Εντοπισμός Κρατήρων Στον Άρη Με Χρήση Νευρωνικού Δικτύου

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Ι. Διαμάντης

Εξεταστική Επιτροπή :
Καθ. Αργιαλάς Δημήτριος, Επιβλέπων
ΔΡ. ΕΥΔΙΠ Κολοκούσης Πολυχρόνης
Αν. Καθ. Καράντζαλος Κωνσταντίνος

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

This page has been intentionally left blank

Περιεχόμενα

Κατάλογος Σχημάτων

Crater Detection on Mars Using Artificial Neural Networks

Abstract

Η ανίχνευση και η καταγραφή των ουράνιων σωμάτων είναι ζωτικής σημασίας για τη μελέτη της δυναμικής ιστορίας του ηλιακού συστήματος. Οι πρόσφατες εξελίξεις στη τεχνολογία και η χρήση ψηφιακών νευρωνικών δικτύων μας δίνει τη δυνατότητα να εντοπίζουμε και να αναλύουμε γεωλογικά χαρακτηριστικά στα ουράνια σώματα με πολύ μεγαλύτερη ακρίβεια και ταχύτητα από ότι με τα συμβατικά μέσα. Από τους πιο σημαντικούς γεωλογικούς σχηματισμούς στην επιφάνεια ενός πλανήτη είναι οι κρατήρες. Η μελέτη των κρατήρων ήταν πάντα ένα από τα πιο καυτά θέματα στη γεωμορφολογία του διαστήματος. Οι πλανήτες καλύπτονται από δισεκατομμύρια κρατήρες μάρτυρες της διαστημικής τους γεωλογίας. Στόχος αυτής της εργασίας είναι η σχεδίαση και υλοποίηση ενός ψηφιακού νευρωνικού δικτύου με στόχο τον εντοπισμό κρατήρων στην επιφάνεια του πλανήτη Άρη. Το νευρωνικό δίκτυο έχει υλοποιηθεί με τη χρήση της σουίτας εργαλείων του MATLAB. Δέχεται σαν είσοδο ψηφιακές εικόνες εδάφους (DTM) τις οποίες αρχικά τις χρησιμοποιεί στη διαδικασία της εκπαίδευσης με αποτέλεσμα να δημιουργείται ένα πολύπλοκο μαθηματικό μοντέλο που στοχεύει στον εντοπισμό κρατήρων στις εικόνες εισόδου. Το δίκτυο που υλοποιήθηκε χρησιμοποίησε τον αλγόριθμο αντίστροφης μετάδοσης (backpropagation) για το στάδιο της εκπαίδευσης. Η αρχιτεκτονική του δικτύου περιείχε τρία επίπεδα το επίπεδο εισόδου, το επίπεδο εξόδου και ένα κρυφό επίπεδο. Δόθηκε βάση στο πρόβλημα της υπερπροσαρμογής και στις τεχνικές με τις οποίες ένα μοντέλο νευρωνικού δικτύου αποφεύγει την υπερπροσαρμογή. Οι τεχνικές αυτές αναφέρονται στους χωρικούς μετασχηματισμούς των εικόνων εισόδου αλλά και στη χρήση ενός σταδίου επαλήθευσης (validation). Με αυτούς τους τρόπους το δίκτυο εντόπισε 1004 από τους 1607 κρατήρες του dataset με ακρίβεια εντοπισμού 71%.

Λέξεις Κλειδιά : Ψηφιακά νευρωνικά δίκτυα, εντοπισμός κρατήρων, DTM, Άρης, αντίστροφη μετάδοση, MATLAB, κρυφό επίπεδο

Keywords : Artificial neural networks, crater detection, DTM, Mars, backprop-

agation, MATLAB, hidden layer

Κεφάλαιο 1

Εισαγωγή

Η ανίχνευση και η καταγραφή των ουράνιων σωμάτων είναι ζωτικής σημασίας για τη μελέτη της δυναμικής ιστορίας του ηλιακού συστήματος. Από την εφεύρεση του τηλεσκοπίου, αυτό συνέβαινε με οπτική επιθεώρηση του ουρανού μέσα από τους φακούς των τηλεσκοπίων. Φυσικά τα τελευταία χρόνια αυτό έχει αλλάξει και όλες οι νέες αστρονομικές παρατηρήσεις γίνονται μέσω ψηφιακών μέσων σε παρατηρητήρια με όργανα τελευταίας τεχνολογίας. Η οπτική επιθεώρηση των εικόνων, ωστόσο, περιορίζει το εύρος, την αποδοτικότητα ή την ακρίβεια της ανάκτησης σημαντικής πληροφορίας, καθώς η οπτική αντίληψη δεν ήταν ποτέ το μεγαλύτερο αγαθό της ανθρωπότητας. Επομένως, αυτό δημιουργεί την ανάγκη για έναν αποτελεσματικό και αξιόπιστο τρόπο ανίχνευσης των κρατήρων.

Η επιστημονική κοινότητα έχει αναπτύξει πολυάριθμους αυτόματους αλγόριθμους αναγνώρισης χαρακτηριστικών για να υπερβεί τους περιορισμούς της χειρωνακτικής εργασίας καθώς η καταγραφή γεωλογικών χαρακτηριστικών μπορεί πρακτικά να επιτευχθεί με οπτική επιθεώρησης εικόνων μόνο για περιορισμένο αριθμό χαρακτηριστικών. Μια από τις πλέον παγκοσμίως αποδεκτές μεθόδους αναγνώρισης δυνατοτήτων παρέχεται με τη μορφή *τεχνητών νευρωνικών δικτύων* (**Artificial Neural Networks ANN**), η οποία παρουσιάστηκε για πρώτη φορά το 1994 [1] και εξακολουθεί να είναι η πιο εξέχουσα μέθοδος για αξιόπιστη αναγνώριση.

1.1 Εισαγωγή στα Νευρωνικά Δίκτυα

Τα νευρωνικά δίκτυα (**ANN**) είναι μια υπολογιστική μέθοδος που ανήκει στη γενική αλγοριθμική κατηγορία της μηχανικής μάθησης (machine learning). Ένα ANN αποτελείται από ένα δίκτυο νευρώνων, ή κόμβων, που διασυνδέονται μεταξύ τους κατά τρόπο που μοιάζει με ένα βιολογικό νευρικό δίκτυο. Τα νευρωνικά δίκτυα είναι αλγόριθμοι **αυτόματης εκμάθησης** (self-learning) τα οποία **εκπαιδεύονται** (train) επεξεργάζοντας ερεθίσματα τα οποία περιέχουν γνώση του ερωτήματος και της απάντησης. Μετά από κάθε επεξεργασία νέου ερεθίσματος ένα ANN εξελίσσεται και γίνεται πιο ακριβές, ή "εξυπνότερο", διαδικασία η οποία ονομάζεται *τεχνητή νοημοσύνη* (Artificial Intelligence - AI).

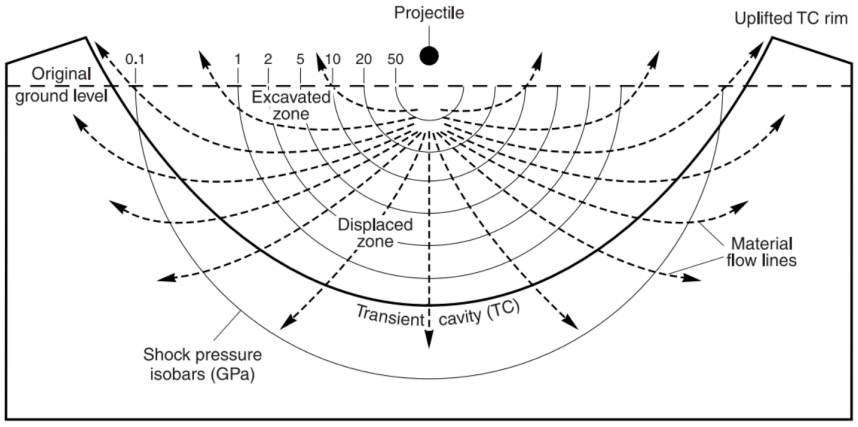
Ένα ANN σε μια συλλογή συνδεδεμένων μονάδων ή κόμβων που ονομάζονται τεχνητοί νευρώνες (Neurons), οι οποίοι προσομοιώνουν τους νευρώνες σε ένα βι-

ολογικό εγκέφαλο. Κάθε σύνδεση, όπως και οι συνάψεις σε ένα βιολογικό εγκέφαλο, μπορεί να μεταδώσει ένα σήμα σε άλλους νευρώνες. Ένας τεχνητός νευρώνας που λαμβάνει ένα σήμα κατόπιν το επεξεργάζεται και μπορεί να σηματοδοτήσει νευρώνες συνδεδεμένους σε αυτό. Το "σήμα" σε μια σύνδεση είναι ένας πραγματικός αριθμός και η έξοδος κάθε νευρώνα υπολογίζεται από κάποια μη γραμμική λειτουργία του αθροίσματος των εισόδων του. Αυτές οι συνδέσεις ονομάζονται άκρα (Edges). Οι νευρώνες και οι άκρες συνήθως έχουν ένα βάρος που προσαρμόζεται συνεχώς κατά τη διάρκεια της εκμάθησης. Το βάρος αυξάνει ή μειώνει την ισχύ του σήματος κατά τη σύνδεση. Οι νευρώνες μπορούν να έχουν ένα κατώφλι (Threshold) τέτοιο ώστε να αποστέλλεται σήμα μόνο εάν το συνολικό σήμα υπερβεί το όριο αυτό.

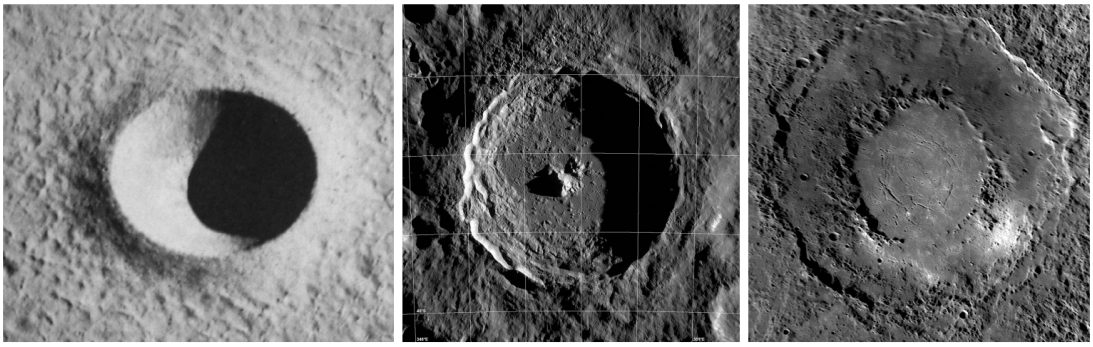
Σήμερα, τα τεχνητά νευρωνικά δίκτυα έχουν βρει εφαρμογές σε πολλούς κλάδους. Οι τομείς εφαρμογής περιλαμβάνουν την αναγνώριση και τον έλεγχο συστημάτων (έλεγχος οχημάτων, πρόβλεψη τροχιάς), χρηματοοικονομικά (όπως έξυπνα εργαλεία για αυτοματοποιημένα συστήματα διαπραγμάτευσης), και το σημαντικότερο, για το πεδίο εφαρμογής αυτού του έργου, την αναγνώριση προτύπων. Παρά τα σχεδόν 50 χρόνια έρευνας και ανάπτυξης σε αυτόν τον τομέα, το γενικό πρόβλημα της αναγνώρισης περίπλοκων προτύπων με αυθαίρετο προσανατολισμό, τοποθεσία και κλίμακα παραμένει ανοιχτό. Ωστόσο, τα νευρωνικά δίκτυα έχουν αποδεδειγμένη εγγραφή επιτυχημένων εφαρμογών για αναγνώριση μοτίβων σε εικόνες [2].

1.2 Μια Επισκόπηση των Κρατήρων

Οι κρατήρες είναι ένα από τα πιο ποικιλόμορφα γεωμορφολογικά τοπογραφικά στοιχεία στην επιφάνεια των πλανητών. Υπάρχουν διάφοροι τύποι κρατήρων ανάλογα με τον τρόπο που σχηματίζονται όπως κρουστικοί, ηφαιστειακοί κρατήρες ή κρατήρες καθίζησης. Το πρώτο βήμα του σχηματισμού κρατήρα είναι η σχεδόν στιγμιαία σύγκρουση μεταξύ της πλανητικής επιφάνειας και ενός άλλου συμπαγούς αντικειμένου. Αυτή η τεράστια ποσότητα κινητικής ενέργειας μεταφέρεται από το μετεωρίτη στην επιφάνεια προκαλώντας σημαντική κατακόρυφη συμπίεση (Σχήμα 1.1) Το συμπιεσμένο έδαφος εν μέρει χαλαρώνει εκπέμποντας μάζα στο περιβάλλον, προκαλώντας μερικές φορές άλλους δευτερεύοντες κρατήρες.



Σχήμα 1.1: Καμπύλες εξέλιξης σχηματισμού κρατήρα [3].



Σχήμα 1.2: Διάφοροι τύποι κρατήρων. Από αριστερά προς τα δεξιά: απλός κρατήρας, πολύπλοκος κρατήρας και γιγάντιος κρατήρας.

Το μέγεθος, η ταχύτητα και, λιγότερο, η τροχιά του μετεωρίτη επηρεάζουν την οπτική μορφή των κρατήρων. Το σχήμα πάντα μοιάζει με κύκλο αλλά η υφή μπορεί να είναι πολύ διαφορετική. (Σχήμα 1.2 [1] αριστερά). Οι κορυφές των κρατήρων σχηματίζονται από τα σωματίδια τα οποία εκσφενδονίστηκαν κατά τη σύγκρουση. Το τοπογραφικό προφίλ έχει σχεδόν ένα τετραγωνικό σχήμα. Όσο μεγαλύτερος είναι ο κρατήρας τόσο πιο ίσιο είναι το κάτω μέρος της κατάπτωσης. Όταν η μάζα του μετεωρίτη είναι σημαντικά μεγάλη ή όταν η ταχύτητά του είναι υψηλή, μπορεί να παρατηρηθεί εφέ αναπήδησης μετά τη φάση συμπίεσης, δημιουργώντας έναν πολύπλοκο κρατήρα (Σχήμα 1.2 κέντρο). Αυτό έχει ως αποτέλεσμα μια κορυφή στο κέντρο του κρατήρα. Τέλος, όταν ο κρατήρας είναι πολύ μεγάλος, οι σεισμικές επιδράσεις και η οριζόντια χαλάρωση του εδάφους μπορούν να σχηματίσουν πολλαπλούς δακτυλίους (Σχήμα 1.2 δεξιά).

Η τελευταία μορφομετρική παράμετρος που χρησιμοποιείται για την περιγραφή των χαρακτηριστικών ενός κρατήρα είναι ο λόγος βάθους/διαμέτρου. Η αναλογία αυτή παρέχει πληροφορίες σχετικά με το μήκος της σκιάς και την οξύτητα του κρατήρα. Ο λόγος αυτός συσχετίζεται με διαφορετικές παραμέτρους όπως το βαρυτικό πεδίο, η παρουσία διαβρωτικών παραγόντων και το μέγεθος του κρατήρα. Όσο μεγαλύτερος είναι ο κρατήρας, τόσο μικρότερος είναι ο λόγος βάθους/διαμέτρου και έχει επίδραση στην οπτική εικόνα του κρατήρα. Όλα αυτά τα στοιχεία είναι παράμετροι που πρέπει να αναλυθούν προκειμένου να δημιουργηθούν μοντέλα που να μπορούν να αναγνωρίσουν το αντικείμενο κρατήρα.

1.3 Η Σημασία της Ανίχνευσης Κρατήρων

Ο Άρης είναι ο τέταρτος σε απόσταση πλανήτης από τον Ήλιο και ο δεύτερος μικρότερος πλανήτης στο Ηλιακό Σύστημα, καθώς είναι μεγαλύτερος μόνο από τον Ερμή. Λόγω της θέσης του πλανήτη σε σχέση με τον Ήλιο, και των πρόσφατων ανακαλύψεων που δείχνουν μεγάλες λίμνες αλμυρού νερού κάτω από τους πόλους [4], ο Άρης γίνεται όλο και πιο πολύ ο καλύτερος υποψήφιος για τη διατήρηση της εξωγήινης ζωής στη γαλαξιακή γειτονιά της Γης. Επιπλέον, από την προσγείωση της Σελήνης το 1969, ο Άρης είναι το επόμενο σύνορο της ανθρωπότητας στην εξερεύνηση του διαστήματος,

[1] Προέλευση : lunarpedia.org, 2013, NASA/GSFC/Arizona State University, 2011 and impact-structure.com, Nov. 2016.

με τις περισσότερες εκτιμήσεις να δείχνουν ότι οι πρώτοι ανθρώπινοι αστροναύτες θα περπατήσουν στην επιφάνεια του Άρη μέχρι τα μέσα του αιώνα.

Το μεγαλύτερο μέρος της σημερινής μας γνώσης για τη γεωλογία του Άρη προέρχεται από τη μελέτη των χαρακτηριστικών εδάφους που φαίνονται σε εικόνες που τραβήχτηκαν από τους δορυφόρους που βρίσκονται σε τροχιά γύρω από τον πλανήτη. Ο Άρης διαθέτει έναν μεγάλο αριθμό διακριτών επιφανειακών χαρακτηριστικών που μαρτυρούν τους τύπους γεωλογικών διεργασιών που έχουν λειτουργήσει στον πλανήτη με την πάροδο του χρόνου. Το υποκεφάλαιο παρουσιάζει αρκετά από τα μεγαλύτερα φυσικά χαρακτηριστικά του Άρη. Όλες αυτές οι περιοχές δείχνουν πως οι γεωλογικές διαδικασίες όπως η ηφαιστειακή και η σεισμική δραστηριότητα, η ροή του νερού, και ο πάγος επηρέασαν το πως έχει διαμορφωθεί η συνολικά η επιφάνεια του πλανήτη.

Κρατήρες εντοπίστηκαν για πρώτη φορά στην επιφάνεια του Άρη από την αποστολή του Mariner 4 το 1965 [6]. Οι πρώτες παρατηρήσεις έδειξαν ότι οι αρειανοί κρατήρες ήταν γενικά πιο ρηχοί και ομαλότεροι από τους σεληνιακούς κρατήρες, υποδεικνύοντας ότι ο Άρης έχει πιο ενεργό ιστορικό διάβρωσης και εναπόθεσης από το Φεγγάρι [7]. Ο Άρης έχει τη μεγαλύτερη ποικιλομορφία στους τύπους κρατήρων από οποιουδήποτε άλλο πλανήτη στο Ηλιακό Σύστημα. Αυτό οφείλεται εν μέρει στο γεγονός ότι η παρουσία βραχωδών και πτητικών στρωμάτων στο υπόστρωμα παράγει ένα ευρύ φάσμα μορφολογιών ακόμη και μεταξύ των κρατήρων των ίδιων τάξεων μεγέθους. Ο Άρης έχει επίσης μια ατμόσφαιρα που παίζει σημαντικό ρόλο στην εκτίναξη των σωματιδίων και την επακόλουθη διάβρωση. Επιπλέον, ο Άρης έχει αρκετά χαμηλό ρυθμό ηφαιστειακής και τεκτονικής δραστηριότητας ώστε οι αρχαίοι και διαβρωμένοι πλέον κρατήρες να εξακολουθούν να διατηρούνται. Επιπροσθέτως η ηφαιστειακή και τεκτονική δραστηριότητα είναι αρκετά υψηλή ώστε να έχουν επανέλθει μεγάλες περιοχές του πλανήτη, δημιουργώντας ένα ευρύ φάσμα μορφολογιών με πολύ διαφορετικές ηλικίες. Περισσότεροι από 42.000 κρατήρες διαμέτρου άνω των 5 χιλιομέτρων έχουν καταγραφεί στον Άρη [8], και ο αριθμός των μικρότερων κρατήρων είναι πιθανότατα αμέτρητος.

Από το 1971, μετά την επιτυχή άφιξη του ρομποτικού οχήματος Sotravelner, άλλες τρεις αποστολές οχημάτων έχουν προσγειωθεί επιτυχώς στον κόκκινο πλανήτη και περισσότερες σχεδιάζονται για το άμεσο μέλλον. Συνεπώς, η γνώση του εδάφους του Άρη είναι κρίσιμη για επιτυχημένες μελλοντικές αποστολές. Η ανίχνευση κρατήρων με τη χρήση τεχνητών νευρωνικών δικτύων μπορεί να αποτελέσει ισχυρό εργαλείο για τη διασφάλιση της επιτυχίας των μελλοντικών εκτοξεύσεων.

1.4 Κίνητρο Εργασίας

Η μελέτη των κρατήρων ήταν πάντα ένα από τα πιο καυτά θέματα στη γεωμορφολογία του διαστήματος. Οι πλανήτες καλύπτονται από δισεκατομμύρια κρατήρες μάρτυρες της διαστημικής τους γεωλογίας. Εξετάζοντας την κατανομή μεγέθους/συχνότητας κρατήρα, παρέχει πληροφορίες σχετικά με την ηλικία της έκθεσης επιφανειών γεωλογικών σχηματισμών. Αυτός είναι ο μόνος τρόπος για να υπολογιστούν ηλικιακά χαρακτηριστικά μέσω της τηλεπισκόπησης

Επιπλέον, οι κρατήρες μπορούν να χρησιμοποιούνται για την αυτόματη πλοήγηση και προσγείωση των διαστημικών μονάδων. Σε αυτό το πλαίσιο, ο στόχος είναι είτε να αποφεύγονται οι κρατήρες και να εντοπίζονται ασφαλή σημεία είτε να αναγνωρίζονται οι γνωστοί κρατήρες για την πλοήγηση σε μια σχετικά ασφαλής θέση. Δεδομένου αυτών, ο εντοπισμός κρατήρα είναι ένα κρίσιμο μέρος για την αποφυγή κινδύνων σε μια διαστημική αποστολή.

Ωστόσο, οι κρατήρες αποτελούν ένα σύνθετο τοπογραφικό χαρακτηριστικό. Η διάμετρος τους κυμαίνεται από λίγα μέτρα έως εκατοντάδες χιλιόμετρα, πολλαπλοί κρατήρες μπορούν να επικαλύπτονται, η μορφολογία του εδάφους μπορεί να είναι διαφορετική, οι κορυφές τους μπορεί να έχουν πολλά επίπεδα διάβρωσης και οι συνθήκες φωτισμού ποικίλλουν ανάλογα με την θέση του πλανήτη σε σχέση με τον ήλιο και η γωνία θέασης τους. Επιπλέον, φυσικά το σχήμα αυτών τους μπορεί διαφέρει προσθέτοντας ένα έξτρα χαρακτηριστικό στη μορφολογία τους. Εν κατακλείδι, είναι δύσκολο να βρεθούν διακριτά χαρακτηριστικά που να μπορούν να ταξινομήσουν σωστά ένα αντικείμενο ως κρατήρα σε μια εικόνα και μέχρι σήμερα δεν υπάρχει ακόμη κάποια αποδεκτή πρότυπη λύση ή απάντηση σε αυτό το πρόβλημα.

Η δομή αυτής της διπλωματικής εργασίας έχει ως εξής. Το κεφάλαιο 2 είναι αφιερωμένο στην ανάλυση των τεχνητών νευρωνικών δικτύων. Παρουσιάζει την εννοιολογική ιδέα πίσω τους, στη συνέχεια επικεντρώνεται σε μια πιο εις βάθος εξέταση της μαθηματικής ανάλυσης που υποστηρίζει αυτούς τους αλγορίθμων και καταλήγει με ένα γνωστό παράδειγμα τεχνητών νευρωνικών δικτύων, τη δυαδική XOR. Στη συνέχεια, το κεφάλαιο 3 παρουσιάζεται το μέρος εφαρμογής της παρούσας εργασίας, και ορίζει όλα τα απαραίτητα βήματα που ακολουθούνται για την ανάπτυξη ενός αλγόριθμου ανίχνευσης με τη χρήση του MATLAB, μαζί με τα πειραματικά αποτελέσματα. Περιλαμβάνει επίσης μια μικρή παρουσίαση της σημερινής τεχνολογίας που βρίσκεται στη λογοτεχνία και παρουσιάζει τα εμπόδια που καθιστούν αυτό το είδος των αλγορίθμων απαιτητικό υπό διάφορες συνθήκες, ειδικά στην έλλειψη επαρκούς υπολογιστικής ισχύος. Τέλος, το τελευταίο κεφάλαιο είναι ο επίλογος και η σύνοψη αυτής της εργασίας.

Κεφάλαιο 2

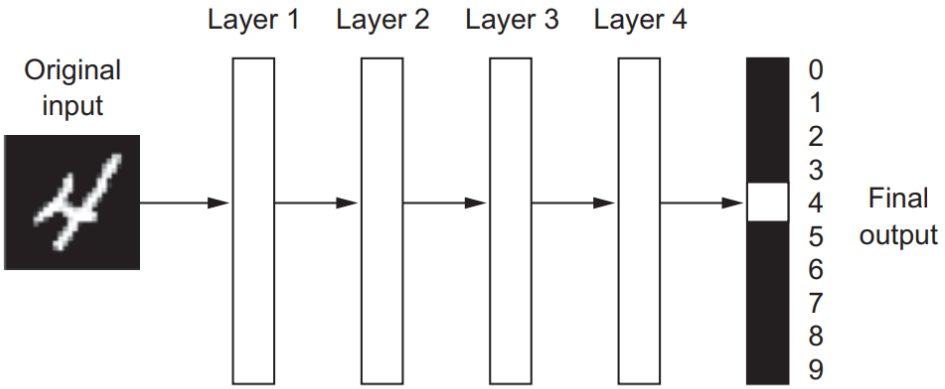
Νευρωνικά Δίκτυα

Το παρόν κεφάλαιο θα παρέχει την απαραίτητη γνώση και ορολογία γύρω από τα νευρωνικά δίκτυα και των αλγορίθμων βαθιάς μάθησης εν γένει (deep learning), που είναι απαραίτητα για την εφαρμογή των μεθόδων αυτών στους αλγόριθμους αναγνώρισης προτύπων που αναπτύσσονται στην παρούσα διατριβή με σκοπό την ανίχνευση κρατήρων.

2.1 Βαθιά Εκμάθηση

Η μηχανική εκμάθηση (Machine learning) αναφέρεται στην αναζήτηση χρήσιμων αναπαραστάσεων ορισμένων δεδομένων εισόδου, εντός ενός προκαθορισμένου χώρου δυνατοτήτων, χρησιμοποιώντας καθοδήγηση από ένα σήμα ανάδρασης. Η βαθιά εκμάθηση είναι ένα υποπεδίο της μηχανικής εκμάθησης: μια νέα αντίληψη για την αναπαράσταση δεδομένων που δίνει έμφαση στην εκμάθηση (training) διαδοχικών επιπέδων με στόχο ολοένα και πιο ουσιαστικών αναπαραστάσεων του αποτελέσματος. Η λέξη "βαθιά" της βαθιάς εκμάθηση δεν αποτελεί αναφορά σε οποιοδήποτε είδος βαθύτερης κατανόησης που επιτυγχάνεται με αυτή την προσέγγιση, αλλά το ότι διαθέτει την ιδέα των διαδοχικών επιπέδων, κάτι που θα εξηγηθεί στη πορεία αυτού του κεφαλαίου. Στη βαθιά εκμάθηση, αυτές οι πολυεπίπεδες αναπαραστάσεις (σχεδόν πάντα) εκπαιδεύονται μέσω μοντέλων που ονομάζονται νευρωνικά δίκτυα, δομές με κυριολεκτικά στρώματα τοποθετημένα το ένα πάνω στο άλλο.

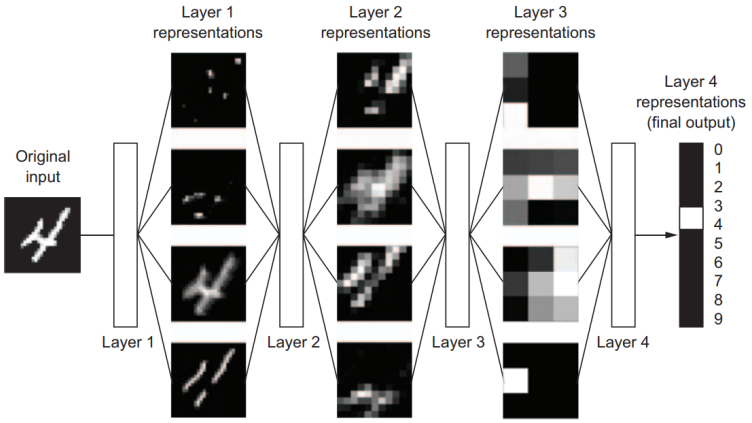
Τα τελευταία χρόνια, η μηχανική εκμάθηση έχει οδηγήσει σε προόδους σε πολλά διαφορετικά πεδία [9]. Αυτή η επιτυχία αποδίδεται σε ένα ευρύ φάσμα τεχνολογικών πλεονεκτημάτων, όπως η εφεύρεση πιο εξελιγμένων μοντέλων μηχανική εκμάθησης, η διαθεσιμότητα μεγάλων συνόλων δεδομένων για την αντιμετώπιση προβλημάτων με την εξελισσόμενη χρήση των έξυπνων αισθητήρων (smart sensors), η ανάπτυξη πλατφορμών λογισμικού που επιτρέπουν την εύκολη χρήση μεγάλων όγκων υπολογιστικών πόρων για εκπαίδευση, καθώς και η υψηλή αποτελεσματικότητα σε πολλούς τομείς, όπως τα χρηματοοικονομικά [10], η ιατρική [11], η ρομποτική [12], και λοιπά. Ωστόσο, η μεγαλύτερη ακρίβεια των νευρωνικών δικτύων είναι ένα πλεονέκτημα εις βάρος της υψηλής υπολογιστικής πολυπλοκότητας. Μέχρι σήμερα, τα υπολογιστικά συστήματα γενικού σκοπού, ειδικά οι μονάδες επεξεργασίας γραφικών (GPU), αποτελούν την κινητήρια δύναμη για μεγάλο μέρος της επεξεργασίας των νευρωνικών δικτύων. Ωστόσο, αναγνωρίζεται όλο και περισσότερο ότι απαιτείται όλο και πιο εξειδικευμένο υλικό (hardware) για τη συνεχή βελτίωση της απόδοσης των υπολογιστικών συστημάτων και της ενεργειακής απόδοσης [13].



