (Σχήμα 1).

$$\mathrm{K} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \mathrm{K} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Σχήμα 1: Αριστερά: Πυρήνας (kernel) 3x3, Δεξιά: Πυρήνας 2x2

Η μαθηματική πράξη που εφαρμόζεται είναι πολλαπλασιασμός πινάκων, πολλαπλασιάζουμε δηλαδή τον πυρήνα με μια περιοχή της εικόνας ίσων διαστάσεων με αυτών του πυρήνα (Σχήμα 2). Συγκεκριμένα ο πολλαπλασιασμός γίνεται element-wise δηλαδή πολλαπλασιάζουμε το κάθε στοιχείο του ενός πίνακα με το στοιχείου του άλλου πίνακα που βρίσκεται στην ίδια θέση.

$$\mathrm{K}_{i,j} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 93 & 139 & 101 \\ 26 & 252 & 196 \\ 135 & 230 & 18 \end{bmatrix} = \begin{bmatrix} -1 \times 93 & 0 \times 139 & 1 \times 101 \\ -2 \times 26 & 0 \times 252 & 2 \times 196 \\ -1 \times 135 & 0 \times 230 & 1 \times 18 \end{bmatrix}$$

Σχήμα 2: Μαθηματική πράξη εφαρμογής πυρήνα σε κάποια πίξελ

Το τελικό αποτέλεσμα που παίρνουμε είναι ένας πίνακας MxN, ίδιων διαστάσεων με τον πυρήνα, όπου προσθέτουμε όλα τα στοιχεία του και το αποτέλεσμα (Σχήμα 3) το τοποθετούμε στο σημείο (i,j) της εικόνας.

$$K_{i,j} = \begin{bmatrix} -93 & 0 & 101 \\ -52 & 0 & 392 \\ -135 & 0 & 18 \end{bmatrix} = 231$$

Σχήμα 3: Αποτέλεσμα εφαρμογής πυρήνα σε κάποια πίξελ

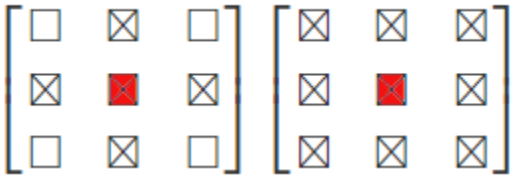
[1] http://setosa.io/ev/image-kernels/

Στον ιστοχώρο http://setosa.io/ev/image-kernels/ μπορούμε να δούμε παραδείγματα εφαρμογής πυρήνων σε εικόνες.

2.1.4 Μορφολογικές Πράξεις

Οι μορφολογικές πράξεις είναι απλοί μετασχηματισμοί που εφαρμόζονται σε ασπρόμαυρες ή σε εικόνες σε κλίμακα του γκρι. Συγκεκριμένα εφαρμόζονται σε σχήματα και δομές εντός της εικόνας. Οι πράξεις αυτές έχουν ως αποτέλεσμα την αύξηση ή την μείωση του μεγέθους αντικειμένων στις εικόνες. Επίσης μπορούμε να εφαρμόσουμε μορφολογικές πράξεις για να κλείσουμε ή να ανοίξουμε κενά μεταξύ σχημάτων.

Οι μορφολογικές πράξεις εξετάζουν μια εικόνα χρησιμοποιώντας ένα δομικό στοιχείο (structuring block), το οποίο ορίζει τα γειτονικά πίξελ που θα εξεταστούν (Σχήμα 4). Το αποτέλεσμα εξαρτάται από την ίδια την πράξη που εφαρμόζεται καθώς και από το μέγεθος του δομικού στοιχείου.



Σχήμα 4: Δομικό στοιχείο μορφολογικών πράξεων, 4 γειτονικών σημείων (αριστερά) και 8 γειτονικών σημείων (δεξιά).

Οι μορφολογικές πράξεις περιλαμβάνουν τους εξής μετασχηματισμούς:

- Διάβρωση (Erosion)
- Διαστολή (Dilation)
- Άνοιγμα (Opening)
- Κλείσιμο (Closing)
- Μορφολογική κλίση (Morphological Gradient)
- Μαύρο καπέλο (Black hat)
- Καπέλο ή "Λευκό καπέλο" (Top hat ή “White hat”)

2.1.4.1 Διάβρωση (Erosion)

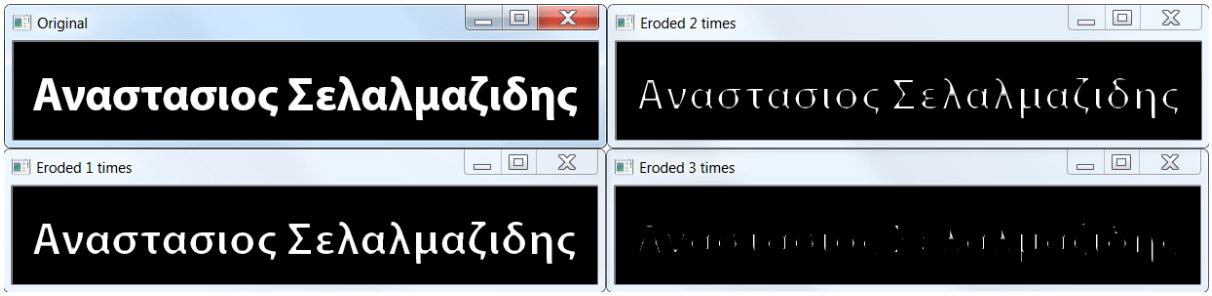
Όπως αποκαλύπτει και ο τίτλος, η διάβρωση μικραίνει, αλλοιώνει, αντικείμενα μέσα σε εικόνες. Συγκεκριμένα αφαιρεί πίξελ που βρίσκονται στα όρια ενός αντικειμένου. Για να το

πετύχουμε αυτό ορίζουμε ένα δομικό στοιχείο το οποίο σαρώνει όλη την εικόνα και εφαρμόζει τον κανόνα:

- Αν όλα τα πίξελ που βρίσκονται εντός του δομικού στοιχείου έχουν τιμή
    - > 0, τότε κρατάμε το πίξελ που βρίσκεται στο κέντρο
    - αλλιώς θέτουμε το πίξελ = 0

Στην εικόνα 9 βλέπουμε το αποτέλεσμα εφαρμογής διάβρωσης σε μια εικόνα μετά από 1, 2 και 3 επαναλήψεις αντίστοιχα με δομικό στοιχείο 3x3, 8 γειτονικών σημείων.

Παρατηρούμε πως μετά από 3 επαναλήψεις το αντικείμενο στην εικόνα έχει σχεδόν εξαφανιστεί. Η διάβρωση μπορεί να χρησιμοποιηθεί επίσης για να ξεχωρίσει δύο ή περισσότερα αντικείμενα.

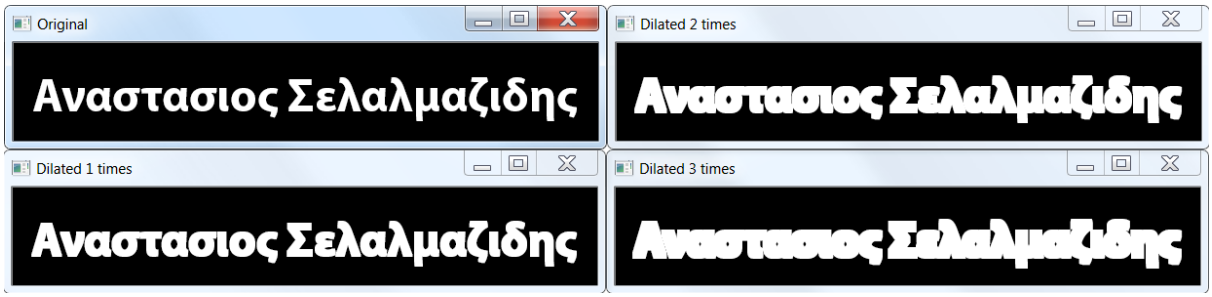


Εικόνα 9: Μορφολογική πράξη διάβρωσης μετά από 1,2 και 3 επαναλήψεις με δομικό στοιχείο 3x3, 8 γειτονικών σημείων

2.1.4.2 Διαστολή (Dilation)

Η διαστολή είναι το αντίθετο της διάβρωσης. Όπως η διάβρωση αφαιρεί πίξελ, έτσι η διαστολή προσθέτει πίξελ με τον ίδιο ακριβώς τρόπο. Το κεντρικό πίξελ του δομικού στοιχείου γίνεται λευκό αν όλα τα πίξελ του δομικού στοιχείου έχουν τιμή > 0.

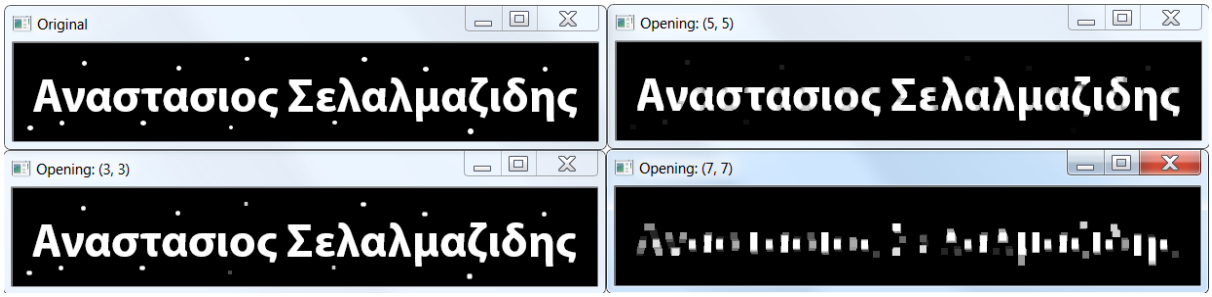
Στην εικόνα 10 βλέπουμε το αποτέλεσμα εφαρμογής διαστολής σε μια εικόνα μετά από 1, 2 και 3 επαναλήψεις αντίστοιχα με δομικό στοιχείο 3x3, 8 γειτονικών σημείων. Σε αντίθεση με την διάβρωση όπου μετά από 3 επαναλήψεις είχαν σχεδόν εξαφανιστεί τα γράμματα από την εικόνα, παρατηρούμε πως τα γράμματα έχουν κολλήσει μεταξύ τους. Καταλαβαίνουμε λοιπόν πως ο μετασχηματισμός της διάβρωσης βοηθάει ώστε να ενώνονται τμήματα αντικειμένων.



Εικόνα 10: Μορφολογική πράξη διαστολής μετά από 1,2 και 3 επαναλήψεις με δομικό στοιχείο 3x3, 8 γειτονικών σημείων

2.1.4.3 Άνοιγμα (Opening)

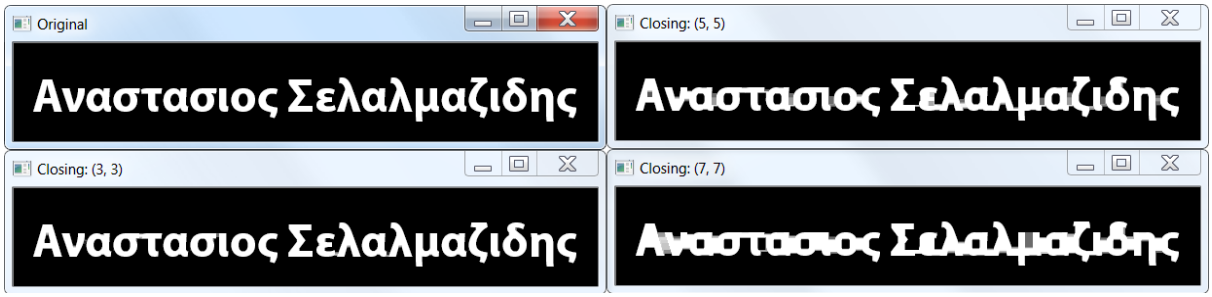
Η μορφολογική πράξη του ανοίγματος είναι μια διάβρωση ακολουθούμενη από μια διαστολή. Χρησιμοποιείται για την αφαίρεση μικρών μαζών που λειτουργούν ως θόρυβος. Πρώτα εκτελείται η διάβρωση για να αφαιρέσει τον θόρυβο και μετά με την διαστολή επαναφέρονται τα υπόλοιπα αντικείμενα στο αρχικό τους μέγεθος. Παρατηρούμε πως με δομικό στοιχείο μεγέθους 5x5 ο θόρυβος έχει σχεδόν εξαφανιστεί από την εικόνα ενώ με δομικό στοιχείο μεγέθους 7x7 αλλοιώνεται εντελώς το αντικείμενο της εικόνας (εικόνα 11).



Εικόνα 11: Μορφολογική πράξη ανοίγματος με δομικά στοιχεία μεγέθους 3x3, 5x5 και 7x7

2.1.4.4 Κλείσιμο (Closing)

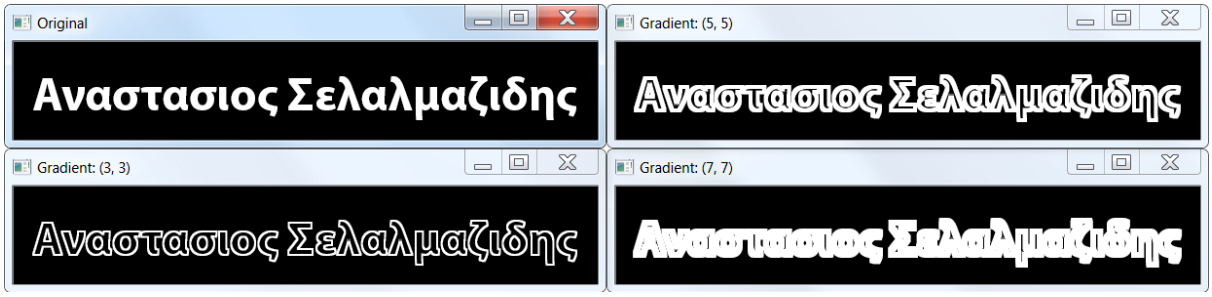
Το κλείσιμο είναι η αντίθετη πράξη του ανοίγματος, δηλαδή είναι μια διαστολή ακολουθούμενη από μια διάβρωση και χρησιμοποιείται για να κλείσει τρύπες μέσα σε αντικείμενα (εικόνα 12). Παρατηρούμε πως όσο μεγαλώνει το μέγεθος του δομικού στοιχείου γίνονται πιο έντονες οι αλλαγές στο αντικείμενο της εικόνας, κλείνοντας τα κενά που υπάρχουν στα γράμματα.



Εικόνα 12: Μορφολογική πράξη κλεισίματος με δομικά στοιχεία μεγέθους 3x3, 5x5 και 7x7

2.1.4.5 Μορφολογική κλίση (Morphological Gradient)

Η πράξη της Μορφολογικής Κλίσης είναι η διαφορά μεταξύ της διαστολής και της διάβρωσης και είναι χρήσιμη για τον εντοπισμό του περιγράμματος αντικειμένων σε εικόνες.



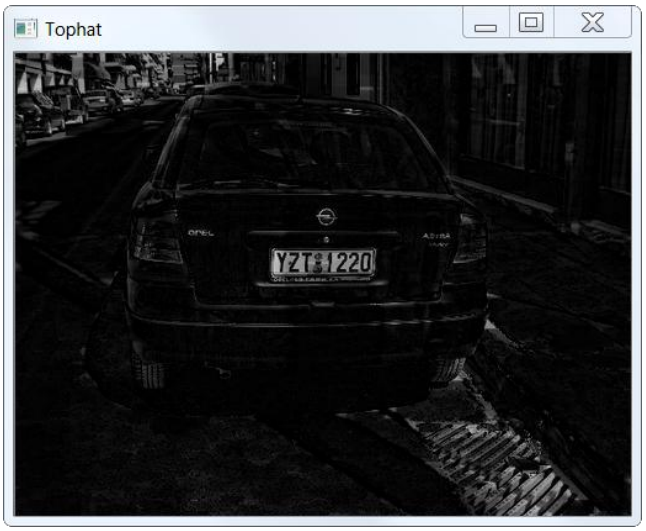
Εικόνα 13: Μορφολογική κλίση με δομικά στοιχεία μεγέθους 3x3, 5x5 και 7x7

2.1.4.6 Καπέλο ή "Λευκό καπέλο" (Top hat ή "White hat")

Η μορφολογική πράξη "White Hat" είναι η διαφορά μεταξύ της πρωτότυπης εικόνας και του ανοίγματος. Χρησιμοποιείται για τον εντοπισμό φωτεινών περιοχών σε σκούρο φόντο. Οι πράξη αυτή καθώς η πράξη "Black Hat" που θα δούμε παρακάτω είναι πιο κατάλληλες για εικόνες σε κλίμακα του γκρι, σε αντίθεση με τις υπόλοιπες πράξεις που εξετάσαμε έως τώρα που εφαρμόζονταν σε δυαδικές (ασπρόμαυρες) εικόνες.

Θα εφαρμόσουμε την πράξη "White Hat" στην εικόνα 14 για να εντοπίσουμε την περιοχή της πινακίδας, η οποία είναι λευκή πάνω σε σκούρο φόντο (το αυτοκίνητο). Για να το πετύχουμε αυτό πρέπει να ορίσουμε ένα δομικό στοιχείο συγκεκριμένου μεγέθους, σχετικό με το μέγεθος του αντικειμένου που θέλουμε να εντοπίσουμε. Παρατηρούμε πως η πινακίδα έχει αναλογία περίπου 1 προς 3. Έτσι το δομικό στοιχείο που θα ορίσουμε θα είναι 13x5. Το αποτέλεσμα φαίνεται στην εικόνα 15. Παρατηρούμε πως έχει εντοπιστεί η περιοχή της πινακίδας ενώ οι χαρακτήρες παραμένουν μαύροι επειδή είναι σκούροι σε φωτεινό φόντο.

Εικόνα 14: Εικόνα πριν την εφαρμογή της μορφολογικής πράξης "White Hat"



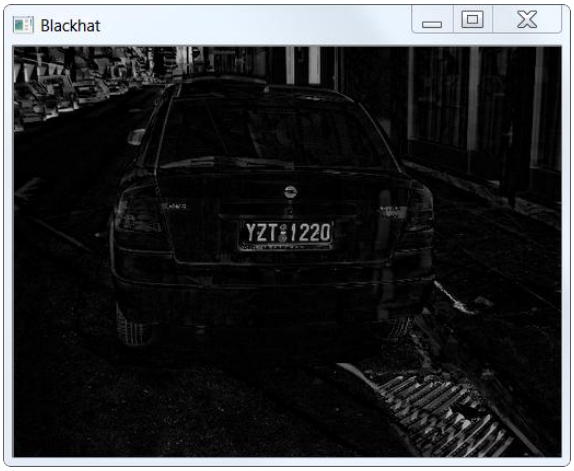
Εικόνα 15: Αποτέλεσμα εφαρμογής της μορφολογικής πράξης "White Hat"

2.1.4.7 Μαύρο καπέλο (Black hat)

Προηγουμένως καταφέραμε να εντοπίσουμε την περιοχή της πινακίδας στην εικόνα με τη χρήση του "White Hat". Αυτό έγινε όμως επειδή το όχημα ήταν σκούρο, έτσι η πινακίδα ήταν φωτεινή σε σκούρο φόντο. Αν όμως το όχημα είχε πιο ανοικτό χρώμα η πράξη αυτή δεν θα μας βοηθούσε. Μια καλύτερη προσέγγιση θα ήταν να προσπαθήσουμε να εντοπίσουμε τους χαρακτήρες της πινακίδας, μιας και στην πλειοψηφία οι χαρακτήρες είναι σκούροι σε φωτεινό φόντο. Αυτό μπορούμε να το καταφέρουμε με την μορφολογική πράξη "Black Hat" που είναι η διαφορά μεταξύ της μορφολογικής πράξης του κλεισίματος και της αρχικής εικόνας. Στην ουσία η πράξη αυτή είναι η αντίθετη της "White Hat".

Στην εικόνα 16 βλέπουμε πως έχουν εντοπιστεί σωστά οι χαρακτήρες της πινακίδας. Την πράξη αυτή σε συνδυασμό με την πράξη του κλεισίματος θα τις χρησιμοποιήσουμε σε

παρακάτω κεφάλαιο για τον εντοπισμό των χαρακτήρων της πινακίδας και την αφαίρεση θορύβου.



Εικόνα 16: Αποτέλεσμα εφαρμογής της μορφολογικής πράξης "Black Hat"

2.1.5 Εξομάλυνση και θόλωμα (Smoothing and Blurring)

Η διαδικασία της Εξομάλυνσης και του Θολώματος μια εικόνας είναι από τα βασικότερα βήματα που γίνονται κατά το στάδιο της προ-επεξεργασίας (pre-processing). Οι πράξεις αυτές εφαρμόζονται συνήθως πριν την εύρεση ακμών (edge detection) ή την εφαρμογή κατωφλιού. Με τον τρόπο αυτό καταφέρνουμε να μειώσουμε τον θόρυβο στην εικόνα, ώστε να μπορούμε να εντοπίσουμε ευκολότερα τα αντικείμενα που μας ενδιαφέρουν.

Παρακάτω θα μελετήσουμε τις εξής μεθόδους Εξομάλυνσης και θολώματος:

1. Μέσος όρος (Averaging)
2. Γκαουσιανό θόλωμα (Gaussian blurring)
3. Διάμεσο φιλτράρισμα (Median filtering)
4. Διμερές φιλτράρισμα (Bilateral filtering)

2.1.5.1 Θόλωμα μέσου όρου (Averaging)

Όπως προδίδει το όνομα αυτής της μεθόδου, αυτό που κάνει είναι να υπολογίζει τον μέσο όρο των τιμών των πίξελ μια περιοχής και να θέτει την τιμή αυτή στο κεντρικό πίξελ της περιοχής. Με τον τρόπο αυτό καταφέρνουμε να μειώσουμε τον θόρυβο αλλά και τις λεπτομέρειες σε μια εικόνα.

Για να το πετύχουμε αυτό χρησιμοποιούμε πυρήνες μεγέθους MxN όπου τα M και N είναι μονοί αριθμοί. Παράδειγμα τέτοιων πυρήνων φαίνεται στο Σχήμα 5.

$$K_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, K_2 = \frac{1}{25}\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Σχήμα 5: Πυρήνες μεγέθους 3x3, 5x5

Καταλαβαίνουμε πως όσο μεγαλύτερος είναι ο πυρήνας τόσο πιο έντονο θα είναι και το θόλωμα της περιοχής. Στην εικόνα 17 βλέπουμε το αποτέλεσμα θολώματος μιας εικόνας για πυρήνες μεγέθους 3x3, 9x9 και 15x15.

Εικόνα 17: Θόλωμα μέσου όρου με πυρήνες 3x3, 9x9 και 15x15

2.1.5.2 Γκαουσιανό θόλωμα (Gaussian Blurring)

Στο θόλωμα με μέσο όρο ορίσαμε τα βάρη για κάθε γειτονικό πίξελ να έχουν την ίδια τιμή. Θα μπορούσαμε να ορίσουμε διαφορετικά βάρη ανάλογα με την απόσταση του κάθε πίξελ από το κεντρικό πίξελ. Αυτό ακριβώς κάνει το Γκαουσιανό θόλωμα.

Το Γκαουσιανό θόλωμα αφαιρεί θόρυβο που ακολουθεί την Γκαουσιανή κατανομή. Σε σχέση με το θόλωμα με μέσο όρο, δεν θολώνει τόσο την εικόνα αλλά χαρίζει μια φυσική θολούρα διατηρώντας τις γωνίες στα αντικείμενα της εικόνας.

Ο τύπος που χρησιμοποιείται για δύο διαστάσεις (x,y) είναι:

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2.1)$$

όπου τα x και y είναι η οριζόντια και κάθετη απόσταση αντίστοιχα, από το κεντρικό σημείο του πυρήνα.

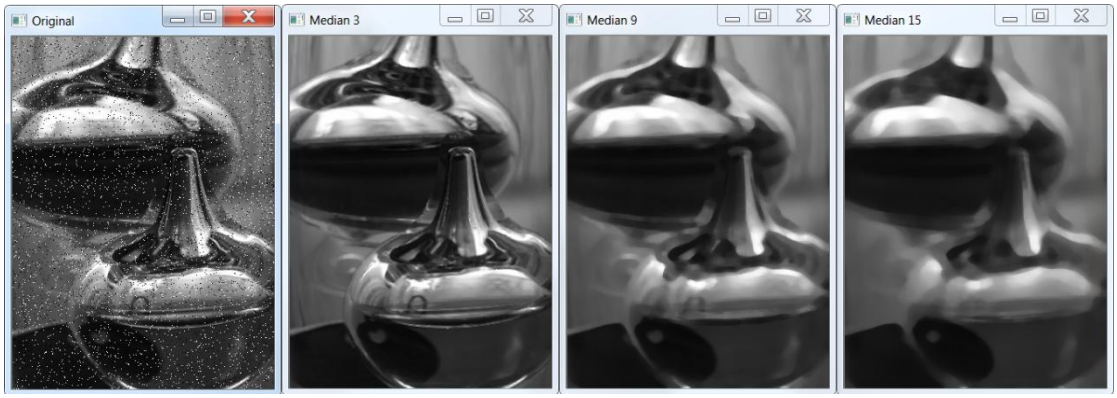
Εικόνα 18: Γκαουσιανό θόλωμα με πυρήνες 3x3, 9x9 και 15x15

2.1.5.3 Διάμεσο φιλτράρισμα (Median filtering)

Το διάμεσο φιλτράρισμα βοηθάει στην αφαίρεση θορύβου που χαρακτηρίζεται ως "αλατοπίπερο" (salt-n-pepper). Πήρε την ονομασία του γιατί ο θόρυβος που παρουσιάζεται μοιάζει με άσπρες και μαύρες κηλίδες.

Σε αντίθεση με τις προηγούμενες μεθόδους όπου ο πυρήνας μπορούσε να είναι παραλληλόγραμμος, εδώ ο πυρήνας πρέπει έχει διαστάσεις KxK να είναι δηλαδή τετράγωνος. Επιπλέον το κεντρικό πίξελ του πυρήνα δεν αντικαθίσταται με τον μέσο όρο των γειτονικών πίξελ αλλά με την διάμεση τιμή.

Ο λόγος που αυτή η μέθοδος είναι πιο αποτελεσματική στην αφαίρεση του θορύβου "salt-n-pepper" είναι γιατί κάθε κεντρικό πίξελ αντικαθίσταται με μια τιμή η οποία υπάρχει ήδη στην εικόνα, διατηρώντας επίσης καλύτερα τις ακμές στα αντικείμενα σε σχέση με τις προηγούμενες μεθόδους (εικόνα 19).



Εικόνα 19: Διάμεσο φιλτράρισμα με πυρήνες 3x3, 9x9 και 15x15

2.1.5.4 Διμερές φιλτράρισμα (Bilateral filtering)

Οι προηγούμενες μέθοδοι που παρουσιάστηκαν ως τώρα είχαν ως κύριο σκοπό την αφαίρεση του θορύβου μη λαμβάνοντας υπόψη τη διατήρηση των ακμών. Το Διμερές Φιλτράρισμα προσπερνάει αυτό το εμπόδιο χρησιμοποιώντας δύο Γκαουσιανές κατανομές. Η πρώτη κατανομή λαμβάνει υπόψη μόνο τα γειτονικά πίξελ. Η δεύτερη κατανομή μοντελοποιεί την ένταση των γειτονικών πίξελ εξασφαλίζοντας πως μόνο τα πίξελ με παρόμοια ένταση θα περιλαμβάνονται στον υπολογισμό της θολούρας. Αυτό είναι λογικό γιατί αν τα γειτονικά πίξελ σε μια μικρή περιοχή έχουν περίπου την ίδια τιμή, τότε είναι πολύ πιθανό αυτά να ανήκουν στο ίδιο αντικείμενο. Το μειονέκτημα αυτής της μεθόδου είναι πως είναι πιο αργή από τις προηγούμενες.

Σε αντίθεση με τις προηγούμενες, δεν γίνεται χρήση κάποιου πυρήνα. Οι παράμετροι που πρέπει να ορίσουμε εδώ είναι:

- Η διάμετρος **d**, αντίστοιχη του πυρήνα στα προηγούμενα παραδείγματα
- Η τυπική απόκλιση του χρώματος $\boldsymbol{\sigma_{color}}$
- Η τυπική απόκλιση του χώρου $\boldsymbol{\sigma_{space}}$

Όσο αυξάνεται η τιμή του $\boldsymbol{\sigma_{color}}$ τόσο περισσότερα γειτονικά χρώματα θα υπολογίζονται στο θόλωμα. Αν η τιμή του αυξηθεί πολύ σε σχέση με την διάμετρο τότε καταρρίπτεται η υπόθεση του διμερούς φιλτραρίσματος, δηλαδή πως μόνο παρόμοια χρώματα συμβάλουν στο θόλωμα.

Τέλος, μεγάλες τιμές στο $\boldsymbol{\sigma_{space}}$ σημαίνουν πως πίξελ που βρίσκονται έξω από την διάμετρο που έχουμε ορίσει θα επηρεάζουν και αυτά το θόλωμα.

Εικόνα 20: Διμερές φιλτράρισμα με τιμές (d=11, σcolor=21, σspasce=7), (d=11, σcolor=41, σspasce=21), (d=11, σcolor=61, σspasce=39)

2.1.6 Χρωματικά Μοντέλα (Color Models) και Χώροι χρωμάτων (Color Spaces)

Βασικά (ή πρωτογενή) χρώματα είναι αυτά τα οποία όταν συνδυαστούν μεταξύ τους μπορούν να παράγουν όλα τους δυνατούς χρωματισμούς. Τα χρώματα που προκύπτουν από τον συνδυασμό των βασικών λέγονται δευτερογενή. Για την περιγραφή των χρωμάτων χρησιμοποιούμε τα χρωματικά μοντέλα. Κάθε ένα από αυτά χρησιμοποιεί κάποια βασικά χρώματα και μία μέθοδο για την περιγραφή των χρωματικών αντιλήψεων που μπορεί να έχουμε.

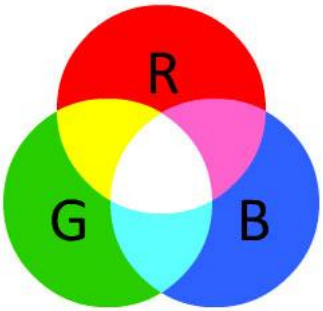
Ένα χρωματικό μοντέλο είναι ένα αφηρημένο μαθηματικό μοντέλο που περιγράφει πώς τα χρώματα μπορούν να αναπαρασταθούν ως πλειάδες αριθμών, τυπικά ως τρεις ή τέσσερις τιμές ή έγχρωμα στοιχεία. Όταν το μοντέλο αυτό συσχετίζεται με μια ακριβή περιγραφή του τρόπου ερμηνείας των συνιστωσών (συνθήκες προβολής κ.λπ.), το σύνολο των χρωμάτων που προκύπτει αποκαλείται χρωματικός χώρος.

Με απλά λόγια, ένας χώρος χρωμάτων είναι απλώς μια συγκεκριμένη οργάνωση χρωμάτων που μας επιτρέπει να εκπροσωπούμε και να αναπαράγουμε με συνέπεια τα χρώματα.

Τα διάφορα χρωματικά μοντέλα έχουν αναπτυχθεί για να γίνει δυνατή η περιγραφή των χρωμάτων με μαθηματική μορφή, κατάλληλη για την επεξεργασία τους από ψηφιακά μέσα.

2.1.6.1 RGB (Red - Green - Blue)

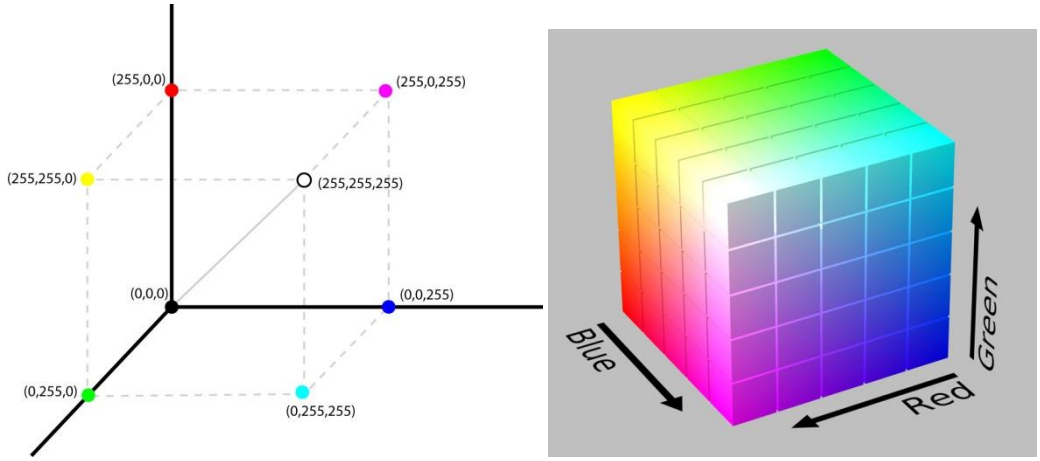
Το πρότυπο χρώματος RGB είναι ένα προσθετικό πρότυπο (Σχήμα 6) στο οποίο τα χρώματα κόκκινο, πράσινο και μπλε (χρώματα που χρησιμοποιούνται συχνά σε προσθετικά χρωματικά πρότυπα) συνδυάζονται με διάφορους τρόπους για να αναπαραχθούν άλλα χρώματα. Το όνομα του προτύπου και η σύντμηση RGB προέρχονται από τα τρία βασικά χρώματα, το κόκκινο (Red), πράσινο (Green), και το μπλε (Blue).



Σχήμα 6: Το προσθετικό πρότυπο χρώματος RGB

Με τα βασικά αυτά χρώματα έχει δημιουργηθεί το χρωματικό μοντέλο RGB με το οποίο μπορεί να γίνει η κωδικοποίηση όλων των χρωμάτων που εμφανίζονται σε μία οθόνη. Στην 8bit έκδοση του χρωματικού αυτού μοντέλου κάθε χρώμα μπορεί να παρασταθεί με μία τριάδα αριθμών από 0 έως 255. Το μοντέλο βασίζεται στο γεγονός ότι όταν μία οθόνη δεν εκπέμπει φως εμφανίζεται μαύρη. Τα υπόλοιπα χρώματα δημιουργούνται με υπέρθεση των τριών βασικών με συγκεκριμένη αναλογία.

Το μοντέλο αυτό μπορεί να παρασταθεί με έναν κύβο χρωμάτων σε ένα καρτεσιανό σύστημα συντεταγμένων (Σχήμα 7). Στην αρχή των αξόνων είναι η κορυφή του κύβου που αντιστοιχεί στο μαύρο χρώμα, ενώ στις κορυφές του κύβου που βρίσκονται πάνω στους άξονες βρίσκονται τα βασικά χρώματα (Κόκκινο, Πράσινο, Μπλε). Τα δευτερογενή χρώματα βρίσκονται στις τρεις κορυφές του κύβου που βρίσκονται απέναντι από τα αντίστοιχα βασικά χρώματα και στην κορυφή απέναντι από το μαύρο βρίσκεται το λευκό. Κάθε χρώμα στο σύστημα αυτό προσδιορίζεται από ένα σημείο στον κύβο με τρεις συντεταγμένες. Στη διαγώνιο μεταξύ μαύρου και λευκού βρίσκονται όλες οι αποχρώσεις του γκρι.



Σχήμα 7: Αναπαράσταση του χρωματικού μοντέλου RGB ως i) καρτεσιανό σύστημα (αριστερά) και ii) κύβος (δεξιά)

Εικόνα 21: Διαχωρισμός εικόνας στα επιμέρους κανάλια του χρωματικού μοντέλου RGB

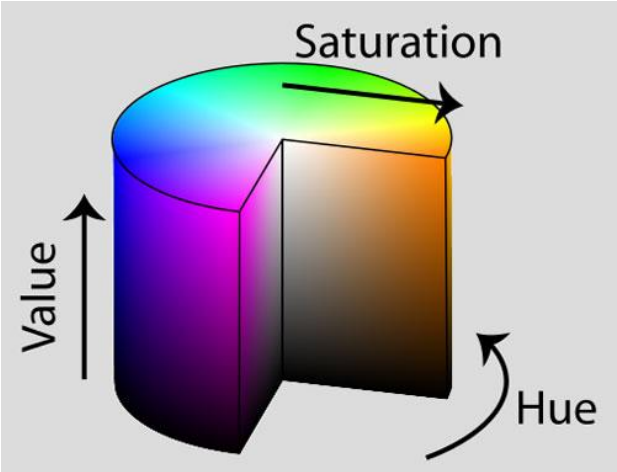
2.1.6.2 HSV (Hue - Saturation - Value)

Ο χώρος χρωμάτων HSV μετασχηματίζει τον χώρο χρωμάτων RGB, αναδιαμορφώνοντας τον ως κύλινδρο (Σχήμα 8) αντί για κύβο. Επιπλέον το λευκό ή η φωτεινότητα ενός χρώματος παύει να είναι ένας αθροιστικός συνδυασμός από κάθε κόκκινο, πράσινο και μπλε κανάλι, αλλά τώρα στον χώρο χρώματος του HSV η φωτεινότητα δίνεται στη δική της ξεχωριστή διάσταση.

