



Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Πολυτεχνική Σχολή
Τμήμα Ηλεκτρολόγων Μηχανικών &
Μηχανικών Γπολογιστών
Τομέας Ηλεκτρονικής και Γπολογιστών

Διπλωματική Εργασία

Εφαρμογή των CNN για Αναγνώριση και
Εντοπισμό Αντικειμένων & Υλοποίηση στο
Ενσωματωμένο Σύστημα Jetson-TK1

Εκπόνηση:
Παναγιώτου Κωνσταντίνος
ΑΕΜ: 7316

Επίβλεψη:
Αν. Καθ. Πέτρου Λουκάς
Υπ. Δρ. Μουσουλιώτης
Παναγιώτης

The robots are going to help us find our crystal...

Είμαστε κάτι διάχυτες αισθήσεις,
χωρίς ελπίδα να συγκεντρωθούμε.
Στα νεύρα μας μπερδεύεται όλη η φύσις
— Κώστας Καρυωτάκης

ΕΥΧΑΡΙΣΤΙΕΣ

Thank you! =]

Περίληψη

Η Μηχανική Μάθηση αποτελεί έναν από τους σημαντικότερους τομείς των τελευταίων ετών εξαιτίας της ανάπτυξης των Τεχνιτών Νευρωνικών Δικτύων. Εμπνευσμένα από τον τρόπο που λειτουργεί ο Εγκέφαλος και το Κεντρικό Νευρικό Σύστημα (ΚΝΣ) του ανθρώπου, αυτά τα υπολογιστικά μοντέλα ξεπερνούν σε απόδοση προηγούμενες μορφές Τεχνητής Νοημοσύνης σε διαδικασίες Μηχανικής Μάθησης. Τα Νευρωνικά Δίκτυα Συνέλιξης (CNNs) αποτελούν μία από τις πιο ενδιαφέρουσες μορφές αρχιτεκτονικής Νευρωνικών Δικτύων, τα οποία χρησιμοποιούνται για την επίλυση προβλημάτων αναγνώρισης προτύπων στην Μηχανική Όραση (CV). Τα CNNs ανήκουν στην κατηγορία των αλγορίθμων εκμάθησης αναπαραστάσεων, που σημαίνει ότι δεν απαιτείται η εξαγωγή χειρόγραφων (από τον άνθρωπο) χαρακτηριστικών για την αναγνώριση των προτύπων, ενισχύοντας έτσι τις μηχανές με την ικανότητα να εξάγουν μόνες τους τα κατάλληλα χαρακτηριστικά από δοσμένα εκ των προτέρων δεδομένα.

Τα ενσωματωμένα συστήματα έχουν περιορισμένη υπολογιστική ικανότητα σε σύγκριση με τα desktop και server συστήματα. Ωστόσο είναι σχεδιασμένα για εφαρμογές με χαμηλή ενεργειακή απαίτηση, όπως είναι για παράδειγμα οι ρομποτικές εφαρμογές. Τα ρομπότ χρειάζεται να είναι εφοδιασμένα με την κατάλληλη επεξεργαστική ισχύ που να τους επιτρέπει να πραγματοποιήσουν βασικές ενέργειες, όπως είναι η πλοϊγή ση στο χώρο, η χαρτογράφηση και η αναγνώριση αντικειμένων χωρίς παράλληλα να καταναλώνουν τόση ενέργεια που να εξαντλεί το σύστημα.

Η παρούσα διπλωματική εργασία εστιάζει στην ανάπτυξη Νευρωνικών Δικτύων Συνέλιξης για την επίλυση θεμάτων αναγνώρισης και εντοπισμού αντικειμένων στο ενσωματωμένο σύστημα Jetson TK1. Ερευνώνται διάφορα μοντέλα CNNs όπως είναι τα AlexNet, VGG16 και YOLO. Για κάθε CNN γίνεται μέτρηση της μνήμης (RAM), του χρόνου εκτέλεσης και της κατανάλωσης ισχύος που απαιτείται με σκοπό να αξιολογηθεί η αποτελεσματικότητα τους. Τα αποτελέσματα αυτών των μετρήσεων συγκρίνονται με τα αντίστοιχα εξαχθέντα από εκτέλεση σε επεξεργαστή Intel-i7-6700.

Επιπρόσθετα, παρουσιάζεται μια σειρά από εργαλεία για την μεγιστοποίηση του λόγου της απόδοσης ανά μονάδα κατανάλωσης ισχύος (performance per watt) στο Jetson TK1. Παράλληλα, περιγράφονται οι απαραίτητες ρυθμίσεις και διαδικασίες εγκατάστασης.

Σύμφωνα με τα αποτελέσματα, είναι ιδιαίτερα σημαντική η επιλογή κατάλληλου λογισμικού για την ανάπτυξη και υλοποίηση αρχιτεκτονικών CNN. Με κατάλληλη επιλογή λογισμικού και μια σειρά από διαδικασίες βελτιστοποίησης, η πλατφόρμα Jetson TK1 είναι ικανή να προσφέρει λύσεις στο πρόβλημα της αναγνώρισης και εντοπισμού αντικειμένων με χρήση CNNs και ως αυτού θεωρείται κατάλληλη για εφαρμογές ρομποτικής.

Title

CNN Applications Towards Object Detection on Jetson TK1

Abstract

Machine Learning has become crucial in recent times due to the evolution of Artificial Neural Networks (ANN). Inspired by how brain networks function, these computational models outperform previous forms of Artificial Intelligence techniques in common machine learning tasks. Convolutional Neural Networks (CNNs) is one of the most fascinating forms of the ANN architecture, used to solve pattern recognition tasks in Computer Vision (CV). CNN is a form of representation learning algorithm, meaning that no hand-written features are required, enhancing machines with the ability to extract proper features on their own, given a specific set of data.

Embedded systems have limited computational power compared to desktop and server systems. On the other hand, embedded systems are being designed for applications where low power consumption is a requirement, for example in robotics. A robot must be provided with enough computational power to perform basic tasks, like navigation, localization, mapping and object identification without consuming too much power and lead the system to exhaustion.

The present Diploma Thesis focuses on implementations of CNN models for solving object recognition and/or localization tasks on the Jetson TK1 embedded system. Several CNN models have been investigated like AlexNet, VGG16 and YOLO. For each CNN model, the required memory (RAM), execution time and power consumption rates have been measured in order to evaluate their efficiency. These rates have been compared with those collected from a PC with an Intel-i7-6700 processor.

In addition, a set of software tools used for maximizing performance per power consumption ratio (performance/watt) is presented, along with proper configuration and setup procedures for the Nvidia Jetson TK1 embedded system.

According to the outcome, selection of proper software for the implementation and deployment of CNN models is critical. With proper software and a series of optimization steps, the Jetson TK1 platform is able to deliver solutions to generic object detection tasks using CNNs, thus it is considered to be appropriate for utilization in robotic applications.

Konstantinos Panayiotou
Electrical & Computer Engineering Department,
Aristotle University of Thessaloniki, Greece
September 2016

Περιεχόμενα

| | |
|-------------------------------------------------------------------------------------------------------|-----------|
| Ευχαριστίες | iii |
| Περίληψη | v |
| Abstract | vii |
| Ακρωνύμια | xv |
| 1 Εισαγωγή | 1 |
| 1.1 Περιγραφή του Προβλήματος | 2 |
| 1.2 Σκοπός - Συνεισφορά της Διπλωματικής Εργασίας | 3 |
| 1.3 Διάρθρωση της Αναφοράς | 3 |
| 2 Τεχνικές Βαθιάς Μηχανικής Μάθησης και Αναγνώρισης Αντικειμένων στον χώρο | 5 |
| 2.1 Εισαγωγή στην Επιστήμη της Σύγχρονης Μηχανικής Μάθησης | 6 |
| 2.2 Νευρωνικά Δίκτυα με Βάθος | 14 |
| 2.2.1 Συναρτήσεις Ενεργοποίησης | 19 |
| 2.2.2 Συναρτήσεις Σφάλματος/Κόστους | 21 |
| 2.2.3 Αλγόριθμος Backpropagation | 22 |
| 2.3 Νευρωνικά Δίκτυα Συνέλιξης | 24 |
| 2.3.1 Επίπεδο Συνέλιξης | 25 |
| 2.3.2 Επίπεδο Υπό-δειγματοληψίας - Pooling layer | 29 |
| 2.3.3 Πλήρως Συνδεδεμένο Επίπεδο - Fully-connected layer | 30 |
| 2.4 Επισκόπηση Τεχνικών Βαθιάς Μηχανικής Μάθησης στην Αναγνώριση και Εντοπισμό Αντικειμένων | 30 |
| 3 Εργαλεία Hardware και Software | 33 |
| 3.1 NVIDIA Jetson TK1 development board | 33 |
| 3.2 Εργαλεία Λογισμικού | 37 |
| 4 Ύλοποιήσεις | 41 |
| 4.1 Μοντέλα CNN | 41 |
| 4.1.1 AlexNet | 43 |
| 4.1.2 VGG16 | 46 |
| 4.1.3 YOLO Net | 48 |
| 4.2 Βελτιστοποίηση των εργαλείων λογισμικού στο Jetson TK1 | 52 |
| 5 Πειράματα - Αποτελέσματα | 55 |
| 5.1 Πρώτη Φάση Πειραμάτων | 55 |
| 5.1.1 AlexNet | 56 |

ΠΕΡΙΕΧΟΜΕΝΑ

| | | |
|----------|-----------------------------------|-----------|
| 5.1.2 | VGG16 | 57 |
| 5.1.3 | Tiny-YOLO | 58 |
| 5.2 | Δεύτερη Φάση Πειραμάτων | 59 |
| 5.2.1 | AlexNet | 60 |
| 5.2.2 | VGG16 | 61 |
| 5.2.3 | Tiny-YOLO | 63 |
| 5.3 | Συγκριτικά Αποτελέσματα | 65 |
| 6 | Συμπεράσματα | 66 |
| 6.1 | Γενικά Συμπεράσματα | 66 |
| 6.2 | Προβλήματα | 67 |
| 7 | Μελλοντικές επεκτάσεις | 68 |
| | Βιβλιογραφία | 70 |

Κατάλογος Σχημάτων

| | | |
|------|-----------------------------------------------------------------------------------------------------------------------------|----|
| 2.1 | Κλάδοι και εφαρμογές της επιστήμης της Τεχνητής Νοημοσύνης | 6 |
| 2.2 | Διάγραμμα Venn των διαφόρων κατηγοριών μηχανικής μάθησης | 8 |
| 2.3 | Παράδειγμα διαφορετικών αναπαραστάσεων των δεδομένων | 12 |
| 2.4 | Απλό μοντέλο Autoncoder με ένα χρυφό επίπεδο | 12 |
| 2.5 | Παράδειγμα απεικόνισης των φίλτρων ενός μοντέλου CNN για αναγνώριση προσώπου | 13 |
| 2.6 | Υπερσύνολα του κλάδου της βαθιάς μηχανικής μάθησης - Διάγραμμα Venn | 14 |
| 2.7 | Βιολογικός Νευρώνας | 15 |
| 2.8 | Μαθηματικό μοντέλο του νευρώνα | 16 |
| 2.9 | Γλοποίηση πύλης AND με χρήση του μαθηματικού μοντέλου του νευρώνα | 17 |
| 2.10 | Απλό μοντέλο NN με ένα χρυφό επίπεδο - Perceptron | 18 |
| 2.11 | Μορφή ενός πολυεπίπεδου νευρωνικού δικτύου | 18 |
| 2.12 | Συνάρτηση Σιγμοειδούς συνάρτησης | 19 |
| 2.13 | Συνάρτηση Υπερβολικής Εφαπτωμένης | 20 |
| 2.14 | Συνάρτηση Rectified Linear Unit - ReLU | 20 |
| 2.15 | Συνάρτηση Leaky ReLU | 21 |
| 2.16 | Συνάρτηση Maxout | 21 |
| 2.17 | Διαδικασία Backward propagation | 23 |
| 2.18 | Τρισδιάστατη κατανομή των νευρώνων στα CNN | 25 |
| 2.19 | Παράδειγμα διασύνδεσης τρισδιάστατης εισόδου με την τρισδιάστατη δομή των νευρώνων ενός επιπέδου συνέλιξης (CONV) | 26 |
| 2.20 | Συνέλιξη φίλτρου ενός επιπέδου συνέλιξης με τον όγκο εισόδου και παραγωγή ενός χάρτη ενεργοποίησης | 26 |
| 2.21 | Διαδικασία Συνέλιξης | 27 |
| 2.22 | Αντιστοιχία του αριθμού των φίλτρων ενός επιπέδου συνέλιξης με το βάθος του όγκου στην έξοδο | 27 |
| 2.23 | Διαδοχικές εφαρμογές του τελεστή συνέλιξης προκαλούν μείωση των διαστάσεων μήκους και πλάτους του όγκου | 28 |
| 2.24 | Zero Padding | 28 |
| 2.25 | Επίπεδο Υποδειγματοληφίας - Pooling layer | 29 |
| 2.26 | Συνάρτηση υπό-δειγματοληφίας Max - Max Pooling | 30 |
| 3.1 | Tegra K1 SOC | 34 |
| 3.2 | Jetson TK1 development board | 34 |
| 3.3 | ARM Cortex-A15 processor | 36 |

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

| | |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 3.4 Παράδειγμα αναπαράστασης ενός μοντέλου νευρωνικού δικτύου σε γράφο. Το NN αποτλείται από ένα χρυφό επίπεδο και ένα ταξινομητή Softmax στο επίπεδο εξόδου | 38 |
| 4.1 Μοντέλο δικτύου AlexNet | 43 |
| 4.2 Πλήρης μορφή του δικτύου AlexNet της υλοποίησης | 46 |
| 4.3 Μορφή του δικτύου VGG16 | 47 |
| 4.4 Πλήρης μορφή του δικτύου VGG16 της υλοποίησης | 48 |
| 4.5 Παράδειγμα τρόπου λειτουργίας του δικτύου YOLO | 49 |
| 4.6 Πλήρης μορφή του δικτύου Tiny-YOLO της υλοποίησης | 52 |
| 5.1 Χρόνοι εκτέλεσης για το δίκτυο AlexNet σε επεξεργαστή i7 | 57 |
| 5.2 Χρόνοι εκτέλεσης για το δίκτυο VGG16 σε επεξεργαστή i7 | 58 |
| 5.3 Χρόνοι εκτέλεσης για το δίκτυο Tiny-YOLO σε επεξεργαστή i7 | 59 |
| 5.4 Χρόνοι εκτέλεσης για το δίκτυο AlexNet στο Jetson TK1 | 61 |
| 5.5 Χρόνοι εκτέλεσης για το δίκτυο VGG16 στο Jetson TK1 | 62 |
| 5.6 Χρόνοι εκτέλεσης για το δίκτυο Tiny-YOLO στο Jetson TK1 | 63 |

Κατάλογος πινάκων

| | | |
|-----|------------------------------------------------------------------------------------------------------------------|----|
| 4.1 | Χαρακτηριστικά και παράμετροι των επιπέδων του δικτύου AlexNet | 43 |
| 4.2 | Χαρακτηριστικά και παράμετροι των επιπέδων του δικτύου VGG16 | 47 |
| 4.3 | Σύγκριση χρόνου εκτέλεσης βασικών πράξεων γραμμικής άλγεβρας μεταξύ των βιβλιοθηκών ATLAS και OpenBLAS | 54 |
| 5.1 | Μετρήσεις πειραμάτων για το δίκτυο AlexNet σε επεξεργαστή Intel-i7-6700 | 56 |
| 5.2 | Μετρήσεις πειραμάτων για το δίκτυο VGG16 σε επεξεργαστή Intel-i7-6700 | 57 |
| 5.3 | Μετρήσεις πειραμάτων για το δίκτυο Tiny-YOLO σε επεξεργαστή Intel-i7-6700 | 58 |
| 5.4 | Μετρήσεις πειραμάτων για το δίκτυο AlexNet στο Jetson TK1 | 61 |
| 5.5 | Μετρήσεις πειραμάτων για το δίκτυο VGG16 στο Jetson TK1 | 62 |
| 5.6 | Μετρήσεις πειραμάτων για το δίκτυο Tiny-YOLO στο Jetson TK1 | 63 |
| 5.7 | Συγκριτικά αποτελέσματα των πειραμάτων σε επεξεργαστή Intel i7-6700 και Jetson TK1 | 65 |

Κατάλογος Αλγορίθμων

| | |
|-----------------------------------------------------------------------------------------|----|
| 2.1 Αλγόριθμος Feedforward για τον υπολογισμό των εξόδων ενός επιπέδου του NN | 19 |
| 2.2 Backpropagation learning algorithm | 24 |

Ακρωνύμια Εγγράφου

Παρακάτω παρατίθενται ορισμένα από τα πιο συχνά χρησιμοποιούμενα ακρωνύμια της παρούσας διπλωματικής εργασίας:

| | |
|-------|--------------------------------|
| AI | → Artificial Intelligence |
| ML | → Machine Learning |
| NN | → Neural Network |
| ANN | → Artificial Neural Network |
| DL | → Deep Learning |
| DNN | → Deep Neural Network |
| MLP | → Multilayer Perceptron |
| CNN | → Convolutional Neural Network |
| YOLO | → You Only Look Once |
| SOC | → System On Chip |
| OS | → Operating System |
| CPU | → Central Processing Unit |
| GPU | → Graphics Processing Unit |
| GPGPU | → General Purpose GPU |

1

Εισαγωγή

Ο όρος *Τεχνητή Νοημοσύνη* αναφέρεται στην ικανότητα των υπολογιστικών συστημάτων να μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς. Η επιθυμία των ανθρώπων να κατασκευάσουν “έξυπνες” μηχανές, καταγράφεται από την εποχή της αρχαίας Ελλάδας. Μυθικές μορφές όπως οι Πυγμαλίωνας, Δαίδαλος και Ήφαιστος μπορούν να θεωρηθούν ως εφευρέτες και δημιουργοί νοούντων μηχανών ανάμεσα σε άλλες την Γαλάτεια, τον Τάλω και την Πανδώρα. Ένας πιο ολοκληρωμένος ορισμός της Τεχνητής Νοημοσύνης είναι [1]:

Ο κλάδος/τομέας της επιστήμης της πληροφορικής, που ασχολείται με την σχεδίαση και κατασκευή ευφυών συστημάτων, δηλαδή συστημάτων που διαθέτουν χαρακτηριστικά που σχετίζονται με την ανθρώπινη νοημοσύνη και συμπεριφορά.

Με την εμφάνιση των πρώτων ηλεκτρονικών και (επανα)προγραμματιζόμενων υπολογιστικών συστημάτων, οι άνθρωποι ξεκίνησαν να σκέφτονται τρόπους για να κατασκευάσουν “έξυπνες” μηχανές. Η ραγδαία εξέλιξη στον κλάδο της επιστήμης της πληροφορικής τις τελευταίες δεκαετίες, έφερε και την εξέλιξη στην επιστήμη της Τεχνητής Νοημοσύνης. Το 1997, η IBM κατασκεύασε ένα υπολογιστικό σύστημα που μπορούσε να παίξει σκάκι (Deep Blue) [2], το οποίο κέρδισε τον παγκόσμιο τότε πρωταθλητή στο σκάκι, Garry Kasparov. Το σκάκι έχει εξήντα τέσσερις θέσεις και τριάντα δύο πιόνια που μπορούν να κινούνται με συγκεκριμένο τρόπο. Η μηχανή Deep Blue, μπορούσε να εκτιμήσει και να αξιολογήσει διακόσια εκατομμύρια πιθανές καταστάσεις της σκακιέρας. Ωστόσο, πρέπει να σημειωθεί ότι η επίλυση του προβλήματος του σκακιού, παρ’ ότι είναι ένα πρόβλημα το οποίο μπορεί να περιγραφεί πλήρως μέσα από μια λίστα με κανόνες, δεν είναι δυνατό να λυθεί αναλυτικά εξαιτίας του τεράστιου αριθμού των δυνατών κινήσεων.

Τα ρομποτικά συστήματα ήδη έχουν περάσει φάση δοκιμής, σε διάφορες χώρες ανά τον κόσμο, όσον αφορά την ένταξή τους στην καθημερινότητα του ανθρώπου. Η “δύναμη” της τεχνητής νοημοσύνης υπερβαίνει τα όρια της φαντασίας.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

Κύριο χαρακτηριστικό ενός “ευφυούς” ρομποτικού συστήματος είναι να αντιλαμβάνεται το περιβάλλον του. Ένας ρομποτικός πράκτορας (robotic agent) για να πραγματοποιήσει μία πληθώρα εφαρμογών, πρέπει να διαθέτει τόσο την ικανότητα να αναγνωρίζει, μέσω εικόνων λήψης από κάμερα, διάφορα αντικείμενα, όσο και την ικανότητα να εντοπίζει επακριβώς την θέση του εκάστοτε αντικειμένου στον χώρο. Δεν θα είχε νόημα για ένα ρομπότ να έχει άποψη για την παρουσία αντικειμένων γύρω του, αν δεν έχει και την ικανότητα να γνωρίζει την θέση των αντικειμένων αυτών.

Ένας σημαντικός παράγοντας της ραγδαίας εξέλιξης της επιστήμης της ρομποτικής, αλλά και της τεχνητής νοημοσύνης γενικότερα, είναι η εμφάνιση και εξέλιξη του κλάδου της *Βαθιάς Μηχανικής Μάθησης* (Deep learning - DL).

Η χρήση τεχνικών βαθιάς μάθησης στην επίλυση προβλημάτων *Μηχανικής Όρασης*, έχει φέρει την συγκεκριμένη τεχνολογία σε θέση να μπορεί να αντιμετωπίσει περίπλοκα προβλήματα τα οποία μέχρι και πριν από λίγα χρόνια θεωρείτο ακατόρθωτο να λυθούν. Ένα χαρακτηριστικό παράδειγμα είναι τα αυτόνομα αυτοκίνητα, τα οποία σήμερα είναι σε φάση δοκιμής.

Σήμερα, το γενικότερο πρόβλημα της ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες λύνεται με την χρήση *Νευρωνικών Δικτύων Συνέλιξης* (Convolutional Neural Networks - CNNs). Το γεγονός ότι τα CNNs, σε συνδυασμό με μονάδες GPU, δίνουν την δυνατότητα επίλυσης προβλημάτων αναγνώρισης και εντοπισμού αντικειμένων σε μικρό χρόνο, τα καθιστά ικανά να χρησιμοποιηθούν σε εφαρμογές πραγματικού χρόνου.

1.1 ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Παρόλο που τα CNN είναι ικανά να δώσουν λύσεις με μεγάλη ακρίβεια, απαιτούν μεγάλη επεξεργαστική ισχύ τόσο για την εκπαίδευσή τους όσο και για την εκτέλεση ενός πειράματος, ιδιαίτερα όταν το πρόβλημα για το οποίο έχουν σχεδιαστεί να δώσουν λύση είναι περίπλοκο. Η ανάγκη μεγάλης επεξεργαστικής ισχύος οφείλεται στο μεγάλο αριθμό επιπέδων από τα οποία αποτελούνται τα μοντέλα CNN. Για παράδειγμα, ένα από τα πρώτα μοντέλα CNN το οποίο σχεδιάστηκε για την αναγνώριση αντικειμένων σε εικόνες, αποτελείται από 16 επίπεδα (AlexNet) και έχει εξήντα εκατομμύρια (60,000,000) παραμέτρους και εξακόσιες πενήντα χιλιάδες (650,000) νευρώνες από τους οποίους οι περισσότεροι εκτελούν πράξεις συνέλιξης. Ο Alex Krizhevsky απέδειξε το 2012 ότι η χρήση σύγχρονων μονάδων GPU για την εκτέλεση πράξεων συνέλιξης, έχει ως αποτέλεσμα την εκπαίδευση μοντέλων CNN σε χρόνους έως και δύο τάξεις μεγέθους πιο κάτω σε σύγκριση με μία ισχυρή επεξεργαστή μονάδα CPU [3]. Ο χρόνος εκτέλεσης ενός πειράματος του δικτύου AlexNet έχει μετρηθεί στα 7.39 δευτερόλεπτα σε οκταπύρηγο επεξεργαστή Haswell @2.9Ghz και στα 0.71 δευτερόλεπτα σε μονάδα GPU, Nvidia K520 [4].