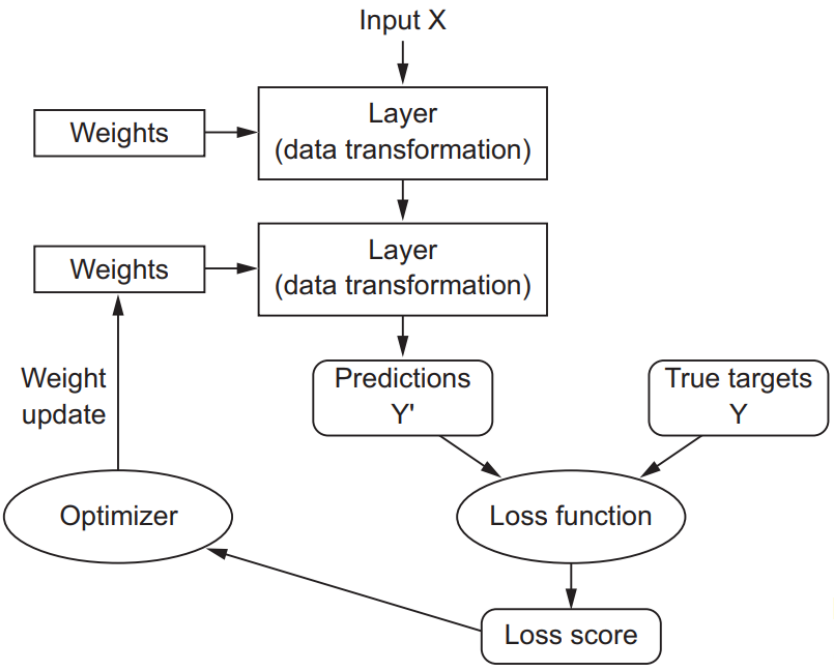
Σχήμα 2.1: Ένα βαθύ νευρωνικό δίκτυο για ταξινόμηση ψηφίων.

Ένα παράδειγμα νευρωνικού δικτύου μπορεί να είναι η κατανόηση ενός χειρόγραφου δεκαδικού ψηφίου που παρουσιάζεται ως εικόνα σε ένα τέτοιο σύστημα όπως αυτό που εμφανίζεται στη Σχήμα 2.1. Αυτό το νευρωνικό δίκτυο αποτελείται από μια εικόνα ως είσοδο, σε αυτή την περίπτωση μια εικόνα με τον αριθμό τέσσερα, έπειτα τέσσερα "κρυφά επίπεδα" (hidden layers) και ένα αποτέλεσμα. Ένα τέτοιο νευρωνικό σύστημα μπορεί να προγραμματιστεί κατά τρόπο ώστε ο υπολογιστής να μπορεί να αναγνωρίζει τα χαρακτηριστικά της αρχικής εικόνας ακόμη καλύτερα σε κάθε επίπεδο, καταφέρνοντας τελικά να ταξινομήσει την αρχική είσοδο ως τον αριθμό τέσσερα. Στο σχήμα 2.2 εμφανίζεται μια πιο λεπτομερής προοπτική. Το δίκτυο μετατρέπει την ψηφιακή εικόνα σε αναπαραστάσεις που διαφέρουν όλο και περισσότερο από την αρχική εικόνα και ολοένα και πιο ενημερωτικές για το τελικό αποτέλεσμα. Πρακτικά, ένα βαθύ δίκτυο είναι μια εργασία απόσταξης πληροφοριών πολλαπλών σταδίων, όπου οι πληροφορίες αυτές περνούν από διαδοχικά φίλτρα και καταλήγουν όλο και πιο καθαρά από υπολογιστική άποψη.

Η προδιαγραφή του τρόπου με τον οποίο ένα επίπεδο επεξεργάζεται τα δεδομένα εισόδου του αποθηκεύεται στο **βάρη** (weights) του επιπέδου, που είναι πρακτικά αριθμοί που προσπαθούν να προσαρμόσουν μια μαθηματική συνάρτηση στα δεδομένα εισόδου. Ένα από τα πιο σημαντικά μέρη ενός νευρικού δικτύου είναι να ποσοτικοποιηθεί η απόσταση του αποτελέσματος από το αναμενόμενο. Αυτή είναι η χρήση της συνάρτησης **loss function** που ονομάζεται επίσης επίσης συνάρτηση **αντικειμενικού σκοπού** (object function). Η λειτουργία της συνάρτησης loss λαμβάνει τις προβλέψεις του δικτύου και τα πραγματικά μετρηθέντα δεδομένα και υπολογίζει μια βαθμολογία απόστασης, καταγράφοντας πόσο καλά έχει προσαρμοστεί το δίκτυο στα δεδομένα εισόδου. Με απλά λόγια, η λειτουργία της loss function είναι ο τρόπος με τον οποίο το δίκτυο θα είναι σε θέση να μετρήσει την απόδοσή του στα δεδομένα εκμάθησης και επομένως πώς θα μπορέσει να κινηθεί προς τη σωστή κατεύθυνση. Η βασική λειτουργία της βαθιάς εκμάθησης είναι να χρησιμοποιείται αυτή η βαθμολογία ως σήμα ανάδρασης για την προσαρμογή της τιμής των βαρών προς την κατεύθυνση που μειώνει τη βαθμολογία loss σε μια ελάχιστη τιμή (Σχήμα 2.3). Αυτή η ανάδραση είναι η **βελτιστοποίηση** (optimizer), η οποία υλοποιεί έναν αλγόριθμο **back-propagation**. Πρακτικά, η βελτιστοποίηση είναι ένας μηχανισμός μέσω του οποίου



Σχήμα 2.2: Διαδοχικές αναπαραστάσεις που έχουν διδαχθεί από ένα μοντέλο ταξινόμησης ψηφίων.



Σχήμα 2.3: Ένα αφαιρετικό διάγραμμα του νευρικού δικτύου, η συνάρτηση απώλειας και ο μηχανισμός ανάδρασης.

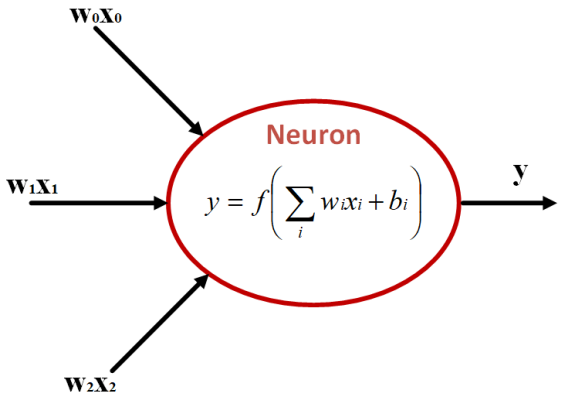
το δίκτυο θα ενημερώνεται με βάση τα δεδομένα που βλέπει και τη λειτουργία του loss function.

Το πλεονέκτημα ενός αποτελεσματικού αλγόριθμου μηχανικής εκμάθησης είναι σαφές. Αντί για την κοπιαστική και μη αποτελεσματική προσέγγιση της δημιουργίας ενός ξεχωριστού, προσαρμοσμένου προγράμματος για την επίλυση κάθε μεμονωμένου προβλήματος σε έναν τομέα, ένας αλγόριθμος εκμάθησης ενός υπολογιστή απλά χρειάζεται να εκπαιδευτεί μια μόνο φορά, μέσω μιας διαδικασίας που ονομάζεται **εκπαίδευση** (training), για να αντιμετωπίσει κάθε νέο πρόβλημα.

2.2 Ο Νευρώνας

Η κατανόηση όλων των παραπάνω μπορεί να ρίξει λίγο φως στη βασική μονάδα βαθιάς μάθησης στα νευρωνικά δίκτυα, τον **νευρώνα** (the neuron). Ο νευρώνας είναι μια υπολογιστική μονάδα που παίρνει τις εισόδους, κάνει ορισμένους υπολογισμούς και παράγει την έξοδο. Έτσι βασικά ένας νευρώνας παίρνει την εισροή εφαρμόζει μερικές παραμέτρους (τα βάρη που αναφέρθηκαν προηγουμένως) και παράγει κάποιο αποτέλεσμα. Στη συνέχεια, τα βάρη ρυθμίζονται από το μηχανισμό μετάδοσης back-propagation, έτσι ώστε να ελαχιστοποιείται το σφάλμα. Ένα απλοποιημένο παράδειγμα ενός νευρώνα εμφανίζεται στο σχήμα 2.4. Αυτή η εικόνα δείχνει ότι ο νευρώνας είναι απλά μια μαθηματική συνάρτηση $f(\cdot)$ με αριθμό εισόδων i και ένα εξερχόμενο y. Οι είσοδοι του νευρώνα αποτελούνται από δύο παράγοντες, την είσοδο x_i (π.χ. ψηφιακή εικόνα) και το βάρος w_i. Έτσι, διαφορετικά βάρη οδηγούν σε διαφορετικές λύσεις για μια είσοδο. Η βασική πτυχή της εκμάθησης μπορεί να θεωρηθεί ως η προσαρμογή των βαρών ως απάντηση σε ένα μαθησιακό ερέθισμα [14].

Η παράμετρος b_i ονομάζεται **πόλωση** (bias) και επιτρέπει τη μετακίνηση του εύρους εξόδου κατά μια σταθερή τιμή, η οποία μπορεί να φανεί χρήσιμη για το φιλτράρισμα των αποτελεσμάτων της **συνάρτηση ενεργοποίησης** (activation function) σε ορισμένα προβλήματα. Αυτοί οι παράγοντες πόλωσης συνήθως εκπαιδεύονται μαζί με τα βάρη, αλλά ορισμένα πρόσφατα δίκτυα [15] καταργούν εντελώς τη χρήση τους.



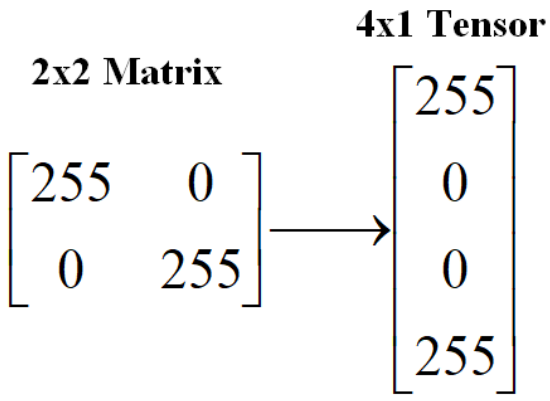
Σχήμα 2.4: Οι συνδέσεις ενός νευρώνα. $x_i, w_i, f(\cdot)$ και b είναι οι είσοδοι, τα βάρη, η μη γραμμική συνάρτηση και η πόλωση αντίστοιχα.

Στις εφαρμογές νευρωνικών δικτύων, τα δεδομένα που χρησιμοποιούνται ως είσοδοι αποθηκεύονται σε πολυδιάστατες συστοιχίες που ονομάζονται **τένσορες** (tensor). Γενικά, όλα τα σύγχρονα συστήματα μηχανικής εκμάθησης χρησιμοποιούν τους τένσορες ως βασική δομή δεδομένων. Ο τένσορας είναι πρακτικά μια δομή δεδομένων η οποία στην πράξη έχει τη μορφή μήτρας. Ένας τένσορας που περιέχει μόνο έναν αριθμό ονομάζεται **scalar**, ένας πίνακας αριθμών ονομάζεται **διανυσμάτων** (vector), ή 1-D τένσορας, ένας πίνακας διανυσμάτων είναι μήτρα ή ένας τένσορας 2-D και λοιπά. Σε νευρωνικά δίκτυα βαθιάς εκμάθησης, όλοι οι μετασχηματισμοί μπορούν να μετατραπούν σε πράξεις τενσόρων που εφαρμόζονται στα αριθμητικά δεδομένα. Για παράδειγμα, είναι δυνατή η προσθήκη, ο πολλαπλασιασμός τενσόρων και λοιπά.

Πολλές εφαρμογές βαθιάς εκμάθησης χρησιμοποιούν ψηφιακές εικόνες ως είσοδοι όπου στην επιστήμη υπολογιστών μια ψηφιακή εικόνα απεικονίζεται από έναν πίνακα με διαστάσεις $N \times M$ και αριθμούς που κυμαίνονται από 0 έως 255 και αντιπροσωπεύουν διαφορετικές αποχρώσεις του γκρι (σε ασπρόμαυρες εικόνες). Στην υλοποίηση των περισσότερων νευρωνικών δικτύων, αυτή η εικόνα μετατρέπεται πρώτα σε τένσορας 1-D διαστάσεων $(N \times M) \times 1$ και στη συνέχεια τροφοδοτείται στο νευρωνικό δίκτυο ως είσοδος (Σχήμα. 2.5).

2.3 Βελτιστοποίηση Βάσει Διαβάθμισης

Η μαθηματική έκφραση που θα ενεργούσε ως συνάρτηση μεταφοράς ενός νευρωνικού δικτύου είναι:



Σχήμα 2.5: Ένας πίνακας διαστάσεων 2×2 μετασχηματισμένος σε έναν τένσορα 4.

$$Y = \widehat{W} \cdot X + \widehat{b} \tag{2.1}$$

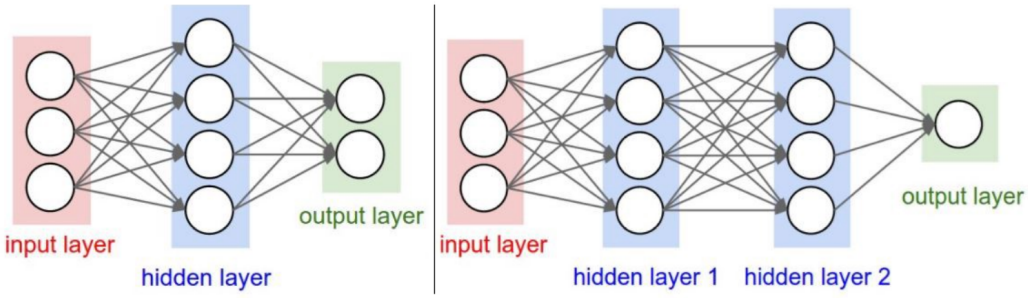
Σε αυτή την παράσταση, τα W και b είναι οι παράγοντες που αποτελούν χαρακτηριστικά του κρυφού επιπέδου (layer). Ονομάζονται "βάρη" ή "παράμετροι συναρμογής" του επιπέδου. Τα βάρη αυτά περιέχουν τις πληροφορίες που αντλεί το δίκτυο. Αρχικά, οι πίνακες βάρους γεμίζουν με τυχαίες τιμές και η παράσταση (2.1) δεν παρέχει χρήσιμες αναπαραστάσεις, αλλά αυτό είναι ένα μόνο το σημείο έναρξης του αλγορίθμου. Η χρήση οποιωνδήποτε τιμών για τα βάρη δημιουργεί εξόδους με πεδίο τιμών που κυμαίνονται από μείων άπειρο έως συν άπειρο. Στις περισσότερες εφαρμογές, το εύρος των εξόδων πρέπει να είναι περιορισμένο και για αυτό το λόγο τα νευρωνικά δίκτυα χρησιμοποιούν μια συνάρτηση ενεργοποίησης (activation).

Η συνάρτηση ενεργοποίησης περιορίζει το εύρος των εξόδων και παράγει τιμές εντός ενός επιθυμητού εύρους που βασίζεται στην επιλογή της κατάλληλης συνάρτησης ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής νευρωνικού δικτύου. Οι πιο συχνές συναρτήσεις ενεργοποίησης παρατίθενται στον παρακάτω πίνακα.

	Activation Function
Sigmoid	$y_s = \frac{1}{1+e^{-x_s}}$
Tanh	$y_s = tanh(x_s)$
ReLu	$y_s = max(0, x_s)$

Έτσι, ένα νευρικό δίκτυο είναι ένα σύνολο επιπέδων (ένα επίπεδο έχει ένα σύνολο νευρώνων) στοιβαγμένα διαδοχικά το ένα μετά το άλλο και η έξοδος ενός επιπέδου θα ήταν η είσοδος του επόμενου όπως φαίνεται στο Σχήμα 2.6. Αυτή η εικόνα απεικονίζει δύο διαφορετικά νευρωνικά δίκτυα. Το πρώτο αποτελείται από ένα **επίπεδο εισόδου** (input layer) με τρεις νευρώνες, όπου κάθε νευρώνας αντιπροσωπεύει κάποιο χαρακτηριστικό του συνόλου δεδομένων εισόδου, ένα **κρυφό επίπεδο** (hidden layer), το οποίο είναι ένα σύνολο από νευρώνες όπου σε κάθε νευρώνα αντιστοιχιστεί μια παράμετρος βάρους. Κάθε κρυφό επίπεδο παίρνει την είσοδο από το προηγούμενο επίπεδο, εκτελεί το εσωτερικό γινόμενο των εισόδων και των βαρών, εφαρμόζει τη

συνάρτηση ενεργοποίησης, παράγει ένα αποτέλεσμα και περνάει τα δεδομένα στο επόμενο επίπεδο. Τέλος, το **επίπεδο εξόδου** (output layer) είναι το ίδιο με το κρυφό επίπεδο, με τη διαφορά ότι δίνει το τελικό αποτέλεσμα. Το δεύτερο παράδειγμα νευρικού δικτύου έχει δύο κρυφά επίπεδα που αυξάνουν την πολυπλοκότητα του δικτύου και είναι κατάλληλα για πιο απαιτητικές εργασίες. Ο αριθμός των κρυφών επιπέδων μπορεί να είναι οποιοσδήποτε αριθμός που αυξάνει το βάθος της πολυπλοκότητας του συστήματος για να παρέχει πιο ακριβή αποτελέσματα με κόστος βέβαια του υπολογιστικού χρόνου.



Σχήμα 2.6: **Αριστερά** : Ένα νευρωνικό δίκτυο δύο επιπέδων, ένα κρυφό επίπεδο με τέσσερις νευρώνες και ένα επίπεδο εξόδου με δύο νευρώνες. **Δεξιά** : Ένα νευρωνικό δίκτυο τριών επιπέδων δύο κρυφά επίπεδα με τέσσερις νευρώνες και ένα επίπεδο εξόδου με έναν νευρώνα.

2.4 Επιλέγοντας τον Αριθμό των Νευρώνων

Ένα ερώτημα που προκύπτει από αυτή τη συζήτηση είναι πώς ορίζετε ο αριθμός των νευρώνων σε κάθε επίπεδο και τελικά σε ολόκληρο το δίκτυο. Δυστυχώς δεν υπάρχει οριστική και τεκμηριωμένη απάντηση σε αυτήν την ερώτηση [16]. Υπάρχουν βέβαια διάφορες προσεγγίσεις για να τον υπολογισμό τον αριθμό των νευρώνων σε ένα κρυφό επίπεδο.

Η **ευθεία προσέγγιση** (forward approach) είναι και επαναλαμβανόμενη μέθοδος δοκιμής και σφάλματος (trial-and-error) που ξεκινά επιλέγοντας έναν μικρό αριθμό κρυφών νευρώνων, εκπαιδεύοντας το δίκτυο, και κατόπιν αυξάνοντας τον αριθμό των νευρώνων. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να βελτιωθεί η εκπαίδευση του δικτύου. Έπειτα, υπάρχει η **Προσέγγιση προς τα πίσω** (Backward Approach), όπου είναι το αντίθετο της προσέγγισης προς τα εμπρός. Σε αυτή τη προσέγγιση, το νευρωνικό δίκτυο ξεκινά με έναν μεγάλο αριθμό κρυφών νευρώνων. Στη συνέχεια, το δίκτυο εκπαιδεύεται και βαθμιαία μειώνει τον αριθμό των νευρώνων μέχρι να βελτιωθεί το αποτέλεσμα της εκπαίδευση δοκιμάζοντας μερικά παραδείγματα για να φανεί η αποτελεσματικότητα της εκπαίδευσης.

Υπάρχουν επίσης γενικοί ευριστικοί κανόνες (heuristics) που δηλώνουν ότι ο αριθμός των νευρώνων θα πρέπει να κυμαίνεται μεταξύ του μεγέθους του επιπέδου εισόδου και του μεγέθους του επιπέδου εξόδου ή ότι ο αριθμός των κρυφών νευρώνων θα πρέπει να είναι τα δύο τρίτα του μεγέθους του επιπέδου εισόδου συν το μέγεθος του νευρώνα εξόδου ή τέλος, υπάρχουν ορισμένες μελέτες που εισάγουν πιο εξελιγμένες μεθόδους για την επιλογή του κατάλληλου αριθμού νευρώνων ο οποίος μεταβάλλεται

ανάλογα με το εκάστοτε πρόβλημα [17].

2.5 Backpropagation - Μαθηματική Ανάλυση

Ο όρος **αντίστροφη μετάδοση** (Backpropagation) [20] αναφέρεται σε έναν ευρέως χρησιμοποιούμενο αλγόριθμο για την εκπαίδευση νευρωνικών δικτύων. Κατά την εκπαίδευση ενός νευρωνικού δικτύου, η αντίστροφη μετάδοση υπολογίζει την διαβάθμιση (gradient) της συνάρτησης απώλειας σε σχέση με τα βάρη του δικτύου για ένα μόνο παράδειγμα εισόδου-εξόδου, και το κάνει αποτελεσματικά, σε αντίθεση με μια αφελή άμεση προσέγγιση υπολογισμού της διαβάθμισης σε σχέση με κάθε βάρος ξεχωριστά. Η αποτελεσματικότητα αυτή καθιστά εφικτή τη χρήση μεθόδων διαβάθμισης για την εκπαίδευση δικτύων πολλαπλών επιπέδων, την ενημέρωση των βαρών για την ελαχιστοποίηση της απώλειας, οι πιο συνηθισμένοι μέθοδοι backpropagation είναι οι **gradient descent** και η παραλλαγή αυτού, ο στοχαστικός **gradient descent**.

Η συνάρτηση κόστους ή ζημίας (cost/loss function) αποτελεί έναν σημαντικό παράγοντα οποίος θα πρέπει να συνοψίζει δίκαια όλες τις πτυχές ενός σύνθετου μοντέλου σε έναν μόνο αριθμό, κατά τέτοιο τρόπο ώστε οι βελτιώσεις σε αυτόν τον αριθμό να αποτελούν ένδειξη ενός καλύτερου μοντέλου [18]. Υπάρχουν πολλές συναρτήσεις που θα μπορούσαν να χρησιμοποιηθούν για την εκτίμηση του σφάλματος ενός συνόλου βαρών σε ένα νευρωνικό δίκτυο, όπως η **εκτίμηση μέγιστης πιθανότητας** (Maximum likelihood estimation), η **διασταυρούμενη εντροπία** (cross-entropy) που αναφέρεται επίσης ως **Λογαριθμική απώλεια** (Logarithmic loss) [19] ή το μέσο τετραγωνικό σφάλμα (MSE). Στα προβλήματα παλινδρόμησης (regression), η πιο διαδεδομένη συνάρτηση σφάλματος είναι η MSE, η οποία υπολογίζεται ως ο μέσος όρος των τετραγωνικών διαφορών μεταξύ των προβλεπόμενων και των πραγματικών τιμών:

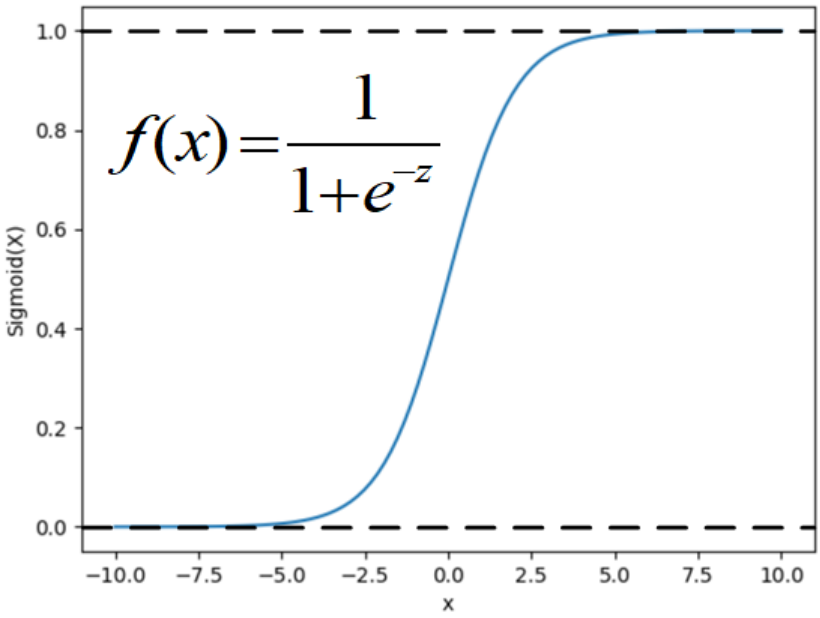
$$E = \frac{1}{n}\sum_{1}^{n}(t_i - y_i)^2 \tag{2.2}$$

Στην παραπάνω εξίσωση, το E είναι το MSE, T_i είναι η τιμή-στόχος (target) και Y_i αντιπροσωπεύει την προβλεπόμενη τιμή ή έξοδο ενός νευρώνα. Το αποτέλεσμα της MSE είναι θετικό ανεξάρτητα από το σημείο των προβλεπόμενων και πραγματικών τιμών και η τιμή του ιδανικού αποτελέσματος είναι 0.

Για κάθε νευρώνα j, η έξοδος o_j ορίζεται ως :

$$o_j = f(net_j) = f\left(\sum_{k=1}^{n} w_{kj}o_k\right) \tag{2.3}$$

όπου η συνάρτηση ενεργοποίησης f (δείτε τον πίνακα παραπάνω) είναι μη γραμμική και διαφοροποιήσιμη. Η συνηθέστερη συνάρτηση ενεργοποίησης είναι η **σιγμοειδής** (sigmoid logistic function) με συνάρτηση μεταφοράς τη καμπύλη που εμφανίζεται στο Σχήμα. 2.7. Από τη γραφική παράσταση φαίνεται ότι η σιγμοειδής συνάρτηση παράγει έναν αριθμό μεταξύ 0 και 1 ανεξάρτητα από την είσοδο.



Σχήμα 2.7: Γραφική παράσταση της σιγμοειδής (Sigmoid) συνάρτησης.

Η είσοδος net_j σε ένα νευρώνα είναι το σταθμισμένο άθροισμα (weighted sum) των εξόδων o_k των προηγούμενων νευρώνων. Εάν ο νευρώνας βρίσκεται στο πρώτο επίπεδο μετά το επίπεδο εισόδου, το o_k του επιπέδου εισόδου είναι απλώς τα δεδομένα εισόδου x_k στο δίκτυο. Ο αριθμός των μονάδων εισόδου στο νευρώνα είναι n. Η μεταβλητή w_{kj} δηλώνει το βάρος μεταξύ του νευρώνα k του προηγούμενου επιπέδου και του νευρώνα j του τρέχοντος επιπέδου.

Ο Υπολογισμός της μερικής παραγώγου του σφάλματος σε σχέση με το βάρος w_{ij} γίνεται εφαρμόζοντας τον κανόνα αλυσίδας δύο φορές:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} \tag{2.4}$$

Στον τελευταίο παράγοντα, στη δεξιά πλευρά της παραπάνω σχέσης, μόνο ένας όρος στο άθροισμα net_j εξαρτάται από w_{ij} ώστε:

$$\frac{\partial net_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^{n} w_{kj} o_k \right) = \frac{\partial}{\partial w_{ij}} w_{ij} o_i = o_i \tag{2.5}$$

Εάν ο νευρώνας βρίσκεται στο πρώτο επίπεδο μετά το επίπεδο εισόδου, τα o_i είναι απλώς x_i. Η παράγωγος της εξόδου του νευρώνα j σε σχέση με την είσοδο του είναι απλώς η μερική παράγωγος της συνάρτησης ενεργοποίησης:

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial f(net_j)}{\partial net_j} \tag{2.6}$$

Σε αυτή τη περίπτωση όπου η συνάρτηση μεταφοράς είναι η σιγμοειδής, η παράγωγος

υπολογίζεται ως :

$$\frac{df(z)}{dz} = f(z) \cdot (1 - f(z)) \tag{2.7}$$

έτσι η εξίσωση 2.6 μετατρέπεται ως εξής :

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial}{\partial net_j} f(net_j) = f(net_j)(1 - f(net_j)) = o_j(1 - o_j) \tag{2.8}$$

Αυτός είναι και ο λόγος γιατί η διαδικασία του back-propagation απαιτεί η συνάρτηση ενεργοποίησης να είναι διαφοροποιήσιμη. Εάν ο νευρώνας βρίσκεται στο επίπεδο εξόδου τότε ο πρώτος παράγοντας υπολογίζεται εύκολα καθώς $o_j = y$ και

$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} \tag{2.9}$$

και αν το μισό του τετραγωνικού σφάλματος χρησιμοποιείται ως συνάρτηση σφάλματος, τότε η παραπάνω εξίσωση ξαναγράφεται ως:

$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y} \frac{1}{2}(t - y)^2 = y - t \tag{2.10}$$

Ωστόσο, εάν το j βρίσκεται σε ένα αυθαίρετο εσωτερικό επίπεδο του δικτύου, η εύρεση της παραγώγου E όσον αφορά τα o_j είναι λιγότερο προφανής. Λαμβάνοντας υπόψη το E ως συνάρτηση με τα δεδομένα εισόδου να είναι όλοι οι νευρώνες $L = \{u, v, .., w\}$ που λαμβάνουν δεδομένα εισόδου από το νευρώνα j :

$$\frac{\partial E(o_j)}{\partial o_j} = \frac{\partial E(net_u, net_v, ...net_w)}{\partial o_j} \tag{2.11}$$

και υπολογίζοντας τη συνολική παράγωγο σε σχέση με o_j, λαμβάνεται η **αναδρομική** (recursive) έκφραση για την παράγωγο:

$$\frac{\partial E}{\partial o_j} = \sum_{\ell \in L} \left(\frac{\partial E}{\partial net_\ell} \cdot \frac{\partial net_\ell}{\partial o_\ell} \right) = \sum_{\ell \in L} \left(\frac{\partial E}{\partial o_\ell} \cdot \frac{\partial o_\ell}{\partial net_\ell} \cdot \frac{\partial net_\ell}{\partial o_j} \right) = \sum_{\ell \in L} \left(\frac{\partial E}{\partial o_\ell} \cdot \frac{\partial o_\ell}{\partial net_\ell} \cdot w_{j\ell} \right) \tag{2.12}$$

Επομένως, η παράγωγος ως προς το o_j μπορεί να υπολογιστεί εάν είναι γνωστές όλες οι παράγωγοι των εξόδων o_ℓ του επόμενου επιπέδου (του πλησιέστερου στο επίπεδο εξόδου). Μια σημαντική σημείωση είναι ότι εάν οποιοσδήποτε από τους νευρώνες στο σύνολο L δεν ήταν συνδεδεμένος με το νευρώνα j, θα ήταν ανεξάρτητος από τα w_{ij} και η αντίστοιχη μερική παράγωγος στο πλαίσιο της άθροισης θα υπολογίζονταν 0.