Το μοντέλο αυτό αναλύει το χρώμα σε τρεις παραμέτρους:

- Χροιά (Hue), δηλώνει την θέση στον χρωματικό κύκλο (τιμές από 0-360) πχ. όλες οι σκιές και οι τόνοι του κόκκινου έχουν την ίδια χροιά

- Κορεσμός (Saturation), δηλαδή η ένταση του χρώματος με τιμές από 0-100%. Ένα χρώμα με τιμή κορεσμού 0 εμφανίζεται ως λευκό ενώ με τιμή 100 εμφανίζεται ως 100% καθαρό χρώμα, πχ. κόκκινο
- Τιμή (Value). Ελέγχει την ένταση της φωτεινότητας του χρώματος, με τιμές από 0-100%. Χαμηλές τιμές πχ. 0 είναι το μαύρο ενώ όσο αυξάνεται η τιμή αυτή παράγονται πιο φωτεινά χρώματα



Σχήμα 8: Αναπαράσταση του χρωματικού μοντέλου HSV ως κύλινδρο

Εικόνα 22: Διαχωρισμός εικόνας στα επιμέρους κανάλια του χρωματικού μοντέλου HSV

2.1.6.3 Grayscale

Το μοντέλο αυτό μπορούμε να πούμε πως δεν είναι ένα χρωματικό μοντέλο αφού είναι μια αναπαράσταση του RGB σε κλίμακα του γκρι. Η αναπαράσταση μια εικόνας στην κλίμακα του γκρι συχνά αναφέρεται ως ασπρόμαυρη, ενώ τεχνικά ο όρος αυτός είναι εσφαλμένος.

Ενώ στις ασπρόμαυρες εικόνες, αλλιώς δυαδικές, το κάθε πίξελ μπορεί να έχει τιμή 0 ή 255 σε αυτό το μοντέλο κάθε πίξελ μπορεί να έχει τιμή στο εύρος [0,255].

Κατά την μετατροπή μιας εικόνας από RGB σε Grayscale, τα κανάλια του RGB δεν δέχονται τα ίδια βάρη:

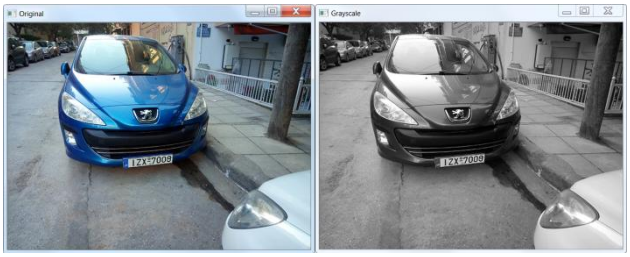
$$Y = \mathbf{0.333} \times R + \mathbf{0.333} \times G + \mathbf{0.333} \times B \quad (2.2)$$

αλλά (Hasan & Karam, 2000):

$$Y = \mathbf{0.299} \times R + \mathbf{0.587} \times G + \mathbf{0.114} \times B \quad (2.3)$$

Αυτό συμβαίνει διότι το ανθρώπινο μάτι είναι πιο ευαίσθητο στο πράσινο και στο κόκκινο από ότι στο μπλε. Παρατηρούμε πως η ποσότητα του πράσινου είναι σχεδόν η διπλή από αυτή του κόκκινου και του κόκκινου περίπου τριπλή από του μπλε.

Η κλίμακα του γκρι είναι χρήσιμη σε περιπτώσεις εντοπισμού προσώπου ή άλλων αντικειμένων όπου το χρώμα του αντικειμένου δεν παίζει κάποιον ρόλο. Επιπλέον η απαλοιφή του χρώματος μειώνει την απαίτηση σε μνήμη.



Εικόνα 23: Μετατροπή εικόνας σε κλίμακα του γκρι

2.1.7 Κατώφλι (Thresholding)

Το κατώφλι είναι μια από τις πιο κοινές και βασικές τεχνικές κατάτμησης στην επεξεργασία εικόνας που μας επιτρέπει να διαχωρίσουμε το προσκήνιο, δηλαδή το αντικείμενο που μας ενδιαφέρει, από το παρασκήνιο της εικόνας. Ο διαχωρισμός επιτυγχάνεται μέσω της μετατροπής της εικόνα από κλίμακα του γκρι σε ασπρόμαυρη.

Υπάρχουν διάφορες υλοποιήσεις αυτής μεθόδου. Στην πιο απλή υλοποίηση πρέπει να υπολογίσουμε μόνοι μας την τιμή **T** για το κατώφλι. Αυτό είναι εύκολο να γίνει σε περιβάλλον όπου είναι ελεγχόμενος ο φωτισμός και υπάρχει μεγάλη αντίθεση μεταξύ προσκήνιο και παρασκήνιο. Μια άλλη υλοποίηση είναι η Otsu όπου η ο υπολογισμός του **T** γίνεται αυτόματα. Υπάρχει επίσης και το προσαρμοσμένο κατώφλι όπου δεν χρησιμοποιείται η ίδια τιμή **T** σε όλη την εικόνα, αλλά η εικόνα χωρίζεται σε περιοχές και υπολογίζεται το κατώφλι για κάθε μια από αυτές.

2.1.7.1 Απλό κατώφλι (Simple Threshold)

Η πιο απλή μορφή μετατροπή μιας εικόνας σε ασπρόμαυρη μέσω κατωφλιού είναι να ορίσουμε μόνοι μας την τιμή **T** για το κατώφλι. Όλα τα πίξελ με τιμή μικρότερη από αυτήν παίρνουν τιμή 255 ενώ όλα με τιμή μεγαλύτερη παίρνουν τιμή 0 (2.4). Υπάρχει και η δυνατότητα να αντιστρέψουμε τη διαδικασία ώστε όλα τα πίξελ με τιμή μεγαλύτερη του **T** να πάρουν τιμή 255 και με μικρότερη να πάρουν την τιμή 0 (2.5). Συγκεκριμένα δίνεται η δυνατότητα να ορίσουμε εμείς την μέγιστη τιμή που μπορεί να πάρει κάποιο πίξελ, στην περίπτωσή μας είναι το 255.

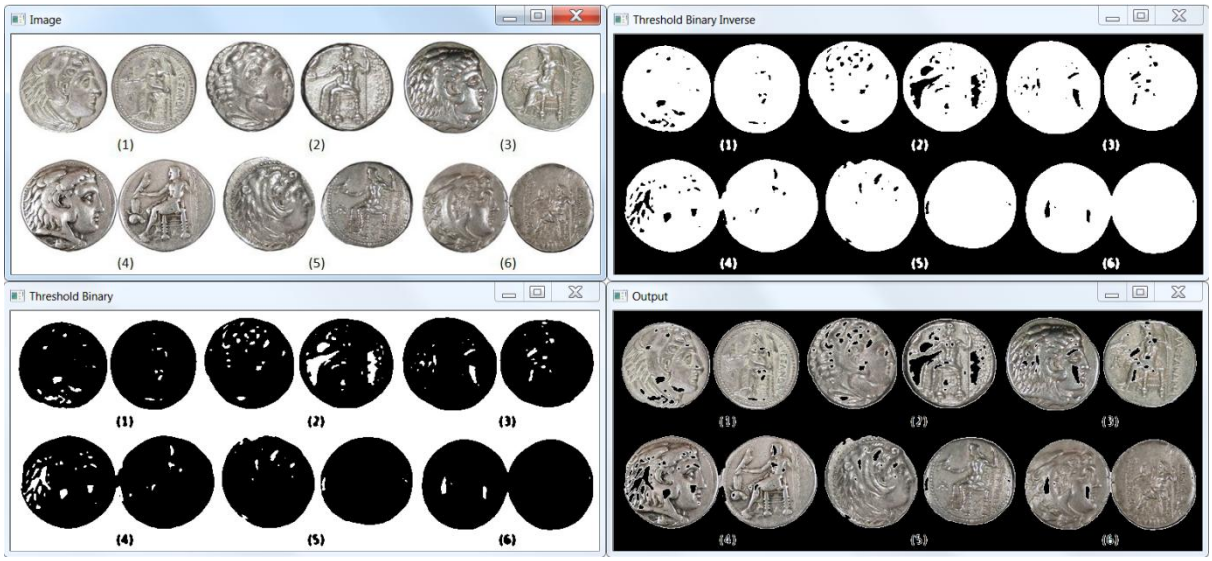
$$output(x,y) = \begin{cases} 0 & src(x,y) > \mathbf{T} \\ maxval & src(x,y) \leq \mathbf{T} \end{cases} \quad (2.4)$$

Υπολογισμός κατωφλιού

$$output(x,y) = \begin{cases} maxval & src(x,y) > \mathbf{T} \\ 0 & src(x,y) \leq \mathbf{T} \end{cases} \quad (2.5)$$

Υπολογισμός αντίστροφου κατωφλιού

Στο παράδειγμα της εικόνας 24 βλέπουμε πως εφαρμόζοντας την τεχνική του κατωφλιού καταφέραμε να απομονώσουμε τα κέρματα από το παρασκήνιο. Παρατηρούμε βέβαια πως υπάρχουν κάποια κενά μέσα στα κέρματα και αυτό οφείλεται στις συνθήκες φωτισμού. Παρόλα αυτά έχουμε πετύχει τον στόχο μας καθώς εντοπίσαμε τα περιγράμματα. Τα μικρά κενά μπορούμε να τα κλείσουμε εφαρμόζοντας μορφολογικές πράξεις, τις οποίες αναλύσαμε σε προηγούμενη παράγραφο, ή πράξεις περιγραμμάτων όπως θα δούμε σε παρακάτω παράγραφο.



Εικόνα 24: Εφαρμογή απλού (κάτω αριστερά) και αντίστροφου (πάνω δεξιά) κατωφλιού με τιμή T=200

2.1.7.2 Αυτόματο κατώφλι (Otsu)

Όπως είδαμε στην προηγούμενη παράγραφο μπορούμε εύκολα να απομονώσουμε κάποιο αντικείμενο σε μια εικόνα από το παρασκήνιο με την μέθοδο του απλού κατωφλιού, αρκεί να δώσουμε εμείς την τιμή **T**. Θα μπορούσαμε να ορίσουμε μια σταθερή τιμή στις εφαρμογές μας για το κατώφλι, όμως αυτό δεν θα την έκανε δυναμική.

Αυτό το πρόβλημα προσπαθεί να λύση η μέθοδος Otsu η οποία δίνει αρκετά καλά αποτελέσματα. Για να το καταφέρει αυτό, υποθέτει πως η εικόνα μας αποτελείται από δύο κλάσεις πίξελ. Μια για το παρασκήνιο και μια για τα αντικείμενα που μας ενδιαφέρουν. Έπειτα υπολογίζει την τιμή **T** τέτοια ώστε η διακύμανση μεταξύ των κατανομών των δύο κλάσεων να είναι η ελάχιστη. Θα πρέπει να αναφέρουμε πως η όλη διαδικασία γίνεται αφού πρώτα μετατραπεί η εικόνα σε κλίμακα του γκρι και πως η μέθοδος Otsu υπολογίζει καθολική τιμή **T** για όλη την εικόνα (εικόνα 25).

Εικόνα 25: Εφαρμογή αυτόματου κατωφλιού μέσω της μεθόδου Otsu. Η τιμή που υπολογίστηκε είναι T=199

2.1.7.3 Προσαρμοσμένο κατώφλι (Adaptive Threshold)

Το πρόβλημα με τις προηγούμενες μεθόδους ήταν πως η τιμή **T** ήταν καθολική για όλη την εικόνα. Προκειμένου να ξεπεραστεί αυτό το πρόβλημα, η μέθοδος του προσαρμοσμένου κατωφλιού υπολογίζει ξεχωριστές τιμές **T** για μικρότερες περιοχές τις εικόνας υποθέτοντας πως αυτές έχουν ομοιόμορφο φωτισμό σε αντίθεση με την εικόνα ως σύνολο.

Η επιλογή του μεγέθους της περιοχής θέλει ιδιαίτερη προσοχή. Η περιοχή πρέπει να είναι αρκετά μεγάλη ώστε να περιέχει αρκετό φόντο αλλά όχι τόσο ώστε να παραβιάζεται η υπόθεση του ομοιόμορφου φωτισμού.

Ο υπολογισμός του κατωφλιού γίνεται βάση του τύπου:

$$T = mean(I_L) - C \tag{2.6}$$

όπου:

- το μέσο (mean) μπορεί να είναι είτε το αριθμητικό είτε το Γκαουσιανό
- το $\mathbf{I_L}$ είναι μια περιοχή της εικόνας **I**
- το **C** είναι μια σταθερά

Εικόνα 26: Σύγκριση μεθόδου Otsu (πάνω δεξιά) και Προσαρμοσμένου Κατωφλιού (κάτω)

Στην εικόνα 26 γίνεται σύγκριση της μεθόδου Otsu και του προσαρμοσμένου κατωφλιού σε μια εικόνα που περιέχει μια πινακίδα κυκλοφορίας. Παρατηρούμε στη μέθοδο Otsu, πως ενώ κατάφερε και βρήκε μια ικανοποιητική τιμή **T**, δεν κατάφερε να διαχωρίσει την βίδα από τον τελευταίο χαρακτήρα. Αυτό συμβαίνει όπως αναφέραμε και νωρίτερα, διότι η αυτόματη μέθοδος υπολογίζει καθολικό κατώφλι. Σε αντίθεση η μέθοδος του προσαρμοσμένου κατωφλιού κατάφερε με επιτυχία να διαχωρίσει όλα τα συστατικά της εικόνας.

2.1.8 Διαβαθμίσεις ή Κλίσεις (Gradients)

Οι διαβαθμίσεις ή κλίσεις (Gradients) χρησιμοποιούνται για τον εντοπισμό ακμών σε εικόνες και κατά επέκταση στην εύρεση περιγραμμάτων. Κατά τον εντοπισμό ακμών λαμβάνουμε

δομικές πληροφορίες σχετικά με τα αντικείμενα μιας εικόνας. Οι ακμές θα μπορούσαν συνεπώς να αντιστοιχούν σε:

- Όρια ενός αντικειμένου σε μια εικόνα.
- Όρια σκίασης ή συνθήκες φωτισμού σε μια εικόνα.
- Όρια των τμημάτων μέσα σε ένα αντικείμενο.

Για την εύρεση των ακμών χρησιμοποιούνται πυρήνες όπως στη διαδικασία του θολώματος. Η διαφορά είναι πως το ζητούμενο στην προκειμένη περίπτωση είναι τα δομικά στοιχεία της εικόνας. Στόχος είναι να βρεθεί η αλλαγή στην κατεύθυνση του κεντρικού πίξελ στην x και y κατεύθυνση.

2.1.8.1 Πυρήνες Sobel και Scharr

Δύο βασικοί πυρήνες για τον εντοπισμό ακμών είναι των *Sobel* και *Scharr*. Στην πραγματικότητα χρησιμοποιούνται δύο πυρήνες για κάθε μέθοδο, ένας για τον εντοπισμό αλλαγών στην κατεύθυνση στον οριζόντιο άξονα X και ένας για αλλαγές στην κατεύθυνση στον κάθετο άξονα Y.

Για την καλύτερη κατανόηση της παραπάνω διαδικασίας ακολουθεί ένα παράδειγμα με τον πυρήνα Sobel (Σχήμα 9). Η διαδικασία είναι αντίστοιχη για τον πυρήνα Scharr (Σχήμα 10).

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Σχήμα 9: Πυρήνες Sobel

$$G_x = \begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix}, G_y = \begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$

Σχήμα 10: Πυρήνες Scharr

Για μια περιοχή $I_{i,j}$ μιας εικόνας μεγέθους 3x3 πολλαπλασιάζουμε όλα τα στοιχεία (i, j) με τα αντίστοιχα στοιχεία (x, y) των πυρήνων G_x και G_yκαι υπολογίζουμε το άθροισμά τους.

$$I_{i,j} = \begin{bmatrix} 93 & 139 & 101 \\ 26 & 252 & 196 \\ 135 & 230 & 18 \end{bmatrix}$$

$$G_x = \sum \begin{bmatrix} -1 \times 93 & 0 \times 139 & 1 \times 101 \\ -2 \times 26 & 0 \times 252 & 2 \times 196 \\ -1 \times 135 & 0 \times 230 & 1 \times 18 \end{bmatrix} = \sum \begin{bmatrix} -93 & 0 & 101 \\ -52 & 0 & 392 \\ -135 & 0 & 18 \end{bmatrix} = 231$$

$$G_y = \sum \begin{bmatrix} -1 \times 93 & -2 \times 139 & -1 \times 101 \\ 0 \times 26 & 0 \times 252 & 0 \times 196 \\ 1 \times 135 & 2 \times 230 & 1 \times 18 \end{bmatrix} = \sum \begin{bmatrix} -93 & -278 & -101 \\ 0 & 0 & 0 \\ 135 & 460 & 18 \end{bmatrix} = 141$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.7}$$

$$\theta = \arctan 2(G_y, G_x) \times \frac{180}{\pi} \tag{2.8}$$

Από την σχέση (2.7) προκύπτει το μέγεθος της κλίσης:

$$G = \sqrt{231^2 + 141^2} = 270.63$$

και από την σχέση (2.8) η κατεύθυνση (γωνία) θ:

$$\theta = \arctan 2(141, 231) \times \frac{180}{\pi} = 31.4^\circ$$

Εφαρμόζοντας τα παραπάνω στην εικόνα 27 προκύπτουν δύο νέες εικόνες, αντίστοιχα για τους πυρήνες G_x και G_y.

Εικόνα 27: Αποτελέσματα εφαρμογής πυρήνων Sobel. Αρχική εικόνα (αριστερά), Πυρήνας Gx (μέση), Πυρήνας Gy (δεξιά)

Από τον συνδυασμό των δύο εικόνων προκύπτει η εικόνα 28.

Εικόνα 28: Συνδυασμός αποτελεσμάτων του πυρήνα Sobel

2.1.9 Περιγράμματα (Contours)

Είδαμε σε προηγούμενο κεφάλαιο πώς με τη χρήση μεθόδων κατωφλιού καταφέραμε να εντοπίσουμε αντικείμενα σε εικόνες και να απομονώσουμε τις περιοχές τους. Με τη χρήση περιγραμμάτων μπορούμε να αποκτήσουμε πρόσβαση στην περιφέρεια αυτών των αντικειμένων και να εκμεταλλευτούμε βασικές και σύνθετες ιδιότητες αυτών. Αρκετές φορές η χρήση περιγραμμάτων μπορεί να μας δώσει λύσεις σε προβλήματα που αλλιώς θα χρειαζόντουσαν την χρήση τεχνικών μηχανικής μάθησης.

2.1.9.1 Εύρεση και σχεδίαση περιγραμμάτων

Πριν μπορέσουμε να μελετήσουμε τις ιδιότητες των περιγραμμάτων πρέπει πρώτα να τα εντοπίσουμε εντός μιας εικόνας. Για να βεβαιωθούμε και να αντιληφθούμε πως εντοπίστηκαν σωστά, μπορούμε να τα σχεδιάσουμε με κάποιο χρώμα.

Πολύ απλά με μια κλήση της μεθόδου *cv2.findContours()* μπορούμε να εντοπίσουμε τα περιγράμματα σε μια εικόνα. Η μέθοδος δέχεται μόνο 3 ορίσματα:

- την εικόνα
- ποια περιγράμματα θέλουμε να μας επιστρέψει
    - εξωτερικά περιγράμματα
    - όλα τα περιγράμματα χωρίς κάποια ιεραρχία
    - όλα τα περιγράμματα με ιεραρχία
- σε τι μορφή να επιστρέψει τα περιγράμματα
    - όλα τα σημεία των περιγραμμάτων
    - τις συντεταγμένες των ακμών

και επιστρέφει δύο μεταβλητές (η δεύτερη είναι προαιρετική):

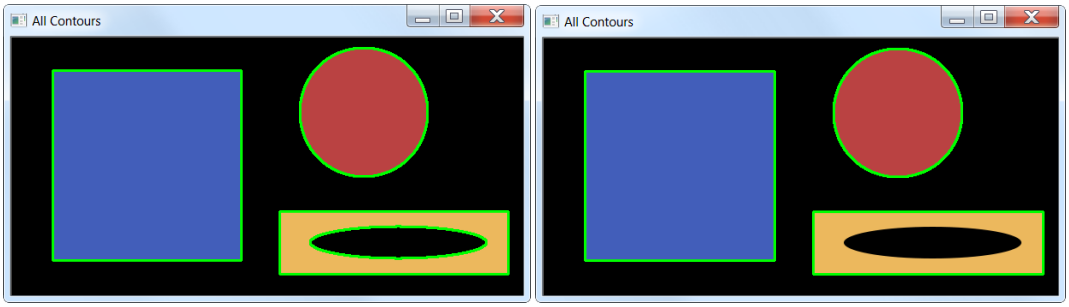
- τα περιγράμματα
- την ιεραρχία των περιγραμμάτων, όπου περιέχονται πληροφορίες όπως η τοπολογία τους πχ. αν ένα περίγραμμα βρίσκεται μέσα σε ένα άλλο. Η μεταβλητή είναι προαιρετική και εξαρτάται από το δεύτερο όρισμα της μεθόδου. Συνήθως αυτή η μεταβλητή δεν μας απασχολεί.

Τις περισσότερες φορές αυτό που μας ενδιαφέρει είναι τα εξωτερικά περιγράμματα και επιθυμούμε μόνο τις συντεταγμένες των ακμών καθώς είναι πιο γρήγορη διαδικασία.

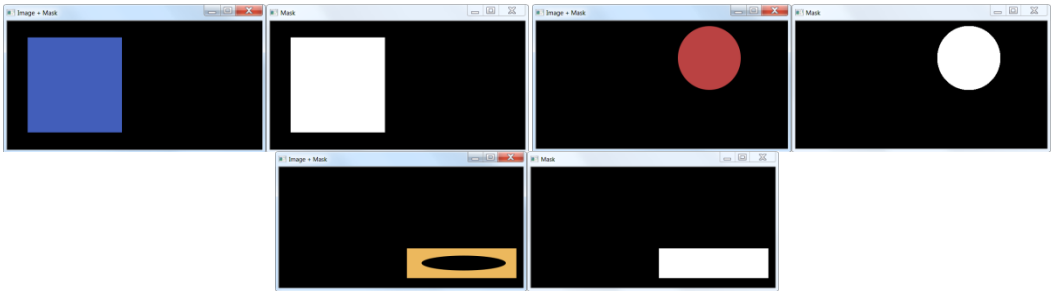
Αντίστοιχη μέθοδος είναι η *cv2.drawContours()* με την οποία μπορούμε να σχεδιάσουμε τα περιγράμματα που έχουν εντοπιστεί με την *cv2.findContours()*. Αυτή η μέθοδος δεν επιστρέφει κάποια μεταβλητή και δέχεται 5 ορίσματα:

- την εικόνα στην οποία θα σχεδιαστούν τα περιγράμματα
- μια λίστα με τα περιγράμματα
- τον αριθμό των περιγραμμάτων που θα σχεδιαστούν. Συνηθίζεται εδώ να δίνουμε την τιμή -1 που σημαίνει όλα τα περιγράμματα
- το χρώμα του περιγράμματος
- το πάχος του περιγράμματος

Στις παρακάτω εικόνες βλέπουμε κάποια παραδείγματα που ακολουθούν όσα περιγράψαμε έως τώρα σε αυτό το κεφάλαιο. Παρατηρούμε στην εικόνα 29 στα αριστερά πως έχει εντοπιστεί και τα εσωτερικό περίγραμμα σε σχήμα έκλειψης του παραλληλόγραμμου, ενώ στην δεξιά έχουν εντοπιστεί μόνο τα εξωτερικά περιγράμματα. Επίσης στην εικόνα 30 βλέπουμε πως με την χρήση μασκών καταφέραμε να απομονώσουμε τα αντικείμενα και να τα τυπώσουμε μόνα τους σε μια νέα εικόνα.



Εικόνα 29: Παράδειγμα εντοπισμού περιγραμμάτων. Αριστερά όλα τα περιγράμματα, Δεξιά μόνο τα εξωτερικά περιγράμματα.



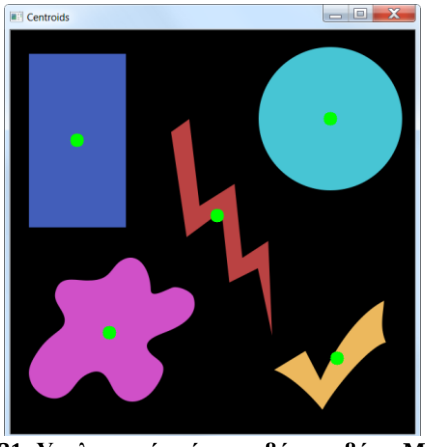
Εικόνα 30: Εφαρμογή μασκών σε περιγράμματα για την απομόνωση αυτών

2.1.9.2 Βασικές ιδιότητες περιγραμμάτων

Από τα περιγράμματα προκύπτουν αρκετές ιδιότητες η οποίες μπορούν να φανούν χρήσιμες σε πολλές εφαρμογές. Κάποιες από τις βασικές ιδιότητες περιγράφονται παρακάτω.

2.1.9.2.1 Κεντρικό σημείο αντικειμένου

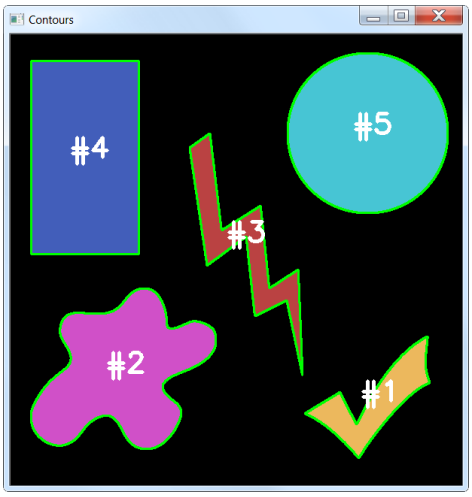
Το κεντρικό σημείο ενός αντικειμένου μπορεί να υπολογιστεί εύκολα με τη χρήση των *Moments* (Ming-Kuei Hu, 1962) τα οποία βασίζονται στον σταθμισμένο μέσο όρο των εντάσεων των πίξελ κατά μήκος του περιγράμματος (συντεταγμένες x, y).



Εικόνα 31: Υπολογισμός κέντρου βάρους βάση Moments

2.1.9.2.2 Εμβαδόν και Περίμετρος

Ο υπολογισμός του εμβαδού και της περιμέτρου ενός αντικειμένου βάση του περιγράμματός του γίνεται εύκολα με τις μεθόδους *cv2.contourArea()* και *cv2.arcLength()* αντίστοιχα. Η πρώτη δέχεται μόνο ένα όρισμα, το περίγραμμα. Η δεύτερη δέχεται ως πρώτο όρισμα το περίγραμμα και ως δεύτερο μια μεταβλητή τύπου boolean πως δηλώνει αν το περίγραμμα είναι κλειστό, δηλαδή αν το περίγραμμα είναι συνεχόμενο χωρίς κενά.



Εικόνα 32: Υπολογισμός εμβαδού και περιγράμματος

Στην εικόνα 32 παρατηρούμε πως έχουμε αριθμήσει τα περιγράμματα βάση της σειράς που έχουν εντοπιστεί. Το σημείο στο οποίο εμφανίζεται η αρίθμηση είναι το κεντρικό σημείο κάθε αντικειμένου και έχει υπολογιστεί μέσω των Moments. Για λόγους πληρότητας παραθέτονται παρακάτω το εμβαδόν και η περίμετρος που έχουν υπολογιστεί και έχουν τυπωθεί στο τερματικό:

1. Εμβαδόν: 8049.50, Περίμετρος: 577.39
2. Εμβαδόν: 30873.00, Περίμετρος: 893.32
3. Εμβαδόν: 10435.50, Περίμετρος: 1038.54
4. Εμβαδόν: 36751.00, Περίμετρος: 800.00
5. Εμβαδόν: 35702.00, Περίμετρος: 705.07

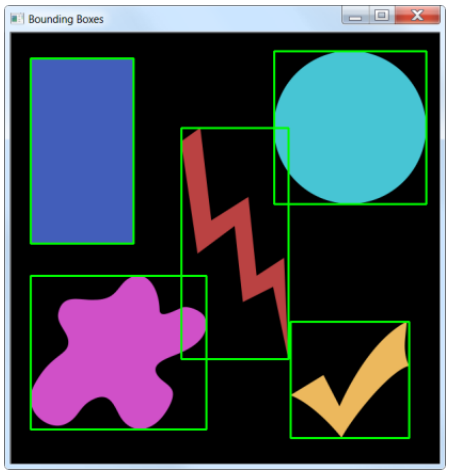
2.1.9.2.3 Πλαίσιο οριοθέτησης

Το πλαίσιο οριοθέτησης είναι ένα παραλληλόγραμμο το οποίο περικλείει το αντικείμενο και μπορεί να εντοπιστεί με την μέθοδο *cv2.boundingRect()*. Η μέθοδος δέχεται ως όρισμα ένα περίγραμμα και επιστρέφει 4 μεταβλητές:

- συντεταγμένη X της πάνω αριστερής ακμής

- συντεταγμένη Y της πάνω αριστερής ακμής
- W, το πλάτος του πλαισίου
- H, το ύψος του πλαισίου

Το πλαίσιο οριοθέτησης (εικόνα 33) είναι χρήσιμο σε περιπτώσεις που μας ενδιαφέρει η παραλληλόγραμμη περιοχή που περικλείει ένα αντικείμενο. Την μέθοδο αυτήν θα την χρησιμοποιήσουμε κατά την αποκοπή των χαρακτήρων από την πινακίδα κυκλοφορίας καθώς πέρα από τον ίδιο τον χαρακτήρα μας ενδιαφέρει και το παρασκήνιο.



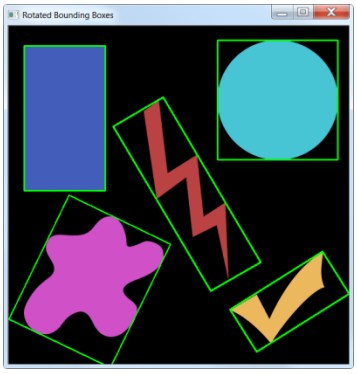
Εικόνα 33: Παράδειγμα πλαισίου οριοθέτησης

2.1.9.2.4 Περιστρεφόμενο πλαίσιο οριοθέτησης

Το περιστρεφόμενο πλαίσιο οριοθέτησης είναι το πλαίσιο οριοθέτησης με το μικρότερο εμβαδόν. Για τον εντοπισμό του γίνεται χρήση δύο μεθόδων της *cv2.minAreaRect()* και της *cv2.cv.BoxPoints()*. Η πρώτη δέχεται ένα περίγραμμα και επιστρέφει ένα νέο περιστρεφόμενο, συγκεκριμένα επιστρέφει:

- συντεταγμένη X της πάνω αριστερής ακμής
- συντεταγμένη Y της πάνω αριστερής ακμής
- W, το πλάτος του πλαισίου
- H, το ύψος του πλαισίου
- την γωνία περιστροφής του πλαισίου

Η δεύτερη, δέχεται τα παραπάνω ως ορίσματα και τα μετατρέπει σε σημεία τέτοια ώστε να σχηματίζουν το τελικό περιστρεφόμενο πλαίσιο οριοθέτησης (εικόνα 34).



Εικόνα 34: Παράδειγμα περιστρεφόμενου πλαισίου οριοθέτησης

2.1.9.3 Σύνθετες ιδιότητες περιγραμμάτων

Από τις βασικές ιδιότητες περιγραμμάτων προκύπτουν κάποιες πιο σύνθετες που μπορούν να χρησιμοποιηθούν για την αναγνώριση ή διαχωρισμό μεταξύ σχημάτων όπως θα δούμε παρακάτω.

2.1.9.3.1 Λόγος διαστάσεων

Αν και είναι απλός ο τρόπος υπολογισμού του λόγου των διαστάσεων ενός αντικειμένου, μπορεί να φανεί χρήσιμος στον διαχωρισμό μεταξύ αντικειμένων όπως ενός παραλληλόγραμμου και ενός τετραγώνου. Η πρακτική αυτή θα εφαρμοστεί αργότερα για τον εντοπισμό και την αποκοπή των χαρακτήρων από τις πινακίδες κυκλοφορίας.

Ο λόγος διαστάσεων προκύπτει από τον τύπο:

$$\text{λόγος διαστάσεων} = \text{πλάτος}/\text{ύψος} \tag{2.9}$$

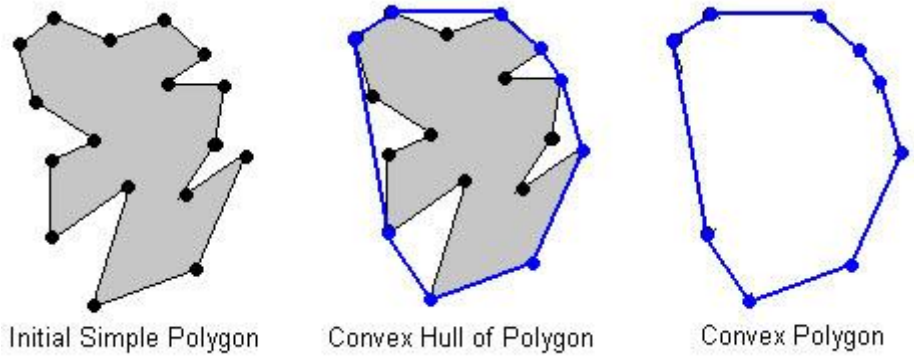
Τύπος υπολογισμού λόγου διαστάσεων

Από την σχέση (2.9) προκύπτει πως αντικείμενα με λόγο μικρότερο ή μεγαλύτερο της μονάδας είναι παραλληλόγραμμα. Συγκεκριμένα έχουν ύψος μεγαλύτερο από ότι φάρδος αν ο λόγος είναι μικρότερος της μονάδας και φάρδος μεγαλύτερο του ύψους αν ο λόγος είναι μεγαλύτερος της μονάδας. Αν ο λόγος είναι ίσως με τη μονάδα, ή πολύ κοντά σε αυτήν, τότε προκύπτει πως το σχήμα είναι είτε κύκλος είτε τετράγωνο.

2.1.9.3.2 Κυρτό περίβλημα

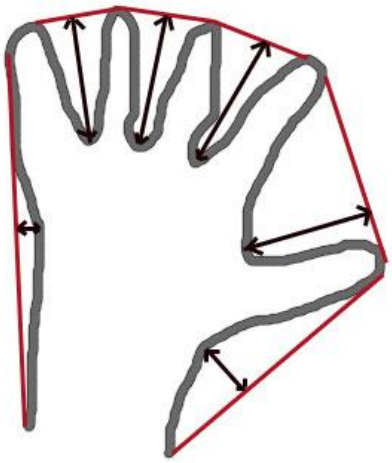
Δεδομένου ενός συνόλου Χ σημείων στον Ευκλείδειο χώρο, το κυρτό περίβλημα είναι το μικρότερο δυνατό κυρτό σύνολο που περιέχει αυτά τα Χ σημεία. Ο ορισμός γίνεται καλύτερα κατανοητός βλέποντας τα σχήματα.

Στο σχήμα 11 βλέπουμε ένα πολύγωνο του οποίου το κυρτό περίβλημα προκύπτει αν ενώσουμε τις κορυφές του έτσι ώστε να δημιουργηθεί το μικρότερο δυνατό κυρτό σύνολο. Θα μπορούσαμε να πούμε πως είναι σαν να "τυλίγουμε" το αντικείμενο με ένα λάστιχο. Η περιοχή που δημιουργεί το λάστιχο είναι το κυρτό περίβλημα.



Σχήμα 11: Παράδειγμα κυρτού περιβλήματος (πολύγωνο)

Ένα ακόμα παράδειγμα είναι αυτό του σχήματος 12 όπου είναι σχεδιασμένη μια παλάμη. Με κόκκινο χρώμα είναι σχεδιασμένο το κυρτό περίβλημα ενώ τα βελάκια ανάμεσα στα δάκτυλα επισημαίνουν την "έλλειψη κυρτότητας" (convexity defect). Η έλλειψη κυρτότητας δημιουργείτε όταν η κυρτότητα είναι προς τα μέσα και σε συνδυασμό με το κυρτό περίβλημα βοηθάνε στην αναγνώριση χειρονομιών ή στην μέτρηση των αριθμών των δακτύλων.



Σχήμα 12: Παράδειγμα κυρτού περιβλήματος (παλάμη). Τα βελάκια επισημαίνουν την "έλλειψη κυρτότητας"

Η ιδιότητα αυτή θα μας βοηθήσει να απομονώσουμε τους χαρακτήρες των πινακίδων κυκλοφορίας, καθώς όπως είπαμε η μέθοδος δημιουργεί το μικρότερο δυνατό πολύγωνο ενός σχήματος.

