ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ & ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ

Σχεδιασμός, Ανάπτυξη και Αξιολόγηση Τεχνικών Βαθιάς Μηχανικής Μάθησης για Ταξινόμηση Τηλεπισκοπικών Δεδομένων Υψηλής Χωρικής Ανάλυσης. Υλοποίηση και Σύνδεση με το Orfeo Toolbox

Διπλωματική Εργασία

Παπαδομανωλάκη Μαρία

Αθήνα,

Ιούλιος 2016

NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF RURAL AND SURVEYING ENGINEERING
REMOTE SENSING LABORATORY

Design, Development and Evaluation of Deep Learning-based Classification Frameworks for High Resolution Remote Sensing Data. Implementation and Integration into Orfeo Toolbox

Diploma Thesis,

Papadomanolaki Maria

Athens,

July 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ & ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΤΟΜΕΑΣ ΤΟΠΟΓΡΑΦΙΑΣ
ΕΡΓΑΣΤΗΡΙΟ ΤΗΛΕΠΙΣΚΟΠΗΣΗΣ

Σχεδιασμός, Ανάπτυξη και Αξιολόγηση Τεχνικών Βαθιάς Μηχανικής Μάθησης για Ταξινόμηση Τηλεπισκοπικών Δεδομένων Υψηλής Χωρικής Ανάλυσης. Υλοποίηση και Σύνδεση με το Orfeo Toolbox

Διπλωματική Εργασία

Παπαδομανωλάκη Μαρία

Τριμελής Εξεταστική Επιτροπή:

Κ. Καράντζαλος	**Δ. Αργιαλάς**	**Ν. Δουλάμης**
….....	**….....**	**….....**
Επ. Καθηγητής Ε.Μ.Π	**Καθηγητής Ε.Μ.Π**	**Καθηγητής Ε.Μ.Π**
Επιβλέπων		

Αφιερώνεται
στην οικογένειά μου

ΠΡΟΛΟΓΟΣ

Η παρούσα Διπλωματική Εργασία εκπονήθηκε στα πλαίσια της ολοκλήρωσης των προπτυχιακών σπουδών μου στη Σχολή Αγρονόμων και Τοπογράφων Μηχανικών (ΣΑΤΜ) του Εθνικού Μετσόβιου Πολυτεχνείου (ΕΜΠ) Αθηνών. Το θέμα που διαπραγματεύεται ανατέθηκε από το Εργαστήριο Τηλεπισκόπισης του Τομέα Τοπογραφίας της Σχολής.

Ο Αγρονόμος Τοπογράφος Μηχανικός έχει τη δυνατότητα να ασχοληθεί με μια πληθώρα εφαρμογών που αφορούν την τηλεπισκόπιση, τη γεωδαισία, τη φωτογραμμετρία, το κτηματολόγιο, τα συγκοινωνιακά έργα, τη διαχείριση υδατικών πόρων και πολλά άλλα. Τα τελευταία χρόνια, η πρόοδος της τεχνολογίας έχει επιφέρει ιδιαίτερη ανάπτυξη στις εφαρμογές αυτές, προσφέροντας έτσι στους χρήστες νέες δυνατότητες και προοπτικές.

Πιο ειδικά, ο τομέας της τηλεπισκόπησης σημειώνει ραγδαία ανάπτυξη αφού τα δορυφορικά δεδομένα αυξάνονται τόσο σε ποσότητα όσο και σε ποιότητα, ενώ τα λογισμικά επεξεργασίας εικόνων αυτοματοποιούνται ολοένα και περισσότερο. Η ανάπτυξη αυτή οφείλεται σε μεγάλο βαθμό στα προγράμματα ελεύθερου λογισμικού που επιτρέπουν στους χρήστες την ελεύθερη πρόσβαση στον πηγαίο κώδικα. Όλα αυτά, μου κίνησαν το ενδιαφέρον και με ώθησαν στην επιλογή του θέματος της εργασίας αυτής, το οποίο αφορά τη διερεύνηση των μεθόδων ταξινόμησης Deep Learning και την ενσωμάτωσή τους στο λογισμικό Orfeo Toolbox.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω ιδιαίτερα τα άτομα που βοήθησαν στην περάτωση της εργασίας μου. Αρχικά, ευχαριστώ τον επιβλέποντα καθηγητή μου, Επίκουρο Καθηγητή κύριο Καράντζαλο Κωνσταντίνο, για τη βοήθεια, την καθοδήγηση και το ενδιαφέρον που έδειξε για την εξέλιξή μου. Επίσης, ευχαριστώ τον κύριο Ιωσηφίδη Χρήστο, Ε.ΔΙ.Π, που μου μίλησε για το ελεύθερο λογισμικό. Τέλος, ευχαριστώ ιδιαιτέρως τη Βακαλοπούλου Μαρία, Υποψήφια Διδάκτωρ, η οποία συνέβαλε σημαντικά στην εξέλιξη της εργασίας μου, μεταλαμπάδευσε σε μένα τις γνώσεις της, με υποστήριξε συναισθηματικά και ήταν πάντα διαθέσιμη για την επίλυση των αποριών μου.

ΠΕΡΙΛΗΨΗ

Η εργασία αυτή επικεντρώνεται στην ανάλυση, μελέτη και αξιολόγηση μοντέλων ταξινόμησης Βαθιάς Μηχανικής Εκμάθησης (Deep Learning) και στην ενσωμάτωσή τους στο πρόγραμμα ελεύθερου λογισμικού Orfeo Toolbox, το οποίο αποτελεί μια ελεύθερη βιβλιοθήκη υλοποίησης τηλεπισκοπικών εφαρμογών σε C++. Οι μέθοδοι ταξινόμησης Deep Learning εφαρμόζονται με τη χρήση νευρωνικών δικτύων και θεωρούνται ως η πιο αποτελεσματική προσέγγιση για την ταξινόμηση δορυφορικών εικόνων σε επίπεδο ακριβειών.

Αρχικά αναλύονται εκτενώς τα δύο κύρια είδη νευρωνικών δικτύων: τα πλήρως συνδεδεμένα και τα συνελικτικά. Τα τελευταία χρησιμοποιούνται για την ταξινόμηση διαφόρων ομάδων δεδομένων με σκοπό τη διερεύνηση της αποτελεσματικότητάς τους. Όλες οι ταξινομήσεις γίνονται με τη βοήθεια της Lua, η οποία αποτελεί γλώσσα προγραμματισμού ελεύθερου λογισμικού. Επίσης, δημιουργούνται τρεις εφαρμογές που αφορούν την ταξινόμηση Deep Learning, συνδέοντας τη Lua στο Orfeo Toolbox. Η σύνδεση της γλώσσας C++, στην οποία στηρίζεται το Orfeo Toolbox, και της Lua γίνεται μέσω του Lua C API (Application Program Interface) και αναλύεται με λεπτομέρεια σε θεωρητικό επίπεδο.

Με την υλοποίηση των παραπάνω εφαρμογών, αναδεικνύονται οι δυνατότητες των μοντέλων ταξινόμησης Deep Learning. Επίσης, η βιβλιοθήκη του Orfeo Toolbox εμπλουτίζεται με νέους τρόπους ταξινόμησης δίνοντας στην κοινότητα και στους χρήστες τη δυνατότητα να στηριχτούν σε αυτούς και να τους εξελίξουν ακόμα περισσότερο.

Τέλος, οι ταξινομήσεις που υλοποιήθηκαν με βάση αρχιτεκτονικές βαθιάς μηχανικής μάθησης ξεπέρασαν ποσοτικά σε απόδοση αντίστοιχες προσεγγίσεις της διεθνούς βιβλιογραφίας. Συγκεκριμένα, η συνολική ακρίβεια της ομάδας δεδομένων DeepSat [Basu et al., 2015] έφτασε το 98.8%, ενώ στην ομάδα δεδομένων 'Zurich Summer Dataset v1.0' [Volpi et al., 2015] η συνολική ακρίβεια έφτασε το 95.1%.

ABSTRACT

This Diploma focuses on the analysis, investigation and evaluation of Deep Learning models for the classification of high resolution remote sensing data. Moreover, a main goal was to integrate certain Deep Learning tools into free software Orfeo Toolbox, which is a library for remote sensing applications in C++. Deep Learning classification methods are performed using neural networks and are considered as the most effective and accurate approach for the classification of satellite images.

First, two main types of neural networks are examined: Fully-Connected and Convolutional. The latter have been recently used for the classification of various datasets. All classification frameworks were developed in Lua, which is a free software programming language. Moreover, three new Deep Learning-related applications are provided by binding Lua to Orfeo Toolbox. The integration of C++, to which Orfeo Toolbox is based on, and Lua is applied through Lua C API (Application Program Interface). The theoretical background of this language binding is analyzed in detail.

The implementation of the above applications highlights the potentials of Deep Learning models. In addition, Orfeo Toolbox library is enriched with new classification methods, offering the community the ability to use them and further develop them.

Lastly, Deep Learning architectures that were used for classification outperformed similar approaches included in state of the art papers. More specifically, the overal accuracy of DeepSat Dataset [Basu et al., 2015] reached the level of 98.8%. In addition, the overall accuracy of 'Zurich Summer Dataset v1.0' [Volpi et al., 2015] reached the level of 95.1%.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ.....i
ΠΕΡΙΛΗΨΗ.....ii
ABSTRACT.....iii

1. Εισαγωγή.....1
1.1 Γενικότερες Έννοιες.....1
1.2 Αντικείμενο και Στόχος.....2
1.3 Συνεισφορά.....2
1.4 Δομή Εργασίας.....3

2. Ανασκόπηση Βιβλιογραφίας.....4
2.1 Θεωρητικές Έννοιες.....4
2.1.1 Προσδιορισμός του όρου Machine Learning.....4
2.1.2 Νευρωνικά Δίκτυα (Neural Networks).....5
2.1.2.1 Βιολογικά Νευρωνικά Δίκτυα.....5
2.1.2.2 Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks).....6
2.1.3 Βαθιά Εκμάθηση (Deep Learning).....8
2.1.3.1 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks).....9
2.1.3.2 Λοιπά είδη επιπέδων που χρησιμοποιούνται.....15
2.1.4 Παράδειγμα εφαρμογών.....16
2.2 Πρόσφατη Βιβλιογραφία/ State-of-the-art.....16
2.2.1 Land Use Classification in Remote Sensing Images by Convolutional Neural Networks [Marco Castelluccio et al.].....16
2.2.2 DeepSat – A Learning framework for Satellite Imagery [Saikat Basu et al.].....17
2.2.3 Benchmarking classification of earth-observation data: From learning explicit features to convolutional networks [Adrien Lagrange et al.].....17
2.2.4 Effective Semantic Pixel labelling with Convolutional Networks and Conditional Random Fields [Sakrapee Paisitkriangkrai et al.].....18
2.2.5 Semantic segmentation of urban scenes by learning local class interactions [Michele Volpi et al.].....20
2.2.6 OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks [Pierre Sermanet et al.].....21
2.2.7 Very Deep Convolutional Networks for Large-Scale Image Recognition [Karen Simonyan ∗ & Andrew Zisserman +].....22
2.2.8 ImageNet Classification with Deep Convolutional Neural Networks [Alex Krizhevsky et al.].....22
2.2.9 Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks [Dimitrios Marmanis et al.].....22
2.2.10 Rich feature hierarchies for accurate object detection and semantic segmentation [Ross Girshick et al.].....22

3. Μεθοδολογία.....24
3.1 Δεδομένα Εκπαίδευσης.....24
3.1.1 Δεδομένα DeepSat.....24
3.1.2 Δεδομένα 'Zurich Summer Dataset v1.0'.....25

3.1.3 Δεδομένα DEIMOS-2....28

3.2 Μοντέλα ταξινόμησης με βάση αρχιτεκτονικές Deep Learning....30
3.2.1 AlexNet....30
3.2.2 ConvNet....32
3.2.3 VGGNet....33
3.2.4 Ρηχές αρχιτεκτονικές και SVM....35
3.4 Αξιολόγηση αποτελεσμάτων....35
3.5 Υλοποίηση....36
3.5.1 Lua....36
3.5.2 Torch....37
3.5.2.1 Εγκατάσταση....37
3.5.3 Orfeo Toolbox (OTB)....38
3.5.3.1 Γενικά στοιχεία του Orfeo Toolbox....38
3.5.3.2 Εγκατάσταση....38
3.5.3.3 Λειτουργία....40
3.6 Σύνδεση εργαλείων Deep Learning στο Orfeo Toolbox....41
3.6.1 Σύνδεση της Lua με τη C++....41
3.6.1.1 Εικονική Στοίβα (Virtual Stack)....41
3.6.1.2 Lua C API – Κώδικας....42
3.6.2 Περιγραφή κώδικα....44
3.6.3 Εκτέλεση εφαρμογών ταξινόμησης με βάση αρχιτεκτονικές Deep Learning....48

4. Αποτελέσματα και Αξιολόγηση....54
4.1 Δεδομένα Deepsat....54
4.1.1 Ποσοτική αξιολόγηση για τα δεδομένα SAT-4....54
4.1.2 Ποσοτική αξιολόγηση για τα δεδομένα SAT-6....55
4.2 Ποσοτική αξιολόγηση για τα δεδομένα δεδομένα 'Zurich Summer Dataset v1.0'....55
4.2.1 Εύρεση βέλτιστων διαστάσεων των patches....57
4.2.1.1 Patches διαστάσεων 5x5....57
4.2.1.2 Patches διαστάσεων 11x11....59
4.2.1.3 Patches διαστάσεων 21x21....61
4.2.1.4 Patches διαστάσεων 29x29....64
4.2.1.5 Patches διαστάσεων 33x33....67
4.2.2 Βέλτιστη διάσταση των patches....70
4.2.2.1 AlexNet....73
4.2.2.2 VGG....75
4.2.2.3 Σύγκριση μοντέλων....76
4.3 Ποσοτική αξιολόγηση για τα δεδομένα δεδομένα DEIMOS-2 και IRIS....80
4.3.1 Δεδομένα DEIMOS-2....80
4.3.2 Δεδομένα IRIS....82

5. Συμπεράσματα και προοπτικές....86
5.1 Ειδικά Συμπεράσματα....86
5.1.1 Συμπεράσματα για την ομάδα δεδομένων Deepsat....86
5.1.2 Συμπεράσματα για την ομάδα δεδομένων 'Zurich Summer Dataset v1.0'....86
5.1.3 Συμπεράσματα για την ομάδα δεδομένων DEIMOS-2 και IRIS....87
5.2 Γενικά Συμπεράσματα....87

5.3 Προτάσεις.....88

Βιβλιογραφία.....89

ΠΑΡΑΡΤΗΜΑ Α.....91

ΠΑΡΑΡΤΗΜΑ Β

1. ΕΙΣΑΓΩΓΗ

Στο κεφάλαιο αυτό γίνεται αναφορά στις γενικότερες έννοιες που διαπραγματεύεται η παρούσα εργασία. Προσδιορίζεται η θεματολογία, ο στόχος της, η χρησιμότητα και τη συνεισφορά της. Τέλος, αναλύεται η οργάνωση και η δομή της.

1.1 Γενικότερες έννοιες

Η επιστήμη της τηλεπισκόπησης στηρίζεται κατά βαση στις δορυφορικές εικόνες, οι οποίες παρέχουν φασματικές πληροφορίες απαραίτητες για τη μελέτη των θαλάσσιων και χερσαίων φαινομένων που λαμβάνουν χώρα σε όλο το μήκος και το πλάτος της γης. Ένας από τους πιο σημαντικούς τομείς της τηλεπισκόπησης είναι η ταξινόμηση, η οποία νοείται ως ομαδοποίηση περιοχών με κοινά φασματικά χαρακτηριστικά. Τα αποτελέσματα που συνάγονται με τη μέθοδο της ταξινόμησης περιοχών είναι χρήσιμα για πολλούς ανθρώπινους τομείς, αφού είναι δυνατό να μελετηθεί όχι μόνο η συγκέντρωση ομοειδών αντικειμένων και εκτάσεων, αλλά και η εξέλιξή τους μέσα στο χρόνο. Γίνεται έτσι αντιληπτό, ότι η ωφελιμότητά της δεν περιορίζεται μόνο στη μελέτη της μορφής των εδαφικών χώρων, αλλά και στην ανάλυση οικονομικών και κοινωνικών φαινομένων.

Όμως, το πλήθος των τηλεπισκοπικών δεδομένων αυξάνεται όλο και περισσότερο με το πέρασμα των χρόνων, καθιστώντας επιτακτική την ανάγκη ανάπτυξης ευέλικτων προγραμμάτων που συντελούν στη λειτουργική και αυτοματοποιημένη διαχείρισή του. Η ανάγκη αυτή, σε συνδυασμό με τη ραγδαία ανάπτυξη της τεχνολογίας και του προγραμματισμού οδήγησαν στη δημιουργία προγραμμάτων ελεύθερου λογισμικού. Ο όρος “ελεύθερο λογισμικό”, χρησιμοποείται για να περιγράψει μια κατηγορία προγραμμάτων τα οποία μπορεί να χρησιμοποιήσει και να τρέξει οποιοσδήποτε χρήστης, αποκτώντας ταυτόχρονα το δικαίωμα να αντιγράψει, να αλλάξει, να βελτιώσει και να διανείμει τυχών λειτουργίες του προγράμματος ανάλογα με τα ενδιαφέροντά του. Πιο αναλυτικά, τα προγράμματα που ανήκουν στην κατηγορία ελεύθερου λογισμικού αποτελούνται από τέσσερις συγκεκριμένες ελευθερίες για το χρήστη. Αυτές είναι οι εξής:

- Η ελευθερία να χρησιμοποιεί το πρόγραμμα για οποιοδήποτε σκοπό (ελευθερία 0).
- Η ελευθερία να μελετά τον τρόπο λειτουργίας του προγράμματος και να τον αλλάζει, ώστε να προσαρμόζεται στις επιθυμίες του (ελευθερία 1). Η πρόσβαση στον πηγαίο κώδικα του προγράμματος είναι απαραίτητη προϋπόθεση για αυτό.
- Η ελευθερία να διανέμει εκ νέου αντίγραφα κώδικα για να βοηθήσει άλλους χρήστες (ελευθερία 2).
- Η ελευθερία να διανέμει αντίγραφα από τον τροποποιημένο κώδικα σε άλλους (ελευθερία 3). Με την πράξη αυτή, είναι δυνατό να επωφεληθεί μια ολόκληρη κοινότητα. Η πρόσβαση στον πηγαίο κώδικα του προγράμματος είναι απαραίτητη προϋπόθεση για αυτό.
  [https://www.gnu.org/philosophy/free-sw.en.html]

Από τα παραπάνω, συμπεραίνεται ότι το ελεύθερο λογισμικό μπορεί να προσφέρει σε όλους την ευκαιρία να γνωρίσουν και να κατανοήσουν οποιαδήποτε πτυχή της λειτουργίας ενός προγράμματος. Η δυνατότητα αυτή είναι επικερδής για το χρήστη, ο

οποίος επιμλουτίζει τις γνώσεις του και συμβάλλει στην ανάπτυξη και βελτίωση των προγραμμάτων. Κατ'επέκταση, ολόκληροι επιστημονικοί τομείς αναμορφώνουν και εξελίσσουν τα λογισμικά τους.

Ανεξάρτητα από την επιλογή προγράμματος για την ταξινόμηση τηλεπισκοπικών δεδομένων, οι μέθοδοι με τις οποίες αυτή υλοποιείται είναι συγκεκριμένες. Μία απο αυτές, η οποία χρησιμοποιείται και στην παρούσα εργασία, είναι η μέθοδος Deep Learning. Αποτελεί υποσύνολο των μεθόδων ταξινόμησης machine learning, οι οποίες στηρίζουν τη λειτουργία τους στη δημιουργία μοντέλων τα οποία σχηματίζονται από το χρήστη. Τα μοντέλα αυτά βελτιώνονται σταδιακά μέσα από επαναλαμβανόμενη εκπαίδευση, και είναι σε θέση να προβλέπουν με κάποιο ποσοστό επιτυχίας την κατηγορία στην οποία ανήκει οποιοδήποτε παράδειγμα τους δοθεί μετά το πέρας της διαδικασίας εκμάθησης. Ο τρόπος σχηματισμού των μοντέλων διαφέρει από μέθοδο σε μέθοδο. Η μέθοδος Deep Learning, χρησιμοποιεί νευρωνικά δίκτυα για το σχηματισμό αυτό.

Η παραπάνω μέθοδος ταξινόμησης, αλλά και οποιαδήποτε άλλη επιλεχθεί από ένα χρήστη, μπορεί να εφαρμοστεί σε προγράμματα ελεύθερου λογισμικού χρησιμοποιώντας γλώσσες προγραμματισμού. Η εφαρμογή της επιθυμητής μεθόδου μπορεί να πραγματοποιηθεί είτε δημιουργώντας κώδικα με την ίδια τη γλώσσα στην οποία είναι γραμμένο το εκάστοτε πρόγραμμα, είτε παράγοντας ανεξάρτητα κομμάτια κώδικα σε διαφορετική γλώσσα, τα οποία συνδέονται με το αρχικό προγράμμα μέσω language binding. Με αυτό τον τρόπο, εφαρμογές που απαιτούν χρονοβόρες διαδικασίες για τις προδιαγραφές ενός λογισμικού, μπορούν να εκτελεστούν χρησιμοποιώντας την πιο κατάλληλη και ευέλικτη γλώσσα για κάθε περίπτωση.

Μέσω του ελεύθερου λογισμικού λοιπόν, δίνεται η δυνατότητα συνεργασίας και ανταλλαγής ιδεών μεταξύ των χρηστών, με αποτέλεσμα την αλματώδη ανάπτυξη των προγραμμάτων και την εξέλιξη της επιστήμης.

1.2 Αντικείμενο και στόχος

Η συγκεκριμένη εργασία εστιάζει στη μελέτη της ταξινόμησης εικόνων με χρήση μεθόδων Deep Learning οι οποίες υλοποιούνται με τη βοήθεια της γλώσσας προγραμματισμού Lua. Δημιουργήθηκαν μοντέλα τα οποία εφαρμόστηκαν σε διάφορες ομάδες δεδομένων με σκοπό τη διερεύνηση της αποτελεσματικότητάς τους. Σχηματίστηκαν επίσης τρεις εφαρμογές σχετικές με την ταξινόμηση Deep Learning, οι οποίες στηρίζονται στη σύνδεση των γλωσσών C++ και Lua. Οι εν λόγω εφαρμογές υλοποιούν το στόχο της εργασίας που αφορά τη διεύρυνση των δυνατοτήτων του λογισμικού Orfeo Toolbox.

1.3 Συνεισφορά

Η εργασία αυτή αναδυκνείει τις δυνατότητες των μεθόδων ταξινόμησης Deep Learning και συμβάλλει στην επέκταση των εφαρμογών του λογισμικού Orfeo Toolbox. Μέχρι τώρα, οι εφαρμογές που σχετίζονται με την ταξινόμηση Deep Learning δεν είναι ιδιαίτερα ανεπτυγμένες για το εν λόγω πρόγραμμα. Οι ταξινομήσεις που υλοποιήθηκαν με βάση αρχιτεκτονικές βαθιάς μηχανικής μάθησης ξεπέρασαν ποσοτικά σε απόδοση αντίστοιχες προσεγγίσεις της πρόσφατης βιβλιογραφίας. Συγκεκριμένα, η συνολική ακρίβεια της ομάδας δεδομένων DeepSat

[Basu et al., DeepSat – A Learning framework for Satellite Imagery] έφτασε το 98.8%, ενώ στην ομάδα δεδομένων 'Zurich Summer Dataset v1.0' [Volpi et al., Semantic segmentation of urban scenes by learning local class interactions] η συνολική ακρίβεια έφτασε το 95.1%.

1.4 Δομή Εργασίας

Η δομή της εργασίας αυτής έχει ως εξής: Στο κεφάλαιο 2 αναλύονται οι θεωρητικές έννοιες που αφορούν τις μεθόδους Deep Learning και τα μοντέλα που χρησιμοποιούνται. Επίσης, γίνεται αναφορά σε διάφορες σχετικές δημοσιεύσεις από τη διεθνή βιβλιογραφία. Στο κεφάλαιο 3, δίνονται τα στοιχεία των ομάδων δεδομένων που επιλέχθηκαν για την εκπαίδευση των μοντέλων και γίνεται αναφορά στα λογισμικά που χρησιμοποιήθηκαν. Στην τελευταία ενότητα του συγκεκριμένου κεφαλαίου, παρουσιάζονται οι εφαρμογές που δημιουργήθηκαν στο ελεύθερο λογισμικό Orfeo Toolbox και δίνονται αναλυτικές οδηγίες για τη χρήση τους. Στη συνέχεια, στο κεφάλαιο 4 παραθέτονται τα αποτελέσματα από τις ταξινομήσεις που πραγματοποιήθηκαν. Τέλος, στο κεφάλαιο 5 αναφέρονται τα συμπεράσματα που προκύπτουν από την εργασία και οι προοπτικές που υπάρχουν στο συγκεκριμένο τομέα.

2. Ανασκόπηση Βιβλιογραφίας

2.1 Θεωρητικές Έννοιες

Στην ενότητα αυτή, αρχικά γίνεται προσπάθεια επεξήγησης της έννοιας 'Machine Learning', η οποία σχετίζεται με την ικανότητα των υπολογιστών να αναπτύσσουν δικιά τους αντίληψη. Έπειτα, αναλύονται οι μέθοδοι ταξινόμησης 'Deep Learning' και γίνεται αναφορά σε σχετικές πρόσφατες δημοσιεύσεις (state of the art).

2.1.1 Προσδιορισμός του όρου Machine Learning

Όπως είναι γνωστό, η ανρθώπινη νοημοσύνη είναι ιδιαίτερα ανεπτυγμένη και δε μπορεί σε καμία περίπτωση να συγκριθεί με αυτή του υπολογιστή. Για την ακρίβεια, όλες οι λειτουργίες του υπολογιστή βασίζονται σε αλγορίθμους που έχουν δημιουργηθεί από τον ίδιο τον άνθρωπο. Παρ'όλα αυτά, η τεχνολογική εξέλιξη οδήγησε στην επινόηση αλγορίθμων ικανών να δημιουργούν στοιχεία ευφυΐας στα υπολογιστικά συστήματα.

Τα παραπάνω σχόλια, αποτελούν μια πρώτη προσέγγιση του όρου 'Machine Learning'. Καθότι όμως είναι αρκετά δύσκολο να δωθεί ένας συγκεκριμένος και σαφής ορισμός για την έννοια αυτή, η σημασία της πρόκειται να αποσαφηνιστεί με τη βοήθεια μιας παρομοίωσης. Συγκεκριμένα, παρακάτω συγκρίνεται η λειτουργία του ανθρώπινου εγκεφάλου με τη λειτουργία του υπολογιστή.

Η εκμάθηση μιας οποιασδήποτε δραστηριότητας από τον ανθρώπινο εγκέφαλο, επιτυγχάνεται εφόσον αυτή πραγματοποιείται για κάποιο αριθμό επαναλήψεων. Για παράδειγμα, η ικανότητα των μικρών παιδιών να αναγνωρίζουν τα διαφορετικά είδη ζώων αποκτάται μέσα από επαναλαμβανόμενη οπτική επαφή με αυτά. Ο χρόνος εμπέδωσης που απαιτείται, εξαρτάται από τις ικανότητες του κάθε νεαρού ατόμου, το βαθμό δυσκολίας της δραστηριότητας αλλά και τον τρόπο μεταλαμπάδευσης της πληροφορίας. Δηλαδή, αν ένα παιδί μάθει εξ'αρχής να αναγνωρίζει το είδος 'σκύλος' ως 'γάτα' λόγω λανθασμένης διδασκαλίας, τότε παρά τις ικανότητες που μπορεί να διαθέτει, η αφομοίωση της πληροφορίας δε θα γίνει ποτέ σωστά. Επίσης, αν σε ένα παιδί ίδιας ηλικίας διδαχτούν τα μαθηματικά ολοκληρώματα, η έλλειψη ανεπτυγμένων ικανοτήτων και προαπαιτούμενων γνώσεων θα οδηγήσει στην παρανόηση των μαθηματικών αυτών εννοιών και στην αδυναμία κατανόησής τους. Τέλος, σημειώνεται ότι ακόμα και στην περίπτωση σωστής εκπαίδευσης, πάντα υπάρχει μια πιθανότητα λάθους από το ίδιο το άτομο.

Όσο αφορά τους υπολογιστές, η εκπαίδευσή τους πραγματοποιείται με τη δημιουργία μοντέλων εκπαίδευσης. Τα είδη των μοντέλων που μπορούν να κατασκευαστούν είναι άπειρα και κάθε ένα από αυτά έχει διαφορετικά χαρακτηριστικά και ικανότητες. Ανάλογα με τη θεματολογία της εκπαίδευσης, ένας αριθμός αντιπροσωπευτικών δεδομένων εισάγεται στα μοντέλα. Τα δεδομένα αυτά αποτελούν τη βάση στην οποία θα στηριχθεί η εκμάθηση των μοντέλων μέσα από επαναληπτικές διαδικασίες. Μετά το πέρας μιας επιτυχημένης εκπαίδευσης, το υπολογιστικό σύστημα έχει αναπτύξει ένα βαθμό τεχνητής νοημοσύνης και είναι σε θέση να αναγνωρίζει οποιοδήποτε στοιχείο, διαφορετικό από τα δεδομένα εκπαίδευσης, του δωθεί. Ένα πολύ γνωστό παράδειγμα μηχανικής εκμάθησης (Machine Learning) συντελείται στον ιστότοπο youtube.com, όπου μία από τις ικανότητες των χρηστών είναι η εύρεση και

παρακολούθηση οποιουδήποτε βίντεο επιθυμούν. Μέσα από την επαναλαμβανόμενη αναζήτηση βίντεο συγκεκριμένου είδους, η ιστοσελίδα 'μαθαίνει' τις προτιμήσεις του χρήστη και εμφανίζει στην αρχική σελίδα δεδομένα με ίδια ή παρόμοια θεματολογία.

Συμπερασματικά λοιπόν, η λογική που ακολουθείται για την εκπαίδευση των υπολογιστών, είναι παρόμοια με αυτή των ανθρώπων και αποκαλείται 'machine learning'. Πιο συγκεκριμένα, το μοντέλο εκπαίδευσης που δημιουργείται αντιστοιχίζεται με το ρόλο του εγκεφάλου, αφού απο τις ικανότητές του εξαρτάται η ευδοκιμότητα της διαδικασίας. Ακόμα, τα δεδομένα εκπαίδευσης αντιπροσωπεύουν τον τρόπο μεταλαμπάδευσης της πληροφορίας. Τούτο διότι αποτελούν τη μόνη πηγή γνώσης για το μοντέλο, επομένως πρέπει να είναι επαρκή και αντιπροσωπευτικά. Είναι επίσης αναγκαίο να περιλαμβάνουν πληροφορία ισοδύναμη με το επίπεδο πολυπλοκότητας του μοντέλου και σχετική με τη θεματολογία της εκπαίδευσης.

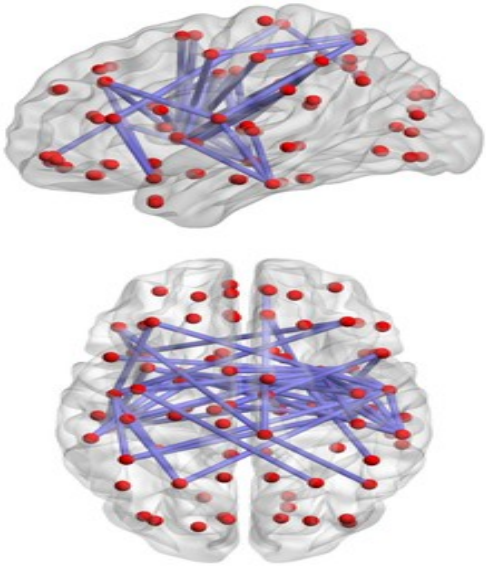
Τα μοντέλα που αντιπροσωπεύουν τη λογική της μηχανικής εκμάθησης(Machine Learning), χρησιμοποιούνται από διάφορους επιστημονικούς κλάδους. Όσο αφορά την τηλεπισκόπηση, τα μοντέλα αυτά είναι ιδιαίτερα αποδοτικά στον τομέα της ταξινόμησης και συνθέτουν την ομάδα μεθόδων 'Deep Learning'. Η μέθοδος αυτή στηρίζεται κυρίως στα νευρωνικά δίκτυα (Neural Networks). Στις παρακάτω ενότητες, οι έννοιες αυτές εξηγούνται αναλυτικά.

2.1.2 Νευρωνικά Δίκτυα (Neural Networks)

Πριν αναλυθεί η λειτουργία των τεχνητών νευρωνικών δικτύων, κρίνεται σκόπιμο να γίνει αναφορά στο σύστημα των βιολογικών νευρωνικών δικτύων του ανρθώπινου εγκεφάλου.

2.1.2.1 Βιολογικά Νευρωνικά Δίκτυα

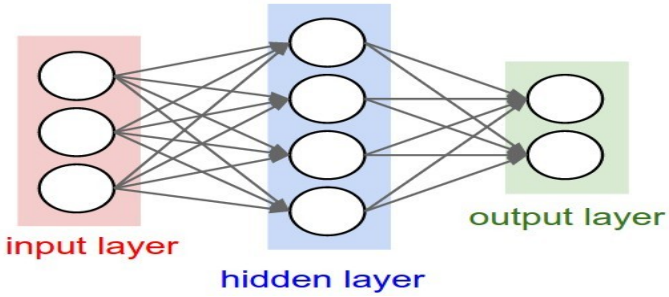
Τα τεχνητά νευρωνικά δίκτυα, είναι ουσιαστικά μια προσομοίωση της λειτουργίας των βιολογικών νευρωνικών δικτύων του εγκεφάλου. Τα βιολογικά νευρωνικά δίκτυα αποτελούνται από νευρώνες, οι οποίοι δέχονται πληροφορίες και ερεθίσματα από το εξωτερικό περιβάλλον και συνδέονται μεταξύ τους μέσω συνάψεων. Αφού γίνει η εισαγωγή της πληροφορίας στο σύστημα, κάθε νευρώνας επεξεργάζεται τα στοιχεία που του έχουν δωθεί τόσο με αυτόνομες διεργασίες, όσο και σε συνεργασία με τους υπόλοιπους νευρώνες. Με τον τρόπο αυτό, ο εγκέφαλος αποκτά τη δυνατότητα να αναγνωρίζει πρότυπα όταν δέχεται συγκεκριμένα δεδομένα. Για την καλύτερη κατανόηση των βιολογικών νευρωνικών δικτύων του εγκεφάλου, στην Εικόνα 2.1 δίνεται μια προσομοίωσή τους.



Εικόνα 2.1: Προσομοίωση των βιολογικών νευρωνικών δικτύων του εγκεφάλου. Οι κόκκινοι κύκλοι αντιπροσωπεύουν τους νευρώνες και οι μπλε γραμμές τις συνάψεις μέσα από τις οποίες συνδέονται.
(Πηγή: https://en.wikipedia.org/wiki/Biological_neural_network)

2.1.2.2 Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks)

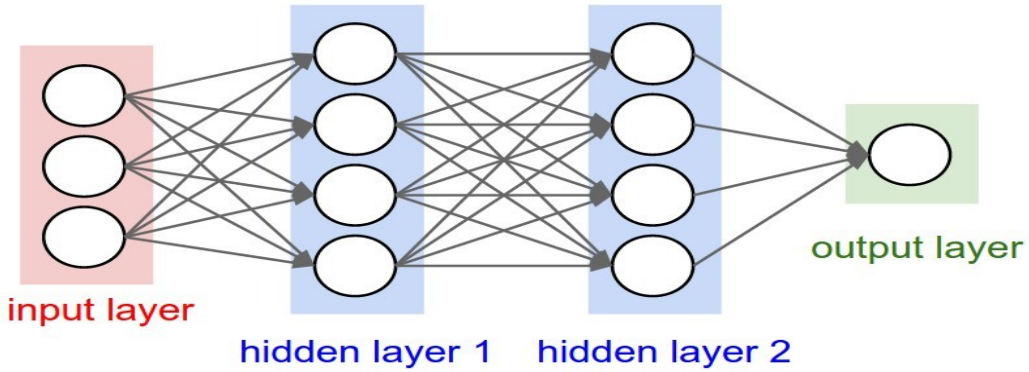
Όπως αναφέρθηκε στην εισαγωγή του κεφαλαίου, η λειτουργία των τεχνητών νευρωνικών δικτύων είναι όμοια με αυτή των βιολογικών. Πιο συγκεκριμένα, ένα τεχνητό νευρωνικό δίκτυο αποτελείται από ένα σύνολο τεχνητών νευρώνων (neurons), οι οποίοι είναι οργανωμένοι σε διαδοχικά επίπεδα (layers) και συνδέονται μεταξύ τους μέσω συνάψεων. Στην Εικόνα 2.2 απεικονίζεται η δομή ενός τεχνητού νευρωνικού δικτύου.



Εικόνα 2.2: Δομή τεχνητού νευρωνικού δικτύου
(Πηγή: http://cs231n.github.io/neural-networks-1/)

Όπως φαίνεται, το δίκτυο αποτελείται από ένα επίπεδο εισόδου (input layer), ένα κρυφό επίπεδο (hidden layer) και ένα επίπεδο εξόδου (output layer). Κάθε επίπεδο αποτελείται από νευρώνες οι οποίοι απεικονίζονται με κύκλους, ενώ τα βέλη που συνδέουν τους νευρώνες αποτελούν τις συνάψεις. Κάθε νευρώνας που ανήκει σε κάποιο επίπεδο i, δέχεται πληροφορίες από όλους τους νευρώνες του επιπέδου i-1 και μεταδίδει πληροφορίες σε όλους τους νευρώνες του επιπέδου i+1. Όταν ισχύει αυτή η συνθήκη, τα επίπεδα ονομάζονται πλήρως συνδεδεμένα (fully-connected). Αντίθετα,

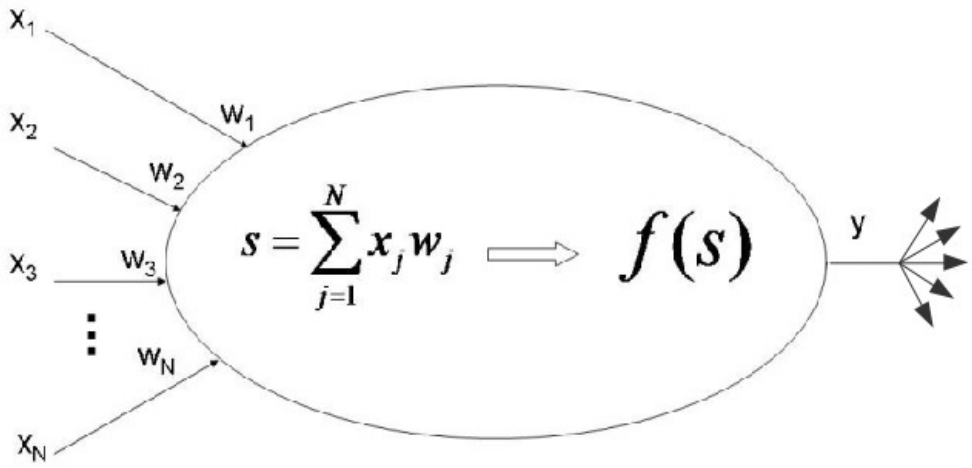
οι νευρώνες που ανήκουν στο ίδιο επίπεδο δεν έχουν καμία σύνδεση. Σημειώνεται επίσης ότι ο αριθμός των κρυφών επιπέδων ορίζεται από το χρήστη χωρίς κάποιο περιορισμό. Μπορεί δηλαδή ένα δίκτυο να έχει ένα ή και περισσότερα κρυφά επίπεδα (Εικόνα 2.3).



Εικόνα 2.3: Τεχνητό νευρωνικό δίκτυο με περισσότερα από ένα κρυφά επίπεδα (Πηγή: http://cs231n.github.io/neural-networks-1/)

Κατά την εκπαίδευση ενός τεχνητού νευρωνικού δικτύου, το επίπεδο εισόδου τροφοδοτείται με κάποιο διάνυσμα εκπαίδευσης το οποίο είναι σε μορφή πίνακα. Στη συνέχεια, το διάνυσμα αυτό υποβάλλεται σε μαθηματική επεξεργασία από τα κρυφά επίπεδα για κάποιο αριθμό επαναλήψεων. Όταν η επεξεργασία αυτή τερματιστεί, οι τελικές μετρήσεις δίνονται ως αποτέλεσμα από το επίπεδο εξόδου. Σημειώνεται ότι οι υπολογιστικές διαδικασίες υλοποιούνται μόνο από τους νευρώνες των κρυφών επιπέδων. Οι νευρώνες εισόδου και εξόδου συμβάλλουν στη λήψη του σήματος και στην έξοδο της τελικής πληροφορίας αντίστοιχα.

Στα κρυφά επίπεδα, ο κάθε νευρώνας δέχεται σαν είσοδο κάποια πληροφορία, δηλαδή κάποια σήματα $x_0,x_1,x_2,\ldots,x_n$, τα οποία αποτελούν έξοδο άλλων νευρώνων. Το σήμα αυτό πολλαπλασιάζεται με τα βάρη που αντιστοιχούν στις συνάψεις μέσα από τις οποίες επικοινωνούν οι νευρώνες του εκάστοτε επιπέδου και σχηματίζεται το άθροισμα $x_0w_0+x_1w_1+x_2w_2+\ldots+x_nw_n+b$, όπου b τα ενδεχόμενα συστηματικά λάθη. Στη συνέχεια, το άθροισμα αυτό διοχετεύεται στη συνάρτηση μεταφοράς(transfer function). Η συνάρτηση αυτή αποτελεί ένα είδος φίλτρου, το οποίο επεξεργάζεται τα δεδομένα που δέχεται και μπορεί να έχει διάφορες μαθηματικές μορφές. Με τη βοήθειά της διαμορφώνεται η τελική τιμή του σήματος. Έπειτα, η τιμή δίνεται είτε ως είσοδος στο επόμενο κρυφό επίπεδο νευρώνων είτε ως τελικό αποτέλεσμα στο επίπεδο εξόδου, ανάλογα με τη θέση του κρυφού επιπέδου που βρίσκεται ο αντίστοιχος νευρώνας. Στην Εικόνα 2.4, φαίνεται η μορφή ενός τεχνητού νευρώνα.



Εικόνα 2.4: Εσωτερική λειτουργία τεχνητού νευρώνα
(Πηγή: http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural2.html)

Μετά την ανάλυση της δομής ενός απλού νευρωνικού δικτύου, είναι δυνατή η προσέγγιση του ορισμού 'Deep Learning', η οποία δίνεται παρακάτω.

2.1.3 Βαθιά Εκμάθηση (Deep Learning)

Οι μέθοδοι εκμάθησης 'Deep Learning', ανήκουν στην ευρύτερη κατηγορία των μεθόδων μηχανικής εκμάθησης (Machine Learning). Και σε αυτή την περίπτωση, η αποσαφήνιση του όρου δε μπορεί να γίνει με ένα συγκεκριμένα ορισμό. Ξανά, η προσέγγισή της έννοιας γίνεται με τη βοήθεια παρομοίωσης.

Έστω ότι κάποιος επιστήμονας αναλαμβάνει να φέρει εις πέρας μια εργασία η οποία χαρακτηρίζεται από υψηλό επίπεδο δυσκολίας και περιλαμβάνει επεξεργασία πολλών δεδομένων. Αν ο επιστήμονας είναι ο μόνος υπεύθυνος για την περάτωσή της, τότε η εργασία θα περιλαμβάνει αποκλειστικά δικές του ιδέες και μεθόδους. Αντίθετα, αν η εργασία ανατεθεί σε μια ομάδα επιστημόνων, θα υπάρχει δυνατότητα συνεχούς ανταλλαγής γνώσεων και απόψεων, γεγονός που πιθανόν να οδηγήσει σε καλύτερα αποτελέσματα. Είναι απαραίτητο όμως να σημειωθεί, ότι ο αριθμός των ατόμων που απαρτίζουν την επιστημονική ομάδα πρέπει να βρίσκεται μέσα σε κάποια λογικά όρια. Αν η ομάδα αποτελείται από υπερβολικά μεγάλο αριθμό μελών, ενδέχεται να δημιουργηθούν προβλήματα συνεννόησης που θα οδηγήσουν σε εσφαλμένα αποτελέσματα.