Αντικαθιστώντας τα παραπάνω αποτελέσματα στην εξίσωση 2.4 :

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot \frac{\partial net_j}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \cdot o_i = o_i \delta_j \tag{2.13}$$

με,

$$\delta_j = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \begin{cases} \frac{\partial L(o_j,t)}{\partial o_j} \cdot \frac{df(net_j)}{dnet_j} & j: \textit{ νευρώνας εξόδου} \\ (\sum_{\ell \in L} w_{j\ell}\delta_\ell) \cdot \frac{df(net_j)}{dnet_j} & j: \textit{ εσωτερικός νευρώνας} \end{cases}$$

εάν το $f(\cdot)$ είναι η σιγμοειδής συνάρτηση και το σφάλμα είναι το τετράγωνο σφάλμα :

$$\delta_j = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial net_j} \begin{cases} (o_j - t_j) \cdot o_j \cdot (1 - o_j) & j: \textit{ νευρώνας εξόδου} \\ (\sum_{\ell \in L} w_{j\ell}\delta_\ell) \cdot o_j \cdot (1 - o_j) & j: \textit{ εσωτερικός νευρώνας} \end{cases}$$

Η τιμή του βάρος w_{ij} στη μέθοδο backpropagation ενημερώνεται χρησιμοποιώντας κλίση διαβάθμισης (gradient descent), άλλος ένας σημαντικός παράγοντας είναι η επιλογή του ρυθμού εκμάθησης, $\eta > 0$. Η αλλαγή βάρους πρέπει να ανταναλκλά την επίπτωση στο σφάλμα E για μια αύξηση ή μείωση στην τιμή w_{ij}. Εάν $\frac{\partial E}{\partial w_{ij}} > 0$ μια αύξηση στο βάρος w_{ij} αυξάνει το E. Αντίστροφως, εάν $\frac{\partial E}{\partial w_{ij}} < 0$, τότε η αύξηση στο βάρος w_{ij} μειώνει το E. Το νέο Δw_{ij} προστίθεται στο παλιό βάρος και το γινόμενο του ρυθμού εκμάθησης η και της παραγώγου, πολλαπλασιασμένο με το *-1* εγγυάται ότι τα w_{ij} αλλάζουν με τρόπο που μειώνει πάντα το E. Με άλλα λόγια, στην επόμενη εξίσωση το $-\eta\frac{\partial E}{\partial w_{ij}}$ πάντα μεταβάλλει το w_{ij} με τέτοιο τρόπο ώστε το σφάλμα E μειώνεται:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta o_i \delta_j \tag{2.14}$$

Οι μέθοδοι Gradient descent μαζί με το backpropagation δεν εγγυάται την εύρεση του ολικού ελάχιστου της συνάρτησης σφάλματος αλλά μόνο ένα τοπικό ελάχιστο. Επιπροσθέτως, αυτή η μέθοδος δυσκολεύεται να ξεπεράσει κάποια όρια (plateaus) στο πεδίο τιμών της συνάρτησης σφάλματος. Το ζήτημα αυτό, το οποίο προκλήθηκε από τη μη κυρτότητα των συναρτήσεων σφάλματος στα νευρωνικά δίκτυα, θεωρούντο για πολύ καιρό ως σημαντικό μειονέκτημα, αλλά στη δημοσίευσή του ο Yann LeCun υποστηρίζει ότι σε πολλά πρακτικά προβλήματα αυτή η θεώρηση δεν ισχύει [20].

2.6 Η Δυαδική XOR

Ένα από τα πιο κλασικά παραδείγματα στην έρευνα τεχνητών νευρικών δικτύων είναι η δυαδική πύλη ή exclusive or (XOR). Φυσικά, δεν υπάρχει πραγματική ανάγκη για ένα νευρωνικό δίκτυο να επιλύσει την εξίσωση της XOR, αλλά μπορεί να χρησιμοποιηθεί για την απλοποίηση των δυνατοτήτων πρόβλεψης των νευρωνικών δικτύων. Μια πύλη XOR θα πρέπει να επιστρέφει μια θετική τιμή (true), όταν οι δύο είσοδοι δεν είναι ίσες και ψευδείς (false) εάν είναι ίσες. Όλες οι πιθανές είσοδοι και οι προβλεπόμενες έξοδοι παρουσιάζονται στον επόμενο πίνακα:

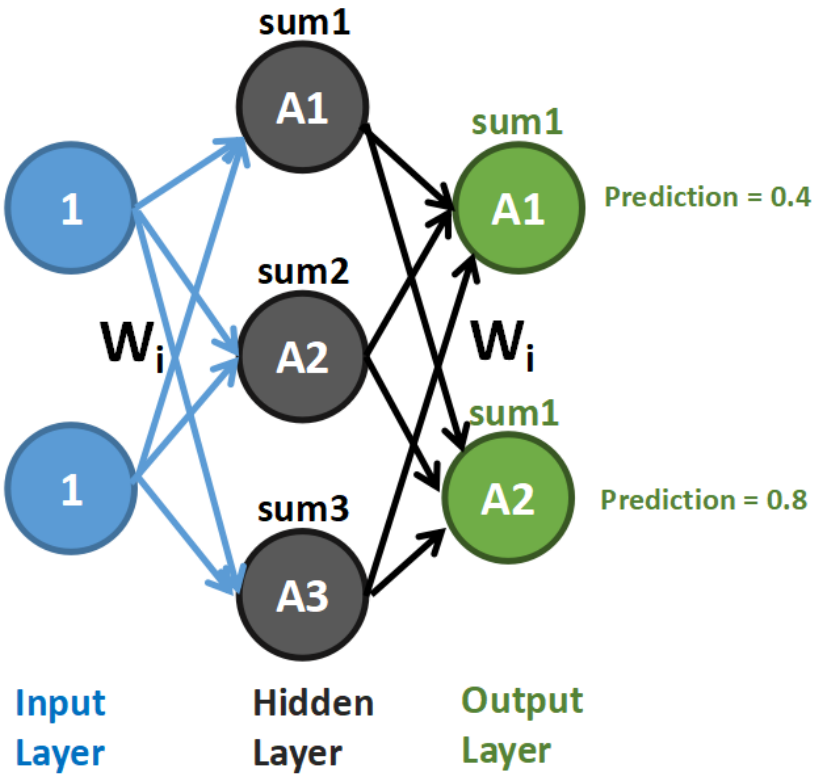
Είσοδος1	Είσοδος2	Έξοδος
0	0	0
0	1	1
1	0	1
1	1	0

Η XOR είναι ένα πρόβλημα **ταξινόμησης** (classification problem) κατά το οποίο οι αναμενόμενες έξοδοι είναι γνωστές εκ των προτέρων. Η αρχιτεκτονική του νευρωνικού δικτύου για την επίλυση του προβλήματος XOR χρησιμοποιεί **forward propagation**. Σε αυτή την απλή διαδικασία, οι είσοδοι του αλγορίθμου τροφοδοτούνται στο επόμενο επίπεδο του δικτύου, ενώ οι έξοδοι του προηγούμενου επιπέδου γίνονται οι είσοδοι στο επόμενο επίπεδο. Στην αρχή ο αλγόριθμος αποδίδει τυχαία βάρη και μια συνάρτηση ενεργοποίησης, σε αυτή την περίπτωση μια σιγμοειδής συνάρτηση, υπολογίζει τα σφάλματα για κάθε νευρώνα.

Κάθε νευρώνας εκτελεί την επόμενη διεργασία η οποία φαίνεται και στο σχήμα 2.8 :

$$sum_i = inputs * W_i \tag{2.15}$$

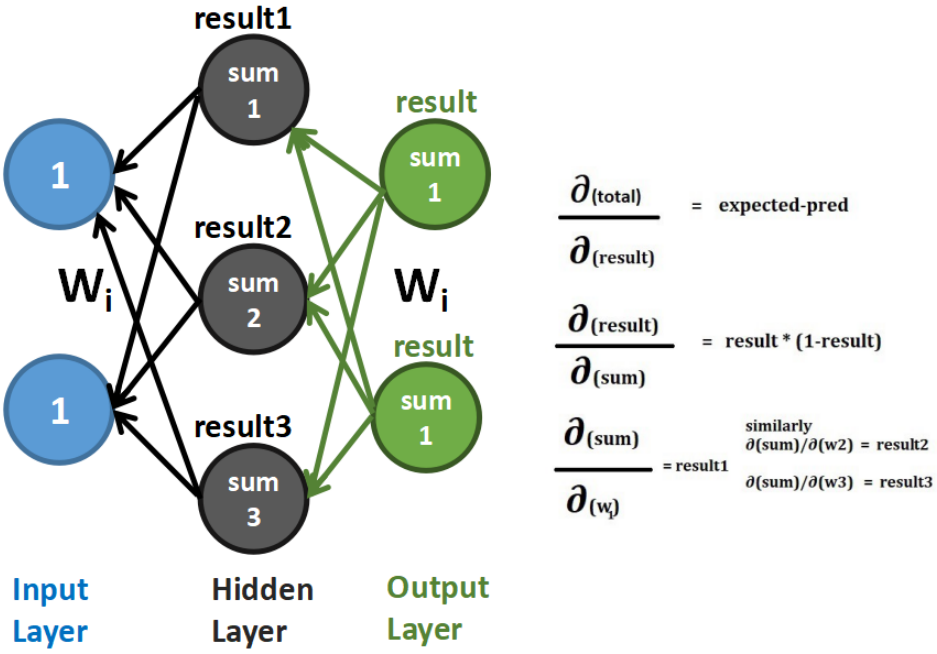
$$A_i = \frac{1}{1 + e^{-sum}} \tag{2.16}$$



Σχήμα 2.8: Παράδειγμα της τεχνικής forward propagation στο πρόβλημα της πύλης XOR.

Στη συνέχεια, το αποτέλεσμα της συνάρτησης ενεργοποίησης στο επίπεδο εξόδου υπαγορεύει την πρόβλεψη του νευρωνικού δικτύου στη πρώτη **εποχή** (epoch) ή την πρώτη επανάληψη. Το επόμενο βήμα σε ένα νευρωνικό δίκτυο είναι η μετάδοση προς τα πίσω (back-propagation). Ο κύριος στόχος του back-propagation είναι η ενημέρωση κάθε ενός από τα βάρη του δικτύου έτσι ώστε το προβλεπόμενο αποτέλεσμα να είναι πιο κοντά στο επιδιωκόμενο αποτέλεσμα, ελαχιστοποιώντας έτσι το σφάλμα για κάθε νευρώνα εξόδου και το δίκτυο συνολικά. Τα επόμενα βήματα είναι ο υπολογισμός του βαθμού στον οποίο το συνολικό σφάλμα μεταβάλλεται σε σχέση με το αποτέλεσμα, του βαθμού στον οποίο το αποτέλεσμα αλλάζει σε σχέση με το άθροισμα και, τέλος, του βαθμού στον οποίο το άθροισμα αλλάζει σε σχέση με τα βάρη.

Οι έννοιες αυτές εκφράζονται σε μαθηματική μορφή στο Σχήμα 2.9, καθώς και ο ίδιος ο μηχανισμός back-propagation. Τέλος, το νευρωνικό δίκτυο επαναλαμβάνει τη διαδικασία προώθησης και ενημέρωσης των βαρών για έναν προκαθορισμένο αριθμό εποχών (ή επαναλήψεων) ή μέχρι να ελαχιστοποιηθεί το σφάλμα. Μόλις ολοκληρωθεί η διαδικασία εκπαίδευσης, το νευρωνικό δίκτυο μπορεί να εκτελέσει προβλέψεις προωθώντας την είσοδο στο εκπαιδευμένο δίκτυο και παίρνοντας το αποτέλεσμα στο επίπεδο εξόδου σύμφωνα με τη συνάρτηση μεταφοράς (Εξίσωση 2.1) που ορίστηκε αρχικά στο κεφάλαιο. Γενικά όσο μεγαλύτερος είναι ο αριθμός των εποχών τοσο μεγαλύτερη είναι και η αποτελεσματικότητα των προβλέψεων του νευρωνικού δικτύου.



Σχήμα 2.9: Παράδειγμα μετάδοσης προς τα πίσω (back-propagation) στο πρόβλημα της XOR και οι μαθηματικές εκφράσεις που χρησιμοποιούνται για τη βαθμονόμηση των βαρών.

2.7 Βασικές Μετρήκες και Στόχοι

2.7.1 Ακρίβεια

Η ακρίβεια χρησιμοποιείται για να υποδείξει την ποιότητα του αποτελέσματος μιας συγκεκριμένης εφαρμογής. Το γεγονός ότι τα νευρωνικά δίκτυα μπορούν να επιτύχουν εξαιρετική ακρίβεια σε ένα ευρύ φάσμα καθηκόντων είναι ένας από τους βασικούς λόγους που οδηγούν τη δημοτικότητα και την ευρεία χρήση τους τη σύγχρονη εποχή. Οι μονάδες που χρησιμοποιούνται για τη μέτρηση της ακρίβειας εξαρτώνται από την εφαρμογή. Για παράδειγμα, για την ταξινόμηση εικόνων, η ακρίβεια αναφέρεται ως το ποσοστό των σωστά ταξινομημένων εικόνων, ενώ για την ανίχνευση αντικειμένων, η ακρίβεια αναφέρεται ως η μέση ακρίβεια (mean average precision), η οποία σχετίζεται με το trade-off μεταξύ του πραγματικού θετικού ποσοστού και του ψευδώς θετικού ποσοστού. Οι παράγοντες που επηρεάζουν την ακρίβεια περιλαμβάνουν τη δυσκολία της εργασίας και του συνόλου δεδομένων.

Η επίτευξη υψηλής ακρίβειας σε δύσκολες εφαρμογές ή μεγάλα σύνολα δεδομένων συνήθως απαιτεί πιο σύνθετα μοντέλα νευρωνικών δικτύων (περισσότερα κρυφά επίπεδα επομένως μεγαλύτερος αριθμός υπολογισμών) τα οποία μπορούν να επηρεάσουν την αποτελεσματικότητα του μοντέλου από δεδομένου ενός hardware. Η ακρίβεια θα πρέπει επομένως να ερμηνεύεται στο πλαίσιο της δυσκολίας της εργασίας και του συνόλου δεδομένων.

2.7.2 Όγκος Αποτελεσμάτων και Ταχύτητα

Ο όγκος αποτελεσμάτων (Throughput) χρησιμοποιείται για να υποδείξει την ποσότητα των δεδομένων που μπορούν να υποβληθούν σε επεξεργασία ή τον αριθμό εκτελέσεων μιας εργασίας που μπορεί να ολοκληρωθεί σε μια δεδομένη χρονική περίοδο. Η επεξεργασία μεγάλου όγκου δεδομένων είναι συχνά κρίσιμη για μια εφαρμογή. Για παράδειγμα, η επεξεργασία βίντεο με ταχύτητα 30 καρέ το δευτερόλεπτο είναι απαραίτητη για την παροχή απόδοσης σε πραγματικό χρόνο. Για την ανάλυση δεδομένων, η υψηλή απόδοση σημαίνει ότι περισσότερα δεδομένα μπορούν να αναλυθούν σε ένα δεδομένο χρονικό διάστημα. Καθώς η ποσότητα των οπτικών δεδομένων αυξάνεται εκθετικά, η υψηλή ικανότητα επεξεργασίας των μεγάλων αναλύσεων αποκτά ολοένα και μεγαλύτερη σημασία, ιδίως εάν κάποιο σύστημα πρέπει να εκτελέσει κάποια δράση με βάση εικόνων υψηλής ανάλυσης. Η ταχύτητα αναφέρεται συχνά ως ο αριθμός των πράξεων ανά δευτερόλεπτο. Στην περίπτωση που το σύστημα αποδίδει συμπερασμάτων, η διεκπεραιωτική ικανότητα αναφέρεται ως τα συμπεράσματα ανά δευτερόλεπτο ή με τη μορφή χρόνου εκτέλεσης σε δευτερόλεπτα ανά συμπέρασμα.

Η ταχύτητα θεωρείται το χρονικό διάστημα μεταξύ της άφιξης των δεδομένων εισόδου σε ένα σύστημα και της δημιουργίας του αποτελέσματος. Ο μικρός χρόνος αναμονής είναι απαραίτητος για αλληλεπιδραστικές εφαρμογές σε πραγματικό χρόνο, όπως η προσαυξημένη πραγματικότητα (augmented reality), η αυτόνομη πλοήγηση και η ρομποτική. Η ταχύτητα αναφέρεται συνήθως σε δευτερόλεπτα.

2.7.3 Ενεργειακή Απόδοση και Κατανάλωση Ενέργειας

Η μετρική ενεργειακή απόδοση χρησιμοποιείται για να υποδείξει την ποσότητα των δεδομένων που μπορούν να υποβληθούν σε επεξεργασία ή τον αριθμό των εκτελέσεων μιας εργασίας που μπορεί να ολοκληρωθεί για μια συγκεκριμένη μονάδα ενέργειας. Η υψηλή ενεργειακή απόδοση είναι σημαντική όταν ένα νευρωνικό δίκτυο εκτελείται σε ασύρματες συσκευές με περιορισμένη χωρητικότητα μπαταρίας. Η εκτέλεση των υπολογισμών σε ασύρματες συσκευές μπορεί να προτιμάται έναντι του υπολογισμού στο νέφος (cloud) για ορισμένες εφαρμογές λόγω περιορισμών χρόνου αναμονής, ασφάλειας ή περιορισμένου εύρους ζώνης επικοινωνίας. Η ενεργειακή απόδοση αναφέρεται συχνά γενικά ως ο αριθμός των πράξεων ανά joule.

Η κατανάλωση ενέργειας χρησιμοποιείται για να υποδείξει την ποσότητα ενέργειας που καταναλώνεται ανά χρόνο μονάδας. Η αυξημένη κατανάλωση ενέργειας έχει ως αποτέλεσμα την αυξημένη παραγωγή θερμότητας κατά συνέπεια, η μέγιστη κατανάλωση ενέργειας υπαγορεύεται από ένα κριτήριο σχεδιασμού που συνήθως ονομάζεται θερμική ισχύς σχεδιασμού (thermal design power, TDP), δηλαδή την ισχύ για την οποία έχει σχεδιαστεί το σύστημα ψύξης. Η κατανάλωση ενέργειας είναι σημαντική ακόμα και κατά την επεξεργασία των νευρωνικών δικτύων στο νέφος, καθώς τα κέντρα δεδομένων (data centers) έχουν αυστηρά ανώτατα όρια ισχύος λόγω του κόστους ψύξης. Παρομοίως, οι συσκευές χειρός (smartphones) και οι φορητές συσκευές (απομακρυσμένοι αισθητήρες) έχουν επίσης αυστηρούς περιορισμούς ισχύος, καθώς ο χρήστης είναι συχνά αρκετά ευαίσθητος στη θερμότητα και τα μεγέθη τέτοιων συσκευών συχνά περιορίζει τους μηχανισμούς ψύξης. Η κατανάλωση ισχύος αναφέρεται συνήθως σε watt ή joule ανά δευτερόλεπτο.

2.7.4 Ευελιξία

Ένα μεγάλο πλεονέκτημα των νευρωνικών δικτύων είναι και η ευελιξία τους. Η ευελιξία αναφέρεται στο εύρος των μοντέλων νευρωνικών δικτύων που μπορούν να υποστηριχθούν από τον επεξεργαστή και στη δυνατότητα του περιβάλλοντος λογισμικού (π.χ. Matlab, Python κ.λπ.) να αξιοποιούν στο μέγιστο τις δυνατότητες του hardware για οποιοδήποτε επιθυμητό μοντέλο νευρωνικού δικτύου. Δεδομένου του ταχέως εξελισσόμενου ρυθμού έρευνας και ανάπτυξης των νευρωνικών δικτύων, είναι ολοένα και πιο σημαντικό οι επεξεργαστές να υποστηρίζουν ένα ευρύ φάσμα μοντέλων και εργασιών.

2.7.5 Κλιμάκωση

Η δυνατότητα κλιμάκωσης έχει γίνει ολοένα και πιο σημαντική λόγω των πολλών εφαρμογών νευρωνικών δικτύων και των συνεχώς νέων αναδυόμενων τεχνολογιών που χρησιμοποιούνται για την κλιμάκωση όχι μόνο του μεγέθους των ολοκληρωμένων κυκλωμάτων (chip), αλλά και για την κατασκευή συστημάτων με πολλαπλά chips. Η δυνατότητα κλιμάκωσης αναφέρεται στο πόσο καλά μπορεί να κλιμακωθεί ένας σχεδιασμός για να επιτευχθεί μεγαλύτερη απόδοση και εξοικονόμηση ενέργειας κατά την αύξηση της ποσότητας των πόρων (π.χ. μεγαλύτερη επεξεργαστική ισχύ). Η αξιολόγηση αυτή γίνεται με την υπόθεση ότι το σύστημα δεν χρειάζεται να επανασχεδιαστεί σημαντικά, δεδομένου ότι οι μεγάλες αλλαγές σχεδιασμού μπορεί να είναι δαπανηρές από άποψη χρόνου και κόστους. Ιδανικά, μια κλιμακούμενη σχεδίαση μπορεί

να χρησιμοποιηθεί για ασύρματες συσκευές χαμηλού κόστους και συσκευές υψηλής απόδοσης στο cloud απλώς με την κλιμάκωση των πόρων χωρίς επιπλέον αλλαγές στην αρχιτεκτονική τους.

2.7.6 Αλληλεπίδραση Μεταξύ των Μετρικών

Είναι σημαντικό να λαμβάνονται υπόψη όλες οι μετρικές προκειμένου να αξιολογούνται δίκαια όλα τα trade-offs κατά το σχεδιασμού των νευρωνικών δικτύων. Για παράδειγμα, χωρίς την ακρίβεια που έχει δοθεί για ένα συγκεκριμένο σύνολο δεδομένων και μια συγκεκριμένη εργασία, θα μπορούσε κανείς να σχεδιάσει ένα απλό νευρωνικό δίκτυο και να διεκδικήσει εύκολα χαμηλή ισχύ, υψηλή απόδοση και χαμηλό κόστος, ωστόσο ο επεξεργαστής ενδέχεται να μην μπορεί να χρησιμοποιηθεί για μια πρακτική εφαρμογή. Εναλλακτικά, χωρίς την αναφορά του εύρους ζώνης εκτός chip, θα μπορούσε κανείς να κατασκευάσει έναν επεξεργαστή με μόνο πολλαπλασιαστές και να διεκδικήσει εύκολα χαμηλό κόστος, υψηλή απόδοση, υψηλή ακρίβεια και χαμηλή ισχύ chip. Δυστυχώς, κατά την αξιολόγηση της ισχύος του συστήματος, η πρόσβαση στη μνήμη που βρίσκεται εκτός του chip θα ήταν σημαντική. Τέλος, πρέπει επίσης να αναφέρεται η διάταξη της δοκιμής (test setup), συμπεριλαμβανομένου του εάν τα αποτελέσματα μετρώνται ή λαμβάνονται από την προσομοίωση και του αριθμού των εικόνων που ελέγχθηκαν.

Σε σύνοψη, η διαδικασία αξιολόγησης για το αν ένα σύστημα νευρωνικού δικτύου αποτελεί βιώσιμη λύση για μια συγκεκριμένη εφαρμογή εξετάζεται ως εξής:

- η ακρίβεια καθορίζει εάν μπορεί να εκτελέσει τη συγκεκριμένη εργασία,
- ο χρόνος αναμονής και η ταχύτητα καθορίζουν εάν μπορούν να λειτουργούν αρκετά γρήγορα και σε πραγματικό χρόνο,
- η κατανάλωση ενέργειας και ισχύος είναι μεγάλος περιορισμός σε ασύρματα συστήματα
- το κόστος, το οποίο εξαρτάται κυρίως από το μέγεθος των chip και το εύρος ζώνης της εξωτερικής μνήμης, καθορίζει το ποσό που θα πλήρωνε κάποιος για τη λύση αυτή,
- η ευελιξία καθορίζει το εύρος των εργασιών που μπορεί να υποστηρίξει
- η δυνατότητα κλιμάκωσης καθορίζει αν είναι δυνατή η χρήση της ίδιας υλοποίησης, χωρίς τον επανασχεδιασμό του δικτύου, σε πολλούς τομείς και εάν το σύστημα μπορεί να κλιμακωθεί αποτελεσματικά με τη χρήση περισσότερων ή λιγότερων υπολογιστικών πόρων.

2.7.7 Μετρικές Ακρίβειας

Η αξιολόγηση ενός αλγορίθμου μηχανικής εκμάθησης αποτελεί κρίσιμο μέρος κάθε εφαρμογής. Τις περισσότερες φορές χρησιμοποιείται η ακρίβεια ταξινόμησης (classification accuracy) για τη μέτρηση της απόδοση του μοντέλου, ωστόσο δεν αρκεί για έναν αντικειμενικό χαρακτηρισμό του μοντέλου. Ο ορισμός αυτής της μετρικής είναι ο εξής :

$$Accuracy = \frac{Correct\,Predictions}{Total\,Number\,of\,Predictions} \tag{2.17}$$

Πρόκειται για το λόγο των σωστών προβλέψεων ως προς το συνολικό αριθμό προβλέψεων. η μετρική αυτή λειτουργεί καλά μόνο εάν υπάρχει ίσος αριθμός δειγμάτων που ανήκουν σε κάθε κλάση. Για παράδειγμα εάν ένα δίκτυο προσπαθεί να εντοπίσει αν σε μια εικόνα εμφανίζεται σκύλος ή γάτα και το 98% των εικόνων περιέχουν έναν σκύλο και το υπόλοιπο 2% περιέχει γάτα τότε το μοντέλο θα έχει ακρίβεια 98% εάν θεωρήσει ότι όλες οι εικόνες παρουσιάζουν έναν σκύλο. Εάν τώρα το 60% των εικόνων περιέχουν σκύλους και το 40% γάτες τότε η ακρίβεια θα πέσει στο 60%. Η ακρίβεια ταξινόμισης λοιπόν δίνει μια ψευδή αίσθηση της επίτευξης υψηλής ακρίβειας.

Επιπροσθέτως, σε πιο δύσκολα προβλήματα αυτός ο ορισμός δεν δίνει πληροφορίες για τυχόν ψευδώς θετικά (ή αρνητικά) αποτελέσματα. Για αυτό το σκοπό χρησιμοποιούνται άλλες δύο έννοια για την ακρίβεια που ονομάζονται precision και recall. Ο ορισμός precision (P) ορίζεται ως ο λόγος των αληθώς θετικών προβλέψεων ως προς τον συνολικό αριθμό προβλέψεων ενώ ο ορισμός του recall (R) είναι ο λόγος των αληθώς θετικών ως προς το συνολικό αριθμό δειγμάτων. Οι μαθηματικοί ορισμοί για αυτές τις δύο έννοιες είναι οι εξής :

$$P = \frac{T_P}{T_P + F_P} \tag{2.18}$$

$$R = \frac{T_P}{T_P + F_N} \tag{2.19}$$

όπου T_P, F_P και F_N είναι οι αριθμοί των αληθώς θετικών, ψευδώς θετικών και ψευδώς αρνητικών αντίστοιχα. Ένα νευρωνικό δίκτυο με υψηλή ακρίβεια P σημαίνει ότι οι περισσότερες προβλέψεις είναι επιτυχημένες. Αντίστοιχα ένα μεγάλο ποσοστό του R υποδηλώνει ότι το δίκτυο είναι αποτελεσματικό στο να εντοπίζει τα απαιτούμενα στοιχεία στις εισόδους που του προσφέρονται. Στην εφαρμογή του εντοπισμού κρατήρων για παράδειγμα, εάν ένα δίκτυο εντοπίσει κρατήρες σε όλη την επιφάνεια του Άρη είτε υπάρχουν είτε όχι τότε θα έχει μεγάλο ποσοστό R καθώς οι περισσότεροι από τους πραγματικούς κρατήρες θα εντοπιστούν. Ταυτόχρονα βέβαια, το ίδιο δίκτυο θα έχει πολύ κακή ακρίβεια P καθώς θα έχει παράξει πολλούς ψευδώς θετικούς κρατήρες. Αντιθέτως, εάν το δίκτυο εντοπίσει μόνο έναν κρατήρα επιτυχημένα τότε θα έχει τέλεια ακρίβεια ($P = 1$) αλλά πολύ κακό R καθώς θα απουσιάζουν πολλοί κρατήρες ή θα υπάρχουν πολλοί ψευδώς αρνητικοί εντοπισμοί.