2.1.9.3.3 Στερεότητα

Από την παραπάνω ιδιότητα προκύπτει η ιδιότητα της στερεότητας, ο τύπος της οποίας είναι:

$$στερεότητα = περιοχή\ περιγράμματος\ /\ περιοχή\ κυρτού\ περιβλήματος \quad (2.10)$$

Η ιδιότητα της στερεότητας μπορεί να φανεί χρήσιμη στην αναγνώριση αντικειμένων αρκεί να εξετάσουμε το εύρος τιμών που μπορεί να έχει το κάθε σχήμα.

2.1.10 Ανάλυση Συνδεδεμένων Συνιστωσών (Connected Component Analysis)

Η επισήμανση συνδεδεμένων εξαρτημάτων (γνωστή και ως ανάλυση συνδεδεμένων συνιστωσών, εξαγωγή άμορφων μαζών ή επισήμανση περιοχής) είναι μια αλγοριθμική εφαρμογή της θεωρίας γραφημάτων που χρησιμοποιείται για τον προσδιορισμό της σύνδεσης των περιοχών που μοιάζουν με άμορφες μάζες σε μια δυαδική εικόνα (Pfaltz & Rosenfeld, 1966).

Συχνά χρησιμοποιούμε ανάλυση συνδεδεμένων συνιστωσών στις ίδιες περιπτώσεις που χρησιμοποιούνται τα περιγράμματα. Ωστόσο, η επισήμανση συνδεδεμένων συνιστωσών μπορεί συχνά να μας δώσει ένα πιο κοκκώδες φιλτράρισμα των κηλίδων σε μια δυαδική εικόνα. Όταν χρησιμοποιούμε την ανάλυση περιγράμματος, είμαστε συχνά περιορισμένοι από την ιεραρχία των περιγραμμάτων (δηλ. ένα περίγραμμα που περιέχεται στο άλλο), αλλά με την ανάλυση των συνδεδεμένων συνιστωσών μπορούμε να χωρίσουμε και να αναλύσουμε πιο εύκολα αυτές τις δομές.

Μετά την εφαρμογή της ανάλυσης συνδεδεμένων συνιστωσών μπορούμε ακόμα να εφαρμόσουμε ιδιότητες περιγράμματος για να ποσοτικοποιήσουμε την περιοχή. Ένα καλό παράδειγμα χρήσης της ανάλυσης συνδεδεμένων συνιστωσών είναι ο υπολογισμός των συνδεδεμένων στοιχείων μιας δυαδικής (δηλ. κατωφλιού) πινακίδας κυκλοφορίας και το φιλτράρισμα των κηλίδων βάση τις ιδιότητές τους, όπως πλάτος, ύψος, επιφάνεια, στερεότητα κλπ.

2.2 *Περιγραφείς εικόνων*

Η διαδικασία της εξαγωγής χαρακτηριστικών (Feature Extraction) διέπει τους κανόνες, τους αλγορίθμους και τις μεθοδολογίες που χρησιμοποιούμε για την ποσοτική εκτίμηση των περιεχομένων μιας εικόνας χρησιμοποιώντας μόνο μια λίστα αριθμών, που ονομάζεται διάνυσμα χαρακτηριστικών (Feature Vector).

Οι περιγραφείς εικόνων (Image Descriptors) και οι περιγραφείς χαρακτηριστικών (Feature Descriptors) καθορίζουν τον τρόπο με τον οποίο γίνεται η ποσοτική εκτίμηση των περιεχομένων μιας εικόνας, ενώ το αποτέλεσμα αυτής της διαδικασίας ονομάζεται διάνυσμα χαρακτηριστικών (σχήμα 13).

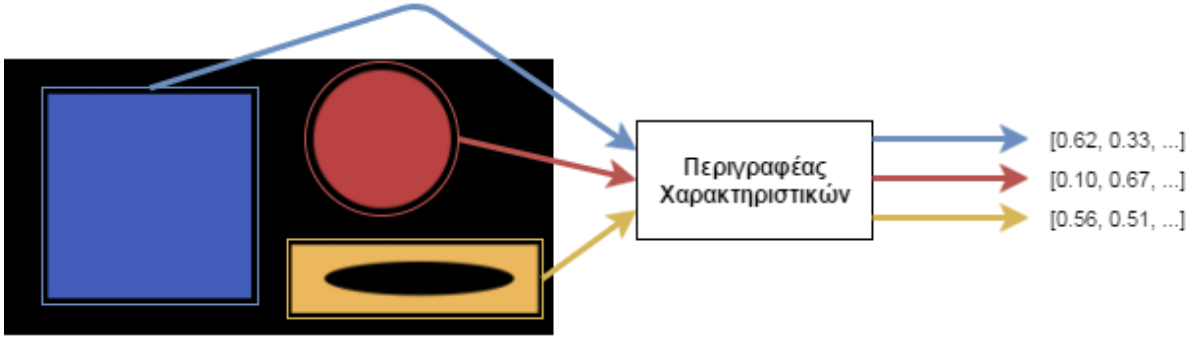
Σχήμα 13: Διαδικασία εξαγωγής διανύσματος χαρακτηριστικών

Υπάρχει μια λεπτή διαφοροποίηση μεταξύ του περιγραφέα εικόνας και του περιγραφέα χαρακτηριστικών η οποία ξεκαθαρίζει με τους παρακάτω ορισμούς:

- Ένας περιγραφέας εικόνας είναι ένας αλγόριθμος και μεθοδολογία που διέπει τον τρόπο με τον οποίο μια εικόνα εισόδου προσδιορίζεται ποσοτικά, σε παγκόσμια κλίμακα (globally), και επιστρέφει ένα διάνυσμα χαρακτηριστικών που αντιπροσωπεύει αφηρημένα τα περιεχόμενα της εικόνας (σχήμα 14).

Σχήμα 14: Περιγραφέας εικόνας

- Ο περιγραφέας χαρακτηριστικών είναι ένας αλγόριθμος και μια μεθοδολογία που καθορίζει τον τρόπο τοπικής ποσοτικοποίησης μιας περιοχής εισόδου μιας εικόνας. Ένας περιγραφέας χαρακτηριστικών δέχεται μια ενιαία εικόνα εισόδου και επιστρέφει πολλαπλά διανύσματα χαρακτηριστικών (σχήμα 15).



Σχήμα 15: Περιγραφέας χαρακτηριστικών

Η εξαγωγή χαρακτηριστικών έχει εφαρμογή σε πολλούς τομείς όπως είναι η κατάταξη εικόνων που επιστρέφονται από μια μηχανή αναζήτησης ή η εκπαίδευση ενός ταξινομητή για την αναγνώριση του περιεχομένου σε μια εικόνα. Για την επίτευξη των παραπάνω συγκρίσεων είναι απαραίτητα τα διανύσματα χαρακτηριστικών ή κάποια χαρακτηριστικά.

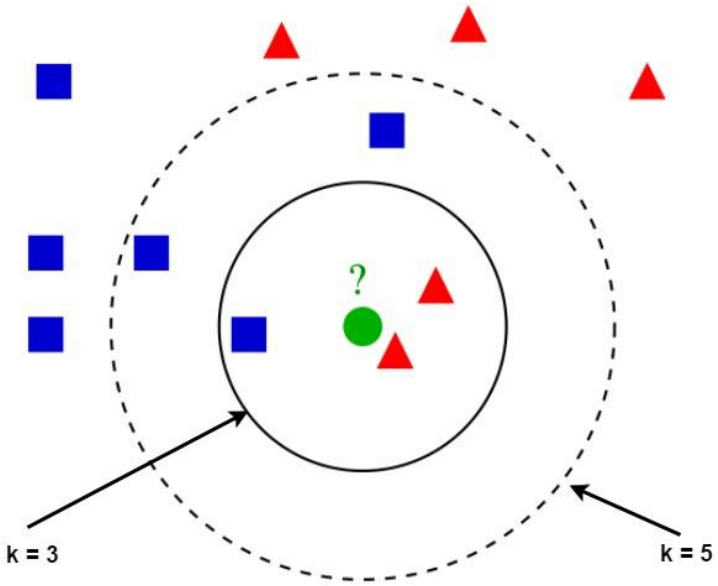
Τα διανύσματα χαρακτηριστικών χρησιμοποιούνται για να αντιπροσωπεύουν μια ποικιλία ιδιοτήτων μιας εικόνας, όπως το σχήμα, το χρώμα ή η υφή ενός αντικειμένου σε μια εικόνα. Μπορούν επίσης να συνδυάζουν διάφορες ιδιότητες. Ένα διάνυσμα χαρακτηριστικών μπορεί να αντιπροσωπεύει τόσο το σχήμα όσο και το χρώμα ταυτόχρονα ή θα μπορούσε να αντιπροσωπεύει την υφή και το σχήμα ή και τα τρία!

2.3 *Μηχανική Μάθηση*

2.3.1 Μέθοδοι Μηχανικής Μάθησης

2.3.1.1 k - Πλησιέστεροι Γείτονες (k - Nearest Neighbors - kNN)

Η μέθοδος των πλησιέστερων γειτόνων είναι μακράν ο απλούστερος αλγόριθμος ταξινόμησης. Στην πραγματικότητα, είναι τόσο απλός που δεν "μαθαίνει" τίποτα. Αντίθετα, αυτός ο αλγόριθμος απλώς βασίζεται στην απόσταση μεταξύ των διανυσμάτων χαρακτηριστικών. Παρ' όλα αυτά εξακολουθεί να χρησιμοποιείται σε μεγάλο βαθμό σε πολλούς αλγόριθμους υπολογιστικής όρασης. Με απλά λόγια, ο αλγόριθμος k-NN ταξινομεί άγνωστα δεδομένα βρίσκοντας την πιο κοινή κλάση μεταξύ των πλησιέστερων k γειτόνων (Silverman & Jones, 1989).



Σχήμα 16: Αλγόριθμος ταξινόμησης k - Πλησιέστεροι Γείτονες (k - Nearest Neighbors - kNN)

Οι πιο συνηθισμένες μετρικές απόστασης είναι:

- Ευκλείδεια
- Μανχάταν
- Minkowski
- Hamming

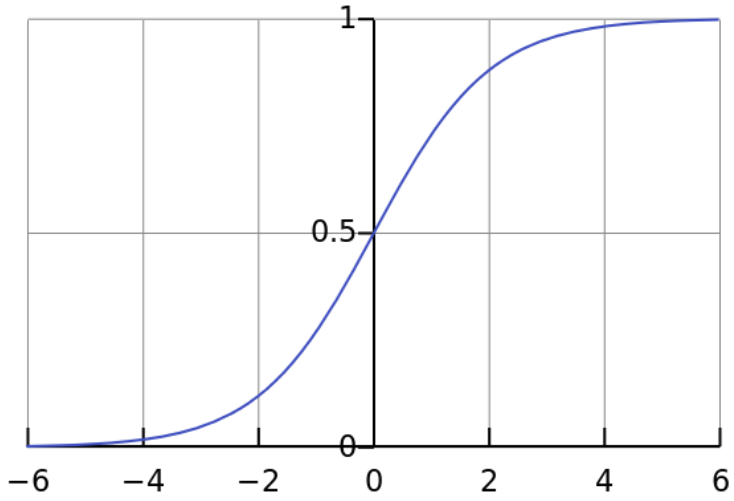
Εκτός από την μέθοδο υπολογισμού της απόστασης, σημαντικό ρόλο έχει η επιλογή της τιμής k. Για την αποφυγή συγκρούσεων μεταξύ διαφορετικών κλάσεων συνηθίζεται η τιμή να είναι μονός αριθμός. Εάν η τιμή του k είναι πολύ μικρή (k=1), τότε κερδίζουμε αποδοτικότητα αλλά ο αλγόριθμος γίνεται ευαίσθητος σε δεδομένα που μπορούν να αποτελέσουν θόρυβο. Ωστόσο, αν το k είναι πολύ μεγάλο, τότε μπορεί να υπερ-εξομαλυνθούν τα αποτελέσματα της ταξινόμησης να και να αυξηθεί η μετατόπιση (*bias*).

Τέλος, ο αλγόριθμος k-NN είναι πιο κατάλληλος για χώρους χαμηλών διαστάσεων. Οι αποστάσεις στους χώρους μεγάλων διαστάσεων είναι συχνά απροσδιόριστες (Domingos, 2012).

2.3.1.2 Λογιστική παλινδρόμηση (Logistic Regression)

Η Λογιστική Παλινδρόμηση, παρόλο που παραπέμπει σε παλινδρόμηση, είναι ένα γραμμικό μοντέλο ταξινόμησης. Η λογιστική συνάρτηση (*logistic function*) που χρησιμοποιεί (2.11) για την ταξινόμηση είναι η σιγμοειδή συνάρτηση (*sigmoid function*) (σχήμα 17).

$$s(t) = \frac{1}{1 + e^{-t}} \tag{2.11}$$



Σχήμα 17: Γράφημα σιγμοειδούς συνάρτησης (sigmoid function)

Για να ταξινομήσουμε δεδομένα που περιέχουν δύο κλάσεις πρέπει πρώτα να κωδικοποιήσουμε τις δύο κλάσεις. Ας υποθέσουμε πως έχουμε να ταξινομήσουμε εικόνες που περιέχουν γάτες και σκύλους. Αντιστοιχούμε τις γάτες στην τιμή 0 και τους σκύλους στην τιμή 1. Για κάθε εικόνα εξάγουμε ένα διάνυσμα χαρακτηριστικών *u* (2.12) και υπολογίζουμε το άθροισμα του γινομένου κάθε χαρακτηριστικού u_i με ένα βάρος w_i (2.13). Όλα τα βάρη αρχικοποιούνται με τιμή 1.

$$u = [\, u_0, u_1, u_2, \ldots, u_n \,] \tag{2.12}$$

$$x = \sum_i w_i u_i \tag{2.13}$$

$$x = w_0 u_0 + w_1 u_1 + w_2 u_2 + \ldots + w_n u_n \tag{2.14}$$

$$x = w^T u \tag{2.15}$$

Έπειτα το άθροισμα *x* εισάγεται στην σιγμοειδή συνάρτηση. Το αποτέλεσμα είναι ένας πραγματικός αριθμός μεταξύ του μηδενός και της μονάδας (2.16) και η ταξινόμηση γίνεται βάση της σχέσης 2.17.

$$0 \leq s(x) \leq 1 \tag{2.16}$$

$$s(x) = \begin{cases} 1\,(cat), & s(x) \geq 0.5 \\ 0\,(dog), & s(x) < 0.5 \end{cases} \quad (2.17)$$

Για την καλή απόδοση της μεθόδου πρέπει να οριστούν κατάλληλες τιμές στα βάρη *w* το οποίο επιτυγχάνεται με την κατάβαση και ανάβαση δυναμικού (*gradient descent and ascent*).

2.3.1.3 Κατάβαση και ανάβαση δυναμικού (Gradient Descent and Ascent)

Η κατάβαση και ανάβαση δυναμικού ή αλλιώς γνωστή και ως Μαζική Κατάβαση/Ανάβαση Δυναμικού (*Batch Gradient Descent and Ascent*) είναι απλοί αλγόριθμοι βελτιστοποίησης που βασίζονται στην παράγωγο της συνάρτησης βελτιστοποίησης και υπολογίζουν τοπικά μέγιστα και ελάχιστα αντίστοιχα.

Για την εφαρμογή κατάβασης δυναμικού για την μέθοδο της λογιστικής παλινδρόμησης στο παράδειγμα της προηγούμενης παραγράφου ακολουθούμε τα παρακάτω βήματα:

1. Εξάγουμε διάνυσμα χαρακτηριστικών για **όλες** τις εικόνες στο σύνολο δεδομένων
2. Αρχικοποιούμε τα βάρη *w* στην τιμή 1
3. Για N επαναλήψεις ή μέχρι να συγκλίνει η μέθοδος
    a. Υπολογίζουμε την κλίση (*gradient*) **για όλα** τα δεδομένα
    b. Ενημερώνουμε τα βάρη βάση των τρέχων τιμών του *w*, κλίσης και ρυθμού εκπαίδευσης *a*
4. Επιστρέφουμε τα βάρη *w*

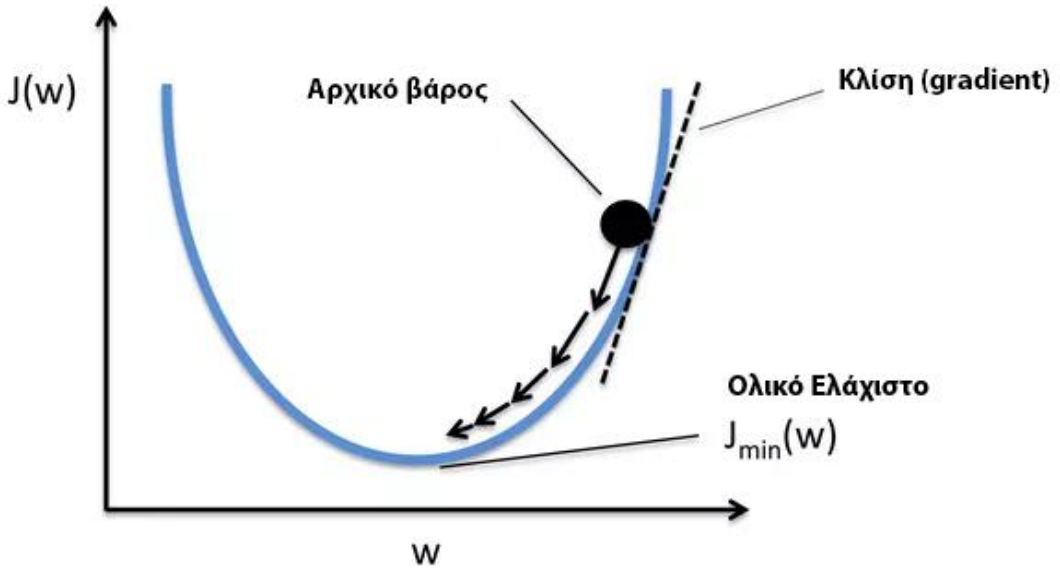
Η κλίση που αναφέρεται στο βήμα 3a είναι το σφάλμα *E* των δεδομένων εκπαίδευσης και υπολογίζεται από τη σχέση:

$$E = (L - s(F_t \times w)) \quad (2.18)$$

όπου *L* είναι μια τιμή από το σύνολο [0, 1] που είχαμε ορίσει για τις κλάσεις μας. Τα νέα βάρη προκύπτουν από τον τύπο:

$$w = w + (\alpha \times (F_t^T \times E)) \quad (2.19)$$

Το *α* ονομάζεται ρυθμός εκπαίδευσης και ελέγχει το μέγεθος του βήματος. Μεγάλες τιμές του *α* έχουν ως αποτέλεσμα να μαθαίνει η μέθοδος πιο γρήγορα αλλά μπορεί να "προσπεράσει" τις βέλτιστες τιμές. Αντίστοιχα μικρές τιμές έχουν ως αποτέλεσμα η μέθοδος να μαθαίνει με πιο αργούς ρυθμούς αλλά μπορεί να "κολλήσει" σε τοπικά ελάχιστα/μέγιστα και να αποτύχει να βρει τις βέλτιστες τιμές για τα βάρη *w*.

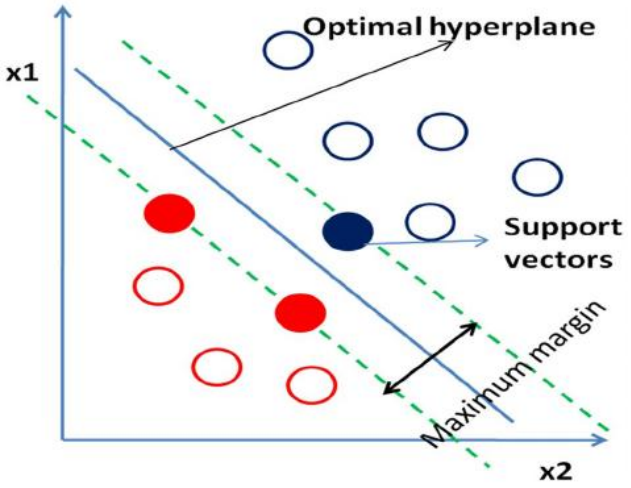


Σχήμα 18: Εύρεση ελαχίστου συνάρτησης με κατάβαση δυναμικού (gradient descent)

2.3.1.4 Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines)

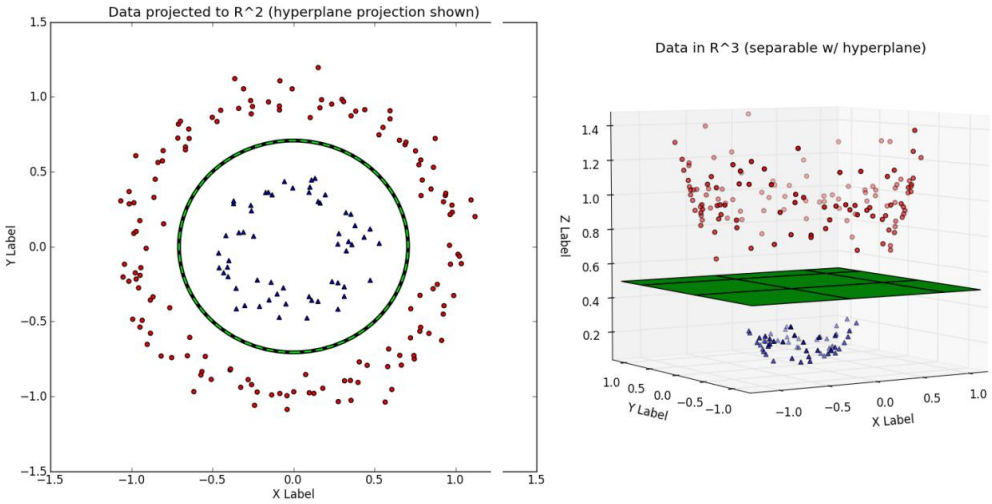
Οι Μηχανές Διανυσμάτων Υποστήριξης είναι μοντέλα μηχανικής μάθησης τα οποία χρησιμοποιούνται για ταξινόμηση *(classification)* και παλινδρόμηση *(regression)*. Το αρχικό μοντέλο υποστήριζε μόνο γραμμικά διαχωρίσιμα δεδομένα (Vapnik & Lerner, 1963) ενώ η εισαγωγή υποστήριξης μη γραμμικών διαχωρίσιμων δεδομένων έγινε αρκετά χρόνια αργότερα χρησιμοποιώντας μια μέθοδο γνωστή ως "τέχνασμα του πυρήνα" *(kernel trick)* (Boser, Guyon & Vapnik, 1992).

Στην περίπτωση της ταξινόμησης, η μέθοδος προσπαθεί να βρει το βέλτιστο υπερεπίπεδο (*optimal hyperplane*) που διαχωρίζει τα δεδομένα σε δύο κλάσεις, αφήνοντας το μέγιστο δυνατό περιθώριο (*maximum margin*) ανάμεσα στις δύο διαχωριζόμενες κλάσεις. Τα δεδομένα που βρίσκονται πιο κοντά στο μέγιστο δυνατό περιθώριο ονομάζονται Διανύσματα Υποστήριξης (σχήμα 19).



Σχήμα 19: Διάνυσμα Υποστήριξης

Όταν τα δεδομένα είναι μη-γραμμικά διαχωρίσιμα η μέθοδος προσπαθεί να τα διαχωρίσει ανάγοντας το πρόβλημα σε χώρο υψηλότερων διαστάσεων όπου μπορεί να είναι πλέον γραμμικά διαχωρίσιμα (σχήμα 20).



Σχήμα 20: Προβολή μη-γραμμικά διαχωρίσιμων δεδομένων (αριστερά) σε χώρο υψηλότερων διαστάσεων όπου πλέον διαχωρίζονται γραμμικά (δεξιά)

Για την εφαρμογή του τεχνάσματος του πυρήνα είναι απαραίτητες συναρτήσεις πυρήνα (*kernel functions*) με τις πιο γνωστές να είναι η γραμμική, η πολυωνυμική (*Polynomial*), η σιγμοειδή (*Sigmoid*) και η συνάρτηση ακτινικής βάσης (*Radial basis function - RBF*).

Για τον διαχωρισμό περισσότερων των δύο κλάσεων χρησιμοποιείται η μέθοδος ένας-εναντίων-ενός όπου για ένα σύνολο κλάσεων N δημιουργούνται N * (N-1) / 2 ταξινομητές και ο καθένας εκπαιδεύεται για δεδομένα δύο κλάσεων (Knerr, Personnaz & Dreyfus, 1990).

Η τελευταία προσθήκη που έγινε στην μέθοδο είναι αυτή "απαλού περιθωρίου" (*soft-margin*) που είναι ένα τρόπος για την ελαχιστοποίηση του σφάλματος όταν τα δεδομένα δεν διαχωρίζονται χωρίς σφάλμα (Cortes & Vapnik, 1995).

2.3.1.5 Στοχαστική Κατάβαση Δυναμικού (Stochastic Gradient Descent - SGD)

Στην μαζική κατάβαση δυναμικού αναφέραμε πως ο αλγόριθμος ανατρέχει σε όλα τα δεδομένα πριν κάνει ένα βήμα διόρθωσης. Τα μειονεκτήματα αυτής της μεθόδου είναι πως είναι πολύ αργή (Wilson & Martinez, 2003), δεν προτείνεται για μεγάλα σύνολα δεδομένων που δεν χωράνε στην μνήμη και δεν επιτρέπει την αλλαγή/ενημέρωση το μοντέλου κατά την εκπαίδευση (Ruder, 2016).

Η διαφορά στην Στοχαστική Κατάβαση Δυναμικού, αλλιώς γνωστή και ως Αυξητική κατάβαση δυναμικού (*Incremental Gradient Descent*), είναι πως τα βάρη *w* διορθώνονται για κάθε πρότυπο εκπαίδευσης που συναντά ο αλγόριθμος. Η μέθοδος αυτή αποτελεί μια στοχαστική προσέγγιση της Μαζικής κατάβασης δυναμικού.

Η Στοχαστική κατάβαση δυναμικού έχει το πλεονέκτημα πως συγκλίνει πιο γρήγορα σε κάποιο ελάχιστο αλλά όπως και η Μαζική κατάβαση δυναμικού δεν εγγυάται ότι θα βρει το ολικό ελάχιστο. Για μεγάλα σύνολα δεδομένων προτιμάται η Στοχαστική κατάβαση δυναμικού ως μέθοδος βελτιστοποίηση (Bottou, 2010).

2.3.1.6 Μέση Στοχαστική Κατάβαση Δυναμικού (Averaged SGD)

Η Μέση στοχαστική κατάβαση δυναμικού είναι μια απλή Στοχαστική κατάβαση δυναμικού. Η ενημέρωση των παραμέτρων γίνεται όπως και στην Στοχαστική κατάβαση δυναμικού αλλά ταυτόχρονα καταγράφει και τον μέσο όρο των βαρών (σχέση 2.20).

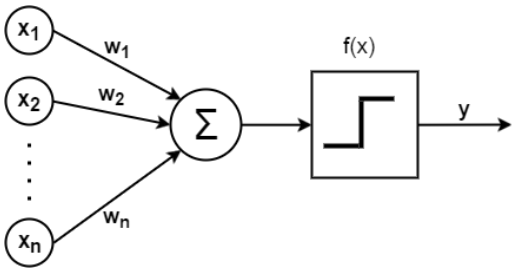
$$\overline{w} = \frac{1}{t}\sum_{i=0}^{t-1} w_i \tag{2.20}$$

Όταν ολοκληρωθεί η βελτιστοποίηση, τα βάρη *w* αντικαθίστανται με αυτά της παραπάνω σχέσης (Polyak & Juditsky, 1992).

2.3.2 Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ)

2.3.2.1 Perceptron

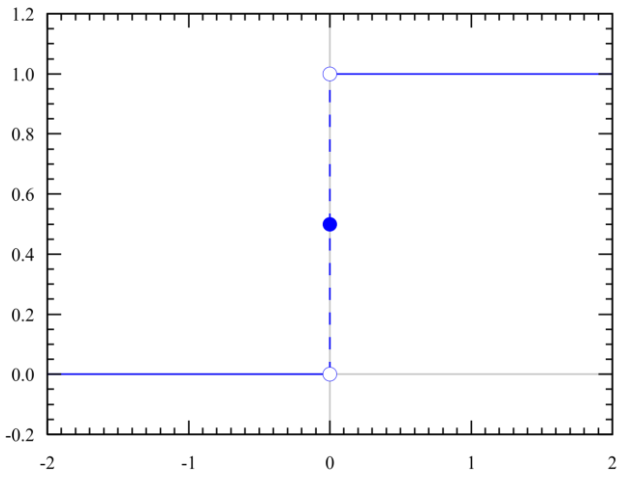
Το *Perceptron* (Rosenblatt, 1958) είναι η πιο απλή μορφή ενός Τεχνητού Νευρωνικού Δικτύου (σχήμα 21) το οποίο αποτελείται από έναν μόνο νευρώνα.



Σχήμα 21: Νευρώνας Perceptron

Ο νευρώνας αποτελείται από τρία στοιχεία:

- ένα σύνολο από *n* εισόδους (συνάψεις)
- έναν αθροιστή
- μια συνάρτηση ενεργοποίησης (*activation function*) η οποία συνήθως είναι η βηματική συνάρτηση (σχήμα 22) όπου το *w* είναι ένα διάνυσμα από βάρη με πραγματικές τιμές και *x* είναι ένα διάνυσμα εισόδου. Το *b* (*bias*) είναι η μετατόπιση της συνάρτησης ενεργοποίησης και είναι ένας σταθερός όρος ο οποίος δεν εξαρτάται από καμία τιμή εισόδου (σχέση 2.21).



Σχήμα 22: Βηματική συνάρτηση

$$f(x) = \begin{cases} 1, & w \cdot x + b > 0 \\ 0, & w \cdot x + b \leq 0 \end{cases} \quad (2.21)$$

Η έξοδος του νευρώνα δίνεται από την σχέση:

$$y = f(\sum_{i=1}^{n} w_i \cdot x_i) \quad (2.22)$$

όπου $\mathbf{X_i}$ είναι το i-οστό στοιχείο του διανύσματος εισόδου και $\mathbf{W_i}$ είναι το i-οστό στοιχείο του διανύσματος βαρών.

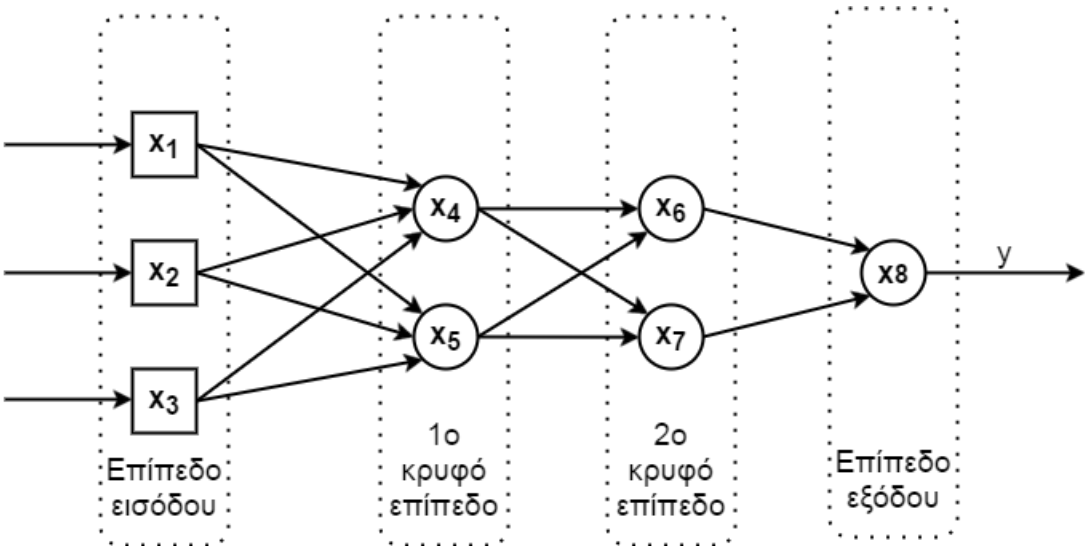
Το *Perceptron* είναι ένας δυαδικός ταξινομητής και μπορεί να χαρακτηριστεί ως ένα απλό είδος ενός εμπροσθο-τροφοδοτούμενου (*feed-forward*) νευρωνικού δικτύου το οποίο μπορεί να εκπαιδευτεί ώστε να ταξινομεί γραμμικά διαχωρίσιμα δεδομένα.

2.3.2.2 Δίκτυο Perceptron Πολλαπλών Στρωμάτων (Multi-Layer Perceptron - MLP)

Ενώ το Perceptron είναι ένας απλός και εύκολος αλγόριθμος, έχει επίσης μερικούς σοβαρούς περιορισμούς, δηλαδή το Perceptron μπορεί να υπολογίσει μόνο τις γραμμικές συναρτήσεις. Οι μη γραμμικές συναρτήσεις, όπως το XOR, δεν μπορούν να εκτιμηθούν χρησιμοποιώντας το Perceptron.

Προκειμένου να επιτευχθεί η μη γραμμική διαχωρισιμότητα, μπορούμε να χρησιμοποιήσουμε εμπροσθο-τροφοδοτούμενου δίκτυα πολλαπλών επιπέδων (*multi-layer feed-forward networks*) με μη γραμμικές συναρτήσεις ενεργοποίησης (σχήμα 23).

Ωστόσο, η μετάβαση σε μια πολυστρωματική αναπαράσταση έρχεται με κόστος - δεν μπορούμε πλέον να χρησιμοποιήσουμε τον απλό κανόνα ενημέρωσης βάρους του Perceptron. Αντίθετα, πρέπει να σχεδιάσουμε μια μέθοδο για τη "διάδοση" των σφαλμάτων στην έξοδο του δικτύου πίσω από τα κρυμμένα στρώματα. Ονομάζουμε αυτόν τον αλγόριθμο οπισθο-διάδοση (*backpropagation*), και είναι ο ακρογωνιαίος λίθος των σύγχρονων συστημάτων NN.



Σχήμα 23: Αρχιτεκτονική νευρωνικού δικτύου MLP με δύο κρυφά επίπεδα

Τα δίκτυα με περισσότερα από ένα κρυφά επίπεδα αποτελούν μια αρχιτεκτονική δικτύων που ονομάζεται Βαθιά Νευρωνικά Δίκτυα (*Deep Neural Networks*) (Goodfellow, Bengio & Courville, 2016).

3

Συστήματα Αυτόματης Αναγνώρισης Πινακίδων Κυκλοφορίας (ΣΑΑΠΚ)

Τα συστήματα αυτόματης αναγνώρισης πινακίδων κυκλοφορίας χρησιμοποιούνται για τον εντοπισμό της πινακίδας σε εικόνες και την αναγνώριση των χαρακτήρων που περιέχονται σε αυτήν. Κάθε τέτοιο σύστημα πρέπει να λειτουργεί διαφορετικά ανάλογα με την γεωγραφική περιοχή στην οποία εφαρμόζεται. Στην Αμερική οι πινακίδες κυκλοφορίας διαφέρουν ακόμα από πολιτεία σε πολιτεία, (εικόνα 35). Έτσι τα συστήματα αυτά πρέπει να προσαρμόζονται ώστε να είναι αποτελεσματικά στην εκάστοτε περιοχή.

Εικόνα 35: Πινακίδα Τέξας (2000) (Αριστερά)[2] , Πινακίδα Αλάσκα (1982) (Δεξιά)[3]

Στις παραπάνω εικόνες βλέπουμε δύο πινακίδες που δημιουργήθηκαν για εκπαιδευτικό σκοπό. Παρατηρούμε πως οι πινακίδες έχουν το ίδιο μέγεθος και η διάταξη είναι σχεδόν ίδια.

[2] http://www.acme.com/licensemaker/licensemaker.cgi?text=DOKIMH&state=Texas&r=140319134

[3] http://www.acme.com/licensemaker/licensemaker.cgi?state=Alaska&text=DOKIMH&plate=1982&r=448056791

Στο πάνω μέρος βρίσκεται το όνομα της πολιτείας στην οποία ανήκει, στο κέντρο υπάρχει ο αριθμός της πινακίδας ενώ στο φόντο υπάρχει κάποια εικόνα. Στο κάτω μέρος υπάρχει μια φράση η οποία μάλλον είναι σχετική με την εικόνα, ενώ οι χαρακτήρες και στις δύο είναι έξι στον αριθμό. Αυτό που καθιστά ξεχωριστές όμως τις πινακίδες είναι τα χρώματα της γραμματοσειράς και της ίδιας της πινακίδας. Για τον λόγο αυτόν πρέπει να χρησιμοποιήσουμε διαφορετικό σύστημα ή αλγόριθμο για κάθε μια. Η ΑΑΠΚ χωρίζεται σε πέντε βασικά βήματα που θα συζητηθούν παρακάτω.