Οι μέθοδοι που αντιπροσωπεύουν την κατηγορία 'deep learning', ακολουθούν την ίδια λογική. Πιο συγκεκριμένα, τα μοντέλα που χρησιμοποιούνται περιλαμβάνουν περισσότερα από ένα επίπεδα επεξεργασίας. Τα επίπεδα αυτά επεξεργάζονται τα δεδομένα για κάποιο αριθμό επαναλήψεων και συνδέονται μεταξύ τους με σχέσεις αλληλεπίδρασης, ανταλλάσσοντας χρήσιμες πληροφορίες. Στην περίπτωση των νευρωνικών δικτύων, η λογική των μεθόδων 'Deep Learning' υλοποιείται με την κατασκευή αλλεπάλληλων κρυφών επιπέδων (hidden layers), κάθε ένα από τα οποία μπορεί να έχει διαφορετικά χαρακτηριστικά και ιδιότητες. Ο αριθμός των επιπέδων δε θα πρέπει να είναι παράλογα μεγάλος, διαφορετικά οι πιθανότητες υπερπροσαρμογής

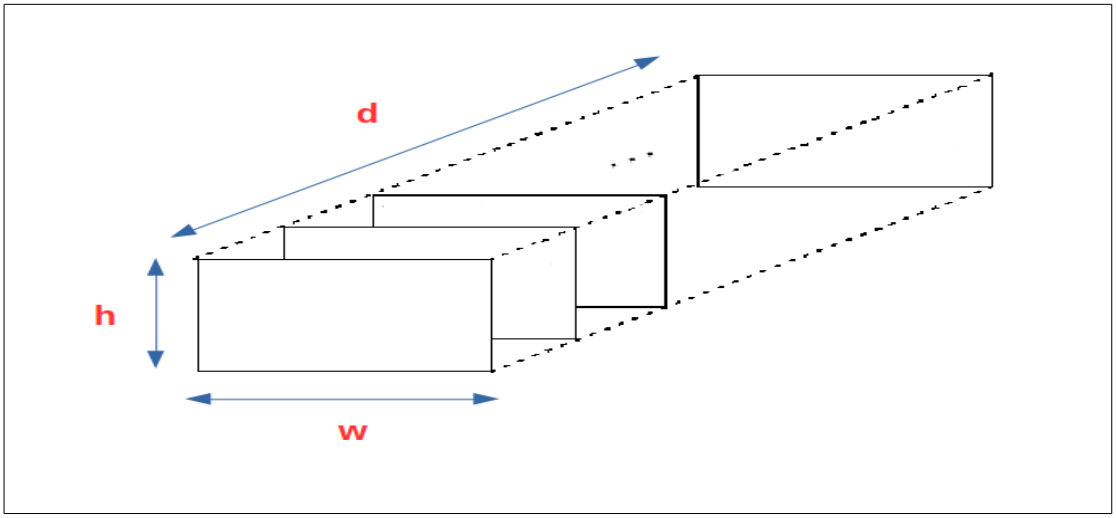
(overfitting) του μοντέλου είναι μεγαλύτερες. Ο όρος υπερπροσαρμογή (overfitting), χαρακτηρίζει την κατάσταση όπου το μοντέλο λαμβάνει υπόψην το θόρυβο των δεδομένων για την εκαπίσευσή του και όχι τις ίδιες τις πληροφορίες των δεδομένων. Η περίπτωση αυτή προκύπτει συνήθως όταν το μοντέλο είναι κατασκευασμένο με ιδαίτερα πολύπλοκο τρόπο σε σχέση με τα δεδομένα που δέχεται. Αν ισχύουν όλα τα παραπάνω, τότε τα μοντέλα αυτά μετονομάζονται από 'Neural Networks', σε 'Deep Neural Networks'.

Τα είδη των νευρωνικών δικτύων που απαρτίζουν τη μέθοδο 'deep learning', είναι αρκετά. Το είδος που χρησιμοποιείται για την κατασκευή των μοντέλων στη συγκεκριμένη εργασία φέρει το όνομα 'Convolutional Neural Networks'. Στην παρακάτω ενότητα, γίνεται η λεπτομερής ανάλυσή του.

2.1.3.1 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks)

Όπως αναφέρθηκε παραπάνω, σε ένα απλό νευρωνικό δίκτυο (Neural Network) τα διανύσματα εκπαίδευσης υποβάλλονται σε επεξεργασία από τους νευρώνες των διαδοχικών, πλήρως συνδεδεμένων (fully-connected) επιπέδων (layers) του. Αυτό σημαίνει ότι αν σε ένα επίπεδο εισόδου (input layer) δωθεί μια εικόνα 224x224x3 (όπου ο αριθμός 3 αντιπροσωπεύει τα διαθέσιμα κανάλια), τότε κάθε νευρώνας του επιπέδου εισόδου θα αποκτά αριθμό βαρών ίσο με τα pixels της εικόνας πολλαπλασιασμένα με τον αριθμό διαθέσιμων καναλιών, δηλαδή 224*224*3=150528 βάρη. Αν αυτό ισχύει μόνο για έναν από τους νευρώνες του επιπέδου, τότε το άθροισμα των βαρών για το σύνολο του επιπέδου θα είναι πολύ μεγαλύτερο. Συνεπώς, η εκπαίδευση με τη συγκεκριμένη τακτική γίνεται όχι μόνο πολύπλοκη αλλά και ακατάλληλη, αφού αυξάνονται οι πιθανότητες του δικτύου να οδηγηθεί σε υπερπροσαρμογή (overfitting).

Το πρόβλημα αυτό επιλύεται με τη χρήση συνελικτικών νευρωνικών δικτύων (Convolutional Neural Networks). Σε γενικά πλαίσια, η λογική που ακολουθούν είναι ίδια με αυτή των απλών νευρωνικών δικτύων(neural networks). Δέχονται δηλαδή ως είσοδο κάποιο διάνυσμα εκπαίδευσης, το οποίο στη συνέχεια υφίσταται επεξεργασία από τους νευρώνες των διαδοχικών επιπέδων που διαθέτουν. Η διαφορά που παρουσιάζουν έγκειται στο γεγονός ότι τα επίπεδα δεν είναι πλήρως συνδεδεμένα(fully-connected) και η πληροφορία που επεξεργάζονται οι νευρώνες είναι σε τρισδιάστατη μορφή (3-D). Δηλαδή, κάθε νευρώνας που ανήκει σε ένα επίπεδο i, αντλεί τρισδιάστατη πληροφορία από ένα μέρος μόνο του επιπέδου i-1 και μεταδίδει τρισδιάστατη πληροφορία σε ένα μέρος μόνο του επιπέδου i+1. Όσο αφορά την τρίτη διάσταση, ονομάζεται βάθος (depth) και αντιπροσωπεύει τον αριθμό των διαθέσιμων καναλιών της εικόνας. Επομένως η εικόνα διαστάσεων 224x224x3 έχει πλάτος (width) 224, ύψος (height) 224 και βάθος (depth) 3. Οι χαρακτήρες που χρησιμοποιούνται για τις μεταβλητές αυτές είναι w,h,d και προκύπτουν από τα αρχικά των αντίστοιχων ονομασιών τους(Εικόνα 2.5). Για την καλύτερη κατανόηση της διαφοράς μεταξύ απλών και συνελικτικών νευρωνικών δικτύων, παρακάτω δίνεται ένα απλό παράδειγμα.

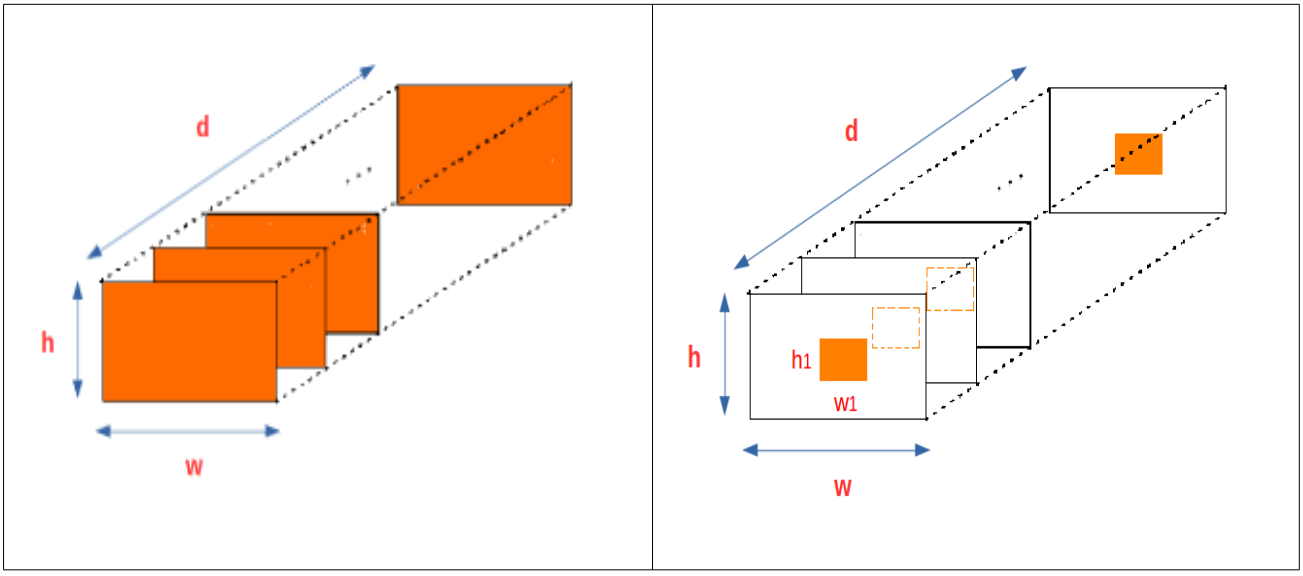


Εικόνα 2.5: Γραφική απεικόνιση των μεταβλητών width,height και depth μιας εικόνας διαστάσεων wxhxd.

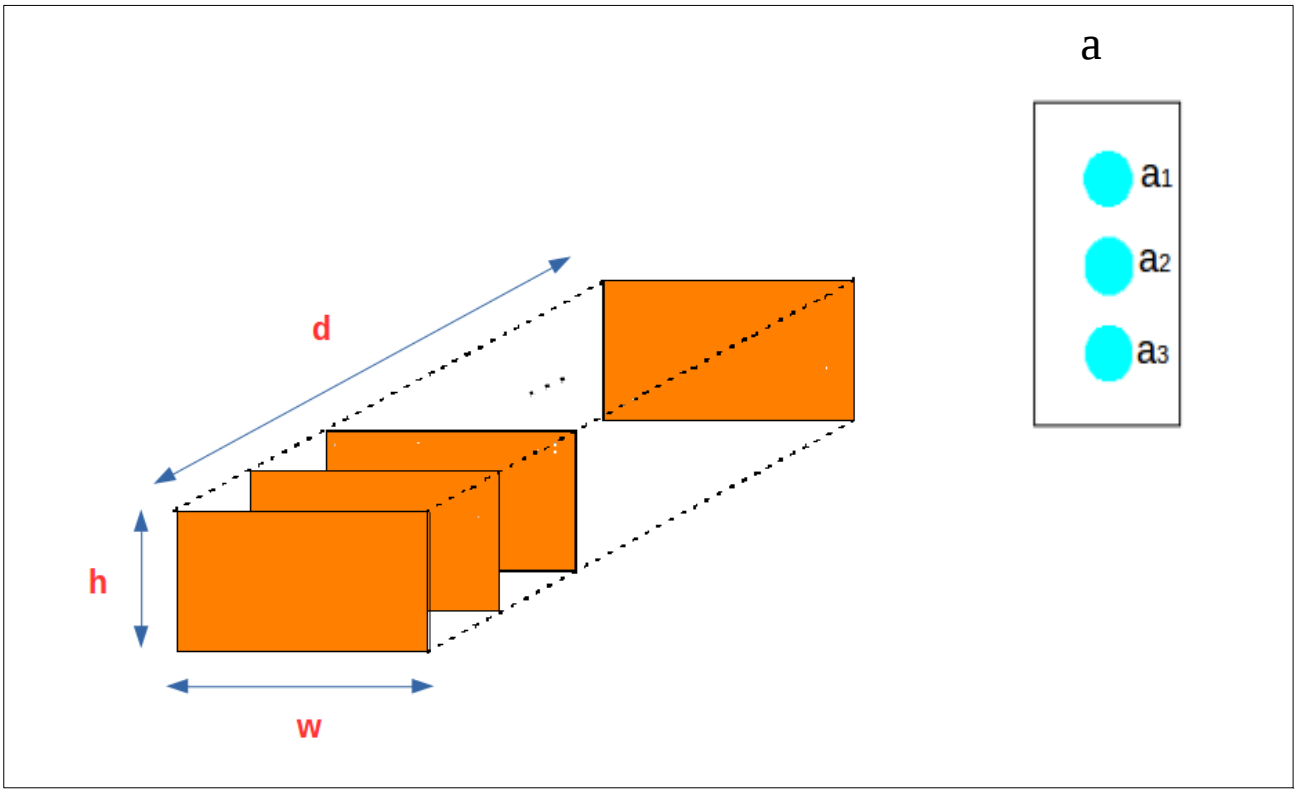
Έστω ότι υπάρχει μια εικόνα διαστάσεων wxhxd. Αν αυτή η εικόνα δοθεί ως διάνυσμα εκπαίδευσης στο επίπεδο εισόδου ενός απλού νευρωνικού δικτύου, τότε κάθε νευρώνας του επιπέδου θα συνδεθεί με wxhxd στοιχεία, όσες και οι τιμές των pixels της εικόνας. Δηλαδή θα έχει αποκτήσει wxhxd βάρη(Εικόνα 2.6, αριστερά). Στην περίπτωση του συνελικτικού νευρωνικού δικτύου, κάθε νευρώνας θα συνδεθεί με ένα τμήμα μόνο της εικόνας, το οποίο ορίζεται από το χρήστη. Ξανά, τονίζεται ότι το τμήμα αυτό έχει τρισδιάστατη μορφή. Στο συγκεκριμένο παράδειγμα, το τμήμα έχει διαστάσεις w_1x$h_1$xd (Εικόνα 2.6, δεξιά), συνεπώς κάθε νευρώνας θα αποκτήσει w_1x$h_1$xd βάρη.

Μέχρι τώρα, εξηγήθηκε ο τρόπος με τον οποίο οι νευρώνες των συνελικτικών δικτύων δέχονται τις πληροφορίες. Θα πρέπει όμως να αναλυθεί και η διάταξη που έχουν μέσα στα επίπεδα του δικτύου. Παρακάτω, περιγράφεται με λεπτομέρεια η μορφή τους.

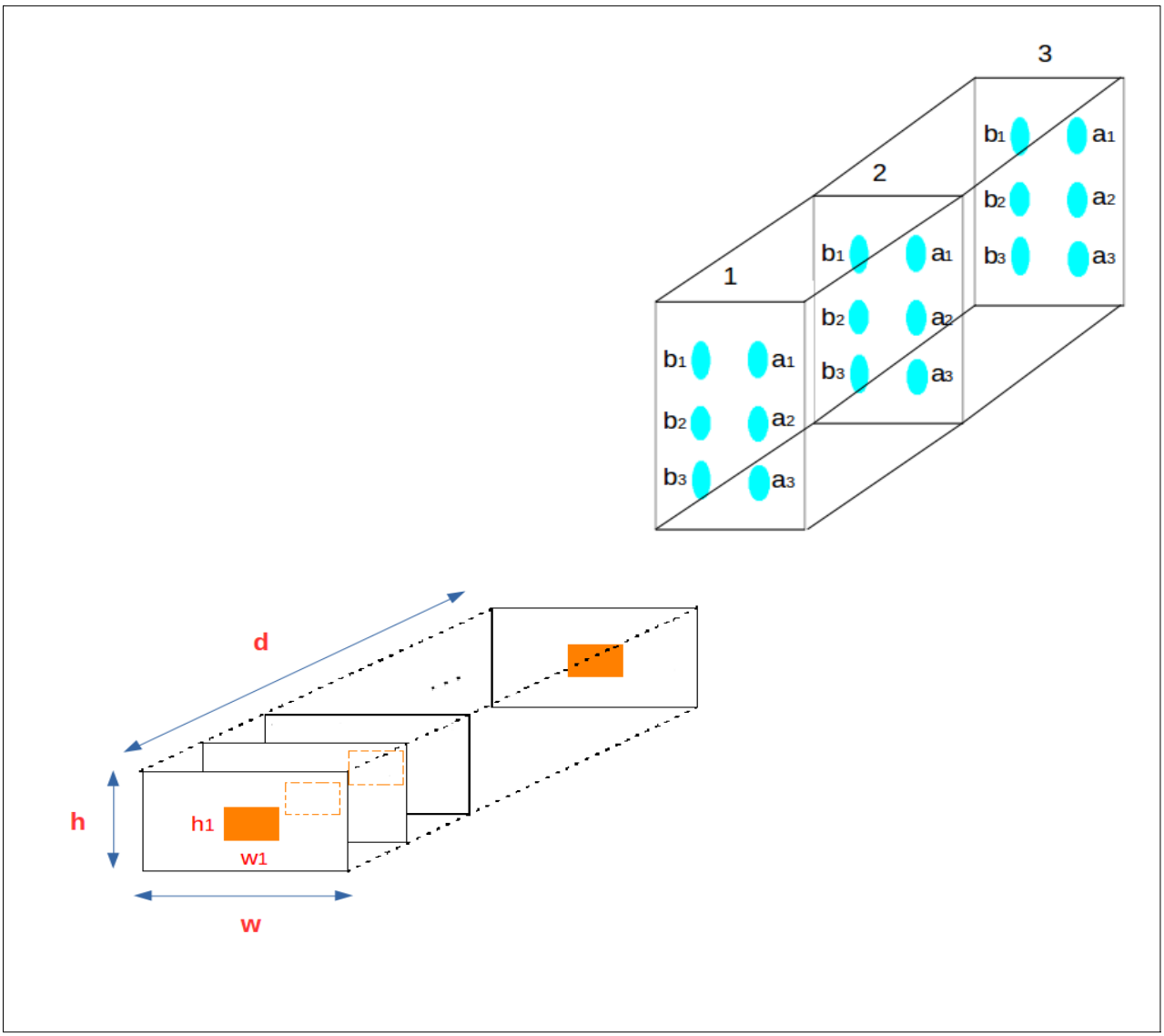
Όπως αναφέρθηκε παραπάνω, αν μια εικόνα διαστάσεων wxhxd δωθεί ως διάνυσμα εκπαίδευσης στο επίπεδο εισόδου ενός απλού νευρωνικού δικτύου, τότε κάθε νευρώνας του επιπέδου θα συνδεθεί με wxhxd στοιχεία. Στην Εικόνα 2.7, απεικονίζεται η εικόνα διαστάσεων wxhxd και το επίπεδο εισόδου ενός απλού νευρωνικού δικτύου. Κάθε ένας από τους τρεις νευρώνες του επιπέδου εισόδου a, αποκτά wxhxd βάρη. Η αντίστοιχη απεικόνιση για ένα συνελικτικό νευρωνικό δίκτυο, φαίνεται στην Εικόνα 2.8.



Εικόνα 2.6: Γραφική απεικόνιση της διαφοράς μεταξύ απλών και συνελικτικών νευρωνικών δικτύων. Οι περιοχές με πορτοκαλί χρωματισμό αντιπροσωπεύουν τα στοιχεία που δέχεται κάθε νευρώνας του επιπέδου εισόδου από μια εικόνα διαστάσεων wxhxd. Αριστερά βρίσκεται η απεικόνιση για τα απλά νευρωνικά δίκτυα και δεξιά για τα συνελικτικά νευρωνικά δίκτυα.



Εικόνα 2.7: Οι πληροφορίες που δέχεται το επίπεδο εισόδου ενός απλού νευρωνικού δικτύου από μια εικόνα διαστάσεων wxhxd. Οι περιοχές με πορτοκαλί χρωματισμό αντιπροσωπεύουν τις πληροφορίες με τις οποίες συνδέεται κάθε νευρώνας. Το επίπεδο εισόδου a, αποτελείται από 3 νευρώνες (a_1, a_2 και a_3).



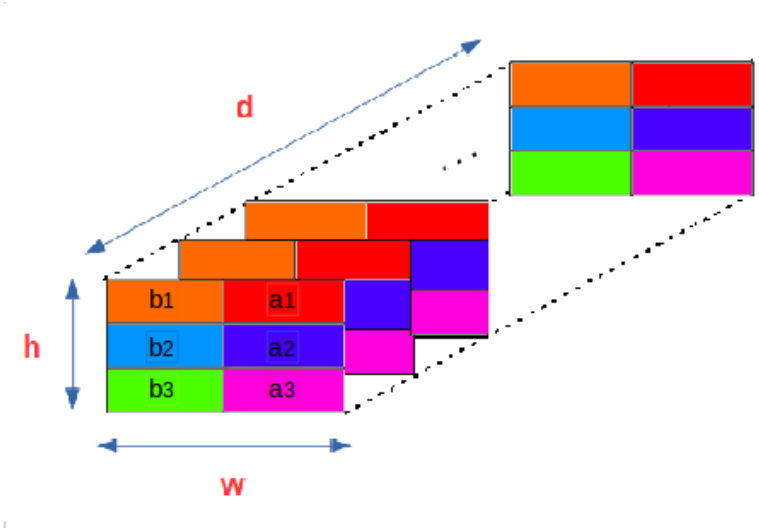
Εικόνα 2.8: Οι πληροφορίες που δέχεται το επίπεδο εισόδου ενός συνελικτικού νευρωνικού δικτύου από μια εικόνα διαστάσεων wxhxd. Οι περιοχές με πορτοκαλί χρωματισμό αντιπροσωπεύουν τις πληροφορίες με τις οποίες συνδέεται κάθε νευρώνας.

Όπως φαίνεται στην Εικόνα 2.8, η τρισδιάστατη μορφή χαρακτηρίζει όχι μόνο την εικόνα διαστάσεων wxhxd, αλλά και το επίπεδο εισόδου. Η λειτουργία των νευρώνων έχει ως εξής: Έστω ότι ο νευρώνας a1 του επιπέδου 1 συνδέεται με την περιοχή w1 x h1 x d και συνεπώς αποκτά w1 x h1 x d βάρη. Τότε και οι υπόλοιποι νευρώνες των επιπέδων 2 και 3 με την ονομασία a1, θα συνδέονται με την ίδια περιοχή διαστάσεων w1 x h1 x d, αλλά με διαφορετικά βάρη. Η ομάδα a1 των νευρώνων ονομάζεται στήλη βάθους (depth-column). Κάθε στήλη βάθους συνδέεται με διαφορετική περιοχή της εικόνας, ίδιων όμως διαστάσεων. Για παράδειγμα, οι στήλες βάθους του επιπέδου εισόδου της Εικόνας 2.8, θα μπορούσαν να αντιστοιχίζονται στις υποπεριοχές της εικόνας που φαίνεται στην Εικόνα 2.9. Η στήλη βάθους a1 της Εικόνας 2.8, αντιστοιχίζεται στην υποπεριοχή a1 της Εικόνας 2.9. Η στήλη βάθους b1 της Εικόνας 2.8, αντιστοιχίζεται στην υποπεριοχή b1 της Εικόνας 2.9 κ.ο.κ. Όσο

αφορά τα επίπεδα 1,2 και 3 (Εικόνα 2.8), ονομάζονται φέτες βάθους (depth-slices). Ξανά, τονίζεται ότι όλες οι υποπεριοχές έχουν ίδιες διαστάσεις. Κάθε υποπεριοχή μπορεί να θεωρηθεί ως ένα φίλτρο, το οποίο μετατοπίζεται συνέχεια σε διαφορετικές θέσεις της εικόνας. Στην ιδιότητα αυτή οφείλουν το όνομά τους τα συνελικτικά νευρωνικά δίκτυα, αφού ουσιαστικά είναι σαν να εκτελείται μία συνέλιξη.

Προκειμένου να τεθεί σε εφαρμογή ένα συνελικτικό νευρωνικό δίκτυο, απαιτείται η ρύθμιση ορισμένων μεταβλητών από τη χρήστη. Οι μεταβλητές αυτές είναι οι εξής :

- οι διαστάσεις της υποπεριοχής, ή αλλιώς το μέγεθος του φίλτρου (F) (filter size)
- η επικάλυψη του περιγράμματος της εικόνας με μηδενικά (P) (zero-padding)
- το βήμα με το οποίο μετατοπίζεται η υποπεριοχή (S) (stride)
- το βάθος κάθε επιπέδου του δικτύου (K)

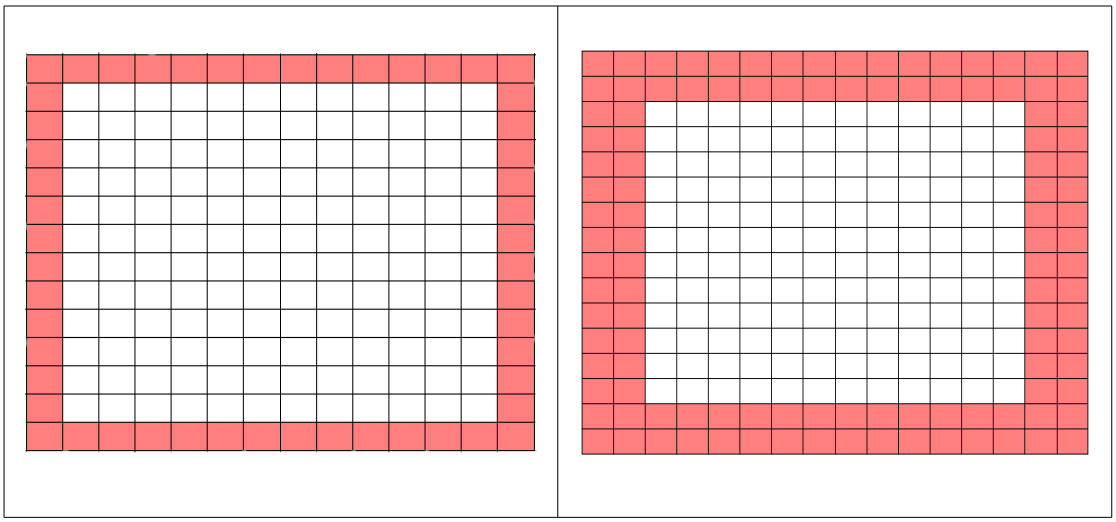


Εικόνα 2.9: Υποπεριοχές της εικόνας διαστάσεων wxhxd στις οποίες αντιστοιχίζονται οι στήλες βάθους(depth-columns) της Εικόνας 2.8.

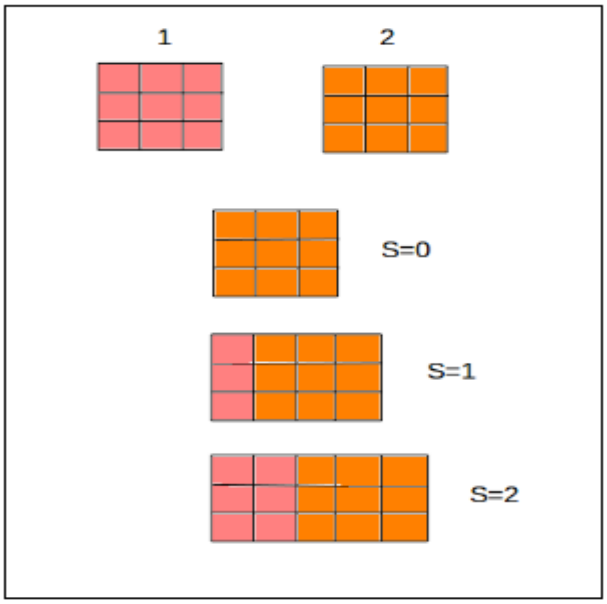
Πιο αναλυτικά, η μεταβλητή F έχει μορφή dxnxm, όπου n, m και d οι διαστάσεις που αντιστοιχούν στον άξονα του πλάτους (w), του ύψους (h) και του βάθους (d) αντίστοιχα. Σημειώνεται ότι όλες οι υποπεριοχές που πρόκειται να σχηματιστούν κατά την εκπαίδευση του δικτύου έχουν ίδιες διαστάσεις. Ακόμα, οι διαστάσεις των υποπεριοχών είναι πάντα τοπικές σε σχέση με τους άξονες w και h, αλλά περιλαμβάνουν πάντα ολόκληρη τη διάσταση του βάθους. Η μεταβλητή P, επιτρέπει στο χρήστη να τοποθετεί μηδενικά στο περίγραμμα των εικόνων. Στην Εικόνα 2.10 φαίνεται ένα σχετικό παράδειγμα. Όσο αφορά τη μεταβλητή S, καθορίζει το βήμα μετατόπισης της υποπεριοχής. Για παράδειγμα, στην Εικόνα 2.9, αν S=1 τότε η υποπεριοχή a1 θα απέχει από την υποπεριοχή b1 μία χωρική μονάδα. Ανάλογα με το βήμα που χρησιμοποιείται, η απόσταση των υποπεριοχών αλλάζει. Έτσι, είναι πιθανό να υπάρχει κάποια επικάλυψη μεταξύ τους. Για την καλύτερη κατανόηση της συγκεκριμένης μεταβλητής, στην Εικόνα 2.11 δίνεται η αντίστοιχη γραφική απεικόνιση, ενώ στην Εικόνα 2.12 σχηματίζεται ένα παράδειγμα. Όσο αφορά τώρα τη μεταβλητή K, καθορίζει τις φέτες βάθους (depth-slices) που έχει κάθε επίπεδο του

δικτύου. Οι φέτες βάθους αντιπροσωπεύουν ένα συγκεκριμένο επίπεδο και δε θα πρέπει να συγχέονται με το σύνολο των επιπέδων του δικτύου.

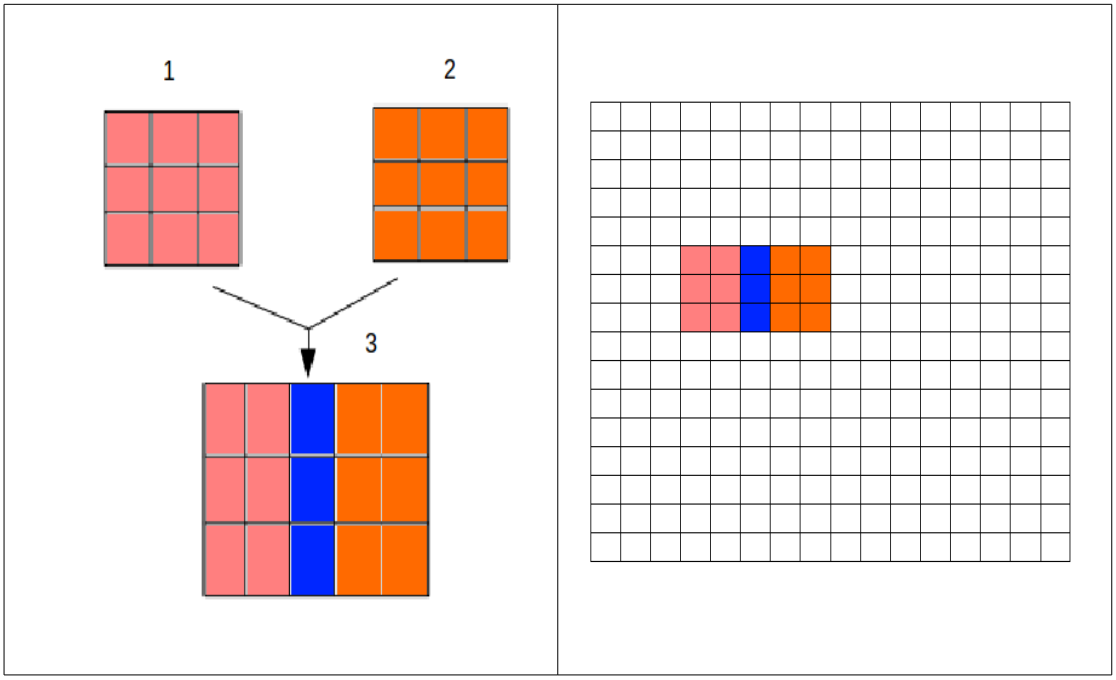
Τέλος, αναφέρεται ότι αν μια εικόνα διαστάσεων wxhxd υποβληθεί σε επεξεργασία από ένα συνελικτικό επίπεδο με μεταβλητές F, S και P, τότε οι διαστάσεις της εικόνας μετά την έξοδό της από το επίπεδο θα είναι [(w-F+2*P)/S+1] x [(h-F+2*P)/S+1].



Εικόνα 2.10: Έστω ότι έχουμε μια εικόνα διαστάσεων 12x12 (κάθε μικρό τετράγωνο αντιστοιχεί σε ένα pixel). Στην περιοχή με ροζ χρωματισμό έχει εφαρμοστεί γέμισμα με μηδενικά(zero-padding). Αριστερά, είναι P=1, ενώ δεξιά είναι P=2.



Εικόνα 2.11: Έστω ότι για κάποια εικόνα, δίνονται οι διαδοχικές υποπεριοχές 1 και 2. Αν υποθέσουμε ότι F=dx3x3, τότε παρπάνω φαίνεται ο τρόπος με τον οποίο υλοποιείται η μεταβλητή S για τις τιμές 0, 1 και 2. Για S=0 υπάρχει πλήρης επικάλυψη των υποπεριοχών.



Εικόνα 2.12: Έστω ότι η εικόνα διαστάσεων 16x16x3(δεξιά) δίνεται σε ένα επίπεδο εισόδου με F=3x3x3 και S=2. Αν το 1 pixel θεωρείται μία χωρική μονάδα, τότε οι υποπεριοχές 1 και 2 που θα σχηματιστούν θα έχουν την επικάλυψη που φαίνεται με μπλε χρωματισμό.

2.1.3.2 Λοιπά είδη επιπέδων που χρησιμοποιούνται

Η ανάλυση των παραπάνω μεταβλητών, αφορούσε τα συνελικτικά επίπεδα(convolutional layers) ενός δικτύου. Υπάρχουν όμως και άλλα είδη επιπέδων που μπορούν να χρησιμοποιηθούν. Αυτά είναι τα εξής:

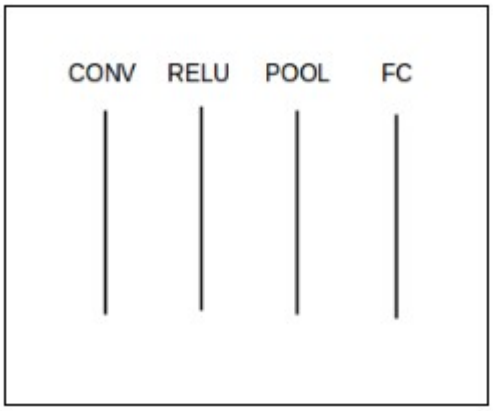
- Επίπεδα 'Pooling'
  Τα επίπεδα αυτά τοποθετούνται συνήθως ανάμεσα στα συνελικτικά επίπεδα προκειμένου να ελαττώσουν τα δεδομένα που υποβάλλονται σε επεξεργασία από τους νευρώνες. Η ιδιότητά τους αυτή είναι πολύ σημαντική αφού μειώνει την πιθανότητα υπερπροσαρμογής του δικτύου. Οι μεταβλητές που ορίζονται για το επίπεδο αυτό από το χρήστη είναι η F και η S.

  Αν μια εικόνα διαστάσεων wxhxd υποβληθεί σε επεξεργασία από ένα επίπεδο pooling με μεταβλητές F και S, τότε οι διαστάσεις της εικόνας μετά την έξοδό της από το επίπεδο θα είναι [(w-F)/S+1] x [(h-F)/S+1].

- Πλήρως συνδεδεμένα επίπεδα(Fully-Connected layers ή FC)
  Τα επίπεδα αυτά ακλουθούν τη λογική των απλών νευρωνικών δικτύων, δηλαδή ένα επίπεδο i συνδέεται πλήρως με το επίπεδο i-1. Επομένως, δεν υπάρχουν μεταβλητές που ορίζονται από το χρήστη.

- Επίπεδα Μεταφοράς Συνάρτησης (Transfer Function)
  Τα επίπεδα αυτά εφαρμόζουν στα δεδομένα κάποια συνάρτηση, όπως η συνάρτηση εφαπτομένης (Tanh).

Μια πιθανή μορφή ενός συνελικτικού νευρωνικού δικτύου είναι αυτή που παρουσιάζεται στην Εικόνα 2.13.



Εικόνα 2.13: Πιθανή μορφή των επιπέδων ενός συνελικτικού νευρωνικού δικτύου.Το επίπεδο ReLU ανήκει στην κατηγορία Transfer Function και εφαρμόζει τη συνάρτηση Rectified Linear Unit.

2.1.4 Παράδειγμα εφαρμογών

Έστω ότι τα επίπεδο της Εικόνας 2.13 έχουν τα εξής χαρακτηριστικά:

- Convolutional Layer: F=3, S=1, P=0, K=16
- Pooling Layer: F=2, S=2

Αν μια εικόνα διαστάσεων 28x28x4 δωθεί ως είσοδος στο δίκτυο, τότε θα συμβούν τα εξής:

- Η εικόνα εισάγεται πρώτα στο συνελικτικό επίπεδο. Κατά την έξοδό της από αυτό έχει διαστάσεις (28-3+2*0)/1+1 x (28-3+2*0)/1+1= 26x26x16.
- Η εικόνα περνάει από το επίπεδο RELU χωρίς καμία αλλαγή στις διαστάσεις της.
- Η εικόνα περνάει από το επίπεδο Pooling, και μετά την έξοδό της έχει αποκτήσει διαστάσεις [(26-2)/2+1] x [(26-2)/2+1]= 13x13x16
- Η εικόνα περνάει από το τελευταίο πλήρως συνδεδεμένο επίπεδο του δικτύου, όπου δεν πραγματοποιείται καμία αλλαγή στις διαστάσεις της και τα αποτελέσματα δίνονται ως έξοδος από το δίκτυο.

2.2 Πρόσφατη Βιβλιογραφία/ State-of-the-art

Στην ενότηταζ αυτή παρουσιάζονται κάποιες δημοσιεύσεις που προέρχονται από τη διεθνή βιβλιογραφία και αφορούν το αντικείμενο της εργασίας αυτής.

2.2.1 Land Use Classification in Remote Sensing Images by Convolutional Neural Networks [Marco Castelluccio et al.]

Στο συγκεκριμένο paper, συγκρίθηκε η αποτελεσματικότητα της εκπαίδευσης δεδομένων με συνελικτικά μοντέλα από το μηδέν, με εκείνη των προεκπαιδευμένων μοντέλων που έχουν προσαρμοστεί στα εκάστοτε δεδομένα. Τα μοντέλα CaffeNet και

GoogleNet χρησιμοποιήθηκαν για τα δεδομένα UC Merced και Brazilian Coffe Scenes. Στην περίπτωση των δεδομένων UC Merced τα προεκπαιδευμένα μοντέλα κατέληξαν σε καλύτερες ακρίβειες με μεγαλύτερη ταχύτητα, ενώ στην περίπτωση των δεδομένων Brazilian Coffe Scenes τα καλύτερα αποτελέσματα προέκυψαν με την εκπαίδευση από το μηδέν. Το τελευταίο πιθανόν οφείλεται στις διαφορετικές τεχνικές διαχείρισης των καλλιεργειών, στισ διαφορετικές ηλικίες των φυτών κ.τ.λ. Σε γενικές γραμμές όμως, τα προεκπαιδευμένα μοντέλα είναι πολλά υποσχόμενα και επιδέχονται περαιτέρω έρευνα.

2.2.2 DeepSat – A Learning framework for Satellite Imagery [Saikat Basu et al.]

Στο paper αυτό παρουσιάστηκαν δύο καινούριες ομάδες δεδομένων που απεικονίζουν μέρη των Ηνωμένων Πολιτειών. Πρόκειται για patches διαστάσεων 28x28 με ονόματα SAT-4 και SAT-6. Προτάθηκε επίσης ένας νέος τρόπος ταξινόμησης σύμφωνα με τον οποίο τα εξαγόμενα χαρακτηριστικά των εικόνων κανονικοποιούνται και δίνονται ως είσοδος σε ένα μοντέλο DBN (Deep Belief Network). Εφαρμόστηκαν επίσης και οι παραδοσιακοί τρόποι ταξινόμησης με απλά Deep Belief Networks, Convolutional Networks και SDAE (Stack Denoising Auto Encoders και συγκρίθηκαν τα αποτελέσματα. Ο προτεινόμενος τρόπος ταξινόμησης ξεπέρασε σε ακρίβεια όλους τους παραδοσιακούς, λόγω της μεγαλύτερης πολυπλοκότητας των χαρακτηριστικών των δορυφορικών εικόνων, της ελλιπούς επισήμανσής τους και των μεγαλύτερων εγγενών διαστάσεών τους. Οι μεγαλύτερες εγγενείς διαστάσεις δικαιολογήθηκαν με χρήση του αλγορίθμου DANCo [C. Ceruti et al.].

2.2.3 Benchmarking classification of earth-observation data: From learning explicit features to convolutional networks [Adrien Lagrange et al.]

Στο συγκεκριμένο paper, αντιμετωπίστηκε η ταξινόμηση των δεδομένων παρατήρησης γης με διάφορες μεθόδους που εφαρμόστηκαν τα τελευταία 15 χρόνια. Χρησιμοποιήθηκαν τα δεδομένα IEEE GRSS Data Fusion Contest του 2015, για τα οποία δημιουργείται το αντίστοιχο groundtruth με 8 κλάσεις. Σημειώνεται επίσης ότι η διαδικασία εκπαίδευσης στηρίχθηκε σε 3 εικόνες, ενώ ο έλεγχος σε 2. Οι ταξινομήσεις που εξετάστηκαν στηρίζονται τόσο στα δεδομένα σε ακατέργαστη μορφή, όσο και σε ιδιαίτερα χαρακτηριστικά που έχουν εξαχθεί.

Αρχικά, εξήχθησαν ακατέργαστα δεδομένα από τις εικόνες, το διαθέσιμο ψηφιακό μοντέλο εδάφους (DSM) και το διαθέσιμο ψευδο-υπέρυθρο (DMSI) από τα δεδομένα Lidar που παρέχονται. Έπειτα, ένα μοντέλο SVM εκπαιδεύτηκε ξεχωριστά για κάθε μία απο τις τρεις ομάδες δεδομένων.

Στη συνέχεια, χρησιμοποιήθηκε ένα γραμμικό μοντέλο SVM στο οποίο εισήχθησαν patches διαστάσεων 16x16 και 32x32. Κάθε ένα από τα patches περιέχει επιπρόσθετη πληροφορία που αφορά τα ιστογράμματα HOG (Histograms of Oriented Gradient). Η εκπαίδευση πραγματοποήθηκε ξεχωριστά για κάθε διάσταση.

Η ταξινόμηση πραγματοποιήθηκε επίσης και με τη χρήση ενός γραμμικού SVM ο οποίος δέχτηκε ως είσοδο superpixels που είχαν εξαχθεί από τις εικόνες και περιγράφονται από το χρωματικό χώρο HSV. Η εκπαίδευση πραγματοποιήθηκε επίσης και σε συνδυασμό με τη μέση τιμή και τη μέση κλίση του DSM.

Η επόμενη διαδικασία εκπαίδευσης αφορά την αντικειμενοστραφή ανίχνευση. Πρόκειται για μοντέλα μιας κατηγορίας, τα οποία αποτελούνται από ένα συνδυασμό ξεχωριστών μοντέλων εκπαιδευμένων με οπτικά ομοιογενή δεδομένα. Τα δείγματα για κάθε αντικείμενο ταξινομήθηκαν με βάση την οπτική εμφάνιση και για κάθε ταξινομημένο αντικείμενο, ένα γραμμικό μοντέλο SVM εκπαιδεύτηκε με βάση τα ιστογράμματα HOG αυτών των δειγμάτων (Discriminatively-trained Model Mixtures, DtMM). Χρησιμοποιήθηκε επίσης και ένας δεύτερος αντικειμενοστραφής ανιχνευτής ο οποίος στηρίχθηκε σε χάρτες SOM (Self-Organizing Maps).

Τέλος, εξετάστηκε η απόδοση των συνελικτικών μοντέλων CNN σε συνδυασμό με το μοντέλο SVM. Συγκεκριμένα, εξήχθησαν τα αποτελέσματα από ενδιάμεσα επίπεδα τριών μοντέλων που χρησιμοποιήθηκαν στο ImageNet [J. Deng et al.], με σκοπό τη δημιουργία δεδομένα εκπαίδευσης. Τα μοντέλα του ImageNet που επιλέχθηκαν είναι το OverFeat, το Caffe και το VGG. Με τον τρόπο αυτό δημιουργήθηκαν patches διαστάσεων 231x231 τα οποία εισήχθησαν σε ένα γραμμικό μοντέλο SVM. Στην περίπτωση του μοντέλου VGG, χρησιμοποιήθηκαν και επιπρόσθετες πληροφορίες που αφορούν τόσο το διαθέσιμο DSM, όσο και ένα ακριβέστερο DSM που προέκυψε με την προβολή του ύψους από το LiDAR. Στον Πίνακα 2.1 δίνονται τα αποτελέσματα που προέκυψαν για όλες τις παραπάνω μεθόδους.

Table 2. Method comparison: F1 measures per class (best: , second: , third:), overall accuracy and Cohen's Kappa.

Algorithm	Imp. surf	Build.	Low veg.	Tree	Car	Clutter	Boat	Water	Overall acc. %	Cohen κ
Expert	58.97	63.87	74.55					92.39	∅	∅
RGB	53.89	53.53	50.32	32.97	24.02	13.75	12.12	98.52	60.77	0.52
RGBD	14.51	67.79	38.03	27.43	7.15	1.12	14.58	98.45	50.76	0.41
RGBID	60.86	69.01	57.12	38.12	11.59	20.49	15.04	94.42	63.83	0.56
HOG32/SVM	28.94	43.17	48.77	27.32	30.24	17.39	12.61	88.02	52.45	0.41
HOG16/SVM	39.52	38.45	35.65	29.99	21.93	16.13	13.52	80.02	49.4	0.36
HSV/SVM	71.60	46.97	68.38	0.12	0.00	13.71	0.00	92.14	70.16	0.60
HSV+Dgrad/SVM	73.30	70.85	68.75	0.17	0.00	17.11	0.00	92.37	73.60	0.65
SOM							51.45		∅	∅
DtMM					48.46				∅	∅
RGB OverFeat/SVM	55.86	63.34	59.48	64.44	36.03	28.31	41.51	92.07	67.97	0.59
RGB Caffe/SVM	62.32	62.66	63.23	60.84	31.34	32.49	46.57	95.61	71.06	0.63
RGB VGG/SVM	63.18	64.66	63.60	66.98	31.46	43.68	51.92	95.93	72.36	0.64
RGBD VGG/SVM	66.02	74.26	65.04	66.94	32.04	44.96	50.61	96.31	74.77	0.67
RGBD$^+$ VGG/SVM	67.66	72.70	68.38	78.77	33.92	45.6	56.10	96.50	76.56	0.70

Πίνακας 2.1: Αποτελέσματα όλων των μεθόδων. (Πηγή: Adrien Lagrange et al.)