Είναι ιδιαίτερα σημαντικό ένα ρομπότ να μπορεί να αντιλαμβάνεται το περιβάλλον του. Αυτό περιλαμβάνει την αναγνώριση ανθρώπων, ζώων και αντικειμένων γενικότερα. Ωστόσο, θέλουμε τα ρομπότ να είναι όσο πιο “ελκυστικά” γίνεται και συνήθως μικρότερα σε μέγεθος από τον άνθρωπο (uncunny valley in robotics) [5], ανάλογα με το task που επιθυμούμε να εκτελέσουν. Αυτό, έχει ως αποτέλεσμα

1.2. ΣΚΟΠΟΣ - ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

να μην μπορούμε να τοποθετήσουμε ογκώδη, άρα με μεγάλη επεξεργαστική ισχύ, υπολογιστικά συστήματα στο σώμα των ρομποτικών συστημάτων σε όλες τις περιπτώσεις.

Παρόλο που σήμερα έχουν σχεδιαστεί μοντέλα CNN τα οποία έχουν την δυνατότητα να αναγνωρίσουν και να εντοπίσουν αντικείμενα ανάμεσα σε χιλιάδες κλάσεις, ο χρόνος εκτέλεσής τους είναι αρκετά μεγάλος (της τάξης των μερικών δευτερολέπτων σε σύγχρονες υπολογιστικές μονάδες CPU). Αυτό κάνει την χρήση των CNN σε εφαρμογές πραγματικού χρόνου, όπως για παράδειγμα στην ρομποτική, ακατάλληλη. Ωστόσο, η επιστημονική κοινότητα σήμερα προσπαθεί να δώσει λύσεις στο συγκεκριμένο πρόβλημα εστιάζοντας το ενδιαφέρον στην εξέλιξη των ενσωματωμένων συστημάτων και την σχεδίαση γρήγορου λογισμικού για υλοποιήσεις μοντέλων CNN τα οποία εκμεταλλεύονται κυρίως την υπολογιστική ισχύ των μονάδων GPU, αλλά και άλλων πολυπύρηνων επεξεργαστικών μονάδων.

1.2 ΣΚΟΠΟΣ - ΣΥΝΕΙΣΦΟΡΑ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η παρούσα διπλωματική εργασία μελετά την χρήση νευρωνικών δικτύων συνέλιξης (CNN) σε εφαρμογές ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων (object recognition and localization - object detection) σε εικόνες.

Στοχεύει στην υλοποίηση μοντέλων CNN και στην εφαρμογή τους σε προβλήματα πραγματικού (σχεδόν) χρόνου, όπου μία από της απαιτήσεις είναι ο γρήγορος χρόνος εκτέλεσης.

Εξετάζεται η ανάπτυξή μοντέλων CNN (για για εφαρμογές αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες) σε ένα σταθερό υπολογιστικό σύστημα (PC) καθώς και η χρήση υβριδικών ενσωματωμένων συστημάτων, τα οποία φέρουν μονάδες CPU και GPU (όπως για παράδειγμα το Jetson TK1).

Επίσης παρουσιάζει μία σειρά από διαδικασίες βελτιστοποίησης για το ενσωματωμένο σύστημα Jetson TK1, με στόχο την γρήγορη εκτέλεση της διαδικασίας προς-τα-εμπρός περάσματος (forward propagation) στο εκάστοτε νευρωνικό δίκτυο, καθώς και την μείωση της κατανάλωσης ισχύος κατά την διαδικασία εκτέλεσης.

1.3 ΔΙΑΡΘΡΩΣΗ ΤΗΣ ΑΝΑΦΟΡΑΣ

Η διάρθρωση της παρούσας διπλωματικής εργασίας είναι η εξής:

- **Κεφάλαιο 2:** Περιγράφονται τα βασικά θεωρητικά στοιχεία στα οποία βασίστηκαν οι υλοποιήσεις. Γίνεται εισαγωγή στην επιστήμη της μηχανικής μάθησης και καταλήγει στην περιγραφή της λειτουργίας και χρήσης των CNN.
- **Κεφάλαιο 3:** Παρουσιάζονται οι διάφορες τεχνικές και τα εργαλεία που χρησιμοποιήθηκαν στις υλοποιήσεις.
- **Κεφάλαιο 4:** Πλήρης περιγραφή των υλοποιήσεων διαφόρων μοντέλων CNN (AlexNet, VGG16, Tiny-YOLO).

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

- **Κεφάλαιο 5:** Παρουσιάζεται αναλυτικά η μεθοδολογία των πειραμάτων και τα αποτελέσματα.
- **Κεφάλαιο 6:** Παρουσιάζονται τα τελικά συμπεράσματα.
- **Κεφάλαιο 7:** Αναφέρονται τα προβλήματα που προέκυψαν και προτείνονται θέματα για μελλοντική μελέτη, αλλαγές και επεκτάσεις.

2

Τεχνικές Βαθιάς Μηχανικής Μάθησης και Αναγνώρισης Αντικειμένων στον χώρο

Τα τελευταία χρόνια, ο κλάδος της Τεχνητής Νοημοσύνης είναι ένας από τους πιο ραγδαία αναπτυσσόμενους κλάδους της επιστήμης της πληροφορικής με τεράστιο ερευνητικό και πρακτικό ενδιαφέρον. Διαιρείται σε δύο υποκατηγορίες: την συμβολική τεχνητή νοημοσύνη και την υποσυμβολική τεχνητή νοημοσύνη. Η πρώτη προσπαθεί να επιλύσει προβλήματα τεχνητής νοημοσύνης χρησιμοποιώντας αλγορίθμικές διαδικασίες, δηλαδή σύμβολα και λογικούς κανόνες, ενώ η δεύτερη προσπαθεί να αναπαράγει την ανθρώπινη “ευφυΐα” μέσα από την χρήση αριθμητικών μοντέλων που με την σύνθεσή τους προσομοιώνουν την λειτουργία του ανθρώπινου εγκεφάλου (υπολογιστική νοημοσύνη).

Η ικανότητα ενός νοούμενου (AI) συστήματος να αποκτά από μόνο του γνώση, εξάγοντας πρότυπα ή/και χαρακτηριστικά σημεία από τα δεδομένα, είναι γνωστή ως *Μηχανική Μάθηση* (Machine Learning - ML) [6].

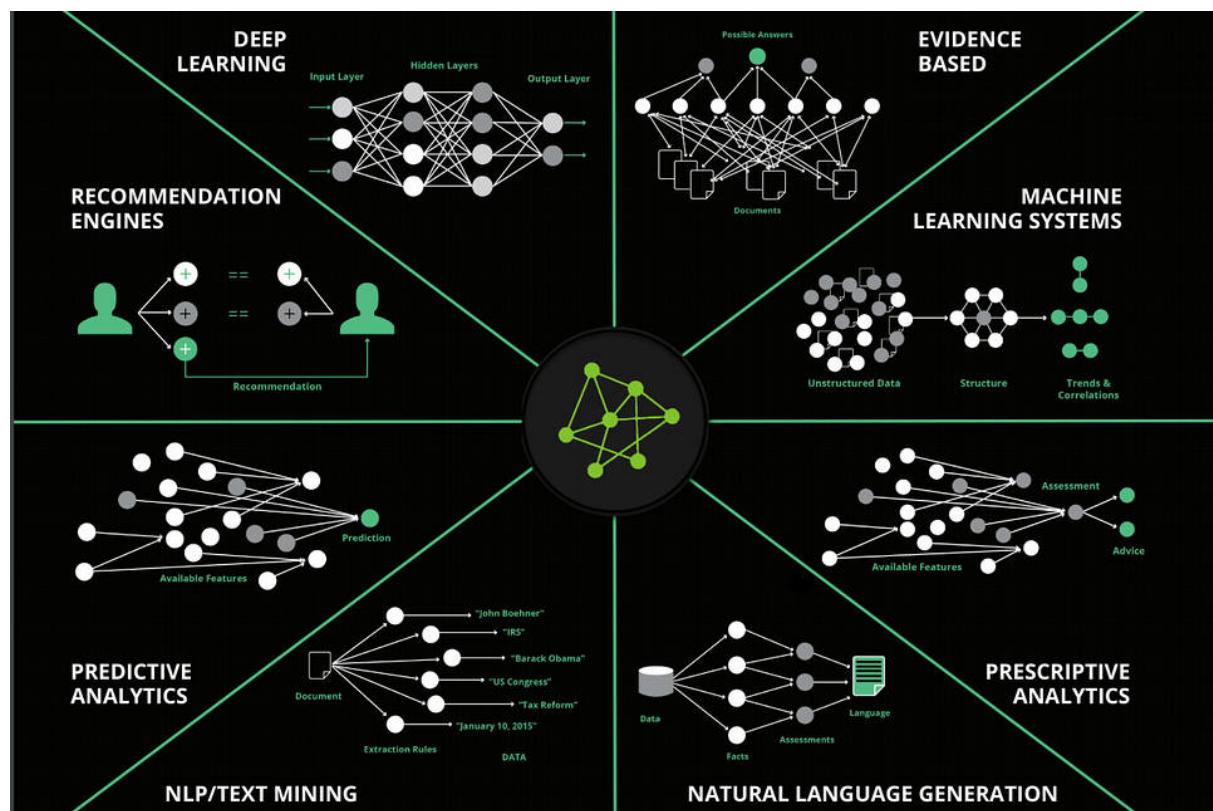
Στο κεφάλαιο αυτό παρουσιάζονται και αναλύονται τεχνικές και αλγόριθμοι *Μηχανικής Μάθησης* με επίκεντρο τα βαθιά νευρωνικά δίκτυα (Deep Neural Networks - DNN). Οι βασικές αρχές και λειτουργίες των νευρωνικών αυτών δικτύων αποτελούν τις βάσεις για την περαιτέρω μελέτη των νευρωνικών δικτύων συνέλιξης (CNN) και των εφαρμογών αυτών στο πρόβλημα της αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες.

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ

2.1 ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΗΣ ΣΥΓΓΧΡΟΝΗΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Η εισαγωγή του κλάδου της μηχανικής μάθησης στην επιστήμη των υπολογιστών, επέτρεψε στους υπολογιστές να μπορούν να αντιμετωπίσουν προβλήματα αντίληψης για τον πραγματικό κόσμο, όσο και να παίρνουν υποκειμενικές αποφάσεις.

Οι αλγόριθμοι ML, επιτρέπουν σε συστήματα AI να προσαρμόζονται εύκολα σε καινούργια προβλήματα, απαιτώντας ελάχιστη επέμβαση από τον άνθρωπο. Για παράδειγμα, ένα Νευρωνικό Δίκτυο που έχει εκπαιδευτεί να αναγνωρίζει γάτες σε εικόνες, δεν απαιτεί να σχεδιαστεί και να εκπαιδευτεί από το μηδέν για να έχει την ικανότητα να αναγνωρίζει και σκύλους.



Σχήμα 2.1: Κλάδοι και εφαρμογές της επιστήμης της Τεχνητής Νοημοσύνης

Πολλά προβλήματα που μέχρι πριν μερικά χρόνια λύνονταν με “χειρόγραφη”, προγραμματισμένη από τον άνθρωπο γνώση, σήμερα επιλύονται με χρήση αλγορίθμων ML (σχήμα 2.1). Κάποια παραδείγματα αφορούν:

- Αναγνώριση ομιλίας - Speech Recognition
- Μηχανική όραση - Computer Vision
 - Αναγνώριση αντικειμένων σε εικόνες - Object Recognition

2.1. ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΗΣ ΣΥΓΧΡΟΝΗΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

- Αναγνώριση και εντοπισμός της θέσης αντικειμένων σε εικόνες - Object Detection
- Αναγνώριση ηλεκτρονικών επιθέσεων στο διαδίκτυο - Cyberattack detection
- Επεξεργασία φυσικής γλώσσας - Natural Language Processing
 - Κατανόηση της φυσικής γλώσσας του ανθρώπου - Natural Language Understanding
 - Μοντελοποίηση και χρήση της φυσικής γλώσσας του ανθρώπου από μηχανές - Natural Language Generation
- Μηχανές αναζήτησης - Search Engines
- Αναπαράσταση γνώσης - Knowledge Representation
- Ρομποτική

Τα προβλήματα Μηχανικής Μάθησης χωρίζονται σε τρεις μεγάλες κατηγορίες:

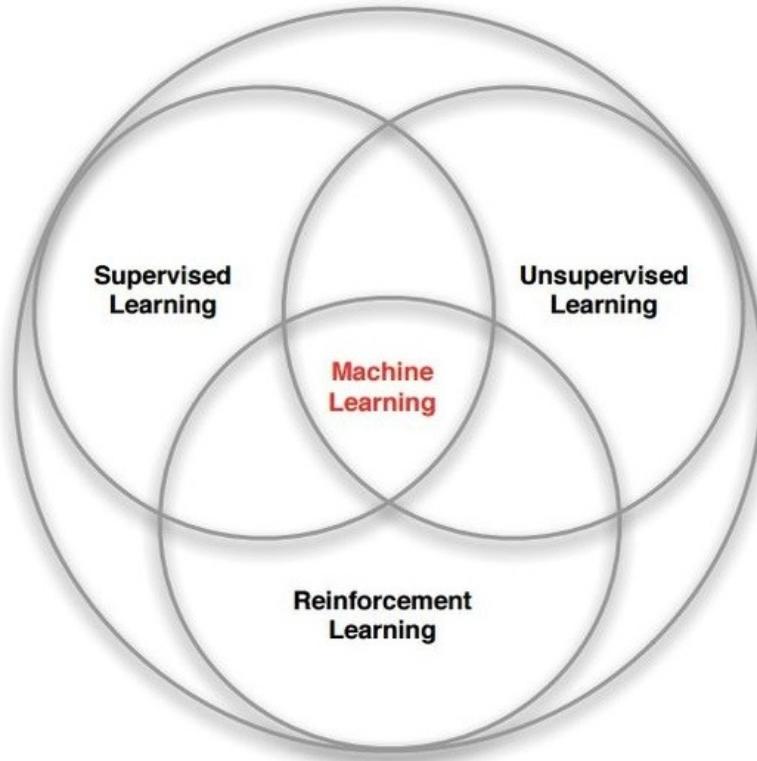
- Ύπο επίβλεψη Μάθηση - Supervised Learning: Στο υπολογιστικό σύστημα δίνονται παραδείγματα εισόδου και επιθυμητής εξόδου, δηλαδή στα δεδομένα έχουν προηγουμένως ανατεθεί ετικέτες (labels) και στόχος είναι να εξαχθεί ένας γενικός κανόνας αντιστοίχισης της εισόδου στην επιθυμητή έξοδο. Η αναγνώριση προκαθορισμένων αντικειμένων σε εικόνες είναι ένα πρόβλημα που ανήκει σε αυτή την κατηγορία.
- Χωρίς επίβλεψη Μάθηση - Unsupervised Learning: Τα δεδομένα δεν έχουν ετικέτες (labels) αφήνοντας έτσι τον αλγόριθμο ML να βρει από μόνος του δομές στα δεδομένα εισόδου.
- Εκμάθηση δια ανταμοιβής - Reinforcement Learning: Ο πράκτορας αλληλεπιδρά με ένα δυναμικό περιβάλλον στο οποίο πρέπει να εκτελέσει ένα συγκεκριμένο στόχο, χωρίς την ύπαρξη ενός “δασκάλου” που να ορίζει ρητά ανέχει φθάσει κοντά στον στόχο. Ένα παράδειγμα εφαρμογής είναι η αυτόματη πλοήγηση ενός οχήματος.

Κάποια προβλήματα είναι υβριδικά, δηλαδή συνδυασμός των πιο πάνω. Στο [σχήμα 2.2](#) απεικονίζεται το διάγραμμα Venn των διαφόρων αλγορίθμικών κατηγοριών ML.

Επιπλέον, οι Supervised Learning αλγόριθμοι χωρίζονται σε 2 κατηγορίες, ανάλογα με την επιθυμητή μορφή της εξόδου του αλγόριθμου ML:

- Ταξινόμησης - Classification: Όταν η έξοδος παίρνει διακριτές τιμές (discrete).
- Regression: Όταν η έξοδος παίρνει συνεχείς τιμές.

Γενικότερα, οι αλγόριθμοι ML ομαδοποιούνται και ανάλογα με την ομοιότητα τους σε σχέση με την λειτουργία που εκτελούν. Πιο κάτω αναφέρονται οι πιο δημοφιλείς αλγόριθμοι ML ομαδοποιημένοι με βάση την λειτουργία τους:

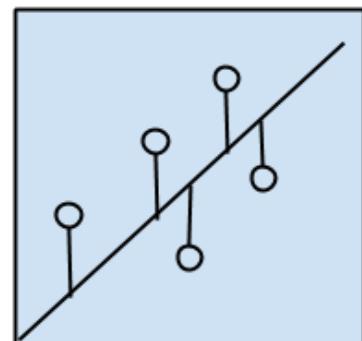


Σχήμα 2.2: Διάγραμμα Venn των διαφόρων κατηγοριών μηχανικής μάθησης

Regression

Ασχολείται με τη μοντέλοποίηση της σχέσης μεταξύ των μεταβλητών που ανανεώνονται επαναληπτικά χρησιμοποιώντας ένα μέτρο σφάλματος για τις προβλέψεις που γίνονται από το μοντέλο

- Ordinary Least Squares Regression (OLSR)
- Linear Regression
- Logistic Regression
- Stepwise Regression
- Multivariate Adaptive Regression Splines (MARS)
- Locally Estimated Smoothing (LOESS)

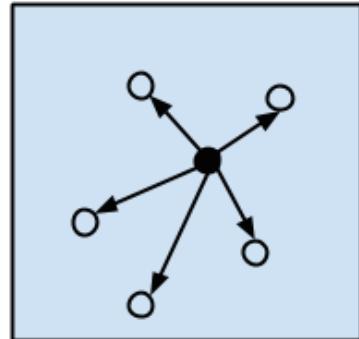


2.1. ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΗΣ ΣΥΓΧΡΟΝΗΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ

Instance-based

Αυτές οι μέθοδοι δημιουργούν μία βάση με παραδείγματα δεδομένων και συγκρίνουν τις νέες εισόδους με αυτές που έχουν καταχωρηθεί στην βάση δεδομένων χρησιμοποιώντας ένα μέτρο ομοιότητας, για την πιθανοτική εύρεση της καλύτερης αντιστοιχίας.

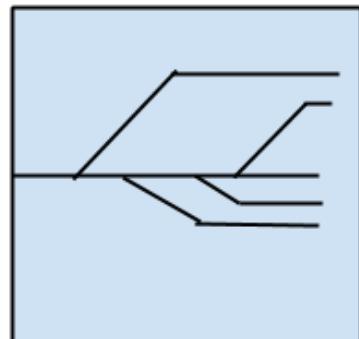
- k-Nearest Neighbour (kNN)
- Learning Vector Quantization (LVQ)
- Self-Organizing Map (SOM)
- Locally Weighted Learning (LWL)



Regularization

Χρησιμοποιούνται σαν επεκτάσεις άλλων μεθόδων και “τιμωρούν” πολύπλοκα μοντέλα, ευνοώντας έτσι απλούστερα μοντέλα τα οποία είναι συνήθως καλύτερα στην γενίκευση της επίλυσης του εκάστοτε προβλήματος.

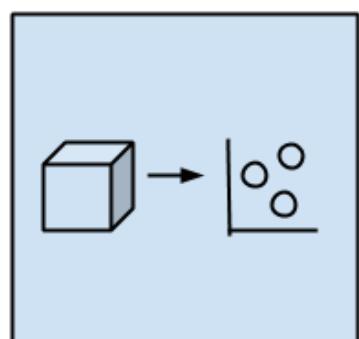
- Ridge Regression
- Least Absolute Shrinkage and Selection Operator (LASSO)
- Least-Angle Regression (LARS)
- Elastic Net



Dimensionality Reduction

Χρησιμοποιούνται για την αφαίρεση ασήμαντης πληροφορίας από τα δεδομένα. Πολλές από τις μεθόδους αυτές χρησιμοποιούνται σαν επεκτάσεις σε μοντέλα επίλυσης προβλημάτων regression και classification

- Principal Component Analysis (PCA)
- Discriminant Analysis: Linear (LDA), Mixture (MDA), Quadratic (QDA), Flexible (FDA)
- Principal Component Regression (PCR)
- Multidimensional Scaling (MDS)

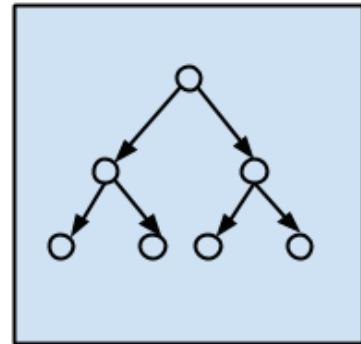


ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ

Decision Trees

Χρησιμοποιούνται για την κατασκευή μοντέλων λήψης αποφάσεων, τα οποία χρησιμοποιούν τις πραγματικές τιμές των χαρακτηριστικών των δεδομένων.

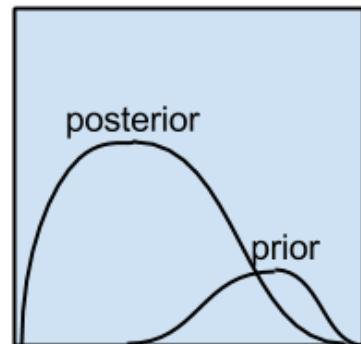
- Classification and Regression Tree (CART)
- Conditional Decision Trees
- M5



Bayesian

Εφαρμόζουν το θεώρημα του Bayes για την επίλυση τόσο προβλημάτων regression, αλλά και classification

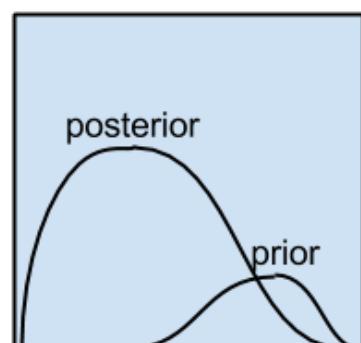
- Naive Bayes
- Gaussian Naive Bayes
- Bayesian Network (BN)
- Bayesian Belief Network (BBN)



Clustering

Περιγράφουν τις κλάσεις του προβλήματος

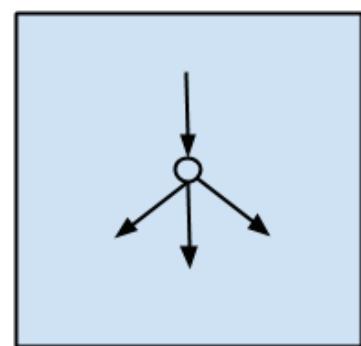
- k-Means
- k-Medians
- Expectation Maximisation (EM)
- Hierarchical Clustering



Artificial Neural Networks (ANN)

Μοντέλα εμπνευσμένα από τη δομή ή/και την λειτουργία των βιολογικών νευρωνικών δικτύων. Χρησιμοποιούνται στην επίλυση προβλημάτων classification ή/και regression.