3.1 *Λήψη της φωτογραφίας*

Το πρώτο βήμα για την ΑΑΠΚ είναι η λήψη της εικόνας η οποία πιθανώς θα περιέχει την πινακίδα που θέλουμε να εντοπίσουμε και να αναγνωρίσουμε. Τα περισσότερα συστήματα ΑΑΠΚ χρησιμοποιούν κάμερες με υπέρυθρες ώστε να μπορούν να τραβάνε φωτογραφίες ακόμα και την νύχτα. Αυτές μπορεί να είναι μέρος κάποιου κλειστού κυκλώματος μιας εταιρείας ή ενός χώρου στάθμευσης ή ακόμα και να είναι συνδεδεμένες σε ένα μεγαλύτερο δίκτυο όπου οι φωτογραφίες θα υπόκεινται σε περαιτέρω επεξεργασία. Για την σωστή λήψη της φωτογραφίας πρέπει να λάβουμε υπόψη το περιβάλλον στο οποίο θα στήσουμε την κάμερα καθώς και τις ρυθμίσεις της. Για παράδειγμα πρέπει να ορίσουμε το ερέθισμα που θα προκαλέσει την κάμερα να τραβήξει την εικόνα. Αυτό συνήθως εφαρμόζεται με ραντάρ και αισθητήρες κίνησης.

3.2 *Εντοπισμός της πινακίδας*

Τώρα που έχουμε την φωτογραφία ενός οχήματος, πρέπει να εντοπίσουμε την περιοχή που περιέχει την/τις πινακίδα/ες. Για να το επιτύχουμε, χρησιμοποιούμε βασικές μεθόδους επεξεργασίας εικόνας αν και αυτό μπορεί να γίνει με τη χρήση μεθόδων μηχανικής μάθησης. Στην εικόνα 36 βλέπουμε πως έχει εντοπιστεί σωστά η πινακίδα, της οποίας το περίγραμμα φαίνεται με έντονο πράσινο χρώμα. Την περιοχή η οποία ορίζεται από το περίγραμμα την ονομάζουμε υποψήφια πινακίδα κυκλοφορίας. Οι περιοχές αυτές επεξεργάζονται ώστε να αποκλειστούν λανθάνουσες περιοχές (false-positives) και έπειτα προχωράμε στο επόμενο βήμα, αυτό της κατάτμησης των χαρακτήρων.

Εικόνα 36: πινακίδας και δημιουργία πλαισίου γύρω από αυτήν

3.3 *Κατακερματισμός των χαρακτήρων*

Στο προηγούμενο βήμα εντοπίσαμε την περιοχή που περιέχει μια υποψήφια πινακίδα κυκλοφορίας. Για να προβούμε όμως στην αναγνώρισή της πρέπει πρώτα να κατακερματίσουμε τους χαρακτήρες της όπως φαίνεται στην εικόνα 37. Θα μπορούσαμε να χρησιμοποιήσουμε μεθόδους οπτικής αναγνώρισης χαρακτήρων (OCR[4]) αλλά οι πινακίδες κυκλοφορίας έχουν πολύ θόρυβο. Άλλες τεχνικές περιλαμβάνουν μεθόδους προσαρμοσμένου κατωφλιού (adaptive thresholding) ή ψαλιδίσματος (scissoring).

Εικόνα 37: Κατάτμηση χαρακτήρων τους οποίους θα στείλουμε στον ταξινομητή μας (classifier) για αναγνώριση

[4] http://en.wikipedia.org/wiki/Optical_character_recognition

3.4 *Αποκοπή των χαρακτήρων*

Η αποκοπή των χαρακτήρων είναι το επόμενο στάδιο της επεξεργασίας πριν την αναγνώριση. Στο προηγούμενο βήμα καταφέραμε και απομονώσαμε τους χαρακτήρες της πινακίδας με τη χρήση των κυρτών περιβλημάτων. Έχοντας τα κυρτά περιβλήματα μπορούμε εύκολα να αποκόψουμε τους χαρακτήρες, όπως φαίνεται στην εικόνα 38.

Εικόνα 38: Αποκοπή χαρακτήρων από την πινακίδα κυκλοφορίας

3.5 *Αναγνώριση των χαρακτήρων*

Αφού έχουν κοπεί οι χαρακτήρες μπορούμε να προχωρήσουμε στο τελικό στάδιο της αναγνώρισης, με μεθόδους μηχανικής μάθησης. Θα μπορούσαμε να τροφοδοτήσουμε τον ταξινομητή μας με τους χαρακτήρες όπως τους έχουμε παραλάβει από το προηγούμενο βήμα. Αντί αυτού όμως θα εισάγουμε τους χαρακτήρες μας σε έναν περιγραφέα (descriptor), μια συνάρτηση που θα μας επιστρέψει κάποια χαρακτηριστικά (features) της εικόνας. Η διαδικασία αυτή ονομάζεται εξαγωγή χαρακτηριστικών και τα χαρακτηριστικά αυτά είναι που θα εισάγουμε στον ταξινομητή. Όπως θα δούμε σε παρακάτω κεφάλαιο, ο περιγραφέας που θα χρησιμοποιήσουμε είναι ο Block Binary Pixel Sum, ενώ υπάρχουν και άλλοι του είδους του. Στην εικόνα 39 φαίνεται το τελικό αποτέλεσμα του συστήματος ΑΑΠΚ. Η αναγνώριση γίνεται χαρακτήρα-χαρακτήρα και το αποτέλεσμα του ταξινομητή τυπώνεται πάνω από την περιοχή της πινακίδας.

Εικόνα 39: Τελικό αποτέλεσμα του συστήματος ΑΑΠΚ όπου η πινακίδα αναγνωρίστηκε με επιτυχία

4

Το πρόβλημα με τα Σύνολα δεδομένων (datasets)

Στην προηγούμενη ενότητα αναφέραμε τα 5 στάδια των συστημάτων ΑΑΠΚ:

- Απόκτηση/λήψη της φωτογραφίας
- Εντοπισμός της πινακίδας
- Κατακερματισμός των χαρακτήρων
- Αποκοπή των χαρακτήρων
- Αναγνώριση των χαρακτήρων

Ένα πρόβλημα που πρέπει να λάβουμε σοβαρά υπόψη όταν φτιάχνουμε τέτοιου είδους συστήματα είναι τα σύνολα δεδομένων (dataset). Θα σκεφτόταν κανείς πως με μια απλή αναζήτηση στο διαδίκτυο θα έβρισκε αρκετές φωτογραφίες με πινακίδες. Όσο όμως και να ακούγεται περίεργο, ειδικά στην ψηφιακή εποχή που ζούμε, είναι πως δεν υπάρχουν διαθέσιμα μεγάλα σύνολα δεδομένων πινακίδων κυκλοφορίας. Ο λόγος είναι πως τα δεδομένα αυτά θεωρούνται ιδιωτικά και η δημοσίευση αυτών θεωρείται παράπτωμα σε πολλές χώρες. Η ερώτηση λοιπόν είναι πού θα βρούμε δεδομένα για να εκπαιδεύσουμε τον ταξινομητή μας;

Ένα από τα μεγαλύτερα σύνολα δεδομένων από φωτογραφίες πινακίδων κυκλοφορίας, αν όχι το μεγαλύτερο, είναι αυτό που έχει δημιουργήσει η Google για την υπηρεσία Street View. Τα οχήματα της Google έχουν σαρώσει σχεδόν όλη την υφήλιο και έχουν τραβήξει φωτογραφίες 360 μοιρών οι οποίες είναι δωρεάν διαθέσιμες. Στις φωτογραφίες αυτές, μεταξύ άλλων, απεικονίζονται εκατομμύρια οχήματα με ορατές τις πινακίδες τους. Για κακή μας τύχη όμως

αυτές είναι παραμορφωμένες, συγκεκριμένα θολές, όπως φαίνεται στην εικόνα 40. Ο λόγος είναι ο ίδιος που αναφέραμε προηγουμένως, η ιδιωτικότητα.

Εικόνα 40: Φωτογραφία από Google Street View όπου οι πινακίδες είναι θολωμένες

4.1 *Σύνολα δεδομένων (datasets)*

Στην ενότητα αυτή θα γίνει σύγκριση κάποιων συνόλων που βρέθηκαν στο διαδίκτυο και θα συζητηθεί γιατί αυτά δεν είναι χρήσιμα στην παρούσα εργασία.

4.1.1 CALTECH 2001 (Rear)

Εικόνα 41: Φωτογραφία από την συλλογή CALTECH 2001 (Rear) [5]

Η συλλογή αυτή περιέχει 526 εικόνες από το πίσω μέρος των αυτοκινήτων. Δυστυχώς η μικρή ανάλυσή τους, μόλις 360x240 πίξελ, τις καθιστά χρήσιμες μόνο για τον εντοπισμό της πινακίδας και όχι για την αναγνώριση των χαρακτήρων.

[5] http://www.vision.caltech.edu/html-files/archive.html

4.1.2 CALTECH 1999 (Rear) 2

Εικόνα 42: Φωτογραφία από την συλλογή CALTECH 1999 (Rear) 2[6]

Η ανάλυση σε αυτές τις φωτογραφίες είναι ικανοποιητική. Συνολικά υπάρχουν 126 φωτογραφίες με ανάλυση 896x592 πίξελ. Απεικονίζουν κυρίως πινακίδες από οχήματα της Καλιφόρνια των Ηνωμένων Πολιτειών. Αν και δεν είναι επισημασμένες, λόγω του μικρού αριθμού τους δεν είναι ιδιαίτερα δύσκολο να τις ταξινομήσουμε με το χέρι. Το μικρό μέγεθος αυτής της συλλογής, δεν την καθιστά κατάλληλη για την έρευνά μας.

4.1.3 Oxford VGG Car Datasets

Εικόνα 43: Φωτογραφία από την συλλογή Oxford VGG Car Datasets[7]

Το Visual Geomtry Group του Πανεπιστημίου της Οξφόρδης έχει αναδημοσιεύσει τα τελευταία χρόνια τις συλλογές CALTECH 2001 (Rear) και CALTECH 1999 (Rear) 2. Η

[6] http://www.vision.caltech.edu/html-files/archive.html

[7] http://www.robots.ox.ac.uk/~vgg/data3.html

έκδοση αυτή περιέχει 1155 φωτογραφίες αλλά στην χαμηλή ανάλυση των 360x240 πίξελ. Και αυτή η συλλογή χρησιμεύει μόνο για τον εντοπισμό της πινακίδας και όχι για την αναγνώριση των χαρακτήρων.

4.1.4 UCSD

Το εργαστήριο υπολογιστικής όρασης του Πανεπιστημίου της Καλιφόρνιας στο Σαν Ντιέγκο διαθέτει μια μεγάλη συλλογή[8] δεδομένων που αποτελείται από:

- 10 ώρες βίντεο με οχήματα που εισέρχονται και εγκαταλείπουν το κάμπους
- 878 καθαρές φωτογραφίες όπου είναι ορατές οι πινακίδες και σημασμένες

Το πρόβλημα είναι πως η συλλογή δεν είναι δημόσια διαθέσιμη και πρέπει κανείς να επικοινωνήσει με τους υπεύθυνους ώστε να πάρει άδεια να τις χρησιμοποιήσει.

4.1.5 Medialab

Το Εργαστήριο Τεχνολογίας Πολυμέσων του Εθνικού Μετσόβιου Πολυτεχνείου διαθέτει αρχείο με αρκετές συλλογές[9] (Anagnostopoulos, Anagnostopoulos, Psoroulas, *et al.*, 2008) πινακίδων κυκλοφορίας που μπορούμε να χρησιμοποιήσουμε για την έρευνά μας. Οι φωτογραφίες αυτές δεν είναι επισημασμένες αλλά αυτό μπορεί να γίνει με το χέρι. Από όλες τις συλλογές επιλέξαμε 139 φωτογραφίες που ήταν σε ικανοποιητική ποιότητα ώστε να προχωρήσουμε με την έρευνά μας.

Εικόνα 44: Φωτογραφία από την συλλογή Medialab

[8] http://vision.ucsd.edu/belongie-grp/research/carRec/car_data.html

[9] http://www.medialab.ntua.gr/research/LPRdatabase.html

4.2 *Δημιουργία δικής μας συλλογής*

Για την δημιουργία ενός καλού συστήματος ΑΑΠΚ δεν αρκούν μόνο οι αλγόριθμοι αναγνώρισης της πινακίδας και των χαρακτήρων αλλά πρέπει να συλλέξουμε τα δικά μας δεδομένα, σε περίπτωση που οι υπάρχουσες συλλογές δεδομένων δεν εξυπηρετούν το σκοπό μας. Η συλλογή των φωτογραφιών είναι μια χρονοβόρα διαδικασία αλλά όχι αδύνατη. Το μόνο που χρειάζεται είναι μια καλή κάμερα, όπως αυτή του κινητού μας τηλεφώνου, και η εύρεση πολλών σταθμευμένων οχημάτων.

Καταλαβαίνουμε λοιπόν πως τα δεδομένα αυτά φέρουν μεγάλης σημασίας και στις περισσότερες περιπτώσεις είναι πιο πολύτιμα από τον αλγόριθμο. Αυτός είναι και ο λόγος που δεν βρίσκονται δημόσια διαθέσιμα τέτοια δεδομένα.

Σε παρακάτω κεφάλαιο θα δούμε πως μπορούμε να εξάγουμε τους χαρακτήρες από τις φωτογραφίες μας και να τους επισημάνουμε ώστε να εκπαιδεύσουμε τον ταξινομητή μας.

5

Εντοπισμός της πινακίδας κυκλοφορίας

Στο προηγούμενο κεφάλαιο συζητήσαμε το πρώτο στάδιο των συστημάτων ΑΑΠΚ που είναι η Απόκτηση/λήψη της φωτογραφίας. Σε αυτήν την ενότητα θα μελετήσουμε πως μπορούμε με απλό τρόπο να εντοπίσουμε περιοχές που περιέχουν πινακίδες κυκλοφορίας σε φωτογραφίες. Όπως αναφέρθηκε παραπάνω, υπάρχουν δύο τρόποι για τον εντοπισμό των πινακίδων. Ο ένας είναι με τη χρήση μεθόδων μηχανικής μάθησης. Τα δεδομένα για την εκπαίδευση των μοντέλων θα πρέπει να αποτελούνται από τις πινακίδες κομμένες, ώστε να εκπαιδευτεί το δίκτυο και να μάθει να αναγνωρίζει πινακίδες. Η μέθοδος αυτή είναι πολύπλοκη, απαιτεί πολλά δεδομένα και είναι χρονοβόρα. Ευτυχώς, υπάρχει μια δεύτερη μέθοδος που μπορούμε να αξιοποιήσουμε, η οποία είναι είναι απλή τόσο στην υλοποίηση όσο και στην κατανόηση. Η μέθοδος αυτή περιλαμβάνει τη χρήση:

- μορφολογικών πράξεων (morphological operations) που θα μας βοηθήσουν να εντοπίσουμε τις περιοχές που πιθανώς να περιέχουν πινακίδες
- περιγραμμάτων (contours) ώστε να αποκόψουμε την περιοχή της πινακίδας από την υπόλοιπη εικόνα

5.1 *Μελέτη του προβλήματος*

Σκοπός μας σε αυτήν την ενότητα είναι να εντοπίσουμε τις περιοχές που πιθανώς περιέχουν πινακίδες και να ζωγραφίσουμε ένα πλαίσιο γύρω από αυτές. Πριν ξεκινήσουμε όμως να γράφουμε τον κώδικά μας, ας αφιερώσουμε λίγο χρόνο ώστε:

- να μελετήσουμε το πρόβλημά μας
- να κάνουμε διάφορες υποθέσεις/παρατηρήσεις που μπορεί να απλουστεύσουν το πρόβλημά μας

Εικόνα 45: Κολάζ φωτογραφιών από την συλλογή Medialab

Βλέποντας την εικόνα 45 παρατηρούμε ότι:

- Το πλάτος της πινακίδας είναι μεγαλύτερο από το ύψος. Γνωρίζοντας αυτήν την ιδιότητα της πινακίδας μπορούμε να ψάξουμε στην φωτογραφία για περιοχές με αυτήν την αναλογία
- Η περιοχή της πινακίδας έχει πάντα λευκό φόντο με μαύρους χαρακτήρες. Αυτό δημιουργεί μια αντίθεση ώστε να είναι ευανάγνωστες από τους ανθρώπους αλλά βοηθάει και στο πρόβλημά μας
- Η πινακίδα είναι ορθογώνια παραλληλόγραμμη. Αυτό θα μας φανεί χρήσιμο και ίσως με την χρήση μια μεθοδολογίας που ονομάζεται προσέγγιση περιγράμματος (contour approximation) καταφέρουμε να λύσουμε το πρόβλημα
- Η πινακίδα αποτελείται από συνολικά 7 χαρακτήρες, ξεκινώντας από 3 γράμματα και συνεχίζοντας με 4 αριθμούς

Ας σταθούμε λίγο και ας σκεφτούμε τις υποθέσεις που κάναμε. Τι θα γινόταν αν δεν κάναμε την πρώτη υπόθεση και αντί αυτού υποθέταμε πως η πινακίδα είναι πάντα πιο φωτεινή από το

όχημα; Αν και με την πρώτη ματιά φαίνεται να είναι σωστή, είναι λανθασμένη. Η πινακίδα είναι πάντα πιο φωτεινή από το όχημα, όταν αυτό έχει σκούρο χρώμα όπως μαύρο, σκούρο μπλε, κτλ. Αν όμως το όχημα έχει ανοικτό χρώμα όπως άσπρο ή ασημένιο τότε η υπόθεση που κάνουμε είναι λάθος. Σημαντικό ρόλο παίζει και ο φωτισμός στο περιβάλλον το οποίο δεν μπορούμε να ελέγξουμε. Δεν μπορούμε να είμαστε 100% σίγουροι πως η υποθέσεις θα είναι σωστές για αυτό πρέπει να μελετάμε το πρόβλημά μας από διαφορετικές οπτικές γωνίες. Καταλαβαίνουμε λοιπόν πως το στάδιο της μελέτης του προβλήματος είναι πολύ σημαντικό και πρέπει να αφιερώσουμε χρόνο σε αυτό, ώστε να εφαρμόσουμε όσο το δυνατόν *a priori* γνώση γίνεται, πριν ξεκινήσουμε να γράφουμε τον κώδικά μας.

5.2 *Δομή του συστήματος ΑΑΠΚ*

```
|--- alpr
|    |--- __init__.py
|    |--- descriptors
|    |    |---- __init__.py
|    |    |--- blockbinarypixelsum.py
|    |--- license_plate
|    |    |---- __init__.py
|    |    |--- licenseplate.py
|--- recognize.py
```

Σχήμα 24: Δομή του συστήματος ΑΑΠΚ

Στο σχήμα 24 βλέπουμε τη δομή του συστήματος ΑΑΠΚ την οποία θα χρησιμοποιήσουμε στα παρακάτω κεφάλαια. Το βασικό μας πακέτο είναι το *alpr* που περιέχει άλλα δύο πακέτα. Στο πακέτο *descriptors* θα υλοποιήσουμε τον περιγραφέα μας, συγκεκριμένα τον BlockBinaryPixelSum. Το πακέτο *license_plate* και συγκεκριμένα το αρχείο *license_plate.py* θα περιέχει όλον τον σχετικό κώδικα υπεύθυνο για την αναγνώριση της πινακίδας. Το αρχείο *recognize.py* έχει τον ρόλο του οδηγού μας ή καλύτερα του συνδετικού κρίκου, εισάγει δηλαδή όλα τα παραπάνω αρχεία και τα χρησιμοποιεί για τον εντοπισμό και την αναγνώριση πινακίδων στις φωτογραφίες.

5.3 *Εντοπισμός της πινακίδας*

Αφού μελετήσαμε το πρόβλημα και κάναμε τις υποθέσεις μας, είμαστε έτοιμοι να τις υλοποιήσουμε στο αρχείο *licenseplate.py*. Ξεκινάμε δηλώνοντας στο αρχείο την κλάση *LicensePlateDetector* και έναν δομητή με 3 παραμέτρους. Η πρώτη παράμετρος είναι η εικόνα που θέλουμε να εξετάσουμε και πιθανώς να περιέχει μια πινακίδα. Οι επόμενες δύο παράμετροι είναι το ελάχιστο πλάτος και ύψος που μπορεί να έχει η πινακίδα και δίνουμε προεπιλεγμένη τιμή 60 και 20 πίξελ αντίστοιχα. Οι τιμές αυτές είναι αρκετά μικρές ώστε να

μην μας ξεφύγει κάποια πινακίδα. Πως αποφασίσαμε όμως να επιλέξουμε αυτές τις τιμές; Οι τιμές αυτές στηρίζονται στην υπόθεση 1 που κάναμε προηγουμένως. Γνωρίζοντας πως το πλάτος είναι μεγαλύτερο από το ύψος, υποθέτουμε πως αυτές οι τιμές έχουν αναλογία 1 προς 3, με 1 να είναι ο ύψος και 3 το πλάτος. Επόμενο βήμα είναι να δημιουργήσουμε την μέθοδο *detect()* που προς το παρόν θα καλεί την μέθοδο *detectPlates()*. Μέσα στην τελευταία θα γράψουμε τον κώδικα υπεύθυνο για τον εντοπισμό της υποψήφιας πινακίδας.

Λαμβάνοντας υπόψη την υπόθεση 2 θα εφαρμόσουμε την μορφολογική πράξη BlackHat στην εικόνα μέσω της μεθόδου *morphologyEx()*, αφού πρώτα την μετατρέψουμε σε κλίμακα του γκρι (εικόνα 47), που θα μας βοηθήσει να αποκαλύψουμε στην εικόνα σκούρες περιοχές με ανοικτό φόντο. Η μέθοδος αυτή παίρνει ως πρώτο όρισμα μια εικόνα, ως δεύτερο όρισμα θα δώσουμε την τιμή *MORPH_BLACKHAT* και για τρίτο θα δώσουμε έναν ορθογώνιο πυρήνα μεγέθους 13x5 (Παρατηρούμε πως και εδώ η αναλογία είναι περίπου 1 προς 3).

Εικόνα 46: Φωτογραφία για εντοπισμό της πινακίδας, αρχική εικόνα

Εικόνα 47: Φωτογραφία για εντοπισμό της πινακίδας, σε κλίμακα του γκρί

Εφαρμόζοντας τα παραπάνω στην εικόνα 47 προκύπτει η εικόνα 48. Παρατηρούμε πως το μεγαλύτερο μέρος της εικόνας μετατράπηκε σε μαύρο, ενώ οι χαρακτήρες της πινακίδας γίνανε άσπροι. Αυτό επιβεβαιώνει πως η υπόθεση 2 που κάναμε ήταν σωστή.

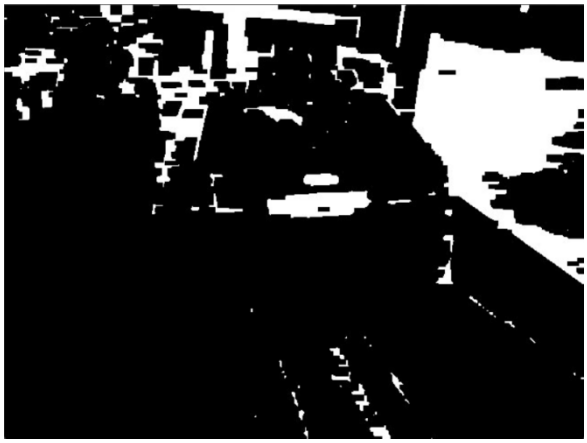
Στο επόμενο βήμα θα υπολογίσουμε το Sobel Gradient κατά τον άξονα X στην εικόνα 48. Με τον τρόπο αυτό γίνονται εμφανείς περιοχές που συμβαδίζουν με την υπόθεση 1, αλλά έχουν και κάθετες αλλαγές στην κλίση (gradient), όπως οι χαρακτήρες της πινακίδας. Το αποτέλεσμα φαίνεται στην εικόνα 49.

Εικόνα 48: Φωτογραφία για εντοπισμό της πινακίδας, μετά την εφαρμογή του BlackHat

Εικόνα 49: Φωτογραφία για εντοπισμό της πινακίδας, μετά την εφαρμογή του Sobel Gradient. Με τον τρόπο αυτό γίνανε πιο έντονες περιοχές με κάθετες αλλαγές στην κλίση, όπως οι χαρακτήρες της πινακίδας

Στη συνέχεια θα επεξεργαστούμε περαιτέρω την εικόνα 49. Εφαρμόζουμε την μέθοδο GaussianBlur με πυρήνα μεγέθους (5, 5) για να εξομαλύνουμε την εικόνα και να απαλλαγούμε από τον θόρυβο που δημιούργησε το Sobel Gradient. Συνεχίζουμε με μια

μορφολογική πράξη κλεισίματος με πυρήνα (13, 5) ώστε να κλείσουμε κενά στην κλίση (gradient). Η πράξη αυτή δεν είναι τίποτα άλλο από μια διαστολή (dilation) ακολουθούμενη από μια διάβρωση (Erosion). Το μέγεθος του πυρήνα δεν είναι τυχαίο καθώς είναι περίπου 3 φορές φαρδύτερος από το ύψος του. Αυτό μας επιτρέπει να κλείσουμε τα κενά ανάμεσα στους χαρακτήρες της πινακίδας. Έπειτα μετατρέπουμε την εικόνα σε ασπρόμαυρη χρησιμοποιώντας την μέθοδο Otsu για τον αυτόματο εντοπισμό του κατωφλιού. Στην εικόνα 50 φαίνεται το αποτέλεσμα των παραπάνω πράξεων, όπου αρχίζει και γίνεται πιο έντονη η παρουσία της πινακίδας.



Εικόνα 50: Φωτογραφία για εντοπισμό της πινακίδας, μετά την εφαρμογή ενός GaussianBlur, ακολουθούμενου από ενός κλεισίματος με πυρήνα (13, 5) και μετατροπή της εικόνας σε ασπρόμαυρη με τη μέθοδο Otsu

Παρόλα αυτά υπάρχει ακόμα αρκετός θόρυβος στην εικόνα. Μια σειρά από διαβρώσεις (erosions) και διαστολές (dilations) θα μας βοηθήσει να απαλλαγούμε από αυτόν θόρυβο, όπως φαίνεται στην εικόνα 51.

Εικόνα 51: Φωτογραφία για εντοπισμό της πινακίδας, μετά την εφαρμογή διαδοχικών διαβρώσεων (erosions) και διαστολών (dilations)

Έχουμε σχεδόν τελειώσει με την επεξεργασία της εικόνας και φτάνουμε στα τελευταία

βήματα του εντοπισμού της πινακίδας. Μένει ένα τελευταίο βήμα ώστε να καθαρίσουμε λίγο ακόμα την εικόνα μας. Αυτό θα το καταφέρουμε εφαρμόζοντας:

- τον τελεστή *KAI (bitwise AND)* στην εικόνα 51, με μάσκα την εικόνα 52
- μια σειρά από διαβρώσεις (erosions) και διαστολές (dilations)

Η εικόνα 52 προκύπτει από την εικόνα 47 μετά από πράξη κλεισίματος με τετραγωνικό πυρήνα (3, 3) και μετατροπή της σε ασπρόμαυρη, με κατώφλι 50.

Εικόνα 52: Φωτογραφία για εντοπισμό της πινακίδας, ασπρόμαυρη αναπαράσταση της εικόνας 14 με κατώφλι 50

Το τελικό αποτέλεσμα της επεξεργασίας φαίνεται στην εικόνα 53, όπου πλέον η πινακίδα γίνεται εμφανής και μπορούμε να την εντοπίσουμε, χρησιμοποιώντας βασικές ιδιότητες περιγραμμάτων (contour properties).

Εικόνα 53: Φωτογραφία για εντοπισμό της πινακίδας, τελική εικόνα πριν την αναγνώριση της πινακίδας

Για να εντοπίσουμε την πινακίδα, ξεκινάμε βρίσκοντας τα περιγράμματα για όλα τα σχήματα που μπορεί να εντοπίσει η μέθοδος *findContours()*. Για τα περιγράμματα αυτά υπολογίζουμε:

1. τον λόγο πλάτος προς ύψος

2. το περιστρεφόμενο πλαίσιο οριοθέτησης (Rotated Bounding Box) ώστε να λάβουμε την μικρότερη περιοχή του περιγράμματος

Αν i) ο λόγος (1) είναι μεγαλύτερος του 3 και μικρότερος του 6 και ii) το πλάτος και ύψος είναι μεγαλύτερα από τις προεπιλεγμένες τιμές, στην περίπτωσή μας 60 και 20 αντίστοιχα, τότε η περιοχή που ορίζει το περιστρεφόμενο πλαίσιο οριοθέτησης είναι υποψήφια πινακίδα κυκλοφορίας. Τέλος η μέθοδος *detectPlates()* επιστρέφει λίστα με όλες τις υποψήφιες πινακίδες. Το μόνο που μένει να κάνουμε είναι να δημιουργήσουμε ένα αντικείμενο της κλάσης *LicensePlateDetector* μέσα στο αρχείο *recognize.py* δίνοντας ως όρισμα μια εικόνα, να καλέσουμε την μέθοδο *detect()* και για κάθε περιοχή που θα μας επιστρέψει να ζωγραφίσουμε ένα πλαίσιο ώστε να το εμφανίσουμε. Παρακάτω βλέπουμε κάποια επιτυχή παραδείγματα εντοπισμού πινακίδων.

Εικόνα 54: Επιτυχές παράδειγμα εντοπισμού πινακίδας κυκλοφορίας

Εικόνα 55: Επιτυχές παράδειγμα εντοπισμού πινακίδας κυκλοφορίας

Υπάρχουν όμως και περιπτώσεις που εντοπίζονται περιοχές που δεν μας ενδιαφέρουν. Αυτό δεν πρέπει να μας αποθαρρύνει καθώς όπως έχουμε ήδη αναφέρει, οι περιοχές αυτές αφορούν υποψήφιες πινακίδες κυκλοφορίας.

Εικόνα 56: Παράδειγμα εντοπισμού πινακίδας κυκλοφορίας και θορύβου

5.4 *Κατακερματισμός των χαρακτήρων*

Στο προηγούμενο κεφάλαιο καταφέραμε με τη χρήση βασικών τεχνικών επεξεργασίας εικόνας, όπως μορφολογικές πράξεις και περιγράμματα, να εντοπίσουμε σε εικόνες περιοχές που μπορεί να περιέχουν πινακίδες κυκλοφορίας. Τις περιοχές αυτές τις ονομάσαμε Υποψήφιες Πινακίδες Κυκλοφορίας. Το επόμενο βήμα είναι να πάρουμε αυτές τις περιοχές και να εξάγουμε τους χαρακτήρες που βρίσκονται στο προσκήνιο, από το παρασκήνιο της πινακίδας.

Το στάδιο αυτό της επεξεργασίας, όσο και να φαίνεται περίεργο, κρύβει τις δικές του δυσκολίες. Θα υπέθετε κανείς πως εφαρμόζοντας ένα απλό thresholding θα αποκαλύπτονταν οι χαρακτήρες στην πινακίδα μας. Όντως, αυτό είναι αλήθεια, αλλά πέρα από τους χαρακτήρες θα εμφανίζονταν και άλλα πράγματα, θόρυβος, και θα προκαλούσαν πρόβλημα στον αλγόριθμό μας. Ο θόρυβος μπορεί να περιλαμβάνει τα εξής:

- οι βίδες που στερεώνουν την πινακίδα
- λογότυπα πάνω στην πινακίδα
- αυτοκόλλητο του ΚΤΕΟ

Στην πορεία θα δούμε πως μπορούμε να απαλλαγούμε από τον θόρυβο και να κρατήσουμε μόνο τους χαρακτήρες που μας ενδιαφέρουν. Συγκεκριμένα θα μελετήσουμε:

- την εξαγωγή της περιοχής της πινακίδας με προοπτικό μετασχηματισμό (perspective transformation)
- την εφαρμογή Ανάλυσης Συνδεδεμένων Συνιστωσών ή Συστατικών (Connected Component Analysis) για τον εντοπισμό τμημάτων που μοιάζουν με χαρακτήρες
- τη χρήση ιδιοτήτων περιγραμμάτων (contour properties) για την κατάτμηση των χαρακτήρων από την πινακίδα

5.4.1 Εντοπισμός των χαρακτήρων

Για τον εντοπισμό των χαρακτήρων θα χρειαστεί να τροποποιήσουμε την κλάση *LicensePlateDetector* του αρχείου *license_plate.py* . Η κλάση μας ως τώρα περιείχε την μέθοδο *detect()* και την μέθοδο *detectPlates()*. Η πρώτη ως τώρα απλά καλούσε την δεύτερη η οποία είναι αυτή που δέχεται μια εικόνα και επιστρέφει τις Υποψήφιες Πινακίδες. Θα προσθέσουμε λοιπόν μια νέα μέθοδο, την *detectCharacterCandidates()*, που θα δέχεται τις Υποψήφιες Πινακίδες της *detectPlates()* και εντοπίζει σε αυτές τους χαρακτήρες.

Πριν ξεκινήσουμε να υλοποιούμε την νέα μέθοδο, πρέπει να κάνουμε κάποιες προσθήκες στον κώδικά μας. Πρώτα θα ορίσουμε έξω από την κλάση μας μια νέα δομή, με όνομα *LicensePlate*, που θα περιέχει τις παρακάτω πληροφορίες σχετικά με τον εντοπισμό και κατακερματισμό της πινακίδας:

- **success**: Boolean τιμή που δείχνει αν ο εντοπισμός και κατακερματισμός ήταν επιτυχής
- **plate**: Εικόνα της πινακίδας που εντοπίστηκε
- **thresh**: Εικόνα της πινακίδας, αφού έχει εφαρμοστεί κάποιο κατώφλι, που αποκαλύπτει τους χαρακτήρες
- **candidates**: Μια λίστα με τους υποψήφιους χαρακτήρες. Αυτήν την λίστα θα χρησιμοποιήσουμε αργότερα για να αναγνωρίσουμε τους χαρακτήρες

Επίσης θα προσθέσουμε άλλα δύο ορίσματα στον δομητή μας, ένα που ορίζει το πλήθος των χαρακτήρων που θα περιέχει μια πινακίδα και ένα που ορίζει το ελάχιστο πλάτος, σε πίξελ, που μπορεί να έχει ένας χαρακτήρας. Στα ορίσματα αυτά δίνουμε αρχικές τιμές 7 και 40 αντίστοιχα.

Η τιμή 7 για το πλήθος των χαρακτήρων δεν είναι τυχαία. Στην παράγραφο "Μελέτη του προβλήματος" κάναμε κάποιες υποθέσεις σχετικά με τις πινακίδες μας. Όπως παρατηρούμε, οι υποθέσεις αυτές είναι πολύ χρήσιμες και μας βοηθάνε στην επίλυση προβλημάτων και στην αποφυγή εμποδίων καθώς προχωράμε στην υλοποίηση του κώδικα.

Ξεκινώντας με την υλοποίηση της νέας μεθόδου, όπως αναφέραμε προηγουμένως θα εφαρμόσουμε έναν προοπτικό μετασχηματισμό στην Υποψήφια Πινακίδα, το αποτέλεσμα της οποίας φαίνεται παρακάτω.

Εικόνα 57: Φωτογραφία πινακίδας μετά τον προοπτικό μετασχηματισμό (perspective transformation)

Όπως ήταν αναμενόμενο, δεν κόψαμε απλά την περιοχή της Υποψήφιας Πινακίδας αλλά εφαρμόζοντας τον προοπτικό μετασχηματισμό, η πινακίδα περιστράφηκε με τέτοιον τρόπο ώστε να φαίνεται σαν να την κοιτάμε από μπροστά (κάθετα) και όχι υπό κάποια άλλη γωνία. Αυτό θα μας βοηθήσει αργότερα στο στάδιο της αναγνώρισης.

Επόμενα βήμα είναι να εφαρμόσουμε κάποιο κατώφλι (threshold) στην εικόνα μας για να γίνουν πιο εμφανείς οι περιοχές των χαρακτήρων. Συνηθίζεται να μετατρέπουμε την εικόνα σε κλίμακα του γκρι πριν εφαρμόσουμε κάποιο κατώφλι. Όμως σε αυτό το στάδιο της επεξεργασίας θα μετατρέψουμε την εικόνα σε κλίμακα HSV και θα κρατήσουμε μόνο την τιμή V. Αυτός ο τρόπος δίνει καλύτερα αποτελέσματα όταν θέλουμε να εξάγουμε σκοτεινές περιοχές σε φωτεινό φόντο ή το αντίθετο. Εφαρμόζουμε λοιπόν ένα προσαρμοσμένο κατώφλι (adaptive threshold) στην εικόνα μας και όχι ένα απλό κατώφλι. Αλλάζουμε το φάρδος της εικόνας σε 400 πίξελ, διατηρώντας την αναλογία των διαστάσεων. Το αποτέλεσμα φαίνεται στην παρακάτω εικόνα.

Εικόνα 58: Φωτογραφία πινακίδας μετά την εφαρμογή προσαρμοσμένου κατωφλιού (adaptive thresholding)

Ο λόγος που εφαρμόσαμε το προσαρμοσμένο κατώφλι και όχι ένα απλό κατώφλι ή αυτόματο, φαίνεται καλύτερα στο παρακάτω παράδειγμα.