2.2.4 Effective Semantic Pixel labelling with Convolutional Networks and Conditional Random Fields [Sakrapee Paisitkriangkrai et al.]

Στο συγκεκριμένο paper τα δεδομένα του διαγωνισμού ISPRS labelling 2016, χρησιμοποιήθηκαν για σημασιολογική σήμανση με τη βοήθεια ενός συνδυασμού των μοντέλων CNN, RF (Random Forest) και CRF (Conditional Random Fields). Πριν την εκτέλεση της προτεινόμενης μεθόδου, εξετάστηκαν και άλλοι τρόποι αντιμετώπισης του προβλήματος οι οποίοι αναλύονται παρακάτω. Σημειώνεται ακόμα

ότι οι περιοχές των εικόνων χωρίστηκαν σε περιοχές εκπαίδευσης (training) και περιοχές επικύρωσης (validation).

Αρχικά χρησιμοποιήθηκε ένα απλό συνελικτικό μοντέλο CNN στο οποίο εισήχθησαν patches διαστάσεων 5x5 που προήλθαν από τις εικόνες, το ψηφιακό μοντέλο εδάφους (DSM) και το κανονικοποιημένο ψηφιακό μοντέλο εδάφους (NDSM). Η εκπαίδευση πραγματοποιήθηκε τόσο για το σύνολο των δεδομένων, όσο και για τα δεδομένα που προήλθαν μόνο από τις εικόνες με σκοπό την ανάδειξη της σημασίας του DSM. Στον Πίνακα 2.2 φαίνονται τα εξαγόμενα αποτελέσματα.

	Imp. surf.	Building	Low veg.	Tree	Car	Average F1	Overall Acc.
Ortho	82.91%	88.13%	67.14%	81.77%	53.50%	74.69%	80.10%
Ortho + DSM	82.93%	88.75%	68.40%	82.44%	51.15%	74.74%	80.71%
Ortho + NDSM	86.07%	92.79%	72.85%	82.85%	54.63%	77.84%	83.46%
Ortho + DSM + NDSM	87.35%	93.34%	74.96%	84.97%	63.32%	80.79%	**85.18%**

Πίνακας 2.2: Αποτελέσματα ταξινόμησης του μοντέλου CNN με και χωρίς το DSM. (Πηγή: Sakrapee Paisitkriangkrai et al.)

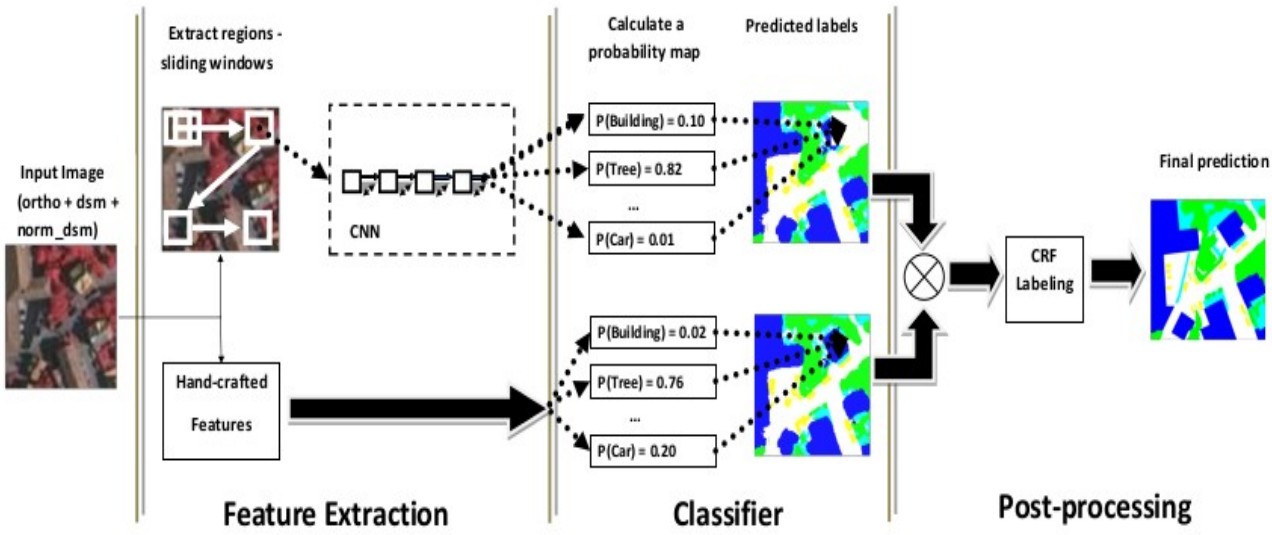
Η επόμενη εφαρμογή περιελάμβανε και πάλι ένα συνελικτικό μοντέλο CNN στο ο ποίο εισήχθησαν patches διαστάσεων 16x16, 32x32 και 64x64. Το μοντέλο εκπαιδεύτηκε επίσης και ξεχωριστά για κάθε διάσταση. Στον Πίνακα 2.3 δίνονται οι ακρίβειες που προέκυψαν. Φαίνεται ότι η χρήση πολλών διαστάσεων βελτιστοποιεί τη διαδικασία εκμάθησης.

Input image resolution (pixels)	Imp. surf.	Building	Low veg.	Tree	Car	Average F1	Overall Acc.
16 × 16	84.70%	92.15%	72.54%	83.51%	42.54%	75.09%	82.78%
32 × 32	85.96%	92.42%	74.09%	84.68%	61.06%	79.64%	84.20%
64 × 64	87.35%	93.34%	74.96%	84.97%	63.32%	80.79%	85.18%
ALL	87.72%	93.27%	75.53%	85.29%	66.89%	81.74%	**85.56%**

Πίνακας 2.3: Αποτελέσματα ταξινόμησης του μοντέλου CNN με patches διαφόρων διαστάσεων. Το ALL αντιπροσωπεύει το συνδυασμό των τριών διαστάσεων (Πηγή: Sakrapee Paisitkriangkrai et al.)

Εκτός από τα patches των εικόνων, εξήχθησαν και χειροποίητα χαρακτηριστικά σε επίπεδο pixel τα οποία δόθηκαν ως είσοδος για την εκπαίδευση ενός μοντέλου RF (Random Forest). Εξετάστηκε επίσης ο συνδυασμός των μοντέλων CNN και RF.

Τέλος, πραγματοποιήθηκε η προτεινόμενη μέθοδος η οποία είναι ο συνδυασμός των μοντέλων CNN και RF με έναν ταξινομητή CRF ο οποίος τοποθετείται τελευταίος και βοηθά στην εξομάλυνση των τελικών αποτελεσμάτων (Εικόνα 2.14). Τα αποτελέσματα όλων των συνδυασμών φαίνονται στον Πίνακα 2.4. Η προτεινόμενη μέθοδος ξεπερνά σε ακρίβεια όλες τις υπόλοιπες που εξετάστηκαν.



Εικόνα 2.14: Προτεινόμενη μέθοδος ταξινόμησης (Πηγή: Sakrapee Paisitkriangkrai et al.)

	Imp. surf.	Building	Low veg.	Tree	Car	Overall F1	Overall Acc.
Multi-res CNN	87.72%	93.27%	75.53%	85.29%	66.89%	81.74%	85.56%
Random forest	85.83%	92.79%	70.88%	83.98%	0.0%	66.69%	83.47%
CNN+RF	88.58%	94.23%	76.58%	86.29%	67.58%	82.65%	86.52%
CNN+RF+CRF	89.10%	94.30%	77.36%	86.25%	71.91%	83.78%	**86.89%**

Πίνακας 2.4: Αποτελέσματα ταξινόμησης από συνδυασμό μεθόδων. Τα αποτελέσματα από το πολυδιάστατο μοντέλο CNN τοποθετούνται για ευκολία. (Πηγή: Sakrapee Paisitkriangkrai et al.)

2.2.5 Semantic segmentation of urban scenes by learning local class interactions [Michele Volpi et al.]

Στο συγκεκριμένο paper, το πρόβλημα διαχωρισμού των κλάσεων σε εικόνες πολύ υψηλής ανάλυσης αντιμετωπίστηκε με τη χρήση ενός μοντέλου CRF (Conditional Random Fields), οι παράμετροι του οποίου είχαν υποστεί προεπεξεργασία από ένα μοντέλο SSVM (Structured Support Vector Machine). Για την εξαγωγή των δεδομένων εκπαίδευσης, σχηματίστηκαν superpixels στις εικόνες.

Πιο συγκεκριμένα, για την εύρεση των μοναδιαίων και των κατά ζεύγη πιθανοτήτων του μοντέλου CRF, προτάθηκε η μέθοδος SRP (Single Ring Potentials) όπου οι πιθανότητες σχηματίζονται με τη βοήθεια ενός κύκλου κεντραρισμένου σε κάθε superpixel. Το μειονέκτημα αυτής της μεθόδου είναι ότι η βέλτιστη ακτίνα του κύκλου δε μπορεί να είναι γνωστή εκ των προτέρων. Για το λόγο αυτό προτάθηκε και η μέθοδος MRP (Multiple Ring Potentials) κατά την οποία επιτρέπεται η ταυτόχρονη αλληλεπίδραση διαφόρων ακτινών. Σημειώνεται εδώ ότι για την εύρεση των κατά ζεύγη πιθανοτήτων χρησιμοποιήθηκε η εξομάλυνση Potts.

Αφού ολοκληρώθηκε η παραπάνω διαδικασία, τα βάρη των πιθανοτήτων δόθηκαν στο μοντέλο SSVM. Με τον τρόπο αυτό οι παράμετροι προσαρμόστηκαν καλύτερα στα δεδομένα εκπαίδευσης.

Τα μοντέλα που σχηματίστηκαν είναι τα εξής:

1. Ένας μη γραμμικός ταξινομητής RF (Random Forest) ο οποίος χρησιμοποιεί τις μοναδιαίες πιθανότητες που έχουν εξαχθεί.
2. Ένας ταξινομητής SVM πολλαπλών κλάσεων που χρησιμοποιεί μόνο τις μοναδιαίες πιθανότητες, δηλαδή μαθαίνει μόνο το γραμμικό συνδυασμό των στοιχείων που εισάγονται (MCSVM, Multi-Class SVM).
3. Ένας ταξινομητής CRF contrast-sensitive (CS Potts).
4. Ένας ταξινομητής PASSIVE RP (Passive Ring Potentials. Πρόκειται ουσιαστικά για τον ταξινομητή που περιγράφηκε στο νούμερο 3 με τη διαφορά ότι εδώ έχουν προστεθεί δεδομένα που αφορούν τη συσχέτιση μεταξύ των κλάσεων.
5. Ένας ταξινομητής με χρήση του μοντέλου SRP για ακτίνα ίση με 20 μέτρα (LEARNED RP 20M).
6. Ένας ταξινομητής με χρήση του μοντέλου SRP για ακτίνα ίση με 40 μέτρα (LEARNED RP 40M).
7. Ένας ταξινομητής με χρήση του μοντέλου SRP για ακτίνα ίση με 60 μέτρα (LEARNED RP 60M).
8. Ένας ταξινομητής με χρήση του μοντέλου MRP για τιμές ακτίνων {20,40,60} (LEARNED MRP).

Για την αξιολόγηση των αποτελεσμάτων χρησιμοποιήθηκαν τα εξής κριτήρια:

- ο μέσος όρος των μέσων ακριβειών (AA)
- ο δείκτης Kappa (κ)
- F1 score (F1)
- η μέση ακρίβεια για κάθε κλάση

Τα αποτελέσματα που προέκυψαν για κάθε μοντέλο φαίνονται στον Πίνακα 1.

Model	AA	κ	F1	Roads	Build.	Trees	Grass	Soil	Water	Rails	Pools
RF UNARY	72.19	80.15	75.10	81.80	**87.04**	94.14	84.38	64.25	91.08	2.11	72.72
MCSVM UNARY	74.55	79.75	*76.43*	80.77	84.88	92.68	84.79	69.99	93.54	9.73	80.03
CS POTTS	73.71	79.24	76.04	*83.86*	86.41	*95.07*	82.88	67.10	91.65	3.46	79.22
PASSIVE RP	73.72	80.63	76.08	83.47	*86.68*	**95.10**	82.49	67.58	91.62	3.32	79.50
LEARNED RP 20M	76.82	**82.08**	**77.85**	83.83	86.39	94.04	86.58	71.32	93.55	16.99	81.87
LEARNED RP 40M	**78.35**	81.28	74.73	**84.02**	83.79	93.05	**86.92**	*74.53*	93.33	**21.35**	*89.77*
LEARNED RP 50M	76.62	80.65	71.76	83.79	83.07	92.10	86.65	73.93	*94.10*	*17.94*	81.39
LEARNED MRP	*78.07*	*81.61*	72.43	80.50	86.63	93.99	*86.72*	**75.51**	**94.31**	14.80	**92.13**

Πίνακας 2.5: Ακρίβειες ταξινομητών. Οι ακρίβειες με έντονο μαύρο είναι οι υψηλότερες που επιτεύχθηκαν (Πηγή: Michele Volpi et al.)

2.2.6 OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks [Pierre Sermanet et al.]

Στο συγκεκριμένο paper, προτάθηκε μια ολοκληρωμένη μέθοδος για ταξινόμηση,

εντοπισμό και ανίχνευση του κύριου αντικειμένου μιας εικόνας. Οι τρεις αυτές διαδικασίες πραγματοποιήθηκαν ταυτόχρονα με τη βοήθεια ενός συνελικτικού μοντέλου (ConvNet). Το προτεινόμενο μοντέλο εφαρμόστηκε στα δεδομένα ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2013, 1000 κλάσεων, και βγήκε 4ο στην ταξινόμηση, 1ο στον εντοπισμό και 1ο στην ανίχνευση. Επίσης, στο paper αυτό δημοσιεύτηκε και ένας εξαγωγέας χαρακτηριστικών (feature extractor) που ονομάζεται 'OferFeat' και προέκυψε από το καλύτερο μοντέλο ConvNet που προέκυψε.

2.2.7 Very Deep Convolutional Networks for Large-Scale Image Recognition [Karen Simonyan ∗ & Andrew Zisserman +]

Στο συγκεκριμένο paper εξετάστηκε η σημαντικότητα του βάθους (depth) των συνελικτικών μοντέλων (ConvNets). Συγκεκριμένα, οι διάφορες εκδοχές μοντέλων που εξετάστηκαν φτάνουν μέχρι και τα 19 επίπεδα (layers). Το προτεινόμενο μοντέλο εφαρμόστηκε στα δεδομένα ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2012 τα οποία περιλαμβάνουν 1000 κλάσεις. Επίσης, τα μοντέλα που δημιουργήθηκαν αποδυκνείεται ότι μπορούν να χρησιμοποιηθούν και σε άλλα δεδομένα, παράγοντας ίσες ή καλύτερες ακρίβειες. Η προσέγγιση αυτή εξασφάλισε την πρώτη και δεύτερη θέση, στον εντοπισμό και την ανίχνευση αντίστοιχα.

2.2.8 ImageNet Classification with Deep Convolutional Neural Networks [Alex Krizhevsky et al.]

Στο paper αυτό, τα δεδομένα ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2010, 1000 κλάσεων, χρησιμοποιήθηκαν για την εκπαίδευση ενός συνελικτικού μοντέλου (ConvNet) πολύ μεγάλου βάθους. Δοκιμάστηκαν διάφοροι παράμετροι καταλήγοντας στο τελικό μοντέλο που περιέχει 60 εκατομμύρια παραμέτρους και 650.000 νευρώνες. Επίσης, για την αποφυγή της υπερπροσαρμογής (overfitting) του μοντέλου έγινε αύξηση των δεδομένων εκπαίδευσης (data augmentation). Η αύξηση αυτή έγκειται στη μεταβολή της έντασης των καναλιών RGB και του προσανατολισμού των patches.

2.2.9 Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks [Dimitrios Marmanis et al.]

Η δημοσίευση αυτή αφορά την αντιμετώπιση της ταξινόμησης σε περίπτωση που τα διαθέσιμα δεδομένα είναι περιορισμένα. Συγκεκριμένα, η ταξινόμηση χωρίστηκε σε δύο στάδια. Στο πρώτο στάδιο χρησιμοποιήθηκε το προεκπαιδευμένο μοντέλο Overfeat [P. Sermanet et al] του ImageNet με σκοπό την εξαγωγή χαρακτηριστικών από τα δεδομένα. Στο δεύτερο στάδιο, τα χαρακτηριστικά που είχαν εξαχθεί εισήχθησαν σε ένα δεύτερο μοντέλο CNN για να πραγματοποιηθεί η διαδικασία εκπαίδευσης. Η προσέγγιση αυτή εφαρμόστηκε στα δεδομένα UC Merced Land Use πετυχαίνοντας πολύ υψηλές ακρίβειες.

2.2.10 Rich feature hierarchies for accurate object detection and semantic segmentation [Ross Girshick et al.]

Το αντικείμενο μελέτης αυτής της δημοσίευσης ήταν η βελτίωση της διαδικασίας ανίχνευσης αντικειμένου (object-detection) σε εικόνες. Για το σκοπό αυτό, εξήχθησαν υποπεριοχές ανεξάρτητων κατηγοριών από τις εικόνες, τα χαρακτηριστικά των

οποίων προέκυψαν από ένα συνελικτικό μοντέλο CNN. Έπειτα, τα χαρακτηριστικά εισήχθηκαν σε ένα γραμμικό μοντέλο SVM. Η συγκεκριμένη αρχιτεκτονική ονομάστηκε R-CNN (Region-CNN) λόγω της χρήσης των υποπεριοχών των εικόνων. Επιπρόσθετα, προτάθηκε ένας τρόπος ταξινόμησης σε περίπτωση που οι διαθέσιμες εικόνες είναι περιορισμένες. Ο τρόπος αυτός περιλαμβάνει την επιβλεπόμενη προεκπαίδευση ενός μοντέλου με τη βοήθεια ενός συνόλου δεδομένων με παρόμοιο περιεχόμενο. Μετά το τέλος της προεκπαίδευσης, το μοντέλο προσαρμόστηκε στα περιορισμένα δεδομένα ενδιαφέροντος πετυχαίνοντας καλύτερα αποτελέσματα. Η προσέγγιση αυτή χρησιμοποιήθηκε στην ομάδα δεδομένων PASCAL VOC 2012, και ξεπέρασε κατά 30% τις ήδη υπάρχουσες ακρίβειες.

3. Μεθοδολογία

Στο κεφάλαιο αυτό περιγράφεται η μεθοδολογία που ακολουθήθηκε για την εκπαίδευση των μοντέλων και ο τρόπος αξιολόγησης των αποτελεσμάτων. Περιγράφονται επίσης τα λογισμικά που χρησιμοποιήθηκαν και οι εφαρμογές που δημιουργήθηκε στο Orfeo Toolbox.

3.1 Δεδομένα εκπαίδευσης

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, τα μοντέλα 'deep learning' χρειάζονται δεδομένα εκπαίδευσης προκειμένου να σχηματιστούν και να είναι ικανά να προβλέπουν οποιοδήποτε άγνωστο στοιχείο τους δωθεί. Τα δεδομένα αυτά είναι ουσιαστικά εικόνες ιδιαίτερα μικρών διαστάσεων (patches) της τάξεως του 20x20 pixels. Η εκπαίδευση των μοντέλων με patches, πραγματοποιήθηκε ξεχωριστά για 3 ομάδες δεδομένων οι οποίες περιγράφονται με λεπτομέρεια παρακάτω.

3.1.1 Δεδομένα DeepSat

Οι μικρές εικόνες (image patches) DeepSat [Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert Dibiano, Manohar Karki and Ramakrishna Nemani, DeepSat - A Learning framework for Satellite Imagery], προέρχονται από την ευρύτερη ομάδα δεδομένων του προγράμματος NAIP(National Agriculture Imagery Program), η οποία περιέχει περίπου 330.000 δορυφορικές εικόνες που αφορούν την περιοχή των Κεντρικών Ηνωμένων Πολιτειών. Η συγκεκριμένη ομάδα έχει μέγεθος περίπου 65 terabytes και κάθε εικόνα έχει 4 κανάλια: κόκκινο, πράσινο, μπλε και υπέρυθρο(NIR). Επίσης, η χωρική ανάλυση ισούται με 1 μέτρο, ενώ η οριζόντια ακρίβεια φτάνει τα 6 μέτρα.

Τα δεδομένα DeepSat αποτελούνται από patches διαστάσεων 28x28 και είναι χωρισμένα σε δύο διακεκριμένες ομάδες με ονόματα SAT-4 και SAT-6. Κάθε ομάδα περιέχει δύο διαφορετικά σύνολα από patches: ένα για τη διαδικασία εκπαίδευσης(training) και ένα για την έλεγχο του μοντέλου(testing). Η ομάδα SAT-4 αποτελείται από 4 διαφορετικές κλάσεις οι οποίες είναι: γυμνό έδαφος (Barren Land), δέντρα (Trees), γρασίδι (Grassland) και μια τέταρτη κλάση στην οποία ταξινομείται οτιδήποτε δεν ανήκει στις προηγούμενες τρεις (Other). Περιέχει 400.000 patches εκπαίδευσης και 100.000 patches ελέγχου. Όσο αφορά τη δεύτερη ομάδα, με όνομα SAT-6, περιέχει 6 διαφορετικές κλάσεις οι οποίες είναι: γυμνό έδαφος (Barren Land), δέντρα (Trees), γρασίδι (Grassland), δρόμος (Roads), κτίρια (Buildings) και νερό (Water Bodies). Περιέχει 324.000 patches εκπαίδευσης και 81000 patches ελέγχου.

Τα παραπάνω δεδομένα δίνονται σε μορφή πολυδιάστατων πινάκων .mat οι οποίοι περιέχουν τις ακόλουθες μεταβλητές:

SAT-4

- **train_x** : πίνακας μεγέθους 28x28x4x400.000 και είδους uint8 (400.000 patches εκπαίδευσης διαστάσεων 28x28 με 4 κανάλια)
- **train_y** : πίνακας μεγέθους 400.000x4 και είδους uint8 (400.000 υποπίνακες μεγέθους 1x4 που περιέχουν αριθμούς 0 και 1 για την αντιστοίχιση των κλάσεων των patches εκπαίδευσης)
- **test_x** : πίνακας μεγέθους 28x28x4x100.000 και είδους uint8 (400.000 patches ελέγχου διαστάσεων 28x28 με 4 κανάλια)

- **test_y** : πίνακας μεγέθους 100.000x4 και είδους uint8 (400.000 υποπίνακες μεγέθους 1x4 που περιέχουν αριθμούς 0 και 1 για την αντιστοίχιση των κλάσεων των patches ελέγχου)

SAT-6

- **train_x** : πίνακας μεγέθους 28x28x4x324.000 και είδους uint8 (324.000 patches εκπαίδευσης διαστάσεων 28x28 με 4 κανάλια)
- **train_y** : πίνακας μεγέθους 324.000x6 και είδους uint8 (324.000 υποπίνακες μεγέθους 1x6 που περιέχουν αριθμούς 0 και 1 για την αντιστοίχιση των κλάσεων των patches εκπαίδευσης)
- **test_x** : πίνακας μεγέθους 28x28x4x81.000 και είδους uint8 (81.000 patches ελέγχου διαστάσεων 28x28 με 4 κανάλια)
- **test_y** : πίνακας μεγέθους 81.000x6 και είδους uint8 (81.000 υποπίνακες μεγέθους 1x6 που περιέχουν αριθμούς 0 και 1 για την αντιστοίχιση των κλάσεων των patches ελέγχου)

Στην περίπτωση των δεδομένων Deepsat, τα έτοιμα patches χρησιμοποιούνται χωρίς καμία προεργασία για την εκπαίδευση και τον έλεγχο του εκάστοτε μοντέλου. Χρησιμοποιούνται διάφορα είδη μοντέλων και στο τέλος συγκρίνονται τα αποτελέσματα. Στην Εικόνα 3.1 φαίνονται μερικά από τα διαθέσιμα patches αυτής της ομάδας δεδομένων.

Εικόνα 3.1: Δεδομένα Deepsat
Παραδείγματα από patches

3.1.2 Δεδομένα 'Zurich Summer Dataset v1.0'

Η δεύτερη ομάδα δεδομένων που χρησιμοποιήθηκε ονομάζεται “Zurich Summer Dataset v1.0” και αποτελείται από 20 εικόνες μορφής geotiff του δορυφόρου Quickbird που απεικονίζουν μέρος της Ζυρίχης τον Αύγουστο του 2002. Τα δεδομένα αυτά έχουν ταξινομηθεί από ερευνητές με τη μέθοδο CRF (Conditional Random Field) χρησιμοποιώντας 19 εικόνες για εκπαίδευση και 1 εικόνα για έλεγχο [Volpi, M. & Ferrari, V.; Semantic segmentation of urban scenes by learning local class interactions]. Κάθε εικόνα έχει διαφορετικές διαστάσεις της τάξης του 1000x1100. Επίσης, όλες οι εικόνες αποτελούνται από 4 κανάλια (υπέρυθρο-κόκκινο-πράσινο-μπλε) και έχουν χωρική ανάλυση ίση με 0.61 μέτρα. Για κάθε εικόνα υπάρχει το αντίστοιχο groundtruth σε μορφή geotiff το οποίο περιέχει 8 κλάσεις. Αυτές είναι: δρόμοι (Roads), κτίρια (Buildings), δέντρα (Trees), γρασίδι (Grassland), γυμνό έδαφος (Bare Soil), νερό (Water), σιδηρόδρομοι (Railways) και πισίνες (Swimming Pool). Επίσης, υπάρχουν και κάποιες αταξινόμητες περιοχές, κυρίως στα όρια των κλάσεων που αναφέρθηκαν, οι οποίες δε χρησιμοποιούνται κατά τη διαδικασία εκπαίδευσης.

Σημειώνεται εδώ ότι κάθε μία από τις 20 εικόνες περιλαμβάνει το πολύ 7 από τις 8 κλάσεις. Στις Εικόνες 3.2 και 3.3 φαίνεται μία από τις 20 εικόνες της συγκεκριμένης

ομάδας δεδομένων μαζί με το αντίστοιχο groundtruth. Με μαύρο χρώμα απεικονίζονται οι δρόμοι (Roads), με γκρι χρώμα τα κτίρια (Buildings), με σκούρο πράσινο τα δέντρα (Trees), με ανοιχτό πράσινο το γρασίδι (Grass), με καφέ χρώμα το γυμνό έδαφος (Bare Soil), με κίτρινο χρώμα οι σιδηρόδρομοι (Railways) και με γαλάζιο οι πισίνες (Swimming Pools). Η κλάση νερό (Water) απεικονίζεται με κόκκινο χρώμα αλλά δεν εμπεριέχεται στη συγκεκριμένη εικόνα. Τα σημεία με άσπρο χρώμα αποτελούν αταξινόμητες περιοχές.

Στην περίπτωση των δεδομένων 'Zurich Summer Dataset v1.0', σε αντίθεση με την ομάδα δεδομένων Deepsat, τα patches εκπαίδευσης σχηματίστηκαν με τη βοήθεια του λογισμικού Matlab.
Από τις 20 εικόνες, οι 18 χρησιμοποιήθηκαν για εκπαίδευση και οι υπόλοιπες 2 για έλεγχο του μοντέλου. Τα patches εκπαίδευσης, κόπηκαν από κάθε εικόνα ακολουθώντας τα εξής βήματα:

1. Μετρήθηκε το πλήθος των pixels που αντιστοιχούσε σε κάθε κλάση με τη βοήθεια του groundtruth.

2. Για κάθε κλάση επιλέχθηκε τυχαία το 10% του πλήθους των pixels.

3. Το κάθε pixel χρησιμοποιήθηκε ως κέντρο για τη δημιουργία patches εκπαίδευσης διαφόρων διαστάσεων. Συγκεκριμένα, οι διαστάσεις που δημιουργήθηκαν είναι οι εξής:

- 5x5
- 11x11
- 21x21
- 29x29
- 33x33

Τα παραπάνω βήματα εφαρμόστηκαν και για τις 18 εικόνες. Τα patches αποθηκεύονταν κάθε φορά σε πίνακα τύπου .mat και διαστάσεων nxbxwxh, όπου n ο αριθμός των patches, b ο αριθμός των καναλιών και w,h οι διαστάσεις τους. Όλες οι διαστάσεις χρησιμοποιούνται για την εκπαίδευση του ίδιου μοντέλου και στο τέλος εφαρμόζεται έλεγχος του μοντέλου στις 2 εικόνες ελέγχου. Συγκρίνονται τα αποτελέσματα και οι διαστάσεις με τις καλύτερες ακρίβειες χρησιμοποιούνται έπειτα και για την εκπαίδευση άλλων μοντέλων.

Εικόνα 3.2: Εικόνα από την ομάδα δεδομένων 'Zurich Summer Dataset v1.0'

Εικόνα 3.3: Groundtruth για την Εικόνα 3.1

3.1.3 Δεδομένα DEIMOS-2

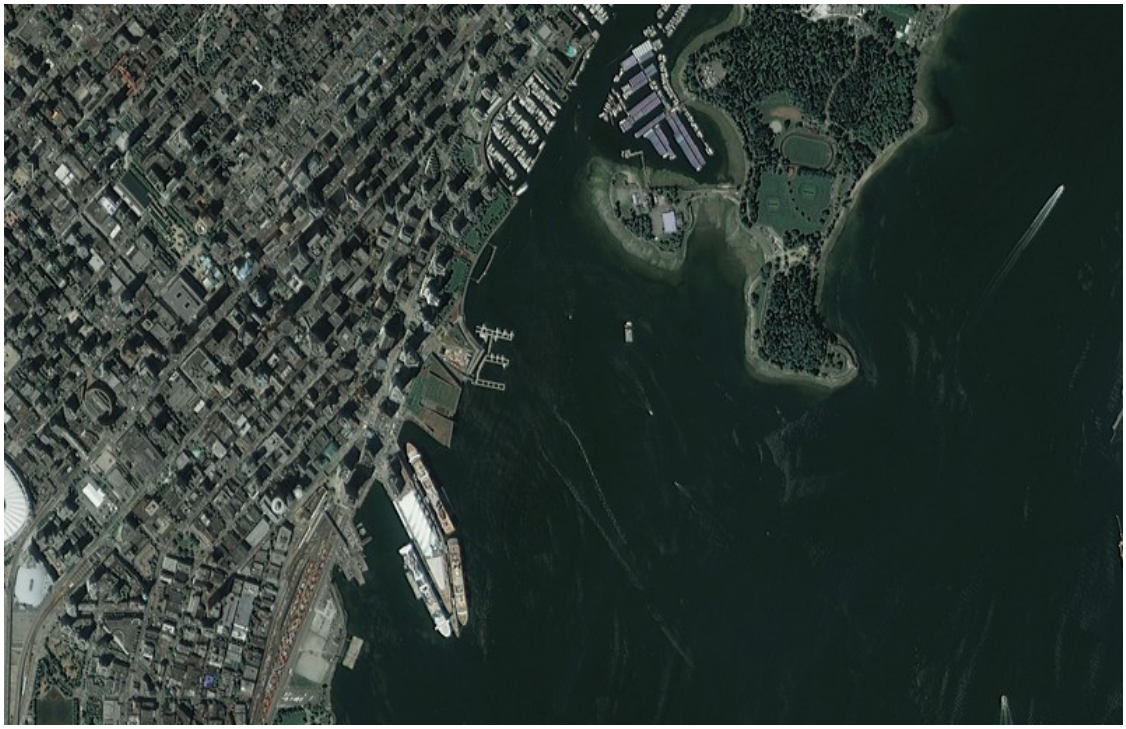
Η τρίτη ομάδα δεδομένων που χρησιμοποιήθηκε δημοσιεύθηκε από τον οργανισμό IADFTC(Image Analysis and Data Fusion Technical Committee) για την υλοποίηση του διαγωνισμού IEEE GRSS Data Fusion που διοργανώθηκε το 2016. Αποτελείται από μία παγχρωματική εικόνα χωρικής ανάλυσης 1 μέτρου, μία πολυφασματική εικόνα (RGB,NIR) χωρικής ανάλυσης 4 μέτρων καθώς και από ένα βίντεο υψηλής ανάλυσης χωρικής ανάλυσης 1 μέτρου. Όλα τα δεδομένα απεικονίζουν μέρος της πόλης του Vancouver των Ηνωμένων Πολιτειών σε διαφορετικές χρονικές στιγμές.

Οι εικόνες που αναφέρθηκαν τραβήχτηκαν από το δορυφόρο DEIMOS-2 στις 31 Μαρτίου 2015 και 1 Μαΐου 2015. Είναι ραδιομετρικά διορθωμένες και έχουν υποστεί pansharpening με τη μέθοδο High Pass Filter δίνοντας ως αποτέλεσμα ένα επικαλυπτόμενο ζεύγος εικόνων μεγέθους περίπου 12760x11000 pixels. Τα επικαλυπτόμενα ζεύγη εικόνων του βίντεο τραβήχτηκαν από την κάμερα Iris που βρίσκεται πάνω στη μονάδα Zvezda του Διεθνούς Διαστημικού Σταθμού ISS (International Space Station) και έχουν μέγεθος περίπου 4720x2680 pixels. Στις Εικόνες 3.4, 3.5 και 3.6 δίνεται και η γραφική απεικόνιση των προαναφερθέντων εικόνων.

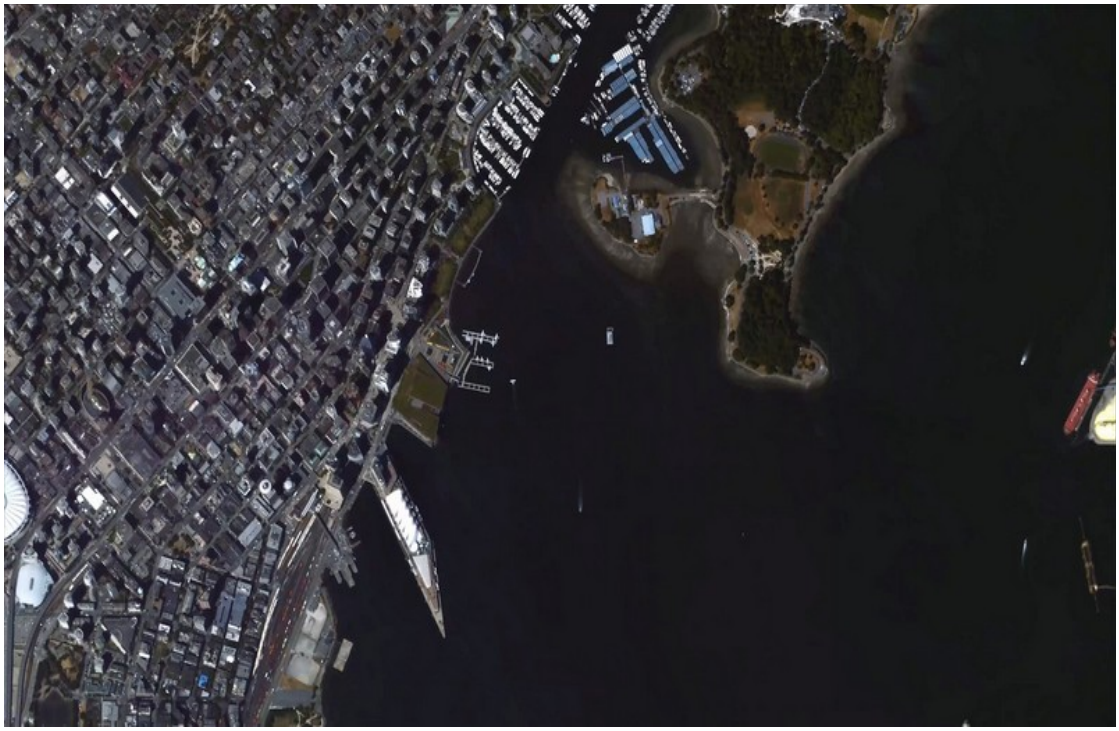
Ο δορυφόρος DEIMOS-2 λειτουργεί σε μια ηλιοσύγχρονη τροχιά με μέσο υψόμετρο 620 χιλιόμετρα. Αποτλείται από κάμερα υψηλής ανάλυσης και περιέχει 5 φασματικά κανάλια: 1 παγχρωματικό και 4 πολυφασματικά (κόκκινο, πράσινο, μπλε, υπέρυθρο). Όσο αφορά την κάμερα IRIS, χρησιμοποιεί έναν ανιχνευτή CMOS (complementary metal oxide semiconductor), έχει 3 φασματικά κανάλια (κόκκινο, πράσινο, μπλε), διαθέτει χωρική ανάλυση 1 μέτρου και παράγει 3 εικόνες ανά δευτερόλεπτο.

Τα παραπάνω δεδομένα χρησιμοποιήθηκαν για την εκπαίδευση ενός μοντέλου. Συγκεκριμένα, patches εκπαίδευσης μεγέθους 4x21x21 κόπηκαν από την επικαλυπτόμενη περιοχή μεγέθους 12760x11000 που προέκυψε από τις εικόνες DEIMOS-2. Σχηματίστηκαν 8 κλάσεις οι οποίες είναι οι εξής: δρόμοι (Roads), κτίρια (Buildings), σκιές κτιρίων (Buildings Shadows), έδαφος (Soil), θάλασσα (Sea), πλοίο (Boat), βλάστηση (Vegetation) και σκιές βλάστησης (Vegetation Shadows). Για κάθε κλάση κόπηκαν περίπου 200.000 patches. Όσο αφορά την εικόνα από το βίντεο της κάμερας IRIS, οι κλάσεις βλάστηση (Vegetation) και σκιές βλάστησης (Vegetation Shadows) συγχωνεύθηκαν σε μία με αποτέλεσμα να σχηματιστούν 7 κλάσεις ίδιες με αυτές των εικόνων DEIMOS-2. Τα patches για την εικόνα του βίντεο είχαν διαστάσεις 3x21x21. Και στις δύο περιπτώσεις κόπηκαν επίσης και patches ελέγχου που χρησιμοποιήθηκαν κατά τη διάρκεια της εκπαίδευσης για τον έλεγχο του μοντέλου. Τέλος, πρέπει να σημειωθεί ότι τα σύννεφα που υπάρχουν στην εικόνα DEIMOS-2 του Μαρτίου προσδιορίστηκαν χειροκίνητα. Αφού ολοκληρώθηκε διαδικασία εκπαίδευσης το μοντέλο χρησιμοποιήθηκε για την ταξινόμηση ολόκληρων των εικόνων. Μετά το πέρας της παραπάνω διαδικασίας, παρατηρήθηκαν οι αλλαγές που έχουν σημειωθεί σε κάθε χρονική στιγμή μέσα από γραφική επεικόνιση.

Εικόνα 3.4: Περιοχή Vancouver, Ηνωμένες Πολιτείες
Εικόνα DEIMOS-2 - 15 Μαρτίου 2015



Εικόνα 3.5: Περιοχή Vancouver, Ηνωμένες Πολιτείες
Εικόνα DEIMOS-2 - 15 Μαΐου 2015



Εικόνα 3.6: Περιοχή Vancouver, Ηνωμένες Πολιτείες
Εικόνα IRIS (βίντεο) - 17 Ιουλίου 2015

3.2 Μοντέλα ταξινόμησης με βάση αρχιτεκτονικές Deep Learning

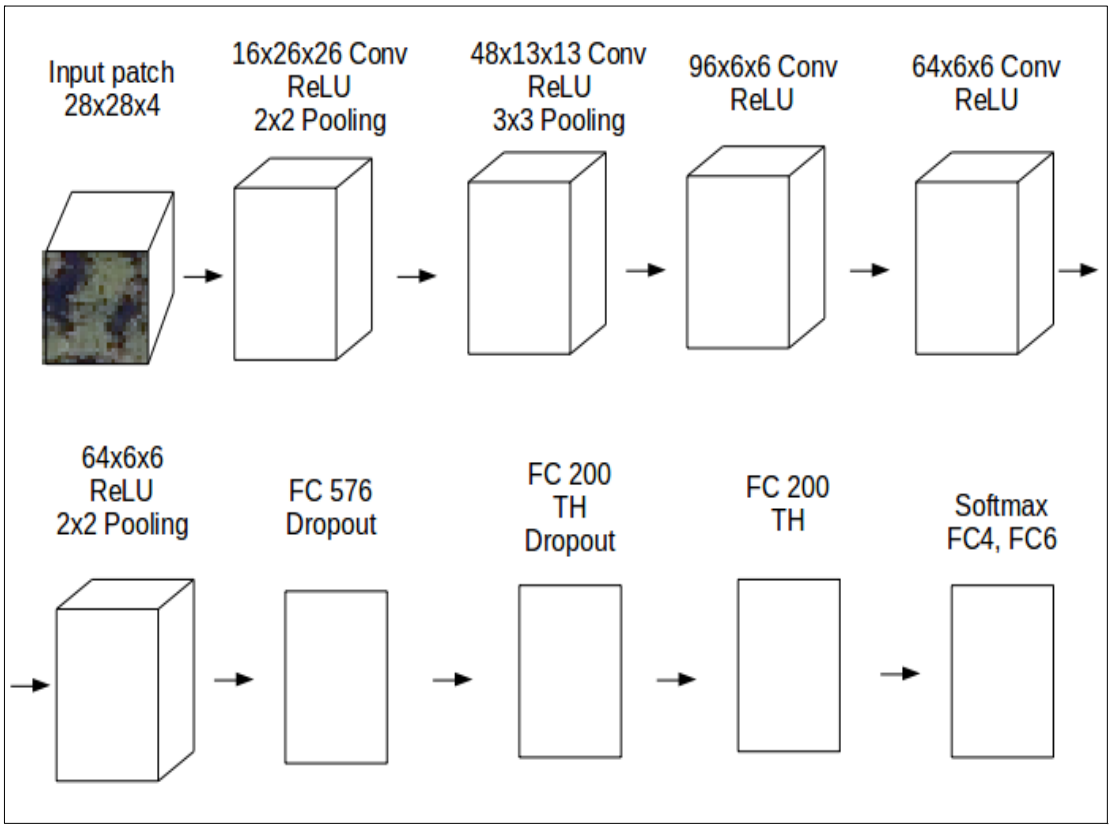
Για κάθε ομάδα δεδομένων χρησιμοποιήθηκαν μοντέλα ταξινόμησης τα οποία κάθε φορά προσαρμόζονταν στις διαστάσεις τους. Η εκπαίδευση των μοντέλων έγινε με χρήση κάρτας GPU για τη γρηγορότερη πραγματοποίηση της όλης διαδικασίας. Τα μοντέλα που ακολουθούν, περιγράφονται με βάση την ομάδα δεδομένων Deepsat. Σε επόμενο κεφάλαιο αναφέρονται οι αλλαγές που χρειάζεται να γίνουν σε κάθε μοντέλο εξαιτίας των διαφορετικών διαστάσεων των patches κάθε ομάδας δεδομένων.

3.2.1 AlexNet

Το συγκεκριμένο μοντέλο αποτελείται από 22 επίπεδα: 5 συνελικτικά (convolutional), 3 pooling, 6 μεταφοράς συνάρτησης(transfer function) και 8 πλήρως συνδεδεμένα (fully-connected). Για την καλύτερη κατανόηση της δομής αυτής, η μορφή του μοντέλου δίνεται σχηματικά στην Εικόνα 3.7. Πιο συγκεκριμένα, το πρώτο συνελικτικό επίπεδο δέχεται ως είσοδο το κάθε patch, το οποίο περιλαμβάνει 4 κανάλια και έχει διαστάσεις 28x28. Το εκάστοτε patch θέτεται σε επεξεργασία από φίλτρα μεγέθους 4x3x3 και βήματος 1 pixel. Μετά το τέλος της επεξεργασίας το κάθε patch έχει μετασχηματιστεί σε πίνακα διαστάσεων 16x26x26, ο οποίος δίνεται ως είσοδος στο δεύτερο επίπεδο του μοντέλου το οποίο εφαρμόζει τη συνάρτηση ReLU (Rectified Linear Unit) στον πίνακα, αφήνοντας τις διαστάσεις αμετάβλητες. Το τρίτο επίπεδο είναι είδους max pooling και μειώνει το μέγεθος του πίνακα προκειμένου να μειώσει τον όγκο υπολογισμών του δκτύου και να αποφευχθεί το φαινόμενο του overfitting. Το συγκεκριμένο επίπεδο χρησιμοποιεί φίλτρα διαστάσεων 4x2x2 με βήμα 2, δίνοντας ως έξοδο ένα πίνακα διαστάσεων 16x13x13. Τα επίπεδα 4,5 και 6

ακολουθούν την ίδια λογική (Convolutional-ReLU-MaxPooling). Το συνελικτικό επίπεδο δέχεται ως είσοδο τον πίνακα διαστάσεων 16x13x13 μετασχηματίζοντάς τον σε 48x13x13 χρησιμοποιώντας φίλτρα 4x3x3 βήματος 1 και zero padding ίσο με 1. Ακολουθούν τα επίπεδα ReLU και MaxPooling που παράγουν ένα πίνακα διαστάσεων 48x6x6 μετά από εφαρμογή φίλτρων διαστάσεων 4x3x3 και βήματος 2. Ακολουθεί το έβδομο συνελικτικό επίπεδο με φίλτρα μεγέθους 4x3x3 και βήματος 1, καθώς και zero padding ίσο με 1. Μετά από αυτό ο υπό επεξεργασία πίνακας έχει διαστάσεις 96x6x6 και δίνεται ως είσοδος στο όγδοο επίπεδο το οποίο εφαρμόζει τη συνάρτηση ReLU. Τα επίπεδα 9,10 και 11,12 ακολουθούν το ίδιο μοτίβο (Convolutional-ReLU). Το ένατο επίπεδο χρησιμοποιεί φίλτρα 4x3x3, βήμα 1 και zero padding ίσο με 1, παράγοντας ένα πίνακα 64x6x6. Το ενδέκατο επίπεδο χρησιμοποιεί τις ίδιες παραμέτρους. Το δωδέκατο επίπεδο είναι είδους MaxPooling και έχει φίλτρα 4x2x2 και βήμα 2 δίνοντας ως έξοδο ένα πίνακα 64x3x3.