Έτσι αυτοί οι δύο ορισμοί δεν μπορούν να χαρακτηρίσουν επαρκώς την απόδοση ενός δικτύου. Μία ακόμη μετρική που συνδυάζει τις δύο προηγούμενες μετρικές είναι ο αρμονικός μέσος όρος (harmonic average) που αποκαλείται F_1 (F_1 *score*) και ορίζεται ως :

$$F_1 = \frac{2PR}{P + R} \tag{2.20}$$

Το πεδίο ορισμού της μετρικής F_1 είναι το [0,1]. Η τιμή αυτή υποδηλώνει όχι μόνο

πόσο ακριβής είναι η πρόβλεψη (πόσες περιπτώσεις προβλέπει σωστά) αλλά και πόσο σταθερές είναι οι προβλέψεις (δηλαδή το ότι δεν αποτυγχάνει να προβλέψει σημαντικό αριθμό παρουσιών). Υψηλές τιμές της μετρικής P αλλά χαμηλές τιμές R δίνει εσφαλμένη αίσθηση επιτυχίας και έτσι η χρήση της μετρικής F_1 αποδίδει πιο αντικειμενικά αποτελέσματα όπου όσο μεγαλύτερη είναι η τιμή του τόσο καλύτερη η απόδοση του μοντέλου.

Κεφάλαιο 3

Νευρωνικό Δίκτυο Ανίχνευσης Κρατήρων

Το κεφάλαιο αυτό είναι αφιερωμένο στην ανάλυση ενός τεχνητού νευρωνικού δικτύου με στόχο την αυτόματη ανίχνευση των κρατήρων σε εικόνες της επιφάνειας του Άρη. Αρχικά, θα παρουσιαστεί μια ανασκόπηση των πιο πρόσφατων δημοσιεύσεων (State-of-the-Art) μαζί με τη γενική ιδέα για την εφαρμογή η οποία εμπνέεται από παρόμοιες εργασίες στη βιβλιογραφία. Το νευρωνικό δίκτυο σχεδιάζεται και υλοποιείται με τη χρήση του MATLAB. Στη συνέχεια θα αναφερθούν μερικές τεχνικές για τη βελτιστοποίηση του χρόνου εκτέλεσης και τέλος θα παρουσιαστούν τα πειραματικά αποτελέσματα.

3.1 State-of-the-Art

Μία από τις πιο ευθείες προσεγγίσεις για το πρόβλημα ανίχνευσης κρατήρων ονομάζεται μη επιβλεπόμενη μέθοδος μάθησης (unsupervised learning method). Ο στόχος των μεθόδων εκμάθησης χωρίς επίβλεψη είναι να χρησιμοποιούνται δομές δεδομένων από μη επισημασμένα δεδομένα (unlabeled data). Οι μέθοδοι που έχουν αναπτυχθεί περιλαμβάνουν συχνά μια διαδικασία βελτίωσης των κορυφών του κρατήρα και στη συνέχεια ανακατασκευάζουν τυπικές γεωμετρίες χρησιμοποιώντας το μετασχηματισμό Hough [21].

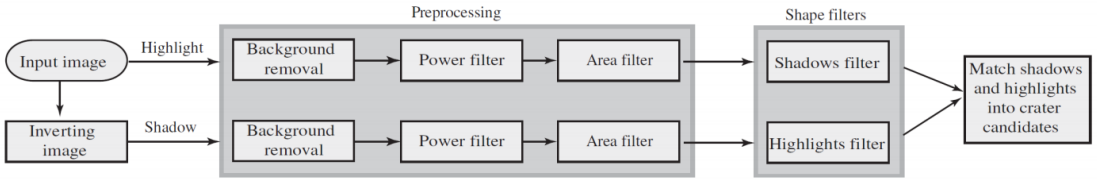
Ο μετασχηματισμός Hough είναι μια τεχνική εξαγωγής χαρακτηριστικών με σκοπό την εύρεση ατελειών των αντικειμένων της εικόνας εντός μιας συγκεκριμένης κατηγορίας σχημάτων μέσω μιας διαδικασίας ψηφοφορίας (voting procedure). Αυτή η διαδικασία ψηφοφορίας διεξάγεται σε έναν χώρο παραμέτρων (parameter space), από τον οποίο οι υποψήφια αντικείμενα λαμβάνονται ως τοπικά μέγιστα και αποθηκεύονται στον αποκαλούμενο συσσωρευτή (accumulator space), ο οποίος είναι αρχικοποιημένος από τον αλγόριθμο για τον υπολογισμό του μετασχηματισμού Hough. Ο κλασικός μετασχηματισμός Hough αφορούσε την αναγνώριση των γραμμών στις εικόνες εισόδου, αλλά αργότερα επεκτάθηκε στην αναγνώριση των θέσεων διάφορων σχημάτων, πιο συχνά κύκλων ή ελλείψεων, με συνέπεια η χρήση του να είναι κατάλληλη για τον εντοπισμό κρατήρων, καθώς η εύρεση των κορυφών των κρατήρων μπορεί να θεωρηθεί πρόβλημα εντοπισμού ακμών (edge detection problem).

Υπάρχουν διαφορετικές μέθοδοι στη βιβλιογραφία που επωφελούνται από τα κλασικά

φίλτρα επεξεργασίας εικόνας [22, 23, 24]. Σε αυτές τις δημοσιεύσεις, χωρικά φίλτρα (spatial filters) όπως το Robert cross, το φίλτρο Canny και το Laplacian φίλτρο εφαρμόζονται σε εικόνες πυ περιέχουν κρατήρες και στη συνέχεια χρησιμοποιούν το μετασχηματισμό Hough με το στόχο είναι να βρουν τις καλύτερες παραμέτρους που ταιριάζουν σε μια επιλεγμένη γεωμετρία όπου στη περίπτωση των κρατήρων είναι ο κύκλος.

Ο μετασχηματισμός Hough είναι ένας τρόπος μετάβασης από το χώρο της εικόνας σε έναν χώρο παραμέτρων που ονομάζεται Hough space. Ο Hough space έχει τόσες διαστάσεις όσες και ο αριθμός των παραμέτρων αναζήτησης. Στην περίπτωση ενός κύκλου, έχουμε τρεις διαστάσεις (θέση **x**, θέση **y** και ακτίνα **r**). Για μια έλλειψη, ο αριθμός των παραμέτρων είναι 5. Η πολυπλοκότητα της προσαρμογής των κορυφών των κρατήρων σε ένα τυπικό σχήμα αυξάνει δραστικά με τον αριθμό των παραμέτρων και για αυτό το λόγο συνηθίζεται να αναζητούνται απλά κυκλικά χαρακτηριστικά αντί για ένα πιο περίπλοκο σχήμα. Επιπλέον, ένα άλλο πρόβλημα των κλασικών χωρικών φίλτρων είναι ότι, εφόσον προσπαθούν να ανιχνεύσουν γεωμετρικά μοτίβα, δεν έχουν τη δυνατότητα να διακρίνουν μεταξύ κρατήρων και άλλων κυκλικών γεωλογικών αντικειμένων, όπως βουνά, λόφους ή κοιλάδες.

Οι επόμενοι αλγόριθμοι ανίχνευσης αποκαλούνται επιβλεπόμενες μέθοδοι (supervised methods) όπου πρόκειται για τεχνικές που χρησιμοποιούν διαφορετικούς αλγόριθμους μηχανικής εκμάθησης για την ανίχνευση κρατήρων σε μια εικόνα. Πολλοί από αυτούς είναι κατασκευασμένοι ώστε να αναγνωρίζουν αντικείμενα μέσα σε οποιονδήποτε χώρο, που σημαίνει ότι ορίζουν μια ετικέτα (label), και όχι απλώς να ανιχνεύουν τους κρατήρες σε μια σκηνή. Επομένως, πριν από την διαδικασία ταξινόμησης πρέπει να εκτελείται ένας αλγόριθμος γεννήτριας υποψηφίων (candidates generator algorithm). Μια από τις μεθόδους που χρησιμοποιούνται περισσότερο στη βιβλιογραφία βρίσκεται στο [25]. Η μέθοδος που αναπτύχθηκε βασίζεται στην παρατήρηση ότι ο κρατήρας είναι συνδυασμός δύο σχημάτων που μοιάζουν με μισοφέγγαρο (ένα για τη σκιά και το άλλο για την φωτισμένη περιοχή). Το διάγραμμα ροής της τεχνικής φαίνεται στη σχήμα 3.1.



Σχήμα 3.1: Αλγόριθμος επιλογής υποψηφίων σύμφωνα με την εργασία στο [25].

Σε κάθε βήμα η μέθοδος εξαλείφει στοιχεία της εικόνας τα οποία δεν περιέχουν τα χαρακτηριστικα για να να χαρακτηριστούν ως κρατήρες. Καθώς ο στόχος είναι η εύρεση φωτεινών και σκοτεινών περιοχών που μοιάζουν με μισοφέγγαρο, η τεχνική αναπτύσσεται μόνο για να καθορίσει τις περιοχές ανοικτών τόνων και, στη συνέχεια, η περιοχή των σκοτεινών τόνων εντοπίζεται απλώς αντιστρέφοντας την εικόνα εισόδου. Το τμήμα του αλγόριθμου που προηγείται της επεξεργασίας (preprocessing) αρχίζει με

την αφαίρεση του φόντου (background removal) και αποτελείται από ένα πολύ μεγάλο φίλτρο αφαίρεσης χαμηλής συχνότητας (low frequency filter subtraction). Σε αυτό το μέρος, η μέθοδος χρησιμοποιεί ένα φίλτρο μέσου όρου (median filter) πολύ μεγάλου μεγέθους για να εξαλείψει τις μεταβολές φωτός που σχετίζονται με τη τοπογραφία της επιφάνειας, όπως μεγάλες κοιλάδες ή λίμνες.

Το φίλτρο ισχύος (power filter) είναι ένας τύπος φίλτρου [26] που στοχεύει στην αφαίρεση αδιακρίτων στοιχείων με χρήση μορφολογικού καθαρισμού εικόνας (morphological image cleaning). Παραδοσιακά, αυτό το φίλτρο χρησιμοποιείται για τη μείωση του θορύβου και για τη βελτίωση των συστημάτων συμπίεσης εικόνων (image compression systems). Το φίλτρο περιοχής (area filter) εξαλείφει πολύ μικρά στοιχεία. Στις δημοσιεύσεις τους, οι συγγραφείς καθόρισαν ένα όριο περιοχής 30 pixel, κάτω από το οποίο δεν είναι δυνατή η αναγνώριση ενός κρατήρα.

Τα φίλτρα σχήματος (shape filters) απορρίπτουν στοιχεία που δεν μοιάζουν με ημισέληνο. Τα φίλτρα χρησιμοποιούν σταθμισμένους μέσους όρους της εικόνας και οι στόχοι τους να είναι διαχωρίσουν τα διαφορετικά χαρακτηριστικά με στόχο να μπορούν να ερμηνευτούν εύκολα. Η χρήση αυτών των φίλτρων δίνει σε κάθε πιθανό υποψήφιο (pre-candidate) μια βαθμολογία (score) που μετρά πόσο μακριά είναι το στοιχείο από σχήμα ημισελήνου. Το τελευταίο βήμα περιλαμβάνει τη συναρμολόγηση των ζευγών των περιοχών ανοικτών και σκοτεινών τόνων. Για να γίνει αυτό εξετάζεται κάθε πιθανό ζεύγος ανοικτών και σκοτεινών περιοχών. Τα κριτήρια για την επιλογή ενός αντικειμένου ως υποψήφιος ανάλογα με τις σκοτεινές και ανοιχτές περιοχές που ανιχνεύτηκαν είναι :

- Η απόσταση μεταξύ των δύο είναι μικρή
- Οι δύο περιοχές έχουν παρόμοιο μέγεθος
- Ο συνδυασμός τους κάνει έναν κύκλο
- Οι περιφέρειες ευθυγραμμίζονται με το ηλιακό αζιμούθιο

Όταν βρεθούν τελικά οι υποψήφιοι, τα μικρά κενά εντός της επιλεγμένης περιοχής καλύπτονται με τη διαδικασία μορφολογικού κλεισίματος (morphological closing operation). Ωστόσο, ένα ακόμα σημαντικό πρόβλημα σε αυτή την εφαρμογή ανίχνευσης κρατήρων είναι να βρεθούν τα ακριβή χαρακτηριστικά που θα μπορούν να διαχωρίσουν σαφώς τι είναι κρατήρας και τι όχι.

Τα τεχνητά νευρωνικά δίκτυα (ANN) είναι νέες τεχνικές που προέρχονται από βαθιά εκμάθηση, και χρησιμοποιούνται ευρέως στην μηχανική όραση (computer vision) με πολλά υποσχόμενα αποτελέσματα [27]-[30]. Στις περισσότερες προσεγγίσεις αλγορίθμων βαθιάς εκμάθησης, το μοντέλο εκπαιδεύεται να αντιλαμβάνεται τα ξεχωριστά χαρακτηριστικά των κρατήρων χωρίς καμία ανθρώπινη παρέμβαση. Ο στόχος είναι να παρουσιαστούν στο μοντέλο αρκετές εικόνες με ετικέτες (labeled images), ώστε να του δοθεί η δυνατότητα να εκπαιδευτεί και να εκτελέσει με επιτυχία την εργασία ταξινόμησης. Όπως και σε προηγούμενες μεθόδους, τα νευρωνικά δίκτυα δεν μπορούν να εντοπίσουν κρατήρες σε μια σκηνή από μόνα τους, αλλά μόνο να ταξινομεί μια εικόνα στο σύνολό της.

Σε όλες τις αναφερόμενες εργασίες στη βιβλιογραφία, οι συγγραφείς σχεδίασαν αυτοματοποιημένους αλγόριθμους ανίχνευσης κρατήρων βάσει τεχνητών νευρωνικών δικτύων που στοχεύουν στην αναγνώριση κυκλικών χαρακτηριστικών τύπου κρατήρα σε Αρειανά ή Σεληνιακά Ψηφιακά Μοντέλα Εδάφους (digital terrain models DTM) με ποικίλες επιδόσεις [27]-[30].

3.2 Ψηφιακά Μοντέλα Εδάφους

Η εικόνα εισόδου αυτής της εργασίας είναι ένα ψηφιακό μοντέλο εδάφους (DTM) ενός μικρού τμήματος της επιφάνειας του Άρη. Τα ψηφιακά μοντέλα εδάφους, που μερικές φορές ονομάζονται ψηφιακά μοντέλα υψομέτρου (Digital Elevation Models DEM), είναι ένα τοπογραφικό μοντέλο πλανητών, φεγγαριών ή αστεροειδών, το οποίο μπορεί να χρησιμοποιηθεί σε προγράμματα υπολογιστών.Τα αρχεία αυτά περιέχουν τα δεδομένα του εδάφους σε ψηφιακή μορφή, η οποία σχετίζεται με ένα ορθογώνιο πλέγμα. Η βλάστηση, τα κτίρια και άλλα χαρακτηριστικά (αυτοκινητόδρομοι, αεροδρόμια, κλπ) στην περίπτωση της Γης αφαιρούνται ψηφιακά, αφήνοντας μόνο το έδαφος. Τα DTM χρησιμοποιούνται ευρέως από τους πολιτικούς μηχανικούς, για τη γεωδαισία και την τοπογράφηση, τη γεωφυσική, τη γεωγραφία και την τηλεπισκόπηση.

Τα DTM συνήθως συνοδεύονται από ένα αρχείο κειμένου που ονομάζεται αρχείο χώρου (world file). Ένα world file είναι ένα αρχείο απλού κειμένου έξι γραμμών που χρησιμοποιείται από συστήματα γεωγραφικών πληροφοριών (geographic information systemsGIS) σε εικόνες χαρτών ράστερ γεωαναφοράς (georeference raster map) που περιγράφουν τη θέση, την κλίμακα και την περιστροφή ενός ράστερ σε ένα χάρτη. Η περιγραφή των έξι παραμέτρων ενός world file είναι:

- Γραμμή 1 (A) : μέγεθος pixel στην κατεύθυνση x σε μονάδες units/pixel
- Γραμμή 2 (D) : περιστροφή σχετικά με τον άξονα y
- Γραμμή 3 (B) : περιστροφή σχετικά με τον άξονα x
- Γραμμή 4 (E) : μέγεθος pixel στην κατεύθυνση y σε μονάδες units/pixel, σχεδόν πάντα αρνητικό
- Γραμμή 5 (C) : συντεταγμένη x του επάνω αριστερού pixel
- Γραμμή 6 (F) : συντεταγμένη y του επάνω αριστερού pixel

Και οι τέσσερις παράμετροι εκφράζονται σε μονάδες χάρτη, οι οποίες περιγράφονται από το σύστημα χωρικής αναφοράς για το ράστερ. Όταν τα D ή B δεν είναι μηδενικά, το πλάτος των pixel δίνεται από:

$$\sqrt{A^2 + D^2} \tag{3.1}$$

και το ύψος του pixel ως:

$$\sqrt{B^2 + E^2} \tag{3.2}$$

Τα world files που περιγράφουν ένα χάρτη στο σύστημα συντεταγμένων εγκάρσιας μερκατορικής προβολής (Universal Transverse Mercator UTM) χρησιμοποιούν τις παρακάτω συμβάσεις:

- Τα D και B είναι συνήθως 0, καθώς τα pixel της εικόνας συνήθως στοιχίζονται με το πλέγμα του UTM
- Το C είναι το easting στο UTM
- Το F είναι το northing στο UTM northing
- Οι μονάδες είναι πάντα μέτρα ανά pixel

Οι τιμές αυτές σε μορφή πινάκων αναπαρίστανται ως:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} A & B & C \\ D & E & F \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

που μπορούν να γραφτούν και ως το παρακάτω σύνολο εξισώσεων:

$$x' = Ax + By + C \tag{3.3}$$

$$y' = Dx + Ey + F \tag{3.4}$$

όπου το x' είναι η υπολογιζόμενη easting του UTM του pixel στο χάρτη, y είναι η υπολογιζόμενη northing τιμή του UTM για το pixel στο χάρτη, x είναι ο αριθμός στήλης του pixel στην εικόνα μετρώντας από αριστερά, y είναι ο αριθμός γραμμής του pixel στην εικόνα που μετρώντας από επάνω, ενώ το A (ή η κλίμακα x) είναι διάσταση ενός pixel σε μονάδες χάρτη στη x κατεύθυνση, τα B και D είναι παράμετροι περιστροφής, C και F είναι όροι μετάφρασης (x, y συντεταγμένες του κέντρου του πάνω αριστερού pixel) και τέλος το E είναι το αρνητικό της κλίμακας y (διάσταση ενός pixel σε μονάδες χάρτη προς κατεύθυνση y).

Η κλίμακα y (E) είναι αρνητική, επειδή η προέλευση μιας εικόνας και το σύστημα συντεταγμένων UTM είναι διαφορετικά. Η αρχή μιας εικόνας βρίσκεται στην επάνω αριστερή γωνία, ενώ η αρχή του συστήματος συντεταγμένων UTM βρίσκεται στην κάτω αριστερή γωνία.

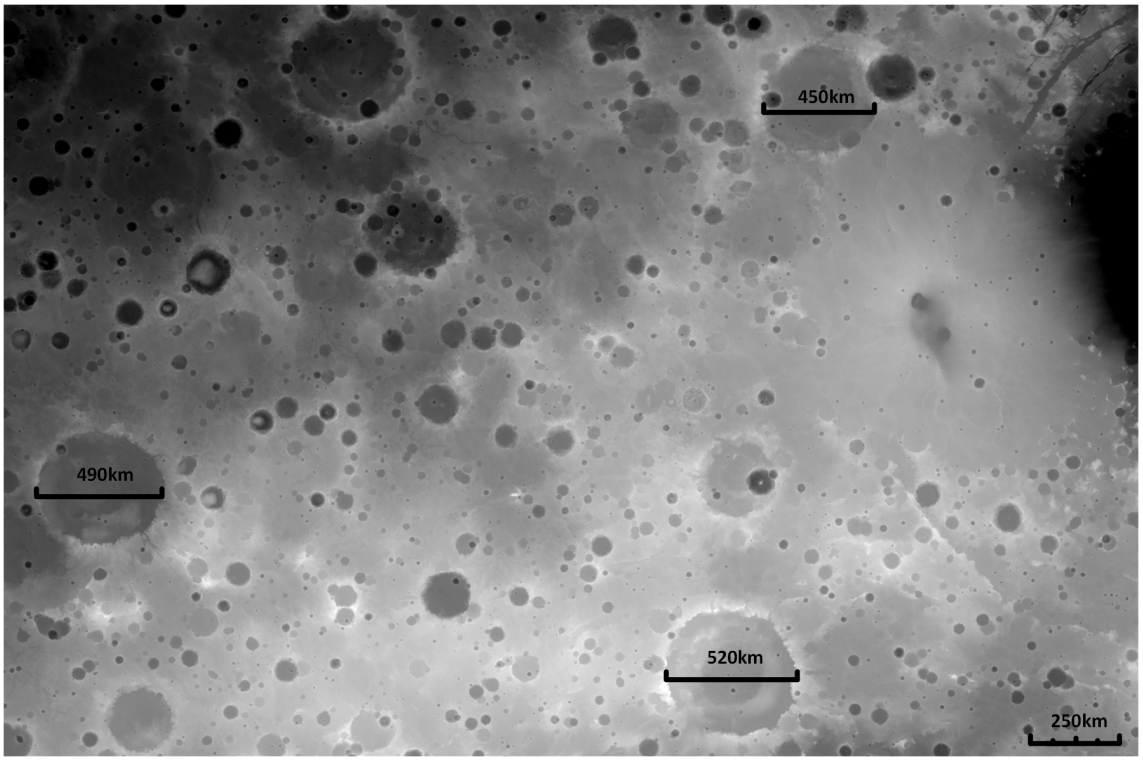
3.3 Δεδομένα Εισόδου

Το ψηφιακό μοντέλο εδάφους που χρησιμοποιείται σε αυτήν την εργασία εμφανίζεται στη Εικόνα 3.2. Αυτό το DTM συνοδεύεται από ένα world file με τα ακόλουθα δεδομένα:

- 463.0836000000
- 0.0000000000

- 0.0000000000
- -463.0836000000
- 641202.9859319335
- 1617005.7697458814

Χρησιμοποιώντας του αριθμούς του world file και τις εξισώσεις που παρουσιάστηκαν στη προηγούμενη ενότητα υπολογίζονται οι τιμές 10.311054° γεωγραφικό μήκος και 14.62272° γεωγραφικό πλάτος [1] όπου αυτό αντιστοιχεί στο επάνω αριστερό pixel της εικόνας. Στην εικόνα του DTM φαίνεται ότι ακόμη και σε μια σχετικά μικρή περιοχή του κόκκινου πλανήτη υπάρχουν πολλοί κρατήρες διαφόρων μεγεθών με το μεγαλύτερο στην εικόνα αυτή περίπου στα 520 χλμ.



Σχήμα 3.2: Ψηφιακό μοντέλο εδάφους (DTM) που χρησιμοποιείται για την υλοποίηση του νευρωνικού δικτύου (10.311054° γεωγραφικό μήκος και 14.62272° γεωγραφικό πλάτος στον πλανήτη Άρη).

Οι διαστάσεις της εικόνας είναι $8449px \times 5888px$. Χρησιμοποιώντας τους αριθμούς από το world file υπολογίζεται ότι η περιοχή της εικόνας είναι περίπου 3912km στην κατεύθυνση x και 2726km στην κατεύθυνση y υπολογίζοντας ότι κάθε pixel αντιστοιχεί σε 463 m/px.

[1] Από το NASA Mars Trek Portal: https://trek.nasa.gov/mars/#v=0.1&x=40.408963042302474&y=1.9931137830638939&z=4&p=urn%3Aogc%3Adef%3Acrs%3AEPSG%3A%3A104905&d=&locale=&b=mars&e=-1.7785361707501801%2C-18.59526473913941%2C82.59646225535514%2C22.581492305267197&sfz=&w=

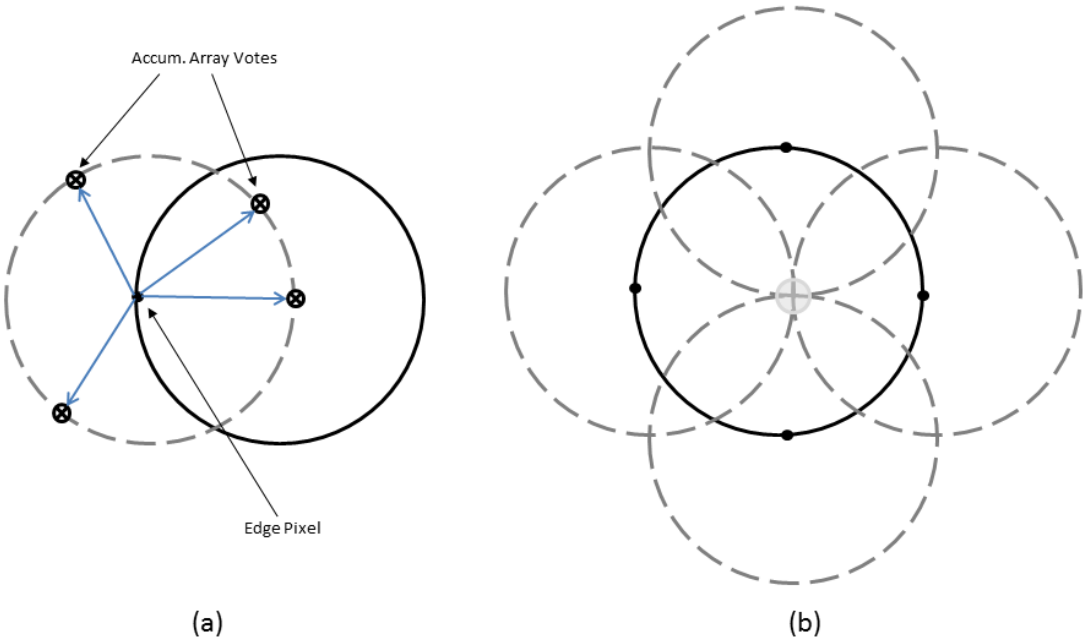
Το πρώτο βήμα σε κάθε εφαρμογή νευρωνικού δικτύου είναι να καθοριστούν τα δεδομένα εισόδου που θα χρησιμοποιηθούν για την εκπαίδευση (training) του δικτύου και στη συνέχεια για τη δοκιμή του (testing). Έτσι, το πρώτο πρόβλημα που προκύπτει κατά την εφαρμογή ενός νευρωνικού δικτύου είναι ότι το DTM δεν περιέχει καμία πληροφορία για την τοποθεσία κανενός κρατήρα αλλά και ποιος είναι ο συνολικός αριθμός των κρατήρων. Επομένως, για να δημιουργηθεί μια βάση δεδομένων κρατήρων από το δεδομένο DTM, έπρεπε να σχεδιαστεί ένας αλγόριθμος επεξεργασίας εικόνας που να εκμεταλλεύεται τις μεθοδολογίες μετασχηματισμού Hough που παρουσιάστηκαν προηγουμένως στην ενότητα 3.1. Λόγω του περιορισμού αυτών των μεθοδολογιών, το τελικό αποτέλεσμα δεν θα περιέχει όλους τους κρατήρες που είναι ορατοί στην εικόνα, αλλά μπορεί να δημιουργήσει ένα επαρκές σύνολο δεδομένων, το οποίο μπορεί στη συνέχεια να χρησιμοποιηθεί στην εκπαίδευση και δοκιμή του νευρικού δικτύου.

3.4 Δημιουργία Συνόλου Δεδομένων

Ο αλγόριθμος για τη δημιουργία του συνόλου δεδομένων εισόδου του δικτύου έχει αναπτυχθεί με τη χρήση του MATLAB. Ο πυρήνας του αλγορίθμου είναι η εντολή `imfindcirles`, η οποία αποτελεί μέρος της εργαλειοθήκης επεξεργασίας εικόνων του Mathworks [31]. Αυτή η εντολή εκμεταλλεύεται το μετασχηματισμό Hough για να εντοπίσει κυκλικά σχήματα σε μια εικόνα. Ο αλγόριθμος του `imfindcircles` εκμεταλλεύεται μια παραλλαγή του μετασχηματισμού Hough ο οποίος ονομάζεται κυκλικός μετασχηματισμός Hough (Circular Hough Transform CHT) που στοχεύει αποκλειστικά στην εύρεση κυκλικών χαρακτηριστικών σε εικόνες. Η προσέγγιση αυτή χρησιμοποιείται λόγω της ανθεκτικότητάς (robustness) της παρουσία θορύβου, αποκλεισμού (occlusion) και ποικίλου φωτισμού. Ο αλγόριθμος CHT δεν είναι ένας αλγόριθμος που έχει καθοριστεί με ακρίβεια, αλλά υπάρχουν διάφορες προσεγγίσεις που μπορούν να υιοθετηθούν κατά την εφαρμογή του. Ωστόσο, υπάρχουν τρία βασικά βήματα που είναι κοινά για όλες τις υλοποιήσεις [32] :

- Υπολογισμός πίνακα συσσωρευτή (Accumulator Array)
- Εκτίμηση κέντρου (Center Estimation)
- Εκτίμηση ακτίνας (Radius Estimation)

Τα pixel προσκηνίου με υψηλή διαβάθμιση (gradient) ορίζονται ως υποψήφια (candidate) pixel και επιτρέπεται η εισαγωγή τους στον πίνακα συσσωρευτή (accumulator array). Σε μια κλασική εφαρμογή του CHT, τα υποψήφια pixel επιλέγονται εάν υπάρχει ένα μοτίβο γύρω τους που σχηματίζει έναν πλήρη κύκλο σταθερής ακτίνας. Το σχήμα 3.3a δείχνει ένα παράδειγμα ενός υποψήφιου pixel που βρίσκεται σε πραγματικό κύκλο (συμπαγής κύκλος) και το κλασικό μοτίβο ψηφοφορίας CHT (διακεκομμένοι κύκλοι) για το υποψήφιο pixel.