Εικόνα 59: Φωτογραφία πινακίδας μετά την εφαρμογή αυτόματου κατωφλιού (Otsu thresholding)

Εικόνα 60: Φωτογραφία πινακίδας μετά την εφαρμογή προσαρμοσμένου κατωφλιού (adaptive thresholding)

Παρατηρούμε πως στην εικόνα 59 με την εφαρμογή αυτόματου κατωφλιού η βίδα της πινακίδας είναι ενωμένη με το μηδενικό το οποίο θα δημιουργήσει πρόβλημα αργότερα στην

αναγνώριση των χαρακτήρων. Στην περίπτωση όμως του προσαρμοστικού κατωφλιού οι χαρακτήρες έχουν ξεχωριστεί καλύτερα το οποίο είναι και αυτό που επιθυμούμε.

Προχωράμε στο επόμενο βήμα της επεξεργασίας που είναι η εφαρμογή Ανάλυσης Συνδεδεμένων Συνιστωσών. Η εφαρμογή αυτής της μεθόδου επιστρέφει ετικέτες (labels) για τα συστατικά της εικόνας μας. Τα συστατικά αυτά αποτελούνται από το φόντο που έχει τιμή μηδέν (0) και από τους χαρακτήρες και τον θόρυβο που έχουν τιμή ένα (1). Δεν φτάνει παρά να επεξεργαστούμε τις ετικέτες ώστε να ξεχωρίσουμε τους χαρακτήρες της πινακίδας. Τα βήματα περιγράφονται στον παρακάτω αλγόριθμο:

Αλγόριθμος 5.1 Εντοπισμός Χαρακτήρων

1. Δημιουργούμε μια μάσκα στο μέγεθος της πινακίδας. Η μάσκα αυτή είναι απλά μια περιοχή γεμάτη μηδενικά, δηλαδή είναι μια μαύρη εικόνα, στην οποία θα προσθέσουμε όσους χαρακτήρες εντοπίσουμε
2. Για κάθε ετικέτα:
3. αν η ετικέτα είναι μηδέν την αγνοούμε γιατί είναι φόντο
4. αλλιώς για την συγκεκριμένη ετικέτα δημιουργούμε μια μάσκα με τιμές 1, αποκαλύπτοντας έτσι περιοχές που ανήκουν μόνο σε αυτήν την ετικέτα
5. αν υπάρχουν περιγράμματα για αυτήν την ετικέτα τότε για κάθε περίγραμμα
   5.1. βρίσκουμε το μεγαλύτερο περίγραμμα που αντιστοιχεί στην ετικέτα
   5.2. υπολογίζουμε το πλαίσιο οριοθέτησης (bounding box) για το περίγραμμα
   5.3. για το πλαίσιο ελέγχουμε τους λόγους i) ύψος/πλάτος, ii) εμβαδόν περιγράμματος/εμβαδόν πλαισίου και iii) ύψος πλαισίου/ύψος πινακίδας αν είναι εντός των επιθυμητών τιμών, αντίστοιχα:
      5.3.1. < 1.0
      5.3.2. > 0.15
      5.3.3. μεταξύ 0.4 και 0.95
   5.4. τότε η ετικέτα είναι χαρακτήρας
   5.5. υπολογίζουμε το κυρτό περίγραμμα (convex hull) του περιγράμματος και το ζωγραφίζουμε στην κενή μάσκα που δημιουργήσαμε

Τέλος καλούμε τη μέθοδο *clear_border()* της βιβλιοθήκης *skimage* η οποία εφαρμόζει Ανάλυση Συνδεδεμένων Συνιστωσών και αφαιρεί τυχόν πίξελ που βρίσκονται στο περίγραμμα της μάσκας. Αυτό το κάνουμε για να αποκλείσουμε περιοχές που μπορεί να είναι false-positive δηλαδή περιοχές που μπορεί να είναι μέρος του περιγράμματος της πινακίδας και όχι χαρακτήρες. Αυτό που επιστρέφει η συνάρτηση *detectCharacterCandidates()* είναι η δομή *LicensePlate().*

Θα πρέπει να διευκρινίσουμε σε αυτό το σημείο πως οι τιμές που ορίστηκαν στον αλγόριθμο ως επιθυμητές, αφορούν το συγκεκριμένο πρόβλημα αναγνώρισης χαρακτήρων σε πινακίδες και οριστικοποιήθηκαν μετά από αρκετές δοκιμές.

Μερικά παραδείγματα της επεξεργασίας που έγινε σε αυτό το κεφάλαιο φαίνονται παρακάτω.

Εικόνα 61: Πινακίδα όπου φαίνονται τα κυρτά περιγράμματα των χαρακτήρων

Στην παραπάνω εικόνα φαίνονται τα αποτελέσματα τεσσάρων πινακίδων. Για κάθε πινακίδα βλέπουμε με τη σειρά:

- την πινακίδα μετά τον προοπτικό μετασχηματισμό
- την πινακίδα μετά το προσαρμοσμένο κατώφλι
- τα κυρτά περιγράμματα των χαρακτήρων

Τα κυρτά περιγράμματα θα μας βοηθήσουν να κόψουμε τους χαρακτήρες τις πινακίδας ώστε σε επόμενα κεφάλαια να ξεκινήσουμε με την αναγνώρισή τους.

5.5 *Αποκοπή των χαρακτήρων*

Στην προηγούμενη ενότητα καταφέραμε να εντοπίσουμε τους χαρακτήρες στην πινακίδα και να βρούμε τα κυρτά περιγράμματά τους, αφαιρώντας τα υπόλοιπα στοιχεία που αποτελούσαν θόρυβο όπως βίδες, αυτοκόλλητα, πλαίσια και την παύλα.

Υπάρχουν όμως περιπτώσεις όπου μετά την επεξεργασία της εικόνας, συγκεκριμένα μετά την Ανάλυση Συνδεδεμένων Συνιστωσών, δεν έχουν αφαιρεθεί κάποιες περιοχές που δεν είναι χαρακτήρες. Οι περιοχές αυτές είναι συνήθως βίδες ή περιοχές έξω από την πινακίδα.

Εικόνα 62: Παραδείγματα πινακίδων όπου φαίνονται τα κυρτά περιγράμματα των χαρακτήρων μαζί με περιοχές που δεν αφαιρέθηκαν μετά την εφαρμογή Ανάλυσης Συνδεδεμένων Συνιστωσών

Στο κεφάλαιο αυτό θα προσπαθήσουμε να λύσουμε το παραπάνω πρόβλημα. Συγκεκριμένα θέλουμε να:

- υλοποιήσουμε ευρετική συνάρτηση ώστε να αφαιρέσουμε τις περιττές περιοχές από την πινακίδα, αφήνοντας μόνο τους χαρακτήρες της πινακίδας
- υλοποίηση μεθόδου για την αποκοπή των χαρακτήρων από την εικόνα, που ορίζονται από τα κυρτά περιγράμματα

5.5.1 Αφαίρεση περιττών περιοχών από την πινακίδα

Για την υλοποίηση αυτού του βήματος θα χρειαστεί να προσθέσουμε την μέθοδο *pruneCandidates()* στην κλάση *LicensePlateDetector*. Η μέθοδος αυτήν θα δέχεται ως ορίσματα:

1. ένα σύνολο υποψήφιων χαρακτήρων που έχουν εντοπιστεί στην *detectCharacterCandidates()*
2. τα περιγράμματά τους

και θα αφαιρεί αυτούς που δεν τηρούν κάποια κριτήρια. Αυτό θα γίνεται όταν το πλήθος των υποψήφιων χαρακτήρων είναι πάνω από ένα επιθυμητό όριο.

Πριν ξεκινήσουμε με την υλοποίηση της *pruneCandidates()* ας αφιερώσουμε λίγο χρόνο ώστε να μελετήσουμε το πρόβλημα. Στην παρακάτω εικόνα βλέπουμε από κοντά μια περίπτωση όπου ενώ έχουν εντοπιστεί σωστά όλοι οι χαρακτήρες, δεν έχει αφαιρεθεί όλος ο θόρυβος. Πρέπει σκεφτούμε λοιπόν έναν τρόπο ώστε να ξεφορτωθούμε τον θόρυβο. Βλέποντας λοιπόν την εικόνα 63 παρατηρούμε πως τους χαρακτήρες που μας ενδιαφέρουν τους διαπερνά περίπου στη μέση μια νοητή ευθεία, όπως φαίνεται στην εικόνα 64. Πρέπει λοιπόν να απαλλαγούμε από όσα περιγράμματα δεν βρίσκονται πάνω σε αυτήν την γραμμή.

Εικόνα 63: Παράδειγμα πινακίδας όπου φαίνονται τα κυρτά περιγράμματα των χαρακτήρων μαζί με περιοχές που δεν αφαιρέθηκαν μετά την εφαρμογή Ανάλυσης Συνδεδεμένων Συνιστωσών

Εικόνα 64: Νοητή γραμμή που διαχωρίζει τους χαρακτήρες της πινακίδας από τον θόρυβο

Ο αλγόριθμος για να το πετύχουμε αυτό είναι ο εξής:

Αλγόριθμος 5.2 Αφαίρεση περιττών περιοχών από την πινακίδα

1. Δημιουργία μάσκας στις διαστάσεις των υποψήφιων χαρακτήρων (*prunedCandidates*)
2. Αρχικοποίηση πίνακα διαστάσεων (*dims*)
3. Αρχικοποίηση πίνακα διαφορών (*diffs*)
4. Αρχικοποίηση πίνακα επιλεγμένων περιγραμμάτων (*selected*)
5. Για κάθε περίγραμμα:
    a. υπολογίζουμε το πλαίσιο οριοθέτησης (bounding box)
    b. προσθέτουμε στον πίνακα διαστάσεων το άθροισμα (ύψος πλαισίου + συντεταγμένη Y πλαισίου)
6. Για κάθε τιμή στον πίνακα διαστάσεων υπολογίζουμε το παρακάτω άθροισμα και το προσθέτουμε στον πίνακα διαφορών

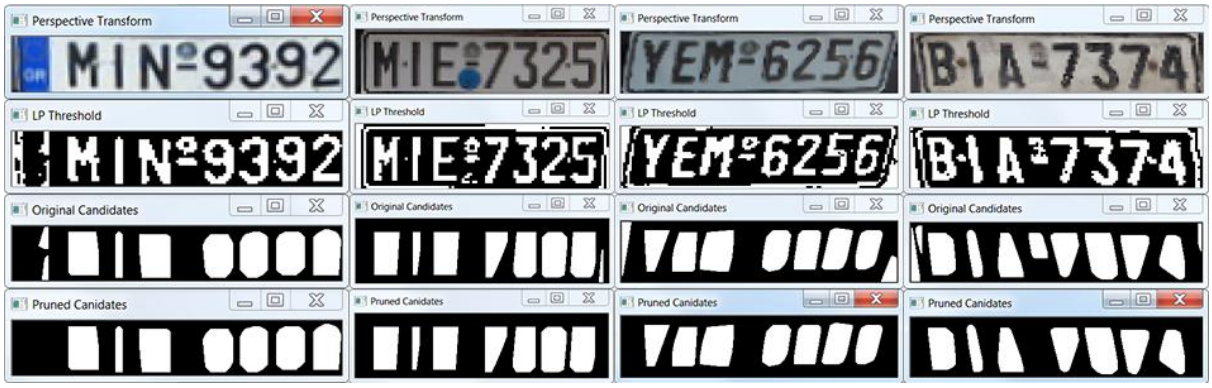
$$\sum_{i=0}^{dims} dims - dims[i]$$

7. Ταξινομούμε τον πίνακα διαφορών από το μικρότερο στο μεγαλύτερο
8. Κρατάμε το επιθυμητό πλήθος χαρακτήρων που αντιστοιχούν στους πρώτους πχ. 7 χαρακτήρες ή όσους ξέρουμε πως περιέχει η πινακίδα
9. Προσθέτουμε τους επιθυμητούς χαρακτήρες στην μάσκα *prunedCandidates* και τα περιγράμματα αυτών στον πίνακα *selected*

Με το άθροισμα **5b** υπολογίζουμε το χαμηλότερο σημείο των πλαισίων. Έτσι μπορούμε να εντοπίσουμε περιοχές που δεν είναι χαρακτήρες της πινακίδας όπως φαίνεται στην εικόνα 65

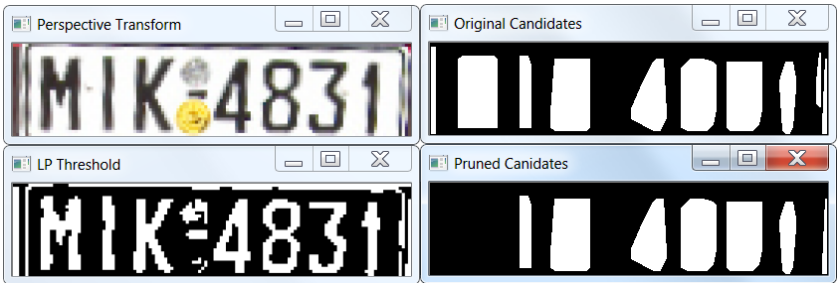
Εικόνα 65: Εντοπισμός θορύβου μετά την εφαρμογή της ευρετικής συνάρτησης pruneCandidates()

Με κόκκινο φαίνεται η περιοχή που είναι πολύ ψηλά και με μπλε η περιοχή που είναι πολύ χαμηλά σε σχέση με τις άλλες περιοχές αντίστοιχα. Μερικά επιτυχή αποτελέσματα της ευρετικής συνάρτησης φαίνονται στην εικόνα 66.



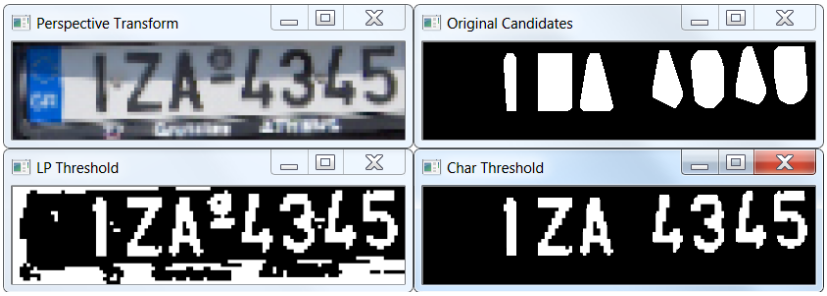
Εικόνα 66: Επιτυχή αποτελέσματα της ευρετικής συνάρτησης pruneCandidates()

Η συνάρτηση βέβαια δεν έχει 100% επιτυχία. Υπάρχουν περιπτώσεις όπου αφαιρούνται κατά λάθος περιοχές που είναι χαρακτήρες όπως φαίνεται στην εικόνα 67. Το σφάλμα οφείλεται στην απόκλιση της κάτω Υ συντεταγμένης του χαρακτήρα Μ με των υπόλοιπων. Η αποτυχία της συνάρτησης δεν πρέπει να μας ανησυχεί και δεν σημαίνει πως ο κόπος μας πήγε χαμένος καθώς είναι εξαιρετικά δύσκολο να πετύχουμε 100% επιτυχία σε προβλήματα επεξεργασίας εικόνας και υπολογιστικής όρασης.



Εικόνα 67: Ανεπιτυχή αποτέλεσμα της ευρετικής συνάρτησης pruneCandidates()

Με την υλοποίηση της ευρετικής συνάρτησης προχωράμε στην ενημέρωση της μεθόδου *detectCharacterCandidates()*. Η *pruneCandidates()* θα καλείται μόνο σε περίπτωση που έχουν εντοπιστεί παραπάνω υποψήφιοι χαρακτήρες από τους επιθυμητούς. Επίσης εφαρμόζουμε τον τελεστή *bitwise_and* στην εικόνα με μάσκα τους υποψήφιους χαρακτήρες. Το αποτέλεσμα φαίνεται στην εικόνα 68.



Εικόνα 68: Αφαίρεση του θορύβου και εμφάνιση μόνο των χαρακτήρων

5.6 *Εξαγωγή χαρακτήρων*

Για την εξαγωγή των χαρακτήρων θα ορίσουμε εντός της κλάσης *LicensePlateDetector* την μέθοδο *scissor()* η οποία δέχεται ως όρισμα την δομή LicensePlate που επιστρέφει η μέθοδος *detectCharacterCandidates()* και επιστρέφει έναν πίνακα με τις περιοχές των χαρακτήρων.

Τα βήματα που ακολουθούμε για την εξαγωγή των χαρακτήρων είναι:

Αλγόριθμος 5.3 Εξαγωγή χαρακτήρων

1. Ορίζουμε τους πίνακες *boxes* και *chars* όπου θα αποθηκεύσουμε τις συντεταγμένες των πλαισίων οριοθέτησης και τις περιοχές των χαρακτήρων αντίστοιχα
2. Βρίσκουμε τα περιγράμματα για κάθε υποψήφιο χαρακτήρα
3. Για κάθε περίγραμμα:
    a. υπολογίζουμε το πλαίσιο οριοθέτησης διατηρώντας το ελάχιστο πλάτος σύμφωνα με τον τύπο όπου

$$dX = min(self.minCharW, self.minCharW - boxW)/2$$
$$boxX = boxX - dX$$
$$boxW = boxW + (dX * 2)$$

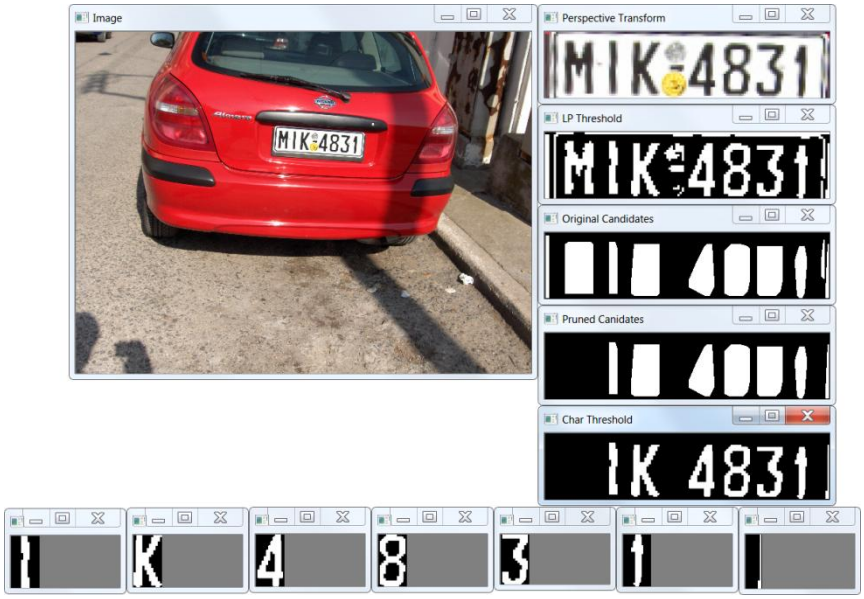
        i. minCharW είναι το ελάχιστο πλάτος που έχουμε ορίσει στον δομητή
        ii. boxW είναι το πλάτος του κάθε πλαισίου οριοθέτησης
    b. Προσθέτουμε στον πίνακα *boxes* τις συντεταγμένες τους πλαισίου
4. Ταξινομούμε τα στοιχεία του πίνακα *boxes* από αριστερά προς τα δεξιά
5. Για κάθε σύνολο συντεταγμένων του πίνακα *boxes*
    a. κόβουμε τον χαρακτήρα που ορίζεται από τις συντεταγμένες και
    b. προσθέτουμε τον χαρακτήρα τον πίνακα *chars*

6. Επιστρέφουμε τον πίνακα *chars*

Για να ολοκληρώσουμε την διαδικασία της εξαγωγής των χαρακτήρων, ενημερώνουμε την μέθοδο *detect()* ώστε να καλεί την *scissor()* αν η *detectCharacterCandidates()* εντοπίσει υποψήφιους χαρακτήρες. Επίσης ενημερώνουμε το αρχείο *recognize.py* ώστε να εμφανίζει όσους χαρακτήρες εντοπίστηκαν. Παρακάτω βλέπουμε δύο επιτυχή παραδείγματα. Στην εικόνα 69 έχουμε περίπτωση όπου εξαρχής έχουν εντοπιστεί σωστά οι 7 χαρακτήρες της πινακίδας. Στην εικόνα 70 βλέπουμε πως έχουν εντοπιστεί παραπάνω υποψήφιοι χαρακτήρες οι οποίοι απομακρύνθηκαν μετά την κλίση της *pruneCandidates().*

Εικόνα 69: Επιτυχής εξαγωγή χαρακτήρων

Εικόνα 70: Επιτυχής εξαγωγή χαρακτήρων μετά την αφαίρεση θορύβου και την εφαρμογή της ευρετικής συνάρτησης



Εικόνα 71: Ανεπιτυχής εξαγωγή όλων των χαρακτήρων μετά την εφαρμογή της ευρετικής συνάρτησης

Πριν προχωρήσουμε στο επόμενο κεφάλαιο, θα ήθελα να δώσουμε ιδιαίτερη προσοχή στο βήμα 3a και συγκεκριμένα στο σημείο όπου διατηρούμε το ελάχιστο πλάτος. Αν και αυτό θα γίνει καλύτερα κατανοητό στο επόμενο κεφάλαιο, να αναφέρουμε μόνο πως έχει να κάνει με τη διαδικασία εξαγωγής χαρακτηριστικών. Για παράδειγμα ας δούμε τους χαρακτήρες “1” και “Ο”. Ο αριθμός “1” δεν είναι τόσο πλατύς όσο ο “Ο”. Αν υπολογίσουμε ως πλαίσιο οριοθέτησης μόνο τον χώρο που δεσμεύει ο “1” υπάρχει περίπτωση το μέγεθός του να είναι κάτω από τα επιθυμητά όρια. Έτσι αν δεν διατηρήσουμε το ελάχιστο πλάτος, δεν θα μπορέσουμε να εξάγουμε σωστά χαρακτηριστικά από τον συγκεκριμένο χαρακτήρα και έτσι όταν φτάσουμε στο σημείο να εκπαιδεύσουμε τον ταξινομητή μας, τα αποτελέσματα που θα παράξει θα είναι άχρηστα.

6

Αναγνώριση χαρακτήρων

Ήμαστε πολύ κοντά πλέον στον τελικό μας στόχο, την αναγνώριση των χαρακτήρων. Στα προηγούμενα κεφάλαια καταφέραμε με τη χρήση απλών μεθόδων επεξεργασίας εικόνας να εντοπίσουμε υποψήφιες πινακίδες κυκλοφορίας σε εικόνες και από αυτές να εξάγουμε τις περιοχές των υποψήφιων χαρακτήρων. Το μοναδικό εμπόδιο που έχουμε να προσπεράσουμε είναι η συλλογή δεδομένων για την εκπαίδευση του δικτύου.

Όπως αναφέραμε σε προηγούμενο κεφάλαιο, η εύρεση κατάλληλων δεδομένων για ένα συγκεκριμένο πρόβλημα είναι δύσκολη υπόθεση. Συνήθως πρέπει κανείς να συλλέξει τα δικά του δεδομένα. Ευτυχώς για εμάς καταφέραμε και εντοπίσαμε την συλλογή Medialab του Εργαστηρίου Τεχνολογίας Πολυμέσων του Εθνικού Μετσόβιου Πολυτεχνείου από την οποία επιλέξαμε 139 εικόνες.

Αυτό που μένει είναι λοιπόν είναι να δημιουργήσουμε κατάλληλα δεδομένα από αυτές τις εικόνες ώστε να εκπαιδεύσουμε τον ταξινομητή μας. Αυτό μπορεί να γίνει με τη χρήση της έως τώρα μελέτης μας, όπως θα δούμε παρακάτω.

6.1 *Επεξεργασία δεδομένων*

Για να μπορέσουμε να προχωρήσουμε στην εκπαίδευση του ταξινομητή μας, πρέπει πρώτα να δημιουργήσουμε κατάλληλα δεδομένα. Τα δεδομένα που χρειαζόμαστε για την εκπαίδευση αποτελούνται από εικόνες, αντιπροσωπευτικές για κάθε έναν χαρακτήρα που

υπάρχει στις ελληνικές πινακίδες κυκλοφορίας. Το πλήθος των δεδομένων παίζει σημαντικό ρόλο. Όσο περισσότερα τα δεδομένα, τόσο καλύτερα μπορεί να εκπαιδευτεί ο ταξινομητής.

Θα μπορούσαμε να δημιουργήσουμε μόνοι μας αυτά τα δεδομένα, παίρνοντας διάφορες γραμματοσειρές (εικόνα 72) και εκπαιδεύοντας το δίκτυο από τις εικόνες μας, έπειτα να τις επεξεργαστούμε, συγκεκριμένα να εξάγουμε κάποια χαρακτηριστικά από αυτές.

Εικόνα 72: Δείγμα χαρακτήρων που θα μπορούσε να χρησιμοποιηθεί για την εκπαίδευση του ταξινομητή

Η παραπάνω προσέγγιση είναι η εύκολη λύση αλλά δεν είναι αποτελεσματική για τους εξής λόγους:

1. Είναι δύσκολο να βρούμε γραμματοσειρές που να μοιάζουν στους χαρακτήρες των πινακίδων.
2. Πιο δύσκολο είναι να βρούμε μεγάλο πλήθος τέτοιων γραμματοσειρών
3. Οι χαρακτήρες των γραμματοσειρών δεν είναι αντιπροσωπευτικοί αυτών των πινακίδων.

Το σημαντικότερο σημείο από τα παραπάνω είναι το (3). Αν και με μια πρώτη ματιά δείχνουν αρκετά όμοιοι οι χαρακτήρες, μην ξεχνάμε πως οι εικόνες των πινακίδων είναι τραβηγμένες από κάποια απόσταση, με διαφορετικές συνθήκες φωτισμού και μάλλον υπό γωνία. Αν κοιτάξουμε από κοντά δύο χαρακτήρες, έναν από μια γραμματοσειρά και έναν από μια πινακίδα, βλέπουμε ξεκάθαρα τις διαφορές μεταξύ τους (εικόνα 73).

Εικόνα 73: Διαφορά μεταξύ χαρακτήρα γραμματοσειράς (αριστερά) και χαρακτήρα που κόψαμε από πινακίδα (δεξιά)

Ο χαρακτήρας στα αριστερά είναι πιο καθαρός από αυτόν στα δεξιά, έτσι αν η εκπαίδευση του ταξινομητή μας γινόταν με αυτά τα δεδομένα τα αποτελέσματα που θα παίρναμε δεν θα ήταν ικανοποιητικά, λόγω των (1), (2), (3). Για τους παραπάνω λόγους θα δημιουργήσουμε τα δεδομένα από τις εικόνες των πινακίδων.

6.1.1 Δημιουργία και Επισήμανση Δεδομένων

Το στάδιο της δημιουργίας των δεδομένων μας, συμπεριλαμβάνει πέρα από την εξαγωγή τους από τις εικόνες και την επισήμανση ή κατηγοριοποίηση τους. Αυτό θα το πετύχουμε εύκολα χρησιμοποιώντας τον αλγόριθμο που περιγράψαμε στα προηγούμενα κεφάλαια. Παρακάτω περιγράφεται η διαδικασία την οποία θα ακολουθήσουμε και θα υλοποιηθεί στο αρχείο *gather_examples.py*.

Για κάθε εικόνα στο σύνολο δεδομένων μας:

Αλγόριθμος 5.4 Δημιουργία και Επισήμανση Δεδομένων

1. Διαβάζουμε την εικόνα
2. Αν το πλάτος της είναι μεγαλύτερο από 640 πίξελ το προσαρμόζουμε αναλογικά στα 640 πίξελ
3. Δημιουργούμε ένα αντικείμενο της κλάσης *LicensePlateDetector* με παραμέτρους i) την εικόνα ii) τον αριθμό 7, αφού τόσοι είναι οι χαρακτήρες της πινακίδας και καλούμε την μέθοδο *detect()* που επιστρέφει τις υποψήφιες πινακίδες και τους υποψήφιους χαρακτήρες
4. Για κάθε έναν χαρακτήρα
    a. τον εμφανίζουμε και ζητάμε από τον χρήστη να πληκτρολογήσει ποιος είναι ο χαρακτήρας που βλέπει
    b. Αν πατήσει το πλήκτρο (`) τότε τον προσπερνάμε και πάμε στον επόμενο χαρακτήρα
    c. Αλλιώς δημιουργούμε έναν νέο φάκελο, αν δεν υπάρχει, με όνομα αυτό που έχει πληκτρολογήσει ο χρήστης, και αποθηκεύουμε τον χαρακτήρα ως εικόνα στον φάκελο με όνομα 00001.png
    d. Αυξάνουμε τον δείκτη για τον συγκεκριμένο χαρακτήρα
    e. Πάμε στο βήμα 4a

Εκτελώντας το παραπάνω για τις 139 εικόνες του Medialab, καταφέραμε να εξάγουμε συνολικά 713 χαρακτήρες από όλους τους χαρακτήρες που μπορούμε να βρούμε σε ελληνικές πινακίδες. Στην καλύτερη περίπτωση θα είχαμε εξάγει 139 x 7 = 973 χαρακτήρες. Η απόκλιση αυτή οφείλεται στο γεγονός πως δεν έχουν αναγνωριστεί σωστά όλες πινακίδες ή όλοι οι χαρακτήρες των πινακίδων. Αυτό μπορεί να οφείλεται στον αλγόριθμο ή στις εικόνες.

Σίγουρα η επεξεργασία που έγινε μπορεί να βελτιωθεί αλλά για να λειτουργήσει σωστά πρέπει και οι εικόνες που έχουμε να είναι σε μεγάλη ανάλυση, τραβηγμένες από καλή γωνία και σε κατάλληλο φωτισμό.

Οι χαρακτήρες που καταφέραμε να εξάγουμε με την παραπάνω επεξεργασία δεν είναι αρκετοί για να προχωρήσουμε στην εκπαίδευση. Αν και στο σύνολο τα 713 δεδομένα δεν είναι λίγα, για μερικούς χαρακτήρες δεν υπάρχουν πολλά δείγματα όπως για τους O, X, T και E (πίνακας 1).

0: 39	**1:** 36	**2:** 44	**3:** 47	**4:** 40	**5:** 46	**6:** 43	**7:** 39
8: 39	**9:** 34	**A:** 13	**B:** 20	**E:** 15	**H:** 12	**I:** 55	**K:** 22
M: 55	**N:** 20	**O:** 3	**P:** 12	**T:** 11	**X:** 7	**Y:** 24	**Z:** 37

Πίνακας 1: 713 χαρακτήρες που εξαγάγαμε από τις 139 εικόνες του Medialab

Για τον λόγο αυτό έγινε μια προσπάθεια να δημιουργηθεί νέα συλλογή δεδομένων. Η νέα συλλογή περιέχει συνολικά 1118 φωτογραφίες οχημάτων με εμφανείς τις πινακίδες κυκλοφορίας. Από τις 1118 φωτογραφίες εξάχθηκαν συνολικά 4340 χαρακτήρες (πίνακας 2). Το τελικό σύνολο δεδομένων που θα χρησιμοποιηθεί για την εκπαίδευση του ταξινομητή θα είναι ο συνδυασμός αυτών των δύο συλλογών.

Όπως και στην προηγούμενη συλλογή έτσι και σε αυτήν, ο αλγόριθμος δεν κατάφερε να εντοπίσει όλες τις πινακίδες κυκλοφορίας και κατά επέκταση όλους τους χαρακτήρες. Όπως αναφέρθηκε και σε προηγούμενες ενότητες, αυτό δεν πρέπει να μας ανησυχεί καθώς σε τέτοιου είδους προβλήματα δεν γίνεται να πετυχαίνουμε επιτυχία 100%. Άλλωστε επιτεύχθηκε ο αρχικός σκοπός που ήταν η συλλογή περισσότερων χαρακτήρων.

0: 205	**1:** 311	**2:** 233	**3:** 241	**4:** 260	**5:** 249	**6:** 248	**7:** 243
8: 265	**9:** 245	**A:** 44	**B:** 58	**E:** 88	**H:** 204	**I:** 300	**K:** 186
M: 95	**N:** 397	**O:** 73	**P:** 100	**T:** 64	**X:** 77	**Y:** 55	**Z:** 99

Πίνακας 2: 4340 χαρακτήρες που εξαγάγαμε από τις 1118 εικόνες που συλλέχθηκαν

6.1.2 Εξαγωγή χαρακτηριστικών (FeatureExtraction)

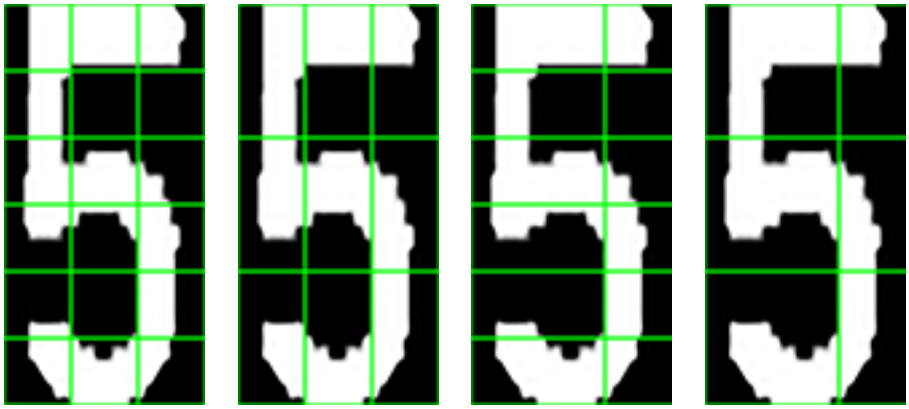
Το τελευταίο στάδιο της επεξεργασίας των δεδομένων, πριν ξεκινήσει η εκπαίδευση, είναι αυτό της εξαγωγής χαρακτηριστικών (*feature extraction*). Τα σύνολα δεδομένων αποτελούνται συνήθως από χαρακτηριστικά. Για παράδειγμα στο σύνολο δεδομένων *Iris* (Anderson, 1936; Fisher, 1936) κάθε εγγραφή αποτελείτε από 4+1 κύρια χαρακτηριστικά:

- το μήκος του σέπαλου σε εκατοστά
- το πλάτος του σέπαλου σε εκατοστά
- το μήκος του πετάλου σε cm
- το πλάτος του πετάλου σε cm
- μια από τις τρεις κλάσεις στις οποίες ανήκει
    - *Iris Setosa*
    - *Iris Versicolour*
    - *Iris Virginica*

Γίνεται κατανοητό πως για να περιγραφεί κάποιο από τα παραπάνω φυτά έπρεπε να εξαχθούν κάποια χαρακτηριστικά που να είναι αντιπροσωπευτικά του συγκεκριμένου είδους.

Υπάρχουν όμως περιπτώσεις όπου τα δεδομένα δεν αποτελούνται από κάποια χαρακτηριστικά τα οποία έχουν εξαχθεί, όπως αυτή των χειρόγραφων αριθμών του συνόλου δεδομένων MNIST (Lecun, Bottou, Bengio, *et al.*, 1998). Κάθε εικόνα στην συλλογή έχει μέγεθος 28x28 πίξελ, αποτελείται δηλαδή συνολικά από 784 πίξελ. Έτσι κάθε εγγραφή στη συλλογή αποτελείται από 784 χαρακτηριστικά, όπου κάθε χαρακτηριστικό είναι και ένα πίξελ της εικόνας με τιμές από 0 έως 255. Μια προσέγγιση θα ήταν να γίνει κανονικοποίηση των πίξελ πριν την εκπαίδευση, διαιρώντας τα όλα με το 255 έτσι ώστε το εύρος τιμών τους να γίνει από 0 έως 1. Αν και συναντάμε αρκετά αυτόν τον τρόπο επεξεργασίας, παραμένει το πρόβλημα των πολλών διαστάσεων (χαρακτηριστικών) που έχει ως αποτέλεσμα την πολλή αργή εκπαίδευση.

Ένας άλλος τρόπος εξαγωγής χαρακτηριστικών είναι με την μέθοδο Block Binary Pixel Sum (BBPS). Με την μέθοδο του BBPS χωρίζεται η εικόνα του χαρακτήρα σε μη επικαλυπτόμενες περιοχές μεγέθους MxN, συγκεκριμένα σε περιοχές διαστάσεων 5x5, 5x10, 10x5 και 10x10 (Chang, Hsu, Lee, *et al.*, 2013). Για μέγεθος εικόνας 30x15 πίξελ δημιουργούνται συνολικά 45 περιοχές (εικόνα 74).