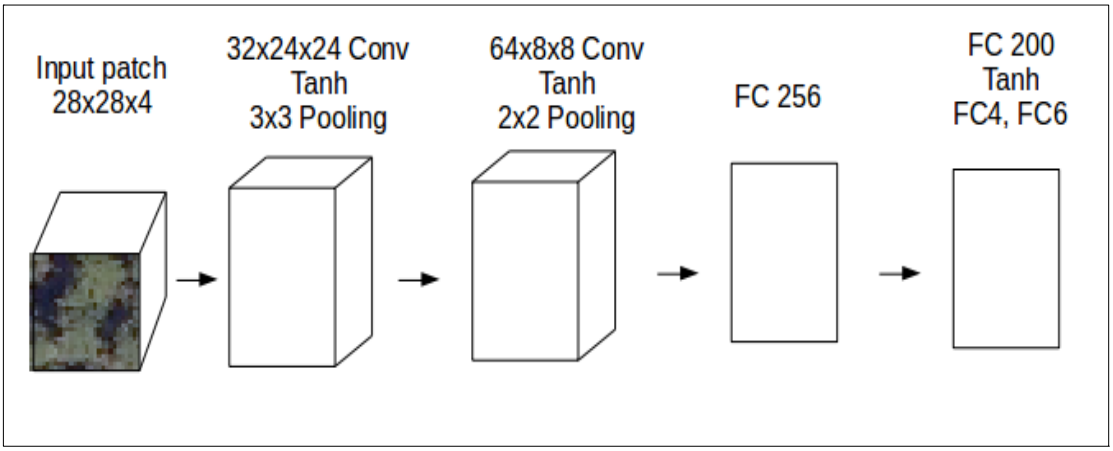
Μετά από αυτό, χρησιμοποιούνται μερικά πλήρως συνδεδεμένα επίπεδα. Το δέκατο τρίτο επίπεδο μετατρέπει τον τρισδιάστατο πίνακα 64x3x3 σε μονοδιάστατο με μέγεθος 64*3*3x1=576x1. Ακολουθεί ένα επίπεδο είδους Dropout, το οποίο με τη βοήθεια της κατανομής Bernoulli δέχεται ή απορρίπτει στοιχεία του πίνακα τα οποία έχουν πιθανότητα 0.5. Το δέκατο πέμπτο επίπεδο είναι γραμμικό και μετασχηματίζει το μέγεθος του πίνακα σε 200x1. Ακολουθεί ένα επίπεδο Threshold. Τα τρία τελευταία επίπεδα έχουν και πάλι τη σειρά Dropout-Linear-Threshold με ίδιες παραμέτρους. Το τελευταίο επίπεδο είναι γραμμικό και δίνει ως αποτέλεσμα τον τελικό πίνακα διαστάσεων 4x1 ή 6x1, ο οποίος αντιστοιχεί στον αριθμό των κλάσεων ανάλογα με την υποομάδα των patches που χρησιμοποιείται. Στην Εικόνα 3.7 δίνεται και η σχηματική απεικόνιση του συγκεκριμένου μοντέλου για την ομάδα δεδομένων DeepSat.



Εικόνα 3.7: Σχηματική απεικόνιση της μορφής του μοντέλου AlexNet για την ομάδα δεδομένων Deepsat.

3.2.2 ConvNet

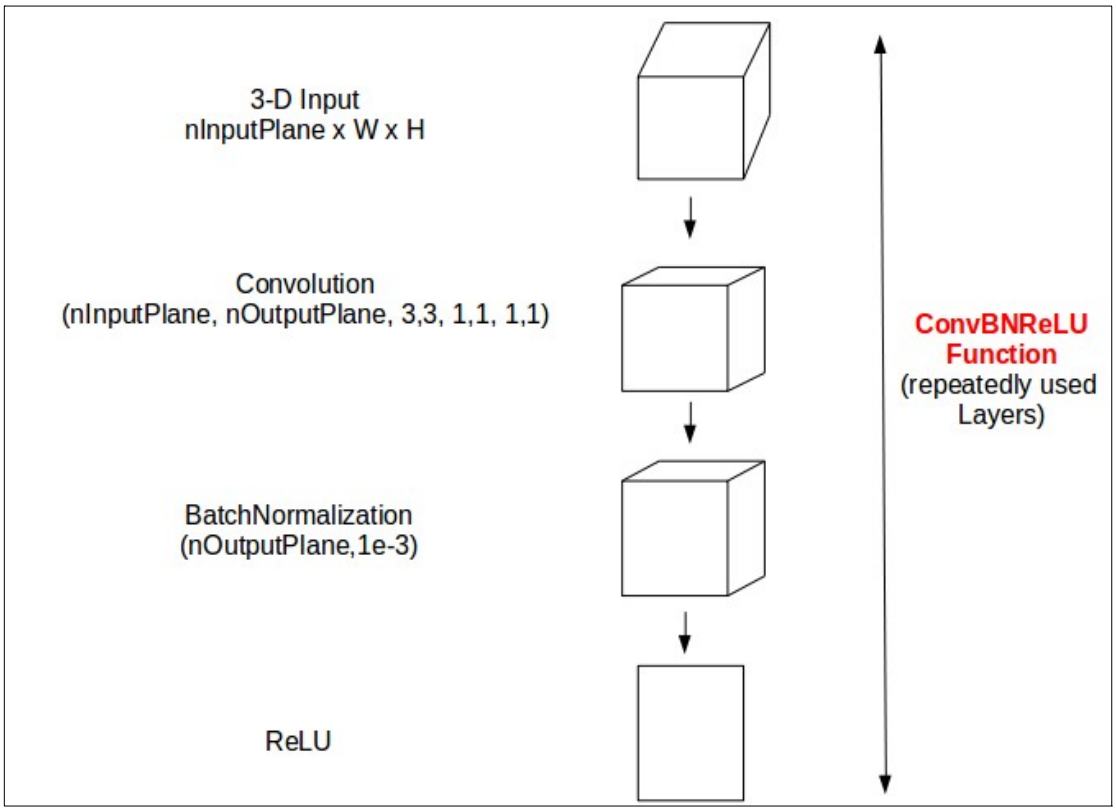
Το δεύτερο μοντέλο που χρησιμοποιήθηκε για την εκπαίδευση των δεδομένων είναι το ConvNet. Αποτελείται από 10 επίπεδα. Το πρώτο από αυτά είναι συνελικτικό και δέχεται ως είσοδο τα patches διαστάσεων 4x28x28 τα οποία επεξεργάζεται με φίλτρα 4x5x5, παράγοντας ένα πίνακα εξόδου 32x24x24. Το δεύτερο επίπεδο εφαρμόζει τη συνάρτηση tangent στον πίνακα. Το τρίτο επίπεδο, είναι είδους MaxPooling και δίνει ως αποτέλεσμα ένα πίνακα 32x8x8 εφαρμόζοντας φίλτρα 4x3x3 με βήμα 3. Τα επίπεδα 4,5 και 6 ακολουθούν την ίδια λογική (Convolutional-Tangent-MaxPooling). Το τέταρτο επίπεδο εφαρμόζει φίλτρα 4x5x5 ενώ το έκτο εφαρμόζει φίλτρα 4x2x2 με βήμα 2 καταλήγοντας σε ένα πίνακα διαστάσεων 64x2x2. Το έβδομο επίπεδο συμβάλλει στο μετασχηματισμό του πολυδιάστατου πίνακα 64x2x2 σε μονοδιάστατο πίνακα διαστάσεων 64*2*2x1=256x1. Το όγδοο επίπεδο μετατρέπει το μονοδιάστατο πίνακα από 256x1 σε 200x1 και το ένατο επίπεδο εφαρμόζει τη συνάρτηση tangent. Το τελευταίο επίπεδο δίνει ως έξοδο τον τελικό πίνακα διαστάσεων 4x1 ή 6x1 ανάλογα με την ομάδα των patches που χρησιμοποιείται. Στην Εικόνα 3.8 δίνεται και η σχηματική απεικόνιση του συγκεκριμένου μοντέλου.



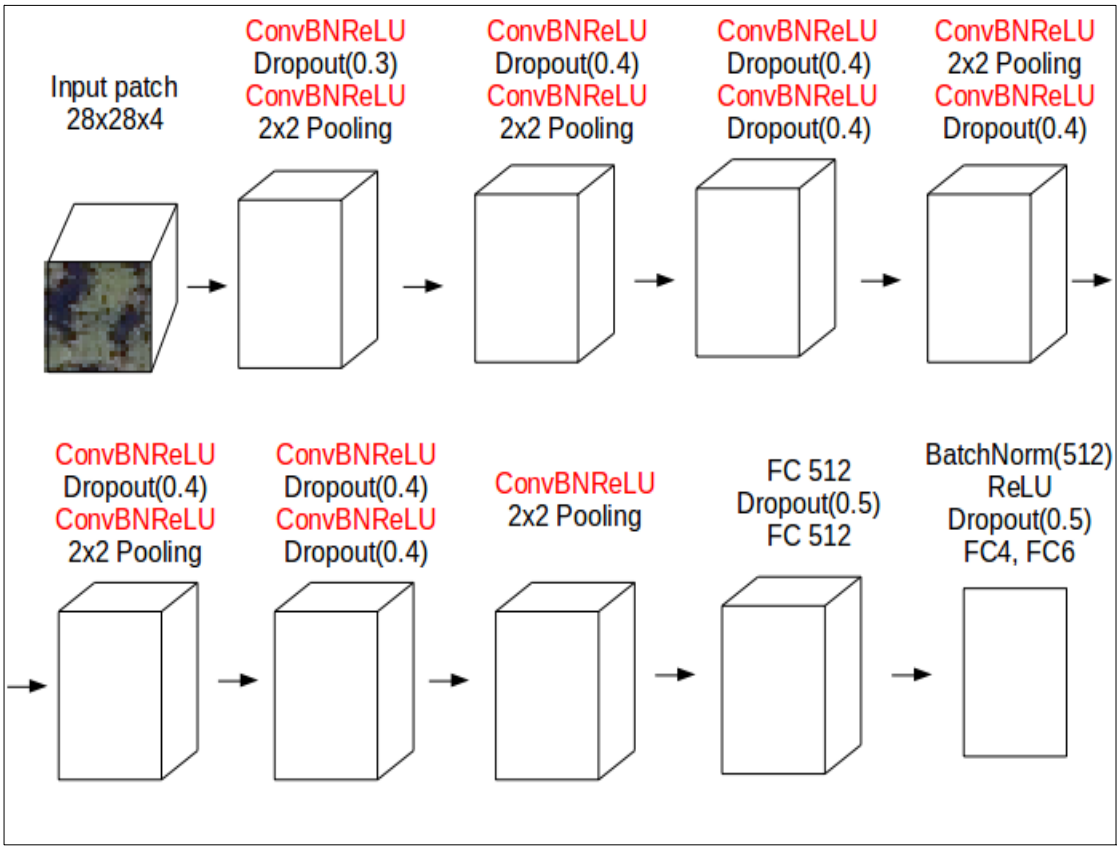
Εικόνα 3.8: Σχηματική απεικόνιση μοντέλου ConvNet για την ομάδα δεδομένων DeepSat

3.2.3 VGG Net

Το τρίτο μοντέλο εκπαίδευσης είναι το VGG, το οποίο αρχικά προτάθηκε και χρησιμοποιήθηκε για αναγνώριση κειμένου. Το συγκεκριμένο μοντέλο κάνει συνεχώς χρήση των επιπέδων dropout και batch normalization. Πρέπει να σημειωθεί ότι τα δύο αυτά είδη επιταχύνουν τη διαδικασία εκπαίδευσης και προστατεύουν από το φαινόμενο overfitting. Το μοντέλο στο σύνολό του αποτελείται από 59 επίπεδα. Τα τρία πρώτα έχουν την εξής μορφή: Αρχικά, τα patches διαστάσεων 4x28x28 δίνονται ως είσοδος σε ένα συνελικτικό επίπεδο το οποίο δίνει ως αποτέλεσμα ένα πίνακα διαστάσεων 64x28x28. Έπειτα ακολουθεί ένα επίπεδο batch normalization με τιμή 0.001 η οποία προσθετεται στην τυπική απόκλιση των πινάκων και τρίτο ακολουθεί ένα επίπεδο ReLU. Τα τρία αυτά επίπεδα επαναλαμβάνονται συνεχώς και λειτουργούν σαν ένα είδος συνάρτησης (Εικόνα 3.9) η οποία καλείται πολλές φορές. Επίσης, χρησιμοποιούνται και κάποια επίπεδα Dropout και MaxPooling. Τα τελευταία 7 επίπεδα του μοντέλου είναι πλήρως συνδεδεμένα και καταλήγουν στον αντίστοιχο αριθμό κλάσεων. Στην Εικόνα 3.10 δίνεται η σχηματική απεικόνιση του συγκεκριμένου μοντέλου για την ομάδα δεδομένων DeepSat.



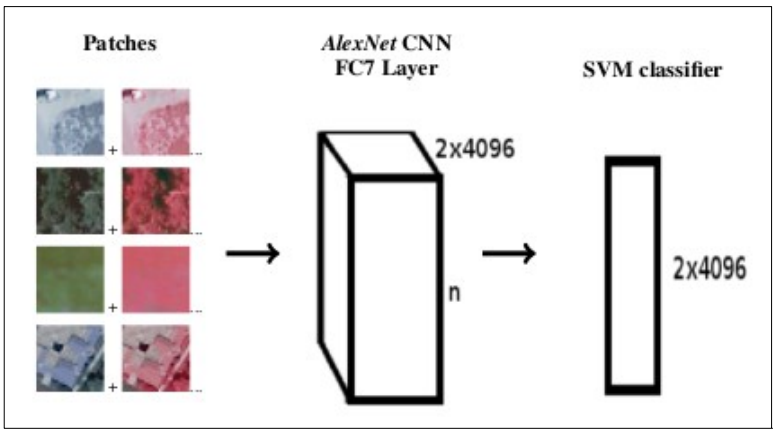
Εικόνα 3.9: Συνάρτηση ConvBNReLU του μοντέλου VGG



Εικόνα 3.10: Σχηματική απεικόνιση μοντέλου VGG

3.2.4 Ρηχές αρχιτεκτονικές και SVM

Εκτός από τα μοντέλα που περιγράφηκαν παραπάνω, χρησιμοποιήθηκε και η μέθοδος ταξινόμησης SVM (Support Vector Machine) [Christopher M. Bishop, Pattern Recognition and Machine Learning**]**. Η μέθοδος αυτή εφαρμόστηκε μόνο στην ομάδα δεδομένων Deepsat. Η εξαγωγή χαρακτηριστικών(feature extraction) από τα patches έγινε με τη βοήθεια του ήδη προεκπαιδευμένου μοντέλου AlexNet [Krizhevsky et al., 2012]. Συγκεκριμένα, εξήχθησαν τα χαρακτηριστικά του τελευταίου επιπέδου (FC7), χρησιμοποιώντας δύο συνδυασμούς φασματικών καναλιών (κόκκινο-πράσινο-μπλε και υπέρυθρο-κόκκινο-πράσινο). Με τον τρόπο αυτό σχηματίστηκε ένας πίνακας χαρακτηριστικών με διαστάσεις 2x4096 για κάθε patch ο οποίος χρησιμοποιήθηκε για την εκπαίδευση του αλγορίθμου SVM. Στην Εικόνα 3.8 δίνεται σχηματικά η μορφή της τεχνικής που περιγράφηκε παραπάνω.



Εικόνα 3.11: Σχηματική απεικόνιση της διαδικασίας εξαγωγής χαρακτηριστικών από την ομάδα δεδομένων Deepsat για την εκπαίδευση του αλγορίθμου SVM.

3.4 Αξιολόγηση αποτελεσμάτων

Η αξιολόγηση των αποτελεσμάτων ταξινόμησης έγινε κυρίως μέσα από την παρατήρηση του πίνακα σύγχησης. Συγκεκριμένα, για κάθε πίνακα σύγχησης βρέθηκαν οι ακρίβειες του χρήστη και του παραγωγού, η συνολική ακρίβεια και ο δείκτης K.

- Ακρίβεια χρήστη (User's accuracy)
  Υπολογίζεται για μια κατηγορία ταξινόμησης αν διαιρεθούν τα patches που σωστά ταξινομήθηκαν σε αυτήν, με το σύνολο των patches που έχουν ταξινομηθεί σε αυτήν. Δηλαδή στην πράξη, η συγκεκριμένη ακρίβεια μελετάται ανά στήλη σε ένα πίνακα σύγχησης.

- Ακρίβεια παραγωγού (Producer's accuracy).
  Υπολογίζεται για μια κατηγορία ταξινόμησης αν διαιρεθούν τα patches που σωστά ταξινομήθηκαν σε αυτήν με το σύνολο των patches που υπάρχουν για αυτή την κατηγορία. Δηλαδή στην πράξη, η ακρίβεια αυτή μελετάται ανά γραμμή σε ένα πίνακα σύγχησης.

- Συνολική ακρίβεια (Overall accuracy)

Υπολογίζεται αν το σύνολο των σωστά ταξινομημένων patches διαιρεθεί με το σύνολο των patches ελέγχου. Αρκεί δηλαδή το άθροισμα των στοιχείων της διαγωνίου ενός πίνακα σύγχησης να διαιρεθεί με το σύνολο των στοιχείων του.

- Δείκτης K (Kappa coefficient)
  Αποτελεί ισχυρότερο μέτρο αξιολόγησης των αποτελεσμάτων από τη συνολική ακρίβεια, αφού λαμβάνει υπόψη τόσο την ακρίβεια του χρήστη όσο και την ακρίβεια του παραγωγού.

$$K = \frac{Overall\ accuracy - p_c}{1 - p_c}$$

Εξίσωση 3.1: Δείκτης K (Kappa coefficient)

$$p_c = \frac{\sum_i n_{i.} n_{.j}}{n}$$

Εξίσωση 3.2: Ακρίβεια Pc
n_i=άθροισμα στοιχείων του πίνακα συγχησης ανά γραμμή, n_j=άθροισμα στοιχείων του πίνακα συγχησης ανά στήλη, n=Συνολικό άθροισμα στοιχείων του πίνακα σύγχησης

3.5 Υλοποίηση

Στην υποενότητα αυτή, περιγράφονται τα λογισμικά που χρησιμοποιήθηκαν για τη διεκπεραίωση της εργασίας αυτής.

3.5.1 Lua

Η υλοποίηση των συνελικτικών νευρωνικών δικτύων σε προγραμματιστικό περιβάλλον έγινε με τη βοήθεια της γλώσσας Lua [Roberto Ierusalimschy, Programming in Lua – Third Edition], η οποία ανήκει στα προγράμματα ελεύθερου λογισμικού και χαρακτηρίζεται από μια σειρά πλεονεκτημάτων.

Ένα από τα πλεονεκτήματα αυτά είναι η ταχύτητα με την οποία εκτελούνται τα αρχεία κώδικα. Τα αρχεία σε κώδικα Lua δεσμεύουν ελάχιστη μνήμη στον υπολογιστή χωρίς να επιβαρύνουν το λειτουργικό σύστημα, ενώ για την υλοποίησή τους το μόνο που απαιτείται είναι η ύπαρξη ενός C compiler. Επίσης, πραγματοποιούν αυτόματη διαγραφή των μεταβλητών που δε χρησιμοποιούνται πια από τον εκάστοτε αλγόριθμο, ελευθερώνοντας τη μνήμη του υπολογιστή από περιττή δέσμευση.

Ένα ακόμα θετικό στοιχείο της γλώσσας, είναι το ιδιαίτερα απλό συντακτικό που διαθέτει, το οποίο στηρίζεται κυρίως στη χρήση πινάκων. Παρά την απλότητα της μορφής της βέβαια, η Lua είναι ικανή να υποστηρίξει εξαιρετικά πολύπλοκες εφαρμογές. Είναι δηλαδή ιδιαίτερα ωφέλιμη για τους χρήστες, αφού η εκμάθησή της

μπορεί να πραγματοποιηθεί εύκολα και γρήγορα, προσφέροντας ταυτόχρονα σημαντικές προγραμματιστικές δυνατότητες.

Πρέπει επίσης να αναφερθεί, ότι η Lua ενσωματώνεται επιτυχώς σε άλλες γλώσσες προγραμματισμού λόγω του απλού API(Application Program Interface) που διαθέτει. Για το λόγο αυτό, σε περίπτωση που επιθυμείται η επέκταση ενός προγράμματος, η δημιουργία ενός εμβόλιμου εκτελέσιμου αρχείου σε κώδικα Lua αποτελεί μία εύκολη και γρήγορη λύση. Στη συγκεκριμένη εργασία, τα κύρια προγράμματα στα οποία ενσωματώνονται τα αρχεία σε κώδικα Lua, είναι γραμμένα σε C++.

3.5.2 Torch

Οι εφαρμογές των πρωτότυπων εκτελέσιμων αρχείων Lua έγιναν με τη βοήθεια του προγραμματιστικού περιβάλλοντος Torch [Collobert, R., Kavukcuoglu, K. and Farabet, C., 2011. Torch7: A Matlab-like Environment for Machine Learning], το οποίο αποτελεί βιβλιοθήκη μηχανικής εκμάθησης (machine learning) ανοιχτού λογισμικού. Είναι βασισμένο στη γλώσσα προγραμματισμού C και παρέχει αλγορίθμους 'deep learning'. Μερικά από τα βασικά χαρακτηριστικά του είναι τα εξής:

- Χρήση πινάκων με ιδιαίτερα μεγάλες διαστάσεις και ευέλικτη διαχείριση αυτών.
- Εύκολη σύνδεση με τη γλώσσα C++
- Αριθμητικές ρουτίνες
- Μοντέλα νευρωνικών δικτύων
- Δυνατότητα χρήσης GPU
- Ιδιαίτερα γρήγορη ταχύτητα

3.5.2.1 Εγκατάσταση

Τα βήματα για την εγκατάσταση του λογισμικού Torch σε περιβάλλον Linux είναι τα εξής:

1. Πρόσβαση του υπολογιστή στο αντίστοιχο αποθετήριο (repository)
git clone https://github.com/torch/distro.git ~/torch --recursive

2. Είσοδος στο σχετικό φάκελο του αποθετηρίου
cd ~/torch; bash install-deps;

3. Εγκατάσταση των απαραίτητων βιβλιοθηκών
./install.sh

4. Τελική αποθήκευση των βιβλιοθηκών στο σωστό path του υπολογιστή.
source ~/.bashrc

5. Προαιρετική εγκατάσταση επιπρόσθετων πακέτων
luarocks install <package name>

Εφόσον η εγκατάσταση πραγματοποιηθεί με επιτυχία, η είσοδος στο προγραμματιστικό περιβάλλον Torch γίνεται με την εντολή *th* στο τερματικό.

3.5.3 Orfeo Toolbox (OTB)

Στην υποενότητα αυτή αναφέρονται κάποια γενικά χαρακτηριστικά του λογισμικού Orfeo Toolbox και αναλύεται ο τρόπος εγκατάστασης και λειτουργίας του.

3.5.3.1 Γενικά στοιχεία του Orfeo Toolbox

Το Orfeo Toolbox αποτελεί μια βιβλιοθήκη ελεύθερου λογισμικού που αφορά εφαρμογές σε C++ [Bjarne Stroustrup, The C++ Programming Language – Fourth Edition] στον τομέα της τηλεπισκόπισης. Δημιουργήθηκε και προωθήθηκε για πρώτη φορά από το Εθνικό Κέντρο Διαστημικών Μελετών της Γαλλίας (CNES) (Centre national d'études spatiales) το 2006. Το συγκεκριμένο λογισμικό στηρίζεται σε διάφορες γλώσσες, οι πιο σημαντικές από τις οποίες είναι η GDAL και η ITK. Η GDAL δίνει τη δυνατότητα στο OTB να μπορεί να διαβάσει οποιοδήποτε είδος εικόνας, και η ITK αποτελεί τη βάση για όλες τις εφαρμογές του. Η τελευταία χρησιμοποιείται κυρίως σε ιατρικούς τομείς και πάνω σε αυτήν στηρίζονται όλοι οι αλγόριθμοι.

Από το 2006 μέχρι σήμερα, το φάσμα εφαρμογών του έχει επεκταθεί σε μεγάλο βαθμό, αφού όλο και περισσότεροι χρήστες εξοικειώνονται με τη λειτουργία του και συμβάλλουν στην ανάπτυξή του. Αναφέρονται επιγραμματικά κάποιες από τις πολυπληθείς εφαρμογές του, όπως η ανίχνευση μεταβολών, ο υπολογισμός των κυρίων συνιστωσών, η εφαρμογή φίλτρων σε εικόνες και η ταξινόμηση.

Η κοινότητα του λογισμικού Orfeo Toolbox αποτελείται από πολλούς και ενεργούς χρήστες που μπορούν να επικοινωνούν μέσα από ταχυδρομική λίστα, η οποία είναι προσβάσιμη από την αντίστοιχη επίσημη ιστοσελίδα(https://www.orfeo-toolbox.org/). Με τον τρόπο αυτό, η επίλυση προβλημάτων και αποριών γίνεται γρήγορα και άμεσα. Στην επίσημη ιστοσελίδα βρίσκεται επίσης λεπτομερής οδηγός εγκατάστασης, καθώς και διεξοδικές πληροφορίες για την εκτέλεση διαφόρων διαδικασιών τηλεπισκόπησης. Τέλος, οι ρουτίνες και τα πρωτόκολλα που χρησιμοποιούνται από το λογισμικό(API) αναλύονται εκτενώς στην ιστοσελίδα https://www.orfeo-toolbox.org/doxygen/.

3.5.3.2 Εγκατάσταση

Προκειμένου ο χρήστης να δημιουργεί εφαρμογές μέσα από τη βιβλιοθήκη του Orfeo Toolbox είναι απαραίτητο πρώτα να το εγκαταστήσει στο περιβάλλον του υπολογιστή του. Παρακάτω περιγράφονται συνοπτικά τα βήματα που χρειάζεται να ακολουθηθούν για την πραγματοποίηση της εγκατάστασης. Σημειώνεται ότι τα βήματα αυτά λειτουργούν μόνο σε περιβάλλον Linux.

Πριν αναφερθούν οι τεχνικές λεπτομέρειες της διαδικασίας εγκατάστασης, κρίνεται σκόπιμο να γίνει αναφορά στο πρόγραμμα ελεύθερου λογισμικού Cmake. Το συγκεκριμένο πρόγραμμα χρησιμοποιείται ως μέσο για την υλοποίηση του compilation σε προγράμματα που αδυνατούν να υποστηριχτούν σε διαφορετικά περιβάλλοντα υπολογιστών. Η αδυναμία υποστήριξης συνήθως προκύπτει όταν ένα πρόγραμμα στο σύνολό του περιλαμβάνει πολλές βιβλιοθήκες και εξαρτάται από αυτές. Το πρόγραμμα Cmake επιλύει το πρόβλημα αυτό, αφού μπορεί να χρησιμοποιηθεί ως ανεξάρτητος compiler για τη διαμόρφωση(configuration) και το

χτίσιμο (building) του πηγαίου κώδικα(source code) ενός προγράμματος. Το μόνο που χρειάζεται να υπάρχει στο εκάστοτε σύστημα υπολογιστή είναι ένας compiler σε C++.

Η σταθερή έκδοση του Cmake μπορεί να ληφθεί από την αντίστοιχη ιστοσελίδα στο GitHub(https://github.com/Kitware/CMake). Ο χρήστης, κατεβάζοντας το φάκελο με όνομα Cmake-master που περιέχει τον πηγαίο κώδικα του Cmake, μπορεί να το εγκαταστήσει ακολουθώντας τις οδηγίες που βρίσκονται μέσα σε αυτόν. Συγκεκριμένα, ανοίγοντας ένα τερματικό στο φάκελο του cmake, τα βήματα εγκατάστασης είναι τα εξής:

1. *./bootstrap (*τρέξιμο του εκτελέσιμου αρχείου bootstrap*)*

2. *make*

3. *make install*

Μετά από λίγη ώρα, το Cmake θα είναι εγκατεστημένο στον υπολογιστή του χρήστη στον αντίστοιχο φάκελο με όνομα Cmake-master.

Εφόσον έχει δημιουργηθεί ο φάκελος του Cmake, μπορούν να ακολουθήσουν τα βήματα εγκατάστασης του Orfeo Toolbox που είναι τα εξής:

1. Λήψη της σταθερής έκδοσης του Orfeo Toolbox από το GitHub (https://github.com/orfeotoolbox/OTB). Μετά τη λήψη, ο πηγαίος κώδικας έχει αποθηκευτεί σε ένα φάκελο με όνομα OTB-master.
2. Δημιουργία ενός φακέλου με όνομα OTB. (Η επιλογή των ονομάτων των φακέλων δεν πρέπει υποχρεωτικά να είναι η ίδια. Τα ονόματα που αναφέρονται εδώ χρησιμοποιούνται για περισσότερη ευκολία του αναγνώστη.)
3. Μεταφορά του φακέλου OTB-master μέσα στο φάκελο OTB.
4. Δημιουργία ενός φακέλου με όνομα OTB-binary μέσα στο φάκελο OTB. (Οπότε τώρα υπάρχουν 2 φάκελοι μέσα στο φάκελο OTB. Ο φάκελος OTB-master και ο φάκελος OTB-binary.)
5. Άνοιγμα του φακέλου OTB-binary και άνοιγμα τερματικού μέσα σε αυτόν.
6. Πραγματοποίηση της εντολής ccmake ../OTB-master στο τερματικό(όπου ../OTB-master είναι ο προσδιορισμός της τοποθεσίας του υπολογιστή (path) όπου βρίσκεται ο φάκελος OTB-master). Μετά το πέρας αυτής της εντολής θα πρέπει να ει εμφανιστεί ένα μενού επιλογών στο τερματικό.
7. Πραγματοποίηση της εντολής configure (c) στο μενού επιλογών μέσα στο τερματικό.
8. Επανάληψη της εντολής configure (c) στο μενού επιλογών μέσα στο τερματικό.
9. Πραγματοποίηση της εντολής generate (g) στο μενού επιλογών μέσα στο τερματικό.
10. Πραγματοποίηση της εντολής make στο τερματικό.
11. Πραγματοποίηση της εντολής make install στο τερματικό.

Μετά από τα παραπάνω βήματα, η βιβλιοθήκη του Orfeo Toolbox είναι πλέον εγκατεστημένη στον υπολογιστή του χρήστη.

3.5.3.3 Λειτουργία

Μετά το πέρας της εγκατάστασης, ο χρήστης είναι έτοιμος να χρησιμοποιήσει τις ήδη υπάρχουσες εφαρμογές της βιβλιοθήκης Orfeo Toolbox. Παρακάτω περιγράφεται ένα κλασσικό παράδειγμα helloworld. Δίνονται με τη σειρά τα βήματα που χρειάζεται να πραγματοποιηθούν.

i. Αρχικά, δημιουργείται ένας φάκελος που θα χρησιμοποιηθεί για την αποθήκευση των αρχείων που θα σχηματιστούν. (Τα ονόματα που χρησιμοποιούνται για τους φακέλους και τα αρχεία δεν είναι υποχρεωτικά, αλλά δίνονται για ευκολία του αναγνώστη.)

ii. Μέσα στο φάκελο folder δημιουργείται ένα αρχείο txt με όνομα CmakeLists.txt (η ονομασία του συγκεκριμένου αρχείου είναι η μόνη υποχρεωτική). Το αρχείο αυτό χρησιμοποιείται από το πρόγραμμα Cmake προκειμένου ο κώδικας της επιθυμητής εφαρμογής να γίνει compile και έχει την εξής μορφή:

```
PROJECT(Tutorials)

FIND_PACKAGE(OTB)
IF(OTB_FOUND)
 INCLUDE(${OTB_USE_FILE})
ELSE(OTB_FOUND)
 MESSAGE(FATAL_ERROR
  "Cannot build OTB project without OTB. Please set OTB_DIR.")
ENDIF(OTB_FOUND)

ADD_EXECUTABLE(helloworld helloworld.cxx )
TARGET_LINK_LIBRARIES(helloworld ${OTB_LIBRARIES})
```

Η πρώτη γραμμή αφορά το όνομα της εφαρμογής που δημιουργείται. Στις επόμενες γραμμές, εξασφαλίζεται η εύρεση από το Cmake της τοποθεσίας του υπολογιστή στην οποία βρίσκεται η βιβλιοθήκη του Orfeo Toolbox. Στην προτελευταία γραμμή αναγράφεται ως πρώτο όρισμα το όνομα του εκτελέσιμου αρχείου που θα δημιουργηθεί και ως δεύτερο όρισμα το αρχείο σε κώδικα C++ που πρόκειται να γίνει compile. Τέλος, στην τελευταία γραμμή προσδιορίζονται οι βιβλιοθήκες που θα χρησιμοποιηθούν από την εφαρμογή.

iii. Δημιουργία ενός αρχείου helloworld.cxx που περιέχει τον κώδικα της εφαρμογής σε C++.

iv. Άνοιγμα τερματικού στο φάκελο folder.

v. Πραγματοποίηση της εντολής *cmake ./* στο τερματικό προκειμένου να γίνει το compilation του κώδικα του αρχείου helloworld.cxx.

vi. Πραγματοποίηση της εντολής *make* στο τερματικό για το σχηματισμό του εκτελέσιμου αρχείου helloworld.

vii. Εφόσον όλα τα παραπάνω βήματα έχουν γίνει σωστά, πραγματοποιείται η εντολή ./helloworld στο τερματικό για την υλοποίηση της εφαρμογής.

3.6 Σύνδεση εργαλείων Deep Learning στο Orfeo Toolbox

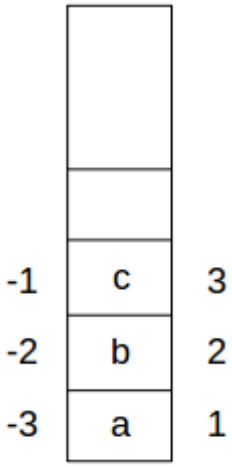
Στην υποενότητα αυτή περιγράφεται ο τρόπος με τον οποίο εφαρμογές σχετικές με την ταξινόμηση Deep Learning μπορούν να υλοποιηθούν στο Orfeo Toolbox. Αρχικά εξηγείται το απαιτούμενο θεωρητικό υπόβαθρο που αφορά κυρίως τη σύνδεση των προγραμματιστικών γλωσσών C++ και Lua. Έπειτα αναλύονται τα αρχεία κώδικα του Orfeo Toolbox που δημιουργήθηκαν τα οποία είναι γραμμένα σε C++. Τέλος, περιγράφονται οι ακριβής οδηγίες χρήσης των αρχείων που έχουν δημιουργηθεί.

3.6.1 Σύνδεση της Lua με τη C++

Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο, στην περίπτωση της χρήσης εμβόλιμων αρχείων, η γλώσσα προγραμματισμού Lua είναι ιδανική. Η σύνδεση των δύο γλωσσών γίνεται μέσω της εικονικής στοίβας (virtual stack) του Lua C API (Application Program Interface).

3.6.1.1 Εικονική Στοίβα (Virtual Stack)

Η σύνδεση της Lua με τη C++ στηρίζεται αποκλειστικά στη χρήση της εικονικής στοίβας. Η στοίβα αυτή ακολουθεί τη λογική LIFO (Last In First Out) και είναι ο μοναδικός τρόπος να μεταφερθεί μία μεταβλητή από τη C++ στη Lua. Για την καλύτερη κατανόηση της λειτουργίας της στοίβας, στην Εικόνα 6.1 δίνεται η γραφική απεικόνιση ενός παραδείγματος, όπου οι μεταβλητές a, b και c μεταφέρονται από τη C++ στη Lua. Η σειρά με την οποία έχουν μεταφερθεί οι μεταβλητές από τη C++ στην εικονική στοίβα είναι η εξής: πρώτη μεταφέρεται η μεταβλητή a, δεύτερη η μεταβλητή b και τελευταία η μεταβλητή c. Εφόσον οι μεταβλητές έχουν τοποθετηθεί στην εικονική στοίβα, το εμβόλιμο αρχείο της Lua δέχεται τις μεταβλητές με την αντίθετη σειρά, δηλαδή πρώτη εισάγεται η μεταβλητή c, δεύτερη η μεταβλητή b και τρίτη η μεταβλητή a. Οι αριθμοί που υπάρχουν δεξιά και αριστερά της στοίβας που απεικονίζεται στην Εικόνα 3.9, αποτελούν δείκτες οι οποίοι χρησιμοποιούνται από το κύριο πρόγραμμα σε C++, προκειμένου αυτό να αποκτήσει πρόσβαση στη μεταβλητή που χρειάζεται κάθε φορά. Μπορούν να χρησιμοποιηθούν είτε οι δείκτες που φαίνονται στην αριστερή πλευρά της στοίβας, είτε αυτοί που φαίνονται στη δεξιά μεριά της στοίβας. Οι πρώτοι ξεκινούν με -1 από την κορυφή της στοίβας και καταλήγουν σε -n στον πάτο της στοίβας, όποι n είναι ο συνολικός αριθμός των μεταβλητών που έχουν μεταφερθεί στη στοίβα. Αντίθετα, οι δεύτεροι ξεκινούν με n από την κορυφή της στοίβας και καταλήγουν σε 1 στον πάτο της στοίβας.



Εικόνα 3.12: Εικονική στοίβα (Virtual Stack)

3.6.1.2 Lua C API – Κώδικας

Στην υποενότητα αυτή περιγράφονται οι βιβλιοθήκες που χρειάζεται να προστεθούν στο κύριο πρόγραμμα σε C++ προκειμένου να γίνει εφικτή η επικοινωνία των δύο γλωσσών. Επίσης, εξηγούνται συγκεκριμένες συναρτήσεις απο το Lua C API που πρόκειται να χρησιμοποιηθούν παρακάτω.

Αρχικά, στο κύριο πρόγραμμα σε C++ χρειάζεται να συμπεριληφθούν τα απαραίτητα header files, ως εξής:

```
#include <lua5.1/lua.hpp>

extern "C" {
        #include "lua.h"
        #include "lualib.h"
        #include "lauxlib.h"
}
```

Το header file lua.hpp περιέχει όλα τα header files της Lua που αντιπροσωπεύουν τη γλώσσα C++.
Το header file lua.h περιλαμβάνει τις βασικές συναρτήσεις που παρέχονται από τη Lua, όπως η συνάρτηση lua_pcall , η οποία χρησιμοποιείται για το διάβασμα και την έξοδο των μεταβλητών. Όλες οι συναρτήσεις που περιέχονται στο συγκεκριμένο header file περιέχουν στην αρχή του ονόματός τους τη λέξη lua. Όσο αφορά το header file lualib.h, περιλαμβάνει συναρτήσεις οι οποίες χρησιμοποιούνται για το άνοιγμα βιβλιοθηκών. Για παράδειγμα, η συνάρτηση luaopen_io κάνει δυνατή τη χρήση των συναρτήσεων I/O όπως οι: io.read, io.write κ.τ.λ. Τέλος, το header file luaxlib.h περιλαμβάνει όλες της συναρτήσεις της βοηθητικής βιβλιοθήκης (auxiliary library), όπως η συνάρτηση luaL_loadfile η οποία φορτώνει ένα αρχείο .lua στο κύριο πρόγραμμα.

Μετά την πρόσθεση των απαραίτητων header files στο κύριο πρόγραμμα, σειρά έχει

η δημιουργία της μεταβλητής L, η οποία είναι υπεύθυνη για τη διαχείριση και τη μεταφορά όλων των μεταβλητών από το κύριο πρόγραμμα σε C++, προς το εμβόλιμο αρχείο της Lua. Η δημιουργία της μεταβλητής αυτής γίνεται ως εξής:

```
lua_State *L;
L = luaL_newstate();
luaL_openlibs(L);
```

Με την εντολή lua_State *L δηλώνεται η μεταβλητή L. Η αμέσως επόμενη εντολή L = luaL_newstate() δημιουργεί ένα καινούριο περιβάλλον Lua. Τέλος, η εντολή luaL_openlibs(L) είναι υπεύθυνη για το άνοιγμα των βιβλιοθηκών που περιλαμβάνονται στο περιβάλλον αυτό.

Οι συναρτήσεις που χρησιμοποιήθηκαν στην εφαρμογή που μελετάται είναι οι εξής:

- void lua_newtable (lua_State *L);
  Η συνάρτηση αυτή δημιουργεί έναν πίνακα τον οποίο προσθέτει στην κορυφή της στοίβας.

- void lua_pushnumber (lua_State *L, lua_Number n);
  Η συγκεκριμένη συνάρτηση μεταφέρει στην κορυφή της στοίβας έναν αριθμό με τιμή n.

- void lua_settable (lua_State *L, int index);
  Η συνάρτηση αυτή εφαρμόζει την εντολή t [k] = v, όπου t είναι η τιμή που βρίσκεται στη θέση με δείκτη index της στοίβας, v είναι η τιμή της μεταβλητής που βρίσκεται στην κορυφή της στοίβας και k είναι η τιμή της μεταβλητής που βρίσκεται ακριβώς κάτω από τη μεταβλητή που βρίσκεται στην κορυφή της στοίβας.

- void lua_pushinteger (lua_State *L, lua_Integer n);
  Η συνάρτηση αυτή μεταφέρει στην κορυφή της στοίβας έναν ακέραιο αριθμό με τιμή n.

- void lua_pushstring (lua_State *L, const char *s);
  Η συνάρτηση αυτή μεταφέρει στην κορυφή της στοίβας την τιμή της μεταβλητής που επισημαίνεται από το δείκτη s.

- void lua_setglobal (lua_State *L, const char *name);
  Αφαιρεί τη μεταβλητή που βρίσκεται στην κορυφή της στοίβας και τη μεταφέρει στο εμβόλιμο αρχείο Lua με το όνομα name.

- int lua_pcall (lua_State *L, int nargs, int nresults, int errfunc);
  Καλεί μια συνάρτηση η οποία έχει αριθμό ορισμάτων ίσο με nargs και αριθμό αποτελεσμάτων ίσο με nresults. Η μεταβλητή errfunc είναι υπεύθυνη για εμφάνιση τυχών λαθών που μπορεί να προκύψουν κατά την πραγματοποίηση της συγκεκριμένης συνάρτησης.

3.6.2 Περιγραφή κώδικα

Παρακάτω εξηγείται ο κώδικας σε C++ που δημιουργήθηκε για τις εφαρμογές που σχετίζονται με την ταξινόμηση Deep Learning στο Orfeo Toolbox. Δημιουργήθηκαν οι εξής 3 εφαρμογές :

1. Κόψιμο μικρών εικόνων patches από μια εικόνα.
2. Εκπαίδευση μοντέλου με ήδη υπάρχονται patches.
3. Έλεγχος εκπαιδευμένου μοντέλου σε μια εικόνα.

Εφαρμογή 1 - Κόψιμο μικρών εικόνων patches από μια εικόνα

Στο αρχείο cut.cpp (Παράρτημα Β, αρχείο κώδικα Β1) πραγματοποιούνται τα εξής:

Στις γραμμές 1-15 γίνεται η δήλωση των απαραίτητων header files. Πιο συγκεκριμένα, το otbImageFileReader.h είναι απαραίτητο για τη χρήση του δείκτη reader ο οποίος είναι υπεύθυνος για την είσοδο της εικόνας. Όσο αφορά το itkImageRegionConstIterator.h, η δήλωσή του είναι απαραίτητη για τη χρήση ενός δείκτη που θα μπορεί να έχει πρόσβαση στις τιμές των pixels της εικόνας. Τέλος, το otbVectorImage.h αντιπροσωπεύει τη διαχείριση πολυκάναλων εικόνων. Στις γραμμές 7-13 γίνεται η δήλωση των header files που χρειάζονται για τη σύνδεση της C++ με τη Lua. Στη γραμμή 15 δηλώνεται η βιβλιοθήκη std της C++.

Στις γραμμές 17-31 δηλώνεται η κύρια συνάρτηση main και δημιουργείται το μήνυμα λάθους που θα εμφανιστεί αν ο χρήστης δώσει λανθασμένο αριθμό μεταβλητών εισόδου στο πρόγραμμα.

Στις γραμμές 34-45 γίνεται η δήλωση των μεταβλητών εισόδου οι οποίες αντιπροσωπεύουν τις τιμές που πρόκειται να δώσει ο χρήστης ως είσοδο στο πρόγραμμα. Οι μεταβλητές αυτές είναι :
1. Το μέγεθος του επιθυμητού patch (patch_size).
2. Το ποσοστό που πρόκειται να χρησιμοποιηθεί από κάθε κλάση για την εκπαίδευση του μοντέλου (perc).
3. Το όνομα του αρχείου στο οποίο θα αποθηκευτούν τα patches εκπαίδευσης (savedp).
4. Το όνομα του αρχείου στο οποίο είναι αποθηκευμένες οι σωστές κλάσεις της εικόνας (labelsfile).
5. Το όνομα του αρχείου στο οποίο θα αποθηκευτούν τα labels που αντιστοιχούν στα κομμένα patches (righttargets).

Στις γραμμές 48-50 γίνεται η δήλωση της μεταβλητής L, η οποία είναι υπεύθυνη για τη διαχείριση των μεταβλητών που πρόκειται να μεταφερθούν στην εικονική στοίβα.