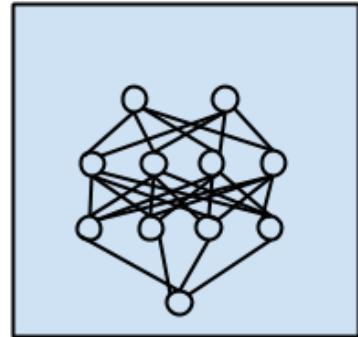
- Perceptron
- Back-Propagation
- Radial Basis Function Network (RBFN)



Deep Learning (DL)

Οι αλγόριθμοι DL είναι η σύγχρονη επέκταση των ANN, τα οποία εκμεταλλεύονται την αφθονία επεξεργαστικής ισχύος των σύγχρονων υπολογιστικών συστημάτων.

- Autoncoder
- Multilayer Perceptron (MLP)
- Deep Boltzmann Machine (DBM)
- Deep Belief Networks (DBN)
- Convolutional Neural Network (CNN)
- Stacked Auto-Encoders
- Recurrent Neural Networks (RNN)

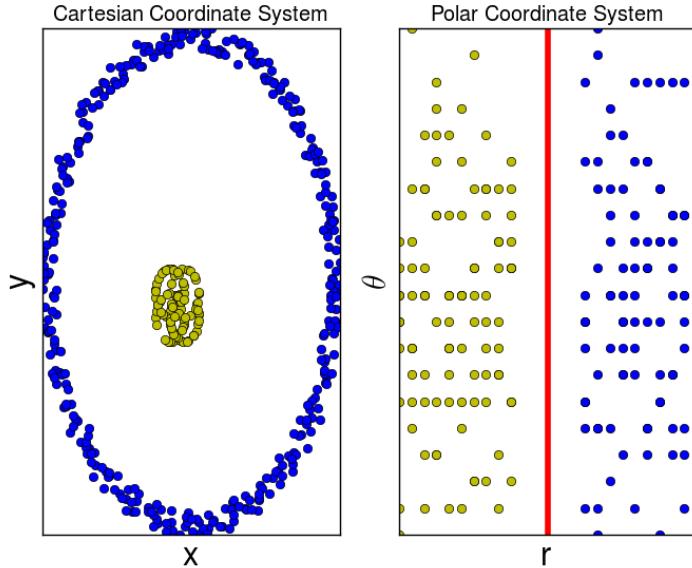


Η μορφή της αναπαράστασης των δεδομένων αποτελεί σημαντικό παράγοντα στην απόδοση των αλγορίθμων ML. Μία αναπαράσταση αποτελείται από χαρακτηριστικά (features). Για παράδειγμα ένα χρήσιμο χαρακτηριστικό στην ταυτοποίηση ομιλητή από δεδομένα ήχου, είναι η εκτίμηση του μεγέθους της φωνητικής έκτασης του ομιλητή. Έτσι, πολλά προβλήματα τεχνητής νοημοσύνης μπορούν να λυθούν με κατάλληλη σχεδίαση και επιλογή των χαρακτηριστικών για το συγκεκριμένο πρόβλημα. Το σύνολο των χαρακτηριστικών αυτών αποτελεί την αναπαράσταση των δεδομένων σε ένα πιο υψηλό και αφαιρετικό επίπεδο αντίληψης για τους υπολογιστές η οποία στην συνέχεια δίνεται σαν είσοδος σε έναν απλό ML αλγόριθμο, ο οποίος έχει μάθει να αντιστοιχεί την αναπαράσταση των δεδομένων στην επιθυμητή έξοδο.

Ένα απλό και κατανοητό παράδειγμα το οποίο δείχνει την εξάρτηση της επίδοσης των αλγορίθμων ML από την μορφή της αναπαράστασης που του δίνεται, φαίνεται στο [σχήμα 2.3](#). Έστω ότι θέλουμε να διαχωρίσουμε τα δεδομένα μας σε δύο κλάσεις, χαράζοντας μία ευθεία μεταξύ τους. Αν αναπαραστήσουμε τα δεδομένα στο Καρτεσιανό σύστημα συντεταγμένων (αριστερό διάγραμμα), τότε η επίλυση του προβλήματος είναι αδύνατη αφού δεν υπάρχει καμία ευθεία που να διαχωρίζει τις δύο κλάσεις. Ωστόσο, αν αναπαραστήσουμε τα δεδομένα στο πολικό σύστημα συντεταγμένων (δεξιό διάγραμμα), τότε το πρόβλημα λύνεται εύκολα χαράζοντας μία κάθετη ευθεία, με $r = a$, $a \in [r1, r2]$.

Στα περισσότερα προβλήματα τεχνητής νοημοσύνης, η επιλογή κατάλληλων χαρακτηριστικών είναι δύσκολο και χρονοβόρο έργο. Έστω ότι θέλουμε να αναγνωρίσουμε πρόσωπα σε εικόνες. Ένα χαρακτηριστικό θα μπορούσε να είναι τα μάτια. Δυστυχώς όμως, η αναγνώριση ματιών είναι ένα δύσκολο πρόβλημα αφού δεν μπορεί να περιγραφεί πάντα επακριβώς έχοντας σαν δεδομένα τις τιμές των pixel της εικόνας. Η γεωμετρική για παράδειγμα μορφή των ματιών σε μία εικόνα λήψης εξαρτάται από την γωνία λήψης της εικόνας, τον φωτισμό, τις ανακλάσεις του φωτισμού, την απόσταση από την οποία γίνεται η λήψη, την ανάλυση της κάμερας,

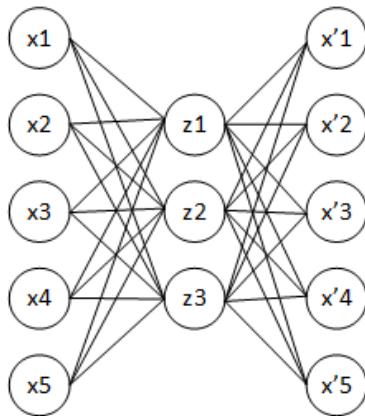
ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ



Σχήμα 2.3: Παράδειγμα διαφορετικών αναπαραστάσεων των δεδομένων

κτλ.

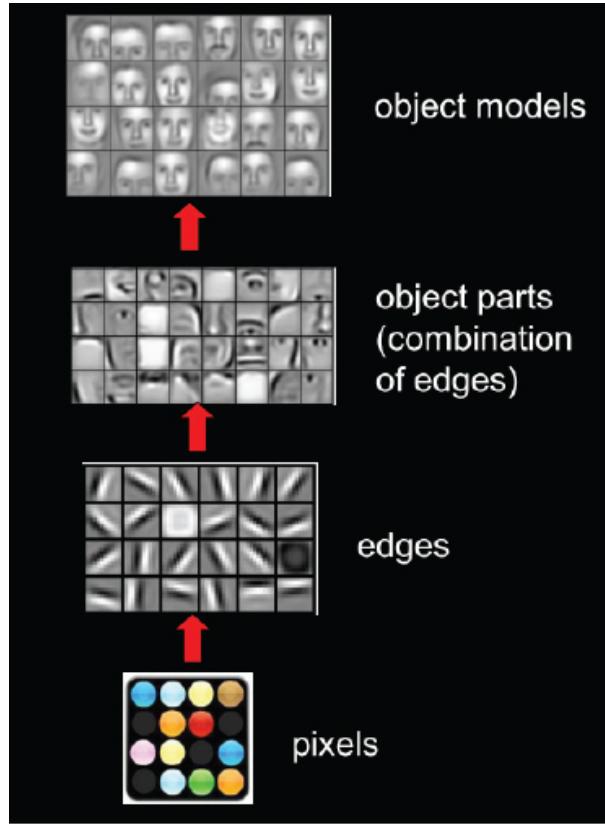
Το πρόβλημα αυτό, της επιλογής κατάλληλης αναπαράστασης, μπορεί να λυθεί χρησιμοποιώντας τεχνικές μηχανικής μάθησης για την εκμάθηση της ίδιας της αναπαράστασης. Αυτή η προσέγγιση είναι γνωστή ως *Εκμάθηση Αναπαραστάσεων* (*Representation Learning*). Οι αλγόριθμοι εκμάθησης αναπαραστάσεων είναι ικανοί να “μάθουν” ένα καλό σετ χαρακτηριστικών (features). Ένα απλό παράδειγμα αλ-



Σχήμα 2.4: Απλό μοντέλο Autoencoder με ένα κρυφό επίπεδο.

γορίθμου εκμάθησης αναπαραστάσεων είναι αυτό του Autoencoder [7] που φαίνεται στο σχήμα 2.4. Ο Autoencoder, στην πιο απλή του μορφή, είναι ο συνδυασμός ενός κωδικοποιητή (encoder) ο οποίος μετατρέπει τα δεδομένα εισόδου σε μία διαφορετική αναπαράσταση, και ενός αποκωδικοποιητή ο οποίος επαναφέρει την αναπαράσταση αυτή στην αρχική μορφή της αναπαράστασης των δεδομένων εισόδου. Οι Autoencoders ανήκουν στην κατηγορία των Νευρωνικών Δικτύων και είναι Unsupervised ML αλγόριθμοι. Ένα συχνά εμφανιζόμενο πρόβλημα σε εφαρμογές

2.1. ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΤΗΣ ΣΥΓΧΡΟΝΗΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ



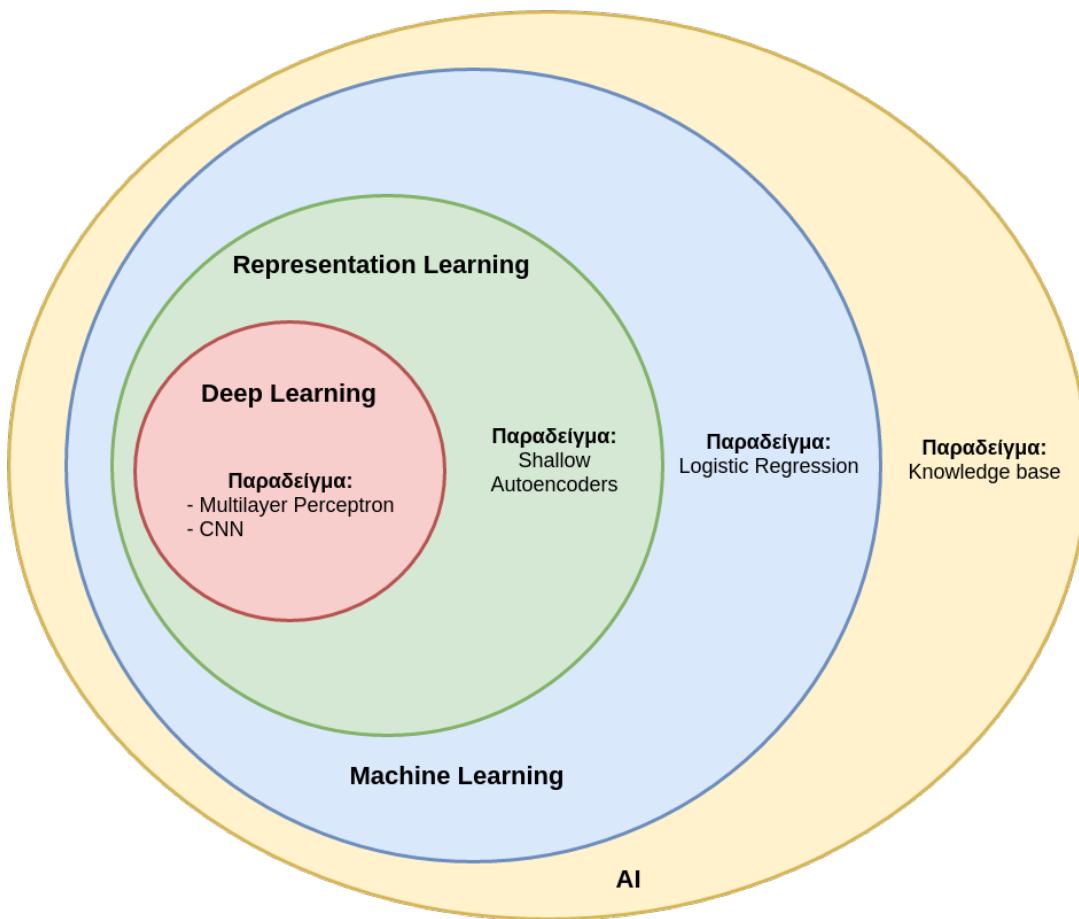
Σχήμα 2.5: Παράδειγμα απεικόνισης των φίλτρων ενός μοντέλου CNN για αναγνώριση προσώπου

τεχνητής νοημοσύνης είναι η εύρεση και εξαγωγή χαρακτηριστικών υψηλού επιπέδου από τα δεδομένα. Τα μοντέλα Βαθιάς Μηχανικής Μάθησης δίνουν λύσεις σε αυτό το πρόβλημα εκμάθησης αναπαραστάσεων με την εισαγωγή χαρακτηριστικών τα οποία εκφράζονται με βάση άλλες, απλούστερες αναπαραστάσεις. Αυτή η προσέγγιση δίνει την δυνατότητα στους υπολογιστές να κατασκευάζουν σύνθετες έννοιες χρησιμοποιώντας απλούστερες. Για παράδειγμα, η αναγνώριση αντικειμένων μπορεί να εκφραστεί με έννοιες όπως το γεωμετρικό σχήμα των αντικειμένων, το οποίο με την σειρά του ορίζεται από γωνίες και περιγράμματα. Επίσης, οι γωνίες και τα περιγράμματα ορίζονται από ακμές. Στο [σχήμα 2.5](#), παρουσιάζονται τα φίλτρα που έμαθε ένα μοντέλο CNN για αναγνώριση προσώπων σε εικόνες. Το συγκεκριμένο μοντέλο CNN έχει 3 χρυφά επίπεδα (hidden layers); το πρώτο χρυφό επίπεδο εξάγει από τα δεδομένα εισόδου (τιμές των πίξελ) πληροφορία σχετικά με τις ακμές, το δεύτερο, έχοντας σαν είσοδο την πληροφορία παρουσίας ακμών, εξάγει πληροφορία σχετικά με τις γωνίες και τα περιγράμματα και το τρίτο παίρνει σαν είσοδο την πληροφορία αυτή και κατασκευάζει μοντέλα αντικειμένων, δηλαδή εξάγει πληροφορία σχετικά με το γεωμετρικό σχήμα των αντικειμένων.

Συνοψίζοντας, η βαθιά μηχανική μάθηση, είναι μία υποκατηγορία του ML και ένας χαρακτηριστικός αντιπρόσωπος της σύγχρονης τεχνητής νοημοσύνης. Πιο συγκεκριμένα, είναι ένα είδος μηχανικής μάθησης, η οποία προσδίδει σε υπολογιστικά συστήματα ευφυΐα, δηλαδή την ικανότητα εκμάθησης με την χρήση εμπειρίας και

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ

δεδομένων. Σύμφωνα με τους *Ian Goodfellow, Yoshua Bengio και Aaron Courville*, η μηχανική μάθηση είναι η μόνη βιώσιμη προσέγγιση στην κατασκευή συστημάτων AI τα οποία μπορούν να αντεπεξέλθουν σε πολύπλοκα περιβάλλοντα και προβλήματα [8]. Η βαθιά μηχανική μάθηση καταφέρνει να μαθαίνει να αναπαριστά τον χόσμο ως μία ένθετη ιεραρχία εννοιών όπου η κάθε έννοια ορίζεται σε σχέση με άλλες πιο απλές έννοιες, και πιο αφηρημένες μορφές αναπαραστάσεων σε σχέση με λιγότερο αφηρημένες. Από το διάγραμμα Venn που βλέπουμε στο [σχήμα 2.6](#) παρατηρούμε ότι η βαθιά μηχανική μάθηση ανήκει στην κατηγορία της εκμάθησης αναπαραστάσεων, η οποία με την σειρά της είναι ένα είδος μηχανικής μάθησης που χρησιμοποιείται για την κατασκευή νοούμενων συστημάτων.



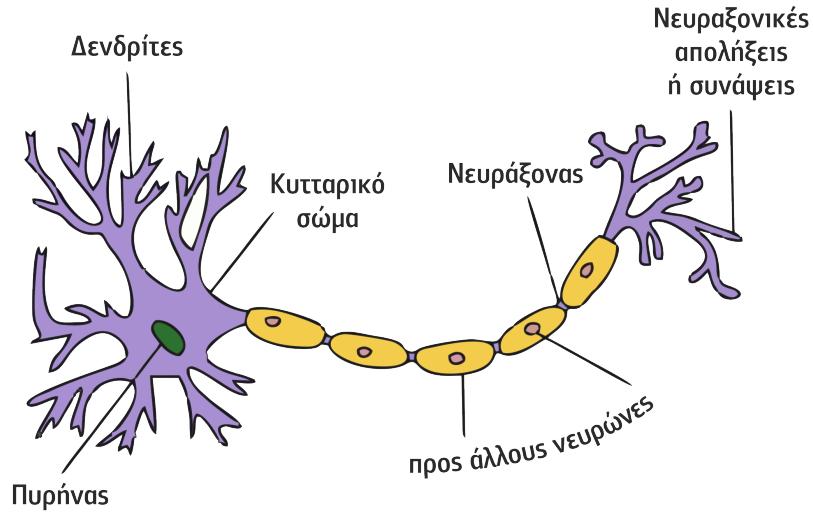
Σχήμα 2.6: Διάγραμμα Venn που δείχνει πως η επιστήμη της βαθιάς μάθησης ανήκει στην κατηγορία τεχνικών εκμάθησης αναπαραστάσεων, οι οποίες τεχνικές ανήκουν με την σειρά τους στην ευρύτερη επιστήμη της μηχανικής μάθησης

2.2 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΜΕ ΒΑΘΟΣ

Τα Νευρωνικά Δίκτυα είναι εμπνευσμένα από το βιολογικό νευρικό σύστημα του ανθρώπου. Η βασική επεξεργαστική μονάδα του εγκεφάλου είναι ο νευρώνας ([σχήμα 2.7](#)), ενώ το ανθρώπινο νευρικό σύστημα αποτελείται από περίπου 86

2.2. ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΜΕ ΒΑΘΟΣ

εκατομμύρια νευρώνες και περίπου $10^{14} - 10^{15}$ διασυνδέσεις. Τα κύρια μέρη ενός



Σχήμα 2.7: Βιολογικός νευρώνας.

νευρώνα είναι τα εξής:

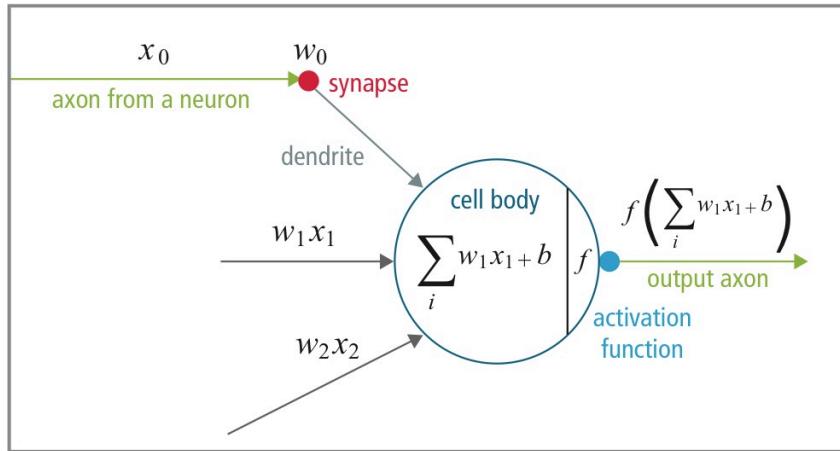
- **Δενδρίτης (Dendrites):** Δέχεται είσοδο από άλλους νευρώνες.
- **Σώμα του κυττάρου (Cell body):** Εξάγει συμπεράσματα, με βάση τις εισόδους.
- **Νευράξονας (Axon):** Συνδέει την έξοδο που λαμβάνεται από το σώμα του κυττάρου με τις νευραξονικές απολήξεις.
- **Νευραξονικές απολήξεις:** Συνδέει τον νευράξονα του εκάστοτε νευρώνα με τους τερματικούς κόμβους, από όπου και μεταφέρεται η πληροφορία στην είσοδο άλλων νευρώνων.

Κάθε νευρώνας δέχεται είσοδο από άλλους νευρώνες μέσω των δενδριτών του και στην συνέχεια επεξεργάζεται το σήμα που λαμβάνει στην είσοδο και στέλνει το αποτέλεσμα στον νευράξονα. Τέλος, άλλοι νευρώνες συνδέονται με αυτόν μέσω των συνάψεων του.

Το αντίστοιχο μαθηματικό μοντέλο του νευρώνα, φαίνεται στο [σχήμα 2.8](#). Η πληροφορία που μεταφέρεται από τις νευραξονικές απολήξεις (x_0), προτού μεταφερθεί στους δενδρίτες των επόμενων νευρώνων, αλληλεπιδρά πολλαπλασιαστικά με τις συνάψεις ($w_0 * x_0$). Οι παράγοντες πολλαπλασιασμού w_n ονομάζονται βάρη και αποτελούν τις μεταβλητές παραμέτρους ενός νευρώνα. Η τιμή των παραμέτρων αυτών ελέγχει την επίδραση μεταξύ των νευρώνων. Η συνάρτηση ενεργοποίησης f ελέγχει την ροή της πληροφορίας στους συνδεδεμένους νευρώνες και προσδίδει ευελιξία και ικανότητα εκτίμησης όσον αφορά πολύπλοκες μη γραμμικές σχέσεις στα δεδομένα εισόδου. Η έξοδος από ένα νευρώνα υπολογίζεται από τη σχέση:

$$a = f\left(\sum_{i=0}^N w_i x_i + b\right)$$

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ



Σχήμα 2.8: Μαθηματικό μοντέλο του νευρώνα

Η πιο απλή μορφή συνάρτησης ενεργοποίησης είναι η σιγμοειδής συνάρτηση $\sigma(x) = 1/(1 + e^{-x})$. Εναλλακτικά, η σιγμοειδής συνάρτηση ενεργοποίησης μπορεί να εκφραστεί σε διακριτή μορφή ως

$$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

Η επιλογή κατάλληλης συνάρτησης ενεργοποίησης των νευρώνων δεν είναι τυχαία, αφού όπως θα δούμε στην συνέχεια επηρεάζει την απόδοση του αλγορίθμου *Backpropagation*, ο οποίος χρησιμοποιείται για την εκπαίδευση των νευρωνικών δικτύων.