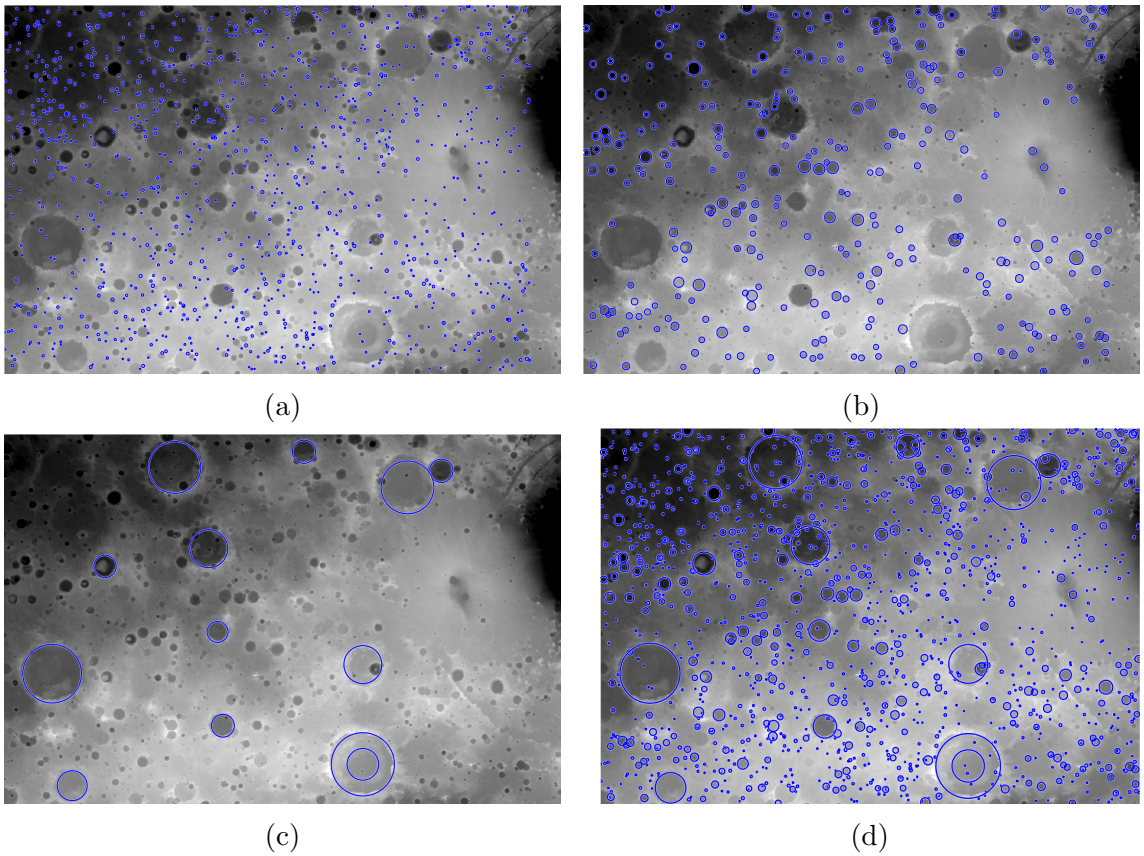


Σχήμα 3.3: Παραδείγματα (α) ενός υποψήφιου pixel που βρίσκεται σε πραγματικό κύκλο, β) των υποψήφιων pixel (συμπαγείς κουκκίδες) που βρίσκονται σε πραγματικό κύκλο (συμπαγής κύκλος) και των μοτίβων ψήφου τους (διακεκομμένοι κύκλοι)

Οι ψήφοι των υποψηφίων pixel που ανήκουν σε ένα κύκλο στην εικόνα τείνουν να συσσωρεύονται στον accumulator array που αντιστοιχεί στο κέντρο του κύκλου. Επομένως, τα κυκλικά κέντρα εκτιμώνται εκτελώντας ανίχνευση των ακμών (edge detection) στον accumulator. Το σχήμα 3.3b εμφανίζει ένα παράδειγμα των υποψηφίων pixel (συμπαγείς κουκκίδες) που βρίσκονται σε πραγματικό κύκλο (συμπαγής κύκλος) και των μοτίβων ψήφου τους (διακεκομμένοι κύκλοι) που συμπίπτουν με το κέντρο του πραγματικού κύκλου. Αν ο ίδιος πίνακας συσσωρευτής χρησιμοποιείται για περισσότερες από μία τιμές ακτίνας, όπως συνήθως γίνεται σε αλγόριθμους CHT, η ακτίνα των κύκλων που ανιχνεύονται πρέπει να εκτιμάται ως ξεχωριστό βήμα. Οι ακτίνες υπολογίζονται με χρήση των εκτιμώμενων κυκλικών κέντρων μαζί με τις πληροφορίες της εικόνας. Η τεχνική βασίζεται στον υπολογισμό ακτινικών ιστογραμμάτων (radial histograms) [33], [34] και βρίσκεται εκτός του πεδίου εφαρμογής αυτής της εργασίας. Η βασική σύνταξη της εντολής `imfindcircle` είναι η ακόλουθη:

```
[centers,radii] = imfindcircles(A,radiusRange)
```

όπου A είναι ένας πίνακας που περιέχει τα pixel της εικόνας και `radiusRange`, ένα διάνυσμα με ελάχιστη και μέγιστη ακτίνα σε pixel. Το αποτέλεσμα, `centers`, είναι ένας πίνακας δύο στηλών που περιέχει τις (x,y) συντεταγμένες των κυκλικών κέντρων στην εικόνα και η παράμετρος `radii` περιέχει τις εκτιμώμενες ακτίνες που αντιστοιχούν σε κάθε κέντρο του κύκλου. Η παράμετρος `radiusRange` είναι προαιρετική, αλλά βελτιώνει την ακρίβεια των αλγορίθμων και μειώνει τον υπολογιστικό χρόνο. Για υψηλή ακρίβεια, σχετικά μικρή ακτίνα θα πρέπει χρησιμοποιείται. Ένας καλός εμπειρικός κανόνας είναι η επιλογή της περιοχής ακτίνας, ώστε [39] :



(a) (b)

(c) (d)

Σχήμα 3.4: Η έξοδος του αλγόριθμου αναζήτησης μέσω μείωσης δειγματοληψίας (downsampling) που περιγράφεται στην ενότητα. Έξοδος αλγορίθμου όταν (α) downsampling=1, (β) downsampling=4, (γ) downsampling=16, (δ) συνδυασμός όλων των ανιχνευμένων κρατήρων.

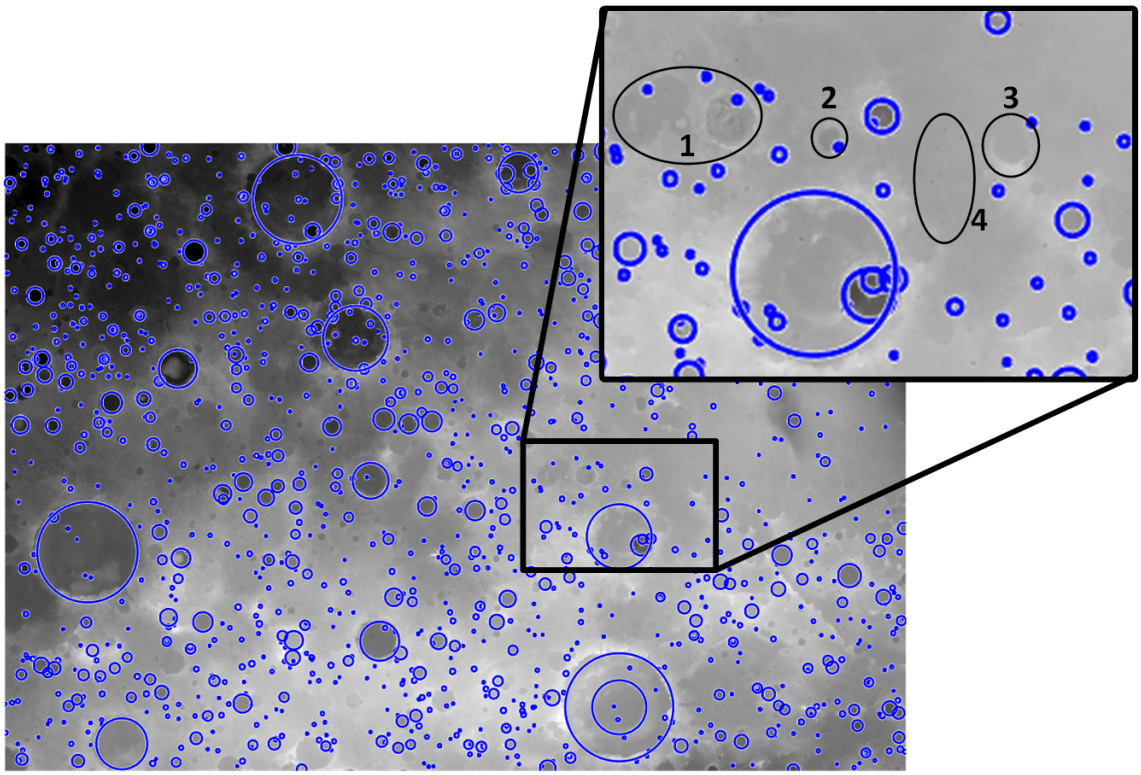
$$Rmax < 3 \times Rmin \text{ και } (Rmax - Rmin) < 100$$

Αυτός είναι και ένας περιορισμός για τον αλγόριθμο καθώς οι κρατήρες στην εικόνα εισόδου έχουν ποικίλα μεγέθη και οι περιορισμοί που αναφέρθηκαν δεν μπορούν να ικανοποιηθούν για όλα τα μεγέθη των κρατήρων. Επιπροσθέτως, ένας ακόμα περιορισμός του αλγορίθμου είναι το γεγονός ότι δεν μπορεί να εντοπίσει κυκλικά χαρακτηριστικά με ακτίνα μικρότερη των 10px, συνεπώς αρκετοί από τους μικρότερους κρατήρες βρίσκονται εκτός του ορίου ανίχνευσης. Για τη κατάρριψη του πρώτου περιορισμού, σχεδιάστηκε και υλοποιήθηκε μια απλή μεθοδολογία. Αρχικά, εκτελείται η διεργασία `imfindcircles` στην αρχικά εικόνα με τέτοιο τρόπο ώστε να αναζητεί κύκλους με εύρος τιμών από 10px έως 30px οι οποίοι αντιστοιχίζονται σε κρατήρες με διαστάσεις που κυμαίνονται από 4.63km έως 13.89km. Το αποτέλεσμα αυτής της εκτέλεσης της `imfindcircles` εντοπίζει τους μικρότερους κρατήρες στην εικόνα οι οποίοι αποθηκεύονται στους πίνακες `centers` και `radii`

Όμως η απλή αύξηση του εύρους ακτίνας αυξάνει δραστικά το χρόνο εκτέλεσης και επίσης μειώνει την ακρίβεια του αλγόριθμου. Για το λόγο αυτό, η προσέγγιση είναι να μειωθεί η δειγματοληψία της εικόνας (downsample) κατά ένα συντελεστή τέσσερα και να εκτελεστεί εκ νέου ο αλγόριθμος χρησιμοποιώντας ένα παρόμοιο εύρος ακτίνας με την πρώτη εκτέλεση. Αυτή τη φορά, ο αλγόριθμος θα εξακολουθεί να ανιχνεύει

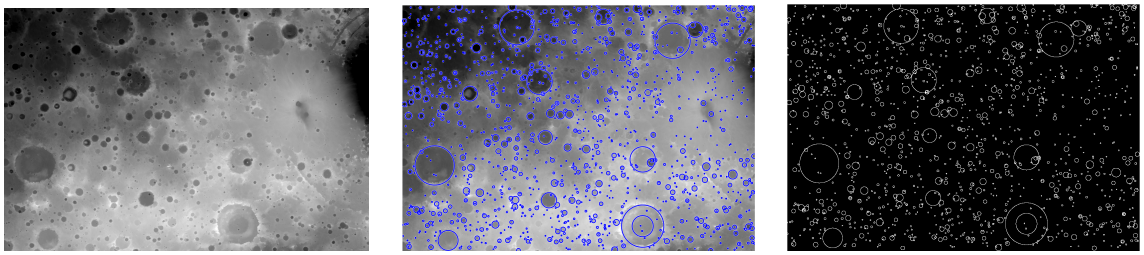
κρατήρες με ακτίνα που κυμαίνεται από 10px έως 30px, αλλά λόγω του συντελεστή μείωσης δειγματοληψίας (τέσσερα), τα πραγματικά μεγέθη κρατήρων αντιστοιχούν σε 40px και 120px, εντοπίζοντας τελικά κυκλικά σχήματα που είναι μεγαλύτερα από αυτά που ανιχνεύθηκαν κατά την πρώτη εκτέλεση. Στη συνέχεια επαναλαμβάνεται η ίδια μέθοδος μείωσης δειγματοληψίας έως ότου ανιχνευθούν και οι μεγαλύτεροι κρατήρες.

Το σχήμα 3.4 εμφανίζει την έξοδο του αλγορίθμου για τρεις διαφορετικούς συντελεστές μείωσης δειγματοληψίας και το συνδυασμένο αποτέλεσμα. Η επάνω αριστερή εικόνα δεν αντιστοιχεί σε μείωση δειγματοληψίας (downsample=1), επομένως ο αλγόριθμος έχει εντοπίσει μόνο τους μικρότερους κρατήρες, στη συνέχεια το σχήμα 3.4(b) εμφανίζει την έξοδο με μείωση δειγματοληψίας κατά 4 και στο σχήμα 3.4(c) με συντελεστή μείωσης δειγματοληψίας 16. Είναι σαφές ότι οι δύο τελευταίες εκτελέσεις κατάφεραν να εντοπίσουν μεγαλύτερους κρατήρες από την πρώτη εκτέλεση. Τελικά, στο σχήμα 3.4(d) εμφανίζεται το συνολικό αποτέλεσμα μετά τον συνδυασμό όλων των μερικών αποτελεσμάτων από τις προηγούμενες εκτελέσεις.



Σχήμα 3.5: Παραδείγματα κρατήρων που δεν εντοπίστηκαν από τον αλγόριθμο μείωσης δειγματοληψίας.

Η μεθοδολογία αυτή κατόρθωσε να ανιχνεύσει 1607 κρατήρες στην εικόνα, ωστόσο υπάρχουν ακόμη περισσότεροι που ο αλγόριθμος δεν μπόρεσε να εντοπίσει. Οι κρατήρες που δεν ήταν δυνατόν να εντοπιστούν είναι όλοι παρόμοιοι σε δύο παραμέτρους, έχουν μικρή διάμετρο και οι συνθήκες φωτισμού δεν τις καθιστούν διακριτές από το περιβάλλον τους. Ένα τέτοιο παράδειγμα παρουσιάζεται στο σχήμα 3.5. Η επιλεγμένη περιοχή εμφανίζει μερικούς από τους κρατήρες που ο αλγόριθμος δεν κατάφερε να εντοπίσει. Ο λόγος για τις αποτυχίες στις περιοχές 1, 2 και 3 είναι οι κακές συνθήκες φωτισμού στον κρατήρα, ενώ οι αποτυχίες στη περιοχή 4 οφείλεται σε πολύ μικρές ακτίνες (πλά-



(a) (b) (c)

Σχήμα 3.6: (a) Εικόνα εισόδου, (b) Κρατήρες που εντοπίστηκαν με τη χρήση του μετασχηματισμού Hough, (c) Δυαδική εικόνα που χρησιμοποιείται για τους στόχους (targets) του νευρωνικού δικτύου.

τος μικρότερο από 10px) όπου ο αλγόριθμος της εντολής `imfindcircles` εμφανίζει αδυναμίες. Πρόκειται για έναν ακόμη περιορισμό των κλασικών προσεγγίσεων επεξεργασίας εικόνας, αλλά για την εκπαίδευση του νευρωνικού δικτύου αυτής της εργασίας οι ανιχνευθέντες κρατήρες είναι επαρκείς.

Μετά τον εντοπισμό των κρατήρων με τη μέθοδο μετασχηματισμού Hough, μπορεί να δημιουργηθεί ένα σύνολο δεδομένων εκπαίδευσης (training dataset). Η εικόνα εδάφους θα είναι η είσοδος του νευρωνικού δικτύου και οι ανιχνευμένοι κρατήρες θα είναι οι στόχοι (targets) που θα προσπαθήσει να επιτύχει ο αλγόριθμος. Η εκπαίδευση (training) και η επαλήθευση (validation) ακολουθούν τη μέθοδο που περιγράφεται στο [30] και οι στόχοι (targets) του νευρωνικού δικτύου εμφανίζονται στο σχήμα 3.6. Οι στόχοι που εμφανίζονται στο σχήμα 3.6(c) έχουν δημιουργηθεί από τους πίνακες *centers* και *radii* που υπολογίστηκαν στα προηγούμενα βήματα.

Αρχικά, δημιουργείται μια μαύρη εικόνα με τις διαστάσεις του αρχικού DTM, χρησιμοποιώντας την εντολή *zeros* του MATLAB (στο MATLAB η τιμή 0 αντιπροσωπεύει το μαύρο χρώμα και η τιμή 255 το λευκό). Στη συνέχεια, οι συντεταγμένες *(x,y)* κάθε κύκλου δημιουργούνται χρησιμοποιώντας τους δύο ακόλουθους τύπους :

Mathematical Formula	**MATLAB Command**
$x = centerX + radius \cdot cos(\theta)$	`x = centerX(i) + radius(i) * cosd(theta)`
$y = centerY + radius \cdot sin(\theta)$	`y = centerY(i) + radius(i) * sind(theta)`

Η παράμετρος i στο MATLAB είναι ένας δείκτης που αναπαριστά κάθε κρατήρα που εντοπίστηκε (το i κυμαίνεται από 0 έως 1607 σε αυτήν την περίπτωση) και το θ είναι ένα διάνυσμα που περιέχει τις τιμές που κυμαίνονται από 0 έως 360 και δημιουργείται με :

```
theta = linspace(0, 360, 4*pi*radius(i))
```

η εντολή `linspace(x1,x1,n)` δημιουργεί n σημεία με την απόσταση μεταξύ των σημείων να είναι $(x2 - x1)/(n - 1)$. Τέλος, ο πίνακας παρουσιάζει τη χρήση των συναρτήσεων *cosd* και *sind*. Αυτές οι δύο συναρτήσεις εξάγουν το συνημίτονο και

το ημίτονο αντίστοιχα με την είσοδο να είναι σε μοίρες (υπενθύμιση ότι τα *cos* και *sin* στο MATLAB απαιτούν είσοδο σε ακτίνια/rads). Μετά τον υπολογισμό των συντεταγμένων *(x,y)*, η δυαδική εικόνα του σχήματος 3.6(c) δημιουργείται με τον απλό βρόχο:

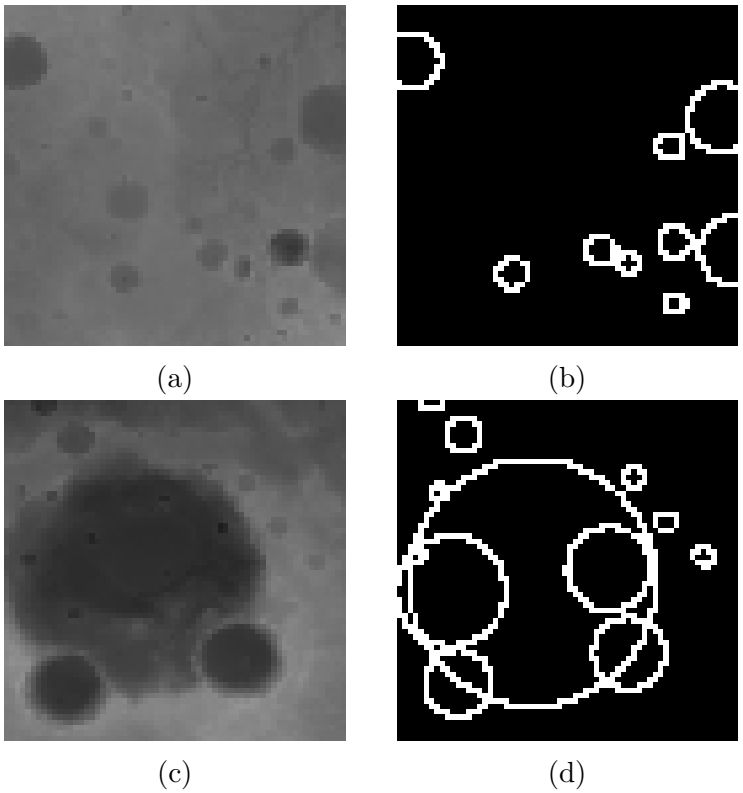
```
for k = 1 : length(x)
     if(y(k) > 0 && x(k)>0 )
          grayImage(y(k), x(k)) = 255;
     end
end
```

όπου `grayImage` είναι η δυαδική εικόνα που περιέχει τους στόχους (Σχήμα. 3.6(c)). Αυτός ο κωδικός MATLAB εκτελείται για κάθε καταχώρηση στα διανύσματα x και y και αποθηκεύει στον πίνακα `grayImage` τον αριθμό 255 που αντιστοιχεί στο λευκό χρώμα.

Αφού δημιουργήθηκε η δυαδική εικόνα (εικόνα στόχος), το επόμενο βήμα είναι ο διαχωρισμός την εικόνα σε μικρότερα τμήματα, δημιουργώντας έτσι μια βάση δεδομένων, που θα είναι τα δεδομένα εκπαίδευσης και επικύρωσης για το νευρικό δίκτυο ανίχνευσης κρατήρων. Το μέγεθος των τμημάτων της εικόνας εξαρτάται αποκλειστικά από την πολυπλοκότητα των υπολογισμών. Για παράδειγμα, μια εικόνα $256px \times 256px$ θα έχει ως αποτέλεσμα μεγαλύτερο υπολογιστικό φόρτο εργασίας από μια μικρότερη εικόνα $128px \times 128px$. Το τελικό μέγεθος τμήματος έχει επιλεγεί να είναι $64px \times 64px$ μετά από εκτεταμένο πειραματισμό για να βρεθεί το μεγαλύτερο δυνατό τμήμα εικόνας που η επικείμενη εκπαίδευση του δικτύου θα έχει υπολογιστικά λογικό χρόνο εκτέλεσης. Είναι κατανοητό ότι αυτό το τμήμα της κατάτμησης δεν βασίζεται σε ένα ισχυρό θεωρητικό υπόβαθρο, αλλά είναι το αποτέλεσμα μιας προσέγγισης δοκιμής και σφάλματος (trial-and-error). Το μπλοκ κώδικα στο MATLAB που δημιουργεί τα τμήματα είναι ως εξής:

```
segmentSize=64;
k=1;
for j=1:1:size(I,2)/segmentSize
     for i=1:1:size(I,1)/segmentSize
          croppedImage=I(segmentSize*(i-1) + 1:segmentSize  + ...
                         segmentSize*(i-1),segmentSize*(j-1)+ ...
                         1:segmentSize+segmentSize*(j-1));
          P(:,k)=croppedImage(:);
          k=k+1;
     end
end
```

Πρόκειται για έναν απλό εμφωλιαμένο βρόχο for (nested for loop) όπου η παράμετρος `I` είναι η εικόνα DTM, η παράμετρος `croppedImage` πρόκειται για έναν προσωρινό (temp) πίνακα που αποθηκεύει μια εικόνα $64px \times 64px$ σε κάθε επανάληψη και τέλος, η μεταβλητή `P` είναι ο πίνακας που αποθηκεύει όλα αυτά τα μεμονωμένα τμήματα εικόνας. Μετά από κάθε επανάληψη, ο πίνακας `croppedImage` αποθηκεύεται στο τέλος της



(a) (b)

(c) (d)

Σχήμα 3.7: Δύο παραδείγματα υποεικόνων (δείγματα) με τους αντίστοιχους στόχους τους.

μεταβλητής P σε διανυσματική μορφή (τένσορας), ώστε τελικά μετά την ολοκλήρωση του βρόγχου, ο πίνακας P να έχει μέγεθος 4096×12144. Αυτοί οι αριθμοί αντιπροσωπεύουν, πρώτον, ότι τα $64px \times 64px$ σε διανυσματική μορφή είναι 4096 και, δεύτερον, ότι η μεγάλη εικόνα DTM είναι κατακερματισμένο σε 12144 υποεικόνες (δείγματα).

Το σχήμα 3.7 εμφανίζει δύο υποεικόνες από το αρχικό DTM με τις αντίστοιχες δυαδικές υποεικόνες στόχους. Αυτά τα δύο παραδείγματα αναδεικνύουν επίσης το πρόβλημα με τα μικρά μεγέθη κρατήρων, καθώς ορισμένοι από τους μικρότερους από αυτούς δεν μπορούσαν να εντοπιστούν, αλλά όπως αναφέρθηκε ήδη, αυτό δεν αποτελεί στην πραγματικότητα πρόβλημα, καθώς η συγκεκριμένη μέθοδο ανίχνευσης είναι επαρκής για το βήμα εκπαίδευσης του νευρωνικού δικτύου αυτής της εργασίας.

3.5 Η Υλοποίηση του Νευρωνικού Δικτύου

Αυτή η ενότητα είναι αφιερωμένη στην παρουσίαση της υλοποίησης του νευρωνικού δικτύου για τον εντοπισμό κρατήρων. Σε κάθε υλοποίηση νευρωνικού δικτύου το πρώτο βήμα είναι η εκπαίδευσή του βάση ενός σετ δεδομένων, όπου στη περίπτωση αυτής της εργασίας, το σετ δεδομένων είναι οι υποεικόνες που αναλύθηκαν στη προηγούμενη ενότητα. Στη συνέχεια, αφού ολοκληρωθεί το βήμα της εκπαίδευσης, ξεκινάει το βήμα της επαλήθευσης του δικτύου κατά το οποίο αποφασίζεται και η αποτελεσματικότητά του. Εφόσον, η αποτελεσματικότητα του δικτύου είναι επαρκής τότε το δίκτυο είναι έτοιμο να παράξει αποτελέσματα. Βέβαια αν η απόδοσή του δεν είναι αποδεκτή, θα

πρέπει να επαναληφθεί το βήμα της εκπαίδευσης έως ότου βελτιωθεί η παραχθούν τα επιθυμητά αποτελέσματα.

Αρχικά, στην υλοποίηση της διαδικασίας της εκπαίδευσης ενός δικτύου είναι η επιλογή των παραμέτρων του. Στις βασικές παραμέτρους εντοπίζουμε το βάθος του δικτύου, δηλαδή τον αριθμό των κρυφών επιπέδων, το πλήθος των νευρώνων σε κάθε κρυφό επίπεδο, τη συνάρτηση ενεργοποίησης (activation function) και τη συνάρτηση σφάλματος (error function). Στη συνέχεια, υπάρχουν μερικοί ακόμα παράμετροι οι οποίοι αφορούν την υλοποίηση του δικτύου και όχι την αρχιτεκτονική του. Αυτοί οι παράμετροι είναι ο μέγιστος αριθμός των εποχών (epochs), η ανοχή του σφάλματος (error tolerance), και δύο σταθεροί παράμετροι οι οποίοι αφορούν την αρχικοποίηση του δικτύου. Ο αριθμός των εποχών και η ανοχή χρησιμοποιούνται για να ορίσουν την ολοκλήρωση του βήματος της εκπαίδευσης του δικτύου. Η εκπαίδευση ολοκληρώνεται όταν η συνάρτηση σφάλματος βγάλει αποτέλεσμα μικρότερο της ανοχής ή όταν ο αλγόριθμος εκπαίδευσης κάνει επαναλήψεις ίσες με τον αριθμό των εποχών που έχει οριστεί. Ο κώδικας που αφορά την αρχικοποίηση των παραμέτρων είναι ο ακόλουθος :

```
S1=size(P,1);       % Number of Neurons First Layer
S2=size(P,1);       % Number of Neurons Output Layer
[R,Q]=size(P);

epochs = 100000;    % number of iterations.
goal_err = 10e-5;   % goal error (tolerance)
a=0.3;              % define the range of random variables
b=-0.3;

% Initialize the NN with random weights
W1=a+(b-a)*rand(S1,R);     % Weights between Input and Hidden Layer
W2=a+(b-a)*rand(S2,S1);    % Weights between Hidden and Output Layer

% Calculations for the neuron variables
% Use of the logsigma activation function
n1=W1*P;
% A1 refers to the hidden layer
A1=logsig(n1);
n2=W2*A1;
% A2 refers to the output layer
A2=logsig(n2);

% Calculate the initial error :
% Large error due to random initialization
e=A2-T;
error=0.5*mean(mean(e.*e));
```

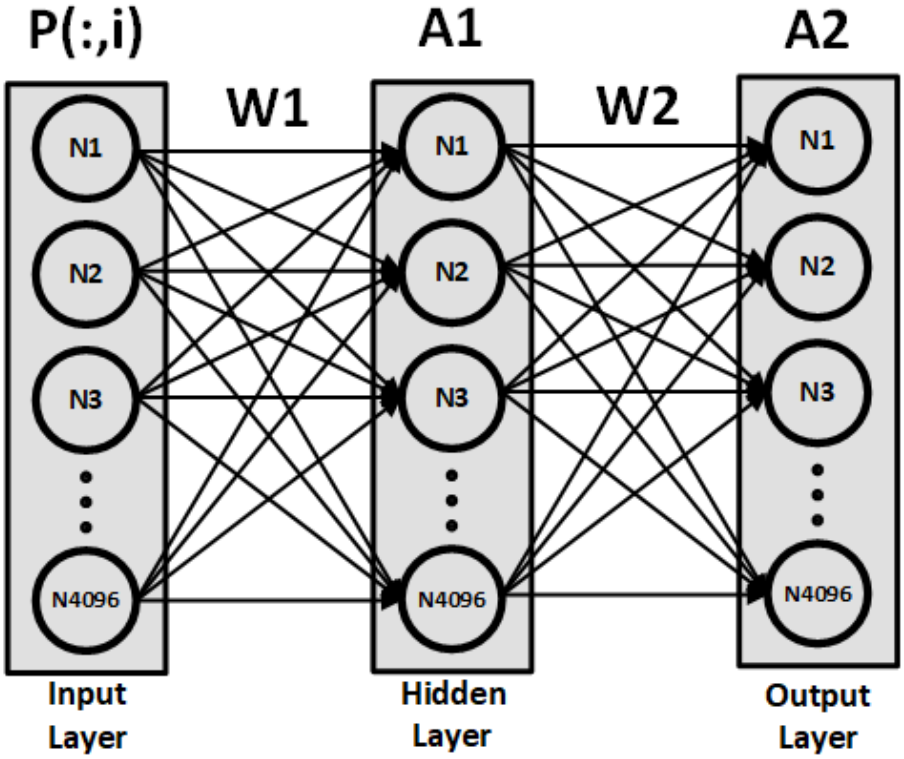
Στην υλοποίηση αυτού του νευρωνικού δικτύου έχει επιλεγεί μόνο ένα κρυφό επίπεδο διότι η χρήση περισσότερων αυξάνει δραματικά τη πολυπλοκότητα του με αποτέλεσμα η διαδικασία της εκπαίδευσης να έχει μη πρακτικό χρόνο εκτέλεσης. Με τη μεταβλητή

S1 ορίζεται ο αριθμός των νευρώνων του κρυφού επιπέδου ενώ η μεταβλητή S2 ορίζει τον αριθμό νευρώνων του επιπέδου εξόδου. Αναμενόμενα, ο αριθμός νευρώνων του σταδίου εξόδου έχει επιλεγεί ίσος με το μέγεθος της διάστασης του τένσορα εισόδου (P) μιας υποεικόνας καθώς η έξοδος του νευρωνικού δικτύου πρόκειται για μια δυαδική εικόνα σαν αυτές που εμφανίζονται στο σχήμα 3.7. Στη συνέχεια ο αριθμός νευρώνων του κρυφού επιπέδου (S1), σύμφωνα με τα σχόλια της ενότητας 2.4, έχει επιλεχθεί ώς το ίδιο μέγεθος με τον αριθμό νευρώνων του επιπέδου εισόδου και εξόδου.