Στις γραμμές 53-88 δημιουργείται ένας μονοδιάστατος πίνακας διαστάσεων (rows*columns*nbBands)x1 ο οποίος περιλαμβάνει τις τιμές των pixels της εικόνας εισόδου. Αναλυτικότερα, στις γραμμές 53-59 δηλώνονται τα στοιχεία της πολυκάναλης εικόνας εισόδου, δηλαδή το είδος των pixels και οι διαστάσεις της. Επίσης δημιουργείται ο pointer reader για την είσοδο της εικόνας η οποία αποτελεί την πρώτη μεταβλητή εισόδου που δίνει ο χρήστης (γραμμή 59). Στη γραμμή 62 ο pointer reader γίνεται update. Στις γραμμές 65-68 εξάγεται το πλήθος των γραμμών

(rows) και των στηλών (columns) της εικόνας, καθώς και ο αριθμός καναλιών που περιέχει (nbBands). Στη γραμμή 72 γίνεται η δήλωση του μονοδιάστατου πίνακα vector διαστάσεων (rows*columns*nbBands)x1, στον οποίο πρόκειται να ανατεθούν οι τιμές των pixels της εικόνας. Η δήλωση του εν λόγω πίνακα γίνεται στη μνήμη heap, αφού λόγω του μεγάλου μεγέθους του δε χωράει στη μνήμη stack. Έπειτα, στις γραμμές 75-78 δημιουργείται ένας επαναλήπτης (iterator) για την πρόσβαση του προγράμματος στις τιμές των pixels της εικόνας. Ακολουθεί η δημιουργία του μονοδιάστατου πίνακα vector στις γραμμές 81-88.

Οι επόμενες γραμμές αφορούν τη μεταφορά των επιθυμητών μεταβλητών στο εμβόλιμο αρχείο Lua. Συγκεκριμένα, στη γραμμή 92 φορτώνεται στο κύριο πρόγραμμα το εμβόλιμο αρχείο Lua με όνομα cut_embed.lua (ΠΑΡΑΡΤΗΜΑ Β, Β4). Το συγκεκριμένο αρχείο τοποθετείται στη θέση 1 της στοίβας. Στη γραμμή 93 δημιουργείται ένας καινούριος πίνακας στη θέση 2 της στοίβας. Στη συνέχεια, στις γραμμές 96-100 μεταφέρεται στη θέση 2 της στοίβας κάθε στοιχείο του πίνακα vector. Μετά το τέλος του συγκεκριμένου for loop, η εικονική στοίβα περιέχει ένα στοιχείο στη θέση 1, το οποίο είναι ο πίνακας vector. Έπειτα, στις γραμμές 103-110 μεταφέρονται στη στοίβα και οι υπόλοιπες απαραίτητες μεταβλητές οι οποίες διαμορφώνονται ως εξής: στη θέση 3 της στοίβας μεταφέρεται η μεταβλητή nbBands, στη θέση 4 μεταφέρεται η μεταβλητή perc, στη θέση 5 η μεταβλητή patch_size, στη θέση 6 η μεταβλητή rows, στη θέση 7 η μεταβλητή columns, στη θέση 8 η μεταβλητή savedp, στη θέση 9 η μεταβλητή labelsfile και στη θέση 10 η μεταβλητή righttargets. Μετά τη μεταφορά των μεταβλητών στη στοίβα, σειρά έχει η μεταφορά τους στο εμβόλιμο αρχείο lua, όπως φαίνεται στις γραμμές 113-121. Η μεταφορά τώρα γίνεται με την αντίθετη σειρά. Δηλαδή, 1η μεταφέρεται η μεταβλητή righttargets με όνομα 'righttargets', 2η η μεταβλητή labelsfile με όνομα 'labelsfile', 3η η μεταβλητή savedp με όνομα 'savedp', 4η η μεταβλητή columns με όνομα 'columns', 5η η μεταβλητή rows με όνομα 'rows', 6η η μεταβλητή patch_size με όνομα 'patch_size', 7η η μεταβλητή perc με όνομα 'perc', 8η η μεταβλητή nbBands με όνομα 'nbBands' και 10ος ο πίνακας vector με όνομα 'mm'.

Τελικά, στις γραμμές 124-125 εκτελείται η συνάρτηση lua_pcall προκειμένου να πραγματοποιηθεί η σύνδεση των αρχείων και να χρησιμοποιηθούν οι μεταβλητές που έχουν μεταφερθεί. Τέλος, στη γραμμή 128 γίνεται διαγραφή του πίνακα vector από τη μνήμη heap.

Εφαρμογή 2 - Εκπαίδευση μοντέλου με ήδη υπάρχονται patches

Στο αρχείο train.cpp (Παράρτημα Β, αρχείο κώδικα Β2) πραγματοποιούνται τα εξής:

Στις γραμμές 1-15 γίνεται η δήλωση των απαραίτητων header files. Πιο συγκεκριμένα, το otbImageFileReader.h είναι απαραίτητο για τη χρήση του δείκτη reader ο οποίος είναι υπεύθυνος για την είσοδο της εικόνας. Όσο αφορά το itkImageRegionConstIterator.h, η δήλωσή του είναι απαραίτητη για τη χρήση ενός δείκτη που θα μπορεί να έχει πρόσβαση στις τιμές των pixels της εικόνας. Τέλος, το otbVectorImage.h αντιπροσωπεύει τη διαχείριση πολυκάναλων εικόνων. Στις γραμμές 7-13 γίνεται η δήλωση των header files που χρειάζονται για τη σύνδεση της C++ με τη Lua. Επίσης, στη γραμμή 15 δηλώνεται η βιβλιοθήκη std της C++.

Στις γραμμές 17-28 δηλώνεται η κύρια συνάρτηση main και δημιουργείται το μήνυμα λάθους που θα εμφανιστεί αν ο χρήστης δώσει λανθασμένο αριθμό μεταβλητών εισόδου στο πρόγραμμα.

Στις γραμμές 31-38 γίνεται η δήλωση των μεταβλητών εισόδου οι οποίες αντιπροσωπεύουν τις τιμές που πρόκειται να δώσει ο χρήστης ως είσοδο στο πρόγραμμα. Οι μεταβλητές αυτές είναι :
1. Το όνομα του αρχείου στο οποίο είναι αποθηκευμένα τα patches που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου (tr_targets).
2. Το όνομα του αρχείου στο οποίο έχουν είναι αποθηκευμένες οι σωστές κλάσεις της εικόνας (targets).
3. Ο αριθμός των κλάσεων που περιέχονται στην εικόνα (ncl).

Στις γραμμές 41-43 γίνεται η δήλωση της μεταβλητής L, η οποία είναι υπεύθυνη για τη διαχείριση των μεταβλητών που πρόκειται να μεταφερθούν στην εικονική στοίβα.

Οι επόμενες γραμμές αφορούν τη μεταφορά των επιθυμητών μεταβλητών στο εμβόλιμο αρχείο Lua. Συγκεκριμένα, στη γραμμή 47 φορτώνεται στο κύριο πρόγραμμα το εμβόλιμο αρχείο Lua με όνομα train_embed.lua (ΠΑΡΑΡΤΗΜΑ Β, Β5). Το συγκεκριμένο αρχείο τοποθετείται στη θέση 1 της στοίβας. Έπειτα, στις γραμμές 50-52 μεταφέρονται στη στοίβα οι απαραίτητες μεταβλητές οι οποίες διαμορφώνονται ως εξής: στη θέση 2 της στοίβας μεταφέρεται η μεταβλητή tr_patches, στη θέση 3 η μεταβλητή targets και στη θέση 4 η μεταβλητή ncl. Μετά τη μεταφορά των μεταβλητών στη στοίβα, σειρά έχει η μεταφορά τους στο εμβόλιμο αρχείο lua, όπως φαίνεται στις γραμμές 55-57. Η μεταφορά τώρα γίνεται με την αντίθετη σειρά. Δηλαδή, 1η μεταφέρεται η μεταβλητή ncl με όνομα 'ncl', 2η η μεταβλητή targets με όνομα 'targets' και 3η η μεταβλητή tr_patches με όνομα 'tr_patches'.

Τέλος, στις γραμμές 60-61 εκτελείται η συνάρτηση lua_pcall προκειμένου να πραγματοποιηθεί η σύνδεση των αρχείων και να χρησιμοποιηθούν οι μεταβλητές που έχουν μεταφερθεί.

Εφαρμογή 3 - Έλεγχος εκπαιδευμένου μοντέλου σε μια εικόνα

Στο αρχείο testing.cpp (Παράρτημα Β, αρχείο κώδικα Β3) πραγματοποιούνται τα εξής:

Στις γραμμές 1-15 γίνεται η δήλωση των απαραίτητων header files. Πιο συγκεκριμένα, το otbImageFileReader.h είναι απαραίτητο για τη χρήση του δείκτη reader ο οποίος είναι υπεύθυνος για την είσοδο της εικόνας. Όσο αφορά το itkImageRegionConstIterator.h, η δήλωσή του είναι απαραίτητη για τη χρήση ενός δείκτη που θα μπορεί να έχει πρόσβαση στις τιμές των pixels της εικόνας. Τέλος, το otbVectorImage.h αντιπροσωπεύει τη διαχείριση πολυκάναλων εικόνων. Στις γραμμές 7-13 γίνεται η δήλωση των header files που χρειάζονται για τη σύνδεση της C++ με τη Lua. Στη γραμμή 15 δηλώνεται η βιβλιοθήκη std της C++.

Στις γραμμές 17-32 δηλώνεται η κύρια συνάρτηση main και δημιουργείται το μήνυμα λάθους που θα εμφανιστεί αν ο χρήστης δώσει λανθασμένο αριθμό μεταβλητών

εισόδου στο πρόγραμμα.

Στις γραμμές 35-48 γίνεται η δήλωση των μεταβλητών εισόδου οι οποίες αντιπροσωπεύουν τις τιμές που πρόκειται να δώσει ο χρήστης ως είσοδο στο πρόγραμμα. Οι μεταβλητές αυτές είναι :
1. Το όνομα της εικόνας στην οποία θα εφαρμοστεί το testing.
2. Το μέγεθος του επιθυμητού patch (patch_size).
3. Ο συνολικός αριθμός των κλάσεων πάνω στις οποίες έχει εκπαιδευτεί το μοντέλο (tncl).
4. Το όνομα του αρχείου στο οποίο περιέχονται οι σωστές κλάσεις της εικόνας (labelsfile).
5. Το όνομα του αρχείου στο οποίο είναι αποθηκευμένο το μοντέλο (modelname).
6. Το όνομα του αρχείου στο οποίο είναι οι μέσοι όροι των δεδομένων εκπαίδευσης (meanname).
7. Το όνομα του αρχείου στο οποίο είναι αποθηκευμένες οι τυπικές αποκλίσεις των δεδομένων εκπαίδευσης (stdvname).

Στις γραμμές 51-54 γίνεται η δήλωση της μεταβλητής L, η οποία είναι υπεύθυνη για τη διαχείριση των μεταβλητών που πρόκειται να μεταφερθούν στην εικονική στοίβα.

Στις γραμμές 57-92 δημιουργείται ένας μονοδιάστατος πίνακας διαστάσεων (rows*columns*nbBands)x1 ο οποίος περιλαμβάνει τις τιμές των pixels της εικόνας εισόδου. Αναλυτικότερα, στις γραμμές 57-63 δηλώνονται τα στοιχεία της πολυκάναλης εικόνας εισόδου, δηλαδή το είδος των pixels και οι διαστάσεις της. Επίσης δημιουργείται ο pointer reader για την είσοδο της εικόνας η οποία αποτελεί την πρώτη μεταβλητή εισόδου που δίνει ο χρήστης (γραμμή 63). Στη γραμμή 66 ο pointer reader γίνεται update. Στις γραμμές 69-72 εξάγεται το πλήθος των γραμμών (rows) και των στηλών (columns) της εικόνας, καθώς και ο αριθμός καναλιών που περιέχει (nbBands). Στη γραμμή 76 γίνεται η δήλωση του μονοδιάστατου πίνακα vector διαστάσεων (rows*columns*nbBands)x1, στον οποίο πρόκειται να ανατεθούν οι τιμές των pixels της εικόνας. Η δήλωση του εν λόγω πίνακα γίνεται στη μνήμη heap, αφού λόγω του μεγάλου μεγέθους του δε χωράει στη μνήμη stack. Έπειτα, στις γραμμές 79-82 δημιουργείται ένας επαναλήπτης (iterator) για την πρόσβαση του προγράμματος στις τιμές των pixels της εικόνας. Ακολουθεί η δημιουργία του μονοδιάστατου πίνακα vector στις γραμμές 85-92.

Οι επόμενες γραμμές αφορούν τη μεταφορά των επιθυμητών μεταβλητών στο εμβόλιμο αρχείο Lua. Συγκεκριμένα, στη γραμμή 95 φορτώνεται στο κύριο πρόγραμμα το εμβόλιμο αρχείο Lua με όνομα testing_embed.lua (ΠΑΡΑΡΤΗΜΑ Β, Β6). Το συγκεκριμένο αρχείο τοποθετείται στη θέση 1 της στοίβας. Στη γραμμή 98 δημιουργείται ένας καινούριος πίνακας στη θέση 2 της στοίβας. Στη συνέχεια, στις γραμμές 101-105 μεταφέρεται στη θέση 2 της στοίβας κάθε στοιχείο του πίνακα vector. Έπειτα, στις γραμμές 108-116 μεταφέρονται στη στοίβα και οι υπόλοιπες απαραίτητες μεταβλητές οι οποίες διαμορφώνονται ως εξής: στη θέση 3 της στοίβας μεταφέρεται η μεταβλητή rows, στη θέση 4 η μεταβλητή columns, στη θέση 5 η μεταβλητή nbBands, στη θέση 6 η μεταβλητή patch_size, στη θέση 7 η μεταβλητή labelsfile, στη θέση 8 η μεταβλητή modelname, στη θέση 9 η μεταβλητή meanname, στη θέση 10 η μεταβλητή stdvname και στη θέση 11 η μεταβλητή tncl. Μετά τη

μεταφορά των μεταβλητών στη στοίβα, σειρά έχει η μεταφορά τους στο εμβόλιμο αρχείο lua, όπως φαίνεται στις γραμμές 119-128. Η μεταφορά τώρα γίνεται με την αντίθετη σειρά. Δηλαδή, 1η μεταφέρεται η μεταβλητή tncl με όνομα 'tncl', 2η η μεταβλητή stdvname με όνομα 'stdvname', 3η η μεταβλητή meanname με όνομα 'meanname', 4η η μεταβλητή modelname με όνομα 'modelname, 5η η μεταβλητή labelsfile με όνομα 'labelsfile', 6η η μεταβλητή patch_size με όνομα 'patch_size', 7η η μεταβλητή nbBands με όνομα 'nbBands', 8η η μεταβλητή columns με όνομα 'columns', 9η η μεταβλητή rows με όνομα 'rows' και 10ος ο πίνακας vector με όνομα 'mm'.

Τελικά, στις γραμμές 131-132 εκτελείται η συνάρτηση lua_pcall προκειμένου να πραγματοποιηθεί η σύνδεση των αρχείων και να χρησιμοποιηθούν οι μεταβλητές που έχουν μεταφερθεί. Τέλος, στη γραμμή 135 γίνεται διαγραφή του πίνακα vector από τη μνήμη heap.

3.6.3 Εκτέλεση εφαρμογών ταξινόμησης με βάση αρχιτεκτονικές Deep Learning

Στην υποενότητα αυτή περιγράφονται αναλυτικά τα βήματα που πρέπει να πραγματοποιηθούν από το χρήστη για την εκτέλεση των τριών εφαρμογών. Υπενθυμίζεται ότι πριν από την εκτέλεση των βημάτων είναι απαραίτητο να έχουν εγκατασταθεί τα λογισμικά Torch και Orfeo Toolbox στο υπολογιστικό σύστημα. Οι οδηγίες εγκατάστασής τους έχουν δωθεί αναλυτικά σε προηγούμενο κεφάλαιο.

Για την καλύτερη περιγραφή των οδηγιών θα χρησιμοποιηθούν συγκεκριμένα ονόματα αρχείων. Κάθε χρήστης μπορεί να χρησιμοποιεί τα δικά του ονόματα. Έστω λοιπόν ότι έχουμε 2 εικόνες (zh1.tif και zh4.tif) και τα αντίστοιχα αρχεία με τις κλάσεις τους (labelsimg1.mat και img4_labels.mat).

Εφαρμογή 1 - Κόψιμο μικρών εικόνων patches από μια εικόνα

Σε περίπτωση που ο χρήστης επιθυμεί να κόψει patches από την zh1.tif, τα βήματα που πρέπει να ακολουθήσει είναι τα εξής:

1. Δημιουργία ενός δισδιάστατου πίνακα 'labelsimg1.mat', ο οποίος περιέχει τις κλάσεις στις οποίες αντιστοιχούν τα pixels της εικόνας που χρησιμοποιείται. Δηλαδή αν η εικόνα έχει διαστάσεις (rows)x(columns)x(nbBands), τότε ο πίνακας θα έχει διαστάσεις (rows)x(columns). Σημειώνεται εδώ, ότι στην περίπτωση που υπάρχουν αταξινόμητες περιοχές στην εικόνα, η κλάση στην οποία ανήκουν θα πρέπει να έχει τιμή 0. Επίσης, το εσωτερικό όνομα του αρχείου θα πρέπει οπωσδήποτε να είναι 'labels'. Διαφορετικά, χρειάζεται να γίνει αλλαγή στη γραμμή 12 του εμβόλιμου αρχείου cut_embed.lua. Αν για παράδειγμα το εσωτερικό όνομα του αρχείου .mat είναι **onoma**, τότε η εντολή της γραμμής 12 θα πρέπει να μετατραπεί ως εξής:
labels=a1.**labels** → labels=a1.**onoma**.

2. Δημιουργία ενός φακέλου με όνομα CUT (προαιρετικό όνομα) και μεταφορά των αρχείων cut.cpp, cut_embed.lua, zh1.tif και labelsimg1.mat μέσα σε αυτόν.

3. Δημιουργία του αρχείου CmakeLists.txt μέσα στο φάκελο CUT, για τη σύνδεση του CMake με τη βιβλιοθήκη του Orfeo Toolbox και τη Lua. Στην Εικόνα 3.10 φαίνεται το περιεχόμενο του συγκεκριμένου αρχείου. Στη γραμμή 13 γίνεται η

σύνδεση του Cmake με το path στο οποίο βρίσκεται εγκατεστημένη η Lua. Στο συγκεκριμένο παράδειγμα χρησιμοποιείται η έκδοση 5.1 της Lua.

```
PROJECT(Tutorials)

cmake_minimum_required(VERSION 2.6)

FIND_PACKAGE(OTB)
IF(OTB_FOUND)
 INCLUDE(${OTB_USE_FILE} )
ELSE(OTB_FOUND)
 MESSAGE(FATAL_ERROR
  "Cannot build OTB project without OTB. Please set OTB_DIR.")
ENDIF(OTB_FOUND)

link_directories(/usr/include/lua5.1)


ADD_EXECUTABLE(cut cut.cpp )
TARGET_LINK_LIBRARIES(cut ${OTB_LIBRARIES} lua5.1)
```

Εικόνα 3.13: Αρχείο CmakeLists.txt

4. Άνοιγμα τερματικού στο φάκελο και εκτέλεση της εντολής *cmake ./* για τη σύνδεση των βιβλιοθηκών.

5. Εκτέλεση της εντολής *make* προκειμένου ο κώδικας να γίνει compilation και να δημιμουργηθεί το εκτελέσιμο αρχείο cut.

6. Εφόσον τα παραπάνω βήματα έχουν πραγματοποιηθεί με επιτυχία, ο χρήστης μπορεί να τρέξει την εφαρμογή δίνοντας ως είσοδο στο πρόγραμμα τις παρακάτω 6 μεταβλητές:

i. Το όνομα της εικόνας.
ii.Τη διάσταση του επιθυμητού patch.
iii.Το ποσοστό που επιθυμείται να εξαχθεί από κάθε κλάση για τη δημιουργία των δεδομένων εκπαίδευσης.
iv.Το όνομα του αρχείου στο οποίο πρόκειται να αποθηκευτούν τα patches που θα δημιουργηθούν.
v.Το αρχείο στο οποίο είναι αποθηκευμένες οι σωστές κλάσεις της εικόνας.
vi. Το όνομα του αρχείου στο οποίο πρόκειται να αποθηκευτούν οι κλάσεις που αντιστοιχούν στα patches που θα δημιουργηθούν.

Στην εικόνα 3.11 φαίνεται η εντολή που πρέπει να δωθεί στο τερματικό ώστε να εκτελεστεί η Εφαρμογή 1 με τα δεδομένα που περιγράφηκαν. Στη συγκεκριμένη περίπτωση, το όνομα της εικόνας είναι 'zh1.tif'. Ο αριθμός 5 αποτελεί το μέγεθος του επιθυμητού patch, το οποίο στη συγκεκριμένη περίπτωση θα είναι 5x5. Ο αριθμός 0.1

αντιπροσωπεύει το ποσοστό των pixels που πρόκειται να εξαχθεί τυχαία από κάθε κλάση για το σχηματισμό των patches. Πρόκειται δηλαδή να εξαχθεί το 10% από κάθε κλάση. Σημειώνεται ότι για τις αταξινόμητες περιοχές με τιμή 0 δεν εξάγονται καθόλου patches. Το όνομα 'patches_img1.mat' αντιπροσωπεύει το όνομα του αρχείου στο οποίο θα αποθηκευτούν τα patches. Το όνομα 'labels_img1.mat' αντιπροσωπεύει το αρχείο που περιέχει τις σωστές κλάσεις της εικόνας και έχει δημιουργηθεί από το χρήστη. Τέλος, το όνομα 'targets_img1.mat αντιπροσωπεύει το αρχείο που πρόκειται να δημιουργηθεί από την εφαρμογή και περιέχει τις κλάσεις που αντιστοιχούν στα κομμένα patches.

```
./cut zh1.tif 5 0.1 patches_img1.mat labelsimg1.mat targets_img1.mat
```

Εικόνα 3.14: Εντολή που πρέπει να δωθεί στο τερματικό για την εκτέλεση της Εφαρμογής 1.

Εφόσον η εντολή της Εικόνας 6.4 πραγματοποιηθεί με επιτυχία, θα έχουν σχηματιστεί στο φάκελο τα αρχεία patches_img1.mat και targets_img1.mat που αντιστοιχούν στα κομμένα patches και τις αντίστοιχες κλάσεις τους.

Σε περίπτωση που ο χρήστης δώσει αριθμό μεταβλητών εισόδου διαφορετικό από 7, θα εμφανιστεί στο τερματικό αντίστοιχο μήνυμα λάθους. Πρέπει εδώ να σημειωθεί ότι η εφαρμογή μπορεί να πραγματοποιηθεί με οποιοδήποτε είδος εικόνας, αρκεί ο χρήστης να αλλάξει τη δήλωση του είδους των pixels στον πηγαίο κώδικα cut.cpp (γραμμή 53).

Εφαρμογή 2 - Εκπαίδευση μοντέλου με ήδη υπάρχονται patches

Σε περίπτωση που ο χρήστης επιθυμεί να εκπαιδεύσει ένα μοντέλο με ένα ήδη δημιουργημένο σύνολο από patches, τα βήματα που θα πρέπει να ακολουθήσει είναι τα εξής (για την περιγραφή χρησιμοποιούνται τα αρχεία που δημιουργήθηκαν στην Εφαρμογή 1):

1. Δημιουργία ενός φακέλου με όνομα TRAIN (προαιρετικό όνομα) και μεταφορά των αρχείων train.cpp, train_embed.lua, targets_img1.mat και patches_img1.mat μέσα σε αυτόν. Μέσα στον ίδιο φάκελο, θα πρέπει επίσης να τοποθετηθεί και ο φάκελος 'models' ο οποίος περιέχει μοντέλα ConvNets για διάφορες διαστάσεις patches (ΠΑΡΑΡΤΗΜΑ Β, Β7).

2. Δημιουργία του αρχείου CmakeLists.txt για τη σύνδεση του CMake με τη βιβλιοθήκη του Orfeo Toolbox και τη Lua.

3. Άνοιγμα τερματικού στο φάκελο και εκτέλεση της εντολής *cmake ./* για τη σύνδεση των βιβλιοθηκών.

4. Εκτέλεση της εντολής *make* προκειμένου ο κώδικας να γίνει compilation και να δημιμουργηθεί το εκτελέσιμο αρχείο train.

5. Εφόσον τα παραπάνω βήματα έχουν πραγματοποιηθεί με επιτυχία, ο χρήστης μπορεί να τρέξει την εφαρμογή δίνοντας ως είσοδο στο πρόγραμμα τις παρακάτω 3

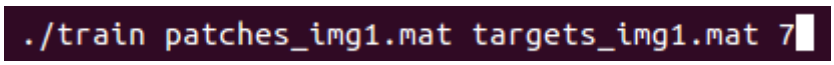
μεταβλητές:

i. Το αρχείο patches_img1.mat το οποίο περιέχει τα κομμένα patches. Το εσωτερικό όνομα του αρχείου θα πρέπει οπωσδήποτε να είναι 'x'. Διαφορετικά, χρειάζεται να γίνει αλλαγή στη γραμμή 23 του εμβόλιμου αρχείου train_embed.lua. Αν για παράδειγμα το εσωτερικό όνομα του αρχείου .mat είναι **onoma**, τότε η εντολή της γραμμής 23 θα πρέπει να μετατραπεί ως εξής:
tr_patches=tr_patches1.**x** → tr_patches=tr_patches1.**onoma**.
ii.Το αρχείο targets_img1.mat το οποίο περιέχει τις κλάσεις που αντιστοιχούν στα κομμένα patches. Το εσωτερικό όνομα του αρχείου θα πρέπει οπωσδήποτε να είναι 'x'. Διαφορετικά, χρειάζεται να γίνει αλλαγή στη γραμμή 24 του εμβόλιμου αρχείου train_embed.lua. Αν για παράδειγμα το εσωτερικό όνομα του αρχείου .mat είναι **onoma**, τότε η εντολή της γραμμής 24 θα πρέπει να μετατραπεί ως εξής:
tr_labels=tr_labels1.**x** → tr_labels=tr_labels1.**onoma**.
iii.Το συνολικό αριθμό των κλάσεων που περιέχονται στα patches.

Σημειώνεται ότι η διαδικασία training μπορεί να πραγματοποιηθεί για patches μεγέθους 5x5, 11x11, 21x21, 29x29 και 33x33. Οποιοδήποτε άλλο μέγεθος δεν υποστηρίζεται από την εφαρμογή.

Σε περίπτωση που ο χρήστης δώσει αριθμό μεταβλητών εισόδου διαφορετικό από 4, θα εμφανιστεί στο τερματικό αντίστοιχο μήνυμα λάθους.

Στην Εικόνα 3.12 φαίνεται η εντολή που πρέπει να δωθεί στο τερματικό ώστε να εκτελεστεί η Εφαρμογή 2 για τα δεδομένα που περιγράφηκαν. Στη συγκεκριμένη περίπτωση, 'patches)img1.mat' είναι το όνομα του αρχείου που περιέχει τα patches, ενώ 'targets_img1.mat' είναι το όνομα του αρχείου που περιέχει τις αντίστοιχες κλάσεις. Όσο αφορά τον αριθμό 7, είναι το σύνολο των κλάσεων.



Εικόνα 3.15: Εντολή που πρέπει να δωθεί στο τερματικό για την εκτέλεση της Εφαρμογής 2.

Εφόσον η εντολή της εικόνας 6.4 έχει πραγματοποιηθεί με επιτυχία, μέσα στο φάκελο TRAIN θα έχουν σχηματιστεί τα αρχεία trained_model.net, mean.t7 και stdv.t7 τα οποία αντιστοιχούν στο εκπαιδευμένο μοντέλο, το μέσο όρο (mean) των patches εκπαίδευσης και την τυπική απόκλιση (stdv) των δεδομένων εκπαίδευσης αντίστοιχα.

Εφαρμογή 3 - Έλεγχος εκπαιδευμένου μοντέλου σε μια εικόνα

Σε περίπτωση που ο χρήστης επιθυμεί να εφαρμόσει testing σε μια εικόνα με ένα ήδη εκπαιδευμένο μοντέλο, τα βήματα που πρέπει να ακολουθήσει είναι τα εξής (για την περιγραφή χρησιμοποιούνται τα αρχεία που δημιουργήθηκαν στην Εφαρμογή 2):

1. Δημιουργία ενός δισδιάστατου πίνακα img4_labels.mat, ο οποίος περιέχει τις κλάσεις στις οποίες αντιστοιχούν τα pixels της εικόνας zh4.tif. Δηλαδή αν η εικόνα έχει διαστάσεις (rows)x(columns)x(nbBands), τότε ο πίνακας θα έχει διαστάσεις (rows)x(columns). Σημειώνεται εδώ, ότι στην περίπτωση που υπάρχουν αταξινόμητες περιοχές στην εικόνα, η κλάση στην οποία ανήκουν θα πρέπει να έχει τιμή 0. Επίσης,

το εσωτερικό όνομα του αρχείου θα πρέπει οπωσδήποτε να είναι 'labels'. Διαφορετικά, χρειάζεται να γίνει αλλαγή στη γραμμή 34 του εμβόλιμου αρχείου testing_embed.lua. Αν για παράδειγμα το εσωτερικό όνομα του αρχείου .mat είναι **onoma**, τότε η εντολή της γραμμής 12 θα πρέπει να μετατραπεί ως εξής:
labels=a1.**labels** → labels=a1.**onoma**.

2. Δημιουργία ενός φακέλου με όνομα TESTING (προαιρετικό όνομα) και μεταφορά των αρχείων testing.cpp, testing_embed.lua, zh4.tif, img4_labels.mat, trained_model.net, mean.t7 και stdv.t7 μέσα σε αυτόν.

3. Δημιουργία του αρχείου CmakeLists.txt για τη σύνδεση του CMake με τη βιβλιοθήκη του Orfeo Toolbox και τη Lua.

4. Άνοιγμα τερματικού στο φάκελο και εκτέλεση της εντολής *cmake ./* για τη σύνδεση των βιβλιοθηκών.

5. Εκτέλεση της εντολής *make* προκειμένου ο κώδικας να γίνει compilation και να δημιμουργηθεί το εκτελέσιμο αρχείο testing.

6. Εφόσον τα παραπάνω βήματα έχουν πραγματοποιηθεί με επιτυχία, ο χρήστης μπορεί να τρέξει την εφαρμογή δίνοντας ως είσοδο στο πρόγραμμα τις παρακάτω 6 μεταβλητές:

i. Το όνομα της εικόνας για την οποία θα πραγματοποιηθεί ο έλεγχος του εκπαιδευμένου μοντέλου.
ii.Το μέγεθος του επιθυμητού patch.
iii. Τον αριθμό των συνολικών κλάσεων σύμφωνα με τον οποίο έχει εκπαιδευτεί το μοντέλο.
iii.Το όνομα του αρχείο στο οποίο είναι αποθηκευμένες οι σωστές κλάσεις της εικόνας ελέγχου.
iv. Το όνομα του αρχείου στο οποίο είναι αποθηκευμένο το εκπαιδευμένο μοντέλο.
v. Το όνομα του αρχείου στο οποίο είναι αποθηκευμένοι οι μέσοι όροι των δεδομένων εκπαίδευσης.
vi. Το όνομα του αρχείου στο οποίο είναι αποθηκευμένες οι τυπικές αποκλίσεις των δεδομένων εκπαίδευσης.

Σε περίπτωση που ο χρήστης δώσει αριθμό μεταβλητών εισόδου διαφορετικό από 7 θα εμφανιστεί στο τερματικό αντίστοιχο μήνυμα λάθους. Πρέπει εδώ να σημειωθεί ότι η εφαρμογή μπορεί να πραγματοποιηθεί με οποιοδήποτε είδος εικόνας, αρκεί ο χρήστης να αλλάξει τη δήλωση του είδους των pixels στον πηγαίο κώδικα train.cpp (γραμμή 57).

Στην εικόνα 3.13 φαίνεται η εντολή που πρέπει να δωθεί στο τερματικό ώστε να εκτελεστεί η Εφαρμογή 3 για τα δεδομένα που περιγράφηκαν. Στη συγκεκριμένη περίπτωση, το όνομα της εικόνας ελέγχου είναι 'zh4.tif'. Όσο αφορά τον αριθμό των κλάσεων, περιλαμβάνει το συνολικό πλήθος των κλάσεων σύμφωνα με το οποίο έχει εκπαιδευτεί το μοντέλο. Εδώ ο αριθμός αυτός είναι ίσος με 7. Αυτό σημαίνει ότι το μοντέλο trained_model.net μπορεί να προβλέψει τις κλάσεις 1,2,3,4,5,6,και 7.

Συνεπώς η εικόνα ελέγχου zh4.tif μπορεί να περιλαμβάνει αριθμό κλάσεων ίσο ή μικρότερο από το 7. Στη συνέχεια, το όνομα 'img4_labels.mat' αντιπροσωπεύει το όνομα του αρχείου στο οποίο είναι αποθηκευμένες οι σωστές κλάσεις της εικόνας ελέγχου zh4.tif. Το όνομα 'trained_model.net' αντιπροσωπεύει το αρχείο στο οποίο είναι αποθηκευμένο το εκπαιδευμένο μοντέλο. Τέλος, τα ονόματα 'mean.t7' και 'stdv.t7' αντιπροσωπεύουν τα αρχεία στα οποία είναι αποθηκευμένοι οι μέσοι όροι και οι τυπικές αποκλίσεις αντίστοιχα, των δεδομένων εκπαίδευσης.

```
./testing zh4.tif 5 7 img4_labels.mat trained_model.net mean.t7 stdv.t7
```

Εικόνα 3.16: Εντολή που πρέπει να δωθεί στο τερματικό για την εκτέλεση της Εφαρμογής 3.

4. Αποτελέσματα και αξιολόγηση

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα των ταξινομήσεων και της ποσοτικής αξιολόγησης που περιγράφηκαν στο προηγούμενο κεφάλαιο.

4.1 Δεδομένα Deepsat

Η αξιολόγηση της ταξινόμησης για τη συγκεκριμένη ομάδα δεδομένων έγινε με απευθείας παρατήρηση των ακριβειών PA και UA, αφού τα ποσοστά ήταν ιδιαίτερα υψηλά, πράγμα που σημαίνει ότι οι κλάσεις ελάχιστα μπερδεύτηκαν μεταξύ τους. Τα αποτελέσματα που παρουσιάζονται παρακάτω προέκυψαν στα πλαίσια της δημοσίευσης 'M. Papadomanolaki, M. Vakalopoulou, S. Zagoruyko, K. Karantzalos – Benchmarking Deep Learning Frameworks for the Classification of Very High Resolution Satellite Multispectral Data', για το συνέδριο ISPRS 2016 στην Πράγα.

4.1.1 Ποσοτική αξιολόγηση για τα δεδομένα SAT-4

Τα αποτελέσματα των μεθόδων ταξινόμησης για την ομάδα δεδομένων SAT-4 δίνονται στον Πίνακα 4.1, όπου accuracy είναι η ακρίβεια του χρήστη (User's accuracy) και precision είναι η ακρίβεια του παραγωγού (Producer's accuracy). Σημειώνεται εδώ ότι στον Πίνακα 4.1, οι ονομασίες AlexNet Pretrained και AlexNet-small αντιστοιχούν στη μέθοδο SVM και στο μοντέλο ConvNet αντίστοιχα. Σύμφωνα με τον πίνακα, όλες οι μέθοδοι ταξινόμησης που εφαρμόστηκαν πέτυχαν ιδιαίτερα υψηλές ακρίβειες. Πιο συγκεκριμένα, το μοντέλο AlexNet Pretrained σημείωσε ποσοστά 99.46% και 98.75% για τις ακρίβειες accuracy και precision αντίστοιχα. Η ακρίβεια accuracy του μοντέλου AlexNet ανήλθε σε ποσοστό 99.98% και η ακρίβεια precision σε 99.94%. Υψηλά ποσοστά πέτυχε και το μοντέλο AlexNet-small με ακρίβεια accuracy ίση με 99.86% και ακρίβεια precision ίση με 99.75%. Τέλος, το μοντέλο VGG είχε ακρίβεια accuracy 99.98% και ακρίβεια precision 99.95%.

Σύμφωνα με τα παραπάνω, τα συνολικά ποσοστά της ακρίβειας accuracy σε όλες τις περιπτώσεις ήταν πάνω από 99.4% και της συνολικής ακρίβειας precision πάνω από 98.7%. Όπως ήταν αναμενόμενο, η χρήση του προεκπαιδευμένου μοντέλου AlexNet για την εκπαίδευση του αλγορίθμου SVM έχει τις χαμηλότερες ακρίβειες, αφού στην περίπτωση αυτή οι υπολογιστικές διαδικασίες είναι ιδιαίτερα πολύπλοκες σε σχέση με τα υπόλοιπα μοντέλα. Όσο αφορά τα άλλα τρία μοντέλα, τα ποσοστά της ακρίβειεας accuracy και precision ξεπέρασαν σε κάθε περίπτωση το ποσοστό 99.80% και 99.7% αντίστοιχα. Συγκεκριμένα, το μοντέλο AlexNet πέτυχε λίγο μεγαλύτερες ακρίβειες από το μοντέλο ConvNet, πράγμα που σημαίνει ότι τα επίπεδα ReLU αντεπεξέρχονται καλύτερα στην ταξινόμηση των συγκεκριμένων δεδομένων. Πιο αναλυτικά, η κλάση με τις χαμηλότερες ακρίβειες ήταν το γρασίδι, καθώς κάποια patches ταξινομήθηκαν εσφαλμένα στην κατηγορία γυμνό έδαφος ή δέντρο. Τα μοντέλα AlexNet και VGG είχαν τις μεγαλύτερες ακρίβειες.

	AlexNet Pretrained		*AlexNet*		*AlexNet-small*		*VGG*	
LC Class	Accuracy	Precision	Accuracy	Precision	Accuracy	Precision	Accuracy	Precision
Barren Land	99.24%	99.02%	99.96%	99.88%	99.83%	99.54%	99.96%	99.85%
Trees	99.73%	99.51%	99.99%	99.98%	99.95%	99.90%	99.99%	99.97%
Grassland	99.13%	96.76%	99.96%	99.90%	99.81%	99.59%	99.95%	99.99%
Other	99.77%	99.73%	99.98%	99.98%	99.96%	99.95%	99.99%	99.99%
Overall	99.46%	98.75%	**99.98%**	99.94%	99.86%	99.75%	**99.98%**	**99.95%**

Πίνακας 4.1: Ακρίβειες όλων των μοντέλων ταξινόμησης για την ομάδα δεδομένων SAT-4.

4.1.2 Ποσοτική αξιολόγηση για τα δεδομένα SAT-6

Όσο αφορά την ταξινόμηση του συνόλου δεδομένων SAT-6, οι ακρίβειες που προέκυψαν φαίνονται στον Πίνακα 2. Όπως και στην ομάδα δεδομένων SAT-4, το προεκπαιδευμένο μοντέλο AlexNet είχε τις χαμηλότερες ακρίβειες σε σχέση με τα υπόλοιπα μοντέλα. Από τα τρία μοντέλα με τις μεγαλύτερες ακρίβειες, το μοντέλο VGG είχε λίγο υψηλότερες ακρίβειες σε σχέση με τα μοντέλα AlexNet και ConvNet. Σε όλες τις περιπτώσεις των τριών αυτών μοντέλων, τα συνολικά επίπεδα της ακρίβειας accuracy και precision ήταν πάνω από 99.9% και 99.5% αντίστοιχα. Επίσης, από τον πίνακα μπορεί κανείς να παρατηρήσει ότι η κλάση νερό ήταν η ευκολότερα αναγνωρίσιμη κλάση αφού έχει τα μεγαλύτερα ποσοστά της υπολογιζόμενης ακρίβειας accuracy. Αντίθετα, η κλάση δρόμος έδωσε τα χαμηλότερα ποσοστά της ακρίβειας accuracy αφού έγιναν πολλές λανθασμένες ταξινομήσεις στις κλάσεις δέντρο και γρασίδι.

	AlexNet Pretrained		*AlexNet*		*AlexNet-small*		*VGG*	
LC Class	Accuracy	Precision	Accuracy	Precision	Accuracy	Precision	Accuracy	Precision
Barren Land	99.91%	98.72%	99.95%	99.41%	99.96%	99.70%	99.99%	100%
Trees	99.14%	98.78%	99.86%	99.58%	99.77%	99.33%	99.96%	99.87%
Grassland	99.04%	99.72%	99.97%	99.92%	99.95%	99.84%	99.99%	99.96%
Roads	99.16%	96.62%	99.84%	99.61%	99.74%	99.43%	99.89%	99.95%
Buildings	99.92%	98.93%	99.95%	99.08%	99.96%	99.08%	99.99%	99.61%
Water Bodies	100%	100%	100%	100%	99.99%	100%	100%	100%
Overall	99.57%	98.80%	99.93%	99.60%	99.90%	99.56%	**99.98%**	**99.91%**

Πίνακας 4.2: Ακρίβειες όλων των μοντέλων ταξινόμησης για την ομάδα δεδομένων SAT-6.

4.2 Ποσοτική αξιολόγηση για τα δεδομένα δεδομένα 'Zurich Summer Dataset v1.0'

Στην ενότητα αυτή περιγράφονται και αναλύονται τα αποτελέσματα της ταξινόμησης για την ομάδα δεδομένων Zurich Summer Dataset v1.0. Ο έλεγχος των μοντέλων έγινε στις 2 εικόνες που δε χρησιμοποιήθηκαν για την εκπαίδευση του μοντέλου. Στις Εικόνες 4.1 και 4.2 φαίνονται η 1η και η 2η εικόνα ελέγχου. Η 1η εικόνα ελέγχου δεν περιλαμβάνει τις κλάσεις γυμνό έδαφος(Barren Land) και πισίνες(Swimming Pools), ενώ η 2η εικόνα ελέγχου δεν περιλαμβάνει τις κλάσεις νερό(Water) και σιδηρόδρομοι(Railways).

Εικόνα 4.1: 1η εικόνα ελέγχου

Εικόνα 4.2: 2η εικόνα ελέγχου

4.2.1 Εύρεση βέλτιστων διαστάσεων των patches

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, για αυτή την ομάδα δεδομένων κόπηκαν patches διαφόρων διαστάσεων. Το μοντέλο που χρησιμοποιήθηκε για την εύρεση των βέλτιστων διαστάσεων είναι το ConvNet. Παρακάτω περιγράφονται τα αποτελέσματα αναλυτικά για όλες τις διαστάσεις.