Γενικότερα, ο νευρώνας μπορεί να είναι και πολωμένος (*biased - b*) και έτσι το μαθηματικό μοντέλο που τον περιγράφει πλήρως παίρνει την μορφή:

$$\sigma = f(\sum_i w_i x_i + b) = \frac{1}{1+e^{-\sum_i w_i x_i + b}}$$

Θα μπορούσαμε να ερμηνεύσουμε το αποτέλεσμα της εφαρμογής της σιγμοειδούς συνάρτησης ενεργοποίησης ως την πιθανότητα μίας από τις κλάσεις:

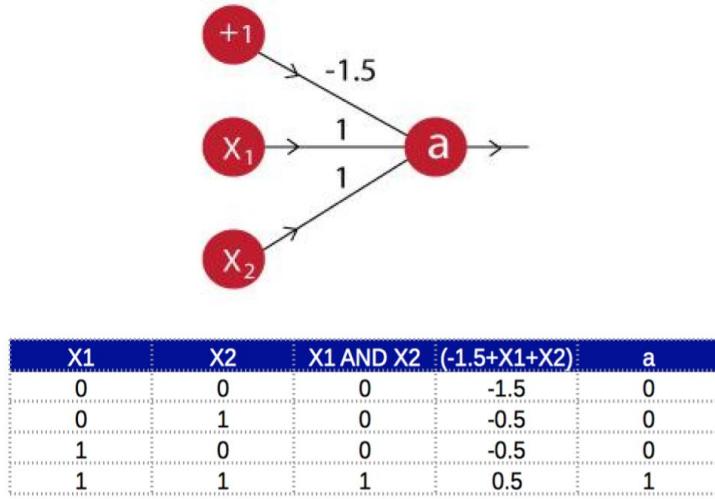
$$P(y_i = 1|x_i; w)$$

$$P(y_i = 0|x_i; w) = 1 - P(y_i = 1|x_i; w)$$

Με εφαρμογή κατάλληλης συνάρτησης σφάλματος στην έξοδο, ο νευρώνας έχει την συμπεριφορά ενός γραμμικού ταξινομητή (linear classifier). Πιο συγκεκριμένα, σε περίπτωση που χρησιμοποιήσουμε την *cross-entropy* συνάρτηση σφάλματος ο νευρώνας μετατρέπεται σε δυαδικό ταξινομητή *Softmax*¹.

Οι τρεις θεμελιώδεις εφαρμογές του μαθηματικού μοντέλου του νευρώνα είναι η μοντελοποίηση των λογικών πυλών *AND*, *OR* και *NOT*. Στο σχήμα 2.9 παρουσιάζεται το αντίστοιχο μοντέλο της λογικής πύλης *AND*. Ο νευρώνας δέχεται σαν είσοδο

¹Ταξινομητής Softmax: <http://cs231n.github.io/linear-classify/#softmax>



Σχήμα 2.9: Υλοποίηση πύλης AND με χρήση του μαθηματικού μοντέλου του νευρώνα

2 σήματα (X_1, X_2) και μία παράμετρο πόλωσης ($b = -1.5$). Η έξοδος a ορίζεται ως:

$$a = f(x) = f(X_1, X_2) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$

$$x = X_1 + X_2 - 1.5$$

Η σύνδεση πολλών νευρώνων σε σε ένα γράφο δομεί ένα *Νευρωνικό Δίκτυο - NN*. Το μοντέλο NN που φαίνεται στο [σχήμα 2.10](#) ονομάζεται *Perceptron* ή αλλιώς *Feedforward Artificial Neural Network - ANN*. Feedforward γιατί η πληροφορία ρέει προς μία μόνο κατεύθυνση, δηλαδή η έξοδος νευρώνα στο i επίπεδο δεν συνδέεται με την είσοδο νευρώνα που βρίσκεται το επίπεδο $k \leq i$.

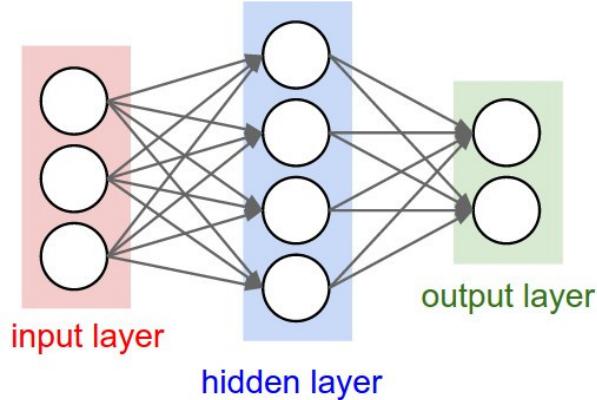
Ένα ANN έχει τα εξής χαρακτηριστικά:

- Οι διασυνδέσεις μεταξύ των νευρώνων δεν σχηματίζουν σε καμία περίπτωση κύκλο.
- Αποτελείται από 2 ή περισσότερα κρυφά επίπεδα; ένα κρυφό και το επίπεδο έξόδου
- Χρησιμοποιείται η σιγμοειδής συνάρτηση ενεργοποίησης

To Perceptron δεν είναι το μόνο μοντέλο NN που ανήκει στην κατηγορία των Feedforward ANN. Όπως θα δούμε στο [υποκεφάλαιο 2.3](#), τα *Νευρωνικά Δίκτυα Συνέλιξης (Convolutional Neural Networks - CNN)* ανήκουν και αυτά στην κατηγορία αυτή.

Η γενική δομή ενός νευρωνικού δικτύου φαίνεται στο [σχήμα 2.11](#). Τα γνωρίσματα ενός τέτοιου, πολυ-επίπεδου NN, είναι τα εξής:

- Αριθμός κρυφών επιπέδων



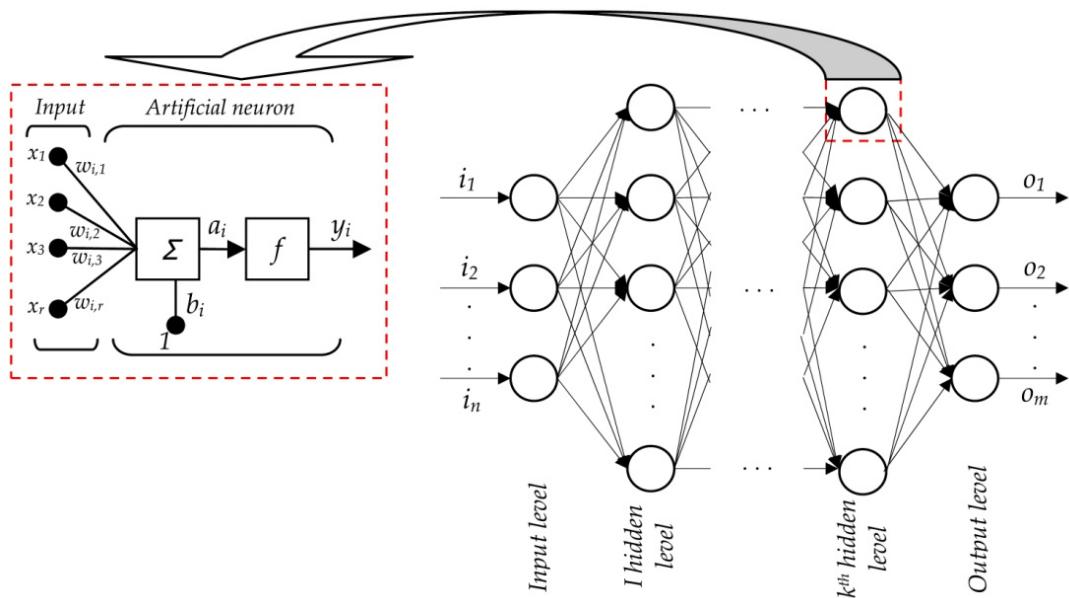
Σχήμα 2.10: Απλό μοντέλο NN με ένα κρυφό επίπεδο - Perceptron

- Αριθμός των νευρώνων στο κάθε επίπεδο

Η έξοδος από το εκάστοτε επίπεδο μπορεί να εκφραστεί ως

$$A_{i+1} = f_i(A_i \bullet W_i + B_i)$$

όπου ο πίνακας A_i αναφέρεται στην είσοδο και έχει διαστάσεις $M \times N$, W είναι ο πίνακας με τα βάρη των νευρώνων του εκάστοτε επιπέδου με διαστάσεις $K \times M$, και τέλος ο πίνακας B διαστάσεων $K \times N$ αναφέρεται στις τιμές πόλωσης. Η τιμή i αναφέρεται στον αριθμό του εκάστοτε επιπέδου του NN. Ο αριθμός των επιπέδων, ή καλύτερα το μέγιστο μήκος του μονοπατιού που ακολουθεί η πληροφορία από την είσοδο μέχρι την έξοδο του NN, ορίζει το βάθος ενός NN.



Σχήμα 2.11: Μορφή ενός πολυεπίπεδου νευρωνικού δικτύου

Ο [αλγόριθμος 2.1](#) υλοποιεί την διαδικασία υπολογισμού της εξόδου ενός νευρωνικού δικτύου (forward pass), έχοντας σαν δεδομένα τα βάρη και τις τιμές πόλωσης των νευρώνων του κάθε επίπεδου, καθώς και τα δεδομένα εισόδου.

Άλγοριθμος 2.1 Άλγοριθμος Feedforward για τον υπολογισμό των εξόδων ενός επιπέδου του NN

```

1: function ACTIVATION(a)
2:     return  $1.0/(1.0 + e^{-a})$ 
3: end function
4:
5: procedure NN_FORWARD(X, W, B, num_layers)
6:     Starting from the input layer, use  $\sigma$  to do a forward pass through the network,
      computing the activities of the neurons at each layer.
7:      $k \leftarrow 0$ 
8:     while  $k < num\_layers$  do
9:          $X^k \leftarrow \text{ACTIVATION}(X \bullet W_k + B_k)$            # If we want to keep output from
           intermediate layers, we must add up one dimension on X.
10:         $k \leftarrow k + 1$ 
11:    end while
12: end procedure

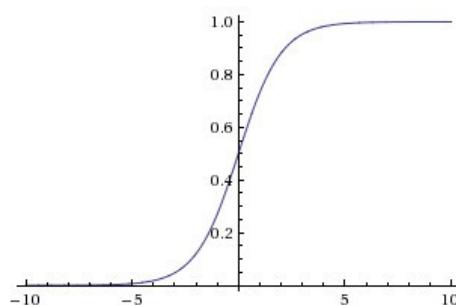
```

2.2.1 Συναρτήσεις Ενεργοποίησης

Σιγμοειδές - Sigmoid

Η σιγμοειδής μη γραμμική συνάρτηση έχει την μορφή που περιγράφηκε στο [υποχεφάλαιο 2.1](#). Παίρνει σαν είσοδο έναν πραγματικό αριθμό και τον κανονικοποιεί στο διάστημα $[0, 1]$

$$\sigma(x) = 1/(1 + e^{-x})$$



Σχήμα 2.12: Συνάρτηση Σιγμοειδούς συνάρτησης

Υπερβολική Εφαπτομένη - Tanh

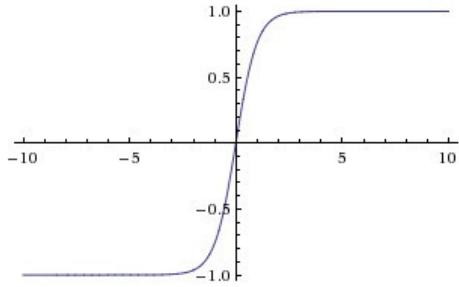
Παίρνει σαν είσοδο έναν θετικό αριθμό και τον κανονικοποιεί στο διάστημα $[-1, 1]$ χρησιμοποιώντας την πιο κάτω σχέση:

$$\tanh x = 2\sigma(2x) - 1$$

ReLU

Μία από τις πιο δημοφιλές συναρτήσεις ενεργοποίησης τα τελευταία χρόνια. Πρακτικά κρατά την ενεργοποίηση οριοθετημένη στο μηδέν και είναι γρήγορη στον

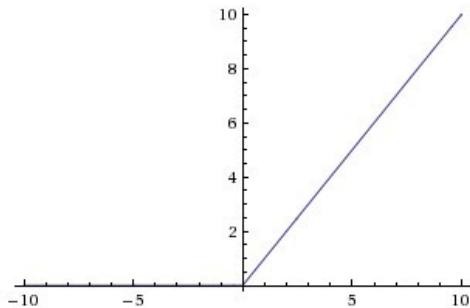
ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ



Σχήμα 2.13: Συνάρτηση Υπερβολικής Εφαπτομένης

υπολογισμό.

$$f(x) = \max(0, x) \equiv f(x) = \begin{cases} x, & \text{Αν } x > 0 \\ 0, & \text{Διαφορετικά} \end{cases}$$



Σχήμα 2.14: Συνάρτηση Rectified Linear Unit - ReLU

Το μειονέκτημά της είναι ότι κατά την διάρκεια της εκπαίδευσης του νευρωνικού δικτύου τα βάρη ανανεώνονται με τέτοιο τρόπο που ο νευρώνας μπορεί να μην ενεργοποιηθεί ποτέ. Αυτό έχει σαν αποτέλεσμα να “σκοτώσει” τον συγκεκριμένο νευρώνα.

Leaky ReLU

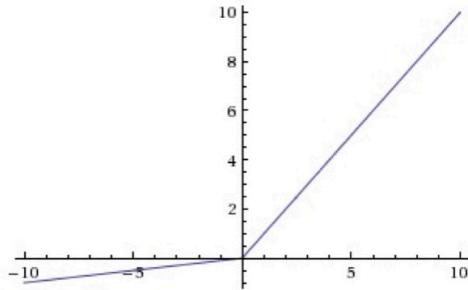
Η συνάρτηση ενεργοποίησης Leaky ReLU προσπαθεί να λύσει το προαναφερθέν πρόβλημα που εμφανίζεται με την χρήση της συνάρτησης ReLU. Αντί να μηδενίζεται για $x < 0$, η συνάρτηση Leaky ReLU έχει μικρή κλίση:

$$f(x) = 1(x < 0)(ax) + 1(x \geq 0)(x) = \begin{cases} x, & \text{Αν } x > 0 \\ ax, & \text{Διαφορετικά} \end{cases}$$

Η τιμή της σταθεράς a ορίζει την κλίση της συνάρτησης για $x < 0$ και μπορεί να δοθεί σαν παράμετρος του εκάστοτε νευρώνα.

Η πρώτη εφαρμογή της συνάρτησης Leaky ReLU σαν συνάρτηση ενεργοποίησης νευρώνων έγινε το 2015 από τον Kaiming He. Οι νευρώνες οι οποίοι χρησιμοποιούν την συνάρτηση Leaky ReLU ονομάζονται νευρώνες *PReLU* [9].

Maxout

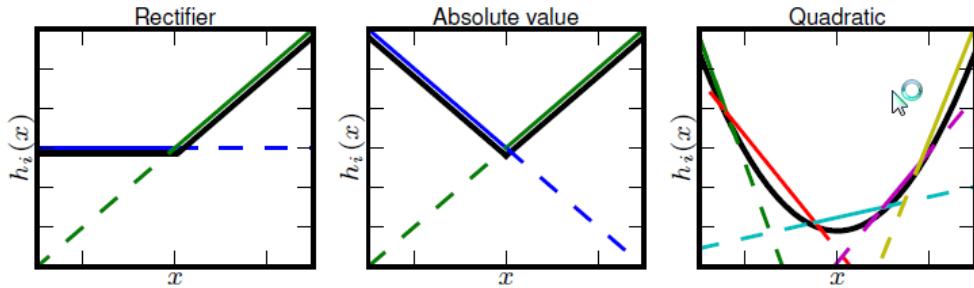


Σχήμα 2.15: Συνάρτηση Leaky ReLU

Η συνάρτηση ενεργοποίησης Maxout [10] είναι η γενίκευση της συνάρτησης Leaky ReLU. Ένας νευρώνας Maxout, υπολογίζει την συνάρτηση

$$f(x) = \max(w_1x + b_1, w_2x + b_2)$$

Η πιο πάνω συνάρτηση έχει τέσσερις παραμέτρους (w_1, w_2, b_1 και b_2). Επίσης, οι συναρτήσεις ReLU και Leaky ReLU είναι ειδικές περιπτώσεις της συνάρτησης Maxout. Για παράδειγμα, για $w_1, b_1 = 0$ παίρνει την μορφή της συνάρτησης ReLU. Το μειονέκτημα αυτής της συνάρτησης ενεργοποίησης, σε σχέση με την συνάρτηση ReLU, είναι ότι διπλασιάζει τις παραμέτρους κάθε νευρώνα.



Σχήμα 2.16: Συνάρτηση Maxout

Πρακτικά, τα μοντέλα NN που χρησιμοποιούν την συνάρτηση Maxout έχουν την ικανότητα να μάθουν, πέρα από την συσχέτιση μεταξύ των χρυμμένων επιπέδων, και την συνάρτηση ενεργοποίησης όλων των νευρώνων ενός ή περισσότερων επιπέδων. Στο [σχήμα 2.16](#) φαίνονται διάφορες μορφές της συνάρτησης Maxout, μετά από την εκπαίδευση δικτύων Maxout.

2.2.2 Συναρτήσεις Σφάλματος/Κόστους

Από τις προηγούμενες παραγράφους γίνεται κατανοητό ότι τα νευρωνικά δίκτυα χρησιμοποιούνται για την επίλυση προβλημάτων classification και regression. Γενικότερος στόχος είναι να παρθεί μία απόφαση, η οποία φαίνεται στην έξοδο του μοντέλου, έχοντας σαν είσοδο τα δεδομένα του εκάστοτε προβλήματος. Έτσι προκύπτει ότι στα μοντέλα NN τα οποία χρησιμοποιούνται για την επίλυση τέτοιων προβλημάτων, ενσωματώνεται ένας ταξινομητής (classifier). Ο ταξινομητής αυτός συνήθως αποτελεί το τελευταίο επίπεδο ενός NN.

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ

Οι συναρτήσεις σφάλματος έχουν στόχο να τιμωρήσουν τις λανθασμένες αποφάσεις που λαμβάνονται από τον ταξινομητή, και άρα στην έξοδο ενός NN, κατά την διάρκεια της εκπαίδευσης του.

Μία συνάρτηση σφάλματος έχει την γενική μορφή:

$$E_{total} = f(target - output)$$

Σε προβλήματα κατηγοριοποίησης (classification), όπως για παράδειγμα η αναγνώριση αντικειμένων σε εικόνες, η πιο γνωστή και συχνά χρησιμοποιούμενη συνάρτηση ταξινόμησης είναι η συνάρτηση *Softmax*, η οποία έχει την μορφή:

$$f_j(z) = \frac{e^{z_j}}{\sum_{k \neq j} e^{z_k}} = P(y = j|x)$$

όπου ο δείκτης j αναφέρεται σε ένα στοιχείο του διανύσματος των πιθανών κλάσεων. Η πιο πάνω μαθηματική εξίσωση μας δίνει την πιθανότητα η έξοδος για να ανήκει σε μία εκ του συνόλου των πιθανών κλάσεων, έχοντας υπό συνθήκη τα δεδομένα εισόδου x . Πιο συγκεκριμένα, παίρνει σαν είσοδο ένα διάνυσμα πραγματικών τιμών (z) και “κατασκευάζει” ένα καινούργιο διάνυσμα του οποίου τα στοιχεία είναι κανονικοποιημένα στο διάστημα $[0, 1]$ και το άθροισμα τους ισούται με την μονάδα, δηλαδή $\sum_{k=1}^N f_j(z) = 1$.

Η συνάρτηση σφάλματος που χρησιμοποιείται στην περίπτωση του ταξινομητή *Softmax* είναι η συνάρτηση *cross-entropy*:

$$L_i = -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$$

Σε προβλήματα regression οι πιο συχνά χρησιμοποιούμενες συναρτήσεις σφάλματος είναι οι *L1 Norm* και *L2 Norm*.

$$\begin{aligned} L1_i &= |f - y_i|_1 \\ L2_i &= |f - y_i|^2 \end{aligned}$$

Η πρώτη είναι η συνάρτηση μέσης τιμής ενώ η δεύτερη είναι η συνάρτηση του τετραγώνου της μέσης τιμής του σφάλματος.

2.2.3 Αλγόριθμος Backpropagation

Ο αλγόριθμος *backpropagation* (αλγόριθμος 2.2) εμφανίστηκε το 1970 και υποτιμήθηκε μέχρι το 1986, όταν και οι David Rumelhart, Geoffrey Hinton, και Ronald Williams απέδειξαν την αποδοτικότητα του στην εκπαίδευση των νευρωνικών δικτύων, κυρίως όσον αφορά στην ταχύτητα [11]. Σήμερα ο αλγόριθμος backpropagation χρησιμοποιείται κατά κόρον για την εκπαίδευση μεγάλων νευρωνικών δικτύων με εκατομμύρια παραμέτρους.

Σκοπός του αλγόριθμου *backpropagation* είναι να ελαχιστοποιήσει το σφάλμα, δοσμένης μίας συνάρτησης σφάλματος, ορισμένη στον χώρο των βαρών w , χρησιμοποιώντας τον αλγόριθμο *Gradient Descent*. Υπολογίζει δε το σφάλμα και ανανεώνει

ανάλογα τις τιμές του πολυεπίπεδου νευρωνικού δικτύου, ακολουθώντας μία προς τα πίσω διαδικασία.

Πρακτικά, ο αλγόριθμος backpropagation εφαρμόζει τον κανόνα της αλυσιδωτής παραγώγισης (gradient chain rule) για τον υπολογισμό των μερικών παραγώγων, δοσμένης μίας συνάρτησης σφάλματος.

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \bullet \frac{\partial y}{\partial x}$$

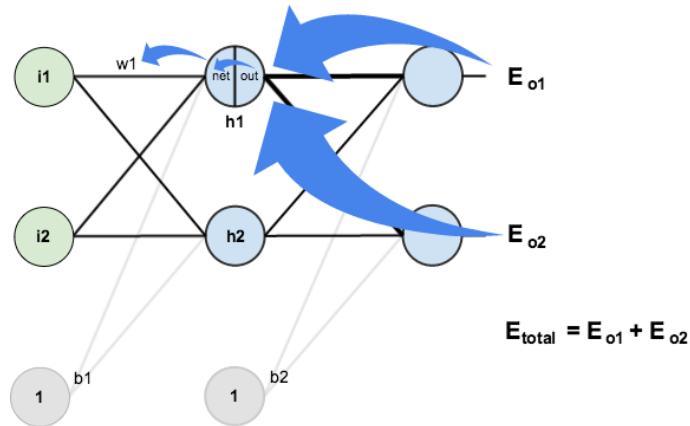
Το πρώτο βήμα για την ελαχιστοποίηση του σφάλματος είναι ο υπολογισμός της παραγώγου της συνάρτησης σφάλματος ως προς τις παραμέτρους w κάθε επιπέδου, ή καλύτερα κάθε νευρώνα, του νευρωνικού δικτύου.

$$\frac{\partial E_{total}}{\partial w_{ij}}$$

Στο [σχήμα 2.17](#) φαίνεται η διαδικασία υπολογισμού της παραγώγου της συνάρτησης σφάλματος ως προς τις παραμέτρους w ενός νευρωνικού δικτύου που αποτελείται από 1 κρυφό επίπεδο και 2 ουπότοπα.

$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$



Σχήμα 2.17: Διαδικασία Backward propagation

Η εξίσωση μερικής παραγώγισης της συνάρτησης σφάλματος ως προς την παράμετρο w_1 μπορεί να γραφεί σε πιο αναλυτική μορφή:

$$\begin{aligned} \frac{\partial E_{total}}{\partial w_1} &= \left(\sum_{k=1}^O \frac{\partial E_{total}}{\partial out_k} * \frac{\partial out_k}{\partial net_k} * \frac{\partial net_k}{\partial out_{h1}} \right) * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1} \\ \frac{\partial E_{total}}{\partial w_1} &= \left(\sum_{k=1}^O \delta_{ho} * w_{ho} \right) * out_{h1}(1 - out_{h1}) * i_1 \\ \frac{\partial E_{total}}{\partial w_1} &= \delta_{h1} i_1 \end{aligned}$$

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ

Η αλυσιδωτή παραγώγιση εμπλέκει και την μερική παραγώγιση της συνάρτησης ενεργοποίησης κάθε νευρώνα ως προς τις παραμέτρους w του. Έτσι, η επιλογή της συνάρτησης ενεργοποίησης επηρεάζει και την απόδοση, τόσο σε χρόνο εκτέλεσης, όσο και σε σφάλματα λόγω παραγώγισης, του αλγορίθμου backpropagation.