Εικόνα 74: Εφαρμογή αλγορίθμου Block binary Pixel Sum (BBPS) σε χαρακτήρα πινακίδας μεγέθους 30x15 και περιοχές 5x5, 5x10, 10x5 και 10x10 πίξελ

Για κάθε περιοχή υπολογίζουμε τον λόγο της σχέσης (6.1). Τα 45 αποτελέσματα που προκύπτουν έχουν εύρος τιμών από 0 έως 1 και αποτελούν το τελικό διάνυσμα χαρακτηριστικών του κάθε χαρακτήρα.

$$\frac{\textit{Λευκά πίξελ στην περιοχή /}}{\textit{συνολικό αριθμό πίξελ στην περιοχή}} \qquad (6.1)$$

6.2 *Εκπαίδευση*

6.2.1 Διαχωρισμός Αριθμών και Χαρακτήρων

Παρατηρώντας το σύνολο δεδομένων βλέπουμε πως κάποια αριθμοί μοιάζουν αρκετά με κάποιους χαρακτήρες (εικόνα 75). Η ομοιότητα οφείλεται κυρίως στις συνθήκες κάτω από τις οποίες τραβήχτηκαν οι φωτογραφίες. Για τον λόγο αυτό θα χωρίσουμε τα δεδομένα μας σε δύο σύνολα, ένα με τους αριθμούς και ένα με τους χαρακτήρες. Ο διαχωρισμός έχει ως σκοπό την εκπαίδευση δύο ξεχωριστών ταξινομητών ώστε να αποφευχθεί η εκπαίδευση όμοιων δεδομένων.

Εικόνα 75: Ομοιότητα αριθμών και χαρακτήρων

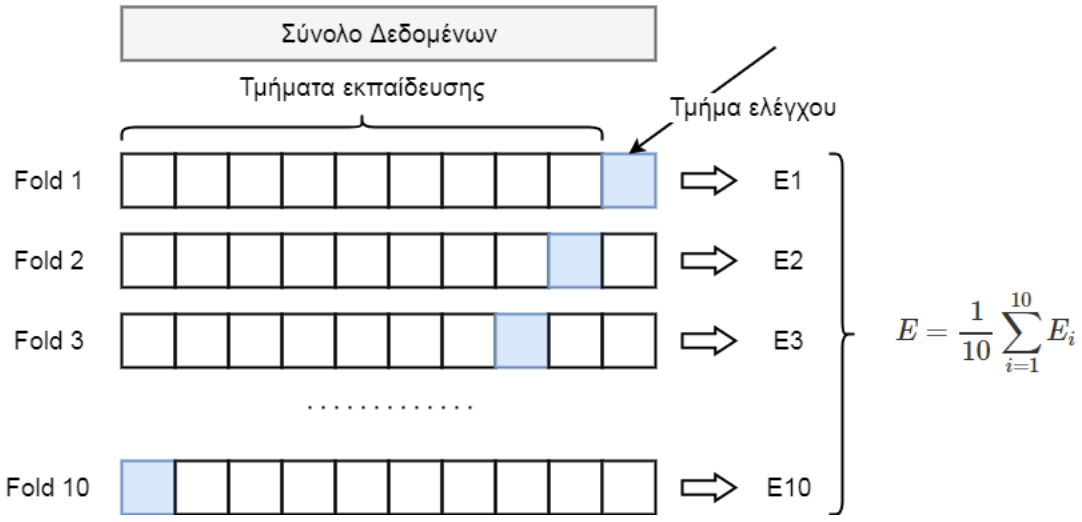
6.2.2 Διαχωρισμός δεδομένων

Κατά το στάδιο της εκπαίδευσης συνηθίζεται να διαχωρίζονται τα δεδομένα σε τμήμα εκπαίδευσης (*training set*) και ελέγχου (*testing set*). Το τμήμα ελέγχου είναι πάντα μικρότερο από το τμήμα εκπαίδευσης, πχ. το 1/3 των δεδομένων, και χρησιμοποιείται για τον έλεγχο του εκπαιδευμένου μοντέλου (Kohavi, 1995).

Ένας άλλος τρόπος, είναι ο διαχωρισμός των δεδομένων σε τμήμα εκπαίδευσης (*training*), επαλήθευσης (*validation*) και ελέγχου (*testing*). Η μέθοδος αυτή συνηθίζεται όταν θέλουμε να επιλέξουμε ανάμεσα σε διαφορετικά μοντέλα (*model selection*). Ο όρος μοντέλο αφορά τον συνδυασμό της μεθόδου μαζί με τις υπερπαραμέτρους.

6.2.2.1 Μέθοδος διασταύρωσης - K-Fold Cross-Validation

Η μέθοδος της διασταύρωσης (*Cross Validation*) είναι η πιο συνηθισμένη για τον διαχωρισμό των δεδομένων σε τμήματα εκπαίδευσης και ελέγχου. Αρχικά τα δεδομένα χωρίζονται τυχαία σε K μη επικαλυπτόμενα τμήματα ίσου μεγέθους. Το K είναι ένα θετικός ακέραιος αριθμός που συνήθως παίρνει τιμή 3, 5, 10 ή και μεγαλύτερη. Στη συνέχεια επιλέγεται ένα τμήμα ως σύνολο ελέγχου και τα υπόλοιπα K-1 τμήματα αποτελούν το σύνολο εκπαίδευσης. Το μοντέλο εκπαιδεύεται στα K-1 τμήματα και ελέγχεται η επίδοσή του στο σύνολο K. Το πείραμα αυτό ονομάζεται Fold και συγκεκριμένα Fold 1. Η ίδια διαδικασία επαναλαμβάνεται K φορές για όλα τα τμήματα. Η τελική αξιολόγηση του μοντέλου είναι ο μέσος όρος των αποτελεσμάτων κάθε Fold (Σχήμα 25).



Σχήμα 25: Μέθοδος διασταύρωσης (Cross-Validation)

6.2.2.2 Μέθοδος διασταύρωσης - Stratified K-Fold Cross-Validation

Στην παραπάνω μέθοδο, ο διαχωρισμός των Κ τμημάτων γίνεται με τυχαίο τρόπο. Αυτό όμως δεν εξασφαλίζει πως σε κάθε τμήμα εκπαίδευσης και ελέγχου ενός Fold θα περιέχεται τουλάχιστον ένα δεδομένο από κάθε κλάση. Υπάρχει περίπτωση δηλαδή κάποιο σύνολο ελέγχου να περιέχει δεδομένα στα οποία δεν έχει εκπαιδευτεί το μοντέλο. Για τον λόγο αυτό θα προτιμηθεί μια παραλλαγή της μεθόδου που ονομάζεται Stratified K-Fold Cross Validation. Η παραλλαγή αυτή διατηρεί την αναλογία των δεδομένων για κάθε κλάση σε όλα τα τμήματα. Έτσι εξασφαλίζεται πως σε κάθε τμήμα θα υπάρχει ένα ποσοστό από όλα τα δεδομένα, ίσο με αυτό όλης της συλλογής (Kohavi, 1995).

6.2.3 Αναζήτηση βέλτιστων Υπερ-παραμέτρων

Οι μέθοδοι μηχανικής μάθησης, εκτός από τις παραμέτρους που πρέπει να υπολογίσουν κατά τη διαδικασία της εκπαίδευσης, όπως είναι τα βάρη *w* στα νευρωνικά δίκτυα, διαθέτουν και άλλες παραμέτρους που ονομάζονται υπερπαράμετροι (*hyperparameters*). Σε αντίθεση με τις παραμέτρους του μοντέλου, οι υπερπαράμετροι στις περισσότερες περιπτώσεις δεν βελτιστοποιούνται κατά τη διαδικασία της εκπαίδευσης αλλά μένουν σταθερές και πρέπει να οριστούν εξ' αρχής από τον χρήστη. Τέτοιες παράμετροι είναι ο αριθμός των κρυφών στρωμάτων και ο αριθμός των νευρώνων σε κάθε στρώμα στα μοντέλα νευρωνικών δικτύων, ο ρυθμός μάθησης στον αλγόριθμο κατάβασης δυναμικού, οι μεταβλητές *C* και *gamma* στους αλγορίθμους SVM και Logistic Regression κ.α.

6.2.3.1 Ευρετική αναζήτηση Υπερ-παραμέτρων

Η αναζήτηση των υπερπαραμέτρων μπορεί να γίνει με ευρετικό τρόπο, δηλαδή ορίζοντας ένα εύρος τιμών για κάθε μια υπερπαράμετρο και στη συνέχεια να εκπαιδευτούν διαφορετικά μοντέλα με αυτές τις τιμές. Με τον τρόπο αυτό μπορούμε να εξασφαλίσουμε τις βέλτιστες τιμές για τα μοντέλα. Το μειονέκτημα είναι πως πρέπει να γράψουμε οι ίδιοι τον κώδικα, συγκεκριμένα τις επαναλήψεις (*for loops*), που θα αναζητούν τις τιμές που έχουμε ορίσει για κάθε υπερπαράμετρο και αυτό καθιστά τον κώδικα δυσανάγνωστο και μπορεί να γίνει αιτία για λάθη.

6.2.3.2 Αναζήτηση πλέγματος (Grid Search)

Στο πρόβλημα της ευρετικής αναζήτησης έρχεται να δώσει λύση η αναζήτηση πλέγματος. Η μέθοδος παρέχει έναν εύκολο τρόπο στον προγραμματιστή να αναζητήσει υπερπαραμέτρους χωρίς να χρειαστεί να μπλέξει με περιττό κώδικα. Το μόνο που χρειάζεται να ορίσει είναι τα ονόματα των υπερπαραμέτρων και το εύρος τιμών που θέλει να γίνει η αναζήτηση. Επιπλέον, η μέθοδος ενσωματώνει λειτουργία για την εκτέλεση Cross Validation πράγμα που καθιστά πολύ δημοφιλή αυτήν την μέθοδο.

Η μέθοδος της διασταύρωσης επαναλαμβάνεται για κάθε συνδυασμό υπερ-παραμέτρων. Έτσι προκύπτει ένας μέσος όρος από τις K επαναλήψεις της διασταύρωσης. Τελικά επιλέγονται οι υπερ-παράμετροι του καλύτερου μοντέλου βάση του μέσου όρου των K επαναλήψεων.

Ένα μειονέκτημα που κρύβει αυτή η μέθοδος είναι πως χρησιμοποιεί τα ίδια δεδομένα τόσο για την αναζήτηση των υπερ-παραμέτρων όσο και για την αξιολόγηση των μοντέλων. Με τον τρόπο αυτό υπάρχει περίπτωση να υπερ-εκπαιδευτούν τα μοντέλα στα δεδομένα εκπαίδευσης με αποτέλεσμα να μην δίνουν τόσο καλά αποτελέσματα σε άγνωστα δεδομένα. Η επίδραση της υπερ-εκπαίδευσης εξαρτάται από τον μέγεθος του συνόλου δεδομένων και από την σταθερότητα του μοντέλου (Cawley & Talbot, 2010).

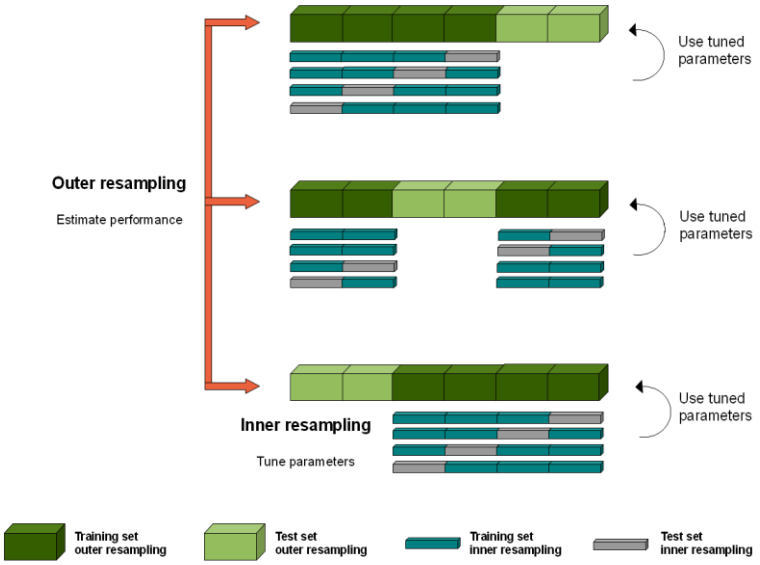
6.2.3.3 Αναζήτηση πλέγματος με Εμφωλευμένη διασταύρωση

Για την αποφυγή της υπερ-εκπαίδευσης κατά την αναζήτηση υπερ-παραμέτρων, προτείνεται η μέθοδος της αναζήτησης πλέγματος (*Grid Search*) με εμφωλευμένη διασταύρωση (*Nested Cross-Validation*) (Cawley & Talbot, 2010). Η μέθοδος αυτή προσθέτει έναν εξωτερικό βρόγχο στην διαδικασία της αναζήτησης υπερ-παραμέτρων και αξιολόγησης του μοντέλου. Στον εσωτερικό βρόγχο γίνεται η αναζήτηση και η αξιολόγηση με τη χρήση της διασταύρωσης και επιστρέφεται το καλύτερο μοντέλο. Στο εξωτερικό βρόγχο αξιολογείται το μοντέλο ως προς την γενίκευση με τη μέθοδο της διασταύρωσης. Έτσι το μοντέλο αξιολογείται σε διαφορετικά δεδομένα από αυτά που έγινε η αναζήτηση του βέλτιστου μοντέλου (Σχήμα 26). Αν η διαδικασία επαναληφθεί αρκετές φορές τότε για τον εξωτερικό βρόγχο προτείνεται ως μέθοδος διασταύρωσης η Stratified K-Fold Cross-Validation αλλά όχι για τον εσωτερικό (Krstajic, Buturovic, Leahy, *et al.*, 2014).

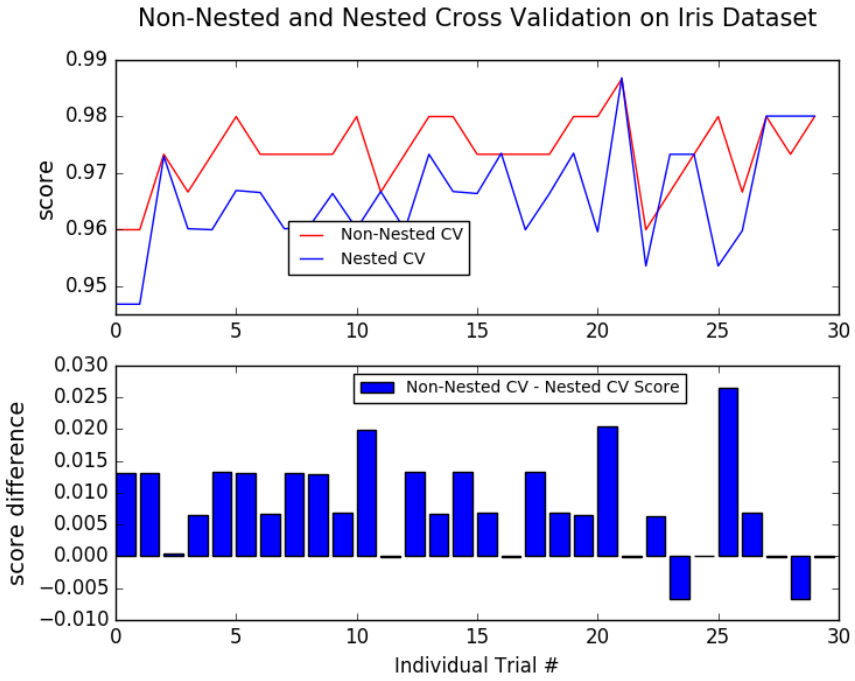
Αν συγκριθούν οι δύο μέθοδοι, της αναζήτησης πλέγματος χωρίς εμφωλευμένη διασταύρωση και της αναζήτησης πλέγματος με εμφωλευμένη διασταύρωση, παρατηρείται μια διαφορά στα αποτελέσματα, η οποία όμως δεν είναι σημαντική. Στο Σχήμα 27[10] υπάρχουν τα

[10] http://scikit-learn.org/stable/auto_examples/model_selection/plot_nested_cross_validation_iris.html

αποτελέσματα της σύγκρισης των δύο μεθόδων στο σύνολο δεδομένων Iris[11]. Τα πειράματα εκτελέστηκαν 30 φορές με K=4 για την διασταύρωση και την μέθοδο SVM ως ταξινομητή. Η διαφορά μεταξύ των αποτελεσμάτων είναι τόσο μικρή που δεν αξίζει η χρήση της εμφωλευμένης μέθοδος καθώς χρειάζεται πολύ περισσότερο χρόνο.



Σχήμα 26: Αναζήτησης πλέγματος (*Grid Search*) με εμφωλευμένη διασταύρωση (*Nested Cross-Validation*)



Σχήμα 27: Σύγκριση εμφωλευμένης και μη-εμφωλευμένης διασταύρωσης στο σύνολο δεδομένων Iris

[11] https://archive.ics.uci.edu/ml/datasets/iris

7

Αξιολόγηση

7.1 *Σύστημα αξιολόγησης*

Η αξιολόγηση των διαφόρων μεθόδων μηχανικής μάθησης έγινε με τη μέθοδο της διαστρωματωμένης διασταύρωσης (*Stratified Cross Validation*) ενώ για τη μέθοδο της Βαθειάς μάθηση χρησιμοποιήθηκε η απλή διασταύρωση (*Cross Validation*) με K=10 και στις δύο περιπτώσεις. Επιπλέον, δόθηκε στις μεθόδους η τιμή *shuffle=True* ώστε να ανακατεύεται το σύνολο δεδομένων σε κάθε Fold με σκοπό την καλύτερη τυχαιοποίηση.

Η βελτιστοποίηση των υπερ-παραμέτρων της εκάστοτε μεθόδου, έγινε με τη μέθοδο της αναζήτησης πλέγματος (*Grid Search*) οι τιμές των οποίων αναγράφονται στον πίνακα 3.

Καθώς πρόκειται για πρόβλημα ταξινόμησης, η μετρική που χρησιμοποιήθηκε για την αξιολόγηση των μεθόδων είναι η ακρίβεια (*accuracy*), δηλαδή το ποσοστό επιτυχημένων αναγνωρισμένων προτύπων προς το συνολικό αριθμό των προτύπων.

7.2 *Επιλογή υπερ-παραμέτρων*

Για την επιλογή των κατάλληλων υπερ-παραμέτρων χρησιμοποιήθηκε η μεθοδολογία coarse-to-fine grid search. Αρχικά η αναζήτηση έγινε σε μεγάλο εύρος τιμών με μεγάλο βήμα. Στη συνέχεια αφού εντοπίστηκαν οι περιοχές τιμών στις οποίες τα μοντέλα έδειχναν καλύτερα

αποτελέσματα, συγκεντρώθηκε η αναζήτηση γύρω από εκείνες τις περιοχές με μικρότερο βήμα έως ότου βρέθηκαν οι καταλληλότερες υπερ-παράμετροι.

Οι υπερ-παράμετροι που χρησιμοποιήθηκαν για κάθε μέθοδο καθώς και το τελικό εύρος τιμών φαίνεται στον πίνακα 3.

Μέθοδος Μηχανικής Μάθησης	**Εύρος τιμών υπερ-παραμέτρων**
SVM Linear	C=1 έως 15, τιμές=5
SVM RBF	C=1 έως 15, τιμές=5gamma=0 έως 0.5, τιμές=5
Linear SVM	C=1 έως 15, τιμές=5
Logistic Regression	C=1 έως 15, τιμές=5 solver={'liblinear', 'newton-cg'}
kNN	K=1 έως 20, τιμές 10 μετρικές { "minkowski", "euclidean", "manhattan"}
SGD	επαναλήψεις=1 έως 110, τιμές=10 loss={ 'hinge', 'log', 'modified_huber', 'squared_hinge'}
ASGD	επαναλήψεις=1 έως 110, τιμές=10 loss={ 'hinge', 'log', 'modified_huber', 'squared_hinge', 'perceptron'}
Perceptron	επαναλήψεις=1 έως 110, τιμές=10 Ρυθμός μάθησης eta0 = [0.0001, 0.001, 0.01, 0.1, 1],
MLP	Αριθμός κρυφών επιπέδων=1-3, Αριθμός νευρώνων ανά επίπεδο=[15,20,25,30,37], Αριθμός επαναλήψεων=[50,100,200,300], Συνάρτηση βελτιστοποίησης=[SGD,Adam], Συνάρτηση ενεργοποίησης=[relu, tanh, linear], Ρυθμός μάθησης=[0.001, 0.01, 0.1, 1], Folds=5,10

Πίνακας 3: Τελικό εύρος τιμών αναζήτησης βέλτιστων υπερ-παραμέτρων

7.3 *Αποτελέσματα*

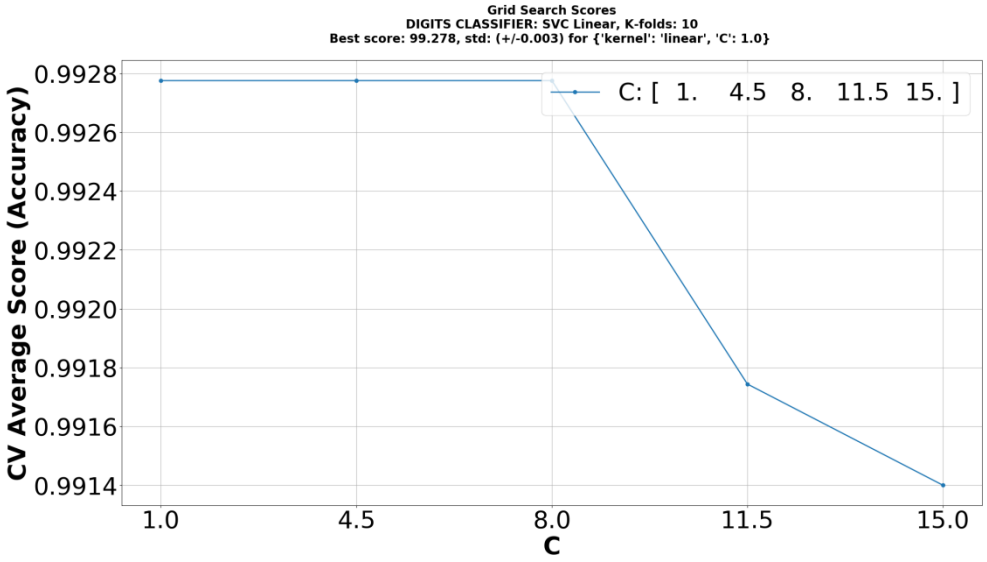
Για την καλύτερη αναπαράσταση των αποτελεσμάτων δημιουργήθηκαν δύο γραφικές παραστάσεις για κάθε μέθοδο, με εξαίρεση την MLP. Στην πρώτη, φαίνεται η ακρίβεια (*accuracy*) για συνδυασμό των δύο υπερ-παραμέτρων και στην δεύτερη φαίνεται η ακρίβεια του τμήματος εκπαίδευσης και του τμήματος ελέγχου σε σχέση με το πλήθος των δεδομένων (καμπύλη εκπαίδευσης).

Για την υλοποίηση του δεύτερου γραφήματος, χρησιμοποιήθηκαν οι βέλτιστες τιμές υπερ-παραμέτρων που βρέθηκαν κατά την αναζήτηση πλέγματος. Έπειτα με την μέθοδο *ShuffleSplit* χωρίστηκε το σύνολο δεδομένων σε 10 τμήματα, δηλαδή για K=10, όπου το 1/10 χρησιμοποιήθηκε ως τμήμα ελέγχου. Η διαδικασία αυτή επαναλήφθηκε όχι 10 αλλά 100 φορές με σκοπό ένα πιο ομαλό μέσο όρο για τα δύο τμήματα.

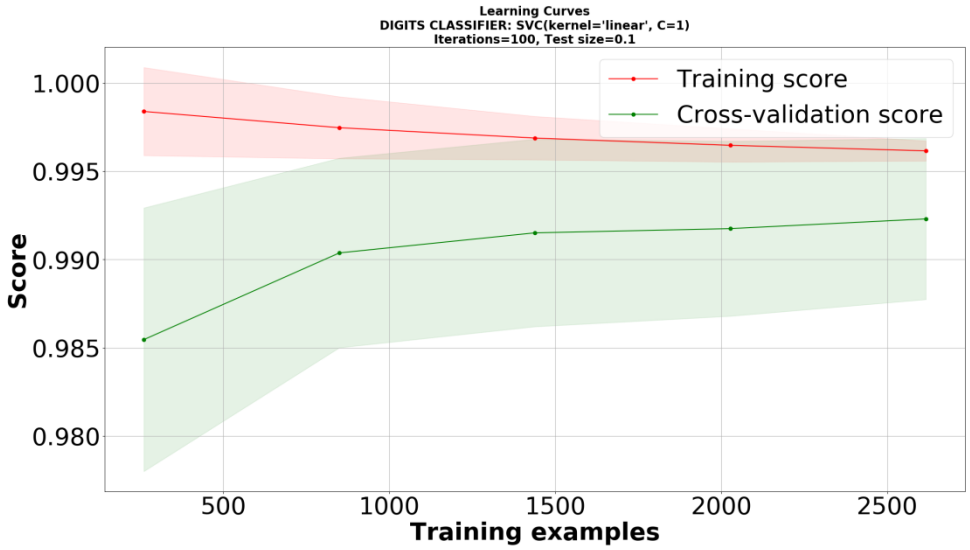
Να υπενθυμίσουμε πως οι αριθμοί και οι χαρακτήρες εκπαιδεύτηκαν ως ξεχωριστά σύνολα δεδομένων. Έτσι ο συνολικός αριθμών των γραφημάτων ανέρχεται στα 34.

Στην μέθοδο MLP δημιουργήθηκε μόνο ένα διάγραμμα. Ο λόγος είναι πως στην συγκεκριμένη μέθοδο αναζητήθηκαν πολλές υπερ-παράμετροι που δεν μπορούσαν να αναπαρασταθούν εύκολα. Έτσι δημιουργήθηκε ένα μόνο διάγραμμα το οποίο δείχνει την ακρίβεια σε σχέση με τις επαναλήψεις.

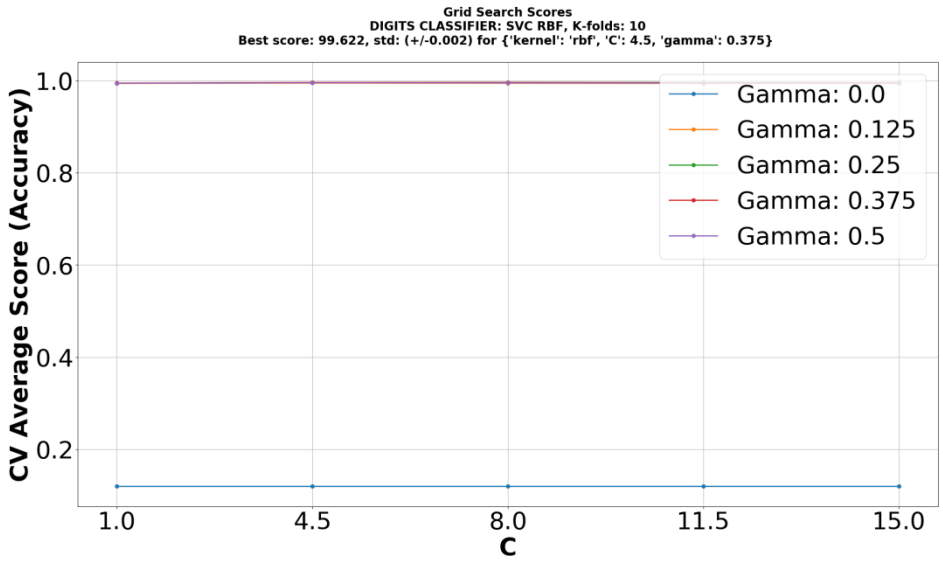
7.3.1 Αποτελέσματα συνόλου δεδομένων αριθμών



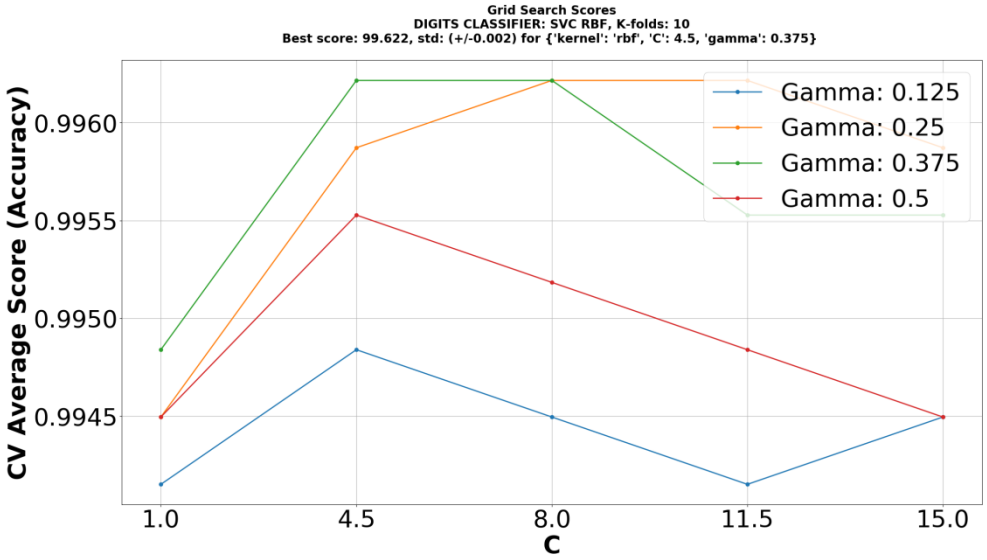
Σχήμα 28: Αναζήτηση πλέγματος, Μέθοδος SVM Linear, Σύνολο δεδομένων Αριθμών



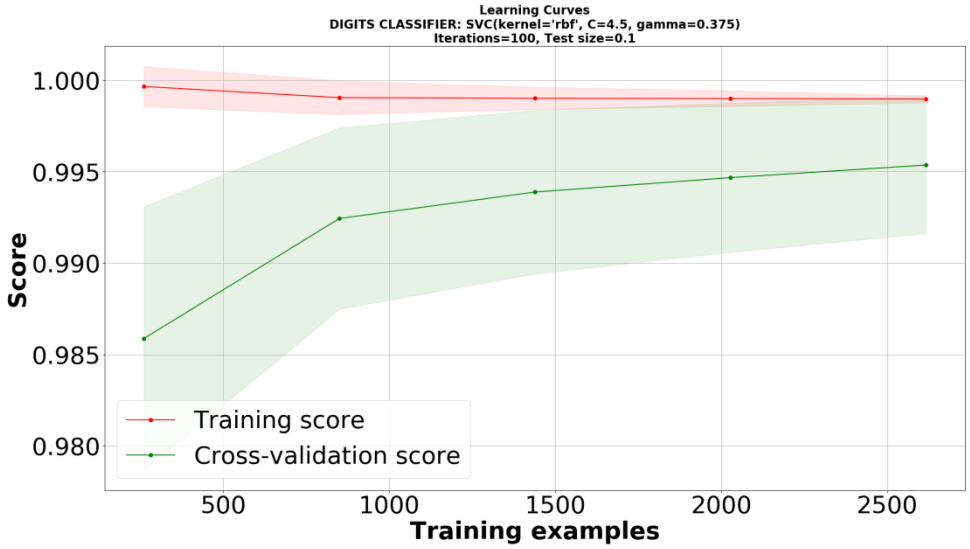
Σχήμα 29: Καμπύλη εκπαίδευσης, Μέθοδος SVM Linear, Σύνολο δεδομένων Αριθμών



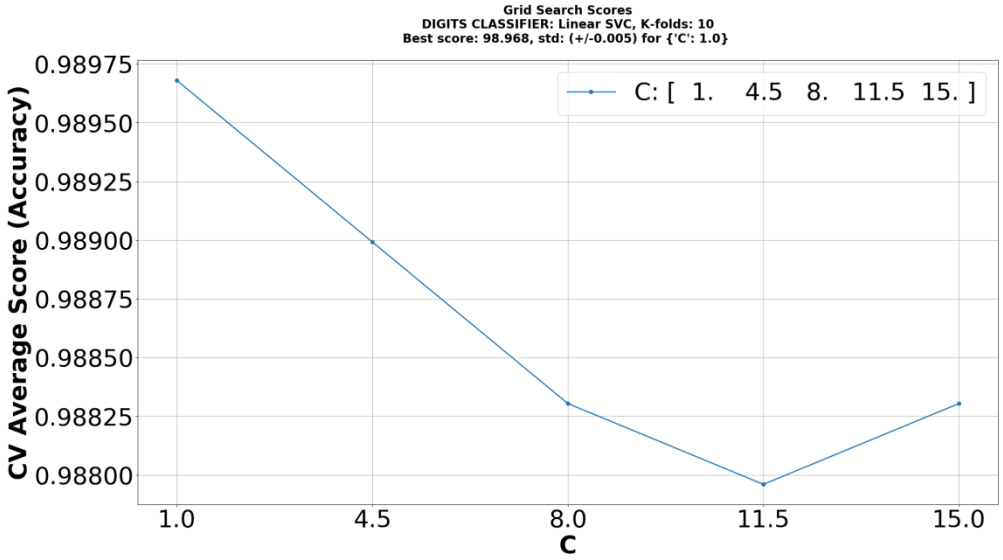
Σχήμα 30: Αναζήτηση πλέγματος, Μέθοδος SVM RBF, Σύνολο δεδομένων Αριθμών (1)



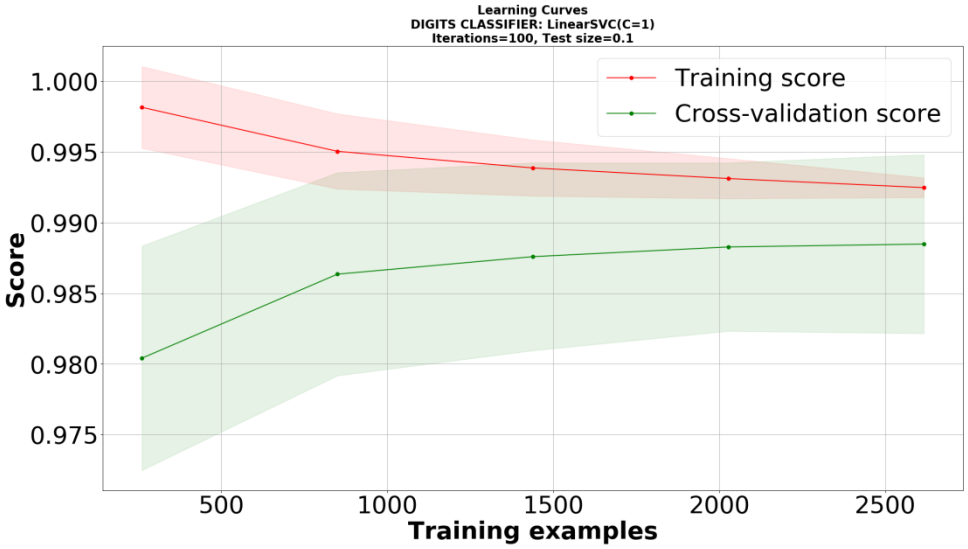
Σχήμα 31: Αναζήτηση πλέγματος, Μέθοδος SVM RBF, Σύνολο δεδομένων Αριθμών (2)



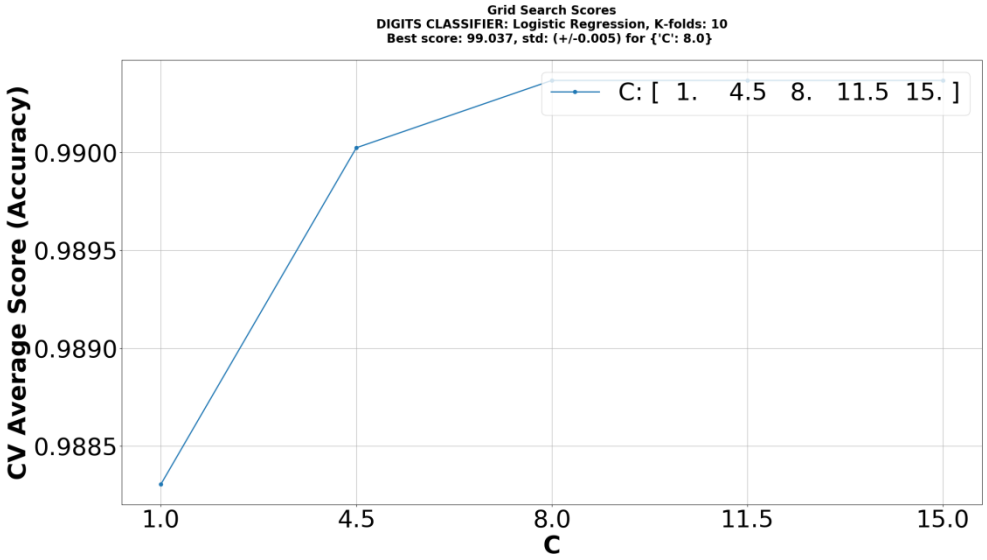
Σχήμα 32: Καμπύλη εκπαίδευσης, Μέθοδος SVM RBF, Σύνολο δεδομένων Αριθμών