4.2.1.1 Patches διαστάσεων 5x5

- **Περιγραφή παραμέτρων του μοντέλου εκπαίδευσης ConvNet**

Το πρώτο επίπεδο του μοντέλου δέχεται ως είσοδο τα patches διαστάσεων 4x5x5 τα οποία επεξεργάζεται με φίλτρα 4x3x3, με βήμα 1 και zero-padding ίσο με 1. Οι διαστάσεις μένουν αμετάβλητες και τα patches εισάγονται στο δεύτερο επίπεδο το οποίο εφαρμόζει σε αυτά τη συνάρτηση tangent. Το τρίτο επίπεδο MaxPooling δίνει ως αποτέλεσμα ένα πίνακα 4x2x2 εφαρμόζοντας φίλτρα 4x3x3 με βήμα 2. Τα επίπεδα 4,5 και 6 ακολουθούν την ίδια λογική(Convolutional-Tangent-MaxPooling). Το τέταρτο επίπεδο είναι ίδιο με το πρώτο ενώ το έκτο εφαρμόζει φίλτρα 4x2x2 με βήμα 2 καταλήγοντας σε ένα πίνακα διαστάσεων 64x1x1. Το έβδομο επίπεδο συμβάλλει στο μετασχηματισμό του πολυδιάστατου πίνακα 64x1x1 σε μονοδιάστατο πίνακα διαστάσεων 64*1*1=64. Το όγδοο επίπεδο μετατρέπει το μονοδιάστατο πίνακα από 64x1 σε 30x1 και το ένατο επίπεδο εφαρμόζει τη συνάρτηση tangent. Το τελευταίο επίπεδο δίνει ως έξοδο τον τελικό πίνακα διαστάσεων 8x1, όπου 8 είναι ο αριθμός των κλάσεων προς ταξινόμηση. Το μοντέλο εκπαιδεύθυηκε για 15 εποχές με learningrate ίσο με 1, weightDecay ίσο με 0.0005, momentum ίσο με 0.9 και learningRateDecay ίσο με 1e-7. Ανά δύο εποχές, το learningrate μειωνόταν στο μισό. Μετά το τέλος της διαδικασίας εκπαίδευσης το μοντέλο χρησιμοποιήθηκε για την ταξινόμηση των 2 εικόνων που δε χρησιμοποιήθηκαν και τα αποτελέσματα ήταν τα εξής:

i. 1η Εικόνα ελέγχου

Στον Πίνακα 4.3 δίνεται ο πίνακας σύγχησης που προέκυψε. Η συνολική ακρίβεια και ο δείκτης Κ είναι 86.8% και 0.827 αντίστοιχα. Παρ'όλα αυτά οι ακρίβειες του παραγωγού (PA) και του χρήστη (UA) δεν είναι ιδιαίτερα ικανοποιητικές για όλες τις κλάσεις. Οι υψηλότερες ακρίβειες σημειώνονται για τις κλάσεις δέντρα (92.05% PA και 93.74% UA), γρασίδι (90.08% PA και 88.87 UA%) και νερό (98.7% PA και 99.7% UA). Τα υψηλά αυτά ποσοστά οφείλονται στο υπέρυθρο κανάλι που περιέχουν οι εικόνες. Όσο αφορά τις κλάσεις δρόμοι (79.08% PA και 82.32% UA) και κτίρια (80.17% PA και 78.11% UA) οι ακρίβειες είναι χαμηλότερες, ενώ οι δύο αυτές κλάσεις συγχέονται αρκετά μεταξύ τους. Αυτό είναι λογικό αφού οι δύο αυτές κατηγορίες έχουν κοινά φασματικά χαρακτηριστικά. Τέλος, η κλάση σιδηρόδρομοι (9.13% PA και 3.02% UA) δεν έχει αναγνωριστεί σχεδόν καθόλου από το μοντέλο με τα περισσότερα patches να ταξινομούνται λανθασμένα στην κλάση κτίρια, και μερικά από αυτά στην κλάση δρόμοι. Η κλάση σιδηρόδρομοι έχει επίσης κοινά φασματικά χαρακτηριστικά με τις κλάσεις δρόμοι και κτίρια, γι'αυτό και συγχέεται με αυτές. Επίσης, οι ιδιαίτερα μικρές διαστάσεις των patches εμποδίζουν το μοντέλο να εντοπίζει τα χαρακτηριστικά που περιβάλλουν τις περιοχές ενδιαφέροντος και έτσι δε βοηθούν στο διαχωρισμό των κλάσεων.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**102308**	23998	265	23	10	218	2545	0	129367	79.08
Buildings	20509	**98095**	344	616	74	65	2654	0	122357	80.17
Trees	810	452	**50347**	2944	0	0	142	0	54695	92.05
Grass	199	162	2729	**28674**	0	8	61	0	31833	90.08
Bare Soil	0	0	0	0	**0**	0	0	0	0	nan
Water	260	1435	6	0	0	**129027**	0	0	130728	98.70
Railways	199	1444	20	9	0	0	**168**	0	1840	9.13
Swimming Pool	0	0	0	0	0	0	0	**0**	0	nan
Total	124285	125586	53711	32266	84	129318	5570	0	**470820**	
UA(%)	82.32	78.11	93.74	88.87	nan	99.77	3.02	nan		
	Overall Accuracy = 86.8 % , Kappa Coefficient = 0.827									

Πίνακας 4.3: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 5x5 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 86.8%

ii. 2η Εικόνα ελέγχου

Η ταξινόμηση εδώ έδωσε συνολική ακρίβεια και δείκτη K 89.5% και 0.843 αντίστοιχα. Η κλάση με τη μεγαλύτερη ακρίβεια είναι οι πισίνες (96.78% PA και 100% UA). Οι κλάσεις της βλάστησης, δηλαδή τα δέντρα (98.38% PA και 94.11% UA) και το γρασίδι (87.75% PA και 89.43 UA) παρουσιάζουν ικανοποιητικές ακρίβειες. Ακολουθούν οι κλάσεις δρόμοι (71.66% PA και 84.88% UA) και κτίρια (86.64% PA και 73.76% UA) οι οποίες και πάλι συγχέονται μεταξύ τους λόγω των κοινών φασματικών χαρακτηριστικών. Τέλος, η κλάση με τη χαμηλότερη ακρίβεια είναι το γυμνό έδαφος (62.80% PA και 94.21% UA) και συγχέεται με τους δρόμους και τα κτίρια λόγω κοινών φασματικών χαρακτηριστικών. Συγχέεται επίσης με τα δέντρα και το γρασίδι, διότι στη συγκεκριμένη εικόνα η βλάστηση περικλείει το γυμνό έδαφος τις περισσότρες φορές.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**36677**	10963	2344	734	209	37	218	0	51182	71.66
Buildings	4036	**34860**	425	242	427	84	160	0	40234	86.64
Trees	79	52	**182143**	2855	0	0	7	0	185136	98.38
Grass	244	64	7596	**57596**	134	0	2	0	65636	87.75
Bare Soil	2172	1234	1027	2976	**12526**	0	10	0	19945	62.80
Water	0	0	0	0	0	**0**	0	0	0	nan
Railways	0	0	0	0	0	0	**0**	0	0	nan
Swimming Pool	0	88	0	0	0	54	0	**4263**	4405	96.78
Total	43208	47261	193535	64403	13296	175	397	4263	**366538**	
UA(%)	84.88	73.76	94.11	89.43	94.21	nan	nan	100		
	Overall Accuracy = 89.5 % , Kappa Coefficient = 0.843									

Πίνακας 4.4: Πίνακας σύγχησης για τη 2η Εικόνα ελέγχου με patches διαστάσεων 5x5 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 89.5%

4.2.1.2 Patches διαστάσεων 11x11

- **Περιγραφή παραμέτρων του μοντέλου εκπαίδευσης ConvNet**

Το μοντέλο ConvNet που χρησιμοποιήθηκε για τη διαδικασία εκπαίδευσης περιέχει ίδιες παραμέτρους με το μοντέλο που περιγράφηκε στην ενότητα 4.2.1.1. Οι μόνες διαφορές έγκεινται στα τελευταία πλήρως συνδεδεμένα επίπεδα. Συγκεκριμένα, το έβδομο επίπεδο δέχεται ως είσοδο έναν πολυδιάστατο πίνακα διαστάσεων 64x2x2 τον οποίο μετατρέπει σε μονοδιάστατο, διαστάσεων 64*2*2=256x1. Το όγδοο γραμμικό επίπεδο μετατρέπει τον πίνακα από 256x1 σε 200x1. Το τελευταίο απίπεδο καταλήγει και πάλι σε πίνακα διαστάσεων 8x1.

i. 1η Εικόνα ελέγχου

Ο πίνακας σύγχησης της συγκεκριμένης εικόνας φαίνεται στον Πίνακα 4.5. Η συνολική ακρίβεια και ο δείκτης Κ είναι 90.3% και 0.827 αντίστοιχα. Τα αποτελέσματα είναι παρόμοια με την περίπτωση των διαστάσεων 5x5. Ξανά, οι κλάσεις με τις μεγαλύτερες ακρίβειες είναι τα δέντρα (93.84% PA και 96.24% UA), το γρασίδι (93.10% PA και 94.20% UA) και το νερό (99.4% PA και 99.78% UA). Ακολουθούν οι κλάσεις δρόμοι (81.52% PA και 88.16% UA) και κτίρια (88.21% PA και 81.78% UA). Τέλος, οι ακρίβειες της κλάσης σιδηρόδρομοι (32.73% PA και 13.63% UA) εξακολουθούν να βρίσκονται σε πολύ χαμηλά επίπεδα.

Οι διαφορές των ακριβειών ανάμεσα στις διαστάσεις 11x11 και 5x5 φαίνονται συγκεντρωμένες στον Πίνακα 4.6. Οι τέσσερις πρώτες κλάσεις έχουν σημειώσει μικρή αύξηση της τάξεως του 5%, ενώ η κλάση νερό έχει παραμείνει σχεδόν σταθερή. Η κλάση σιδηρόδρομοι έχει σημειώσει τη μεγαλύτερη βελτίωση αφού και οι δύο ακρίβειες (PA και UA) έχουν αυξηθεί σχεδόν στο τριπλάσιο σε σχέση με την περίπτωση των patches διαστάσεων 5x5. Επίσης, η συνολική ακρίβεια αυξήθηκε κατά 3.5% και ο δείκτης Κ κατά 0.045.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**104081**	21622	222	18	4	162	1565	0	127674	81.52
Buildings	12069	**107136**	239	360	244	101	1302	0	121451	88.21
Trees	1258	577	**50986**	1427	6	0	81	0	54335	93.84
Grass	427	179	1529	**29327**	0	12	26	0	31500	93.10
Bare Soil	0	0	0	0	**0**	0	0	0	0	nan
Water	168	583	0	0	0	**126572**	10	0	127333	99.40
Railways	52	916	0	0	0	0	**471**	0	1439	32.73
Swimming Pool	0	0	0	0	0	0	0	**0**	0	nan
Total	118055	131013	52976	31132	254	126847	3455	0	**463732**	
UA(%)	88.16	81.78	96.24	94.20	nan	99.78	13.63	nan		
	Overall Accuracy = 90.3 % , Kappa Coefficient = 0.872									

Πίνακας 4.5: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 11x11

Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 90.3%

1η Εικόνα		
Δ [(11x11) – (5x5)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	2.440	5.840
Buildings	8.040	3.670
Trees	1.790	2.500
Grass	3.020	5.330
Bare Soil	nan	nan
Water	0.700	0.010
Railways	23.600	10.610
Swim. Pools	nan	nan
	Δ(OA) (%)	**Δ(K)**
	3.500	0.045

Πίνακας 4.6: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 11x11 και 5x5 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 3.5%

ii. 2η Εικόνα ελέγχου

Η 2η εικόνα ελέγχου έδωσε συνολική ακρίβεια και δείκτη K 91.1% και 0.866 αντίστοιχα. Οι υψηλότερες ακρίβειες εξακολουθούν να αφορούν τις κλάσεις πισίνες (98.64% PA και 99.75% UA), δέντρα (99.29% PA και 94.52 UA) και γρασίδι (87.69% PA και 92.95% UA). Ακολουθούν οι κλάσεις δρόμοι (74.02% PA και 88.92% UA), κτίρια (91.03% PA και 75.93% UA) και γυμνό έδαφος (67.1 PA και 93.36% UA).

Στον Πίνακα 4.8, φαίνεται ότι οι ακρίβειες PA και UA έχουν σημειώσει αύξηση της τάξεως του 3% για τις δύο πρώτες κλάσεις, ενώ οι ακρίβειες της κλάσης δέντρα έχουν παραμείνει σχεδόν σταθερές. Όσο αφορά την κλάση γρασίδι, η ακρίβεια PA έχει παραμείνει σχεδόν σταθερή και η ακρίβεια UA έχει αυξηθεί κατά 3.52%. Στην κλάση γυμνό έδαφος, η ακρίβεια PA έχει αυξηθεί κατά 4.3% ενώ η ακρίβεια UA έχει παραμείνει σχεδόν σταθερή. Η ακρίβεια PA της κλάσης πισίνες έχει αυξηθεί κατά 1.86% ενώ η ακρίβεια UA δεν έχει ιδιαίτερες διαφορές. Τέλος, η συνολική ακρίβεια σε σχέση με τα patches διαστάσεων 5x5 έχει αυξηθεί κατά 1.6% και ο δείκτης K κατά 0.043%.

Από τα παραπάνω προκύπτει ότι και για τις 2 εικόνες ελέγχου η αύξηση των διαστάσεων των patches ήταν ωφέλιμη, ιδιαίτερα για τις κλάσεις δρόμοι, κτίρια και γυμνό έδαφος. Επίσης, αυξήθηκε η συνολική ακρίβεια και ο δείκτης K.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**37385**	10040	2374	397	103	4	190	11	50504	74.02
Buildings	2769	**36127**	317	174	80	126	96	0	39689	91.03
Trees	135	41	**181826**	1115	0	0	3	0	183120	99.29
Grass	367	15	6862	**56855**	739	0	1	0	64839	87.69
Bare Soil	1387	1301	997	2624	**12965**	0	47	0	19321	67.10
Water	0	0	0	0	0	**0**	0	0	0	nan
Railways	0	0	0	0	0	0	**0**	0	0	nan
Swimming Pool	0	57	0	0	0	3	0	**4345**	4405	98.64
Total	42043	47581	192376	61165	13887	133	337	4356	**361878**	
UA(%)	88.92	75.93	94.52	92.95	93.36	nan	nan	99.75		
	Overall Accuracy = 91.1 % , Kappa Coefficient = 0.886									

Πίνακας 4.7: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 11x11
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 91.1%

2η Εικόνα		
Δ [(11x11) – (5x5)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	2.360	4.040
Buildings	4.390	2.170
Trees	0.910	0.410
Grass	-0.060	3.520
Bare Soil	4.300	-0.850
Water	nan	nan
Railways	nan	nan
Swim. Pools	1.860	-0.250
	Δ(OA) (%)	**Δ(K)**
	1.600	0.043

Πίνακας 4.8: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 11x11 και 5x5
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 1.6%

4.2.1.3 Patches διαστάσεων 21x21

- **<u>Περιγραφή παραμέτρων του μοντέλου εκπαίδευσης ConvNet</u>**

Στην περίπτωση των διαστάσεων 21x21, οι παράμετροι του μοντέλου ConvNet των 10 επιπέδων έχουν ως εξής: Το πρώτο επίπεδο δέχεται ως είσοδο τα patches διαστάσεων 4x21x21 τα οποία επεξεργάζεται με φίλτρα 4x5x5, παράγοντας ένα πίνακα εξόδου 32x17x17. Το δεύτερο επίπεδο εφαρμόζει τη συνάρτηση tangent στον πίνακα. Το τρίτο επίπεδο MaxPooling δίνει ως αποτέλεσμα ένα πίνακα 32x8x8 εφαρμόζοντας φίλτρα 4x3x3 με βήμα 2. Τα επίπεδα 4,5 και 6 ακολουθούν την ίδια λογική(Convolutional-Tangent-MaxPooling). Το τέταρτο επίπεδο εφαρμόζει φίλτρα

5x5 ενώ το έκτο εφαρμόζει φίλτρα 4x2x2 με βήμα 2 καταλήγοντας σε ένα πίνακα διαστάσεων 64x2x2. Τα τέσσερα τελευταία επίπεδα είναι ίδια με αυτά που περιγράφηκαν στην περίπτωση των patches διαστάσεων 11x11.

i. 1η Εικόνα ελέγχου

Η 1η εικόνα ελέγχου έδωσε συνολική ακρίβεια και δείκτη K 92.4% και 0.900 αντίστοιχα, όπως φαίνεται στον Πίνακα 4.9. Οι ακρίβειες των κλάσεων ακολουθούν το ίδιο μοτίβο με τις προηγούμενες περιπτώσεις. Δηλαδή, πρώτες σε ακρίβεια είναι οι κλάσεις δέντρα (93.87% PA και 97.12% UA), γρασίδι (94.83 PA και 95.7% UA) και νερό (99.46% PA και 99.85% UA). Ακολουθούν οι κλάσεις δρόμοι (88.24% PA και 88.2% UA) και κτίρια (88.49% PA και 87.28% UA) οι οποίες εξακολουθούν να συγχέονται μεταξύ τους. Τέλος, η κλάση σιδηρόδρομοι (51.03% PA και 29.01% UA) παραμένει σε μη ικανοποιητικά επίπεδα όσο αφορά τις ακρίβειές της και συγχέεται με τις κλάσεις δρόμοι και κτίρια.

Στον Πίνακα 4.10 παρουσιάζονται οι διαφορές που σημειώθηκαν στις ακρίβειες των patches διαστάσεων 21x21 με τις ακρίβειες των patches διαστάσεων 11x11. Στην κλάση δρόμοι, η ακρίβεια PA έχει αυξηθεί κατά 6.72% ενώ η ακρίβεια UA έχει παραμείνει σχεδόν σταθερή. Στην κλάση κτίρια, η ακρίβεια PA έχει παραμείνει σχεδόν σταθερή και η ακρίβεια UA έχει αυξηθεί κατά 5.51%. Οι ακρίβειες της κλάσης δέντρα παρέμειναν ίδιες, ενώ οι ακρίβειες της κλάσης γρασίδι σημείωσαν ελάχιστη αύξηση της τάξης του 1%. Η κλάση νερό δε σημείωσε ιδιαίτερες διαφορές και οι ακρίβειες της κλάσης σιδηρόδρομοι αυξήθηκαν σχεδόν στο διπλάσιο σε σχέση με τα patches διαστάσεων 11x11. Τέλος, η συνολική ακρίβεια αυξήθηκε κατά 2.1% και ο δείκτης K κατά 0.028.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**109472**	13321	251	23	42	114	837	0	124060	88.24
Buildings	12991	**106126**	293	135	124	65	194	0	119928	88.49
Trees	1211	892	**50265**	1163	0	1	15	0	53547	93.87
Grass	206	393	946	**29366**	0	7	48	0	30966	94.83
Bare Soil	0	0	0	0	**0**	0	0	0	0	nan
Water	183	472	3	0	0	**121115**	0	0	121773	99.46
Railways	49	380	0	0	0	0	**447**	0%	876	51.03
Swimming Pool	0	0	0	0	0	0	0	**0**	0	nan
Total	124112	121584	51758	30687	166	121302	1541	0	**451150**	
UA(%)	88.20	87.29	97.12	95.70	nan	99.85	29.01	nan		
	Overall Accuracy = 92.4 % , Kappa Coefficient = 0.900									

Πίνακας 4.9: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 21x21

Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 92.4%

1η Εικόνα		
Δ [(21x21) – (11x11)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	6.720	0.040
Buildings	0.280	5.510
Trees	0.030	0.880
Grass	1.730	1.500
Bare Soil	nan	nan
Water	0.060	0.070
Railways	18.300	15.380
Swim. Pools	nan	nan
	Δ(OA) (%)	**Δ(K)**
	2.100	0.028

Πίνακας 4.10: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 21x21 και 11x11 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 2.1%

ii. 2η Εικόνα ελέγχου

Η 2η εικόνα ελέγχου έδωσε συνολική ακρίβεια 92.7% και δείκτη K 0.890. Από τον Πίνακα 4.11 προκύπτει ότι πρώτες σε ακρίβεια είναι οι κλάσεις δέντρα (99.24% PA και 95.44% UA), γρασίδι (90.93 PA και 92.77% UA) και πισίνες (99.21% PA και 98.14% UA). Ακολουθούν οι κλάσεις δρόμοι (80.15% PA και 91.35% UA), κτίρια (91.36% PA και 80.81% UA) και γυμνό έδαφος (69.07% PA και 96.33% UA)).

Στον Πίνακα 4.12, οι διαφορές των ακριβειών μεταξύ των patches διαστάσεων 21x21 και 11x11 έχουν ως εξής: Η κλάση δρόμοι σημείωσε αύξηση κατά 6.13% και 2.43% στις ακρίβειες PA και UA αντίστοιχα. Η κλάση κτίρια παρουσίασε αύξηση κατά 4.88% στην ακρίβεια UA, ενώ η ακρίβεια PA παρέμεινα σχεδόν σταθερή. Οι ακρίβειες της κλάσης δέντρα παρέμειναν σταθερές, ενώ η κλάση γρασίδι σημείωσε αύξηση κατά 3.24% στην ακρίβεια PA και σταθερότητα στην ακρίβεια UA. Όσο αφορά την κλάση γυμνό έδαφος, παρουσίασε αύξηση κατά 1.97% και 2.97% στις ακρίβειες PA και UA αντίστοιχα. Οι πισίνες παρουσίασαν μικρή μείωση στην ακρίβεια UA της τάξης του 1% και σταθερότητα στην ακρίβεια PA.

Από τα παραπάνω συμπεραίνεται ότι η αύξηση των διαστάσεων των patches από 11x11 σε 21x21 επέφερε αύξηση στις ακρίβειες, ιδιαίτερα για τις κλάσεις δρόμοι, κτίρια και σιδηρόδρομοι. Επίσης, αυξήθηκε η συνολική ακρίβεια και ο δείκτης K.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**39555**	6866	2096	620	63	0	114	39	49353	80.15
Buildings	2740	**35274**	304	160	0	87	2	44	38611	91.36
Trees	111	77	**177727**	1166	0	0	0	0	179081	99.24
Grass	176	67	5079	**57481**	413	0	0	0	63216	90.93
Bare Soil	698	1351	1016	2534	**12510**	0	2	0	18111	69.07
Water	0	0	0	0	0	**0**	0	0	0	nan
Railways	0	0	0	0	0	0	**0**	0	0	nan
Swimming Pool	19	16	0	0	0	0	0	**4370**	4405	99.21
Total	43299	43651	186222	61961	12986	87	118	4453	**352777**	
UA(%)	91.35	80.81	95.44	92.77	96.33	nan	nan	98.14		
	Overall Accuracy = 92.7 % , Kappa Coefficient = 0.890									

Πίνακας 4.11: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 21x21
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 92.7%

2η Εικόνα		
Δ [(21x21) – (11x11)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	6.130	2.430
Buildings	0.330	4.880
Trees	-0.050	0.920
Grass	3.240	-0.180
Bare Soil	1.970	2.970
Water	nan	nan
Railways	nan	nan
Swim. Pools	0.570	-1.610
	Δ(OA) (%)	**Δ(K)**
	1.600	0.004

Πίνακας 4.12: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 21x21 και 11x11
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 1.6%

4.2.1.4 Patches διαστάσεων 29x29

- **<u>Περιγραφή παραμέτρων του μοντέλου εκπαίδευσης ConvNet</u>**

Το μοντέλο ConvNet που χρησιμοποιήθηκε για τη διαδικασία εκπαίδευσης περιέχει ίδιες παραμέτρους με το μοντέλο που περιγράφηκε στην ενότητα 4.2.1.3, εκτός από το τρίτο επίπεδο MaxPooling το οποίο εφαρμόζει φίλτρα διαστάσεων 4x2x2 με βήμα 2. Επίσης, τα τελευταία πλήρως συνδεδεμένα επίπεδα παρουσιάζουν τις ακόλουθες δαφορές: Το έβδομο επίπεδο δέχεται ως είσοδο έναν πολυδιάστατο πίνακα διαστάσεων 64x4x4 τον οποίο μετατρέπει σε μονοδιάστατο, διαστάσεων 64*4*4=1024x1. Το όγδοο γραμμικό επίπεδο μετατρέπει τον πίνακα από 1024x1 σε

500x1. Το τελευταίο απίπεδο καταλήγει και πάλι σε πίνακα διαστάσεων 8x1.

i. 1η Εικόνα ελέγχου

Στον Πίνακα 4.13 φαίνεται ότι η 1η εικόνα ελέγχου έδωσε συνολική ακρίβεια και δείκτη K 93.2% και 0.911 αντίστοιχα. Ξανά, οι υψηλότερες ακρίβειες σημειώνονται στις κλάσεις δέντρα (93.03% PA και 97.45% UA), γρασίδι (96.19% PA και 95.05% UA) και νερό (99.61% PA και 99.94% UA). Ακολουθούν οι κλάσεις δρόμοι (86.55% PA και 92% UA), κτίρια (93.24% PA και 86.81% UA) και σιδηρόδρομοι (39.41% PA και 15.53% UA).

Στον Πίνακα 4.14, φαίνονται οι διαφορές που σημειώθηκαν στις ακρίβειες μεταξύ των patches διαστάσεων 29x29 και των patches διαστάσεων 21x21. Η κλάση δρόμοι παρουσίασε μείωση κατά 1.69% στην ακρίβεια PA και αύξηση 3.8% στην ακρίβεια UA. Η κλάση κτίρια παρουσίασε αύξηση κατά 4.75% στην ακρίβεια PA ενώ η ακρίβεια UA παρέμεινα σχεδόν σταθερή. Οι ακρίβειες των κλάσεων δέντρα και νερό παρέμειναν σταθερές, ενώ η κλάση γρασίδι σημείωσε αύξηση κατά 1.36% στην ακρίβεια PA και σταθερότητα στην ακρίβεια UA. Όσο αφορά την κλάση σιδηρόδρομοι, παρουσιάστηκε μεγάλη μείωση στις ακρίβειες. Συγκεκριμένα, η ακρίβεια PA μειώθηκε κατά 11.62% και η ακρίβεια UA κατά 13.48%. Τέλος, η συνολική ακρίβεια φαίνεται να αυξήθηκε κατά 0.8% και ο δείκτης K κατά 0.011.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**104494**	14813	276	102	0	63	984	0	120732	86.55
Buildings	7602	**110786**	286	53	0	0	94	0	118821	93.24
Trees	1202	1104	**49159**	1373	0	0	4	0	52842	93.03
Grass	202	233	721	**29359**	0	7	0	0	30522	96.19
Bare Soil	0	0	0	0	**0**	0	0	0	0	nan
Water	72	389	0	0	0	**116998**	0	0	117459	99.61
Railways	8	298	0	0	0	0	**199**	0	505	39.41
Swimming Pool	0	0	0	0	0	0	0	**0**	0	nan
Total	113580	127623	50442	30887	0	117068	1281	0	**440881**	
UA(%)	92.00	86.81	97.46	95.05	nan	99.94	15.53	nan		
	Overall Accuracy = 93.2 % , Kappa Coefficient = 0.911									

Πίνακας 4.13: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 29x29

Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.2%

1η Εικόνα		
Δ [(29x29) – (21x21)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	-1.690	3.800
Buildings	4.750	-0.480
Trees	-0.840	0.340
Grass	1.360	-0.650
Bare Soil	nan	nan
Water	0.150	0.090
Railways	-11.620	-13.480
Swim. Pools	nan	nan
	Δ(OA) (%)	**Δ(K)**
	0.800	0.011

Πίνακας 4.14: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 29x29 και 21x21 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.8%

ii. 2η Εικόνα ελέγχου

Η χρήση των patches διαστάσεων 29x29 για τη 2η εικόνα ελέγχου έδωσε συνολική ακρίβεια ίση με 93.5% και δείκτη K ίσο με 0.903. Πρώτες και πάλι σε ακρίβεια είναι οι κλάσεις δέντρα (98.77% PA και 96.44% UA), γρασίδι (92.43% PA και 93.11% UA) και πισίνες (99.73% PA και 89.78% UA). Ακολουθούν οι κλάσεις δρόμοι (80.61% PA και 94.89% UA), κτίρια (94.07% PA και 81.78% UA) και γυμνό έδαφος (76.36% PA και 91.26% UA).

Όσο αφορά τις διαφορές των ακριβειών μεταξύ των patches διαστάσεων 29x29 και των patches διαστάσεων 21x21, αυτές φαίνονται στον Πίνακα 4.16. Πιο αναλυτικά, η κλάση δρόμοι παρουσίασε αύξηση κατά 3.54% στην ακρίβεια UA ενώ η ακρίβεια PA παρέμεινε σχεδόν σταθερή. Η δεύτερη κλάση, η οποία είναι τα κτίρια, παρουσίασε μικρή αύξηση στις ακρίβειεα PA και UA κατά 2.71% και 0.97% αντίστοιχα. Οι κλάσεις της βλάστησης, δέντρα και γρασίδι, παρέμειναν σχεδόν σταθερές. Η κλάση γυμνό έδαφος παρουσίασε αύξηση στην ακρίβεια PA κατά 7.29% και μείωση στην κλάση UA κατά -5.07%. Τέλος, οι πισίνες είχαν μείωση κατά -8.36% στην ακρίβεια UA και σταθερότητα στην ακείβεια PA. Η συνολική ακρίβεια αυξήθηκε κατά 0.8% και ο δείκτης K κατά 0.013.

Συμπερασματικά, η αύξηση των διαστάσεων των patches από 21x21 σε 29x29 δεν είχε ιδιαίτερες αυξήσεις για τις έξι πρώτες κλάσεις και την κλάση νερό. Οι κλάσεις σιδηρόδρομοι και πισίνες αντίθετα, σημείωσαν σημαντική μείωση.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**39012**	6255	2017	780	144	26	0	160	48394	80.61
Buildings	1407	**35420**	198	175	0	112	0	340	37652	94.07
Trees	222	100	**173327**	1835	0	0	0	0	175484	98.77
Grass	247	59	3274	**57234**	1110	0	0	0	61924	92.43
Bare Soil	225	1464	917	1446	**13094**	0	1	0	17147	76.36
Water	0	0	0	0	0	**0**	0	0	0	nan
Railways	0	0	0	0	0	0	**0**	0	0	nan
Swimming Pool	0	12	0	0	0	0	0	**4393**	4405	99.73
Total	41113	43310	179733	61470	14348	138	1	4893	**345006**	
UA(%)	94.89	81.78	96.44	93.11	91.26	nan	nan	89.78		
	Overall Accuracy = 93.5 % , Kappa Coefficient = 0.903									

Πίνακας 4.15: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 29x29
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.5%

2η Εικόνα		
Δ [(29x29) – (21x21)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	0.460	3.540
Buildings	2.710	0.970
Trees	-0.470	1.000
Grass	1.500	0.340
Bare Soil	7.290	-5.070
Water	nan	nan
Railways	nan	nan
Swim. Pools	0.520	-8.360
	Δ(OA) (%)	**Δ(K)**
	0.800	0.013

Πίνακας 4.16: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 29x29 και 21x21
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.8%

4.2.1.5 Patches διαστάσεων 33x33

- **<u>Περιγραφή παραμέτρων του μοντέλου εκπαίδευσης ConvNet</u>**

Το μοντέλο ConvNet που χρησιμοποιήθηκε για τη διαδικασία εκπαίδευσης περιέχει ίδιες παραμέτρους με το μοντέλο που περιγράφηκε στην ενότητα 4.2.1.4, εκτός από το τρίτο επίπεδο MaxPooling το οποίο εφαρμόζει φίλτρα διαστάσεων 4x3x3 με βήμα 2. Επίσης, τα τελευταία πλήρως συνδεδεμένα επίπεδα παρουσιάζουν τις ακόλουθες δαφορές: Το έβδομο επίπεδο δέχεται ως είσοδο έναν πολυδιάστατο πίνακα διαστάσεων 64x5x5 τον οποίο μετατρέπει σε μονοδιάστατο, διαστάσεων

64*5*5=1600x1. Το όγδοο γραμμικό επίπεδο μετατρέπει τον πίνακα από 1600x1 σε 700x1. Το τελευταίο απίπεδο καταλήγει και πάλι σε πίνακα διαστάσεων 8x1.

i. 1η Εικόνα ελέγχου

Σύμφωνα με τον Πίνακα 4.17, τα υψηλότερα ποσοστά ακριβειών ανήκουν και πάλι στις κλάσεις δέντρα (94.87% PA και 96.64% UA), γρασίδι (94.79% PA και 96.87% UA) και νερό (99.6% PA και 99.89% UA). Ακολουθούν οι κλάσεις δρόμοι (85.73% PA και 92.27% UA), κτίρια (93.47% PA και 86.16% UA) και σιδηρόδρομοι (0% PA και 0% UA). Οι κλάσεις δρόμοι και κτίρια εξακολουθούν να συγχέονται μεταξύ τους ενώ η κλάση σιδηρόδρομοι δεν έχει αναγνωριστεί καθόλου. Επίσης, η συνολική ακρίβεια ισούται με 93.2% και ο δείκτης K με 0.910.

Στον Πίνακα 4.18 φαίνεται ότι η αύξηση των διαστάσεων των patches από 29x29 σε 33x33 παρουσίασε μείωση σχεδόν σε όλες τις κλάσεις και ιδιαίτερα στην κλάση σιδηρόδρομοι. Επίσης, η συνολική ακρίβεια παρέμεινε σταθερή και ο δείκτης K μειώθηκε ελάχιστα κατά 0.001%.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**102082**	15636	403	143	0	91	717	0	119072	85.73
Buildings	7336	**110537**	315	44	1	11	18	0	118262	93.47
Trees	970	980	**49764**	740	0	0	3	0	52457	94.87
Grass	195	366	999	**28718**	0	20	0	0	30298	94.79
Bare Soil	0	0	0	0	**0**	0	0	0	0	nan
Water	57	393	12	0	0	**114882**	0	0	115344	99.60
Railways	0	375	0	0	0	0	**0**	0	375	0.00
Swimming Pool	0	0	0	0	0	0	0	**0**	0	nan
Total	110640	128287	51493	29645	1	115004	738	0	**435808**	
UA(%)	92.27	86.16	96.64	96.87	nan	99.89	0.00	nan		
	Overall Accuracy = 93.2 % , Kappa Coefficient = 0.910									

Πίνακας 4.17: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 33x33

Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.2%

1η Εικόνα		
Δ [(33x33) – (29x29)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	-0.820	0.270
Buildings	0.230	-0.650
Trees	1.840	-0.820
Grass	-1.400	1.820
Bare Soil	nan	nan
Water	-0.010	-0.050
Railways	-39.410	-15.530
Swim. Pools	nan	nan
	Δ(OA) (%)	**Δ(K)**
	0.000	-0.001

Πίνακας 4.18: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 33x33 και 29x29 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.0%

ii. 2η Εικόνα ελέγχου

Οι κλάσεις δέντρα (99.29% PA και 94.58% UA), γρασίδι (88.66% PA και 91.9% UA) και πισίνες (98.98% PA και 88.65% UA) σημείωσαν τις υψηλότερες ακρίβειες. Ακολουθούν και πάλι οι κλάσεις δρόμοι (79.98% PA και 94.11% UA), κτίρια (92.35% PA και 81.9% UA) και γυμνό έδαφος (68.54% PA και 95.86% UA). Η συνολική ακρίβεια ισούται με 92.4% και ο δείκτης K με 0.886. Τα αριθμητικά αποτελέσματα που αναφέρθηκαν δίνονται στον Πίνακα 4.19.

Οι διαφορές που φαίνονται στον Πίνακα 4.20 δείχνουν ότι η αύξηση των διαστάσεων των patches από 29x29 σε 33x33 είχε ως επί το πλείστον αρνητική επίδραση στις ακρίβειες.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**38363**	6085	2470	790	60	11	2	182	47963	79.98
Buildings	1700	**34342**	314	280	0	174	0	376	37186	92.35
Trees	134	133	**172348**	960	1	0	0	0	173576	99.29
Grass	265	44	6202	**54311**	433	0	0	0	61255	88.66
Bare Soil	300	1284	900	2760	**11425**	0	0	0	16669	68.54
Water	0	0	0	0	0	**0**	0	0	0	nan
Railways	0	0	0	0	0	0	**0**	0	0	nan
Swimming Pool	0	45	0	0	0	0	0	**4360**	4405	98.98
Total	40762	41933	182234	59101	11919	185	2	4918	**341054**	
UA(%)	94.11	81.90	94.58	91.90	95.86	nan	nan	88.65		
	Overall Accuracy = 92.4 % , Kappa Coefficient = 0.886									

Πίνακας 4.19: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 33x33
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 92.4%

2η Εικόνα		
Δ [(33x33) – (29x29)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	-0.630	-0.780
Buildings	-1.720	0.120
Trees	0.520	-1.860
Grass	-3.770	-1.210
Bare Soil	-7.820	4.600
Water	nan	nan
Railways	nan	nan
Swim. Pools	-0.750	-1.130
	Δ(OA) (%)	**Δ(K)**
	-1.100	-0.017

Πίνακας 4.20: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 33x33 και 29x29 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του -1.1%

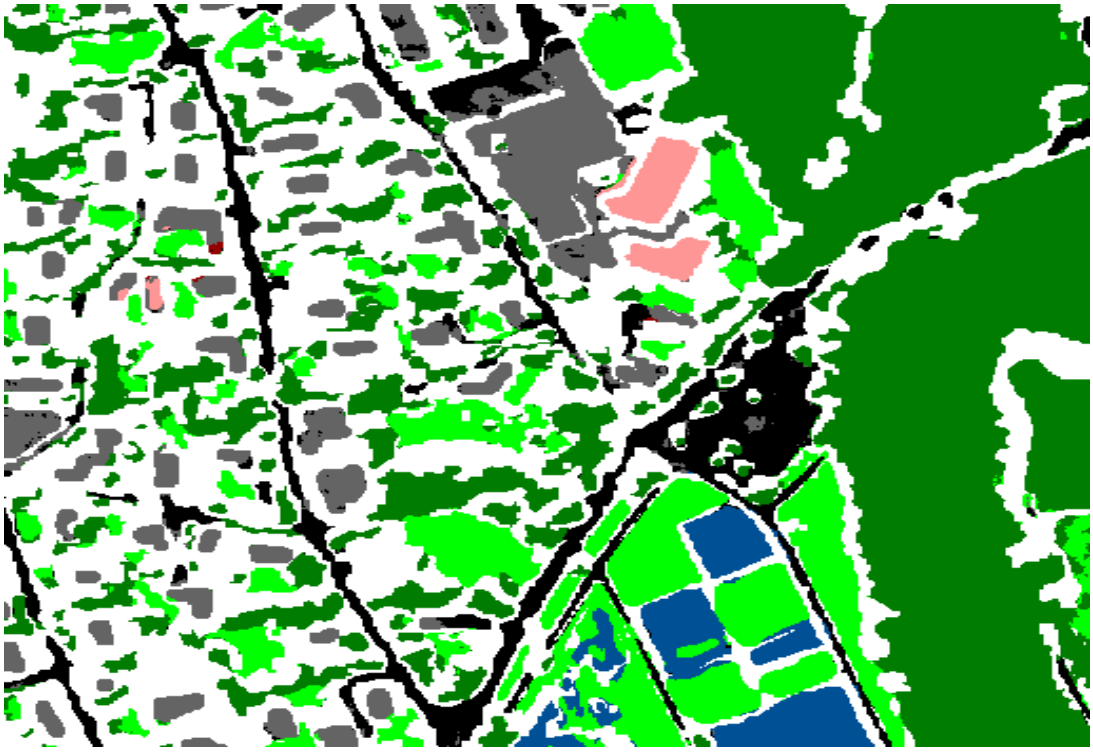
4.2.2 Βέλτιστη διάσταση των patches

Σύμφωνα με τα παραπάνω αποτελέσματα για τα δεδομένα 'Zurich Summer Dataset v1.0', η βέλτιστη διάσταση των patches είναι ίση με 29x29, αφού στην περίπτωση αυτή η συνολική ακρίβεια και ο δείκτης K είχαν τις μεγαλύτερες τιμές.

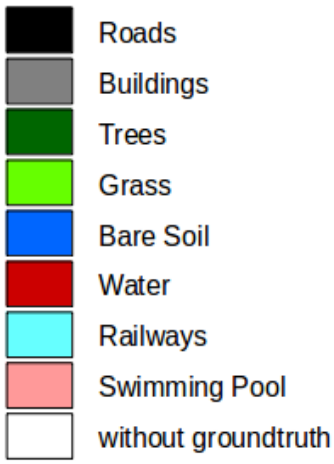
Στις Εικόνες 4.3 και 4.4 δίνεται η γραφική απεικόνιση των προβλέψεων του μοντέλου ConvNet για patches διαστάσεων 29x29. Τα χρώματα που αντιστοιχούν σε κάθε κλάση δίνονται στην Εικόνα 4.5. Μέσα από τη σύγκριση των ταξινομημένων αυτών εικόνων με τις αρχικές (Εικόνες 4.6 και 4.7), προκύπτουν και σε ποιοτική μορφή τα αποτελέσματα των πινάκων ταξινόμησης. Συγκεκριμένα, στην 1η εικόνα ελέγχου η μεγαλύτερη σύγχηση επικρατεί μεταξύ των κλάσεων δρόμοι (μαύρο) και κτίρια (γκρι). Επίσης, η κλάση σιδηρόδρομοι (γαλάζιο) έχει ταξινομηθεί στο μεγαλύτερο μέρος της λανθασμένα ως δρόμος (μαύρο). Το αποτέλεσμα αυτό προέκυψε λόγω των κοινών φασματικών χαρακτηριστικών των δύο αυτών κλάσεων. Όσο αφορά τη 2η εικόνα ελέγχου, επικρατεί και πάλι σύγχηση μεταξύ των κλάσεων δρόμοι και κτίρια, ενώ το γυμνό έδαφος (μπλε) μπερδεύεται συχνά με τα κτίρια, τα δέντρα (σκούρο πράσινο) και το γρασίδι (ανοιχτό πράσινο). Τέλος, ένα μικρό μέρος των κτιρίων φαίνεται να ταξινομείται λανθασμένα ως πισίνα (ροζ).

Με τη χρήση των βέλτιστων διαστάσεων 29x29 εκπαιδεύθηκαν επίσης τα μοντέλα AlexNet και VGG. Οι παράμετροι των μοντέλων είναι ίδιοι με αυτούς που περιγράφηκαν στο κεφάλαιο 3 για την ομάδα δεδομένων Deepsat. Παρακάτω δίνονται τα αποτελέσματα απο την ταξινόμηση των μοντέλων και για τις δύο εικόνες ελέγχου.

Εικόνα 4.3: Ποιοτική αξιολόγηση του μοντέλου ConvNet με patches διαστάσεων 29x29 για την 1η Εικόνα ελέγχου

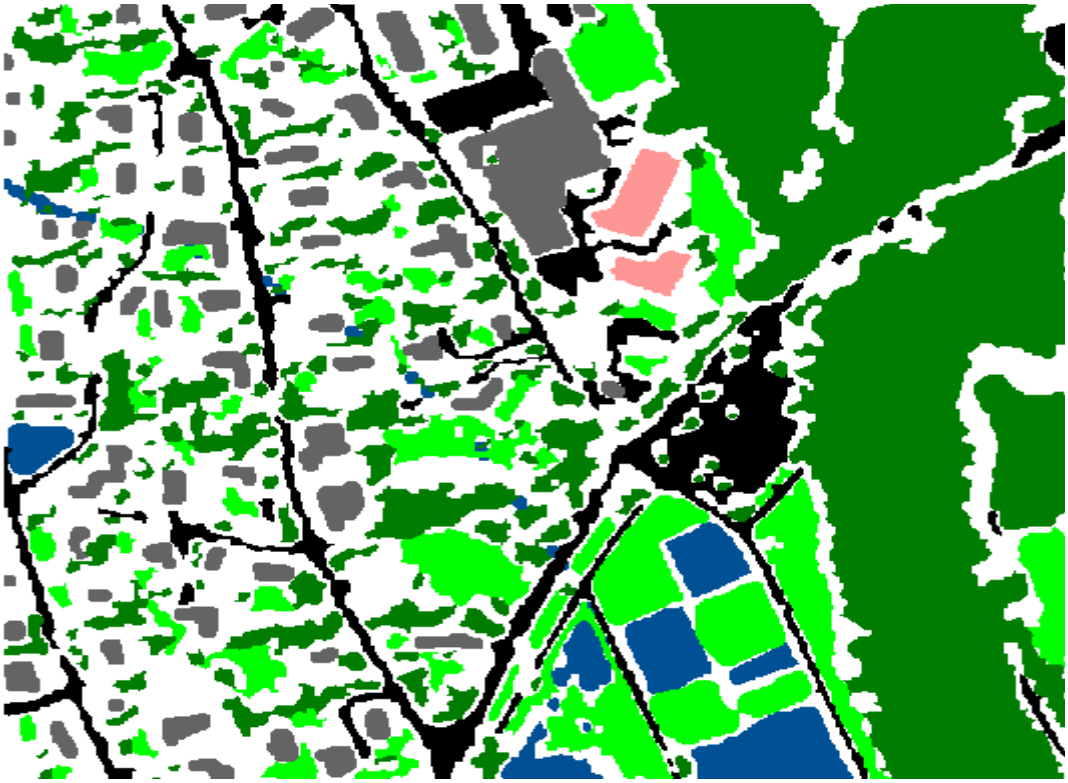


Εικόνα 4.4: *Ποιοτική αξιολόγηση του μοντέλου ConvNet με patches διαστάσεων 29x29 για τη 2η Εικόνα ελέγχου*



Εικόνα 4.5: Χρώματα κλάσεων για τις Εικόνες 4.3 και 4.4

Εικόνα 4.6: Groundtruth της 1ης εικόνας ελέγχου



Εικόνα 4.7: Groundtruth της 2ης εικόνας ελέγχου

4.2.2.1 AlexNet

Για την εκπαίδευση του μοντέλου AlexNet πραγματοποιήθηκαν 15 εποχές, ενώ ανά δύο εποχές ο ρυθμός εκμάθησης (learningRate) μειονόταν στο μισό.

i. 1η Εικόνα ελέγχου

Σύμφωνα με τον Πίνακα 4.21, οι τιμές των ακριβειών παρουσιάζουν το ίδιο μοτίβο. Δηλαδή, πρώτες σε ακρίβεια είναι οι κλάσεις δέντρα (93.23% PA και 97.11% UA), γρασίδι (95.66% PA και 92.99% UA) και νερό (99.56% PA και 99.95% UA). Ακολουθούν οι κλάσεις δρόμοι (88.56% PA και 90.22% UA) και κτίρια (90.89% PA και 88.91% UA), ενώ τελευταία είναι η κλάση σιδηρόδρομοι (50.89% PA και 17.33% UA).

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**106924**	12268	406	241	47	40	806	0	120732	88.56
Buildings	9635	**107999**	385	151	235	13	403	0	118821	90.89
Trees	1246	506	**49265**	1809	2	1	13	0	52842	93.23
Grass	391	255	661	**29198**	13	0	4	0	30522	95.66
Bare Soil	0	0	0	0	**0**	0	0	0	0	nan
Water	220	285	13	0	0	**116941**	0	0	117459	99.56
Railways	93	155	0	0	0	0	**257**	0	505	50.89
Swimming Pool	0	0	0	0	0	0	0	**0**	0	nan
Total	118509	121468	50730	31399	297	116995	1483	0	**440881**	
UA(%)	90.22	88.91	97.11	92.99	nan	99.95	17.33	nan		
	Overall Accuracy = 93.1 % , Kappa Coefficient = 0.910									

Πίνακας 4.21: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 29x29

Μοντέλο AlexNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.1%

ii. 2η Εικόνα ελέγχου

Στον Πίνακα 4.22 φαίνονται τα αποτελέσματα της ταξινόμησης για τη 2η εικόνα ελέγχου. Πρώτες σε ακρίβεια είναι και πάλι οι κλάσεις δέντρα (98.81% PA και 95.58% UA), γρασίδι (91.47% PA και 92.64% UA) και πισίνες (99.27% PA και 88.70% UA). Χαμηλότερες ακρίβειες παρουσιάζουν οι κλάσεις δρόμοι (80.91% PA και 90.48% UA), κτίρια (92.76% PA και 85.74% UA) και γυμνό έδαφος (73.94% PA και 94.85% UA).

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**39154**	4843	2711	929	149	9	117	482	48394	80.91
Buildings	2035	**34927**	270	282	90	9	5	34	37652	92.76
Trees	167	82	**173394**	1799	1	0	0	41	175484	98.81
Grass	641	40	4157	**56640**	446	0	0	0	61924	91.47
Bare Soil	1275	818	889	1487	**12678**	0	0	0	17147	73.94
Water	0	0	0	0	0	**0**	0	0	0	nan
Railways	0	0	0	0	0	0	**0**	0	0	nan
Swimming Pool	0	28	0	0	2	0	2	**4373**	4405	99.27
Total	43272	40738	181421	61137	13366	18	124	4930	**345006**	
UA(%)	90.48	85.74	95.58	92.64	94.85	nan	nan	88.70		
	Overall Accuracy = 93.1% , Kappa Coefficient = 0.897									

Πίνακας 4.22: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 29x29

Μοντέλο AlexNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.1%

4.2.2.2 VGG

Για την εκπαίδευση του μοντέλου VGG πραγματοποιήθηκαν 40 εποχές, ενώ ανά 4 εποχές ο ρυθμός εκμάθησης (learningRate) μειωνόταν στο μισό.

i. 1η Εικόνα ελέγχου

Στην περίπτωση του μοντέλου VGG, πρώτες σε ακρίβεια είναι οι κλάσεις δέντρα (93.46% PA και 97.88% UA), γρασίδι (96.24% PA και 94.54% UA) και νερό (99.76% PA και 99.81% UA). Ακολουθούν οι κλάσεις δρόμοι (93.24% PA και 92.38% UA), κτίρια (93.20% PA και 92.37% UA) και σιδηρόδρομοι (9.50% PA και 28.92% UA). Τα παραπάνω φαίνονται στον Πίνακα 4.23.