Η δε ανανέωση των βαρών w γίνεται με βάση την σχέση

$$w_i^+ = w_i - step * \frac{\partial E_{total}}{\partial w_i}$$

στην οποία αφαιρούνται από την αρχική τιμή η μεταβολή του σφάλματος πολλαπλασιασμένη με ένα βαθμωτό μέγεθος. Το μέγεθος αυτό οποίο ονομάζεται ρυθμός ανανέωσης των βαρών και είναι σταθερά της διαδικασίας εκπαίδευσης.

Αλγόριθμος 2.2 Backpropagation learning algorithm

```
for d in data do
    FORWARDS PASS
    Starting from the input layer, do a forward pass through the network,
    computing the activities of the neurons at each layer.
    BACKWARDS PASS
    Compute the derivatives of the error function with respect to
    the output layer activities
    for layer in layers do
        Compute the derivatives of the error function with respect to
        the inputs of the upper layer neurons
        Compute the derivatives of the error function with respect to
        the weights between the outer layer and the layer below
        Compute the derivatives of the error function with respect
        to the activities of the layer below
    end for
    Updates the weights.
end for
```

Η πλήρης ανάλυση του αλγόριθμου backpropagation ξεφεύγει από τα πλαίσια της παρούσας διπλωματικής εργασίας γιατί δεν ασχολείται με την εκπαίδευση των νευρωνικών δικτύων.

2.3 ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΣΥΝΕΛΙΞΗΣ

Έως τώρα έγινε αναφορά στα πολυεπίπεδα νευρωνικά δίκτυα και την γενικότερη λειτουργία τους. Σε αυτή την παράγραφο γίνεται αναφορά σε συγκεκριμένα μοντέλα πολυεπίπεδων νευρωνικών δικτύων και πιο συγκεκριμένα για τα Νευρωνικά Δίκτυα Συνέλιξης. Τα συγκεκριμένα μοντέλα χρησιμοποιούνται σήμερα κυρίως στα προβλήματα της αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες.

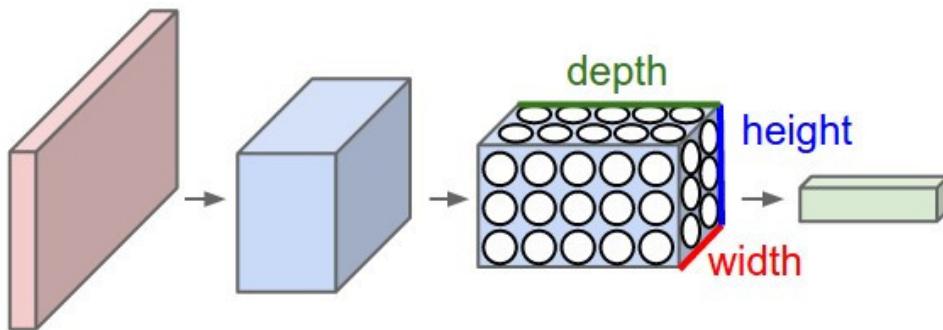
Ο τρόπος λειτουργίας τους είναι όμοιος με αυτόν που παρουσιάστηκε στο [υποκεφάλαιο 2.2](#). Δηλαδή, αποτελούνται από πολλά επίπεδα, όπου το κάθε επίπεδο

2.3. ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΣΥΝΕΛΙΞΗΣ

αποτελείται από έναν αριθμό νευρώνων οι οποίοι έχουν σαν παραμέτρους εκμάθησης τα βάρη τους (w^i_j) και την τιμή πόλωσης (b^i). Κάθε νευρώνας δέχεται ένα σύμα εισόδου, εφαρμόζει μία πράξη εσωτερικού γινομένου σε αυτό, και προαιρετικά εφαρμόζει στο αποτέλεσμα μία μη γραμμική συνάρτηση. Το τελευταίο επίπεδο των CNN είναι πλήρες συνδεδεμένο και έχει μία συνάρτηση σφάλματος. Η διαφορά των μοντέλων CNN από τα κλασσικά ANN είναι ότι τα δεδομένα εισόδου είναι εικόνες.

Τα CNN μοντελοποιούν μικρά τμήματα πληροφορίας, τα οποία στην συνέχεια ενώνονται για να δημιουργήσουν υψηλότερου επιπέδου πληροφορία. Για παράδειγμα σε ένα μοντέλο CNN το πρώτο επίπεδο προσπαθεί να εντοπίσει ακμές, το δεύτερο επίπεδο παίρνει την πληροφορία των ακμών και προσπαθεί να εντοπίσει περιγράμματα, κτλ.

Σε κάθε εικονοστοιχείο της εικόνας αντιστοιχούν 3 τιμές (RGB) και άρα η είσοδος σε ένα CNN έχει τρεις διαστάσεις όπως φαίνεται και στο [σχήμα 2.18](#). Για παράδειγμα ένα CNN το οποίο έχει σχεδιαστεί να δέχεται σαν είσοδο εικόνες ανάλυσης 80×60 έχει επίπεδο εισόδου διαστάσεων $80 \times 60 \times 3$.



Σχήμα 2.18: Τρισδιάστατη κατανομή των νευρώνων στα CNN

Κάθε επίπεδο ενός CNN παίρνει σαν είσοδο μία μορφή όγκου την οποία και μετασχηματίζει σε μία άλλη μορφή όγκου.

Οι τρεις βασικοί τύποι επιπέδων που χρησιμοποιούνται σε αρχιτεκτονικές CNN είναι:

- Επίπεδο Συνέλιξης - Convolutional Layer (CONV)
- Επίπεδο Υπόδειγματοληψίας- Pooling Layer (POOL)
- Πλήρως Συνδεδεμένο Επίπεδο - Fully-Connected Layer (FC)

Τα επίπεδα CONV και FC έχουν παραμέτρους, δηλαδή βάρη και τιμή πόλωσης των νευρώνων, ενώ τα επίπεδα POOL εκτελούν λειτουργία δειγματοληψίας στα δεδομένα εισόδου τους.

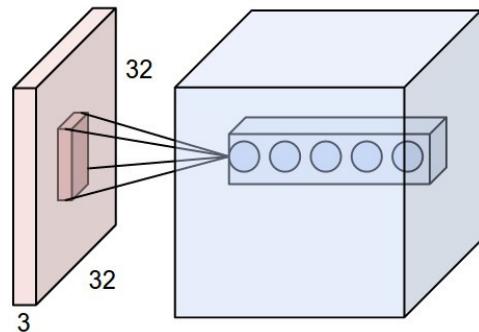
2.3.1 Επίπεδο Συνέλιξης

Τα επίπεδα συνέλιξης είναι ο πυρήνας των μοντέλων CNN. Οι παράμετροι ενός επιπέδου CONV είναι μία σειρά από δισδιάστατα φίλτρα τα οποία όμως εκτείνονται

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ

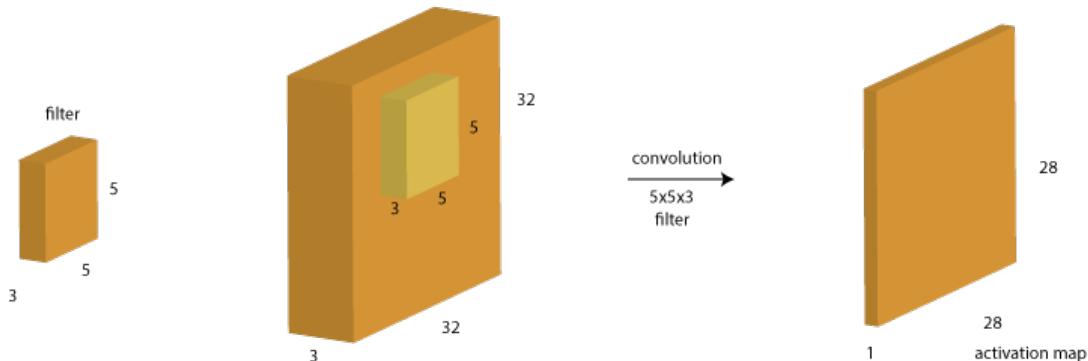
σε όλο το σε όλο το βάθος του όγκου εισόδου. Το βάθος των φίλτρων αυτών ισούται με το βάθος του όγκου στην είσοδο.

Όπως προαναφέρθηκε, τα επίπεδα CONV εφαρμόζουν πράξη συνέλιξης πάνω στα δεδομένα εισόδου. Αυτό επηρεάζει την δομή των "τοπικών" διασυνδέσεων. Στο παράδειγμα του σχήματος 2.19 βλέπουμε πως ο κάθε νευρώνας του επιπέδου συνέλιξης συνδέεται με μία περιοχή του όγκου στην είσοδο του.



Σχήμα 2.19: Παράδειγμα διασύνδεσης τρισδιάστατης εισόδου με την τρισδιάστατη δομή των νευρώνων ενός επιπέδου συνέλιξης (CONV)

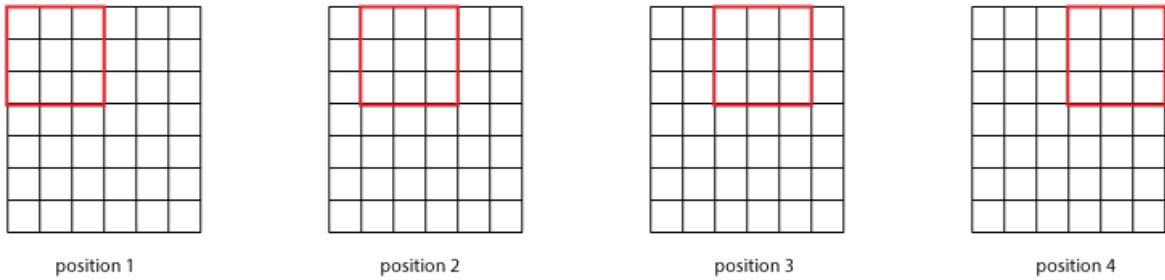
Η συνέλιξη ενός φίλτρου με τον τον όγκο εισόδου παράγει έναν χάρτη ενεργοποίησης (*activation map*), με τον τρόπο που φαίνεται στο σχήμα 2.20. Στο παράδειγμα αυτό εφαρμόζεται φίλτρο διαστάσεων $5 \times 5 \times 3$ σε έναν όγκο $32 \times 32 \times 3$ και παράγεται ένας χάρτης ενεργοποίησης διαστάσεων $28 \times 28 \times 1$.



Σχήμα 2.20: Συνέλιξη φίλτρου ενός επιπέδου συνέλιξης με τον όγκο εισόδου και παραγωγή ενός χάρτη ενεργοποίησης

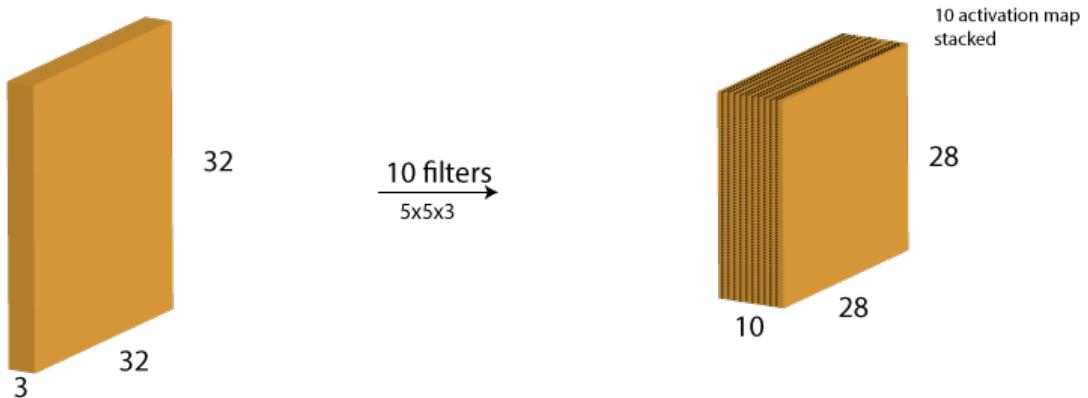
Η μείωση των διαστάσεων μήκους και πλάτους από 32×32 σε 28×28 οφείλεται στον τρόπο με τον οποίο εκτελείται η πράξη της συνέλιξης των φίλτρων με τον όγκο εισόδου (σχήμα 2.21). Οι διαστάσεις του όγκου εξόδου, έχοντας σαν είσοδο όγκο διαστάσεων $N \times N \times d$ και φίλτρων $F \times F \times d$ υπολογίζονται, στην απλούστερη περίπτωση με βάση την σχέση

$$outsize = (N - F) + 1$$



Σχήμα 2.21: Διαδικασία Συνέλιξης

Η τιμή του βάθους του όγκου στην έξοδο επιπέδου CONV αντιστοιχεί στον αριθμό των φίλτρων που εφαρμόζονται στον όγκο εισόδου. Δηλαδή ο αριθμός των χαρτών ενεργοποίησης αντιστοιχεί στον αριθμό των φίλτρων. Αν για παράδειγμα ο όγκος εισόδου είναι διαστάσεων $32 \times 32 \times 3$ και εφαρμοστούν δέκα φίλτρα συνέλιξης διαστάσεων $5 \times 5 \times 3$, ο όγκος εξόδου θα είναι διαστάσεων $28 \times 28 \times 3$ [σχήμα 2.22](#). Ο αριθμός των φίλτρων είναι μία παράμετρος, ή καλύτερα υπερ-παράμετρος των επιπέδων συνέλιξης.



Σχήμα 2.22: Αντιστοιχία του αριθμού των φίλτρων ενός επιπέδου συνέλιξης με το βάθος του όγκου στην έξοδο

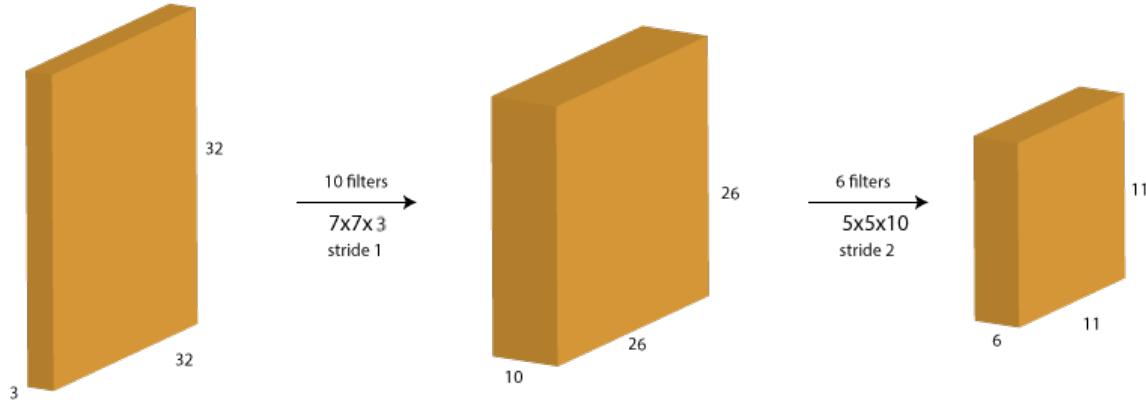
Ωστόσο, το βήμα μετατόπισης (stride) του φίλτρου πάνω στην είσοδο είναι και αυτό μία υπέρ-παράμετρος (hyperparameter) των επιπέδων συνέλιξης. Χρησιμοποιώντας βήμα μετατόπισης (S) διάφορο της μονάδας καταλήγουμε στην πιο κάτω εξίσωση για τον υπολογισμό του όγκου εξόδου:

$$\text{outsize} = (N - F)/S + 1$$

Ένα πρόβλημα που εμφανίζεται στην περίπτωση των μοντέλων CNN με μεγάλο αριθμό χρυφών επιπέδων είναι η γρήγορη μείωση των διαστάσεων μήκους και πλάτους του όγκου, το οποίο είναι αποτέλεσμα της διαδοχικής εφαρμογής πράξεων συνέλιξης ([σχήμα 2.23](#)).

Αυτή η συμπεριφορά είναι ανεπιθύμητη αφού περιορίζει και τις διαστάσεις των φίλτρων που μπορούμε να χρησιμοποιήσουμε σε κάθε επίπεδο CONV. Η χρήση

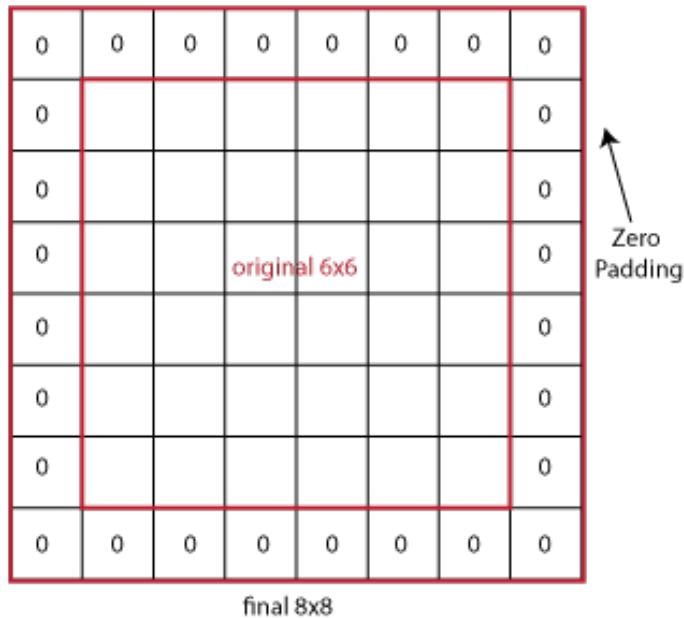
ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ



Σχήμα 2.23: Διαδοχικές εφαρμογές του τελεστή συνέλιξης προκαλούν μείωση των διαστάσεων μήκους και πλάτους του όγκου

φίλτρων μεγάλων διαστάσεων φέρει σαν αποτέλεσμα την γρήγορη μείωση των διαστάσεων του όγκου.

Για να αποτρέψουμε αυτή την συμπεριφορά μπορούμε να επεκτείνουμε τις διαστάσεις μήκους και πλάτους, προσθέτοντας μηδενικά στα σύνορα του όγκου εισόδου του εκάστοτε επιπέδου CONV. Η διαδικασία αυτή ονομάζεται *zero-padding* και φαίνεται στο [σχήμα 2.24](#). Το μέγεθος του συνόρου που προστίθεται είναι η τρίτη



Σχήμα 2.24: Zero Padding

υπέρ-παράμετρος ενός επιπέδου συνέλιξης.

Με την εισαγωγή της υπέρ-παραμέτρου zero-padding η εξίσωση υπολογισμού του όγκου εξόδου έχει την μορφή:

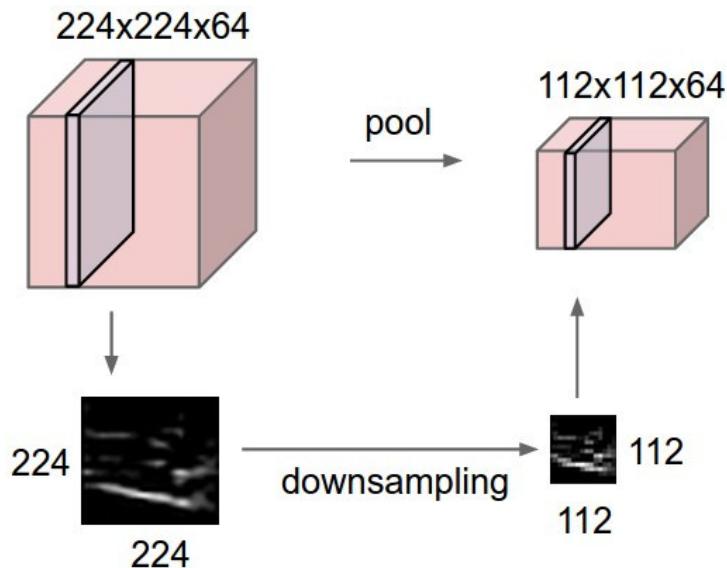
$$outsize = (N - F + 2P)/S + 1$$

Συνοψίζοντας, ένα επίπεδο συνέλιξης έχει τα εξής χαρακτηριστικά:

- Διαστάσεις όγκου εισόδου: $W_1 \times H_1 \times D_1$
- Hyperparameters:
 - K: Αριθμός φίλτρων
 - F: Μέγεθος του φίλτρου ($F \times F$)
 - S: Βήμα μετατόπισης
 - P: Ποσότητα zero-padding
- Διαστάσεις όγκου εξόδου: $W_2 \times H_2 \times D_2$, $D_2 = K$ όπου:
 - $W_2 = (W_1 - F)/S + 1$
 - $H_2 = (H_1 - F)/S + 1$
 - $D_2 = D_1$

2.3.2 Επίπεδο Υπό-δειγματοληψίας - Pooling layer

Συνήθως τα επίπεδα υπό-δειγματοληψίας προστίθενται στο δίκτυο, μεταξύ διαδοχικών επιπέδων συνέλιξης. Η λειτουργία τους είναι να μειώσουν τις χωρικές διαστάσεις των αναπαραστάσεων, μειώνοντας έτσι τον αριθμό των παραμέτρων και άρα τους υπολογισμούς που γίνονται στο νευρωνικό δίκτυο, ενεργώντας σαν μία συνάρτηση υποδειγματοληψίας (σχήμα 2.25).



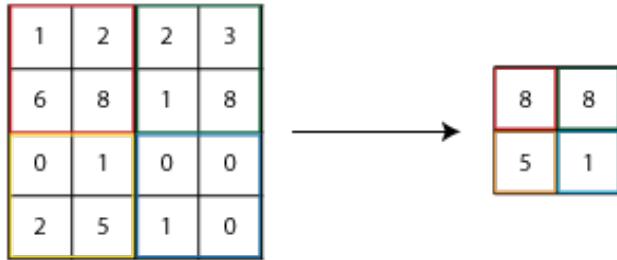
Σχήμα 2.25: Επίπεδο Υπό-δειγματοληψίας - Pooling layer

Πιθανές συναρτήσεις υπό-δειγματοληψίας είναι οι συναρτήσεις *max*, *average* και *L2-Norm*.

Στο σχήμα 2.26 βλέπουμε το αποτέλεσμα της εφαρμογής της συνάρτησης δειγματοληψίας $\text{max}(\vec{v})$ πάνω σε ένα πλέγμα διαστάσεων 4×4 .

Τα χαρακτηριστικά των συναρτήσεων υπό-δειγματοληψίας είναι:

ΚΕΦΑΛΑΙΟ 2. ΤΕΧΝΙΚΕΣ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΚΑΙ ΑΝΑΓΝΩΡΙΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΤΟΝ ΧΩΡΟ



Σχήμα 2.26: Συνάρτησης υπό-δειγματοληψίας Max - Max Pooling

- Διαστάσεις όγκου εισόδου: $W_1 \times H_1 \times D_1$
- Hyperparameters:
 - F: Χωρική τους έκταση ($F \times F$)
 - S: Βήμα μετατόπισης
- Διαστάσεις όγκου εξόδου: $W_2 \times H_2 \times D_2$, $D_2 = K$ όπου:
 - $W_2 = (W_1 - F)/S + 1$
 - $H_2 = (H_1 - F)/S + 1$
 - $D_2 = D_1$

2.3.3 Πλήρως Συνδεδεμένο Επίπεδο - Fully-connected layer

Ένα πλήρως συνδεδεμένο επίπεδο συνδέεται με όλους τους νευρώνες στο προηγούμενο επίπεδο, όπως γίνεται στα απλά μοντέλα NN (Πολυεπίπεδος Perceptron),

Συγκεκριμένα έχει τόσους νευρώνες όσες και οι αλάσεις της πρόβλεψης. Για παράδειγμα, ένα CNN που χρησιμοποιείται για αναγνώριση αντικειμένων σε εικόνες CIFAR-10 έχει το τελευταίο επίπεδο του πλήρως συνδεδεμένο και αποτελείται από 10 νευρώνες.