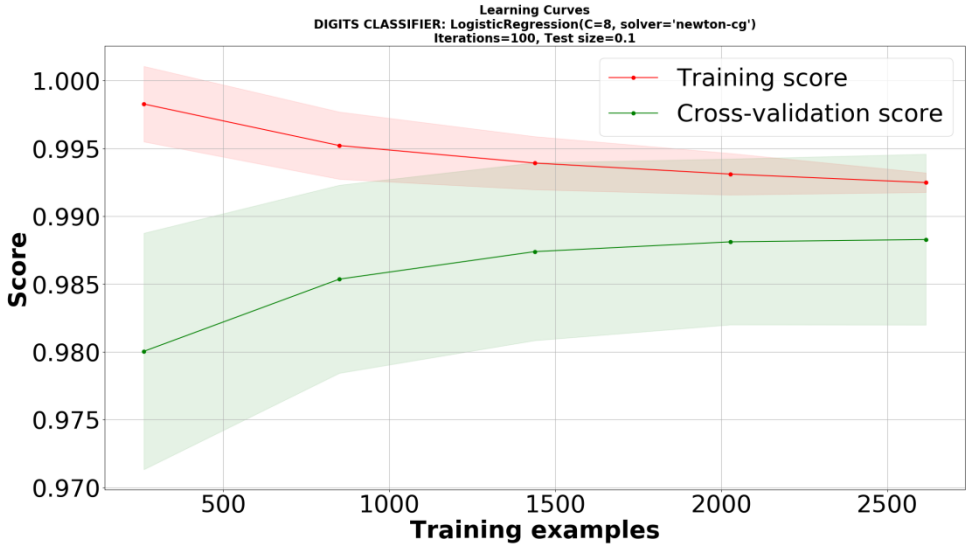
Σχήμα 33: Αναζήτηση πλέγματος, Μέθοδος Linear SVM, Σύνολο δεδομένων Αριθμών



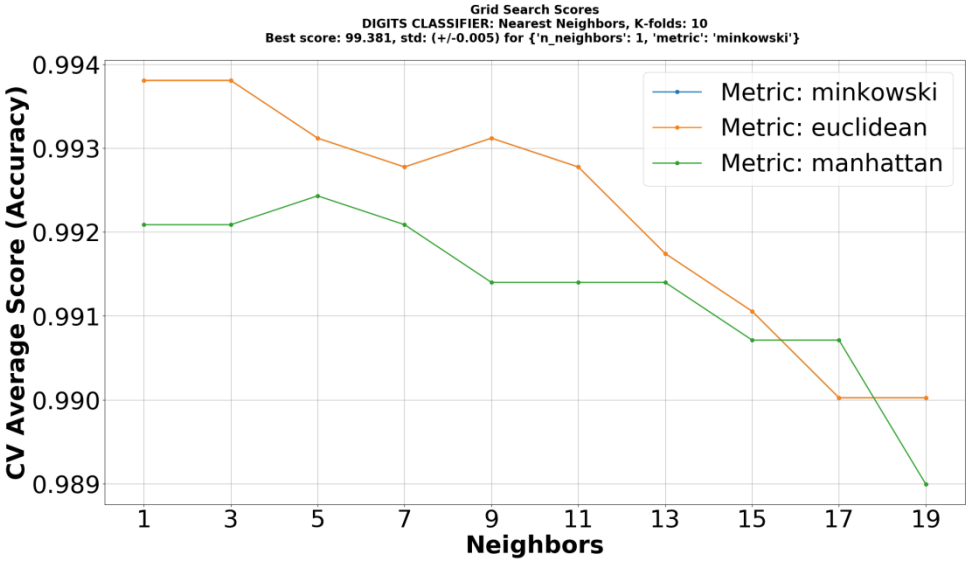
Σχήμα 34: Καμπύλη εκπαίδευσης, Μέθοδος Linear SVM, Σύνολο δεδομένων Αριθμών



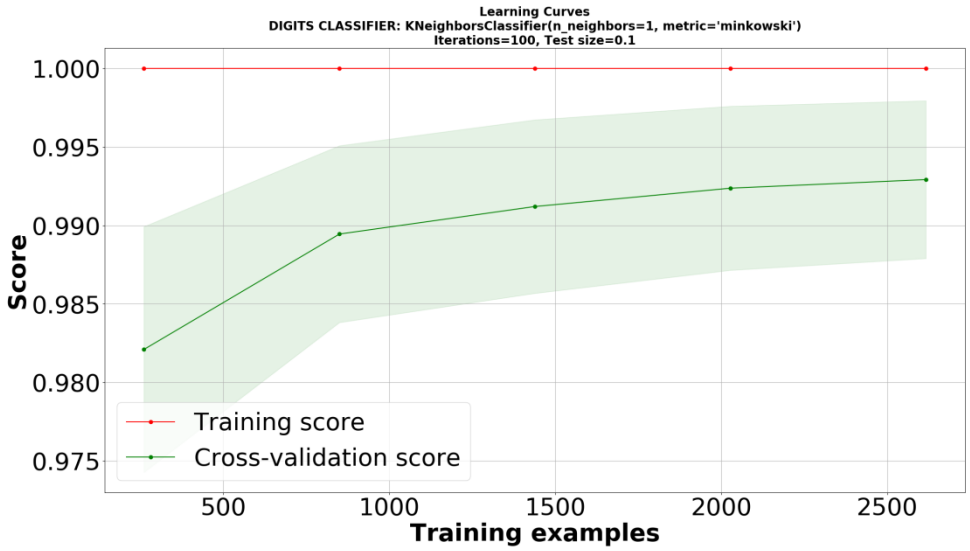
Σχήμα 35: Αναζήτηση πλέγματος, Μέθοδος Logistic Regression, Σύνολο δεδομένων Αριθμών



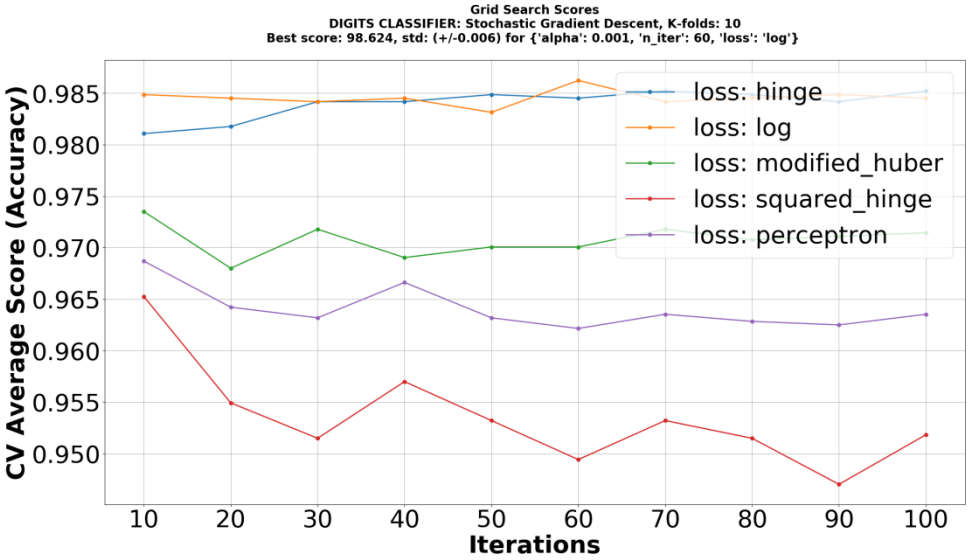
Σχήμα 36: Καμπύλη εκπαίδευσης, Μέθοδος Logistic Regression, Σύνολο δεδομένων Αριθμών



Σχήμα 37: Αναζήτηση πλέγματος, Μέθοδος KNN, Σύνολο δεδομένων Αριθμών

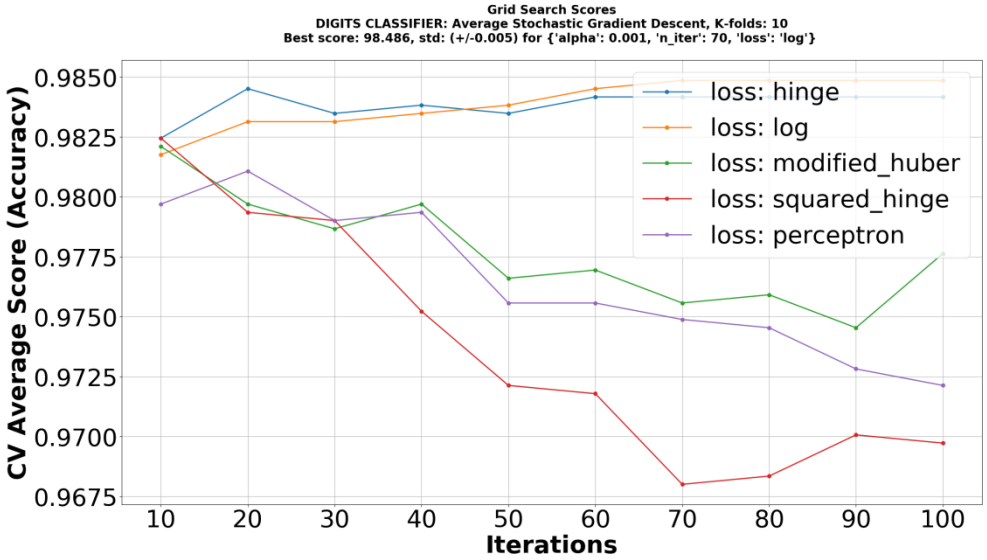


Σχήμα 38: Καμπύλη εκπαίδευσης, Μέθοδος KNN, Σύνολο δεδομένων Αριθμών

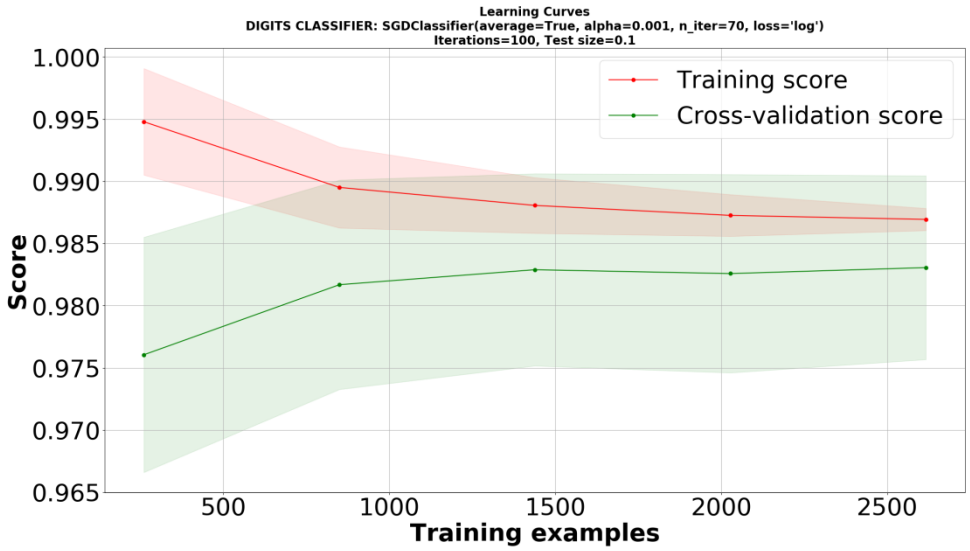


Σχήμα 39: Αναζήτηση πλέγματος, Μέθοδος SGD, Σύνολο δεδομένων Αριθμών

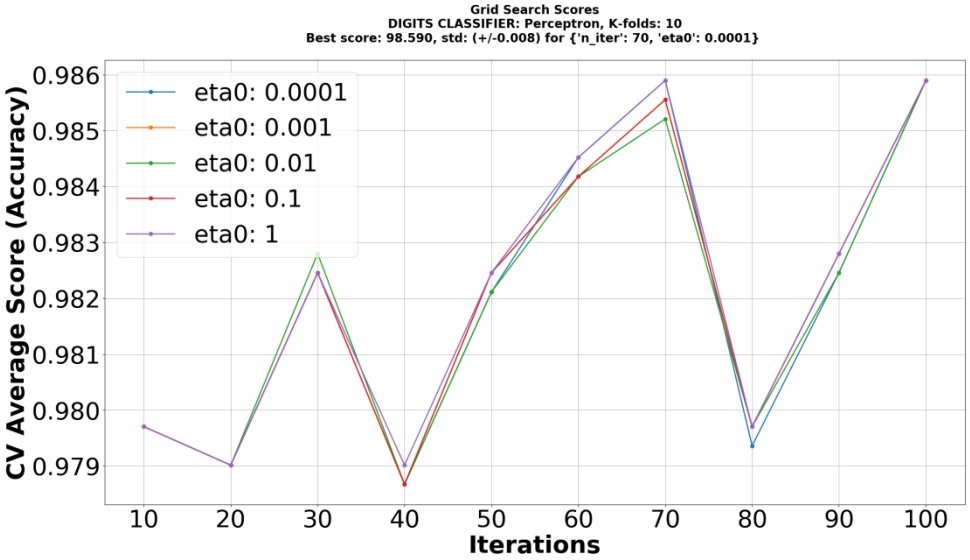
Σχήμα 40: Καμπύλη εκπαίδευσης, Μέθοδος SGD, Σύνολο δεδομένων Αριθμών



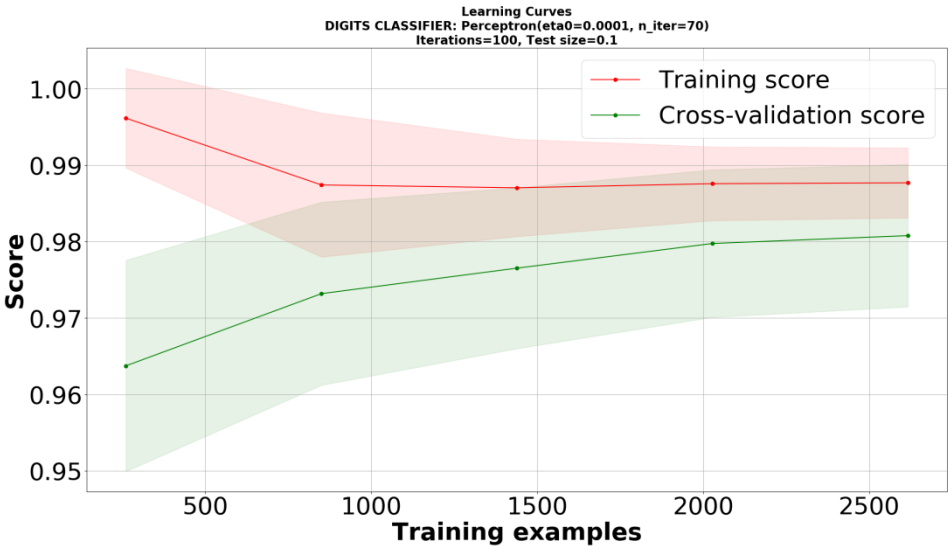
Σχήμα 41: Αναζήτηση πλέγματος, Μέθοδος ASGD, Σύνολο δεδομένων Αριθμών



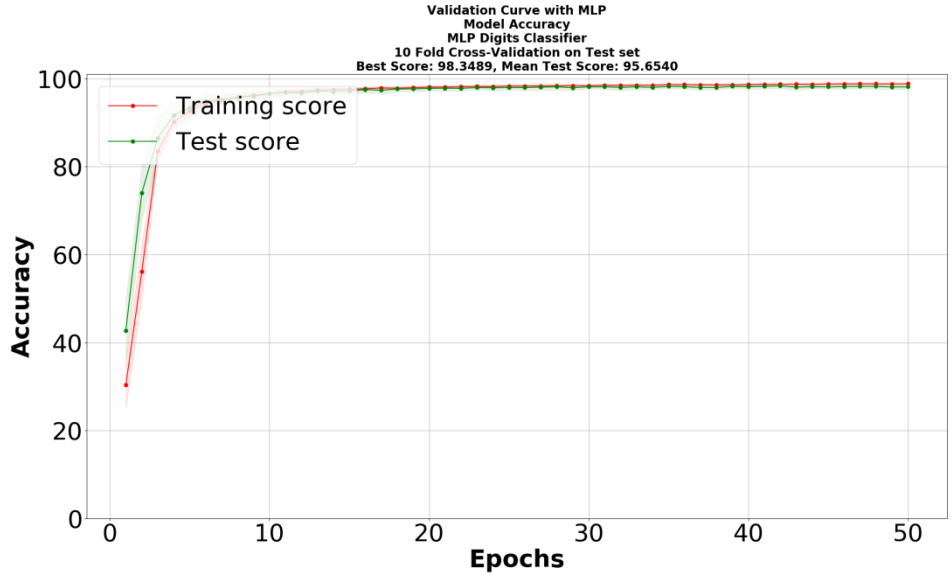
Σχήμα 42: Καμπύλη εκπαίδευσης, Μέθοδος ASGD, Σύνολο δεδομένων Αριθμών



Σχήμα 43: Αναζήτηση πλέγματος, Μέθοδος Perceptron, Σύνολο δεδομένων Αριθμών



Σχήμα 44: Καμπύλη εκπαίδευσης, Μέθοδος Perceptron, Σύνολο δεδομένων Αριθμών

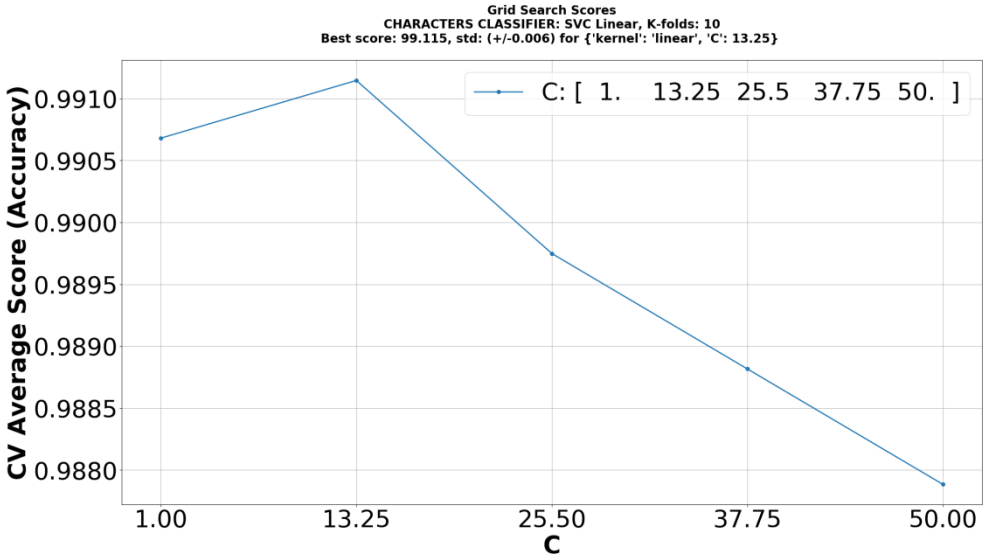


Σχήμα 45: Καμπύλη επικύρωσης, Μέθοδος MLP, Σύνολο δεδομένων Αριθμών

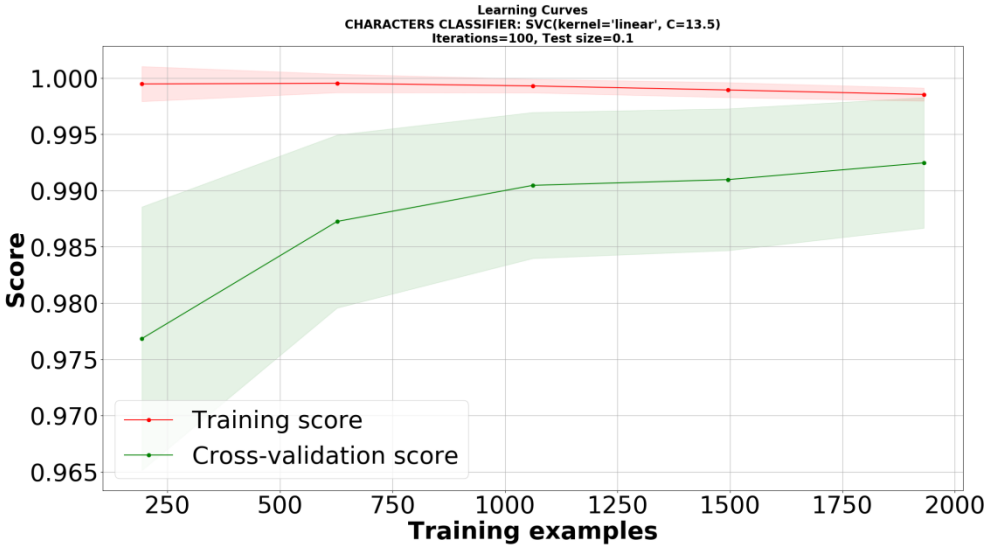
Μέθοδος Μηχανικής Μάθησης	**Βέλτιστες τιμές υπερ-παραμέτρων**	**Best Score % (Accuracy)**	**+/- std dev**
SVM Linear	C=1	99.278	0.003
SVM RBF	C=4.5, gamma=0.375	**99.622**	**0.002**
Linear SVM	C=1	98.968	0.005
Logistic Regression	C=8	99.037	0.005
kNN	K=1, μετρική=minkowski	99.381	0.005
SGD	alpha=0.001 επαναλήψεις=60 loss={ log}	98.624	0.006
ASGD	alpha=0.001 επαναλήψεις=70 loss={ 'log'}	98.486	0.005
Perceptron	Eta0={ 0.0001} επαναλήψεις=70	98.590	0.008
MLP	1ο κρυφό επίπεδο 37 νευρώνες, 2ο κρυφό επίπεδο 20 νευρώνες, Folds=10, Επαναλήψεις=50, Optimizer=Adam, Ρυθμός μάθησης=0.001, Συνάρτηση ενεργοποίησης=relu	98.349	-

Πίνακας 4: Αποτελέσματα εκπαίδευσης Αριθμών

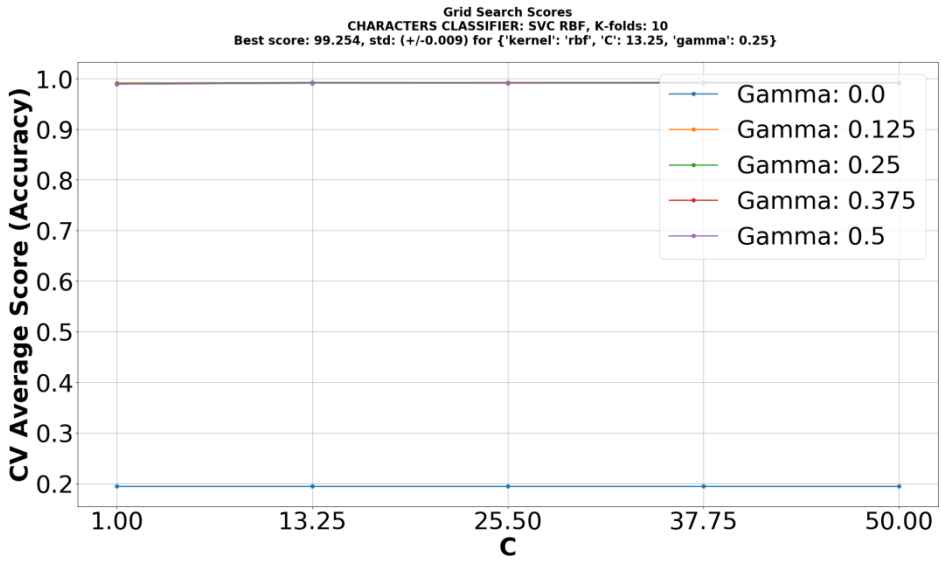
7.3.2 Αποτελέσματα συνόλου δεδομένων χαρακτήρων



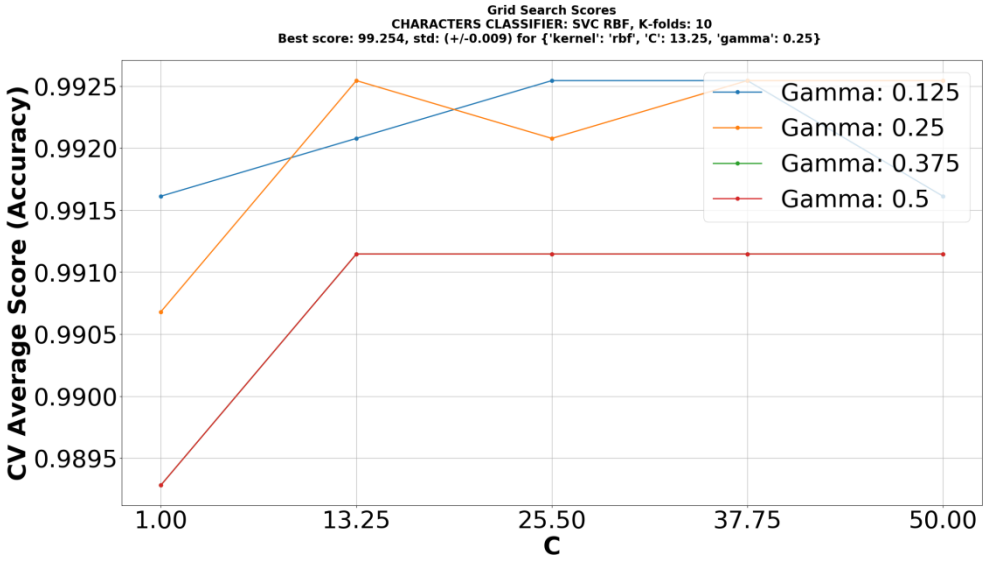
Σχήμα 46: Αναζήτηση πλέγματος, Μέθοδος SVM Linear, Σύνολο δεδομένων Χαρακτήρων



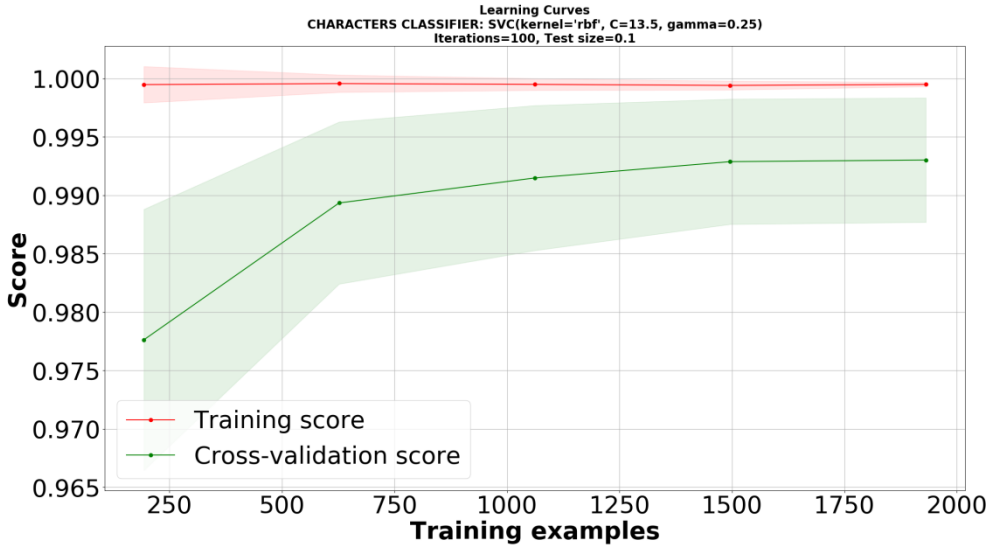
Σχήμα 47: Καμπύλη εκπαίδευσης, Μέθοδος SVM Linear, Σύνολο δεδομένων Χαρακτήρων



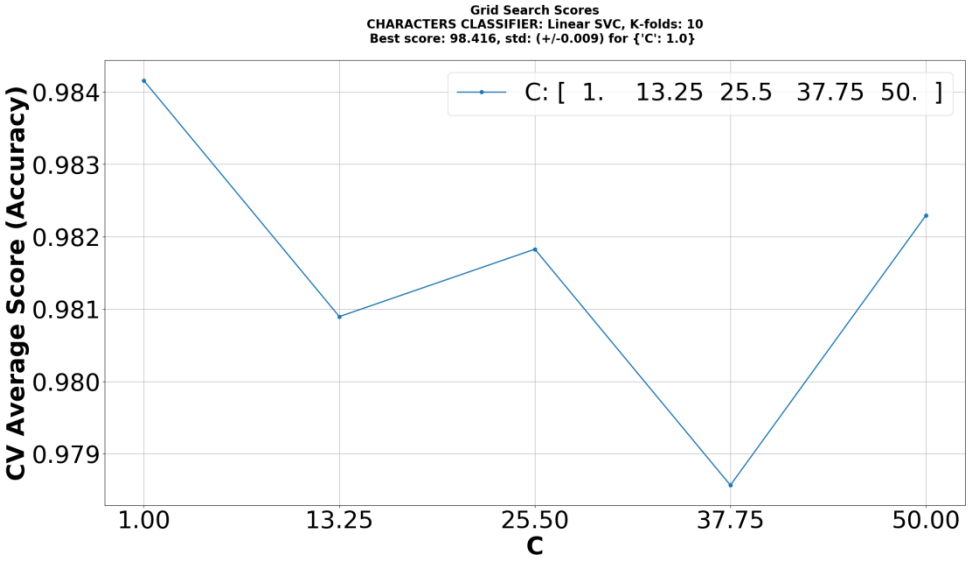
Σχήμα 48: Αναζήτηση πλέγματος, Μέθοδος SVM RBF, Σύνολο δεδομένων Χαρακτήρων (1)



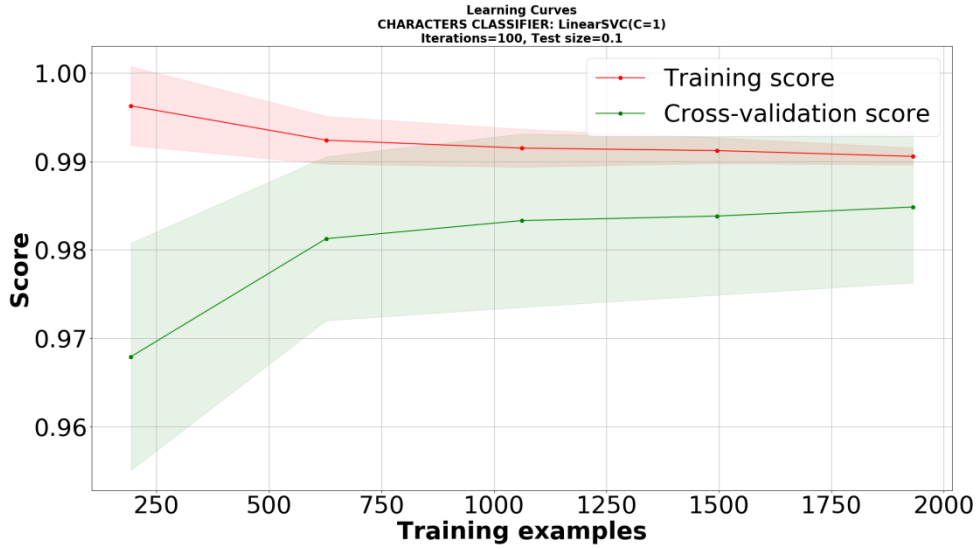
Σχήμα 49: Αναζήτηση πλέγματος, Μέθοδος SVM RBF, Σύνολο δεδομένων Χαρακτήρων (2)



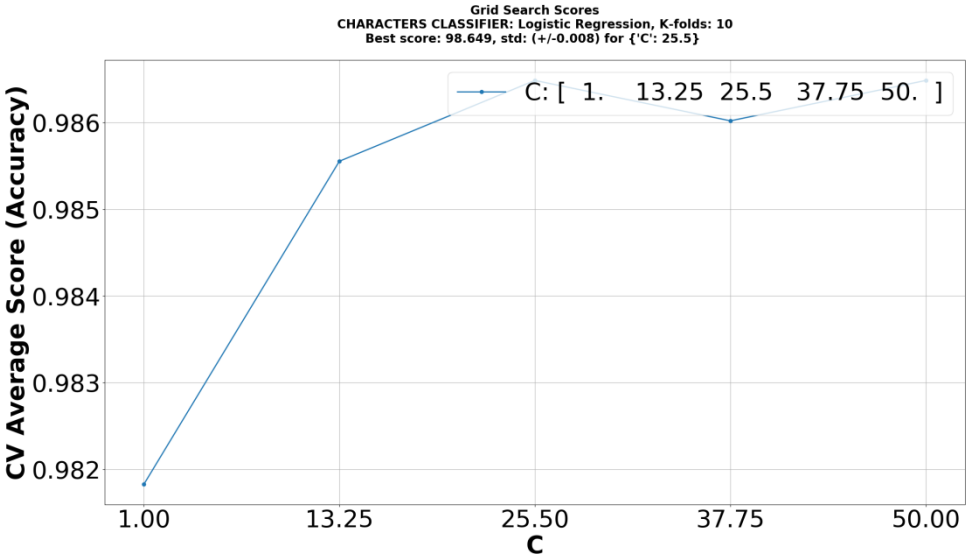
Σχήμα 50: Καμπύλη εκπαίδευσης, Μέθοδος SVM RBF, Σύνολο δεδομένων Χαρακτήρων



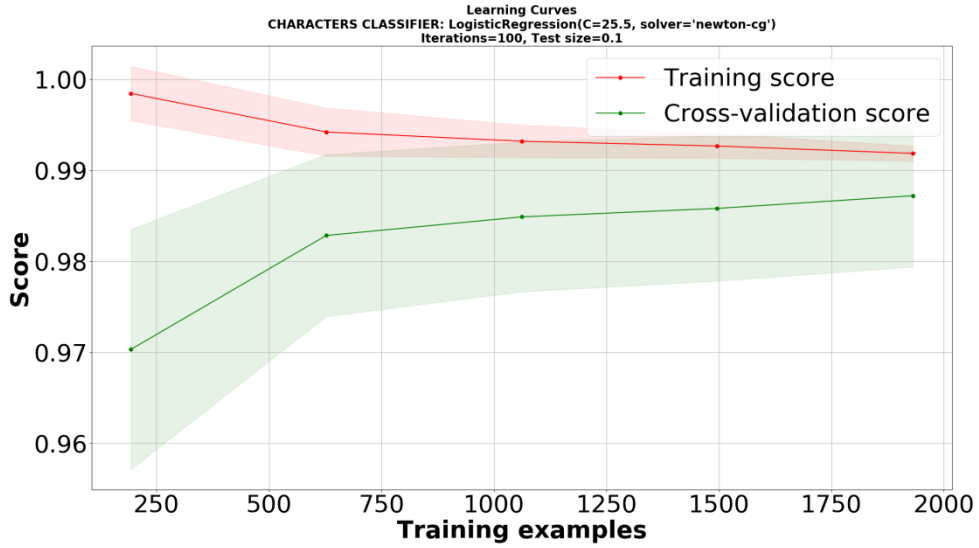
Σχήμα 51: Αναζήτηση πλέγματος, Μέθοδος Linear SVM, Σύνολο δεδομένων Χαρακτήρων



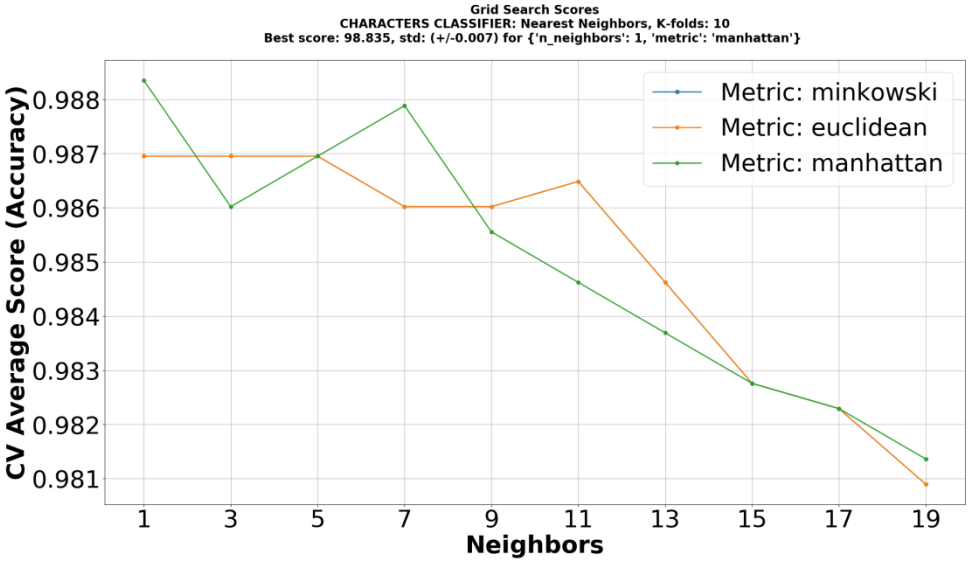
Σχήμα 52: Καμπύλη εκπαίδευσης, Μέθοδος Linear SVM, Σύνολο δεδομένων Χαρακτήρων