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**112571**	7169	529	251	0	205	7	0	120732	93.24
Buildings	7295	**110746**	330	296	37	6	111	0	118821	93.20
Trees	1341	963	**49386**	1150	0	2	0	0	52842	93.46
Grass	486	441	211	**29375**	0	9	0	0	30522	96.24
Bare Soil	0	0	0	0	**0**	0	0	0	0	nan
Water	166	115	0	0	0	**117178**	0	0	117459	99.76
Railways	2	455	0	0	0	0	**48**	0	505	9.50
Swimming Pool	0	0	0	0	0	0	0	**0**	0	nan
Total	121861	119889	50456	31072	37	117400	166	0	**440881**	
UA(%)	92.38	92.37	97.88	94.54	nan	99.81	28.92	nan		
	Overall Accuracy = 95.1 % , Kappa Coefficient = 0.936									

Πίνακας 4.23: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 29x29
Μοντέλο VGG: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 95.1%

ii. 2η Εικόνα ελέγχου

Στον Πίνακα 4.24 φαίνονται τα αποτελέσματα ταξινόμησης του μοντέλου VGG για τη 2η εικόνα ελέγχου. Πρώτες σε ακρίβεια είναι οι κλάσεις δέντρα (98.35% PA και 96.08% UA), γρασίδι (92.72% PA και 90.32% UA) και πισίνες (99.77% PA και 88.52% UA). Ακολουθούν οι κλάσεις δρόμοι (84.62% PA και 92.68% UA), κτίρια (95.14% PA και 89.09% UA) και γυμνό έδαφος (69.04% PA και 96.14% UA).

	Classification									
# of pixels	Roads	Buildings	Trees	Grass	Bare Soil	Water	Railways	Swimming Pool	**Total**	**PA(%)**
Reference Data										
Roads	**40950**	3767	2231	744	16	95	35	556	48394	84.62
Buildings	1464	**35823**	157	202	3	3	0	0	37652	95.14
Trees	291	141	**172593**	2444	1	0	0	14	175484	98.35
Grass	220	73	3757	**57419**	455	0	0	0	61924	92.72
Bare Soil	1261	398	889	2761	**11838**	0	0	0	17147	69.04
Water	0	0	0	0	0	**0**	0	0	0	nan
Railways	0	0	0	0	0	0	**0**	0	0	nan
Swimming Pool	0	10	0	0	0	0	0	**4395**	4405	99.77
Total	44186	40212	179627	63570	12313	98	35	4965	**345006**	
UA(%)	92.68	89.09	96.08	90.32	96.14	nan	nan	88.52		
	Overall Accuracy = 93.6 % , Kappa Coefficient = 0.905									

Πίνακας 4.24: Πίνακας σύγχησης για τη 2η Εικόνα ελέγχου με patches διαστάσεων 29x29
Μοντέλο VGG: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.6%

4.2.2.3 Σύγκριση μοντέλων

Στην υποενότητα αυτή τα αποτελέσματα των μοντέλων Alexnet και VGG συγκρίνονται τόσο μεταξύ τους, όσο και με το μοντέλο ConvNet που χρησιμοποιήθηκε για την εύρεση της βέλτιστης διάστασης. Οι συγκρίσεις γίνονται για διάσταση ίση με 29x29.

i. 1η Εικόνα ελέγχου

- Σύγκριση των μοντέλων AlexNet και ConvNet
  Στον Πίνακα 4.25 έχουν υπολογιστεί οι διαφορές που προκύπτουν αν αφαιρέσουμε τις ακρίβειες του μοντέλου ConvNet από τις ακρίβειες του μοντέλου AlexNet, για διάσταση ίση με 29x29. Οι ακρίβειες των κλάσεων της 1ης εικόνας ελέγχου δεν έχουν ιδιαίτερες διαφορές συγκρίνοντας τα δύο μοντέλα, εκτός από την κλάση σιδηρόδρομοι, της οποίας η ακρίβεια PA αυξήθηκε κατά 11.48%. Η συνολική ακρίβεια και ο δείκτης K παρέμειναν σχεδόν σταθεροί.

- Σύγκριση των μοντέλων VGG και ConvNet
  Το μοντέλο VGG επιφέρει αύξηση σε όλες τις ακρίβειες των κλάσεων, εκτός από την κλάση σιδηρόδρομοι η οποία σημειώνει μείωση στην ακρίβεια PA κατά -29.91%. Αντίθετα η ακρίβεια PA για την κλάση σιδηρόδρομοι αυξάνεται κατά 13.39%. Οι διαφορές των ακριβειών φαίνονται στον Πίνακα 4.26. Όσο αφορά τη συνολική ακρίβεια και το δείκτη K, αυξάνονται κατά 1.9% και 0.025 αντίστοιχα.

- Σύγκριση των μοντέλων VGG και AlexNet

  Οι διαφορές των ακριβειών μεταξύ των μοντέλων VGG και AlexNet για την 1η εικόνα ελέγχου φαίνονται στον Πίνακα 4.27. Στην περίπτωση αυτή, σχεδόν όλες οι κλάσεις έχουν σημειώσει μικρή αύξηση και η συνολική ακρίβεια έχει βελτιωθεί κατά 2%. Το ίδιο ισχύει και για το δείκτη K ο οποίος αυξήθηκε κατά 0.026. Η μόνη αστοχία εμφανίζεται στην κλάση σιδηρόδρομοι, όπου παρ'όλο που η ακρίβεια UA αυξάνεται κατά 11.529%, η ακρίβεια UA σημειώνει σημαντική μείωση κατά 41.39%.

1η Εικόνα (29x29)		
Δ [(AlexNet) – (ConvNet)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	2.010	-1.780
Buildings	-2.350	2.100
Trees	0.200	-0.350
Grass	-0.530	-2.060
Bare Soil	nan	nan
Water	-0.050	0.010
Railways	11.480	1.800
Swim. Pools	nan	nan
	Δ(OA) (%)	**Δ(K)**
	-0.100	-0.001

Πίνακας 4.25: Σύγκριση των μοντέλων AlexNet και ConvNet
Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου AlexNet για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του -0.1%

1η Εικόνα (29x29)		
Δ [(VGG) – (ConvNet)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	6.690	0.380
Buildings	-0.040	5.560
Trees	0.430	0.420
Grass	0.050	-0.510
Bare Soil	nan	nan
Water	0.150	-0.130
Railways	-29.910	13.390
Swim. Pools	nan	nan
	Δ(OA) (%)	**Δ(K)**
	1.900	0.025

Πίνακας 4.26: Σύγκριση των μοντέλων VGG και ConvNet
Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 1.9%

1η Εικόνα (29x29)		
Δ [(VGG) – (AlexNet)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	4.680	2.160
Buildings	2.310	3.460
Trees	0.230	0.770
Grass	0.580	1.550
Bare Soil	nan	nan
Water	0.200	-0.140
Railways	-41.390	11.590
Swim. Pools	nan	nan
	Δ(OA) (%)	**Δ(K)**
	2.000	0.026

Πίνακας 4.27: Σύγκριση των μοντέλων VGG και AlexNet
Οι ακρίβειες του μοντέλου AlexNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 2.0%

ii. 2η Εικόνα ελέγχου

- Σύγκριση των μοντέλων AlexNet και ConvNet
  Στον Πίνακα 4.28, φαίνεται ότι οι περισσότερες ακρίβειες έχουν σημειώσει μικρή μείωση στην ακρίβεια. Επίσης, η συνολική ακρίβεια έχει μειωθεί κατά 0.4% και ο δείκτης K κατά 0.006.

- Σύγκριση των μοντέλων VGG και ConvNet
  Στην περίπτωση αυτή οι ακρίβειες δεν έχουν ιδιαίτερες διαφορές, σύμφωνα με τον Πίνακα 4.29. Η συνολική ακρίβεια αυξάνεται κατά 0.1% και ο δείκτης K κατά 0.002.

- Σύγκριση των μοντέλων VGG και AlexNet
  Από τον Πίνακα 4.30, συμπεραίνεται ότι με τη χρήση του μοντέλου VGG, όλες οι κλάσεις της δεύτερης εικόνας ελέγχου σημειώνουν μικρή αύξηση. Το ίδιο και η συνολική ακρίβεια και ο δείκτης K που αυξάνονται κατά 0.5 % και 0.008 αντίστοιχα.

2η Εικόνα (29x29)		
Δ [(AlexNet) – (ConvNet)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	0.300	-4.410
Buildings	-1.310	3.960
Trees	0.040	-0.860
Grass	-0.960	-0.470
Bare Soil	-2.420	3.590
Water	nan	nan
Railways	nan	nan
Swim. Pools	-0.460	-1.080
	Δ(OA) (%)	**Δ(K)**
	-0.400	-0.006

Πίνακας 4.28: Σύγκριση των μοντέλων AlexNet και ConvNet
Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου AlexNet για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του -0.4%

2η Εικόνα (29x29)		
Δ [(VGG) – (ConvNet)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	4.010	-2.210
Buildings	1.070	7.310
Trees	-0.420	-0.360
Grass	0.290	-2.790
Bare Soil	-7.320	4.880
Water	nan	nan
Railways	nan	nan
Swim. Pools	0.040	-1.260
	Δ(OA) (%)	**Δ(K)**
	0.100	0.002

Πίνακας 4.29: Σύγκριση των μοντέλων VGG και ConvNet
Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.1%

2η Εικόνα (29x29)		
Δ [(VGG) – (AlexNet)]		
	Δ(PA) (%)	**Δ(UA) (%)**
Roads	3.710	2.200
Buildings	2.380	3.350
Trees	-0.460	0.500
Grass	1.250	-2.320
Bare Soil	-4.900	1.290
Water	nan	nan
Railways	nan	nan
Swim. Pools	0.500	-0.180
	Δ(OA) (%)	**Δ(K)**
	0.500	0.008

Πίνακας 4.30: Σύγκριση των μοντέλων VGG και AlexNet
Οι ακρίβειες του μοντέλου AlexNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.5%

4.3 Ποσοτική αξιολόγηση για τα δεδομένα δεδομένα DEIMOS-2 και IRIS

Στην ενότητα αυτή αναλύονται τα αποτελέσματα της ταξινόμησης ξεχωριστά για τις εικόνες DEIMOS-2 και IRIS

4.3.1 Δεδομένα DEIMOS-2

Τα παρακάτω αποτελέσματα προέκυψαν στα πλαίσια της δημοσίευσης 'Vakalopoulou M., Platias C., Papadomanolaki M., Paragios N., Karantzlos K. - Simultaneous Registration, Segmentation and Change Detection from Multisensor, Multitemporal Satellite Image Pairs', που πήρε τη 2η θέση στο διαγωνισμό IEEE GRSS Data Fusion που διοργανώθηκε το 2016.

- **Περιγραφή παραμέτρων του μοντέλου εκπαίδευσης ConvNet**

Η εκπαίδευση πραγματοποιείται με τη χρήση του μοντέλου ConvNet των 10 επιπέδων. Το πρώτο επίπεδο δέχεται ως είσοδο τα patches διαστάσεων 4x21x21 τα οποία επεξεργάζεται με φίλτρα 4x5x5, παράγοντας ένα πίνακα εξόδου 32x17x17. Το δεύτερο επίπεδο εφαρμόζει τη συνάρτηση tangent στον πίνακα. Το τρίτο επίπεδο MaxPooling δίνει ως αποτέλεσμα ένα πίνακα 32x8x8 εφαρμόζοντας φίλτρα 4x3x3 με βήμα 2. Τα επίπεδα 4,5 και 6 ακολουθούν την ίδια λογική(Convolutional-Tangent-MaxPooling). Το τέταρτο επίπεδο εφαρμόζει φίλτρα 5x5 ενώ το έκτο εφαρμόζει φίλτρα 4x2x2 με βήμα 2 καταλήγοντας σε ένα πίνακα διαστάσεων 64x2x2. Τα τέσσερα τελευταία επίπεδα είναι ίδια με αυτά που περιγράφηκαν στην περίπτωση των patches διαστάσεων 11x11. Το έβδομο επίπεδο συμβάλλει στο μετασχηματισμό του πολυδιάστατου πίνακα 64x2x2 σε μονοδιάστατο πίνακα διαστάσεων 64*2*2x1=256x1. Το όγδοο επίπεδο μετατρέπει το μονοδιάστατο πίνακα από 256x1 σε 200x1 και το ένατο επίπεδο εφαρμόζει τη συνάρτηση tangent. Το τελευταίο επίπεδο δίνει ως έξοδο τον τελικό πίνακα διαστάσεων 4x1 ή 6x1 ανάλογα με την ομάδα των patches που χρησιμοποιείται.

- **<u>Αποτελέσματα εικόνας Μαρτίου</u>**

Μετά το τέλος της διαδικασίας εκπαίδευσης, το μοντέλο ConvNet χρησιμοποιήθηκε για την ταξινόμηση ολόκληρων των εικόνων DEIMOS-2. Πιο συγκεκριμένα, σε κάθε pixel των εικόνων κεντράρεται ένα patch μεγέθους 4x21x21 και το εκπαιδευμένο πλέον μοντέλο προβλέπει σε ποιά κλάση ανήκει. Οι πίνακες σύγχησης των εικόνων του Μαρτίου και του Μαΐου φαίνονται στον Πίνακα 4.31 και στον Πίνακα 4.32 αντίστοιχα.

Από τον Πίνακα 4.31 προκύπτει ότι στην εικόνα του Μαρτίου οι μεγαλύτερες ακρίβειες σημειώθηκαν στις κλάσεις σκιές βλάστησης (85.9% PA και 86.2% UA), βλάστηση (86.1% PA και 98.1% UA) και θάλασσα (97% PA και 82.8% UA). Αυτό δείχνει ότι το υπέρυθρο κανάλι του δορυφόρου DEIMOS-2 συνέβαλλε σημαντικά στο διαχωρισμό των κλάσεων αυτών από τις υπόλοιπες. Χαμηλότερες ακρίβειες σημείωσαν οι κλάσεις πλοίο (72.9% PA και 61.4% UA), σκιά κτιρίου (57.7% PA και 80.7% UA), κτίριο (65.2% PA και 75.3% UA), δρόμοι (77.2% PA και 58.9% UA) και έδαφος (69.3% PA και 77.3% UA). Οι κλάσεις αυτές συγχέονται αρκετά μεταξύ τους αφού έχουν κοινά φασματικά χαρακτηριστικά.

	Classification									
# of pixels	Boat	Building Shadow	Building	Roads	Sea	Soil	Veget. Shadow	Vegeta-tion	**Total**	**PA(%)**
Reference data										
Boat	**30186**	490	9652	124	322	572	5	11	41362	72.9
Building Shadow	143	**26394**	2848	623	15326	4	355	0	45693	57.7
Building	16476	4981	**66530**	12327	648	943	75	18	101998	65.2
Roads	45	88	8230	**29079**	21	197	5	0	37665	77.2
Sea	2282	152	80	50	**84832**	0	12	7	87415	97.0
Soil	3	77	134	3964	1025	**11952**	80	0	17235	69.3
Veget. Shadow	0	514	28	116	113	2	**9596**	793	11162	85.9
Vegetation	0	0	807	3043	134	1784	1000	**42083**	48851	86.1
Total	49135	32696	88309	49326	102421	15454	11128	42912	**391381**	
UA(%)	61.4	80.7	75.3	58.9	82.8	77.3	86.2	98.1		
	Overall accuracy = 76.6%, Kappa coefficient = 0.719									

Πίνακας 4.31: Πίνακας σύγχησης
Εικόνα DEIMOS-2 - 15 Μαρτίου 2015
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 76.6%

- **<u>Αποτελέσματα εικόνας Μαΐου</u>**

Στην εικόνα του Μαΐου οι κλάσεις βλάστηση (92.78% PA και 84.9% UA), σκιές βλάστησης (78.3 PA και 85.8% UA) και θάλασσα (96.8% PA και 95% UA) χαρακτηρίζονται και πάλι από υψηλές ακρίβειες. Σε αντίθεση με την εικόνα του Μαρτίου, εδώ η κλάση έδαφος (98.2% PA και 94.8% UA) έχει διαχωριστεί με ιδιαίτερη επιτυχία και δε συγχέεται με τις κλάσεις πλοίο (78.9% PA και 59.4% UA), σκιά κτιρίου (68.2% PA και 87% UA), κτίριο (56.8% PA και 76.7% UA) και δρόμος (75.3% PA και 59.9% UA) οι οποίες χαρακτηρίζονται από χαμηλότερες ακρίβειες. Τα παραπάνω φαίνονται στον Πίνακα 4.32.

# of pixels	Classification: Boat	Building Shadow	Building	Roads	Sea	Soil	Veget. Shadow	Vegeta-tion	**Total**	**PA(%)**
Reference data										
Boat	**44398**	428	10051	180	41	220	0	0	54318	78.9
Building Shadow	53	**19936**	6	23	8202	0	1031	0	29251	68.2
Building	22556	1431	**55641**	16012	1353	978	5	20	97996	56.8
Roads	1000	48	6638	**25248**	0	3	0	600	33537	75.3
Sea	5974	40	0	0	**184687**	42	0	0	190743	96.8
Soil	90	8	58	23	1	**22899**	0	245	23324	98.2
Veget. Shadow	0	1008	6	4	0	0	**6260**	713	7991	78.3
Vegetation	0	0	86	600	2	6	2	**8926**	9622	92.7
Total	73071	22899	72486	42090	194286	24148	7298	10504	**446782**	
UA(%)	59.4	87.0	76.7	59.9	95.0	94.8	85.8	84.9		
	Overall accuracy = 82.2%, Kappa coefficient = 0.761									

Πίνακας 4.32: Πίνακας σύγχησης
Εικόνα DEIMOS-2 - 15 Μαΐου 2015
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 82.2%

4.3.2 Δεδομένα IRIS

- **<u>Περιγραφή παραμέτρων του μοντέλου εκπαίδευσης ConvNet</u>**

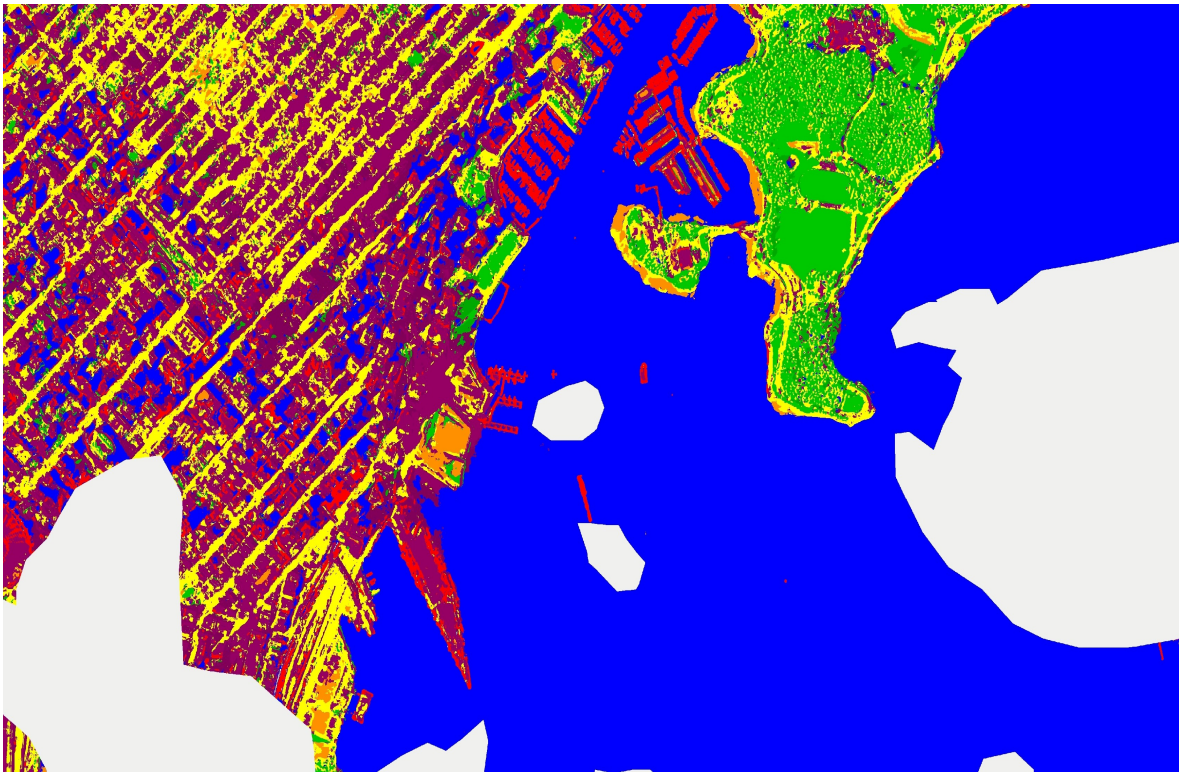
Το μοντέλο εκπαίδευσης που χρησιμοποιήθηκε για την ταξινόμηση της εικόνας IRIS έχει ακριβώς τις ίδιες παραμέτρους με το μοντέλο που περιγράφηκε για τις εικόνες DEIMOS-2, με μόνη διαφορά ότι τα κανάλια εισόδου είναι 3 και όχι 4, δηλαδή τα φίλτρα που εφαρμόζονται από το πρώτο επίπεδο είναι 3x5x5. Η διαδικασία πρόβλεψης από το μοντέλο πραγματοποίηθηκε και για την εικόνα IRIS του βίντεο. Τα αποτελέσματα του πίνακα σύγχησης φαίνονται στον Πίνακα 4.33. Το υπέρυθρο κανάλι αποτελεί σημαντική έλλειψη σε αυτή την περίπτωση, αφού η ακρίβεια της κλάσης βλάστηση (70.4% PA και 86.4% UA) δεν είναι ιδιαίτερα υψηλή εδώ. Η κλάση θάλασσα (83.2% PA και 80.9% UA) χαρακτηρίζεται από υψηλή ακρίβεια λόγω της διαφορετικότητάς της από τις υπόλοιπες. Οι κλάσεις πλοίο (68.1% PA και 66.7% UA), κτίριο (50.6% PA και 69.3% UA) και σκιά κτιρίου (57.7% PA και 60.7% UA) έχουν μέτριες ακρίβειες και συγχέονται μεταξύ τους, ενώ η κλάση δρόμος (86.1% PA και 43.3% UA) διαχωρίζεται με ιδιαίτερη επιτυχία από τις υπόλοιπες.

# of pixels	Classification								
	Boat	Building	Building Shadow	Soil	Sea	Vegeta-tion	Roads	**Total**	**PA(%)**
Reference data									
Boat	**12453**	4554	18	0	800	0	458	18283	68.1
Building	5805	**18785**	2147	326	160	12	9853	37088	50.6
Building Shadow	2	818	**22782**	0	9989	2543	3347	39481	57.7
Soil	0	32	0	**17560**	753	1867	5361	25573	68.6
Sea	340	0	11748	320	**61849**	0	80	74337	83.2
Vegetation	4	5	800	2832	2868	**28161**	5341	40011	70.4
Roads	87	2906	25	0	0	0	**18728**	21746	86.1
Total	17989	33400	38502	21038	75298	32583	27325	**246135**	
UA(%)	66.7	69.3	60.7	83.5	80.9	86.4	43.3		
	Overall accuracy = 70.34%, Kappa coefficient = 0.642								

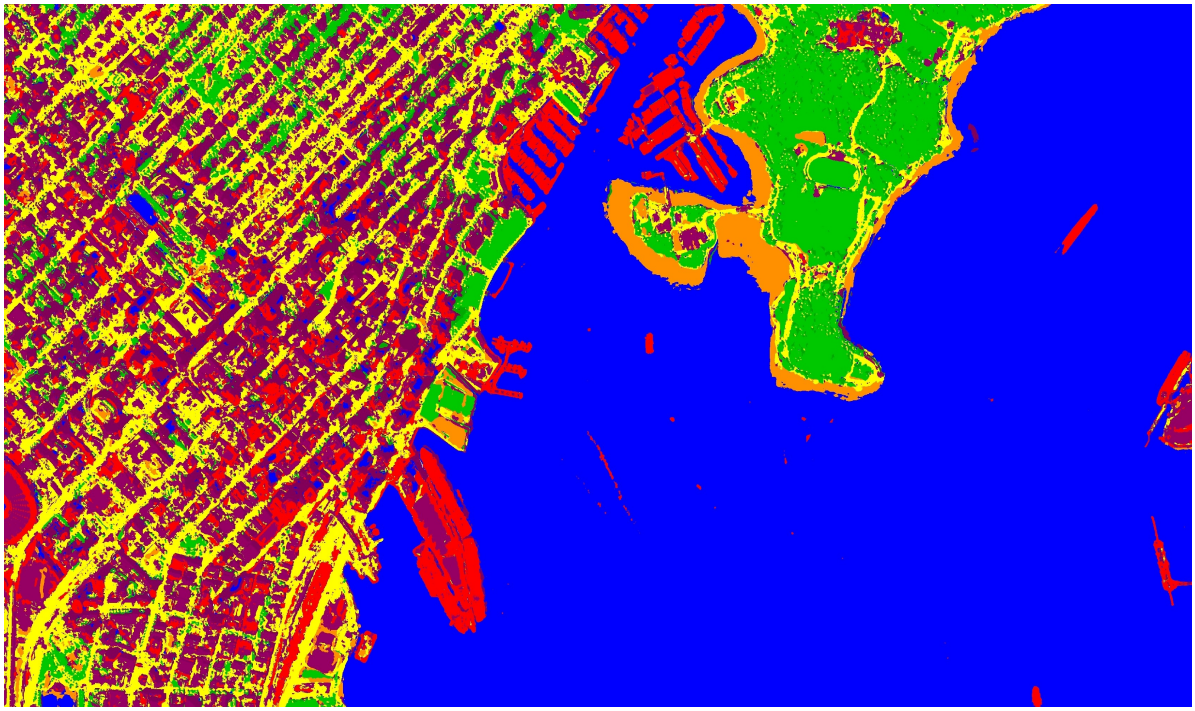
Πίνακας 4.33: Πίνακας σύγχησης
Εικόνα IRIS (βίντεο) – 17 Ιουλίου 2015
Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 70.34%

Στις Εικόνες 4.3, 4.4 και 4.5 δίνεται η γραφική απεικόνιση των προβλέψεων του μοντέλου για κάθε μία από τις τρεις εικόνες που χρησιμοποιήθηκαν. Τα χρώματα που αντιστοιχούν σε κάθε κλάση δίνονται στην Εικόνα 4.6. Μέσα από τη σύγκριση των ταξινομημένων αυτών εικόνων με τις αρχικές, προκύπτουν και σε ποιοτική μορφή τα αποτελέσματα των πινάκων ταξινόμησης.

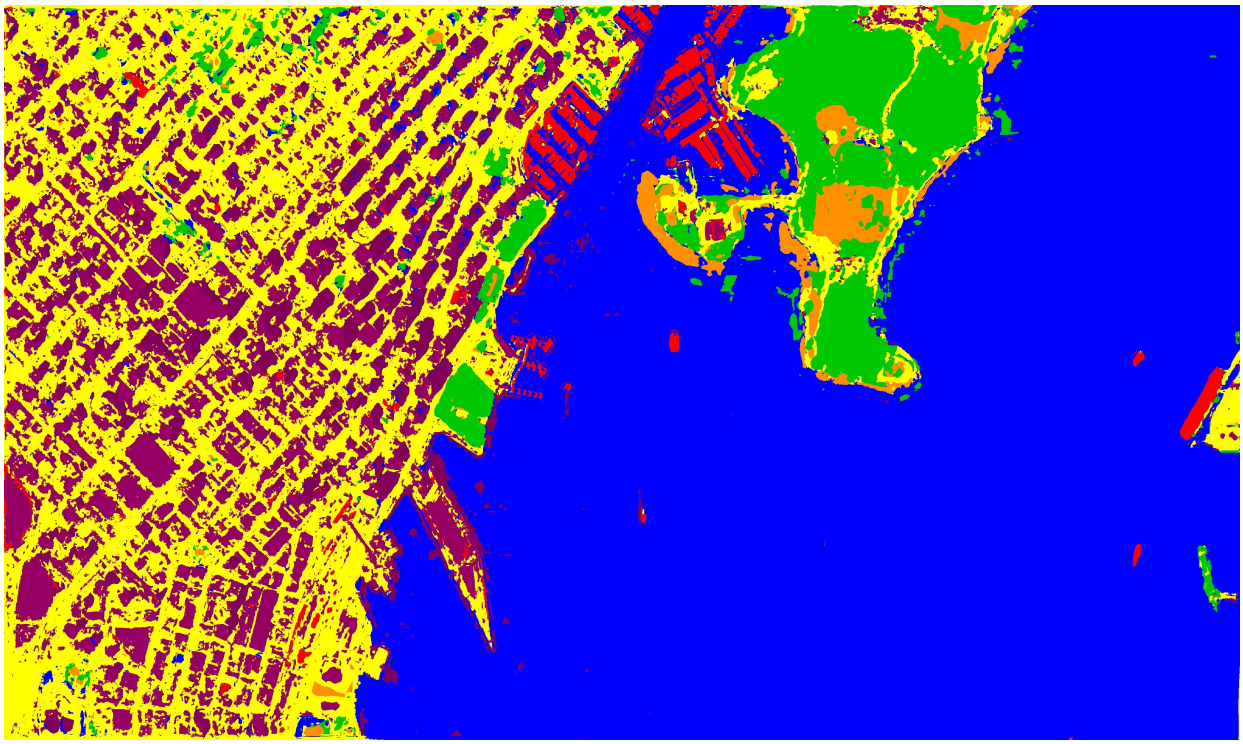
Συγκεκριμένα, στην εικόνα του Μαρτίου τα κτίρια (μωβ) μπερδεύονται αρκετά συχνά με τους δρόμους (κίτρινο) αλλά και τις σκιές κτιρίου (σκούρο μωβ) λόγω των κοινών φασματικών χαρακτηριστικών που διαθέτουν. Οι σκιές κτιρίου συγχέονται επίσης αρκετά συχνά και με τη θάλασσα (μπλε) για τους ίδιους λόγους. Άλλη μια λανθασμένη ταξινόμηση γίνεται επίσης και μεταξύ των πλοίων (κόκκινο) και των άσπρων κτιρίων (μωβ). Όσο αφορά την εικόνα του Μαΐου, οι συγχήσεις των κλάσεων είναι παρόμοιες με την εικόνα του Μαρτίου αλλά σε μικρότερο βαθμό. Η μόνη εντονότερη σύγχηση σε σχέση με την εικόνα του Μαρτίου φαίνεται να είναι μεταξύ των πλοίων και των άσπρων κτιρίων. Τέλος, η εικόνα του βίντεο παρουσιάζει τη μεγαλύτερη σύγχηση μεταξύ δρόμων και κτιρίων. Η χαμηλότερης ποιότητας ταξινόμηση που προέκυψε για την εικόνα του βίντεο πιθανόν αφείλεται στην έλλειψη του υπέρυθρου καναλιού.



Εικόνα 4.8: Χάρτης Ταξινόμησης
Εικόνα DEIMOS-2 - 15 Μαρτίου 2015
Μοντέλο ConvNet



Εικόνα 4.9: Χάρτης Ταξινόμησης
Εικόνα DEIMOS-2 - 15 Μαΐου 2015
Μοντέλο ConvNet



Εικόνα 4.10: Χάρτης Ταξινόμησης
Εικόνα IRIS (βίντεο) – 17 Ιουλίου 2015
Μοντέλο ConvNet

Εικόνα 4.11: Χρώματα κλάσεων για τους χάρτες ταξινόμησης

5. Συμπεράσματα και προοπτικές

Στο κεφάλαιο αυτό περιγράφονται τα συμπεράσματα που προκύπτουν απο τα αποτελέσματα των ταξινομήσεων που παρουσιάστηκαν στο Κεφάλαιο 4. Αρχικά γίνονται σχόλια για κάθε ομάδα δεδομένων που χρησιμοποιήθηκε και τέλος δίνονται κάποια γενικά συμπεράσματα.

5.1 Ειδικά Συμπεράσματα

Στην υποενότητα αυτή αναφέρονται τα συμπεράσματα που αφορούν τις ομάδες δεδομένων που χρησιμοποιήθηκαν.

5.1.1 Συμπεράσματα για την ομάδα δεδομένων Deepsat

Τα αποτελέσματα των ταξινομήσεων για την ομάδα δεδομένων Deepsat ήταν ιδιαίτερα ικανοποιητικά. Όλα τα συνελικτικά μοντέλα ξεπέρασαν σε ακρίβεια τη μέθοδο SVM, πράγμα που δείχνει ότι τα μοντέλα deep learning φαίνεται να έχουν υψηλότερες επιδόσεις. Από τα τρία μοντέλα που χρησιμοποιήθηκαν, δηλαδή το AlexNet small, το AlexNet και το VGG, το τελευταίο έδωσε τις καλύτερες ακρίβειες, με μικρή όμως διαφορά. Το μοντέλο που χρειάστηκε το λιγότερο χρόνο για την εκπαίδευσή του ήταν το ConvNet, σε αντίθεση με το μοντέλο VGG το οποίο ήταν το περισσότερο χρονοβόρο. Επίσης, το μοντέλο VGG χρειάζεται αρκετά περισσότερες εποχές προκειμένου να εκπαιδευθεί. Πρέπει εδώ να σημειωθεί, ότι η συγκεκριμένη ομάδα δεδομένων αποτελείται από μικρές εικόνες (patches) οι οποίες στο μεγαλύτερο μέρος της επιφάνειάς τους επεικονίζουν μία από τις κλάσεις ενδιαφέροντος. Συνεπώς, είναι λογικό η εκπαίδευση και ο έλεγχος των μοντέλων να γίνεται με επιτυχία. Για το λόγο αυτό τα μοντέλα εκπαιδεύθηκαν και με τις υπόλοιπες δύο ομάδες δεδομένων, των οποίων τα συμπεράσματα περιγράφονται παρακάτω.

5.1.2 Συμπεράσματα για την ομάδα δεδομένων 'Zurich Summer Dataset v1.0'

Για αυτή την ομάδα δεδομένων, η εκπαίδευση του μοντέλου ConvNet έγινε ξεχωριστά για κάθε διαφορετική διάσταση των patches. Υπενθυμίζεται ότι οι διαστάσεις ήταν οι εξής: 5x5, 11x11, 21x21, 29x29 και 33x33. Καθώς αυξάνονταν οι διαστάσεις των patches, οι ακρίβειες των ταξινομήσεων βελτιώνονταν μέχρι τη διάσταση 29x29. Πιο συγκεκριμένα, οι διαστάσεις 21x21 και 29x29 είχαν παρόμοια αποτελέσματα. Η περεταίρω αύξηση των διαστάσεων, δηλαδή τα patches διαστάσεων 33x33, επέφεραν μείωση στις ακρίβειες.

Τα patches ιδιαίτερα μικρών διαστάσεων, όπως 5x5 και 11x11, περιλαμβάνουν φασματική πληροφορία που στις περισσότερες περιπτώσεις αφορά μόνο την κλάση ενδιαφέροντος. Με την αύξηση των διαστάσεων, οι πληροφορίες εμπλουτίζονται αφού αρχίζουν να απεικονίζονται και τα στοιχεία που περικλείουν την εκάστοτε κλάση ενδιαφέροντος. Για το λόγο αυτό τα patches διαστάσεων 21x21 και 29x29 κατέληξαν σε υψηλότερες ακρίβειες, αφού βοήθησαν το μοντέλο να αναγνωρίσει καλύτερα τα πρότυπα των κλάσεων. Αντίθετα, τα patches διαστάσεων 33x33 επέφεραν μείωση στις ακρίβειες, το οποίο σημαίνει ότι η υπερβολική φασματική πληροφορία αποπροσανατολίζει τη διαδικασία εκπαίδευσης και οδηγεί σε ακρίβειες χαμηλότερου ποσοστού.

Τα patches διαστάσεων 29x29 χρησιμοποιήθηκαν και για την εκπαίδευση των μοντέλων AlexNet και VGG. Από τα δύο αυτά μοντέλα, το VGG είναι περισσότερο

χρονοβόρο και χρειάζεται περισσότερες εποχές για την εκπαίδευσή του λόγω του μεγαλύτερου βάθους του. Η χρήση του φαίνεται να επιφέρει αύξηση σχεδόν σε όλες τις κλάσεις, εκτός από την κλάση σιδηρόδρομοι. Η πιθανή αιτία αυτού του προβλήματος είναι ότι κατά τη διαδικασία εκπαίδευσης, τα patches που αντιστοιχούσαν στην κατηγορία σιδηρόδρομοι ήταν λιγότερα σε σχέση με τις υπόλοιπες κατηγορίες. Αυτό όμως δε μπορούσε να αποφευχθεί αφού οι ίδιες οι εικόνες περιείχαν λιγότερη πληροφορία για τη συγκεκριμένη κατηγορία. Επίσης, τα κοινά φασματικά χαρακτηριστικά που παρουσιάζει με τις κλάσεις δρόμοι και κτίρια, επέφεραν ακόμα μεγαλύτερη δυσκολία στην εκπαίδευση του μοντέλου. Αντίθετα, οι περιορισμένες πληροφορίες των εικόνων για την κλάση πισίνες δε δημιούργησαν πρόβλημα εξαιτίας των διαφορετικών φασματικών χαρακτηριστικών από τις υπόλοιπες κατηγορίες.

5.1.3 Συμπεράσματα για την ομάδα δεδομένων DEIMOS-2 και IRIS

Τόσο οι δύο εικόνες του δορυφόρου DEIMOS-2 όσο και η εικόνα του βίντεο, είχαν αρκετά ικανοποιητικά αποτελέσματα ταξινόμησης με συνολική ακρίβεια πάνω από 70% και στις τρεις περιπτώσεις. Η χρήση ενός μοντέλου με μεγαλύτερο βάθος πιθανόν να κατέληγε σε υψηλότερες ακρίβειες. Όμως, το πείραμα αυτό δείχνει για ακόμη μία φορά ότι ένα απλό μοντέλο ConvNet μπορεί να εκπαιδευθεί σε λίγο χρόνο και να είναι σε θέση να αναγνωρίζει καινούριες πληροφορίες με αρκετά καλές ακρίβειες.

5.2 Γενικά Συμπεράσματα

Σύμφωνα με τα παραπάνω, προκύπτουν δύο σημαντικά συμπεράσματα σχετικά με την εκπαίδευση μοντέλων με patches εικόνων πολύ υψηλής ανάλυσης (VHR).

Το πρώτο αφορά το μέγεθος των patches. Συγκεκριμένα, τα patches ιδιαίτερα μικρών διαστάσεων είναι καλύτερο να αποφεύγονται, λόγω της περιορισμένης πληροφορίας που διαθέτουν. Η πληροφορία αυτή αφορά τα φασματικά χαρακτηριστικά, την υφή, τα πρότυπα, τα σχήματα και γενικά οτιδήποτε μπορεί να οδηγήσει το μοντέλο στον επιτυχή εντοπισμό κάποιας κατηγορίας. Βέβαια, το υπερβολικό μέγεθος των patches μπορεί να επιφέρει χειρότερα αποτελέσματα αφού υπάρχει μεγάλη πιθανότητα σύγχησης μεταξύ των κλάσεων και υπερπροσαρμογής του μοντέλου. Για το λόγο αυτό, τα patches μετρίων διαστάσεων είναι περισσότερο ασφαλή για τις εικόνες πολύ υψηλής ανάλυσης. Όλα αυτά βέβαια εξαρτώνται πάντα και από το περιεχόμενο της εικόνας.

Το δεύτερο συμπέρασμα αφορά τα είδη των μοντέλων. Όπως φάνηκε, όσο περισσότερο βάθος έχει ένα μοντέλο, τόσο καλύτερα αποτελέσματα πετυχαίνει. Η εκπαίδευση όμως ενός μοντέλου με αρκετά μεγάλο βάθος, είναι καλό να γίνεται με ομοιόμορφα δεδομένα τα οποία περιέχουν ίδιο αριθμό patches από κάθε κατηγορία, είναι αντιπροσωπευτικά για κάθε κλάση και έχουν το κατάλληλο μέγεθος. Έτσι εξασφαλίζεται η επιτυχής εκπαίδευση του μοντέλου και η αποφυγή του φαινομένου της υπερπροσαρμογής.

Όσο αφορά τη σχέση των μοντέλων deep learning με το ελεύθερο λογισμικό, οι εφαρμογές που μπορούν να δημιουργηθούν είναι ποικίλλες. Σε αυτό βοηθάει και η ελεύθερη πρόσβαση στον πηγαίο κώδικα των εφαρμογών, η οποία προσφέρει στους

χρήστες τη δυνατότητα αλληλοβοήθειας και ανταλλαγής ιδεών.

5.3 Προτάσεις

Η μέθοδος ταξινόμησης deep learning είναι προς το παρόν ο αποτελεσματικότερος τρόπος αντιμετώπισης του προβλήματος διαχωρισμού των κατηγοριών σε δορυφορικές εικόνες. Για το λόγο αυτό, η ενασχόληση των χρηστών με το συγκεκριμένο πεδίο έρευνας θα ήταν ιδιαίτερα ελπιδοφόρα, αφού υπάρχουν ακόμα πολλά περιθώρια βελτίωσης των μοντέλων. Παράλληλα βέβαια, θα ήταν αρκετά ωφέλιμη η δημιουργία περισσότερων δεδομένων εκπαίδευσης, διαθέσιμων για πειράματα και αξιολόγηση των μοντέλων.

Όσο αφορά το ελεύθερο λογισμικό Orfeo Toolbox, προτείνεται ο περαιτέρω εμπλουτισμός του με βαθιές αρχιτεκτονικές τύπου Deep Learning, αφού για το συγκεκριμένο είδος ταξινόμησης τα διαθέσιμα εργαλεία δεν είναι ιδιαίτερα ανεπτυγμένα. Επίσης, ενθαρρύνεται η βελτίωση των εφαρμογών που δημιουργήθηκαν στη συγκεκριμένη εργασία από οποιοδήποτε χρήστη το επιθυμεί. Τέλος, θα ήταν ιδιαίτερα ωφέλιμη η δημιουργία ενός μοντέλου Deep Learning ικανού να ανταπεξέλθει στην ταξινόμηση διαφόρων ομάδων δεδομένων, με διαφορετικά χαρακτηριστικά και ιδιαιτερότητες.