2.4 ΕΠΙΣΚΟΠΗΣΗ ΤΕΧΝΙΚΩΝ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΚΑΙ ΕΝΤΟΠΙΣΜΟ ΑΝΤΙΚΕΙΜΕΝΩΝ

Τόσο η αναγνώριση αντικειμένων (object recognition) όσο και ο εντοπισμός της θέσης των αντικειμένων αυτών (detection/localization) σε εικόνες, είναι μία ερευνητική περιοχή με τεράστιο ενδιαφέρον, η οποία απασχολεί πληθώρα ερευνητών. Η επιστήμη της Μηχανικής Όρασης (Computer Vision) στοχεύει στο να δώσει λύσεις στα συγκεκριμένα προβλήματα, εισάγοντας αναλυτικά ή και πιθανοτικά μαθηματικά μοντέλα.

Ο κλάδος της Βαθιάς Μηχανικής Μάθησης (Deep Learning - DL) [8] ανάγει το πρόβλημα της εύρεσης χαρακτηριστικών σημείων για την αναγνώριση αντικειμένων, στην εκμάθηση αναπαραστάσεων [12] με την χρήση Νευρωνικών Δικτύων

2.4. ΕΠΙΣΚΟΠΗΣΗ ΤΕΧΝΙΚΩΝ ΒΑΘΙΑΣ ΜΗΧΑΝΙΚΗΣ ΜΑΘΗΣΗΣ ΣΤΗΝ ΑΝΑΓΝΩΡΙΣΗ ΚΑΙ ΕΝΤΟΠΙΣΜΟ ΑΝΤΙΚΕΙΜΕΝΩΝ

Συνέλιξης. Οι πρώτες εφαρμογές Νευρωνικών Δικτύων Συνέλιξης, για την αναγνώριση αντικειμένων σε εικόνες, αναπτύχθηκαν το 1990 από τον Yann LeCun. Η πιο γνωστή και επιτυχής είναι το δίκτυο LeNet [13], το οποίο χρησιμοποιήθηκε για την αναγνώριση ψηφίων σε εικόνες. Ωστόσο, η εισαγωγή των CNN στον κλάδο της Μηχανικής Όρασης έγινε το 2012 με την ανάπτυξη του δικτύου AlexNet [3], από τους Alex Krizhevsky, Ilya Sutskever και Geoffrey E. Hinton. Το δίκτυο AlexNet χρησιμοποιήθηκε στον διαγωνισμό ImageNet ILSVRC challenge, το 2012, κερδίζοντας με διαφορά 10,9% στο σφάλμα αναγνώρισης αντικειμένων σε σύνολο 1000 κλάσεων. Με την εμφάνιση του δικτύου AlexNet, η ερευνητική κοινότητα ξεκίνησε να πιστεύει στην αποτελεσματικότητα των Νευρωνικών Δικτύων Συνέλιξης σε εφαρμογές αναγνώρισης αντικειμένων σε εικόνες. Συνέχεια στο έργο του Alex Krizhevsky δόθηκε το 2013, αναπτύσσοντας το ZF-Net [14] το οποίο είναι βασισμένο στην αρχιτεκτονική του δικτύου AlexNet.

Τα προαναφερθέντα μοντέλα Νευρωνικών Δικτύων Συνέλιξης δίνουν λύσεις μόνο στο πρόβλημα της αναγνώρισης και όχι του εντοπισμού της θέσης των αντικειμένων. Το 2013, ερευνητές εργαζόμενοι στην Google Inc. σχεδίασαν και υλοποίησαν ένα μοντέλο CNN το οποίο δίνει λύση στο πρόβλημα της ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες. Το μοντέλο αυτό, που φέρει το όνομα *DetectorNet*, είναι ομαδική εργασία των Christian Szegedy, Alexander Toshev και Dumitru Erhan [15]. Ωστόσο οι τεράστιες απαιτήσεις του *DetectorNet* σε υπολογιστικούς πόρους αποτελεί ένα από τα μειονεκτήματά του που το καθιστούν ακατάλληλο για εφαρμογή σε προβλήματα σχεδόν πραγματικού χρόνου όπως για παράδειγμα σε ένα ρομποτικό σύστημα.

Την ίδια χρονιά (2013), η ερευνητική ομάδα του πανεπιστημίου New York University (Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun) εμφανίστηκε στον διαγωνισμό ILSVRC challenge 2013 με ένα μοντέλο CNN το οποίο έχει την μορφή ενός απλού δικτύου με την ικανότητα εκμάθησης των πλαισίων (bounding boxes) στα οποία ανήκουν τα αντικείμενα που αναγνωρίζονται. Το μοντέλο φέρει το όνομα *OverFeat* [16] και ήταν ο νικητής του συγκεκριμένου διαγωνισμού στην κατηγορία εντοπισμού αντικειμένων.

Τα πιο πρόσφατα και πλέον δημοφιλή νευρωνικά δίκτυα για ταυτόχρονη αναγνώριση και εντοπισμό αντικειμένων σε εικόνες είναι τα μοντέλα ResNet [17], Fast-RCNN [18] και YOLO (You Only Look Once) [19], με τα τελευταία δύο να επικεντρώνονται στο πρόβλημα της γρήγορης εκτέλεσης. Συγκεκριμένα το δίκτυο YOLO έχει την ικανότητα να επεξεργαστεί εικόνες με ταχύτητα 45 fps (frames per second) σε μονάδα GPU (NVIDIA Titan X).

3

Εργαλεία Hardware και Software

Στο κεφάλαιο αυτό παρουσιάζονται τα βασικά εργαλεία που χρησιμοποιήθηκαν τόσο για τις υλοποιήσεις όσο και για τα πειράματα. Στο υποκεφάλαιο 3.1 παρουσιάζεται το ενσωματωμένο σύστημα της NVIDIA, Jetson TK1, ενώ τα εργαλεία λογισμικού που χρησιμοποιήθηκαν παρουσιάζονται στο υποκεφάλαιο 3.2.

3.1 NVIDIA JETSON TK1 DEVELOPMENT BOARD

Στο κεφάλαιο αυτό παρουσιάζεται η ενσωματωμένη πλατφόρμα Jetson TK1 της NVIDIA, η οποία χρησιμοποιήθηκε για εφαρμογή των υλοποιήσεων για Αναγνώριση και Εντοπισμό Αντικειμένων σε συστήματα πραγματικού χρόνου, χρησιμοποιώντας Νευρωνικά Δίκτυα Συνέλιξης.

Ο Tegra K1 είναι το πρώτο SOC της NVIDIA για φορητές συσκευές, με προηγμένη αρχιτεκτονική και χαρακτηριστικά, καθώς και χαμηλή κατανάλωση ισχύος. Η τυπική κατανάλωση ισχύος αναφέρεται στα 0.6 – 3 Watt ενώ η μέγιστη μπορεί να φθάσει μέχρι και τα 15 Watt σε κατάσταση μέγιστης λειτουργίας των μονάδων CPU, GPU και ISP.

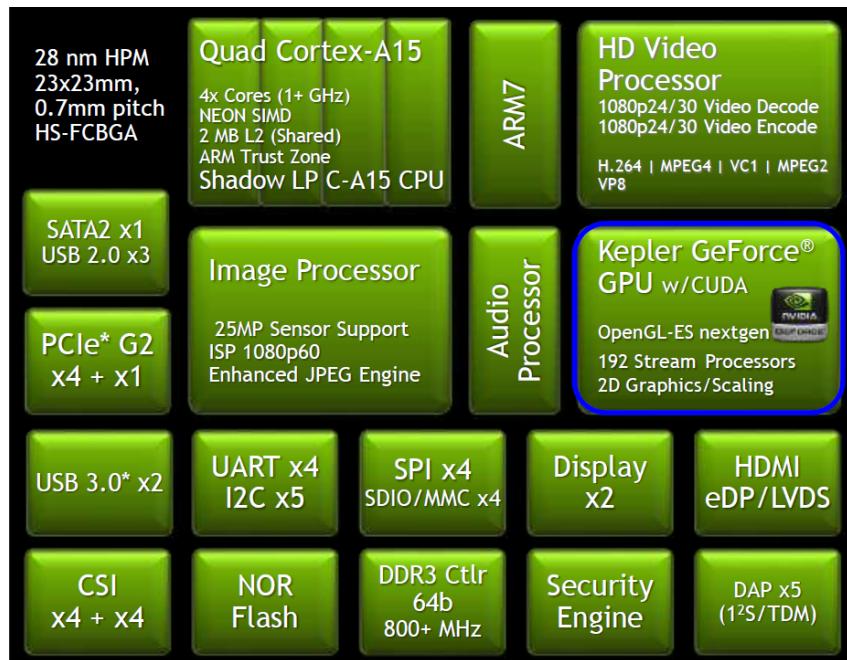
Ένα επίσης ενδιαφέρον χαρακτηριστικό του συγκεκριμένου SOC είναι η τιμή του λόγου της απόδοσης ανά μονάδα ισχύος κατανάλωσης (performance per watt), η οποία αναφέρεται στα $26SPGiga-flops/Watt$ ²

Όπως φαίνεται στο σχήμα 3.1, τα βασικά τεχνικά χαρακτηριστικά του Tegra K1 SOC είναι:

- CPU: Quad-core ARM Cortex-A15 CPU, 2.3Ghz
- GPU: GK20A Kepler-based GPU with 192 CUDA cores
- RAM: DDR3L/LPDDR3, up to 8GB

²Λόγος απόδοσης ανά μονάδα ισχύος: <http://tinyurl.com/lrl9glf>

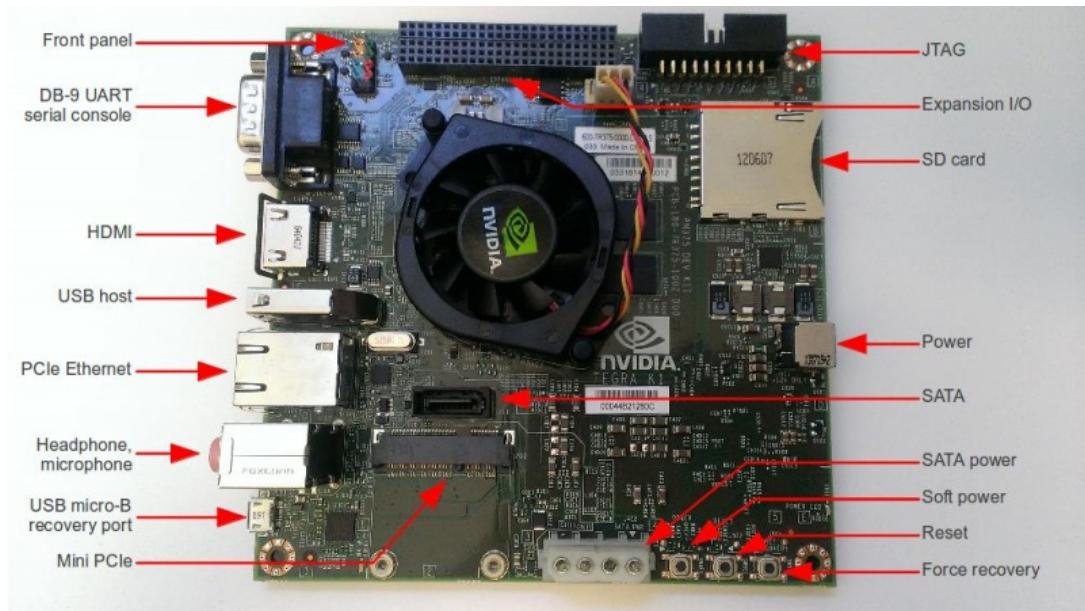
ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ HARDWARE KAI SOFTWARE



Σχήμα 3.1: Tegra K1 SOC

- Peripherals I/O: USB, eMMC/SD-card, LVDS, HDMI, SPI, UART, I2C, SATA, PCIe
- ISP: Image processor

Στα πλαισια της παρούσας διπλωματικής εργασίας, επιλέχθηκε η χρήση του *Jetson TK1* development board της NVIDIA, που φαίνεται στο [σχήμα 3.2](#). To Jetson



Σχήμα 3.2: Jetson TK1 development board

TK1 ενσωματώνει το Tegra K1 SOC (CPU+GPU+ISP) και είναι πλήρως συμβατό με

3.1. NVIDIA JETSON TK1 DEVELOPMENT BOARD

διάφορες διανομές λειτουργικών συστημάτων Linux (Ubuntu, Debian, Arch, Fedora, openSUSE, Gentoo). Η πλήρης συμβατότητα και υποστήριξη του λειτουργικού συστήματος Linux ήταν βασικό κριτήριο στην επιλογή του συγκεκριμένου ενσωματωμένου συστήματος, αφού επιτρέπει την εγκατάσταση εργαλείων με τον ίδιο τρόπο όπως σε ένα σταθερό υπολογιστικό σύστημα (Desktop PC) το οποίο τρέχει Linux OS. Πέρα από την συμβατότητα με κλασσικές διανομές Linux OS, η NVIDIA έχει αναπτύξει δικό της λειτουργικό σύστημα, *Linux4Tegra*, το οποίο έχει ως βάση τα Ubuntu-14.04, με κάποιες επεκτάσεις για πλήρη υποστήριξη του hardware του Jetson TK1. Επιπλέον, παράγοντας στην επιλογή της συγκεκριμένης πλατφόρμας είναι η πληθώρα των διεπαφών (interfaces) που διαθέτει, κάνοντάς την χρήσιμη για εφαρμογή σε ρομποτικά συστήματα όπου η σύνδεση διαφόρων περιφερειακών συσκευών, όπως για παράδειγμα αισθητήρες, κάμερες, κινητήρες, σερβο-κινητήρες, είναι απαίτηση. Πιο κάτω δίνονται οι βασικές διεπαφές που προσφέρει η πλατφόρμα Jetson TK1:

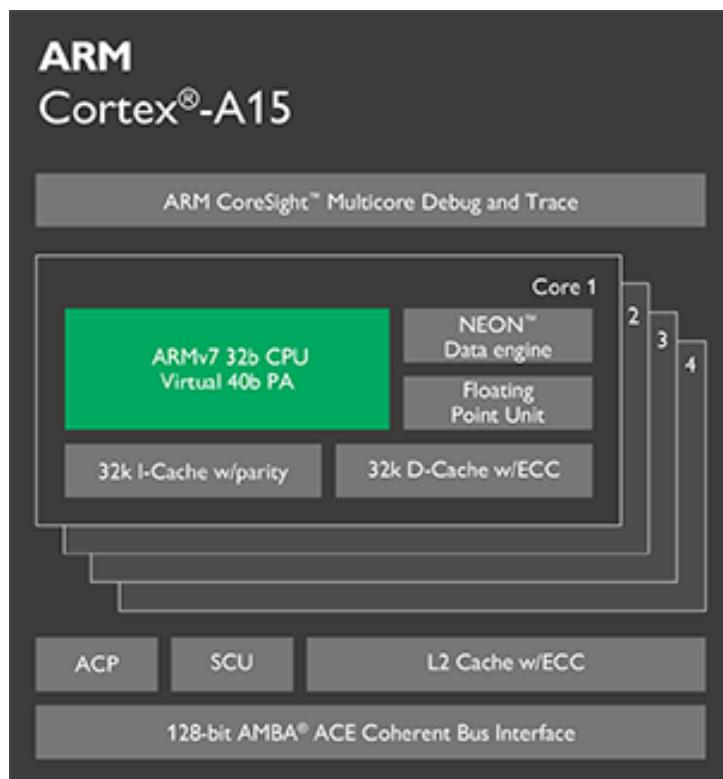
- mini-PCIe: Σύνδεση πρόσθετων συσκευών στον δίαυλο PCI-Express όπως, Wifi cards, SSD δίσκων, κτλ.
- USB 2.0 port: Σύνδεση συσκευών ή/και αισθητήρων με διεπαφή eHCI (Extended Host Controller Interface)
- USB 3.0 port: Σύνδεση συσκευών ή/και αισθητήρων με διεπαφή xHCI (eXtensible Host Controller Interface)
- HDMI: Δυνατότητα σύνδεσης οθόνης
- RS232 port: Παρόλο που το RS232 είναι αρκετά παλιό πρωτόκολλο, εξακολουθεί να ενσωματώνεται σε διάφορες συσκευές που δεν απαιτούν μεγάλο όγκο μεταφοράς δεδομένων, όπως οι οδηγοί κινητήρων
- Audio IO
- Gigabit Ethernet LAN: Δικτύωση της πλατφόρμας με τον "έξω" κόσμο
- SATA: Επιτρέπει την σύνδεση σκληρού δίσκου SATA
- JTAG port: Το JTAG προσφέρει την δυνατότητα σύνδεσης συσκευής/προγράμματος εντοπισμού σφαλμάτων (debugger), για HW-oriented αποσφαλμάτωση
- UART port
- I2C ports: Διαθέτει τρεις θύρες I2C για σύνδεση αισθητήρων/συσκευών που οδηγούνται με το συγκεκριμένο πρωτόκολλο
- GPIO: Προσφέρει δύο θύρες επέκτασης (expansion ports), με 50 και 75 ακροδέκτες αντίστοιχα. Χρήσιμο κυρίως για επικοινωνία συσκευών με SPI, οδήγηση σερβο-κινητήρων, διακλάδωση τροφοδοσίας σε τρίτες συσκευές, κτλ.

Η κατανάλωση ισχύος της πλατφόρμας Jetson TK1 εξαρτάται τόσο από την χρήση του SOC, αλλά και από την χρήση των περιφερειακών διεπαφών που διαθέτει. Η ζήτηση (κατα απόλυτο μέγεθος) σε ηλεκτρική ισχύ για το παρόν ενσωματωμένο σύστημα, σε περίπτωση πλήρους λειτουργίας του (περιλαμβανόμενης της χρήσης του διάυλου SATA & PCIe) αναφέρεται περίπου στα 58 Watt ³.

³http://elinux.org/Jetson/Jetson_TK1_Power#Typical_power_draw_of_Jetson_TK1

ΚΕΦΑΛΑΙΟ 3. ΕΡΓΑΛΕΙΑ HARDWARE KAI SOFTWARE

Όσον αφορά την υλοποίηση και ανάπτυξη Νευρωνικών Δικτύων σε ενσωματωμένα συστήματα, η πλατφόρμα Jetson TK1 θεωρείται ιδανική αφού υποστηρίζει CUDA και cuDNN. Η cuDNN (CUDA Deep Neural Network library) είναι GPU-accelerated βιβλιοθήκη για Νευρωνικά Δίκτυα, η οποία αναπτύχθηκε από την NVIDIA και προσφέρει υψηλού επιπέδου υλοποίησεις για ρουτίνες όπως συνέλιξη, κανονικοποίηση δεδομένων, pooling, επίπεδα ενεργοποίησης, κτλ. Η βιβλιοθήκη cuDNN χρησιμοποιείται και ενσωματώνεται στα πιο δημοφιλή εργαλεία σχεδίασης και υλοποίησης μοντέλων DNN, τα οποία και παρουσιάζονται στο [υποκεφάλαιο 3.2](#).



Σχήμα 3.3: ARM Cortex-A15 processor

Ένα από τα μειονεκτήματα της πλατφόρμας Jetson TK1 είναι η αρχιτεκτονική 32 bit του επεξεργαστή ARM Cortex-A15 MPCore που έχει ενσωματωμένο ([σχήμα 3.3](#)). Με την εμφάνιση των επεξεργαστών ARM Cortex-A57, οι οποίοι είναι αρχιτεκτονικής 64 bit, η υποστήριξη σε λογισμικό, τόσο από την πλευρά της NVIDIA όσο και από την ανοικτού-κώδικα (open-source) κοινότητα, μειώθηκε. Η NVIDIA σταμάτησε να υποστηρίζει μηχανήματα αρχιτεκτονικής 32 bit, τις βιβλιοθήκες CUDA και cuDNN, από την έκτη και δεύτερη έκδοση αντίστοιχα. Η CUDA είναι σήμερα στην έκδοση 8 και η cuDNN στην έκδοση 5, οι οποίες φέρουν αναβαθμίσεις στην επίδοση των αλγορίθμων βαθιάς εκμάθησης⁴. Στο [κεφάλαιο 6](#) γίνεται αναφορά στα προβλήματα που παρουσιάστηκαν εξαιτίας της ασυμβατότητας των συγκεκριμένων εργαλείων λογισμικού στο Jetson TK1.

⁴<https://developer.nvidia/cudnn>

3.2 ΕΡΓΑΛΕΙΑ ΛΟΓΙΣΜΙΚΟΥ

Εξαιτίας της ραγδαίας εξέλιξης της επιστήμης της βαθιάς μηχανικής μάθησης, τα τελευταία χρόνια έχουν αναπτυχθεί πολλά εργαλεία λογισμικού (βιβλιοθήκες, SDKs, frameworks) για γρήγορη ή/και αποτελεσματική σχεδίαση και υλοποίηση πολυεπίπεδων νευρωνικών δικτύων.

Μερικά από τα πιο γνωστά εργαλεία σχεδίασης και ανάπτυξης αλγορίθμων και μοντέλων πολυεπίπεδων νευρωνικών δικτύων, τα οποία χρησιμοποιούνται από την επιστημονική κοινότητα είναι:

- Caffe⁵ [20]: Framework για σχεδίαση, υλοποίηση και δοκιμή μοντέλων βαθιάς μηχανικής μάθησης. Είναι ένα από τα πρώτα ανοικτού κώδικα εργαλεία βαθιάς μηχανικής μάθησης που αναπτύχθηκαν.
- Tensorflow⁶ [21]: Ανοικτού κώδικα βιβλιοθήκη για αριθμητικούς υπολογισμούς χρησιμοποιώντας γράφους ροής δεδομένων ανεπτυγμένο από την ερευνητική ομάδα της Google. Η ευέλικτη αρχιτεκτονική της επιτρέπει την ανάπτυξη λογισμικού σε μία ή και περισσότερες κεντρικές μονάδες επεξεργασίας CPU ή μονάδες GPU.
- Theano⁷ [22][23][24]: Βιβλιοθήκη για Python η οποία επιτρέπει τον ορισμό, βελτιστοποίηση και αξιολόγηση μαθηματικών συναρτήσεων που αφορούν πράξεις με πολυδιάστατους πίνακες.
- Torch⁸ [25][26][27]: Framework για γρήγορη ανάπτυξη και δοκιμή αλγορίθμων μηχανικής μάθησης.
- Keras⁹ [28]: Ανοικτού κώδικα, υψηλού επιπέδου βιβλιοθήκη σε Python για σχεδίαση και ανάπτυξη πολυεπίπεδων νευρωνικών δικτύων. Ενσωματώνει και χρησιμοποιεί τις βιβλιοθήκες Theano και Tensorflow για την εκτέλεση των μαθηματικών εκφράσεων.
- CNTK¹⁰ [29]: Σετ από εργαλεία για εφαρμογές βαθιάς μηχανικής μάθησης, ανεπτυγμένο από την ερευνητική ομάδα της Microsoft.

Όλα τα προαναφερθέντα εργαλεία χρησιμοποιούν μοντέλα γράφων για την συμβολική αναπαράσταση των μαθηματικών υπολογισμών. Η αναπαράσταση σε μορφή γράφων έχει αποδειχθεί αποτελεσματική όσον αφορά την ταχύτητα υπολογισμών μαθηματικών εκφράσεων, αλλά και την απλότητα ανάπτυξης σχήμα 3.4.