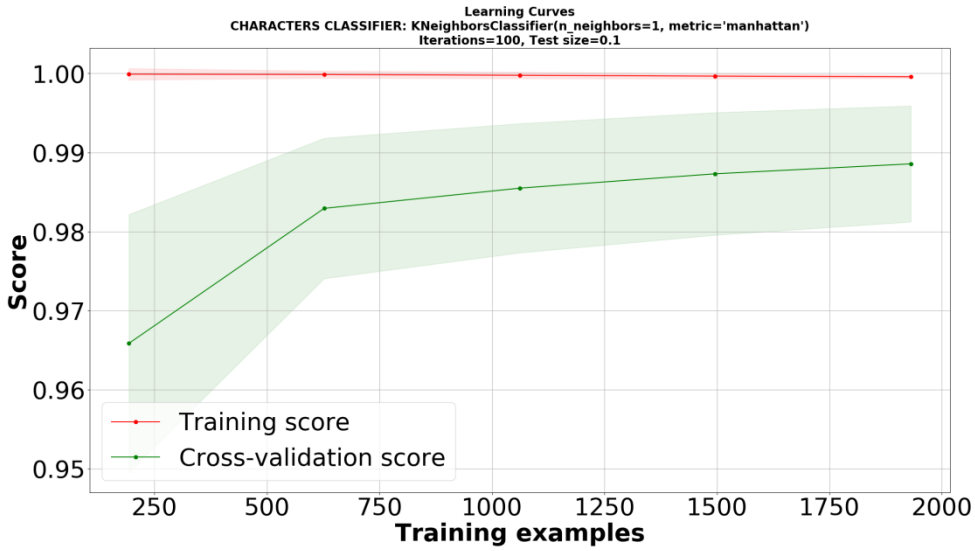
Σχήμα 53: Αναζήτηση πλέγματος, Μέθοδος Logistic Regression, Σύνολο δεδομένων Χαρακτήρων



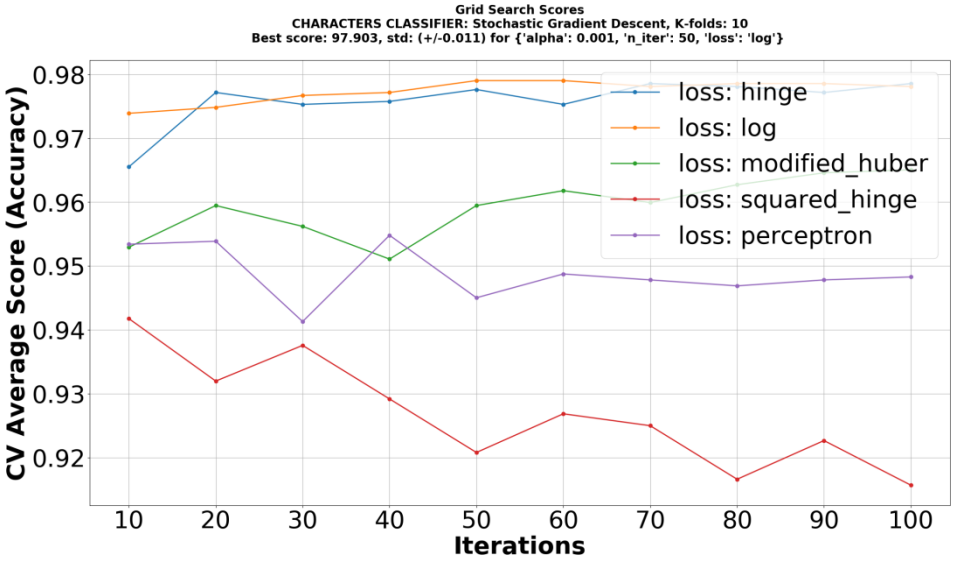
Σχήμα 54: Καμπύλη εκπαίδευσης, Μέθοδος Logistic Regression, Σύνολο δεδομένων Χαρακτήρων



Σχήμα 55: Αναζήτηση πλέγματος, Μέθοδος KNN, Σύνολο δεδομένων Χαρακτήρων

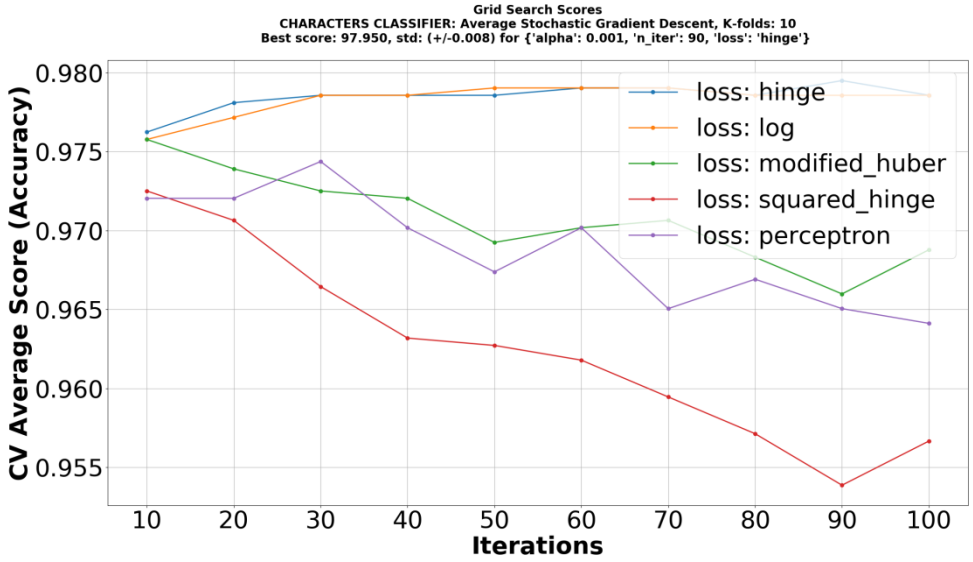


Σχήμα 56: Καμπύλη εκπαίδευσης, Μέθοδος KNN, Σύνολο δεδομένων Χαρακτήρων

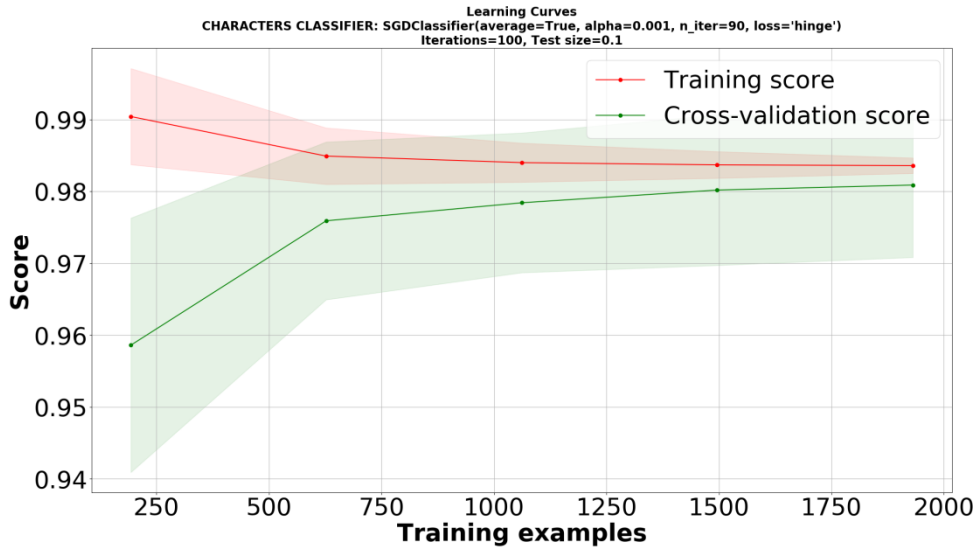


Σχήμα 57: Αναζήτηση πλέγματος, Μέθοδος SGD, Σύνολο δεδομένων Χαρακτήρων

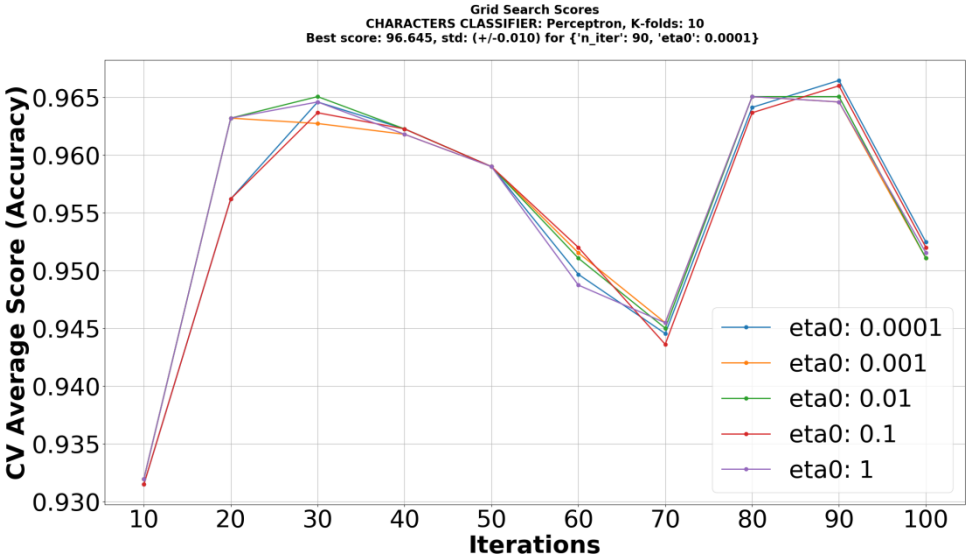
Σχήμα 58: Καμπύλη εκπαίδευσης, Μέθοδος SGD, Σύνολο δεδομένων Χαρακτήρων



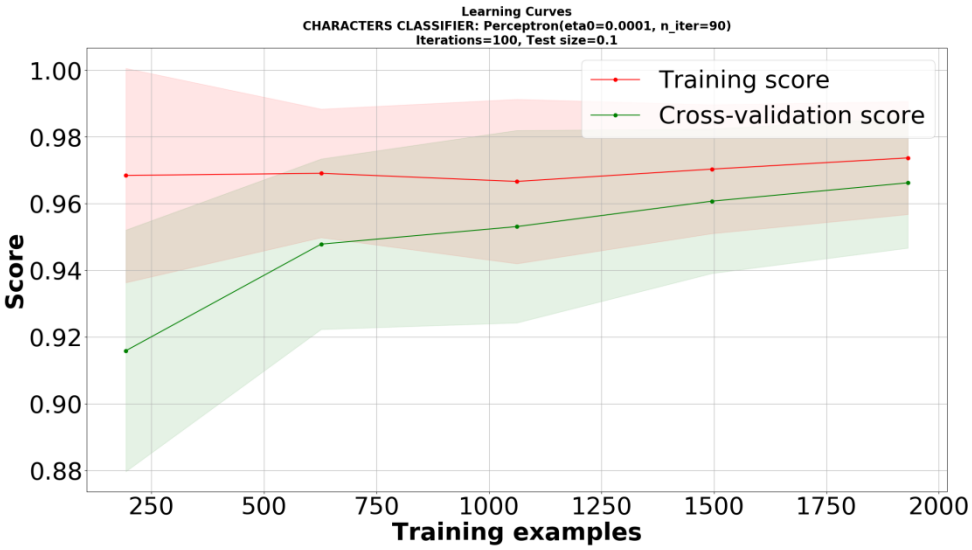
Σχήμα 59: Αναζήτηση πλέγματος, Μέθοδος ASGD, Σύνολο δεδομένων Χαρακτήρων



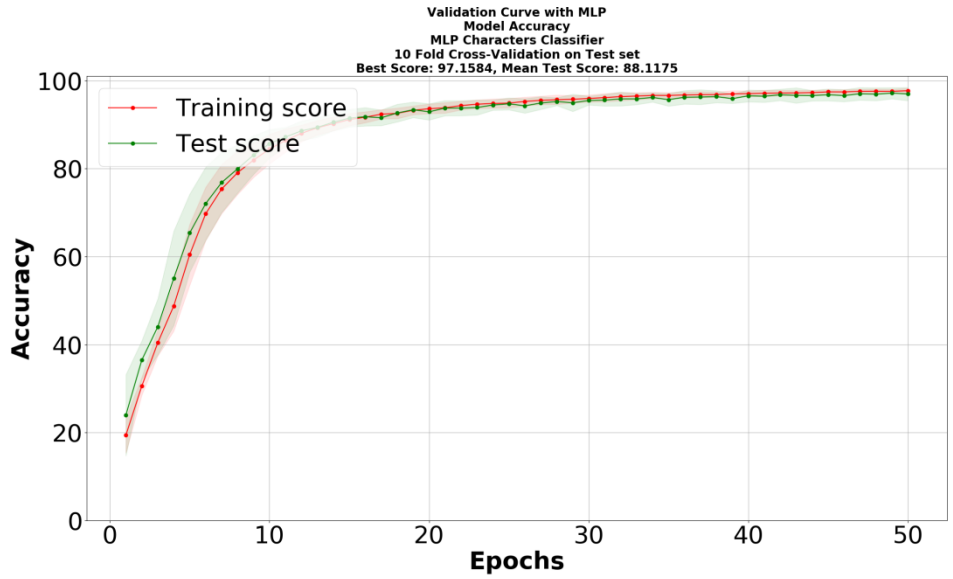
Σχήμα 60: Καμπύλη εκπαίδευσης, Μέθοδος ASGD, Σύνολο δεδομένων Χαρακτήρων



Σχήμα 61: Αναζήτηση πλέγματος, Μέθοδος Perceptron, Σύνολο δεδομένων Χαρακτήρων



Σχήμα 62: Καμπύλη εκπαίδευσης, Μέθοδος Perceptron, Σύνολο δεδομένων Χαρακτήρων



Σχήμα 63: Καμπύλη επικύρωσης, Μέθοδος MLP, Σύνολο δεδομένων Χαρακτήρων

Μέθοδος Μηχανικής Μάθησης	**Βέλτιστες τιμές υπερ-παραμέτρων**	**Best Score % (Accuracy)**	**+/- std dev**
SVM Linear	C=13.5	99.115	0.006
SVM RBF	C=13.25, gamma=0. 25	**99.254**	**0.009**
Linear SVM	C=1	98.416	0.009
Logistic Regression	C=25.5	98.649	0.008
kNN	K=1, μετρική=manhattan	98.835	0.007
SGD	alpha=0.001 επαναλήψεις=50 loss={ log}	97.903	0.011
ASGD	alpha=0.001 επαναλήψεις=90 loss={ 'hinge'}	97.950	0.008
Perceptron	eta0={ 0.0001} επαναλήψεις=90	96.645	0.010
MLP	1ο κρυφό επίπεδο 37 νευρώνες, 2ο κρυφό επίπεδο 20 νευρώνες, Folds=10, Επαναλήψεις=50, Optimizer=Adam, Ρυθμός μάθησης=0.001, Συνάρτηση ενεργοποίησης=relu	97.158	-

Πίνακας 5: Αποτελέσματα εκπαίδευσης Χαρακτήρων

7.4 *Σύνοψη συμπερασμάτων αξιολόγησης*

Όπως φαίνεται στα διαγράμματα όλες οι μέθοδοι έχουν δώσει πολύ ικανοποιητικά αποτελέσματα, της τάξης των 96% έως 99% για τους χαρακτήρες και 98% έως 99% για τους αριθμούς. Και στα δύο σύνολα δεδομένων η μέθοδος που ξεχώρισε είναι η SVM με πυρήνα RBF και είχε μέγιστο σκορ επιτυχίας (ακρίβεια) 99.622% για το σύνολο δεδομένων των αριθμών και 99.254% για το σύνολο δεδομένων των χαρακτήρων.

Επίσης, από τα διαγράμματα με τις καμπύλες εκπαίδευσης και τις καμπύλες επικύρωσης φαίνεται πως όλα τα μοντέλα έχουν εκπαιδευτεί χωρίς *over-fitting* ή *under-fitting.* Αυτό γίνεται κατανοητό διότι οι γραμμές Training score και Test score ή Cross-validation score αντίστοιχα, συγκλίνουν σε μεγάλη ακρίβεια. Σε αντίθετη περίπτωση θα λέγαμε πως τα μοντέλα έχουν υπερ-εκπαιδευτεί (over-fitting) εάν οι δύο γραμμές αποκλίνανε, ενώ εάν οι γραμμές συγκλίνανε σε χαμηλή ακρίβεια τα μοντέλα μας θα χρειαζόντουσαν καλύτερη εκπαίδευση (under-fitting).

Παρατηρούμε πως οι αλλαγές στην υπερ-παράμετρο *gamma* της μεθόδου SVM με πυρήνα RBF και η υπερ-παράμετρος *eta0* του νευρώνα Perceptron είχαν πολύ μικρές επιρροές στα αποτελέσματα, ενώ στην μέθοδο KNN οι μετρικές *minkowski* και η ευκλείδεια απόσταση δίνουν ακριβώς τα ίδια αποτελέσματα.

Θα περίμενε κανείς πως η μέθοδος βαθειάς μάθησης MLP θα είχε καλύτερα αποτελέσματα από τις υπόλοιπες μεθόδους. Κατά την αναζήτηση πλέγματος δοκιμάστηκαν περισσότερες επαναλήψεις, συγκεκριμένα έως 300, με βελτίωση στην απόδοση μόλις 0.3%.

8

Τεχνικές λεπτομέρειες

Στο κεφάλαιο αυτό καταγράφονται τα προγραμματιστικά εργαλεία και οι εκδόσεις που χρησιμοποιήθηκαν για την υλοποίηση των πειραμάτων και της εφαρμογής.

8.1 *Πλατφόρμες και προγραμματιστικά εργαλεία*

Όλος ο κώδικας στην παρούσα εργασία γράφτηκε στην γλώσσα προγραμματισμού Python. Ο λόγος που επιλέχτηκε η συγκεκριμένα γλώσσα είναι η ευελιξία και η απλότητά της καθώς και οι βιβλιοθήκες για τις μεθόδους μηχανικής μάθησης και επεξεργασίας εικόνας που υπάρχουν για αυτήν. Όλα τα πειράματα εκτελέστηκαν σε λειτουργικό σύστημα Windows 7 64bit αλλά ο κώδικας είναι συμβατός με όλα τα λειτουργικά συστήματα χωρίς να γίνει κάποια αλλαγή. Παρακάτω δίνεται μια περιγραφή των βασικών βιβλιοθηκών που χρησιμοποιήθηκαν καθώς και οι εκδόσεις τους.

Επειδή η βιβλιοθήκη που υλοποιεί τις μεθόδους βαθειάς μάθησης απαιτεί νεότερη έκδοση της Python από τις υπόλοιπες βιβλιοθήκες, δημιουργήθηκαν δύο ξεχωριστά περιβάλλοντα εργασίας, οι βιβλιοθήκες και εκδόσεις των οποίων καταγράφονται στους πίνακες 6 και 7.

8.1.1 OpenCV 2.4.13.2

Για την επεξεργασία εικόνας επιλέχτηκε η βιβλιοθήκη OpenCV[12] η οποία παρέχει δυνατότητα σύνδεσης με αρκετές γλώσσες προγραμματισμού όπως Python, Java, C, C++, Matlab, Ruby κ.α. ενώ είναι συμβατή με λειτουργικά συστήματα Windows, Linux, MacOS, BSD, Android, iOS και BlackBerry.

Η έκδοση της βιβλιοθήκης που χρησιμοποιήθηκε είναι η 2.4.13.2 ενώ η τελευταία έκδοση είναι η 3.2 στην οποία έχουν γίνει αρκετές προσθήκες αλλά ο κώδικας που γράφτηκε στην παρούσα εργασία δεν είναι 100% συμβατή με αυτήν.

8.1.2 Python

Η υλοποίηση της επεξεργασίας εικόνας και η εκτέλεση των πειραμάτων με μεθόδους μηχανικής μάθησης, εκτός της βαθειάς μάθησης, έγινε σε Python[13] 2.7.12.

8.1.3 Scikit-learn

Η βιβλιοθήκη scikit-learn[14] υλοποιεί πολλές μεθόδους μηχανικής μάθησης και προ-επεξεργασίας των δεδομένων, όπως μείωση διαστάσεων (*dimensionality reduction*) και διασταύρωσης (*Cross Validation*) και είναι η νούμερο 1 επιλογή όταν πρόκειται για επίλυση προβλημάτων μηχανικής μάθησης στη γλώσσα προγραμματισμού Python. Η έκδοση που χρησιμοποιήθηκε είναι η 0.18.1 η οποία κατά το διάστημα της συγγραφής ήταν η τελευταία σταθερή έκδοση της βιβλιοθήκης.

8.1.4 Keras

Η βιβλιοθήκη Keras[15] είναι ένα *wrapper* υψηλότερου επιπέδου των βιβλιοθηκών Tensoflow[16] και Theano[17] και υλοποιεί μεθόδους βαθειάς μάθησης. Ο χρήσης καλείται να επιλέξει ποια από τις δύο βιβλιοθήκες θέλει να χρησιμοποιήσει ως backend. Για την εκτέλεση μεθόδων

[12] http://opencv.org/

[13] https://www.python.org/

[14] http://scikit-learn.org/

[15] https://keras.io/

[16] https://www.tensorflow.org/

[17] http://deeplearning.net/software/theano/

βαθειάς μάθησης χρησιμοποιήθηκε η τελευταία σταθερή έκδοση της βιβλιοθήκης (2.0.4) με backend το Tensoflow και συγκεκριμένα την βιβλιοθήκη που υλοποιεί επιτάχυνση μέσω κάρτας γραφικών (GPU), για την ταχύτερη εκτέλεση της εκπαίδευσης.

8.1.5 Περιβάλλον Εργασίας Python 2

boto==2.39.0	PGen==0.2.1
commentjson==0.6	Pillow==3.2.0
cycler==0.10.0	progressbar==2.3
Cython==0.23.4	pyparsing==2.2.0
dask==0.9.0	python-dateutil==2.6.0
decorator==4.0.9	pytz==2017.2
functools32==3.2.3.post2	redis==2.10.5
httplib2==0.9.2	scikit-image==0.12.3
imutils==0.3.5	scikit-learn==0.18.1
mahotas==1.4.1	scipy==0.18.1
matplotlib==2.0.2	sh==1.12.10
networkx==1.11	six==1.10.0
numpy==1.13.0+mkl	toolz==0.7.4
opencv-python==2.4.13.2	twilio==5.4.0

Πίνακας 6: Βιβλιοθήκες περιβάλλοντος Python 2

8.1.6 Περιβάλλον Εργασίας Python 3.5

appdirs==1.4.3	networkx==1.11	PyWavelets==0.5.2
bleach==1.5.0	numpy==1.13.0+mkl	PyYAML==3.12
curses==2.2	olefile==0.44	scikit-image==0.13.0
cycler==0.10.0	packaging==16.8	scikit-learn==0.18.1
decorator==4.0.11	pandas==0.20.2	scipy==0.19.0
graphviz==0.7.1	Pillow==4.1.1	six==1.10.0
h5py==2.7.0	protobuf==3.3.0	tensorflow-gpu==1.1.0
html5lib==0.9999999	pydot==1.2.3	tflearn==0.3

Keras==2.0.4	pyparsing==2.2.0	Theano==0.9.0
Markdown==2.2.0	python-dateutil==2.6.0	Werkzeug==0.12.1
matplotlib==2.0.0	pytz==2017.2	

Πίνακας 7: Βιβλιοθήκες περιβάλλοντος Python 3.5

9

Επίλογος

Στο κεφάλαιο αυτό θα γίνει μια σύνοψη της διπλωματικής εργασίας, θα παρουσιαστούν κάποια γενικά συμπεράσματα με βάση τα πειραματικά αποτελέσματα και τέλος θα αναφερθούν πιθανές μελλοντικές επεκτάσεις της διπλωματικής

9.1 *Σύνοψη και συμπεράσματα*

Σκοπός της διπλωματικής εργασίας ήταν να γίνει μια παρουσίαση βασικών μεθόδων επεξεργασίας εικόνας, ο συνδυασμός των οποίων μπορεί εύκολα να χρησιμοποιηθεί για την υλοποίηση αλγορίθμου εντοπισμού πινακίδων κυκλοφορίας αυτοκινήτων σε φωτογραφίες και την εξαγωγή των χαρακτήρων της. Με τους χαρακτήρες που εξάχθηκαν, εκπαιδεύτηκαν 9 μοντέλα μηχανικής μάθησης, και η αξιολόγησή τους έγινε ως προς το ποσοστό αναγνώρισης των χαρακτήρων (ακρίβεια).

Ένα βασικό πρόβλημα που προέκυψε ήταν η έλλειψη καλών συνόλων δεδομένων με φωτογραφίες από αυτοκίνητα που να είναι εμφανής η πινακίδα κυκλοφορίας. Ένα καλό σύνολο δεδομένων που εντοπίστηκε online ήταν αυτό του Medialab[18] (Εργαστήριο Τεχνολογίας Πολυμέσων του Εθνικού Μετσόβιου Πολυτεχνείου) με μοναδικό μειονέκτημα τον μικρό αριθμό φωτογραφιών, μόλις 139.

Έτσι δημιουργήθηκε νέα συλλογή δεδομένων με συνολικά 1118 φωτογραφίες αυτοκινήτων με εμφανείς τις πινακίδες κυκλοφορίας. Οι φωτογραφίες λήφθηκαν με κινητές τηλεφωνικές

[18] http://www.medialab.ntua.gr/research/LPRdatabase.html

συσκευές και ψηφιακές φωτογραφικές μηχανές, υπό διαφορετικές γωνίες και συνθήκες φωτισμού. Το τελικό σύνολο δεδομένων που χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία ήταν ο συνδυασμός των δύο αυτών συνόλων.

Ο αλγόριθμος εντοπισμού της πινακίδας και εξαγωγής χαρακτήρων δεν είχε τα επιθυμητά αποτελέσματα όσο αφορά την ακρίβεια. Συγκεκριμένα από τις 1257 εικόνες, ο αλγόριθμος κατάφερε να εντοπίσει σωστά και να εξάγει μόνο 5053 χαρακτήρες (από τους συνολικά 1257x7=8799 χαρακτήρες που θα μπορούσε να εξάγει), δηλαδή είχε ένα ποσοστό επιτυχίας 57,42%. Το χαμηλό αυτό ποσοστό μπορεί να οφείλεται στην υλοποίηση του αλγορίθμου αλλά σίγουρα οφείλεται στις διαφορετικές συνθήκες που λήφθηκαν οι φωτογραφίες.

Παρόλα αυτά, τα αποτελέσματα κατά την εκπαίδευση των μοντέλων ήταν εντυπωσιακά. Όλα τα μοντέλα απέδωσαν ποσοστό επιτυχίας πάνω από 96%. Το μοντέλο SVM με πυρήνα RBF κατείχε την πρώτη θέση τόσο στους αριθμούς όσο και στα γράμματα με ποσοστό 99,6% και 99,2% αντίστοιχα. Να υπενθυμίσουμε πως οι αριθμοί και τα γράμματα εκπαιδεύτηκαν ως ξεχωριστά σύνολα δεδομένων για την αποφυγή σφαλμάτων λόγο της ομοιότητας κάποιων γραμμάτων και αριθμών.

Το μοντέλο βαθειάς μάθησης MLP δεν έδωσε τα αναμενόμενα αποτελέσματα και είχε ποσοστό επιτυχίας μόλις 97,158% στα γράμματα και 98,349% στους αριθμούς. Αν και σαν ποσοστό επιτυχίας είναι πολύ καλό, δυστυχώς δεν ξεπέρασε τον μοντέλο SVM. Η διαφορά στο ποσοστό επιτυχίας ίσως να οφείλεται στην φύση του προβλήματος. Τα δίκτυα βαθειάς μάθησης γενικά δείχνουν να έχουν καλύτερη επιτυχία σε μεγάλα σύνολα δεδομένων ή σε σύνολα δεδομένων όπου οι κλασικές μέθοδοι δεν φέρουν καλά αποτελέσματα.

9.2 *Μελλοντικές επεκτάσεις και σχετικές εργασίες*

Μια ενδιαφέρουσα επέκταση όσο αφορά τον εντοπισμό της πινακίδας θα ήταν η χρήση μεθόδων μηχανικής μάθησης ή της μεθόδου κυλιόμενου παραθύρου (Anagnostopoulos, Anagnostopoulos, Tsekouras, *et al.*, 2005). Αυτές οι μέθοδοι δίνουν πολύ ικανοποιητικά αποτελέσματα αλλά είναι σχετικά αργές, τις τάξης μερικών δευτερολέπτων, σε αντίθεση με τον αλγόριθμο που υλοποιήθηκε στην παρούσα εργασία, όπου ο εντοπισμός (όπου είναι εφικτός) μαζί με την αναγνώριση των χαρακτήρων γίνεται σε περίπου 1 δευτερόλεπτο. Οι (Kim, Kim, Kim, *et al.*, 2000) χρησιμοποιούν ένα νευρωνικό δίκτυο TDNN (Time-Delay Neural Network) για τον εντοπισμό της πινακίδας που τους δίνει ποσοστό επιτυχίας 97,5% σε λιγότερο από ένα δευτερόλεπτο, αλλά όπως αναφέρουν υπάρχει *a priori* γνώση για την πιθανή τοποθεσία της πινακίδας.

Μια σύγχρονη μέθοδος βαθειάς μάθησης είναι τα δίκτυα CNN (Convolutional Neural Networks) (LeCun & Bengio, 1998) τα οποία επιτυγχάνουν πολύ καλά αποτελέσματα όταν πρόκειται για αναγνώριση αντικειμένων σε εικόνες. Το μειονέκτημα αυτών των δικτύων είναι η σχετικά αργή εκπαίδευση και πως απαιτούν πολύ μεγάλα σύνολα δεδομένων, της τάξης των χιλιάδων προτύπων για κάθε κλάση. Μια υλοποίηση με αυτά τα δίκτυα είναι των (Goodfellow, Bulatov, Ibarz, *et al.*, 2013) της Google. το δίκτυο χρησιμοποιήθηκε για τον εντοπισμό και την αναγνώριση αριθμών σε κατοικίες. Αναφέρουν μάλιστα πως όσο πιο βαθύ το δίκτυο, τόσο πιο καλά αποτελέσματα είχε. Η συγκεκριμένη υλοποίηση είχε 11 κρυφά επίπεδα.

Τέλος, μια ακόμα ενδιαφέρουσα επέκταση είναι η δημιουργία περισσότερων δεδομένων στην συλλογή από τα ήδη υπάρχουσα πρότυπα. Ένας τρόπος για να το πετύχουμε είναι απλά μετακινώντας τον κάθε χαρακτήρα κατά 1 ή 2 πίξελ προς όλες τις κατευθύνσεις. Διαφορετικά θα μπορούσαμε να περιστρέψουμε τον κάθε χαρακτήρα κατά ελάχιστες μοίρες, δημιουργώντας έτσι νέα πρότυπα. Μια διαφορετική προσέγγιση είναι των (Gregor, Danihelka, Graves, *et al.*, 2015) όπου εκπαιδεύοντας ένα δίκτυο RNN (Recurrent Neural Network) στο σύνολο δεδομένων MNIST, καταφέρνει το δίκτυο να παράγει χειρόγραφους αριθμούς που δεν ξεχωρίζουν από τους πραγματικούς.

10

Βιβλιογραφία

Anagnostopoulos, C., Anagnostopoulos, I., Tsekouras, G., Kouzas, G., et al. (2005) *Using sliding concentric windows for license plate segmentation and processing*. In: [Online]. 2005 IEEE. pp. 337–342. Available from: doi:10.1109/SIPS.2005.1579889 [Accessed: 25 June 2017].

Anagnostopoulos, C.N.E., Anagnostopoulos, I.E., Psoroulas, I.D., Loumos, V., et al. (2008) License Plate Recognition From Still Images and Video Sequences: A Survey. *IEEE Transactions on Intelligent Transportation Systems*. [Online] 9 (3), 377–391. Available from: doi:10.1109/TITS.2008.922938.

Anderson, E. (1936) The Species Problem in Iris. *Annals of the Missouri Botanical Garden*. [Online] 23 (3), 457. Available from: doi:10.2307/2394164.

Boser, B.E., Guyon, I.M. & Vapnik, V.N. (1992) *A training algorithm for optimal margin classifiers*. In: [Online]. 1992 ACM Press. pp. 144–152. Available from: doi:10.1145/130385.130401 [Accessed: 20 May 2017].

Bottou, L. (2010) Large-Scale Machine Learning with Stochastic Gradient Descent. In: Yves Lechevallier & Gilbert Saporta (eds.). *Proceedings of COMPSTAT'2010*. [Online]. Heidelberg, Physica-Verlag HD. pp. 177–186. Available from: doi:10.1007/978-3-7908-2604-3_16 [Accessed: 22 May 2017].

Cawley, G.C. & Talbot, N.L.C. (2010) On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. *Journal of Machine Learning Research*. 11 (Jul), 2079–2107.

Chang, Y., Hsu, H., Lee, J., Chueh, C., et al. (2013) *License Plate Character Recognition Using Block-Binary-Pixel-Sum Features*. In: [Online]. 2013 Atlantis Press. p. Available from: doi:10.2991/iccnce.2013.27 [Accessed: 27 May 2017].

Cortes, C. & Vapnik, V.N. (1995) *Support-Vector Networks*. 20 (3), 273–297.

Domingos, P. (2012) A few useful things to know about machine learning. *Communications of the ACM*. [Online] 55 (10), 78. Available from: doi:10.1145/2347736.2347755.

Fisher, R.A. (1936) THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. *Annals of Eugenics*. [Online] 7 (2), 179–188. Available from: doi:10.1111/j.1469-1809.1936.tb02137.x.

Goodfellow, I., Bengio, Y. & Courville, A. (2016) *Deep learning*. Adaptive computation and machine learning. Cambridge, Massachusetts, The MIT Press.

Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., et al. (2013) Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks. *arXiv:1312.6082 [cs]*. [Online] Available from: http://arxiv.org/abs/1312.6082 [Accessed: 25 June 2017].

Gregor, K., Danihelka, I., Graves, A., Rezende, D.J., et al. (2015) DRAW: A Recurrent Neural Network For Image Generation. *arXiv:1502.04623 [cs]*. [Online] Available from: http://arxiv.org/abs/1502.04623 [Accessed: 25 June 2017].

Hasan, Y.M.Y. & Karam, L.J. (2000) Morphological text extraction from images. *IEEE Transactions on Image Processing*. [Online] 9 (11), 1978–1983. Available from: doi:10.1109/83.877220.

Kim, K.K., Kim, K.I., Kim, J.B. & Kim, H.J. (2000) Learning-based approach for license plate recognition. In: *Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No.00TH8501)*. [Online]. 2000 pp. 614–623 vol.2. Available from: doi:10.1109/NNSP.2000.890140.

Knerr, S., Personnaz, L. & Dreyfus, G. (1990) Single-layer learning revisited: a stepwise procedure for building and training a neural network. In: Françoise Fogelman Soulié & Jeanny Hérault (eds.). *Neurocomputing*. [Online]. Berlin, Heidelberg, Springer Berlin Heidelberg. pp. 41–50. Available from: doi:10.1007/978-3-642-76153-9_5 [Accessed: 20 May 2017].

Kohavi, R. (1995) A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection. In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'95. [Online]. 1995 San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. pp. 1137–1143. Available from: http://dl.acm.org/citation.cfm?id=1643031.1643047 [Accessed: 3 June 2017].

Krstajic, D., Buturovic, L.J., Leahy, D.E. & Thomas, S. (2014) Cross-validation pitfalls when selecting and assessing regression and classification models. *Journal of Cheminformatics*. [Online] 6 (1), 10. Available from: doi:10.1186/1758-2946-6-10.

LeCun, Y. & Bengio, Y. (1998) *The Handbook of Brain Theory and Neural Networks*. In: Michael A. Arbib (ed.). [Online]. Cambridge, MA, USA, MIT Press. pp. 255–258. Available from: http://dl.acm.org/citation.cfm?id=303568.303704 [Accessed: 24 June 2017].

Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. (1998) Gradient-based learning applied to document recognition. *Proceedings of the IEEE*. [Online] 86 (11), 2278–2324. Available from: doi:10.1109/5.726791.

Ming-Kuei Hu (1962) Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*. [Online] 8 (2), 179–187. Available from: doi:10.1109/TIT.1962.1057692.

Pfaltz, J.L. & Rosenfeld, A. (1966) Sequential Operations in Digital Picture Processing. *Journal of the ACM*. [Online] 13 (4), 471–494. Available from: doi:10.1145/321356.321357.

Polyak, B.T. & Juditsky, A.B. (1992) Acceleration of Stochastic Approximation by Averaging. *SIAM Journal on Control and Optimization*. [Online] 30 (4), 838–855. Available from: doi:10.1137/0330046.

Rosenblatt, F. (1958) The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*. [Online] 65 (6), 386–408. Available from: doi:10.1037/h0042519.

Ruder, S. (2016) An overview of gradient descent optimization algorithms. *arXiv:1609.04747 [cs]*. [Online] Available from: http://arxiv.org/abs/1609.04747 [Accessed: 22 May 2017].

Silverman, B.W. & Jones, M.C. (1989) E. Fix and J.L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951). *International Statistical Review / Revue Internationale de Statistique*. [Online] 57 (3), 233. Available from: doi:10.2307/1403796.

Vapnik, V.N. & Lerner, A. (1963) *Generalized portrait method for pattern recognition*. 24 (6), 774–780.

Wilson, D.R. & Martinez, T.R. (2003) The general inefficiency of batch training for gradient descent learning. *Neural Networks*. [Online] 16 (10), 1429–1451. Available from: doi:10.1016/S0893-6080(03)00138-2.