Η επόμενη γραμμή του κώδικα ορίζει δύο μεταβλητές, τις R και Q όπου αποθηκεύονται το μέγεθος του τένσορα εισόδου και ο αριθμός των υποεικόνων αντίστοιχα. Ώς αριθμός εποχών έχουν οριστεί οι 100000 επαναλήψεις και η ανοχή του σφάλματος ως 10^{-5}. Οι αριθμοί αυτοί επιλέχθηκαν μετά από πειραματισμούς με στόχο τη βελτιστοποίηση της αποτελεσματικότητας του νευρωνικού δικτύου και του χρόνου που απαιτείται για το στάδιο της εκπαίδευσής του. Στη συνέχεια οι παράμετροι a και b είναι δύο σταθεροί αριθμοί που χρησιμοποιούνται μόνο για το στάδιο της αρχικοποίησης και περιορίζουν το εύρος τιμών από το οποίο θα επιλεγούν τυχαία οι τιμές της αρχικοποίησης.

Οι δύο επόμενες γραμμές ορίζουν την αρχικοποίηση των τιμών για τα βάρη μεταξύ του επιπέδου εισόδου, κρυφού επιπέδου και επιπέδου εξόδου. Ο παράγοντας $a+(b-a)$ απλώς ορίζει τη περιοχή τιμών και η επιλογή των τυχαίων τιμών επιτυγχάνεται με την εντολή `rand()` του MATLAB. Η εντολή αυτή παράγει έναν τυχαίο αριθμό μεταξύ του 0 και 1 ακολουθώντας ομοιόμορφη κατανομή. Τα δύο ορίσματα που περιλαμβάνει αναφέρονται στο μέγεθος του πίνακα που θα επιστρέψει ως αποτέλεσμα. Έτσι στη πρώτη περίπτωση (`rand(S1,R)`) ο πίνακας που ενώνει το στάδιο εισόδου και το κρυφό επίπεδο θα είναι ένας πίνακας με διαστάσεις $S1 \times R$ δηλαδή 4096×4096 στη συγκεκριμένη υλοποίηση. Εφόσον το στάδιο εισόδου πρόκειται για εικόνες διαστάσεων 64×64 σε μορφή τένσορα (4096×1) τότε είναι αντιληπτό ότι και οι νευρώνες του σταδίου εισόδου θα είναι 4096. Από τη στιγμή που οι νευρώνες του κρυφού επιπέδου έχουν επιλεγεί ίσοι με τους νευρώνες του πρώτου επιπέδου τότε οι διαστάσεις του πίνακα W1 (4096×4096) υποδηλώνουν ότι από κάθε νευρώνα του σταδίου εισόδου υπάρχει μία σύναψη που τον ενώνει με κάθε έναν νευρώνα του κρυφού επιπέδου. Για παράδειγμα, `W2(100,200)` είναι το βάρος που ενώνει τον νευρώνα 100 του σταδίου εισόδου με το νευρώνα 200 του κρυφού επιπέδου. Με αντίστοιχη λογική, η μεταβλητή W2 όπου ορίζει τα βάρη του κρυφού επιπέδου και της εξόδου έχει επίσης διαστάσεις 4096×4096.

Εφόσον έχουν αρχικοποιηθεί τα βάρη του νευρωνικού δικτύου, η επόμενη εργασία στο βήμα της αρχικοποίησης είναι ο ορισμός των νευρώνων του κρυφού επιπέδου και της εξόδου. Οι παράμετροι που αντιστοιχούν στο κρυφό επίπεδο και στο επίπεδο εξόδου είναι οι A1 και A2 αντίστοιχα. Στην ενότητα 2.3 είχε οριστεί ότι οι νευρώνες ενός επιπέδου υπολογίζονται ως το εσωτερικό γινόμενο των νευρώνων εισόδου και των αντίστοιχων βαρών τους. Επομένως, επιστρέφοντας πίσω στον κώδικα του MATLAB παρουσιάζεται η μεταβλητή n1 όπου έχει αυτόν ακριβώς το ρόλο. Στη γλώσσα του MATLAB το σύμβολο "*" αναφέρεται στο εσωτερικό γινόμενο όταν οι τελεστές αριστερά και δεξιά του είναι πίνακες (ή διανύσματα). Έτσι, στη μεταβλητή n1 αποθηκεύεται το αποτέλεσμα του εσωτερικού γινομένου μεταξύ της εισόδου P και των βαρών W1 μεταξύ του πρώτου και του κρυφού επιπέδου. Αντίστοιχα, η μεταβλητή n2



Σχήμα 3.8: Διάγραμμα αρχιτεκτονικής της υλοποίησης νευρωνικού δικτύου.

περιέχει το αποτέλεσμα του εσωτερικού γινομένου μεταξύ των νευρώνων του κρυφού επιπέδου `A1` και των βαρών `W2`.

Όπως αναφέρθηκε στην ενότητα 2.2 μια σημαντική παράμετρος που ολοκληρώνει την αλγοριθμική μορφή του νευρώνα είναι η μη γραμμική συνάρτηση ενεργοποίησης (activation function) που έχει ως στόχο τη απεικόνιση (mapping) των τιμών της εισόδου της σε ένα συγκεκριμένο εύρος τιμών. Στην υλοποίηση αυτής της εργασίας επιλέχθηκε η σιγμοειδής (sigmoid) συνάρτηση με καμπύλη που εμφανίζεται στο σχήμα 2.7. Στον κώδικα MATLAB αυτό υλοποιείται με τις εντολές `A1=logsig(n1)` και `A2=logsig(n2)` όπου αφορούν τους νευρώνες του κρυφού επιπέδου και τους νευρώνες εξόδου αντίστοιχα.

Η γραφική απεικόνιση της αρχιτεκτονικής που αναφέρθηκε έως τώρα εμφανίζεται στο σχήμα 3.8. Αριστερά με το γράμμα `P` ορίζεται το επίπεδο εισόδο (input layer) στο οποίο ξεχωρίζονται 4096 νευρώνες οι οποίοι είναι η αναπαράσταση κάθε υποεικόνας σε μορφή τένσορα. Στο κέντρο με το γράμμα `A1` ορίζεται το κρυφό επίπεδο (hidden layer) το οποίο επίσης αποτελείται από 4096 νευρώνες. Μεταξύ του `P` και του `A1` υπάρχουν βέλη (συνάψεις) που συνδέουν τους νευρώνες των δύο αυτών επιπέδων τα οποία αναπαριστούν τα βάρη `W1`. Αντίστοιχα, στο στάδιο εξόδου `A2`, που βρίσκεται στο δεξί μέρος της εικόνας, εμφανίζονται τα βάρη `W2`.

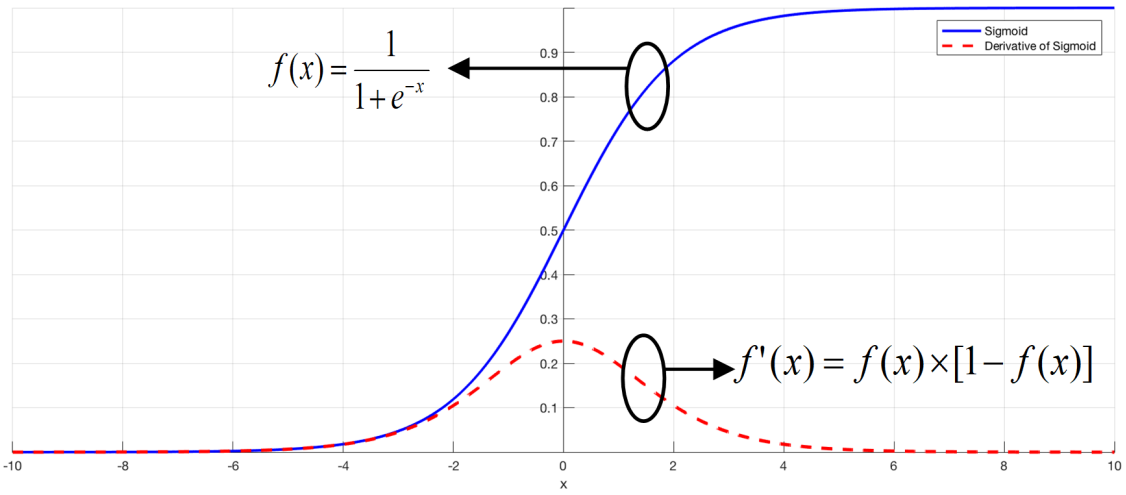
Τέλος, το κομμάτι κώδικα που παρουσιάζεται είναι ο ορισμός της συνάρτησης σφάλματος (error function), και η αρχικοποίηση του σφάλματος. Η μεταβλητή `e` πρόκειται για έναν πίνακα ο οποίος περιέχει τη διαφορά του αποτελέσματος του νευρ-

ωνικού δικτύου και των στόχων (targets) T που πρόκειται για τις δυαδικές απεικονίσεις σαν αυτές του σχήματος 3.7. Τέλος, η συνάρτηση σφάλματος που επιλέχθηκε είναι η συνάρτηση του μέγιστου τετραγωνικού σφάλματος (mean square error MSE) που παρουσιάζεται στη τελευταία γραμμή του κώδικα.

Εφόσον ολοκληρώθηκε η αρχικοποίηση του νευρωνικού δικτύου, το επόμενο βήμα είναι η εκπαίδευσή του (training). Για το βήμα της εκπαίδευσης υλοποιήθηκε ο παρακάτω κώδικας MATLAB:

```
for  itr=1:epochs
% If the error tolerance is reached, finish training
    if error <= goal_err
        break
    else

    for i=1:Q
% Derivative of logsigma for hidden and output layers
        df1 = dlogsig( n1 , A1(:,i) );
        df2 = dlogsig( n2 , A2(:,i) );

% Backpropagation parameters calculation
        s2 = -2 * ( diag(df2) * e(:,i) );
        s1 = diag(df1) * ( W2' * s2 );

% Application of Backpropagation for W2 and W1
        W2 = W2 - 0.005 * ( s2 * A1(:,i)' );
        W1 = W1 - 0.005 * ( s1 * P(:,i)' );

% Neuron Recalculation applying logsigma function
        A1(:,i) = logsig( W1 * P(:,i) );
        A2(:,i) = logsig( W2 * A1(:,i) );

    end

% Error Recalculation
    e = T - A2;
    error=0.5*mean(mean(e.*e));
    disp(sprintf('Iteration :%5d mse :%12.6f%',itr,error));
    mse(itr)=error;

    end
end
```

Πρόκειται για έναν κώδικα ο οποίος αποτελείται από έναν εμφολιασμένο βρόγχο for (nested for loop). Ο εξωτερικός βρόγχος είναι εύκολος στη κατανόηση, καθώς είναι υπεύθυνος να μετρά τον αριθμό των επαναλήψεων και να ολοκληρώνει την εκτέλεση της διαδικασίας της εκπαίδευσης όταν ο αριθμός των επαναλήψεων υπερβεί τον αριθμό



Σχήμα 3.9: Γραφικές παραστάσεις της σιγμοειδούς συνάρτησης (μπλε γραμμή) και της παραγώγου της (κόκκινη γραμμή).

των εποχών που έχει οριστεί με τη μεταβλητή `epochs` στο βήμα της αρχικοποίησης. Μέσα στον πρώτο βρόγχο φαίνεται και η δεύτερη συνθήκη τερματισμού μέσα στη δομή `if`. Η δεύτερη συνθήκη ορίζει ότι η εκπαίδευση ολοκληρώνεται εφόσον το σφάλμα που υπολογίζεται γίνει μικρότερο της ανοχής `goal_err` που ορίστηκε 10^{-5} στο βήμα της αρχικοποίησης.

Εφόσον δεν έχει επιτευχθεί ο μέγιστος αριθμός επαναλήψεων και η ελαχιστοποίηση του σφάλματος, ο αλγόριθμος μπαίνει στο βήμα της εκπαίδευσης. Η πρώτη διαδικασία που εκτελείται είναι η παραγώγιση των τιμών του νευρώνων ως προς το αποτέλεσμα που παράγεται από την εφαρμογή της συνάρτησης ενεργοποίησης (logsigma). Η εντολή `dlogsig()` πρόκειται για μια συνάρτηση του MATLAB η οποία ανήκει στο πακέτο εργαλείων για νευρωνικά δίκτυα (Deep Learning Toolbox [35]). Η `dlogsig()` δέχεται σαν ορίσματα την είσοδο των νευρώνων ενός επιπέδου, `n1` σε αυτή τη περίπτωση, και την έξοδο του `A1`. Η έξοδος της `dlogsig()` είναι η παράγωγος της σιγμοειδούς συνάρτησης (σχήμα 3.9) που ορίζεται μαθηματικά ως :

$$f'(x) = f(x)(1 - f(x)) \tag{3.5}$$

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.6}$$

Αφού υπολογιστούν οι παράγωγοι του κρυφού επιπέδου (`A1`) και του επιπέδου εξόδου (`A2`), το επόμενο βήμα είναι η αλγοριθμική συγχώνευση του σφάλματος έτσι ώστε να υπολογιστούν οι παράγοντες που θα συμμετάσχουν στο βήμα του backpropagation (ενότητα 2.5). Για το σκοπό αυτό δημιουργήθηκαν οι μεταβλητές `s2` και `s1`. Η `s2` υπολογίζεται ως το εσωτερικό γινόμενο της παραγώγου του σταδίου εξόδου και του υπολογισμένου σφάλματος. Ύστερα το αποτέλεσμα αυτού του γινομένου μπαίνει ως είσοδος στον υπολογισμό της παραμέτρου `s1`, ως εκ τούτου υλοποιώντας το πρώτο κομμάτι του backpropagation.

Στη συνέχεια, αφού έχουν υπολογιστεί το ποσοστό διόρθωσης (`s1`, `s2`) των βαρών, έρχεται το στάδιο της ανανέωσης των τιμών τους. Η ανανέωση αυτή φαίνεται στις δύο

αναθέσεις των `W1` και `W2` όπου ο κώδικας που εκτελείται αφαιρεί από τις παλιές τιμές των βαρών τα μεγέθη `0.005*(s2*A1(:,i)')` για τα βάρη `W2` και `0.005*(s1*P(:,i)')` για τα `W1`. Με αυτές τις αναθέσεις ολοκληρώνεται ο αλγόριθμος του backpropagation. Ο παράγοντας 0.005 που εμφανίζεται και στις δύο αναθέσεις πρόκειται για τον ρυθμό εκμάθησης η που έχει οριστεί στην ενότητα 2.5. Η τιμή του ρυθμού εκμάθησης έχει επιλεγεί μετά από πειραματισμούς με στόχο τη ταχύτερη σύγκλιση του αλγορίθμου. Οι επόμενες δύο γραμμές κώδικα εφαρμόζουν τη συνάρτηση ενεργοποίησης (logsigma) επανυπολογίζοντας τις τιμές των νευρώνων (`A1`, `A2`) σύμφωνα με τις ανανεωμένες τιμές των βαρών του νευρωνικού δικτύου. Ο αλγόριθμος αυτός του backpropagation θα επαναληφθεί `Q` φορές μέσα σε μια εποχή (epoch), όπου `Q` είναι ο αριθμός των υποεικόνων (12144 στη συγκεκριμένη περίπτωση).

Κάθε φορά που ολοκληρώνεται ο αλγόριθμος του backpropagation για μία εποχή υπολογίζεται το ανανεωμένο σφάλμα του δικτύου. Όπως και στο στάδιο της αρχικοποίησης το σφάλμα ορίζεται ως η σχέση του μέσου τετραγωνικού σφάλματος (MSE) της διαφοράς πινάκων `T-A2` όπου συνεπάγεται με την απόσταση της εκτιμώμενης απάντησης του δικτύου και των στόχων (targets) που έχουν υπολογιστεί στην ενότητα 3.4. Όπως αναφέρθηκε και προηγουμένως, ο αλγόριθμος της εκπαίδευσης ολοκληρώνεται όταν το σφάλμα γίνει μικρότερο της ανοχής 10^{-5}, ή όταν οι επαναλήψεις γίνουν περισσότερες από τον μέγιστο αριθμό εποχών που έχει οριστεί ως 100000 στη συγκεκριμένη υλοποίηση.

Είναι κατανοητό ότι εάν το σφάλμα δεν φτάσει ποτέ το 10^{-5} και ολοκληρωθούν οι 100000 επαναλήψεις τότε ο εμφωλιασμένος βρόγχος θα εκτελέσει τον αλγόριθμο του backpropagation $1,2144 \cdot 10^{10}$ φορές ($Q \times epochs = 100000 \times 12144 = 1.2144 \cdot 10^{10}$). Ευτυχώς, η επιλογή των 100000 επαναλήψεων είναι μια πολύ πεσιμιστική προσέγγιση και ο αλγόριθμος συγκλίνει σε πολύ λιγότερες επαναλήψεις. Παρόλα αυτά είναι εμφανές ότι ο χρόνος εκτέλεσης του backpropagation είναι ύψιστης σημασίας για την υλοποίηση ενός πρακτικού νευρωνικού δικτύου.

Για τη σχεδίαση μιας γρήγορης υλοποίησης backpropagation απαιτείται μια έξυπνη προσέγγιση με τη σειρά εκτέλεσης των εσωτερικών γινομένων. Προαπαιτούμενη γνώση για την κατανόηση της επόμενης ιδέας είναι ότι για την εκτέλεση του εσωτερικού γινομένου δύο πινάκων A και B με διαστάσεις (n, m) και (m, p) αντίστοιχα απαιτούνται $n \times m \times p$ υπολογισμοί. Δεδομένου αυτού, η βασική ιδέα της βελτιστοποίησης είναι η εξής, εάν πρέπει να εκτελεστούν διαδοχικά εσωτερικά γινόμενα, έστω $A * B * C$ όπου το μέγεθος των πινάκων είναι, $A : (n, m)$, $B : (m, p)$ και $C : (p, q)$ τότε ο αριθμός των πράξεων που θα εκτελεστούν είναι $n \times m \times p$ για το πρώτο γινόμενο $A * B$ και $n \times p \times q$ για το δεύτερο γινόμενο φτάνοντας τελικά σε ένα σύνολο πράξεων που υπολογίζεται ως :

$$(n \times m \times p) + (n \times p \times q) = n \times p \times (m + q) \tag{3.7}$$

Στη συνέχεια είναι κατανοητό πως τα γινόμενα $A * B * C$ και $A * (B * C)$ παράγουν το ίδιο αποτέλεσμα. Στη περίπτωση όπου γίνεται χρήση παρενθέσεων ορίζεται ότι το πρώτο γινόμενο που εκτελείται είναι το $(B * C)$ και στη συνέχεια εκτελείται το γινόμενο του πίνακα A και του αποτελέσματος της πράξης $(B * C)$. Εάν οι διαστάσεις των πινάκων είναι ίδιες με αυτές του προηγούμενου παραδείγματος τότε ο αριθμός των

πράξεων της έκφρασης $A * (B * C)$ είναι :

$$m \times q \times (n + p) \tag{3.8}$$

Όπως αποδείχθηκε, η επιλογή της σειράς των εσωτερικών γινομένων έχει άμεση σχέση με τον αριθμό των υπολογισμών και κατ' επέκταση της πολυπλοκότητας. Για αυτό το λόγο στον κώδικα που παρουσιάστηκε προηγουμένως, έχει γίνει η χρήση των παρενθέσεων στις αλυσίδες εσωτερικών γινομένων με στόχο τη μείωση των υπολογισμών και ως συνέπεια την βελτιστοποίηση του χρόνου εκτέλεσης.

Αυτή ήταν η πρώτη τεχνική για τη βελτιστοποίηση του χρόνου εκτέλεσης. Για τη δεύτερη τεχνική, η οποία είναι και πιο σημαντική, γίνεται η εκμετάλλευση της μεγάλης δυνατότητας παράλληλων υπολογισμών της κάρτας γραφικών (Graphics Processing Unit - GPU). Όλες οι εντολές της γλώσσας του MATLAB εκτελούνται με τη χρήση του επεξεργαστή. Οι κάρτες γραφικών, από τη φύση του πως χειρίζονται τα γραφικά, απαιτούν μεγάλη χρήση παραλληλισμού και αυτό επιτυγχάνεται με πολλού παράλληλους πυρήνες CUDA (Compute Unified Device Architecture) που εμφανίζονται σε κάρτες γραφικών NVIDIA [36]. Για παράδειγμα, ο υπολογιστής που χρησιμοποιήθηκε σε αυτή την εργασία είχε επεξεργαστή (Intel Core i7-9700K) με 8 πυρήνες ενώ η κάρτα γραφικών (GeForce RTX™ 2060) 1920 CUDA πυρήνες δίνοντας πολύ μεγαλύτερη δυνατότητα παραλληλισμού.

Τα νευρωνικά δίκτυα είναι εγγενώς αλγόριθμοι υψηλού παραλληλισμού και οι περισσότερες εφαρμογές εύρεσης κρατήρων στη βιβλιογραφία (για παράδειγμα οι [29], [30]) κάνουνε χρήση καρτών γραφικών για το στάδιο της εκπαίδευσής τους και είναι άξιο να σημειωθεί πως ακόμα και κάρτες γραφικών γενικού σκοπού, σαν τη κάρτα που χρησιμοποιήθηκε σε αυτή την εργασία, παράγουν αποτελέσματα με τάξεις μεγέθους καλύτερο χρόνο εκτέλεση από ότι οι επεξεργαστές.

Το περιβάλλον του MATLAB παρέχει τη σουίτα παράλληλου προγραμματισμού Parallel Computing Toolbox [37], η οποία δίνει τη δυνατότητα στους χρήστες να κατανείμουν δύσκολες υπολογιστικά διαδικασίες (όπως η εκπαίδευση του νευρωνικού δικτύου) σε παράλληλα υπολογιστικά συστήματα (clusters) αλλά και να κάνουν χρήση της κάρτας γραφικών ενός υπολογιστή. Αυτό επιτυγχάνεται με τη χρήση των εντολών `gpuArray` και `gather`. Η εντολή `gpuArray` δέχεται σαν όρισμα έναν πίνακα και τον αποθηκεύει στη κάρτα γραφικών επιστρέφοντας ένα αντικείμενο (object) τύπου *gpuArray*. Στη συνέχεια με τη χρήση αυτού του αντικειμένου γίνονται όλες οι πράξεις αξιοποιώντας το παραλληλισμό της κάρτας γραφικών.

Για να επιστραφούν τα δεδομένα από τη μνήμη της κάρτας γραφικών πίσω στον επεξεργαστή χρησιμοποιείται ή εντολή `gather`. Η `gather` δέχεται σαν όρισμα ένα αντικείμενο τύπου *gpuArray* και επιστρέφει τις τιμές του πίνακα εισόδου από τη κάρτα γραφικών στον επεξεργαστή. Η `gather` είναι εν γένει μια ακριβή υπολογιστικά διαδικασία, έτσι εκτελείται μόνο όταν είναι απαραίτητη, όπως για παράδειγμα για να τυπωθούν κάποια δεδομένα στο χρήστη. Βέβαια, για να είναι αποτελεσματική η μέθοδος του παράλληλου προγραμματισμού με τη κάρτα γραφικών, τα μεγέθη των πινάκων πρέπει να είναι μεγάλα, για παράδειγμα δύο πίνακες με διαστάσεις 1000×1000 έχουν περίπου τον ίδιο χρόνο εκτέλεσης με επεξεργαστή και κάρτα γραφικών. Ευτυχώς, οι πίνακες που χρησιμοποιούνται σε αυτή την εργασία έχουν αρκετά μεγαλύτερο μέγεθος

από το προηγούμενο παράδειγμα, και έτσι η χρήση της κάρτας γραφικών παράγει δραματικά μικρότερο χρόνο εκτέλεσης.

Για τη σύγκριση του χρόνου εκτέλεσης του αλγορίθμου backpropagation με χρήση επεξεργαστή και με χρήση της κάρτας γραφικών χρησιμοποιήθηκε το ζευγάρι εντολών `tic`, `toc`. Αυτές οι δύο εντολές χρησιμοποιούνται μαζί και περικλείουν ένα μπλοκ εντολών, στη περίπτωση αυτής της εργασίας το backpropagaion. Ο σκοπός τους είναι η μέτρηση του χρόνου που απαιτείται για την εκτέλεση ενός μπλοκ εντολών :

```
tic
.... Some Commands
toc

>> Elapsed time is x seconds.
```

Η μέτρηση του χρόνου εκτέλεσης του backpropagation με τη χρήση των `tic`, `toc` υπολογίστηκε ως 0.8 δευτερόλεπτα με τη χρήση του επεξεργαστή και 16.3 milliseconds με τη χρήση της κάρτας γραφικών. Από αυτό το παράδειγμα φαίνεται η υπεροχή της κάρτας γραφικών καθώς ο χρόνος εκτέλεσης του ίδιου μπλοκ κώδικα μειώθηκε κατά έναν παράγοντα περίπου ίσο με 500. Επιπρόσθετα, καθώς ο αλγόριθμος του backpropagation θα εκτελείται 12144 φορές σε κάθε εποχή (επανάληψη), φαίνεται ότι η χρήση της κάρτας γραφικών είναι απαραίτητη μειώνοντας το χρόνο εκμάθησης μίας εποχής από 2.6 ώρες σε μόλις 3.3 δευτερόλεπτα. Με αυτό το σχόλιο ολοκληρώνεται η παρουσίαση της υλοποίησης του αλγορίθμου εκμάθησης του νευρωνικού δικτύου. Η επόμενη ενότητα είναι αφιερωμένη στη παρουσίαση της εκτέλεσης του αλγορίθμου και τα πειραματικά αποτελέσματα.