Οι κόμβοι στον γράφο αναπαριστούν μαθηματικές πράξεις ή εκφράσεις, ενώ τα φύλλα αναπαριστούν πολυδιάστατους πίνακες δεδομένων, οι οποίοι ονομάζονται Tensors.

⁵<http://caffe.berkeleyvision.org/>

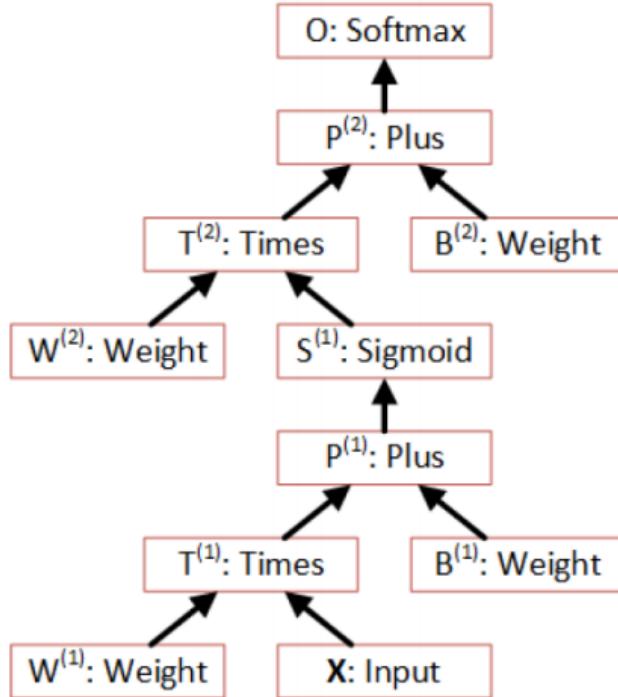
⁶<https://www.tensorflow.org/>

⁷<http://deeplearning.net/software/theano/>

⁸<http://torch.ch/>

⁹<https://keras.io/>

¹⁰<https://www.cntk.ai/>



Σχήμα 3.4: Παράδειγμα αναπαράστασης ενός μοντέλου νευρωνικού δικτύου σε γράφο. Το NN απειλείται από ένα κρυφό επίπεδο με συνάρτηση ενεργοποίησης την σιγμοειδή συνάρτηση και έναν ταξινομητή Softmax στο επίπεδο εξόδου

Η αναπαράσταση των νευρωνικών δικτύων σε μορφή γράφου ροής δεδομένων προσθέτει ένα επίπεδο ευελιξίας στην ανάπτυξη τους. Μία σημαντική παρατήρηση είναι ότι ο γράφος μπορεί χωριστεί σε μικρότερους υπογράφους και άρα να κατανεμηθεί σε περισσότερες από μία υπολογιστικές μονάδες (CPUs / GPUs) ή ακόμα και σε ετερογενή κατανεμημένα συστήματα (CPUs + GPUs) [30].

Ένα ακόμη αξιοσημείωτο εργαλείο είναι η βιβλιοθήκη *ConvNetJS* [31], η οποία αναπτύχθηκε από τον Andrej Karpathy και υποστηρίζεται απευθείας από σύγχρονους Web Browsers¹¹ ή/και σε υπολογιστικά συστήματα¹².

Στα πλαίσια της παρούσας διπλωματικής εργασίας επιλέχθηκε να χρησιμοποιηθεί η βιβλιοθήκη *Keras* για τους εξής λόγους:

- Προσφέρει υψηλού επιπέδου ρουτίνες για ανάπτυξη νευρωνικών δικτύων
- Εύκολη και γρήγορη σχεδίαση και ανάπτυξη
- Υποστηρίζει Νευρωνικά Δίκτυα Συνέλιξης
- Παρέχει πολλά παραδείγματα σχεδίασης και ανάπτυξης

¹¹Στον ιστοχώρο <http://cs.stanford.edu/people/karpathy/convnetjs/> υπάρχουν παραδείγματα μοντέλων νευρωνικών δικτύων τα οποία μπορεί ο αναγνώστης να δοκιμάσει κατευθείαν μέσα από τον browser του

¹²Η βιβλιοθήκη ConvNetJS μπορεί να τρέξει σε συστήματα τα οποία υποστηρίζουν Node.js (server-side javascript interpreter)

- Τρέχει σε CPU και GPU
- Επιτρέπει την επιλογή μεταξύ των βιβλιοθηκών Theano και Tensorflow για την εκτέλεση των μαθηματικών εκφράσεων
- Είναι από τα πιο ενεργά προγράμματα ανάπτυξης λογισμικού για DL

Το τελευταίο σημείο είναι σημαντικό, αφού επιτρέπει την ανάπτυξη μοντέλων και την αξιολόγηση της απόδοσή τους, κυρίως σε χρόνο εκτέλεσης, με χρήση τόσο του Theano αλλά και του Tensorflow, χωρίς να χρειαστεί επαναπρογραμματισμός.

Η πλήρης λίστα των εργαλείων λογισμικού που χρησιμοποιήθηκαν παρουσιάζεται πιο κάτω:

- Γλωσσα προγραμματισμού Python2.7
- Keras: Σχεδίαση και ανάπτυξη των CNN
- Theano: Keras backend που χρησιμοποιείται για τους υπολογισμούς των μαθηματικών εκφράσεων
- Matplotlib¹³: Βιβλιοθήκη Python για κατασκευή διαγραμμάτων
- PyDot¹⁴: Βιβλιοθήκη Python για κατασκευή γράφων

¹³<http://matplotlib.org/>

¹⁴<https://pypi.python.org/pypi/pydot>

4

Υλοποιήσεις

Η παρούσα διπλωματική εργασία καταπιάνεται με το πρόβλημα της ανάπτυξης νευρωνικών δικτύων στην πλατφόρμα Jetson TK1 και στοχεύει στην ανάπτυξη ενός νευρωνικού δικτύου για ταυτόχρονη αναγνώριση και εντοπισμό αντικειμένων (object detection) σε εικόνες.

Ένα CNN αποτελείται από πολλά επίπεδα συνέλιξης, τα οποία με την σειρά τους αποτελούνται από εκατομμύρια νευρώνες το κάθε ένα. Αυτό σημαίνει και εκατομμύρια παραμέτρους όπου η καθεμία είναι ένας αριθμός κινητής υποδιαστολής 32 bit. Το ενσωματωμένο σύστημα Jetson TK1 έχει περιορισμένη χωρητικότητα μνήμης (2GB), γεγονός που λήφθηκε υπόψη κατά την επιλογή των μοντέλων CNN που υλοποιήθηκαν.

Οι υλοποιήσεις χωρίζονται σε 2 μέρη:

- Ανάπτυξη διαφόρων μοντέλων CNN για αναγνώριση αντικειμένων σε εικόνες
- Ρυθμίσεις και βελτιστοποιήσεις στο Jetson TK1

Στο πρώτο μέρος παρουσιάζονται και αναλύονται τα μοντέλα CNN που αναπτύχθηκαν, ενώ στο δεύτερο δίνεται μία πλήρης περιγραφή των διαδικασιών ρύθμισης και βελτιστοποίησης που έγιναν στο ενσωματωμένο σύστημα Jetson TK1 καθώς και τα εργαλεία που χρειάστηκε να εγκατασταθούν.

4.1 ΜΟΝΤΕΛΑ CNN

Όπως αναφέρθηκε στο [υποκεφάλαιο 3.2](#), το βασικό εργαλείο λογισμικού που χρησιμοποιήσαμε για την ανάπτυξη των μοντέλων CNN είναι το Keras.

Η βιβλιοθήκη Keras προσφέρει τις υλοποιήσεις όλων των επιπέδων που απαιτούνται για για την ανάπτυξη ενός CNN¹⁵. Πιο συγκεκριμένα, τα επίπεδα που χρησιμοποιήθηκαν, καθώς και οι βασικές παράμετροι τους περιγράφονται πιο κάτω:

¹⁵Πλήρες περιγραφή της λίστας των διαθέσιμων επιπέδων: <https://keras.io/layers/core/>

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΕΙΣ

- InputLayer: Επίπεδο Εισόδου του νευρωνικού δικτύου
 - Διαστάσεις του όγκου εισόδου (tensor shape)
- Convolution2D: Επίπεδο Συνέλιξης
 - Μορφολογία της εισόδου
 - Αριθμός των φίλτρων συνέλιξης
 - Διαστάσεις των φίλτρων συνέλιξης
 - Συνάρτηση ενεργοποίησης
- MaxPooling2D: Επίπεδο Υποδειγματοληψίας
 - Διαστάσεις πλαισίου
 - Βήμα μετατόπισης
- ZeroPadding2D: Προσθέτει πλαίσιο με μηδενικά στον όγκο εισόδου
 - Διαστάσεις του πλαισίου
- Activation: Επίπεδο ενεργοποίησης. Εφαρμόζει συνάρτηση ενεργοποίησης στον όγκο εξόδου του προηγούμενου επιπέδου
- Dropout: Επίπεδο πρόληψης υπέρ-προσαρμογής [32]
- Dense: Πλήρες συνδεδεμένο επίπεδο
 - Διαστάσεις του όγκου εισόδου (Προαιρετικό)
 - Διαστάσεις του όγκου εξόδου
 - Συνάρτηση ενεργοποίησης
- Flatten: Μετασχηματίζει τον όγκο εισόδου σε επίπεδη αναπαράσταση (π.χ. για όγκο εισόδου $64 \times 32 \times 32$ η έξοδος θα είναι επίπεδη με 65536 νευρώνες)
- BatchNormalization: Εφαρμόζει μετασχηματισμό για να διατηρήσει την μέση τιμή και την τυπική απόκλιση των ενεργοποιήσεων του προηγούμενου επιπέδου στις τιμές 0 και 1 αντίστοιχα

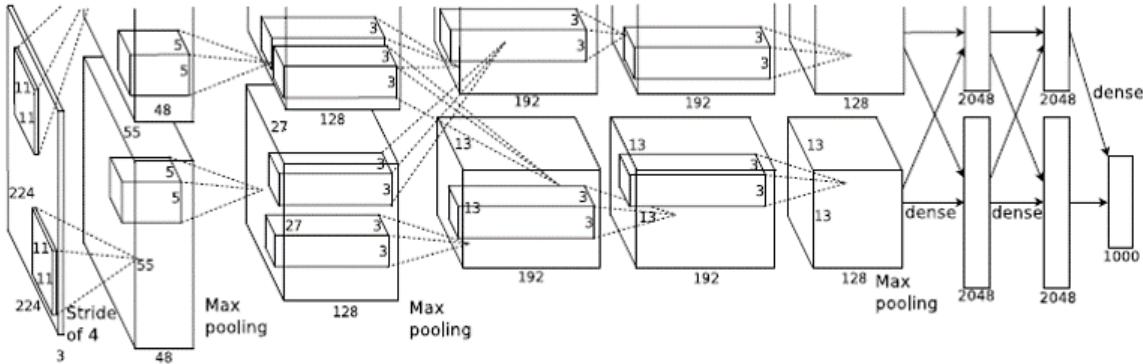
Επειδή η επιστήμη της βαθιάς μηχανικής μάθησης βρίσκεται σε πρώιμο στάδιο, δεν υπάρχουν συγκεντρωμένες οι υλοποιήσεις των διαφόρων επιπέδων και γενικότερα των μοντέλων σύγχρονων ANN.

Επιπρόσθετα, η επιλογή των μοντέλων CNN στηρίχθηκε στην ύπαρξη προ-εκπαίδευμένων βαρών για τα αντίστοιχα CNN στο διαδίκτυο για 2 λόγους:

- Η διαδικασία εκπαίδευσης είναι χρονοβόρα διαδικασία και προϋποθέτει τη χρήση μίας ή περισσοτέρων ισχυρών μονάδων GPU (NVIDIA Titan X GPU)
- Η εκπαίδευση νευρωνικών δικτύων ξεφεύγει από τα πλαίσια της παρούσας διπλωματικής εργασίας

4.1.1 AlexNet

Το δίκτυο AlexNet ήταν η αρχή της εισαγωγής της βαθιάς μηχανικής μάθησης στην επιστήμη της μηχανικής όρασης. Εμφανίστηκε και χρησιμοποιήθηκε στον διαγωνισμό ImageNet ILSVRC challenge, το 2012, κερδίζοντας με διαφορά 10,9%, στο σφάλμα αναγνώρισης αντικειμένων σε σύνολο 1000 κλάσεων.



Σχήμα 4.1: Μοντέλο δικτύου AlexNet [3]

Το δίκτυο AlexNet αποτελείται από σύνολο οκτώ επιπέδων, ή καλύτερα ομάδες επιπέδων – πέντε (5) επίπεδα συνέλιξης και τρία (3) πλήρως συνδεδεμένα (βλ. πίνακα 4.1).

Πίνακας 4.1: Χαρακτηριστικά και παράμετροι των επιπέδων του δικτύου AlexNet

| Επίπεδο | Τύπος | Αριθμός καναλιών | Διαστάσεις φίλτρων |
|---------|----------------|------------------|--------------------|
| 1 | Conv+Pool+Norm | 96 | 11 × 11 |
| 2 | Conv+Pool+Norm | 256 | 5 × 5 |
| 3 | Conv | 384 | 3 × 3 |
| 4 | Conv | 384 | 3 × 3 |
| 5 | Conv+Pool | 256 | 3 × 3 |
| 6 | Full | 4096 | N/A |
| 7 | Full | 4096 | N/A |
| 8 | Full | 1000 | N/A |

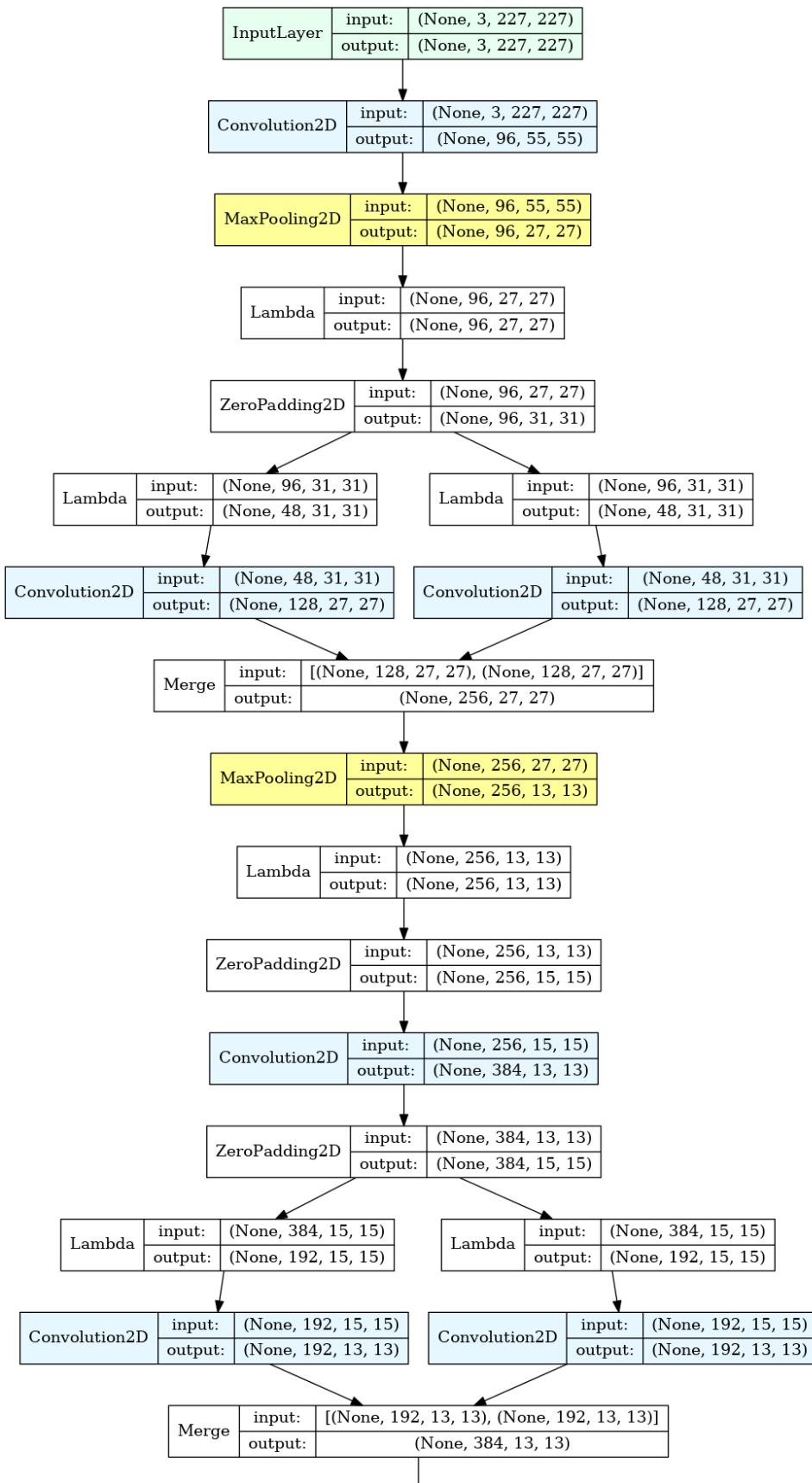
Η συνάρτηση υποδειγματοληψίας που χρησιμοποιεί το μοντέλο είναι η συνάρτηση MaxPooling2D με βήμα μετατόπισης ίσο με την μονάδα και στους 2 άξονες (1×1). Σε όλα τα επίπεδα εκτός του τελευταίου χρησιμοποιήθηκε η συνάρτηση ενεργοποίησης ReLU. Επίσης χρησιμοποιεί και πολλά επίπεδα ZeroPadding με πλαίσιο διαστάσεων 1×1 . Το τελευταίο επίπεδο παίζει τον ρόλο του ταξινομητή και συγκεκριμένα είναι ένας ταξινομητής Softmax.

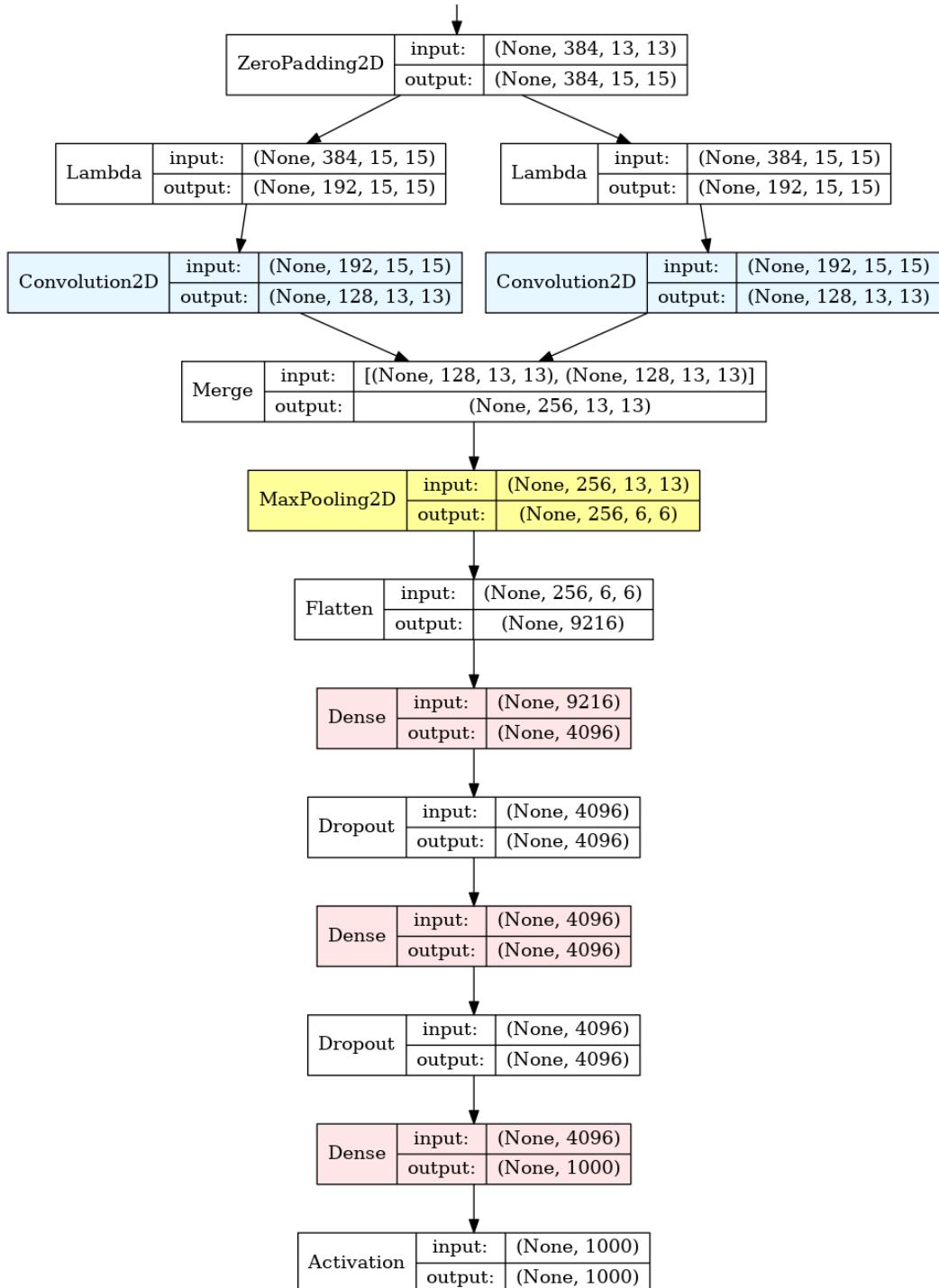
Η υλοποίηση στηρίχτηκε στην αντίστοιχη που υπάρχει με το εργαλείο Caffe¹⁶. Πρακτικά μεταφράστηκε ο πηγαίος κώδικας της υλοποίησης από το Caffe στο

¹⁶Υλοποίηση του δικτύου AlexNet στο Caffe: https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΕΙΣ

Keras. Ο αντίστοιχος γράφος της υλοποίησης του μοντέλου φαίνεται φαίνεται στο [σχήμα 4.2](#).



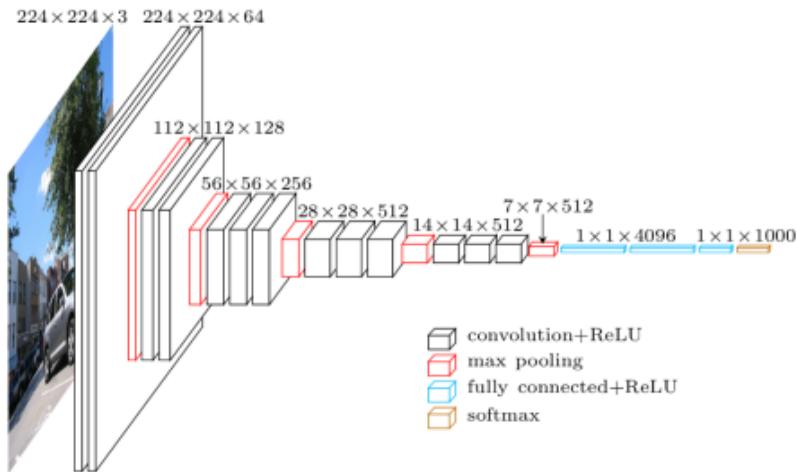


Σχήμα 4.2: Πλήρης μορφή του δικτύου AlexNet της υλοποίησης

4.1.2 VGG16

Το δίκτυο VGG16 ήταν ο νικητής του διαγωνισμού ImageNet ILSVRC-2014 με σφάλμα κατηγοριοποίησης 7.5% σε σύνολο 1000 κλάσεων αντικειμένων [33].