Βιβλιογραφία

1. Marco Castelluccio, Giovanni Poggi, Carlo Sansone, Luisa Verdoliva, 2015. **Land Use Classification in Remote Sensing Images by Convolutional Neural Networks**, Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems Article No. 61

2. Saikat Basu, Sangram Ganguly, Supratik Mukhopadhyay, Robert DiBiano, Manohar Karki, Ramakrishna Nemani, 2015. **DeepSat – A Learning framework for Satellite Imagery**, 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems

3. Michele Volpi, Vittorio Ferrari, 2015. **Semantic segmentation of urban scenes by learning local class interactions**, 2015. In IEEE CVPR 2015 Workshop "Looking from above: when Earth observation meets vision" (EARTHVISION), Boston, USA, 2015

4. Adrien Lagrange, Bertrand Le Saux, Anne Beaupere, Alexandre Boulch, Adrien Chan-Hon-Tong, Stephane Herbin, Hicham Randrianarivo, Marin Ferecatu, 2015. **Benchmarking classification of earth-observation data: From learning explicit features to convolutional networks**, 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)

5. Sakrapee Paisitkriangkrai, Jamie Sherrah, Pranam Janney, Anton Van-Den Hengel, 2015. **Effective Semantic Pixel labelling with Convolutional Networks and Conditional Random Fields**, 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)

6. Pierre Sermanetk David Eigenk Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun, 2014. **OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks**, International Conference on Learning Representations

7. Karen Simonyan, Andrew Zisserman, 2015. **Very Deep Convolutional Networks for Large-Scale Image Recognition**, Published as a conference paper at ICLR 2015

8. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, 2012. **ImageNet Classification with Deep Convolutional Neural Networks**, Advances in Neural Information Processing Systems Conference

9. Dimitrios Marmanis, Mihai Datcu, Thomas Esch, Uwe Stilla, 2016. **Deep Learning Earth Observation Classification Using ImageNet Pretrained Networks**, 2016. IEEE Geoscience and Remote Sensing Letters

10. Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, 2014. **Rich feature hierarchies for accurate object detection and semantic segmentation**, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014

11. M. Vakalopoulou, C. Platias, M. Papadomanolaki, N. Paragios, K. Karantzalos, 2016. **Simultaneous Registration, Segmentation and Change Detection from Multisensor, Multitemporal Satellite Image Pairs**, IEEE GRSS Data Fusion Contest 2016

12. M. Papadomanolaki, M. Vakalopoulou, S. Zagoruyko, K. Karantzalos, 2016. **Benchmarikng Deep Learning Frameworks for the Classification of Very High Resolution Satellite Multispectral Data**, ISPRS 2016 – XXIII ISPRS Congress, Prague, Chech republic

13. Collobert, R., Kavukcuoglu, K. and Farabet, C., 2011. **Torch7: A Matlab-like Environment for Machine Learning**. In: BigLearn, NIPS Workshop.
14. Christopher M. Bishop, **Pattern Recognition and Machine Learning**

15. Roberto Ierusalimschy, **Programming in Lua – third edition**

16. Stroustrup Bjarne, **The C++ Programming Language**, Fourth Edition

17. Christopher M. Bishop, **Pattern Recognition and Machine Learning**

Ηλεκτονικές Πηγές

1. http://www.gnu.org/philosophy/free-sw.html

2. http://cs231n.github.io/convolutional-networks/

3. http://www.lua.org/

4. https://www.orfeo-toolbox.org/

5. http://www.cplusplus.com/doc/tutorial/

6. http://www.lua.org/manual/5.1/manual.html#3

7. https://github.com/torch/torch7/wiki/Cheatsheet

ΠΑΡΑΡΤΗΜΑ Α

Α1. Πίνακας Εικόνων

Εικόνα 2.1: Προσομοίωση των βιολογικών νευρωνικών δικτύων του εγκεφάλου. Οι κόκκινοι κύκλοι αντιπροσωπεύουν τους νευρώνες και οι μπλε γραμμές τις συνάψεις μέσα από τις οποίες συνδέονται.....6

Εικόνα 2.2: Δομή τεχνητού νευρωνικού δικτύου.....6

Εικόνα 2.3: Τεχνητό νευρωνικό δίκτυο με περισσότερα από ένα κρυφά επίπεδα.....7

Εικόνα 2.4: Εσωτερική λειτουργία τεχνητού νευρώνα.....8

Εικόνα 2.5: Γραφική απεικόνιση των μεταβλητών width,height και depth μιας εικόνας διαστάσεων wxhxd.....10

Εικόνα 2.6: Γραφική απεικόνιση της διαφοράς μεταξύ απλών και συνελικτικών νευρωνικών δικτύων. Οι περιοχές με πορτοκαλί χρωματισμό αντιπροσωπεύουν τα στοιχεία που δέχεται κάθε νευρώνας του επιπέδου εισόδου από μια εικόνα διαστάσεων wxhxd. Αριστερά βρίσκεται η απεικόνιση για τα απλά νευρωνικά δίκτυα και δεξιά για τα συνελικτικά νευρωνικά δίκτυα.....11

Εικόνα 2.7: Οι πληροφορίες που δέχεται το επίπεδο εισόδου ενός απλού νευρωνικού δικτύου από μια εικόνα διαστάσεων wxhxd. Οι περιοχές με πορτοκαλί χρωματισμό αντιπροσωπεύουν τις πληροφορίες με τις οποίες συνδέεται κάθε νευρώνας. Το επίπεδο εισόδου a, αποτελείται από 3 νευρώνες (a_1, a_2 και a_3).....11

Εικόνα 2.8: Οι πληροφορίες που δέχεται το επίπεδο εισόδου ενός συνελικτικού νευρωνικού δικτύου από μια εικόνα διαστάσεων wxhxd. Οι περιοχές με πορτοκαλί χρωματισμό αντιπροσωπεύουν τις πληροφορίες με τις οποίες συνδέεται κάθε νευρώνας.....12

Εικόνα 2.9: Υποπεριοχές της εικόνας διαστάσεων wxhxd στις οποίες αντιστοιχίζονται οι στήλες βάθους(depth-columns) της Εικόνας 2.8.....13

Εικόνα 2.10: Έστω ότι έχουμε μια εικόνα διαστάσεων 12x12 (κάθε μικρό τετράγωνο αντιστοιχεί σε ένα pixel). Στην περιοχή με ροζ χρωματισμό έχει εφαρμοστεί γέμισμα με μηδενικά(zero-padding). Αριστερά, είναι P=1, ενώ δεξιά είναι P=2.....14

Εικόνα 2.11: Έστω ότι για κάποια εικόνα, δίνονται οι διαδοχικές υποπεριοχές 1 και 2. Αν υποθέσουμε ότι F=dx3x3, τότε παρπάνω φαίνεται ο τρόπος με τον οποίο υλοποιείται η μεταβλητή S για τις τιμές 0, 1 και 2. Για S=0 υπάρχει πλήρης επικάλυψη των υποπεριοχών.....14

Εικόνα 2.12: Έστω ότι η εικόνα διαστάσεων 16x16x3(δεξιά) δίνεται σε ένα επίπεδο εισόδου με F=3x3x3 και S=2. Αν το 1 pixel θεωρείται μία χωρική μονάδα, τότε οι

υποπεριοχές 1 και 2 που θα σχηματιστούν θα έχουν την επικάλυψη που φαίνεται με μπλε χρωματισμό.....15

Εικόνα 2.13: Πιθανή μορφή των επιπέδων ενός συνελικτικού νευρωνικού δικτύου.Το επίπεδο ReLU ανήκει στην κατηγορία Transfer Function και εφαρμόζει τη συνάρτηση Rectified Linear Unit.....16

Εικόνα 2.14: Προτεινόμενη μέθοδος ταξινόμησης.....20

Εικόνα 3.1: Δεδομένα Deepsat Παραδείγματα από patches.....25

Εικόνα 3.2: Εικόνα από την ομάδα δεδομένων 'Zurich Summer Dataset v1.0'.....27

Εικόνα 3.3: Groundtruth για την Εικόνα 3.1.....27

Εικόνα 3.4: Περιοχή Vancouver, Ηνωμένες Πολιτείες Εικόνα DEIMOS-2 - 15 Μαρτίου 2015.....29

Εικόνα 3.5: Περιοχή Vancouver, Ηνωμένες Πολιτείες Εικόνα DEIMOS-2 - 15 Μαΐου 2015.....29

Εικόνα 3.6: Περιοχή Vancouver, Ηνωμένες Πολιτείες Εικόνα IRIS (βίντεο) - 17 Ιουλίου 2015.....30

Εικόνα 3.7: Σχηματική απεικόνιση της μορφής του μοντέλου AlexNet για την ομάδα δεδομένων Deepsat.....32

Εικόνα 3.8: Σχηματική απεικόνιση μοντέλου ConvNet για την ομάδα δεδομένων DeepSat.....33

Εικόνα 3.9: Συνάρτηση ConvBNReLU του μοντέλου VGG.....34

Εικόνα 3.10: Σχηματική απεικόνιση μοντέλου VGG.....34

Εικόνα 3.11: Σχηματική απεικόνιση της διαδικασίας εξαγωγής χαρακτηριστικών από την ομάδα δεδομένων Deepsat για την εκπαίδευση του αλγορίθμου SVM.....35

Εικόνα 3.12: Εικονική στοίβα (Virtual Stack).....42

Εικόνα 3.13: Αρχείο CmakeLists.txt.....49

Εικόνα 3.14: Εντολή που πρέπει να δωθεί στο τερματικό για την εκτέλεση της Εφαρμογής 1.....50

Εικόνα 3.15: Εντολή που πρέπει να δωθεί στο τερματικό για την εκτέλεση της Εφαρμογής 2.....51

Εικόνα 3.16: Εντολή που πρέπει να δωθεί στο τερματικό για την εκτέλεση της

Εφαρμογής 3.....53

Εικόνα 4.1: 1η εικόνα ελέγχου.....56

Εικόνα 4.2: 2η εικόνα ελέγχου.....56

Εικόνα 4.3: *Ποιοτική αξιολόγηση του μοντέλου ConvNet με patches διαστάσεων 29x29 για την 1η Εικόνα ελέγχου*.....71

Εικόνα 4.4: *Ποιοτική αξιολόγηση του μοντέλου ConvNet με patches διαστάσεων 29x29 για τη 2η Εικόνα ελέγχου*.....71

Εικόνα 4.5: Χρώματα κλάσεων για τις Εικόνες 4.3 και 4.4.....72

Εικόνα 4.6: Groundtruth της 1ης εικόνας ελέγχου.....72

Εικόνα 4.7: Groundtruth της 2ης εικόνας ελέγχου.....73

Εικόνα 4.8: Χάρτης Ταξινόμησης Εικόνα DEIMOS-2 - 15 Μαρτίου 2015 Μοντέλο ConvNet.....84

Εικόνα 4.9: Χάρτης Ταξινόμησης Εικόνα DEIMOS-2 - 15 Μαΐου 2015 Μοντέλο ConvNet.....84

Εικόνα 4.10: Χάρτης Ταξινόμησης Εικόνα IRIS (βίντεο) – 17 Ιουλίου 2015 Μοντέλο ConvNet.....85

Εικόνα 4.11: Χρώματα κλάσεων για τους χάρτες ταξινόμησης.....85

Α2. Πίνακας Πινάκων

Πίνακας 2.1: Αποτελέσματα όλων των μεθόδων.....18

Πίνακας 2.2: Αποτελέσματα ταξινόμησης του μοντέλου CNN με και χωρίς το DSM.....19

Πίνακας 2.3: Αποτελέσματα ταξινόμησης του μοντέλου CNN με patches διαφόρων διαστάσεων. Το ALL αντιπροσωπεύει το συνδυασμό των τριών διαστάσεων.....19

Πίνακας 2.4: Αποτελέσματα ταξινόμησης από συνδυασμό μεθόδων. Τα αποτελέσματα από το πολυδιάστατο μοντέλο CNN τοποθετούνται για ευκολία.....20

Πίνακας 2.5: Ακρίβειες ταξινομητών. Οι ακρίβειες με έντονο μαύρο είναι οι υψηλότερες που επιτεύχθηκαν.....21

Πίνακας 4.1: Ακρίβειες όλων των μοντέλων ταξινόμησης για την ομάδα δεδομένων.....55

Πίνακας 4.2: Ακρίβειες όλων των μοντέλων ταξινόμησης για την ομάδα δεδομένων SAT-6.....55

Πίνακας 4.3: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 5x5 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 86.8%.....58

Πίνακας 4.4: Πίνακας σύγχησης για τη 2η Εικόνα ελέγχου με patches διαστάσεων 5x5 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 89.5%.....58

Πίνακας 4.5: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 11x11 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 90.3%.....59

Πίνακας 4.6: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 11x11 και 5x5 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 3.5%.....60

Πίνακας 4.7: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 11x11 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 91.1%.....61

Πίνακας 4.8: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 11x11 και 5x5 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 1.6%.....61

Πίνακας 4.9: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 21x21 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 92.4%.....62

Πίνακας 4.10: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 21x21 και 11x11 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 2.1%.....63

Πίνακας 4.11: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 21x21 και 11x11 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 92.7%.....64

Πίνακας 4.12: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 21x21 και 11x11 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 1.6%.....64

Πίνακας 4.13: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 29x29 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.2%.....65

Πίνακας 4.14: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 29x29 και 21x21

Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.8%.....66

Πίνακας 4.15: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 29x29 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.5%.....67

Πίνακας 4.16: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 29x29 και 21x21 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.8%.....67

Πίνακας 4.17: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 33x33 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.2%.....68

Πίνακας 4.18: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 33x33 και 29x29 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.0%.....69

Πίνακας 4.19: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 33x33 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 92.4%.....69

Πίνακας 4.20: Διαφορές ακριβειών μεταξύ των patches διαστάσεων 33x33 και 29x29 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του -1.1%.....70

Πίνακας 4.21: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 29x29 Μοντέλο AlexNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.1%.....74

Πίνακας 4.22: Πίνακας σύγχησης για την 2η Εικόνα ελέγχου με patches διαστάσεων 29x29 Μοντέλο AlexNet*: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 95.1%*.....74

Πίνακας 4.23: Πίνακας σύγχησης για την 1η Εικόνα ελέγχου με patches διαστάσεων 29x29 Μοντέλο VGG: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 95.1%.....75

Πίνακας 4.24: Πίνακας σύγχησης για τη 2η Εικόνα ελέγχου με patches διαστάσεων 29x29 Μοντέλο VGG: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 93.6%.....76

Πίνακας 4.25: Σύγκριση των μοντέλων AlexNet και ConvNet Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου AlexNet για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του -0.1%.....77

Πίνακας 4.26: Σύγκριση των μοντέλων VGG και ConvNet Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 1.9%.....77

Πίνακας 4.27: Σύγκριση των μοντέλων VGG και AlexNet Οι ακρίβειες του μοντέλου AlexNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 2.0%.....78

Πίνακας 4.28: Σύγκριση των μοντέλων AlexNet και ConvNet Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου AlexNet για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του -0.4%.....79

Πίνακας 4.29: Σύγκριση των μοντέλων VGG και ConvNet Οι ακρίβειες του μοντέλου ConvNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.1%.....79

Πίνακας 4.30: Σύγκριση των μοντέλων VGG και AlexNet Οι ακρίβειες του μοντέλου AlexNet έχουν αφαιρεθεί από τις ακρίβειες του μοντέλου VGG για διάσταση ίση με 29x29. Η ποσοτική αξιολόγηση έδωσε διαφορά συνολικής ακρίβειας (Δ(OA)) της τάξης του 0.5%.....80

Πίνακας 4.31: Πίνακας σύγχησης Εικόνα DEIMOS-2 - 15 Μαρτίου 2015 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 76.6%.....81

Πίνακας 4.32: Πίνακας σύγχησης Εικόνα DEIMOS-2 - 15 Μαΐου 2015 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 82.2%.....82

Πίνακας 4.33: Πίνακας σύγχησης Εικόνα IRIS (βίντεο) – 17 Ιουλίου 2015 Μοντέλο ConvNet: Η ποσοτική αξιολόγηση έδωσε συνολική ακρίβεια (OA) της τάξης του 70.34%.....83

Α3. Πίνακας Εξισώσεων

Εξίσωση 3.1: Δείκτης K (Kappa coefficient).....36

Εξίσωση 3.2: Ακρίβεια Pc n_i=άθροισμα στοιχείων του πίνακα συγχησης ανά γραμμή, n_j=άθροισμα στοιχείων του πίνακα συγχησης ανά στήλη, n=Συνολικό άθροισμα στοιχείων του πίνακα σύγχησης.....36

ΠΑΡΑΡΤΗΜΑ Β

Β1. Αρχείο Κώδικα cut.cpp

```
1  //include OTB header files
2  #include "otbImageFileReader.h"
3  #include "otbMultiToMonoChannelExtractROI.h"
4  #include "otbVectorImage.h"
5
6  //include lua header files
7  #include </usr/include/lua5.1/lua.hpp>
8
9  extern "C" {
10  #include "/usr/include/lua5.1/lua.h"
11  #include "/usr/include/lua5.1/lualib.h"
12  #include "/usr/include/lua5.1/lauxlib.h"
13 }
14
15 using namespace std;
16
17 int main(int argc, char * argv[])
18 {
19
20  if (argc != 7)
21  {
22          cout<<"Usage: "<<argv[0]<<"\n";
23          cout<<"Input Variables in the correct order: \n";
24          cout<<"1. inputImageFile \n";
25          cout<<"2. PatchSize \n";
26          cout<<"3. Percentage per class \n";
27          cout<<"4. Training_patches filename \n";
28          cout<<"5. Labels_of_image filename \n";
29          cout<<"6. Saved_Right_Targets \n";
30 return EXIT_FAILURE;
31  }
32
33 //declare input variables
34 const char *patch_size;
35 const char *savedp;
36 const char *labelsfile;
37 const char *perc;
38 const char *righttargets;
39
40 //order of input variables
41 patch_size=argv[2];
42 perc=argv[3];
43 savedp=argv[4];
44 labelsfile=argv[5];
45 righttargets=argv[6];
46
47 //declare lua variable L, which will be used for transfering
       data from C++ to Lua
```

```
48 lua_State *L;
49 L = luaL_newstate();
50 luaL_openlibs(L);
51
52 //define pixeltype of training image
53 typedef unsigned short int PixelType ;
54
55 //create pointer reader to store the input training image
56 typedef otb :: VectorImage < PixelType , 2> VectorImageType;
57 typedef otb :: ImageFileReader < VectorImageType > ReaderType ;
58 ReaderType :: Pointer reader = ReaderType :: New ();
59 reader -> SetFileName ( argv [1]);
60
61 //update reader
62 reader -> Update();
63
64 //extract how many rows, columns and bands the training image
       contains
65 VectorImageType::RegionType largest = reader->GetOutput()->
       GetLargestPossibleRegion();
66 unsigned int rows = largest.GetSize()[1];
67 unsigned int columns = largest.GetSize()[0];
68 unsigned  short int nbBands = reader -> GetOutput()->
       GetNumberOfComponentsPerPixel();
69
70 //create a one-dimensional vector 'vector' used to store the
       pixel values of the input training image.
71 //Heap memory is used because of the large size of the vector
72 unsigned short int* vector = new unsigned short int[rows*columns
       *nbBands];
73
74 //create pointer constIterator used to get all pixel values
75 typedef itk::ImageRegionConstIterator<VectorImageType>
       ConstIteratorType;
76 VectorImageType::ConstPointer inputImage=reader->GetOutput();
77 ConstIteratorType constIterator( inputImage, inputImage->
       GetRequestedRegion() );
78 constIterator.GoToBegin();
79
80 //put all pixel values to vector 'vector'
81 int cnt=1;
82 while (!constIterator.IsAtEnd()) {
83  for (int j=0; j<nbBands; j++) {
84   vector[cnt]=constIterator.Get()[j];
85   cnt=cnt+1;
86  }
87  ++constIterator;
88 }
89
90 //load embeding lua file
91 std::cout << "[C++] Loading the Lua script\n" << std::endl;
92 int status = luaL_loadfile(L, "cut_embed.lua");
93 lua_newtable(L);
94
```

```
95  //push vector 'vector' to stack
96  for (int i=1; i<=rows*columns*nbBands; i++) {
97   lua_pushnumber(L,i);
98   lua_pushnumber(L,vector[i]);
99   lua_settable(L,-3);
100 }
101
102 //push variables to stack
103 lua_pushinteger (L, nbBands);
104 lua_pushstring (L, perc);
105 lua_pushstring (L, patch_size);
106 lua_pushinteger(L,rows);
107 lua_pushinteger(L,columns);
108 lua_pushstring(L,savedp);
109 lua_pushstring(L,labelsfile);
110 lua_pushstring(L,righttargets);
111
112 //define the names used by lua to recognize passing variables
113 lua_setglobal(L,"righttargets");
114 lua_setglobal(L,"labelsfile");
115 lua_setglobal(L,"savedp");
116 lua_setglobal(L,"columns");
117 lua_setglobal(L,"rows");
118 lua_setglobal(L, "patch_size");
119 lua_setglobal(L,"perc");
120 lua_setglobal(L, "nbBands");
121 lua_setglobal(L, "mm");
122
123 //begin process
124 int result = 0;
125 result = lua_pcall(L, 0, LUA_MULTRET, 0);
126
127 //delete vector 'vector' from heap memory
128 delete [] vector;
129
130 }
```

B2. Αρχείο Κώδικα train.cpp

```
1  //include OTB header files
2  #include "otbImageFileReader.h"
3  #include "otbMultiToMonoChannelExtractROI.h"
4  #include "otbVectorImage.h"
5
6  //include lua header files
7  #include </usr/include/lua5.1/lua.hpp>
8
9  extern "C" {
10  #include "/usr/include/lua5.1/lua.h"
11  #include "/usr/include/lua5.1/lualib.h"
12  #include "/usr/include/lua5.1/lauxlib.h"
13 }
14
15 using namespace std;
16
17 int main(int argc, char * argv[])
18 {
19
20  if (argc != 4)
21  {
22   cout<<"Usage: "<<argv[0]<<"\n";
23   cout<<"Input Variables in the correct order: \n";
24   cout<<"1. Training_patches_filename \n";
25         cout<<"2. Corresponding_training_labels_filename \n";
26         cout<<"3. Number_Of_Different_Classes \n";
27 return EXIT_FAILURE;
28  }
29
30 //declare input variables
31 const char *tr_patches;
32 const char *targets;
33 const char *ncl;
34
35 //order of input variables
36 tr_patches=argv[1];
37 targets=argv[2];
38 ncl=argv[3];
39
40 //declare lua variable L, which will be used for transfering
       data from C++ to Lua
41 lua_State *L;
42 L = luaL_newstate();
43 luaL_openlibs(L);
44
45 //load embeding lua file
46 std::cout << "[C++] Loading the Lua script\n" << std::endl;
47 int status = luaL_loadfile(L, "train_embed.lua");
48
49 //push variables to stack
```

```
50 lua_pushstring (L, tr_patches);
51 lua_pushstring(L,targets);
52 lua_pushstring(L,ncl);
53
54 //define the names used by lua to recognize passing variables
55 lua_setglobal(L,"ncl");
56 lua_setglobal(L,"targets");
57 lua_setglobal(L, "tr_patches");
58
59 //begin process
60 int result = 0;
61 result = lua_pcall(L, 0, LUA_MULTRET, 0);
62
63 }
```

B3. Αρχείο Κώδικα testing.cpp

```
1  //include OTB header files
2  #include "otbImageFileReader.h"
3  #include "otbMultiToMonoChannelExtractROI.h"
4  #include "otbVectorImage.h"
5
6  //include lua header files
7  #include </usr/include/lua5.1/lua.hpp>
8
9  extern "C" {
10  #include "/usr/include/lua5.1/lua.h"
11  #include "/usr/include/lua5.1/lualib.h"
12  #include "/usr/include/lua5.1/lauxlib.h"
13 }
14
15 using namespace std;
16
17 int main (int argc , char * argv [])
18 {
19
20   if (argc != 8)
21  {
22          cout<<"Usage: "<<argv[0]<<"\n";
23          cout<<"Input Variables in the correct order: \n";
24          cout<<"1. inputImageFile \n";
25          cout<<"2. PatchSize \n";
26          cout<<"3. Total_Number_of_Training_Classes \n";
27          cout<<"4. Labels_of_image filename \n";
28          cout<<"5. Modelname \n";
29          cout<<"6. mean_name \n";
30          cout<<"7. stdv_name \n";
31 return EXIT_FAILURE;
32  }
33
34 //declare input variables
35 const char* labelsfile;
36 const char* patch_size;
37 const char *modelname;
38 const char *meanname;
39 const char *stdvname;
40 const char *tncl;
41
42 //order of input variables
43 patch_size=argv[2];
44 tncl=argv[3];
45 labelsfile=argv[4];
46 modelname=argv[5];
47 meanname=argv[6];
48 stdvname=argv[7];
49
50 //declare lua variable L, which will be used for transfering
```

```
    data from C++ to Lua
lua_State *L;
L = luaL_newstate();
luaL_openlibs(L);
int status;

//define pixeltype of training image
typedef unsigned short int PixelType ;

//create pointer reader to store the input training image
typedef otb :: VectorImage < PixelType , 2> VectorImageType;
typedef otb :: ImageFileReader < VectorImageType > ReaderType ;
ReaderType :: Pointer reader = ReaderType :: New ();
reader -> SetFileName ( argv [1]);

//update reader
reader -> Update();

//extract how many rows, columns and bands the training image
    contains
VectorImageType::RegionType largest = reader->GetOutput()->
    GetLargestPossibleRegion();
unsigned int rows = largest.GetSize()[1];
unsigned int columns = largest.GetSize()[0];
unsigned  short int nbBands = reader -> GetOutput()->
    GetNumberOfComponentsPerPixel();

//create a one-dimensional vector 'vector' used to store the
    pixel values of the input training image.
//Heap memory is used because of the large size of the vector
unsigned short int* vector = new unsigned short int[rows*columns
    *nbBands];

//create pointer constIterator used to get all pixel values
typedef itk::ImageRegionConstIterator<VectorImageType>
    ConstIteratorType;
VectorImageType::ConstPointer inputImage=reader->GetOutput();
ConstIteratorType constIterator( inputImage, inputImage->
    GetRequestedRegion() );
constIterator.GoToBegin();

//put all pixel values to vector 'vector'
int cnt=1;
while (!constIterator.IsAtEnd()) {
 for (int j=0; j<nbBands; j++) {
  vector[cnt]=constIterator.Get()[j];
  cnt=cnt+1;
 }
 ++constIterator;
}
//load embeding lua file
std::cout << "[C++] Loading the Lua script\n" << std::endl;
status = luaL_loadfile(L, "testing_embed.lua");

```

```
97  //create table and push it to stack
98  lua_newtable(L);
99
100 //push vector 'vector' to stack
101 for (int i=1; i<=rows*columns*nbBands; i++) {
102  lua_pushnumber(L,i);
103  lua_pushnumber(L,vector[i]);
104  lua_settable(L,-3);
105 }
106
107 //push variables to stack
108 lua_pushinteger(L,rows);
109 lua_pushinteger(L,columns);
110 lua_pushinteger(L,nbBands);
111 lua_pushstring(L,patch_size);
112 lua_pushstring(L,labelsfile);
113 lua_pushstring(L,modelname);
114 lua_pushstring(L,meanname);
115 lua_pushstring(L,stdvname);
116 lua_pushstring(L,tncl);
117
118 //define the names used by lua to recognize passing variables
119 lua_setglobal(L,"tncl");
120 lua_setglobal(L,"stdvname");
121 lua_setglobal(L,"meanname");
122 lua_setglobal(L,"modelname");
123 lua_setglobal(L,"labelsfile");
124 lua_setglobal(L,"patch_size");
125 lua_setglobal(L,"nbBands");
126 lua_setglobal(L,"columns");
127 lua_setglobal(L,"rows");
128 lua_setglobal(L,"mm");
129
130 //begin process
131 int result = 0;
132 result = lua_pcall(L, 0, LUA_MULTRET, 0);
133
134 //delete vector 'vector' from heap memory
135 delete [] vector;
136
137 }
```

B4. Εμβόλιμο Αρχείο Κώδικα στο cut.cpp (cut_embed.lua)

```
require 'mattorch'
require 'torch'
require 'xlua'
require 'image'
require 'torchx'
require 'optim'
require 'nn'
require 'cunn'

--load label file of image
a1=mattorch.load(labelsfile)
labels=a1.labels
labels=labels:transpose(1,2)

--get vector 'vector' from cut.cpp file and reshape it to image dimensions
mm=torch.Tensor(mm) -- 'mm' is the name which lua uses to identify the transfered vector from C++ to Lua
pre_image=torch.Tensor(rows,columns,nbBands)
pre_image=mm:view(rows,columns,nbBands)
image=pre_image:transpose(2,3)
image=image:transpose(1,2)
siz1=math.floor(patch_size/2)
print('Image size is : ')
print(image:size())

--find the number that represents each class and store it to Tensor 'classes'
labels_1d=labels:reshape((labels:size(1))*(labels:size(2)))
labels_1d_s=torch.sort(labels_1d)
classes=torch.Tensor(100,1)
classes[1]=labels_1d_s[1]

total_size=1
cnt=2
i=2
value=labels_1d_s[1]
while i~=labels_1d_s:size(1) do
if labels_1d_s[i]~=value then
value=labels_1d_s[i]
classes[cnt]=value
cnt=cnt+1
total_size=total_size+1
end
i=i+1
end

classes=classes[{{1,total_size}}]

if classes[1][1]==0 then
```

```
48 s_cl=classes:size(1)
49 classes=classes[{{2,total_size}}]
50 end
51
52 print('Image classes found: ')
53 print(classes)
54
55 --Training patches are centered to random image pixels. The
       whole region of the patch needs to be inside the image
       borders. In order to achive that, certain rows and columns
       are filled with 999 so that the pixels included in them will
       not be picked by the program.
56 labels[{ {1,siz1},{} }]=999
57 labels[{ {rows-siz1-1,rows},{} }]=999
58 labels[{ {},{1,siz1} }]=999
59 labels[{ {},{columns-siz1-1,columns} }]=999
60
61 howmanycl=torch.Tensor(classes:size(1)) --create a matrix that
       includes how many patches are cut for each class
62 wh_find_perc=torch.Tensor(10000000,1) --tensor used to save the
       locations of the extracted pixels. Size is uknown, so
       10000000 is used for safety reasons
63 right_targets=torch.Tensor(10000000,1) --tensor used to save
       corresponding classes of tensor wh_find_perc
64 total_size=0
65 --start a for loop to extract the appropriate amount of pixels
       for each class
66 for i=1,classes:size(1) do
67 cl=torch.find(labels,classes[i][1])
68 cl=torch.Tensor(cl)
69 --extract the given percentage of pixels for each class
70 accepted=torch.randperm(cl:size(1))[{{1,perc*(cl:size(1))}}]:
       long()
71 perc_cl=cl[{{1,cl:size(1)}}]:index(1,accepted)
72
73 if i==1 then
74 size_prev=perc_cl:size(1)
75 wh_find_perc[{{1,size_prev}}]=perc_cl
76 total_size=total_size+size_prev
77 howmanycl[i]=size_prev
78 right_targets[{{1,size_prev}}]=classes[i][1]
79 end
80 if i~=1 then
81 size_next=perc_cl:size(1)
82 total_size=total_size+size_next
83 wh_find_perc[{{total_size-size_next+1,total_size}}]=perc_cl
84 right_targets[{{total_size-size_next+1,total_size}}]=classes[i
       ][1]
85 howmanycl[i]=size_next
86 end
87
88 end
89
90 wh_find_perc=wh_find_perc[{{1,total_size}}]
```

```
91  right_targets=right_targets[{{1,total_size}}]
92  mattorch.save(righttargets,right_targets)
93
94  for i=1,classes:size(1) do
95  print('Class ' .. classes[i][1] .. ' : ' .. howmanycl[i] .. '
        patches created ')
96  end
97  print(' ')
98
99  wh_find_perc=wh_find_perc[{{1,total_size}}]
100 right_targets=right_targets[{{1,total_size}}]
101
102 xy=torch.Tensor(total_size,2) --tensor used to save the x,y
        positions of the pixels found
103
104 for i=1,total_size do
105 v1=wh_find_perc[i][1]%columns --definition of y position
106 v2=math.floor(wh_find_perc[i][1]/columns) -- definition of x
        position
107 xy[i][1]=v2+1
108 xy[i][2]=v1
109 end
110
111 --create training patches
112 train_patches=torch.Tensor(xy:size(1),nbBands,patch_size,
        patch_size)
113 for i=1,train_patches:size(1) do
114 train_patches[{{i},{},{},{}}]=image[{{},{xy[i][1]-siz1,xy[i][1]+
        siz1},{xy[i][2]-siz1,xy[i][2]+siz1}}]
115
116 end
117 print('Size of training patches:')
118 print(train_patches:size())
119
120 print('Saving training patches..')
121
122 mattorch.save(savedp,train_patches,'-v7.3')
```

B5. Εμβόλιμο Αρχείο Κώδικα στο train.cpp (train_embed.lua)

```
require 'mattorch'
require 'torch'
require 'xlua'
require 'image'
require 'torchx'
require 'optim'
require 'nn'
require 'cunn'
local c = require 'trepl.colorize'

--set training options
batchSize=100
learningRate=1
learningRateDecay=1e-7
weightDecay=0.0005
momentum=0.9
epoch_step=2
max_epoch=15

--load training patches and corresponding target classes
tr_patches1=mattorch.load(tr_patches,'-v7.3')
tr_labels1=mattorch.load(targets,'-v7.3')
tr_patches=tr_patches1.x
tr_labels=tr_labels1.x

--get number of training classes
ncl2=torch.range(1,ncl)
nbcl=ncl2:size(1)

--define patch_size
patch_size=tr_patches:size(3)

if patch_size==5 then
model=dofile('models/conv5x5.lua')
elseif patch_size==11 then
model=dofile('models/conv11x11.lua')
elseif patch_size==21 then
model=dofile('models/conv21x21.lua')
elseif patch_size==29 then
model=dofile('models/conv29x29.lua')
elseif patch_size==33 then
model=dofile('models/conv33x33.lua')
end

--transfer model to cuda
model:cuda()

-- retrieve parameters and gradients
parameters,gradParameters = model:getParameters()

```

```
51 --define confusion matrix dimensions
52 confusion = optim.ConfusionMatrix(nbcl)
53
54 --set training criterion
55 criterion = nn.CrossEntropyCriterion():cuda()
56
57 print('model used:')
58 print(model)
59
60 print('==>' .. ' configuring optimizer')
61 optimState = {
62   learningRate = learningRate,
63   weightDecay = weightDecay,
64   momentum = momentum,
65   learningRateDecay = learningRateDecay,
66 }
67
68 print '<trainer> preprocessing data (color space + normalization
       )'
69 collectgarbage()
70
71 trainData = {
72    data=tr_patches,
73    labels=tr_labels
74 }
75
76 mean = {} -- store the mean, to normalize the test set in the
       future
77 stdv  = {} -- store the standard-deviation for the future
78 for i=1,tr_patches:size(2) do -- over each image channel
79     mean[i] = trainData.data[{ {}, {i}, {}, {}  }]:mean() --
           mean estimation
80     print('Channel ' .. i .. ', Mean: ' .. mean[i])
81     trainData.data[{ {}, {i}, {}, {}  }]:add(-mean[i]) -- mean
           subtraction
82
83     stdv[i] = trainData.data[{ {}, {i}, {}, {}  }]:std() -- std
           estimation
84     print('Channel ' .. i .. ', Standard Deviation: ' .. stdv[i
           ])
85     trainData.data[{ {}, {i}, {}, {}  }]:div(stdv[i]) -- std
           scaling
86 end
87 torch.save('mean.t7',mean)
88 torch.save('stdv.t7',stdv)
89
90
91 -- training function
92 function train()
93   model:training()
94   epoch = epoch or 1
95
96   -- drop learning rate every "epoch_step" epochs
97  if epoch % epoch_step == 0 then learningRate = learningRate/2
```

```
    end
98
99    print(c.blue '==>'.." online epoch # " .. epoch .. ' [
          batchSize = ' .. batchSize .. ']')
100
101   local targets = torch.CudaTensor(batchSize)
102   local indices = torch.randperm(trainData.data:size(1)):long():
          split(batchSize)
103   -- remove last element so that all the batches have equal size
104   indices[#indices] = nil
105
106   local tic = torch.tic()
107   for t,v in ipairs(indices) do
108     xlua.progress(t, #indices)
109
110     local inputs = trainData.data:index(1,v)
111     targets:copy(trainData.labels:index(1,v))
112
113     local feval = function(x)
114       if x ~= parameters then parameters:copy(x) end
115       gradParameters:zero()
116
117       local outputs = model:forward(inputs)
118       local f = criterion:forward(outputs, targets)
119       local df_do = criterion:backward(outputs, targets)
120       model:backward(inputs, df_do)
121       confusion:batchAdd(outputs, targets)
122
123       return f,gradParameters
124     end
125     optim.sgd(feval, parameters, optimState)
126  end
127
128   confusion:updateValids()
129   print(('Train accuracy: '..c.cyan'%.2f'..' %%\t time: %.2f s')
          :format(confusion.totalValid * 100, torch.toc(tic)))
130   train_acc = confusion.totalValid * 100
131   print(confusion)
132   print('Train Accuracy:'..train_acc)
133   confusion:zero()
134   torch.save('trained_model.net',model)
135   epoch = epoch + 1
136 end
137
138
139 for i=1,max_epoch do
140   train()
141 end
```

B6. Εμβόλιμο Αρχείο Κώδικα στο testing.cpp (testing_embed.lua)

```
require 'mattorch'
require 'nn'
require 'cunn'
require 'optim'
require 'torch'
require 'xlua'
require 'image'
require 'torchx'

--get number of all training classes from testing.cpp file
tncl2=torch.range(1,tncl)
tnbcl=tncl2:size(1)

--load trained model
model=torch.load(modelname)
model:cuda()

--load mean and stdv of training data
mean=torch.load(meanname)
stdv=torch.load(stdvname)

--get vector 'vector' from cut.cpp file and reshape it to image dimensions
mm=torch.Tensor(mm) -- 'mm' is the name of the vector which was transfered from C++ to Lua
pre_image=torch.Tensor(rows,columns,nbBands)
pre_image=mm:view(rows,columns,nbBands)
image=pre_image:transpose(2,3)
image=image:transpose(1,2)
print('Image size is :')
print(image:size())
siz1=math.floor(patch_size/2)

--load labels of testing image
a1=mattorch.load(labelsfile)
labels=a1.labels
labels=labels:transpose(1,2)
print('Labels size is :')
print(labels:size())

--create tensor for storing the final predicted labels
final_labels=torch.Tensor(labels:size(1),labels:size(2))
final_labels[{ {1,siz1},{} }]=0
final_labels[{ {rows-siz1-1,rows},{} }]=0
final_labels[{ {},{1,siz1} }]=0
final_labels[{ {},{columns-siz1-1,columns} }]=0

--create tensor for storing the final predicted class scores
final_scores=torch.Tensor(labels:size(1),labels:size(2),tnbcl)
final_scores[{ {1,siz1},{},{} }]=0
```

```
49 final_scores[{ {rows-siz1-1,rows},{},{} }]=0
50 final_scores[{ {},{1,siz1},{} }]=0
51 final_scores[{ {},{columns-siz1-1,columns,{}} }]=0
52
53 --Testing patches are going to be centered to all image pixels.
       The whole region of the patch needs to be inside the image
       borders. In order to achive that, certain rows and columns
       are excluded.
54 labels_new=labels[{ {siz1+1,rows-siz1},{siz1+1,columns-siz1} }]
55
56 new_rows=labels_new:size(1)
57 new_columns=labels_new:size(2)
58
59 --reshape labels tensor to 1-D
60 labels_1d=labels_new:reshape(new_rows*new_columns)
61
62 --cut testing patches
63 test_patches=torch.Tensor(new_rows*new_columns,nbBands,
       patch_size,patch_size);
64 cnt=0
65 for i=1+siz1,rows-siz1 do
66
67  for j=1+siz1,columns-siz1 do
68    test_patches[{{cnt+j-siz1},{},{},{}}]=image[{{},{i-siz1,i+
         siz1},{j-siz1,j+siz1}}]
69  end
70 cnt=cnt+columns-2*siz1
71 end
72
73 --organize test data for convenience
74 labs=torch.Tensor(labels_1d:size(1),1)
75 for i=1,labels_1d:size(1) do
76 labs[i]=labels_1d[i]
77 end
78
79 testData = {
80    data = test_patches,
81    labels = labs,
82 }
83
84 --preprocessing testing patches
85 for i=1,test_patches:size(2) do -- over each image channel
86     testData.data[{ {}, {i}, {}, {}  }]:add(-mean[i]) -- mean
         subtraction
87     testData.data[{ {}, {i}, {}, {}  }]:div(stdv[i]) -- std
         scaling
88 end
89
90 --set confusion matrix dimensions
91 confusion = optim.ConfusionMatrix(tnbcl)
92
93 idx=torch.DoubleTensor(testData.data:size(1),1)
94 klaseis=torch.DoubleTensor(testData.data:size(1),1)
95
```

```
96  print('testing ....')
97  scores=torch.Tensor(new_rows*new_columns,tnbcl)
98    for i=1,testData.data:size(1) do
99       outputs = model:forward(testData.data[{ {i}, {}, {}, {}
          }])
100      outputs=outputs:double()
101      scores[{ {i},{} }]=outputs
102      meg=outputs[1]:max()
103        for u=1,tnbcl do
104          if outputs[1][u]==meg then idx[i]=u end
105        end
106      confusion:add(idx[i][1],testData.labels[i][1])
107      klaseis[i]=idx[i]
108
109   end
110
111 --reshape final labels and scores and save it to files
112 prefinal_scores=scores:reshape(new_rows,new_columns,tnbcl)
113 final_scores[{ {siz1+1,rows-siz1},{siz1+1,columns-siz1},{} }]=
      prefinal_scores
114 final_klaseis=klaseis:reshape(new_rows,new_columns)
115 final_labels[{ {siz1+1,rows-siz1},{siz1+1,columns-siz1} }]=
      final_klaseis
116 final_labels=final_labels:transpose(1,2)
117 final_scores=final_scores:transpose(1,3)
118 final_scores=final_scores:transpose(2,3)
119 mattorch.save('final_labels.mat',final_labels)
120 mattorch.save('final_scores.mat',final_scores)
121
122 --print confusion matrix
123 confusion:updateValids()
124 print(confusion)
```

B7. Φάκελος 'models'

B7.1 Μοντέλο ConvNet για patches διαστάσεων 5x5 (conv5x5.lua)

```
1  ----------model----------
2
3
4  model = nn.Sequential()
5  model:add(nn.Copy('torch.DoubleTensor','torch.CudaTensor'))
6        ------------------------------------
7        -- convolutional network
8        ------------------------------------
9        -- stage 1 : mean suppresion -> filter bank -> squashing
             -> max pooling
10       model:add(nn.SpatialConvolutionMM(4, 32, 3,3,1,1,1,1)) --
             5->5
11       model:add(nn.Tanh())
12       model:add(nn.SpatialMaxPooling(3, 3, 2,2)) -- 5->2
13       -- stage 2 : mean suppresion -> filter bank -> squashing
             -> max pooling
14       model:add(nn.SpatialConvolutionMM(32, 64, 3,3,1,1,1,1)) --
              2->2
15       model:add(nn.Tanh())
16       model:add(nn.SpatialMaxPooling(2, 2, 2, 2)) -- 2->1
17       -- stage 3 : standard 2-layer MLP:
18       model:add(nn.Reshape(64))
19       model:add(nn.Linear(64, 30))
20       model:add(nn.Tanh())
21       model:add(nn.Linear(30, nbcl))
22
23
24 return model
```

B7.2 Μοντέλο ConvNet για patches διαστάσεων 11x11 (conv11x11.lua)

```
1  ----------model----------
2
3
4  model = nn.Sequential()
5  model:add(nn.Copy('torch.DoubleTensor','torch.CudaTensor'))
6        ------------------------------------
7        -- convolutional network
8        ------------------------------------
9        -- stage 1 : mean suppresion -> filter bank -> squashing
             -> max pooling
10       model:add(nn.SpatialConvolutionMM(4, 32, 3,3,1,1,1,1)) --
             11->11
```

```
11        model:add(nn.Tanh())
12        model:add(nn.SpatialMaxPooling(3, 3, 2,2)) -- 11->5
13        -- stage 2 : mean suppresion -> filter bank -> squashing
              -> max pooling
14        model:add(nn.SpatialConvolutionMM(32, 64, 3,3,1,1,1,1)) --
              5->5
15        model:add(nn.Tanh())
16        model:add(nn.SpatialMaxPooling(3, 3, 2, 2)) -- 5->2
17        -- stage 3 : standard 2-layer MLP:
18        model:add(nn.Reshape(256))
19        model:add(nn.Linear(256, 200))
20        model:add(nn.Tanh())
21        model:add(nn.Linear(200, ncl))
22
23  return model
```

B7.3 Μοντέλο ConvNet για patches διαστάσεων 21x21 (conv21x21.lua)

```
1   ----------model----------
2
3
4   model = nn.Sequential()
5   model:add(nn.Copy('torch.DoubleTensor','torch.CudaTensor'))
6         ----------------------------------
7         -- convolutional network
8         ----------------------------------
9         -- stage 1 : mean suppresion -> filter bank -> squashing
              -> max pooling
10        model:add(nn.SpatialConvolutionMM(4, 32, 5, 5)) -- 21->17
11        model:add(nn.Tanh())
12        model:add(nn.SpatialMaxPooling(3, 3, 2,2)) -- 17->8
13        -- stage 2 : mean suppresion -> filter bank -> squashing
              -> max pooling
14        model:add(nn.SpatialConvolutionMM(32, 64, 5, 5)) -- 8->4
15        model:add(nn.Tanh())
16        model:add(nn.SpatialMaxPooling(2, 2, 2, 2)) -- 4->2
17        -- stage 3 : standard 2-layer MLP:
18        model:add(nn.Reshape(64*2*2))
19        model:add(nn.Linear(64*2*2, 200))
20        model:add(nn.Tanh())
21        model:add(nn.Linear(200, ncl))
22
23  return model
```

B7.4 Μοντέλο ConvNet για patches διαστάσεων 29x29 (conv29x29.lua)

```
1  ----------model----------
2
3
4  model = nn.Sequential()
5  model:add(nn.Copy('torch.DoubleTensor','torch.CudaTensor'))
6        ------------------------------------
7        -- convolutional network
8        ------------------------------------
9        -- stage 1 : mean suppresion -> filter bank -> squashing
             -> max pooling
10       model:add(nn.SpatialConvolutionMM(4, 32, 5,5)) -- 29->25
11       model:add(nn.Tanh())
12       model:add(nn.SpatialMaxPooling(3, 3, 2,2)) -- 25->12
13       -- stage 2 : mean suppresion -> filter bank -> squashing
             -> max pooling
14       model:add(nn.SpatialConvolutionMM(32, 64, 5,5)) -- 12->8
15       model:add(nn.Tanh())
16       model:add(nn.SpatialMaxPooling(2, 2, 2, 2)) -- 8->4
17       -- stage 3 : standard 2-layer MLP:
18       model:add(nn.Reshape(1024))
19       model:add(nn.Linear(1024, 500))
20       model:add(nn.Tanh())
21       model:add(nn.Linear(500, ncl))
22
23 return model
```

B7.5 Μοντέλο ConvNet για patches διαστάσεων 33x33 (conv33x33.lua)

```
1  ----------model----------
2
3
4  model = nn.Sequential()
5  model:add(nn.Copy('torch.DoubleTensor','torch.CudaTensor'))
6        ------------------------------------
7        -- convolutional network
8        ------------------------------------
9        -- stage 1 : mean suppresion -> filter bank -> squashing
             -> max pooling
10       model:add(nn.SpatialConvolutionMM(4, 32, 5,5)) -- 33->29
11       model:add(nn.Tanh())
12       model:add(nn.SpatialMaxPooling(3, 3, 2,2)) -- 29->14
13       -- stage 2 : mean suppresion -> filter bank -> squashing
             -> max pooling
14       model:add(nn.SpatialConvolutionMM(32, 64, 5,5)) -- 14->10
15       model:add(nn.Tanh())
```

```
16          model:add(nn.SpatialMaxPooling(2, 2, 2, 2)) -- 10->5
17          -- stage 3 : standard 2-layer MLP:
18          model:add(nn.Reshape(1600))
19          model:add(nn.Linear(1600, 700))
20          model:add(nn.Tanh())
21          model:add(nn.Linear(700, ncl))
22
23  return model
```