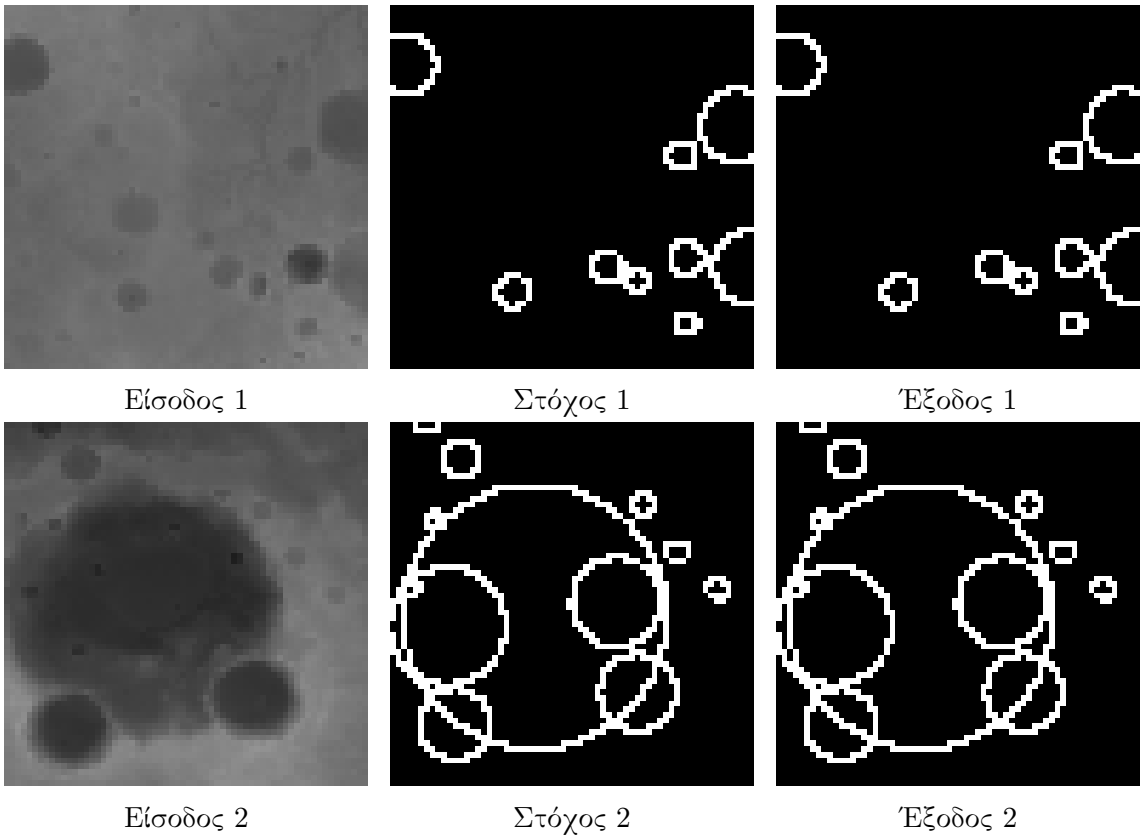
3.6 Πειραματικά Αποτελέσματα

3.6.1 Εκπαίδευση Χωρίς Επαλήθευση

Από το σύνολο των εικόνων που προέκυψαν κατά τη δημιουργία του dataset εισόδου, αρχικά χρησιμοποιήθηκε το 90% αυτών για το στάδιο της εκπαίδευσης και το υπολειπόμενο 10% χρησιμοποιήθηκε για την επαλήθευση της λειτουργίας του δικτύου σε άγνωστες εισόδους. Ο κώδικας που εφαρμόζει το εκπαιδευμένο νευρωνικό δίκτυο στις εικόνες δοκιμών είναι ο εξής :

```
threshold=0.9;
N = testImages;
n1_test = W1 * N;
A1_test = logsig(n1_test);
n2_test = W2 * A1_test;
A2_test = logsig(n2_test);
NNOut = real(A2_test > threshold);
```

Στον παραπάνω κώδικα η μεταβλητή `testImages` αποτελεί έναν τένσορα με τις εικόνες εισόδου, στη συνέχεια οι επόμενες γραμμές εφαρμόζουν τα βάρη `W1` και `W2`



Είσοδος 1 Στόχος 1 Έξοδος 1

Είσοδος 2 Στόχος 2 Έξοδος 2

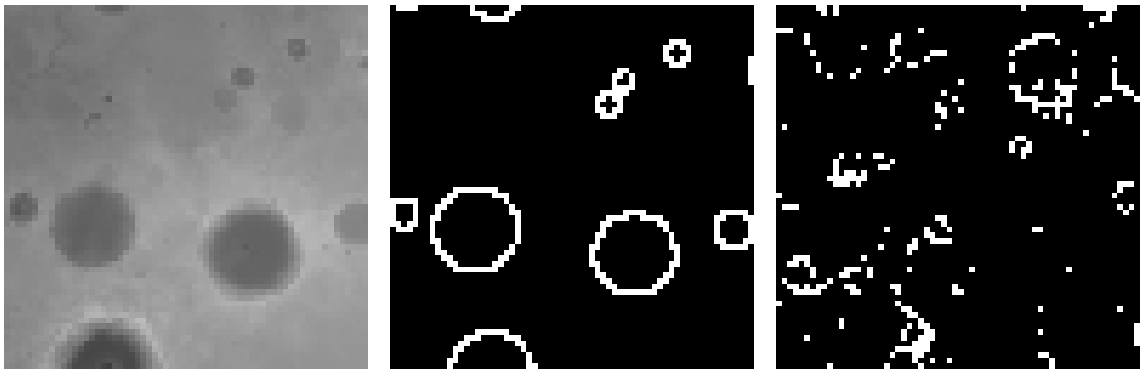
Σχήμα 3.10: Δύο παραδείγματα εισόδου από το dataset εκπαίδευσης και η σύγκριση των αντίστοιχων στόχων (targets) με την έξοδο του νευρωνικού δικτύου.

που υπολογίστηκαν στο στάδιο της εκπαίδευσης πάνω στις εικόνες εισόδου όπου τελικά το αποτέλεσμα αποθηκεύεται στη μεταβλητή `NNOut` αφότου εφαρμοστεί ένα απλό φίλτρο κατωφλίου (threshold) για τη μείωση του θορύβου. Όσο πιο μεγάλη τιμή χρησιμοποιείται για το κατώφλι τόσο μεγαλύτερη και η ακρίβεια του δικτύου.

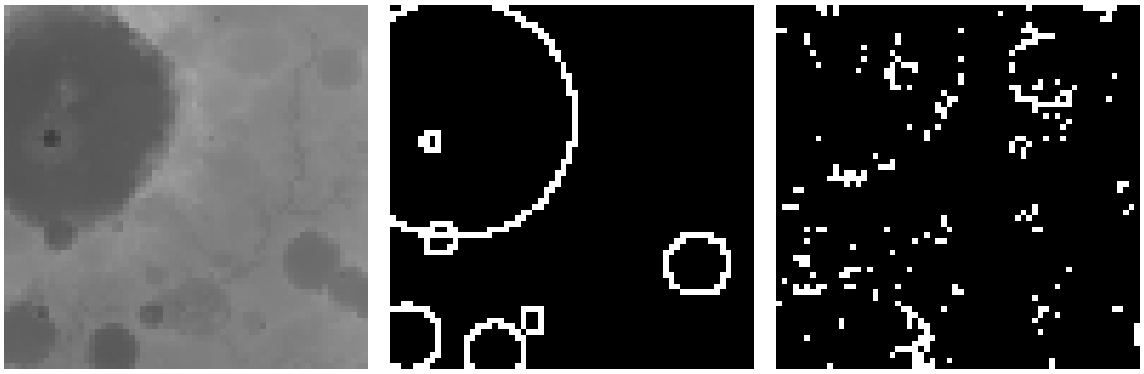
Η πρώτη δοκιμή του νευρωνικού δικτύου γίνεται χρησιμοποιώντας ένα υποσύνολο από τις εικόνες εκπαίδευσης (τένσορας `P`). Τα αποτελέσματα που παράγονται από την εφαρμογή του δικτύου πάνω στον τένσορα εισόδου φαίνονται στις εικόνες του σχήματος 3.10.

Οι εικόνες παρουσιάζουν δύο παραδείγματα της εφαρμογής του νευρωνικού πάνω σε εικόνες εισόδου που αποτέλεσαν κομμάτι της εκπαίδευσης. Οι κεντρικές εικόνες είναι οι στόχοι της εκάστοτε εικόνας εισόδου και στη τελευταία στήλη παρουσιάζεται η έξοδος του νευρωνικού δικτύου. Οπτικά φαίνεται ότι ο αλγόριθμος εντόπισε ακριβώς τους κρατήρες που εμφανίζονται και στις εικόνες στόχων. Για την επιβεβαίωση αυτού χρησιμοποιείται ο παρακάτω κώδικας :

```
wrong = size( find(NNOut - Target), 1);
recognition_rate = 100 * ( size(N,1) - wrong ) / size(N,1)

>> recognition_rate =

    100
```



Είσοδος 1 Στόχος 1 Έξοδος 1



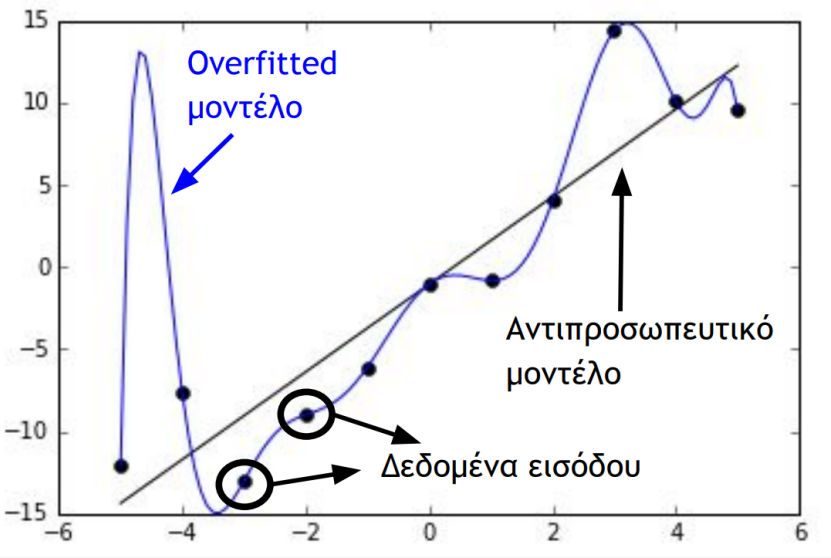
Είσοδος 2 Στόχος 2 Έξοδος 2

Σχήμα 3.11: Δύο παραδείγματα άγνωστων εισόδου και η σύγκριση των αντίστοιχων στόχων (targets) με την έξοδο του νευρωνικού δικτύου.

Στον παραπάνω κώδικα αφαιρείται ο πίνακας στόχων από τον πίνακα εξόδου `NNOut` και το αποτέλεσμά τους χρησιμοποιείται σαν είσοδος στη συνάρτηση `find` όπου επιστρέφει τιμές διαφορετικές του μηδενός. Έτσι στη περίπτωση όπου κάποιο στοιχείο (pixel) από τις εικόνες εξόδου διαφέρει από του στόχους τότε η εντολή `find` θα το εντοπίσει. Στη συνέχεια τα μη μηδενικά στοιχεία (εσφαλμένα pixels) μετρώνται από τη συνάρτηση `size` και αποθηκεύονται στη μεταβλητή `wrong`. Τέλος η μεταβλητή `recognition_rate` περιέχει το ποσοστό των σωστών στοιχείων (pixels). Η παραπάνω εκτέλεση επιστρέφει σαν αποτέλεσμα 100% επιτυχία στις εικόνες εκπαίδευσης.

Αφού παρατηρήθηκε επιτυχία στην εκπαίδευση του δικτύου, το επόμενο βήμα είναι η δοκιμή του δικτύου με εισόδους τις άγνωστες εικόνες που περιέχονται στο 10% του dataset που δεν χρησιμοποιήθηκε στη διαδικασία της εκπαίδευσης. Ο κώδικας που εφαρμόζεται είναι ο ίδιος με τη προηγούμενη περίπτωση με τη μόνη διαφορά πως αυτή τη φορά ο τένσορας `testImages` περιέχει άγνωστες εικόνες. Το αποτέλεσμα του κώδικα φαίνεται στις εικόνες του σχήματος 3.11.

Αυτή τη φορά παρατηρείται ότι η έξοδος του νευρωνικού δικτύου δεν ταιριάζει με τις εικόνες στόχων όπως στη προηγούμενη εκτέλεση του αλγορίθμου. Αντιθέτως, εμφανίζονται διάσπαρτα λευκά pixels τα οποία δεν συσχετίζονται με τον εντοπισμό των κρατήρων στις δύο εικόνες εισόδου αλλά μοιάζουν πιο πολύ με θόρυβο. Η εκτέλεση του κώδικα για τον υπολογισμό του ποσοστού επιτυχίας αυτή τη φορά επιστρέφει 0% επιτυχία το οποίο σημαίνει ότι το νευρωνικό δίκτυο αποτυγχάνει να εντοπίσει κρατήρες



Σχήμα 3.12: Παράδειγμα υπερπροσαρμοσμένου μαθηματικού μοντέλου. Οι κουκκίδες αναπαριστούν τα δεδομένα εισόδου, η μαύρη γραμμή ένα αντιπροσωπευτικό μοντέλο και η μπλέ γραμμή ένα υπερπροσαρμοσμένο μοντέλο.

σε εικόνες εισόδου όπου δεν χρησιμοποιήθηκαν στο στάδιο της εκπαίδευσης. Αυτό είναι ένα γνωστό φαινόμενο στον τομέα της τεχνητής νοημοσύνης όπου ονομάζεται **υπερπροσαρμογή** (**overfitting**).

3.6.2 Υπερπροσαρμογή

Ο όρος υπερπροσαρμογή (overfitting) εμφανίζεται στη στατιστική και αναφέρεται σε οποιοδήποτε στατιστικό μοντέλο το οποίο εφαρμόζεται με μεγάλη ακρίβεια πάνω σε ένα συγκεκριμένο σύνολο δεδομένων με συνέπεια να μην επαληθεύεται με πρόσθετα δεδομένα ή να προβλέψει μελλοντικά δεδομένα αξιόπιστα [38]. Επομένως η υπερπροσαρμογή έχει ως αποτέλεσμα τη κακή γενίκευση του μοντέλου. Ένα παράδειγμα υπερπροσαρμοσμένου μοντέλου εμφανίζεται στο σχήμα[2] 3.12.

Σε αυτό το παράδειγμα οι κουκίδες αντιπροσωπεύουν τις εισόδους, η ευθεία μαύρη γραμμή ένα αντιπροσωπευτικό μοντέλο για αυτά τα δεδομένα και η μπλέ γραμμή παρουσιάζει το παράδειγμα ενός υπερπροσαρμοσμένο μοντέλου. Η λύση στο πρόβλημα αυτό συνήθως απαιτεί τη χρήση ενός μεγαλύτερου σετ δεδομένων. Στη περίπτωση της μηχανικής μάθησης βέβαια υπάρχουν τεχνικές οι οποίες έχουν ως στόχο την επίλυση αυτού του προβλήματος. Έτσι εκτός από τη χρήση ενός μεγαλύτερου dataset εκπαίδευσης μπορούν να εισαχθούν δύο επιπλέον βήματα στη διαδικασία της εκπαίδευσης που στοχεύουν την αύξηση της γενίκευσης. Αυτά τα βήματα είναι η χρήση χωρικών μετασχηματισμών στις εικόνες εκπαίδευσης και η εισαγωγή του βήματος της **επαλήθευσης** (**validation**) στη ροή της εκπαίδευσης [29], [30].

Ο όρος χωρικός μετασχηματισμός αναφέρεται απλώς στις διαδικασίες της περιστροφής (rotation) και της αναστροφής (flip) των εικόνων. Εφαρμόζοντας αυτούς

[2]https://upload.wikimedia.org/wikipedia/commons/6/68/Overfitted_Data.png

τους μετασχηματισμούς στις εικόνες εκπαίδευσης σε τυχαίο χρονικό σημείο κατά τη διαδικασία της εκπαίδευσης, το νευρωνικό δίκτυο αποκτά τη δυνατότητα της γενίκευσης βελτιώνοντας το πρόβλημα της υπερπροσαρμογής χωρίς τη χρήση περισσότερων εικόνων εκπαίδευσης.

Το στάδιο της επαλήθευσης (validation) είναι μία διαδεδομένη τεχνική στο σχεδιασμό νευρωνικών δικτύων. Η τεχνική του validation εφαρμόζεται παράλληλα με τη διαδικασία της εκπαίδευσης κατά την οποία εισάγονται τυχαίες εικόνες που δεν ανήκουν στο dataset εκπαίδευσης. Με αυτό το τρόπο ενισχύεται η γενίκευση καθώς το νευρωνικό δίκτυο εκπαιδεύεται στο να εντοπίζει πιο γενικά χαρακτηριστικά στις εικόνες. Από το συνολικό dataset πλέον χρησιμοποιείται και ένα ποσοστό για τη διαδικασία της επαλήθευσης, άρα πλέον υπάρχουν τρία datasets για την εκπαίδευση, την επαλήθευση και τις δοκιμές.

Τα βήματα των χωρικών μετασχηματισμών και της επαλήθευσης παρουσιάζονται στο παρακάτω κομμάτι κώδικα :

```
...
[RV,QV]=size(V);
for  itr =1:epochs
    if error <= goal_err
        break
    else
    op = randi([1 10],1,1); % Get random number for next operation
    if( op == 1 ) % 90 degres rotation
        Pin_=imrotate(reshape(P(:,i),[segmentSize segmentSize]),90);
        Tin_=imrotate(reshape(T(:,i),[segmentSize segmentSize]),90);
        Pin=Pin_(:);
        Tin=Tin_(:);
    elseif( op == 2 ) % 180 degres rotation
        Pin_=imrotate(reshape(P(:,i),[segmentSize segmentSize]),180);
        Tin_=imrotate(reshape(T(:,i),[segmentSize segmentSize]),180);
        Pin=Pin_(:);
        Tin=Tin_(:);
    elseif( op == 3 ) % 270 degres rotation
        Pin_=imrotate(reshape(P(:,i),[segmentSize segmentSize]),270);
        Tin_=imrotate(reshape(T(:,i),[segmentSize segmentSize]),270);
        Pin=Pin_(:);
        Tin=Tin_(:);
    elseif( op == 4 ) % Get Image From Validation Set
        l=randi([1 QV],1,1);  % Get random image
        Pin=V(:,l);
        Tin=TV(:,l);
    else % Do no operation - Continue with normal Training Set
        Pin=P(:,i);
        Tin=T(:,i);
    end
    df1 = dlogsig(n1,A1(:,i));
...
```

Αρχικά ορίζεται ένας πίνακας `V` ο οποίος περιέχει τυχαίες εικόνες του dataset που θα χρησιμοποιηθούν για το βήμα της επαλήθευσης. Σε αυτή τη περίπτωση το 70% των εικόνων του dataset χρησιμοποιήθηκε για την εκπαίδευση, το 20% για την επαλήθευση και τέλος οι υπολειπόμενες εικόνες για τις δοκιμές με άγνωστες εικόνες. Ο κώδικας ορίζει μια μεταβλητή `op` η οποία αποθηκεύει τυχαίους αριθμούς από το 1 έως το 10. Αυτοί οι αριθμοί παράγονται από την εντολή του MATLAB `rand()`. Ανάλογα με τον αριθμό που επιστρέφεται από τη συνάρτηση `rand()` η ροή της εκπαίδευσης επηρεάζεται και εκτελεί μία από τις παρακάτω διεργασίες :

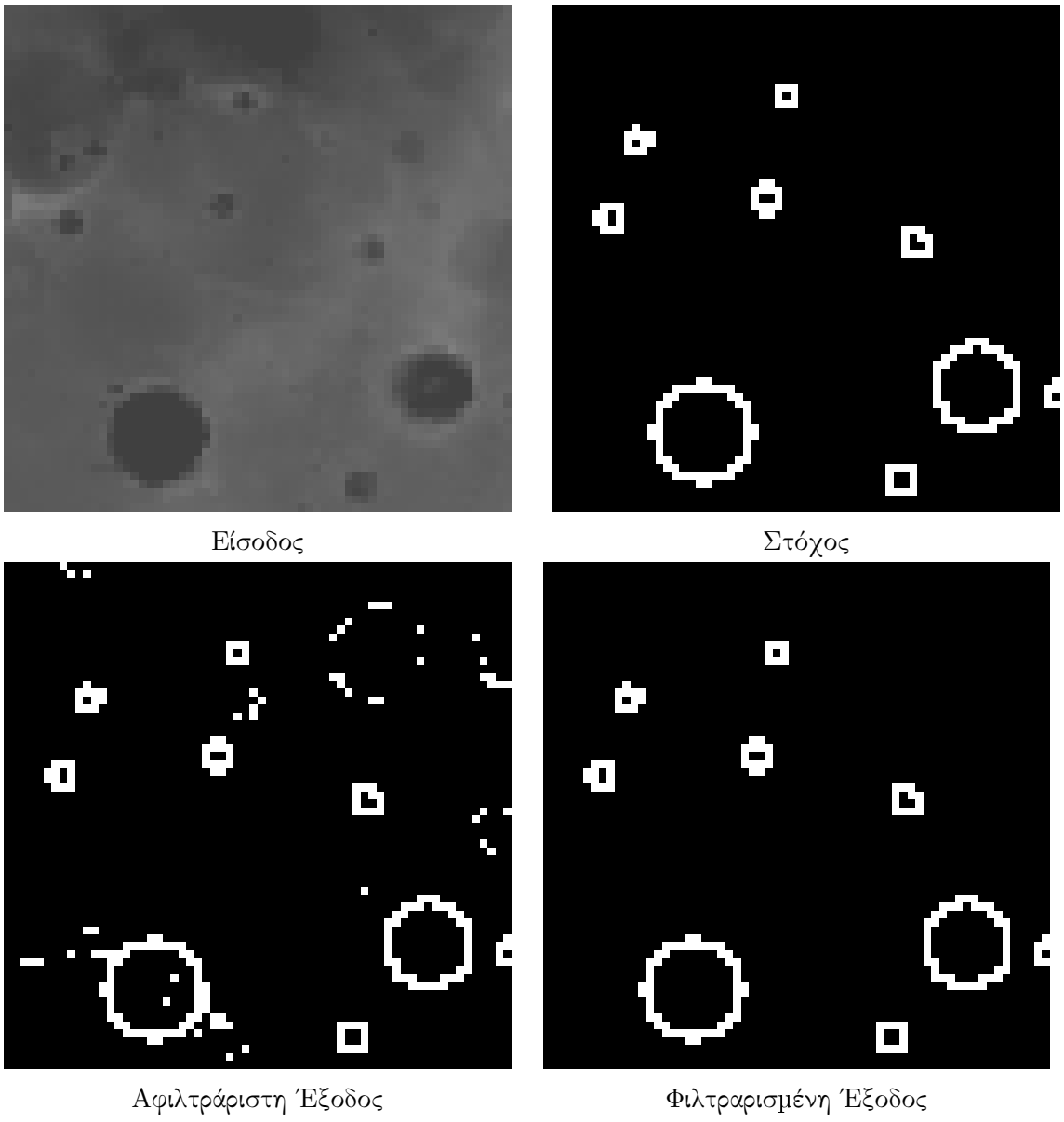
Τιμή op	**Διεργασία**	**Πιθανότητα**
1	Περιστροφή 90°	10%
2	Περιστροφή 180°	10%
3	Περιστροφή 270°	10%
4	Επαλήθευση	10%
5-10	Κανονική Λειτουργία	60%

Επομένως, σύμφωνα με τον πίνακα, στις περισσότερες επαναλήψεις (60%) δεν επηρεάζεται ο αλγόριθμος της εκπαίδευσης. Στο 30% των επαναλήψεων εφαρμόζεται ο χωρικός μετασχηματισμός της περιστροφής σε κάποια από τις εικόνες εκπαίδευσης και στο υπολειπόμενο 10% εισάγεται στον αλγόριθμο μία τυχαία εικόνα επαλήθευσης. Η διεργασία της περιστροφής επιτυγχάνεται με τη χρήση της συνάρτησης `imrotate()` του MATLAB. Δέχεται σαν όρισμα έναν πίνακα-εικόνα και έναν αριθμό που αντιστοιχεί στις μοίρες περιστροφής. Στη συγκεκριμένη υλοποίηση, δεδομένου ότι οι εικόνες είναι αποθηκευμένες σε τένσορες με διάσταση `1 x N`, πρέπει πρώτα να εφαρμοστεί η συνάρτηση του MATLAB `reshape` η οποία μετατρέπει τον τένσορα σε πίνακα. Δέχεται σαν όρισμα τον τένσορα και ένα διάνυσμα με τις διαστάσεις του πίνακα εξόδου. Οι διαστάσεις του πίνακα εξόδου ορίζονται από τη μεταβλητή `segmentSize` όπου σε αυτή την υλοποίηση έχει οριστεί `64 x 64`. Αφότου εκτελεστεί ο έλεγχος για το είδος της διεργασίας που θα εφαρμοστεί πάνω στην εκάστοτε εικόνα εκπαίδευσης, ο αλγόριθμος του backpropagation είναι ακριβός ο ίδιο όπως αυτός στην ενότητα 3.5.

3.6.3 Εκπαίδευση Με Επαλήθευση

Μετά την μετατροπή του αλγορίθμου για την αποφυγή της υπερπροσαρμογής, από το σύνολο των εικόνων που προέκυψαν κατά τη δημιουργία του dataset εισόδου, πλέον χρησιμοποιείται το 70% αυτών για το στάδιο της εκπαίδευσης, το 20% για το νέο βήμα της επαλήθευσης και το υπολειπόμενο 10% χρησιμοποιήθηκε πάλι για τις δοκιμές του δικτύου σε άγνωστες εισόδους. Όπως είναι εύκολα αντιληπτό, οι επιπλέον διεργασίες για την αποφυγή ενός υπερποσαρμοσμένου μοντέλου οδηγούν σε έναν μεγαλύτερο χρόνο εκτέλεσης του αλγορίθμου της εκπαίδευσης. Ο επιπλέον χρόνος αυτός εξαρτάται από το μέγεθος των σετ εκπαίδευσης και επαλήθευσης, καθώς και τις πιθανότητες που έχουν επιλεχθεί για τις διεργασίες εκτός της κανονικής ροής της εκπαίδευσης που χρησιμοποιήθηκε στην υποενότητα 3.6.1.

Το πρώτο πείραμα εκτελείται χρησιμοποιώντας σαν είσοδο τις εικόνες εκπαίδευσής. Στο σχήμα 3.13 φαίνεται το αποτέλεσμα του αλγορίθμου εντοπισμού κρατήρων. Χαρακτηριστικό παράδειγμα ότι πλέον το μοντέλο έχει αποφύγει την υπερπροσαρμογή είναι



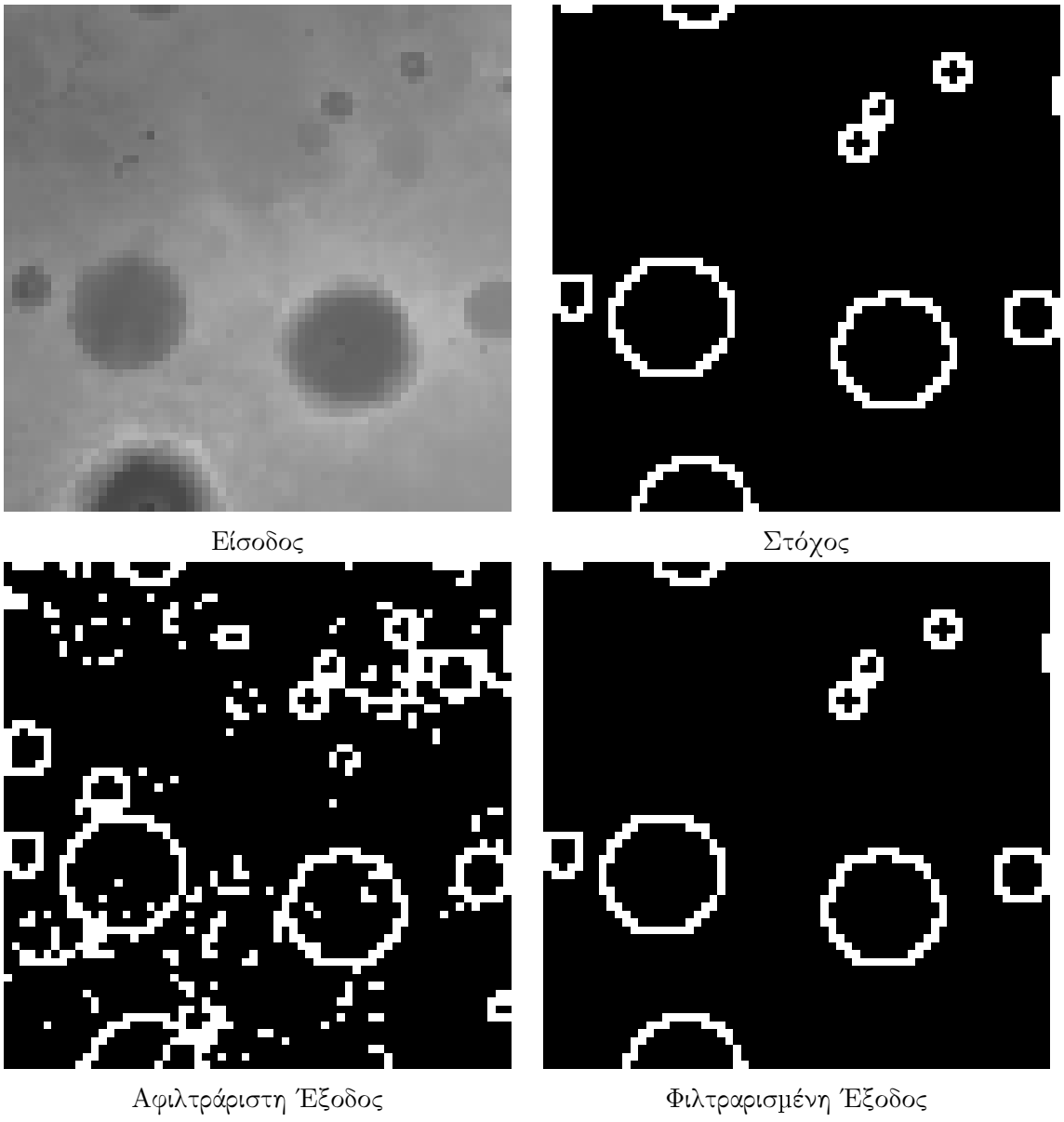
Είσοδος Στόχος

Αφιλτράριστη Έξοδος Φιλτραρισμένη Έξοδος

Σχήμα 3.13: Παράδειγμα εισόδου από το dataset εκπαίδευσης. Παρατηρείται ότι η εικόνα εξόδου απαιτεί φιλτράρισμα για καλύτερη απόδοση.

η αρχική εικόνα εξόδου όπου παρατηρείται ένα φαινόμενο θορύβου με αποτέλεσμα η εικόνα εξόδου να μην είναι ακριβώς ίδια με το στόχο της εικόνας εισόδου. Για την αποφυγή του θορύβου χρησιμοποιείται ένα απλό φίλτρο κατωφλίου όπως φαίνεται στο επόμενο κομμάτι κώδικα :

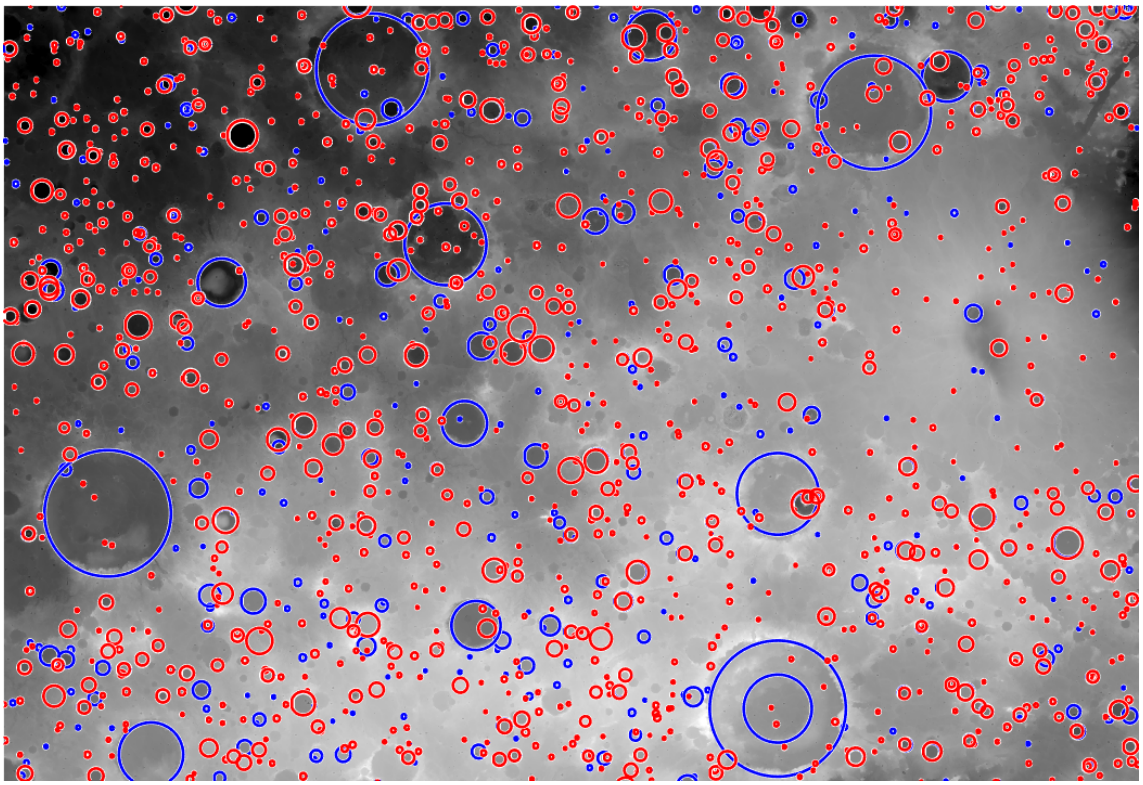
```
% Threshold of the system (higher threshold = more accuracy)
threshold=0.8;
N = testImages;
n1_test = W1 * N;
A1_test = logsig(n1_test);
n2_test = W2 * A1_test;
A2_test = logsig(n2_test);
NNOut = real(A2_test > threshold);
```



Σχήμα 3.14: Παράδειγμα εισόδου από το dataset δοκιμών.

Η μεταβλητή `threshold` έχει τεθεί ώς 0.8 με αποτέλεσμα οι τιμές του τένσορα εξόδου που είναι μικρότερες από αυτό το κατώφλι μηδενίζονται. Ο αριθμός αυτός δείχνει πρακτικά τη βεβαιότητα του αλγορίθμου για τον εντοπισμό κρατήρων. Στη προηγούμενη εκπαίδευση όπου το μοντέλο είχε υπερπροσαρμοστεί στις εικόνες εκπαίδευσης δεν απαιτείται μεγάλη τιμή κατωφλίου καθώς το νευρωνικό δίκτυο είναι "βέβαιο" για το αποτέλεσμα που παράγει το οποίο αναπαριστάται από τιμές κοντά στη μονάδα. Τώρα όμως υπάρχει μια μικρή αβεβαιότητα και έτσι ο αλγόριθμος χρήζει αυτού του φίλτρου. Φυσικά όσο πιο μεγάλο είναι το dataset εκπαίδευσης τόσο λιγότερο φιλτράρισμα απαιτείται.

Στη συνέχεια το δίκτυο δοκιμάζεται με εικόνες άγνωστες σε αυτό, δηλαδή το υπολειπόμενο 10% των εικόνων του dataset. Προηγουμένως παρατηρήθηκε ότι χωρίς το στάδιο της επαλήθευσης, η απόδοση του αλγορίθμου ήταν πολύ κακή. Ένα παράδειγμα εκτέλεσης του νέου μοντέλου παρουσιάζεται στο σχήμα 3.14. Σε αυτή τη περίπτωση



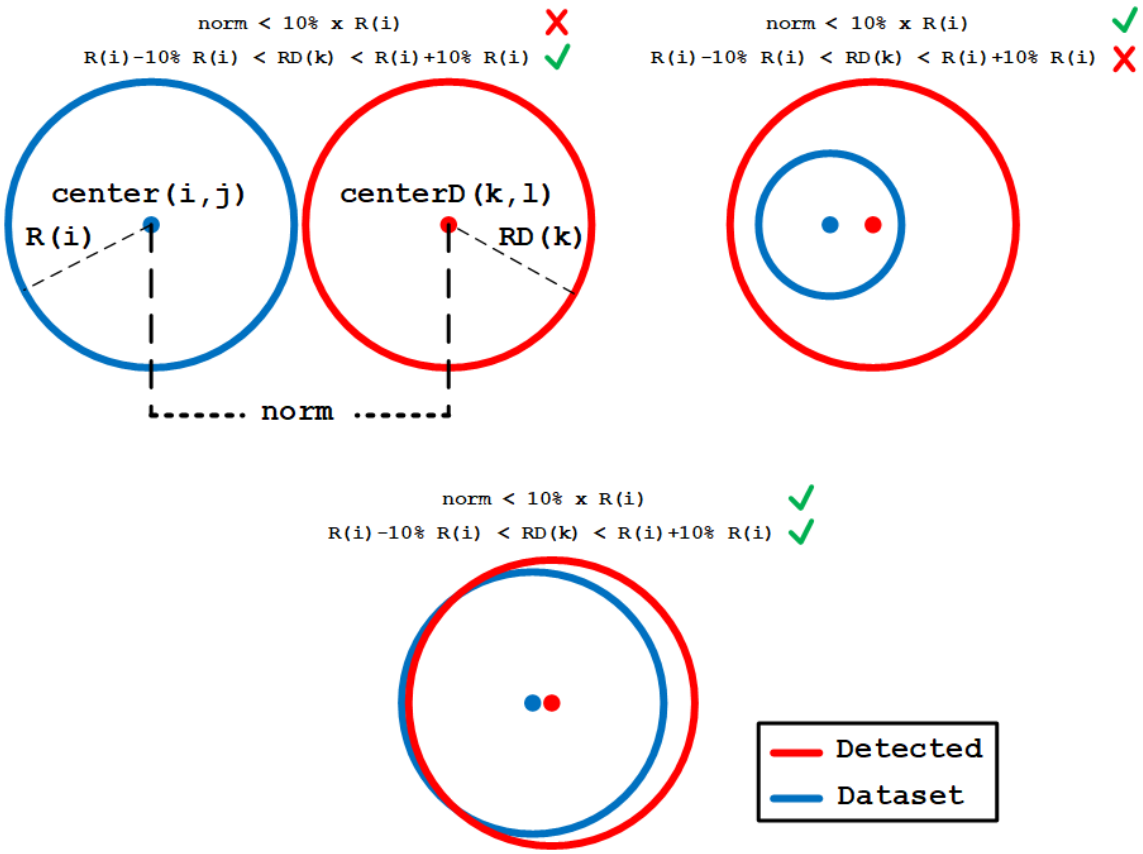
Σχήμα 3.15: Αποτελέσματα του νευρωνικού δικτύου στον εντοπισμό κρατήρων της εικόνας εισόδου (DTM). Με μπλε χρώμα εμφανίζονται οι δεδομένοι κρατήρες και με κόκκινο οι έξοδοι του νευρωνικού δικτύου.

το μοντέλο είχε πολύ μεγαλύτερη επιτυχία στον εντοπισμό των κρατήρων σε σχέση με το αποτέλεσμα του υπερποσαρμοσμένου μοντέλου (σχήμα 3.11). Φυσικά πάλι η έξοδος του δικτύου απαιτεί ένα επιπλέον βήμα φιλτραρίσματος για την αποφυγή του θορύβου. Έτσι αφού εφαρμοστεί το φίλτρο κατωφλίου με τιμή πάλι 0.8 παρατηρείται το αποτέλεσμα της τελευταίας εικόνας η οποία έχει 100% επιτυχία στον εντοπισμό όλων των κρατήρων της υποεικόνας σύμφωνα με τους στόχους (targets) που είχαν τεθεί ως δεδομένα.

Τέλος στο σχήμα 3.15 παρουσιάζεται το αποτέλεσμα του νευρωνικού δικτύου πάνω σε όλη την εικόνα εισόδου DTM. Με μπλε χρώμα είναι κυκλωμένοι οι κρατήρες που εντοπίστηκαν στο στάδιο της δημιουργίας του dataset (ενότητα 3.4) και με κόκκινο οι κρατήρες που εντόπισε το νευρωνικό δίκτυο. Το πρώτο που παρατηρείται είναι ότι το νευρωνικό δίκτυο δεν κατάφερε να εντοπίσει τους κρατήρες με μεγάλη διάμετρο. Στη συγκεκριμένη υλοποίηση αυτό ήταν αναμενόμενο καθώς η κατάτμηση του DTM σε υποεικόνες μεγέθους `64 x 64` δεν καθιστά δυνατό να χρησιμοποιηθούν κρατήρες τέτοιου μεγέθους στο στάδιο της εκπαίδευσης.

3.6.4 Απόδοση Εντοπισμού Κρατήρων

Πριν τον υπολογισμό της απόδοσης του δικτύου σύμφωνα με τις μετρικές της ενότητας 2.7.7, είναι απαραίτητη η υλοποίηση ενός αλγορίθμου που θα ταιριάζει (match) τους κρατήρες που εντόπισε το δίκτυο με τους κρατήρες του dataset εισόδου, έτσι ώστε



Σχήμα 3.16: Παράδειγμα αλγορίθμου ταιριάσματος (match) των εντοπισμένων κρατήρων και των κρατήρων του dataset.

να βρεθούν τα θετικά και αρνητικά αποτελέσματα (ψευδώς ή αληθώς αντίστοιχα). Για να γίνει αυτό, κάθε εντοπισμός του δικτύου πρέπει να συγκριθεί με κάθε κρατήρα του dataset έτσι ώστε να βρεθεί αν υπάρχει αντιστοιχία.

Για τη σύγκριση δύο κρατήρων απαιτούνται τα εξής στοιχεία : συντεταγμένες του κέντρου και ακτίνα. Στη περίπτωση των κρατήρων του dataset αυτά τα δεδομένα έχουν παραχθεί με τον αλγόριθμο της ενότητας 3.4. Στη περίπτωση των εντοπισμένων κρατήρων όμως το δεδομένο είναι μια δυαδική εικόνα που περιέχει τα περιγράμματα των κρατήρων. Δεδομένου όμως ότι τα περιγράμματα αυτά είναι όλα κυκλικές δομές, μπορεί να γίνει ξανά η χρήση του αλγορίθμου της ενότητας 3.4 έτσι ώστε οι διαδοχικές εκτελέσεις της `imfindcircles` θα εντοπίσει τα κέντρα και τις ακτίνες των εντοπισμένων κρατήρων. Η εκτέλεση του αλγορίθμου αυτή τη φορά επιστρέφει όλους τους κρατήρες καθώς στη συγκεκριμένη δυαδική εικόνα δεν υπάρχουν τα προβλήματα φωτισμού και μικρές ακτίνες όπως στη περίπτωση της εικόνας εισόδου (DTM).

Εφόσον τα δεδομένα των εντοπισμένων κρατήρων είναι διαθέσιμα, ο αλγόριθμος για το ταίριασμα των κύκλων είναι ο παρακάτω :

```
matches=0;
for i=1:size(detectedCraters)
    centerDiff = [centerD(i,1)-center(:,1), centerD(i,2)-center(:,2)];
    % Calculate the distance (norm) of all the dataset circles
    % use sqrt(diag()) for vector-wise calculation instead of norm()
```

```
    centerDistance = sqrt(diag(centerDiff * centerDiff'));

    index = find( centerDistance < 0.1*R(i) && ...
                  (RD(i) > R(i) - 0.1*R(i) && RD(i) < R(i) + 0.1*R(i))
          );

    % If index is non-empty --> Crater Match
    if size(index,1) != 0
         matches += 1
    end
end
```

Το παραπάνω κομμάτι κώδικα παρουσιάζει έναν βρόγχο στον οποίο γίνεται το ταίριασμα των εντοπισμένων κύκλων με τους κύκλους του dataset. Οι πίνακες `centerD` και `center` περιέχουν τις συντεταγμένες των κέντρων για τους εντοπισμένους και τους δεδομένους κύκλους (κρατήρες) αντίστοιχα. Επίσης οι πίνακες `RD` και `R` περιέχουν τις ακτίνες των κύκλων.

Ο αλγόριθμος βασίζεται στον υπολογισμό της γεωμετρικής απόστασης μεταξύ των κέντρων. Εάν η απόσταση είναι μικρότερη του 10% της ακτίνας ενός δεδομένου κύκλου του dataset και ταυτόχρονα η ακτίνα του είναι συγκρίσιμη με την ακτίνα του εντοπισμένου κύκλου ($\pm 10\%$) τότε θεωρείται επιτυχημένος εντοπισμός. Παράδειγμα του αλγορίθμου παρουσιάζεται στην εικόνα 3.16. Με μπλε χρώμα παρουσιάζονται οι δεδομένοι κρατήρες και με κόκκινο οι εντοπισμένοι. Η πρώτη εικόνα δείχνει ένα παράδειγμα όπου ο εντοπισμένος κύκλος έχει μεγάλη απόσταση από τον δεδομένο έτσι δεν μπορεί να θεωρηθεί ταίριασμα. Στη δεύτερη εικόνα υπάρχει το παράδειγμα όπου η γεωμετρική απόσταση είναι μικρή αλλά οι ακτίνες δεν είναι συγκρίσιμες, έτσι ούτε εδώ μπορεί να θεωρηθεί ταίριασμα. Η τελευταία εικόνα δείχνει ότι πρέπει να ικανοποιούνται και οι δύο συνθήκες για επιβεβαιωθεί ο επιτυχημένος εντοπισμός κρατήρα.

Στο παραπάνω κώδικα η γεωμετρική απόσταση υπολογίζεται από τις πρώτες δύο γραμμές του βρόγχου. Αρχικά, αφαιρούνται οι συντεταγμένες ενός εντοπισμένου κέντρου από κάθε δεδομένο. Το αποτέλεσμα αυτό αποθηκεύεται στη μεταβλητή `centerDiff`. Στη συνέχεια η επόμενη γραμμή εκτελεί τον υπολογισμό της γεωμετρικής απόστασης σε κάθε γραμμή του πίνακα `centerDiff` και το αποτέλεσμα αποθηκεύεται στον πίνακα `centerDistance`. Η επόμενη γραμμή κάνει χρήση της εντολής `find` του MATLAB η οποία επιστρέφει τους δείκτες γραμμής και στήλης για την οποία ικανοποιείται η συνθήκη που δέχεται σαν όρισμα. Έτσι εάν η συνθήκη του αλγορίθμου ικανοποιείται η μεταβλητή `index` είναι μη μηδενική και ο αλγόριθμος θεωρεί ότι έγινε επιτυχημένος εντοπισμός κρατήρα. Ο αριθμός των επιτυχιών αποθηκεύεται στη μεταβλητή συσσωρευτή (accumulator) `matches`.

Η απόδοση του δικτύου μπορεί να μετρηθεί με τις μετρικές που παρουσιάστηκαν στην ενότητα 2.7.7. Βέβαια το dataset εισόδου δεν περιέχει όλους του κρατήρες που υπάρχουν στην εικόνα (DTM) και το νευρωνικό δίκτυο αναμένεται να εντοπίσει μερικούς από αυτούς τους κρατήρες που δεν υπάρχουν στο dataset. Με τη μετρική F_1 σε αυτή τη περίπτωση το δίκτυο "τιμωρείται" με χαμηλότερο σκορ απόδοσης παρότι κατάφερε και εντόπισε νέους κρατήρες. Όλοι οι νέοι κρατήρες χαρακτηρίζονται ως ψευδά θετικοί εντοπισμοί μειώνοντας τη τιμή του F_1. Ως N ορίζεται άλλη μία

μετρική όπου λαμβάνει υπόψιν τους πιθανούς νέους εντοπισμούς κρατήρων έτσι ώστε το μοντέλο να μην επιβαρύνεται με χαμηλότερες τιμές F_1 παρότι έχει επιτύχει στον αληθή εντοπισμό νέων κρατήρων. Ο ορισμός του N είναι ο εξής :

$$N = \frac{F_P}{F_P + T_R} \tag{3.9}$$

Πρόκειται για τον λόγο των ψευδώς θετικών εντοπισμών ως προς το άθροισμα των ψευδώς θετικών και το σύνολο των κρατήρων του dataset. Ως T_R ορίζεται ο αριθμός των κρατήρων του dataset. Αυτό το μοντέλο κατάφερε να εντοπίσει 161 πραγματικούς κρατήρες οι οποίοι δεν ανήκαν στο dataset. Ο παρακάτω πίνακας συνοψίζει την απόδοση του νευρωνικού δικτύου στον εντοπισμό κρατήρων σε αντιπαραβολή με αντίστοιχα δημοσιευμένα αποτελέσματα ([29], [30]).

Μετρική	Αποτέλεσμα Εργασίας	[29]	[30]
Αριθμός Κρατήρων Dataset	1607	57.564	-
Κρατήρες Νευρωνικού Δικτύου	1414	57.767	-
Αληθείς Εντοπισμοί	1004	42.891	-
Νέοι Κρατήρες	11.5%	21%	42%
Precision	71%	74%	56%
Recall	62.5%	75%	92%
Harmonic Average F_1	66.5%	74%	69%

Από τον παραπάνω πίνακα παρατηρούνται τα εξής στοιχεία. Το μοντέλο αυτής της εργασίας παρότι έχει ένα πολύ μικρότερο dataset από ότι αυτό του [30], έχει μεγάλο ποσοστό επιτυχίας, συγκρίσιμο με αυτό των δημοσιεύσεων. Δεδομένου όμως του μικρότερου dataset είναι λογικό η αποτελεσματικότητά του να είναι μικρότερη. Το ποσοστό των νέων κρατήρων είναι μικρότερο από αυτό των άλλων δύο δημοσιεύσεων

Κεφάλαιο 4

Συμπεράσματα

Αυτή η εργασία παρουσίασε τα βήματα σχεδιασμού και υλοποίησης ενός νευρωνικού δικτύου με στόχο το εντοπισμό κρατήρων σε ψηφιακές εικόνες του πλανήτη Άρη. Αρχικά έγινε μια μικρή εισαγωγή στη φιλοσοφία και τη θεωρία των νευρωνικών δικτύων. Παρουσιάστηκαν τα απαραίτητα βήματα στο σχεδιασμό και την υλοποίηση ενός δικτύου, οι μετρικές απόδοσης καθώς και η μαθηματική πλαισίωση της θεωρίας.

Στη συνέχεια έγινε μια αναφορά σε αντίστοιχες δημοσιεύσεις που συναντήθηκαν στη βιβλιογραφία παρουσιάζοντας μεθόδους απλής επεξεργασίας εικόνας καθώς και

υλοποιήσεις νευρωνικών δικτύων. Δεδομένου ότι οι εικόνες εισόδου ανήκουν σε αρχεία εδάφους DTM, η εργασία συνεχίστηκε με μια παρουσίαση της θεωρίας του DTM. Ύστερα παρουσιάστηκε μία μέθοδος για τη δημιουργία μιας βάσης δεδομένων κρατήρων με τη χρήση επεξεργασίας εικόνων. Η δημιουργία της βάσης αποτελεί το πρώτο βασικό βήμα για την υλοποίηση του νευρωνικού δικτύου έτσι ώστε να χρησιμοποιηθεί για την εκπαίδευση, την επαλήθευση και τον έλεγχο απόδοσης του νευρωνικού δικτύου.

Μετά τη δημιουργία της βάσης δεδομένων κρατήρων παρουσιάζεται η υλοποίηση του νευρωνικού δικτύου. Το δίκτυο αποτελείται από τρία επίπεδα, το επίπεδο εισόδου, το κρυφό επίπεδο και το επίπεδο εξόδου. Στόχος του δικτύου είναι η εκπαίδευσή ενός μοντέλου για τον εντοπισμό κυκλικών χαρακτηριστικών στις εικόνες εισόδου τα οποία παραπέμπουν σε κρατήρες. Κατά τη διαδικασία των πειραμάτων εμφανίστηκε μια γνωστή αδυναμία του νευρωνικού δικτύου, η υπερπροσαρμογή (overfitting). Ο όρος υπερπροσαρμογή (overfitting) αναφέρεται σε οποιοδήποτε στατιστικό μοντέλο το οποίο παρουσιάζει μεγάλη ακρίβεια πάνω σε ένα συγκεκριμένο σύνολο δεδομένων με συνέπεια να μην επαληθεύεται με πρόσθετα δεδομένα ή να προβλέψει μελλοντικά δεδομένα αξιόπιστα. Για την αποφυγή της υπερπροαρμογής, οι υλοποιήσεις εκπαίδευσης νευρωνικών δικτύων χρησιμοποιούν το βήμα της επαλήθευσης (validation). Η τεχνική του validation εφαρμόζεται παράλληλα με τη διαδικασία της εκπαίδευσης κατά την οποία εισάγονται τυχαίες εικόνες που δεν ανήκουν στο dataset εκπαίδευσης. Με αυτό το τρόπο ενισχύεται η γενίκευση καθώς το νευρωνικό δίκτυο εκπαιδεύεται στο να εντοπίζει γενικά χαρακτηριστικά στις εικόνες.

Μετά την εισαγωγή του βήματος της επαλήθευσης παρουσιάστηκε ότι το νευρωνικό δίκτυο απέφυγε την υπερπροσαρμογή και εξάγει καλύτερα αποτελέσματα. Τέλος η εργασία παρουσιάζει τις μετρικές απόδοσης του δικτύου σε αντιπαραβολή με μετρικές αντίστοιχων εργασιών [29], [30]. Η σύγκριση των εργασιών παρουσιάζει ότι το νευρωνικό δίκτυο της εργασίας έχει μεγάλο ποσοστό επιτυχίας, συγκρίσιμο με αυτό των δημοσιεύσεων, παρότι χρησιμοποιεί ένα πολύ μικρότερο dataset εισόδου.

4.1 Προοπτικές

Δεδομένου ότι η μέθοδος που παρουσιάστηκε έχει επιτύχει στον εντοπισμό ενός ικανοποιητικού ποσοστού από τους κρατήρες της εικόνας εισόδου, μία βελτίωση που μπορεί να γίνει είναι η χρήση μεγαλύτερου δείγματος εικόνων για την εκπαίδευση του δικτύου. Στις εφαρμογές νευρωνικών δικτύων ένα μεγάλο δείγμα οδηγεί σε ένα πιο πολύπλοκο μοντέλο με πολύ καλύτερη γενίκευση άρα και σε μεγαλύτερη ακρίβεια. Βέβαια όταν το δείγμα μεγαλώνει έχει επίπτωση στο χρόνο εκτέλεσης του αλγορίθμου της εκπαίδευσης και μάλιστα σε εκθετικό βαθμό. Άρα είναι απαραίτητη η μελέτη του αλγορίθμου έτσι ώστε να βελτιωθεί ο χρόνος εκτέλεσης.

Φυσικά ο χρόνος εκτέλεσης συνδέεται άμεσα με το software και hardware που χρησιμοποιείται. Η γλώσσα προγραμματισμού του MATLAB γενικά είναι ιδανική για μοντελοποίηση και σχεδιασμό πρωτότυπων αλλά δεν είναι αποδοτική σε σχέση με άλλες γλώσσες όπως η Python ή ακόμα καλύτερα η C. Έτσι η μετάβαση σε μία από αυτές τις γλώσσες θα έδιναν τη δυνατότητα στη ταχύτερη εκπαίδευση άρα και στη χρήση ενός μεγαλύτερου dataset. Ταυτόχρονα η χρήση ενός σύγχρονου και πιο αποδοτικού

harware θα παρουσίαζε καλύτερη απόδοση χρόνου.

Τέλος το μοντέλο εντοπισμού κρατήρων στην επιφάνεια του Άρη θα μπορούσε να χρησιμοποιηθεί όχι μόνο για τον εντοπισμό κρατήρων σε εικόνες από άλλα ουράνια σώματα. Επίσης με τη κατάλληλη προσαρμογή θα μπορούσαν να εντοπιστούν και άλλα γεωλογικά χαρακτηριστικά πέρα από τους κρατήρες. Αυτό θα μπορούσε να οδηγήσει σε ένα μοντέλο όπου αυτόματα από τις ψηφιακές εικόνες θα μπορούσε να δημιουργηθεί ένας χάρτης που τονίζει όλα τα γεωλογικά χαρακτηριστικά.

Βιβλιογραφία

[1] J.D. Paola, R.A. Schowengerdt, "Comparisons of neural networks to standard techniques for image classification and correlation", *IEEE International Symposium on Geoscience and Remote Sensing (IGARSS)*, Vol. 3, pp. 1404-1406, Aug. 1994.

[2] J. K. Basu, D. Bhattacharyya, T. H. Kim, "Use of artificial neural network in pattern recognition. International journal of software engineering and its applications", *International journal of software engineering and its applications*, Vol. 4, No. 2, Apr. 2010.

[3] Q. Glaude, "CraterNet: a fully convolutional Neural Network for lunar crater detection based on remotely sensed data", Master Thesis 2017.

[4] S.E. Lauro, E. Pettinelli, G. Caprarelli, et al, "Multiple subglacial water bodies below the south pole of Mars unveiled by new MARSIS data", *Nature Astronomy*, pp. 1-8, Sep. 2020.

[5] B. Joseph, "The Smithsonian Book of Mars", *Smithsonian Institution Press*, 2002.

[6] R.B. Leighton, B.C. Murray, R.P. Sharp, J.D. Allen, R.K. Sloan, "Mariner IV Photography of Mars: Initial Results", *Science*, Vol 149, No. 3684, pp. 627–630, Aug. 1965.

[7] R.B. Leighton, N.H. Horowitz, B.C. Murray, R.P. Sharp, et al, "Mariner 6 and 7 Television Pictures: Preliminary Analysis", *Science*, Vol. 166, No. 3901, pp. 49-67, Oct. 1969.

[8] N.G. Nadine, "Crater size-frequency distributions and a revised Martian relative chronology", *Icarus*, Vol 75, No. 2, pp. 285-305, Aug. 1988.

[9] M. Abadi, P. Barham, J. Chen, Z. Chen, et al, "TensorFlow: A system for large-scale machine learning", *12th USENIX symposium on operating systems design and implementation (OSDI)*, pp. 265-283, Nov. 2016.

[10] R. R. Trippi, E. Turban, "Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance", *McGraw-Hill, Inc.*, 1992.

[11] W. Penny, D. Frost "Penny, Will, and David Frost. "Neural networks in clinical medicine", *Medical Decision Making*, Vol. 16, No. 4, pp. 386-398, Oct. 1996.

[12] B. A. Erol, et al, "Improved deep neural network object tracking system for applications in home robotics", *Computational Intelligence for Pattern Recognition*, Springer Cham, Vol. 777, pp. 369-395, May 2018.

[13] J. L. Hennessy and D. A. Patterson, "A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development", *IEEE International Symposium on Computer Architecture (ISCA)*, Jun. 2018.

[14] V. Sze, et al, "Efficient processing of deep neural networks", *Synthesis Lectures on Computer Architecture*, Vol 15, No. 2, Jun. 2020.

[15] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition", *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, Jun. 2016.

[16] F. S. Panchal, M. Panchal "Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network." *International Journal of Computer Science and Mobile Computing*, Vol 3, No. 11, pp 455-464, Nov 2014.

[17] J. M. Alvarez, M. Salzmann, "Learning the number of neurons in deep networks", *Advances in Neural Information Processing Systems*, Vol. 29, pp. 2270-2278, Dec. 2016.

[18] R. Reedm R. J. Marks II, "Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks", *Mit Press*, 1999.

[19] C. M. Bishop, "Neural Networks for Pattern Recognition", *Oxford university press*, 1995.

[20] Y. LeCun, Y. Bengio, G. Hinton, "Deep learning", *Nature*, Vol 521, No. 7553, pp. 436-444, May 2015.

[21] L. Shapiro, G. Stockman, "Computer Vision", *Prentice-Hall*, 2001.

[22] R. Honda, et al, "Data mining system for planetary images-crater detection and categorization", *Proceedings of the International Workshop on Machine Learning of Spatial Knowledge in conjunction with ICML*, pp. 103-108, Sep. 2000.

[23] D. Meng, C. Yunfeng, W. Qingxian, "Method of passive image based crater autonomous detection", *Chinese Journal of Aeronautics*, Vol 22, No. 3, pp. 301-306, Jun 2009.

[24] T. Bouramtane, et al, "Automatic Detection and Evaluation of Geological linear Features from Remote Sensing Data Using the Hough Transform Algorithm in Eastern Anti-Atlas (Morocco)", *Proceedings of the 2nd International Conference on Computing and Wireless Communication Systems*, pp. 1-6, Nov. 2017.