Αποτελείται από 16 επίπεδα – 13 επίπεδα συνέλιξης και 3 πλήρως συνδεδεμένα (σχήμα 4.3) – των οποίων τα χαρακτηριστικά δίνονται στον πίνακα 4.2.



Σχήμα 4.3: Μορφή του δικτύου VGG16

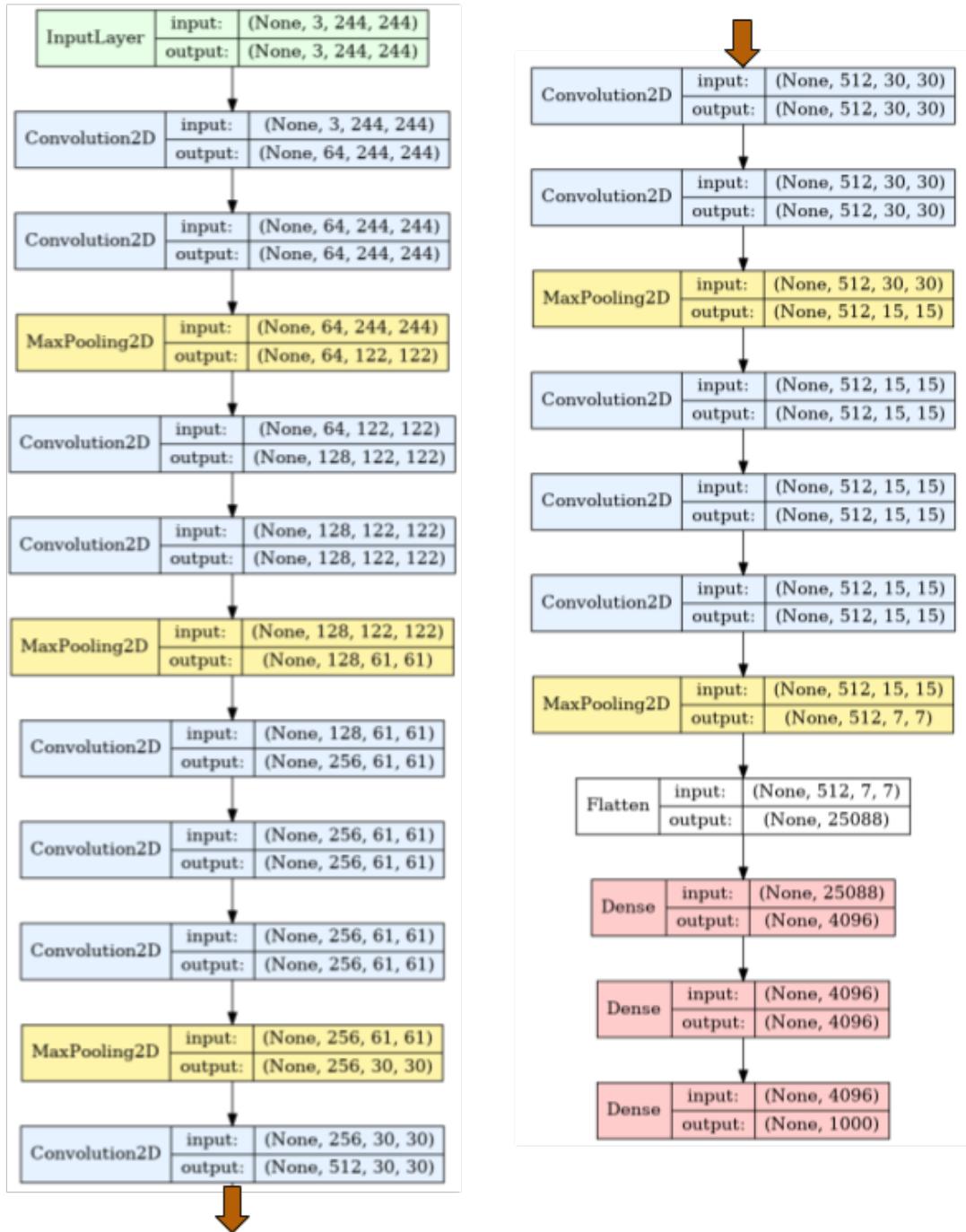
Πίνακας 4.2: Χαρακτηριστικά και παράμετροι των επιπέδων του δικτύου VGG16

| Επίπεδο | Τύπος | Αριθμός καναλιών | Διαστάσεις φίλτρων |
|---------|----------------|------------------|--------------------|
| 1 | Conv | 64 | 3×3 |
| 2 | Conv+Pool | 64 | 3×3 |
| 3 | Conv | 128 | 3×3 |
| 4 | Conv+Pool | 128 | 3×3 |
| 5 | Conv | 256 | 3×3 |
| 6 | Conv | 256 | 3×3 |
| 7 | Conv+Pool | 256 | 3×3 |
| 8 | Conv | 512 | 3×3 |
| 9 | Conv | 512 | 3×3 |
| 10 | Conv+Pool | 512 | 3×3 |
| 11 | Conv | 512 | 3×3 |
| 12 | Conv | 512 | 3×3 |
| 13 | Conv+Pool | 512 | 3×3 |
| 14 | FullyConnected | 4096 | N/A |
| 15 | FullyConnected | 4096 | N/A |
| 16 | FullyConnected | 1000 | N/A |

Παρομοίως με το δίκτυο AlexNet, σε όλα τα επίπεδα εκτός του τελευταίου χρησιμοποιήθηκε η συνάρτηση ενεργοποίησης ReLU, ενώ το τελευταίο επίπεδο παίζει τον ρόλο του ταξινομητή και συγκεκριμένα είναι ένας ταξινομητής Softmax. Το βήμα μετατόπισης των συναρτήσεων υποδειγματοληψίας είναι διαστάσεων 1×1 .

Το μοντέλο του δικτύου VGG16 υπάρχει υλοποιημένο στην λίστα με τα παραδείγματα του εργαλείου Keras¹⁷. Ο αντίστοιχος γράφος της υλοποίησης του μοντέλου φαίνεται στο σχήμα 4.4.

¹⁷<https://github.com/fchollet/keras/tree/master/keras/applications>



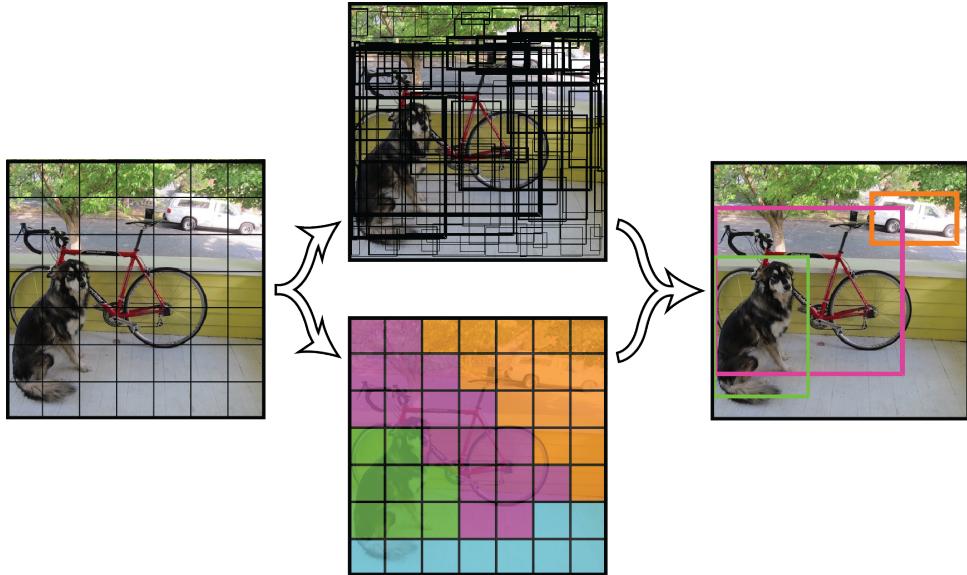
Σχήμα 4.4: Πλήρης μορφή του δικτύου VGG16 της υλοποίησης

4.1.3 YOLO Net

Το δίκτυο YOLO (You Only Look Once), είναι το πρώτο μοντέλο νευρωνικού δικτύου συνέλιξης το οποίο προσπαθεί να επιλύσει το πρόβλημα της ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες, με ένα προς-τα-εμπρός πέρασμα (forward pass). Η ιδιαιτερότητά του είναι ότι αντιμετωπίζει το πρόβλημα σαν ένα πρόβλημα regression και όχι classification.

Μία ακόμη ιδιαιτερότητα του συγκεκριμένου δικτύου είναι ότι θέτει σαν απαίτηση την εφαρμογή του σε προβλήματα σχεδόν πραγματικού χρόνου και άρα στοχεύει κυρίως στην ταχύτητα της αναγνώρισης. Φυσικά αυτό έχει σαν αποτέλεσμα την μείωση της ακρίβειας αναγνώρισης, η οποία είναι χαμηλότερη σε σχέση με άλλα μοντέλα, όπως για παράδειγμα τα δίκτυα Fast-RCNN [18], Overfeat και DetectorNet.

Η έξοδος του δικτύου αντιστοιχεί τόσο στις κλάσεις των αντικειμένων που αναγνωρίστηκαν, καθώς και στις συντεταγμένες όπου αυτά εντοπίστηκαν.



Σχήμα 4.5: Παράδειγμα τρόπου λειτουργίας του δικτύου YOLO

Το βασικό μοντέλο του δικτύου YOLO έχει την δυνατότητα να επεξεργάζεται εικόνες με ταχύτητα 45 fps χρησιμοποιώντας μονάδα GPU (Nvidia Titan X).

Λόγω της ιδιαιτερότητας του συγκεκριμένου δικτύου, όσον αφορά τον τρόπο προσέγγισης του προβλήματος της ταυτόχρονης αναγνώρισης και εντοπισμού των αντικειμένων, θεωρούμε σημαντικό να αναφέρουμε και να εξηγήσουμε τον τρόπο λειτουργίας του. Το πρώτο βήμα είναι να χωρίσει την εικόνα εισόδου σε ένα πλέγμα διαστάσεων $S \times S$ (αριστερή εικόνα στο [σχήμα 4.5](#)). Κάθε κελί του πλέγματος προβλέπει B οριοθετημένα πλαίσια (bounding boxes) μαζί με ένα σκορ "εμπιστοσύνης" για το κάθε πλαίσιο (πάνω μεσαία εικόνα στο [σχήμα 4.5](#)). Το σκορ εμπιστοσύνης ερμηνεύεται ως η βεβαιότητα να ανήκει ένα αντικείμενο στο συγκεκριμένο πλαίσιο μαζί με την ακρίβεια ότι το συγκεκριμένο αντικείμενο ανήκει σε αυτό το πλαίσιο. Η μαθηματική έκφραση ορίζεται ως:

$$\text{confidence} = P(\text{object} | \text{box} = i) * IOU_{\text{pred}}^{\text{truth}}$$

Από την παραπάνω μαθηματική έκφραση φαίνεται ότι σε περίπτωση που σε ένα κελί δεν υπάρχει αντικείμενο, το σκορ εμπιστοσύνης μηδενίζεται. Επίσης, κάθε κελί του πλέγματος προβλέπει και τις πιθανότητες της κλάσης του αντικειμένου, $P(\text{class}_i | \text{object})$ (κάτω μεσαία εικόνα στο [σχήμα 4.5](#)).

Χρησιμοποιώντας τις 2 αυτές μαθηματικές εκφράσεις των υπό-συνθήκη πιθα-

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΕΙΣ

νοτήτων, καταλήγουμε στην σχέση

$$\text{class} - \text{confidence} = P(\text{class}_i | \text{object}) * P(\text{object} | \text{box} = i) * IOU_{\text{pred}}^{\text{truth}} = \\ P(\text{class}_i) * IOU_{\text{pred}}^{\text{truth}}$$

η οποία μας δίνει το σκορ εμπιστοσύνης για μία συγκεκριμένη κλάση.

Κάθε οριοθετημένο πλαίσιο αποτελείται από πέντε προβλέψεις; x, y, w, h και το σκορ εμπιστοσύνης. Τα x και y αντιστοιχούν στις συντεταγμένες του κέντρου του πλαισίου ενώ οι τιμές w και h αναφέρονται στο μήκος και το πλάτος του.

Τέλος, οι προβλέψεις στην έξοδο του δικτύου φαίνονται σαν ένας όγκος διαστάσεων

$$S \times S \times (B * 5 + C)$$

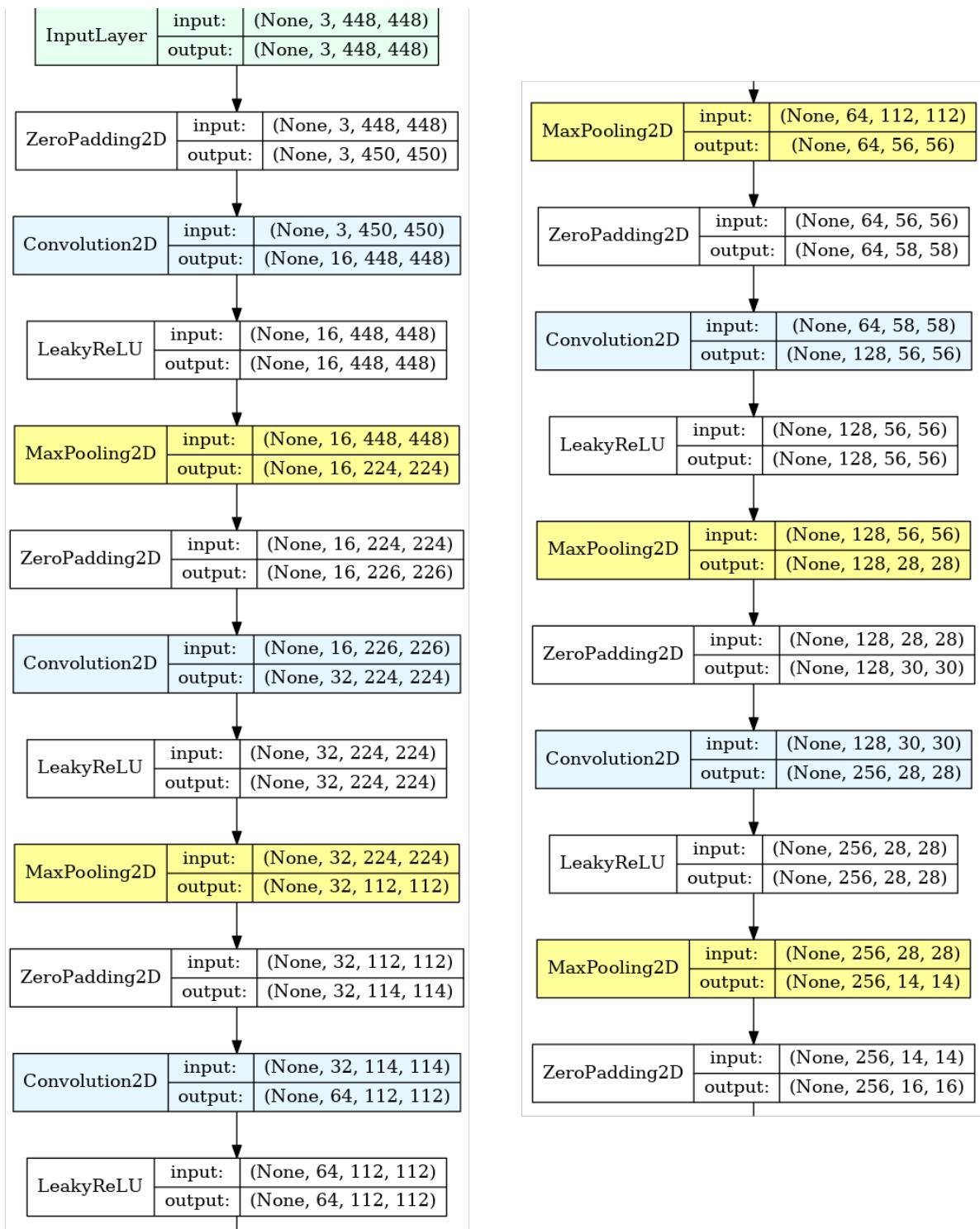
Το πλήρες μοντέλο YOLO αποτελείται από 24 επίπεδα συνέλιξης και 2 πλήρως συνδεδεμένα.

Πέρα από το πλήρες μοντέλο, έχει σχεδιαστεί και εκπαιδευτεί, από την ίδια ομάδα ερευνητών, ένα μοντέλο με παρόμοιο τρόπο λειτουργίας αλλά με πολύ λιγότερα επίπεδα συνέλιξης και άρα πιο γρήγορο στην εκτέλεση (Tiny/Fast YOLO). Το TinyYOLO αποτελείται από 9 επίπεδα συνέλιξης και 2 πλήρως συνδεδεμένα. Σε όλα τα επίπεδα εκτός του τελευταίου χρησιμοποιείται η συνάρτηση ενεργοποίησης *LeakyReLU* που παρουσιάστηκε στην [ενότητα 2.2.1](#).

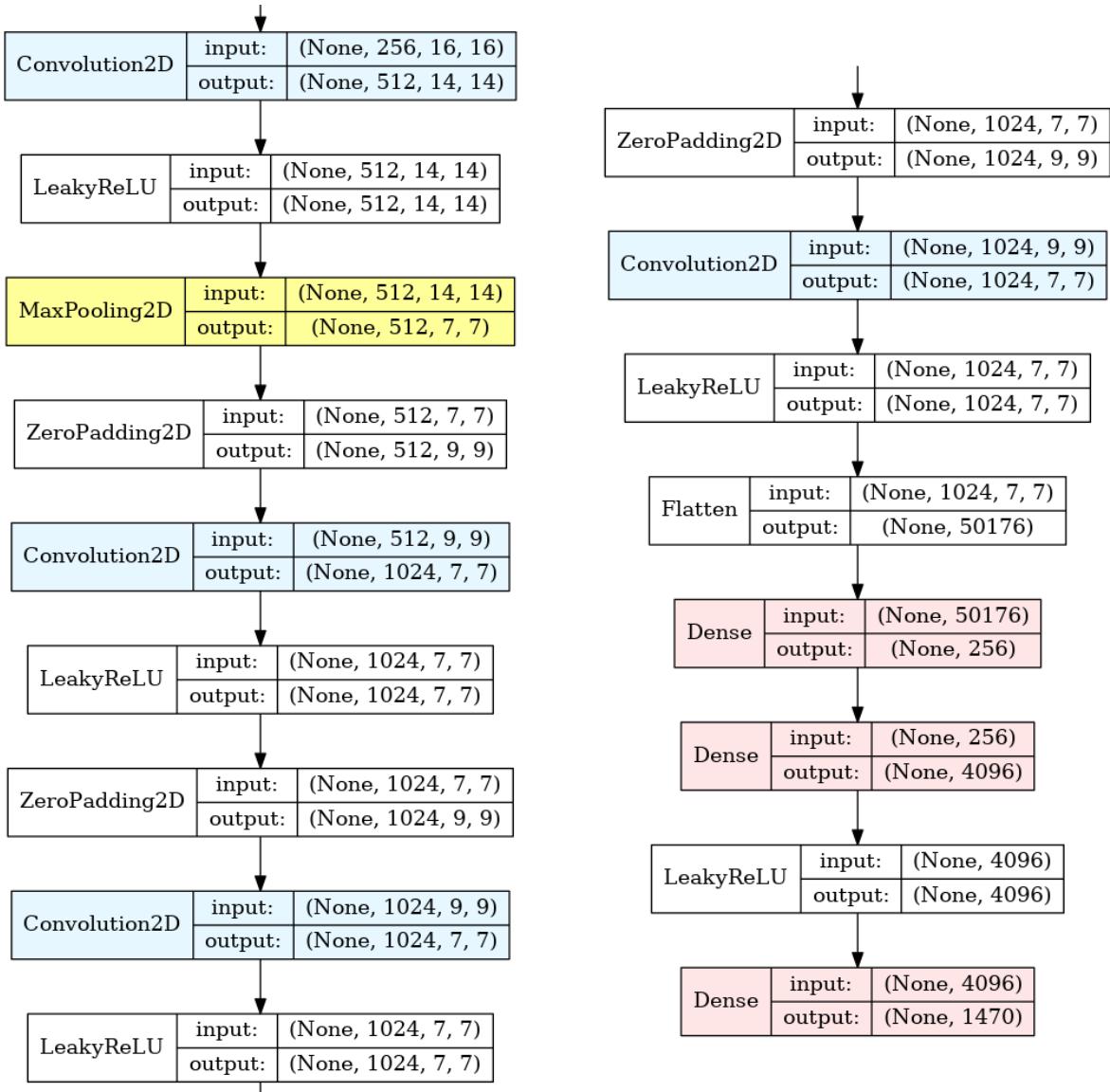
Η υλοποίηση του μοντέλου Tiny-YOLO είναι βασισμένη στην αντίστοιχη που έγινε από την συγκεκριμένη ερευνητική ομάδα [34], η οποία είναι γραμμένη σε γλώσσα προγραμματισμού C^{18} . Ο γράφος της υλοποίησης του δικτύου Tiny-YOLO στο Keras φαίνεται στο [σχήμα 4.6](#).

¹⁸Βασική υλοποίηση του δικτύου YOLO: <http://pjreddie.com/darknet/>

4.1. MONTELA CNN



ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΕΙΣ



Σχήμα 4.6: Πλήρης μορφή του δικτύου Tiny-YOLO της υλοποίησης

4.2 ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΤΩΝ ΕΡΓΑΛΕΙΩΝ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ JETSON TK1

Στο [υποκεφάλαιο 3.1](#) αναφέρθηκε ότι η πλατφόρμα διαθέτει λειτουργικό *Linux4Tegra* το οποίο είναι ουσιαστικά διανομή Ubuntu 14.04 με εγκατεστημένους τους απαραίτητους drivers για το συγκεκριμένο σύστημα.

Ένα από τα πρώτα βήματα ήταν η απεγκατάσταση από το λειτουργικό το γραφικό περιβάλλον, για να ελευθερωθεί μνήμη αφού η χωρητικότητα μνήμης του ενσωματωμένου συστήματος Jetson TK1 είναι περιορισμένη στα 2 Gigabytes και μάλιστα είναι κοινή μεταξύ των μονάδων GPU και CPU (shared memory). Το γραφικό περιβάλλον που είναι προ-εγκατεστημένο στην διανομή Linux4Tegra απαιτεί γύρω στα 114 Megabytes μνήμη RAM. Το Jetson TK1 συνδέθηκε μέσω θύρας ethernet στο δί-

4.2. ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΤΩΝ ΕΡΓΑΛΕΙΩΝ ΛΟΓΙΣΜΙΚΟΥ ΣΤΟ JETSON TK1

κτυο ενός σταθερού υπολογιστή και έτσι ο χειρισμός του έγινε μέσω καναλιού ssh¹⁹, κερδίζοντας έτσι 114 Megabytes σε μνήμη. Αυτό το κέρδος, όπως θα φανεί αργότερα, έπαιξε σημαντικό ρόλο, καθώς τα νευρωνικά δίκτυο που υλοποιήθηκαν απαιτούν αρκετή μνήμη κατά την διάρκεια εκτέλεσής τους (της τάξης των Gigabytes).

Πιο κάτω παρατίθενται οι βασικές τροποποιήσεις και ρυθμίσεις που έγιναν στο λειτουργικό σύστημα:

- Απενεργοποίηση της θύρας HDMI (κέρδος 0.6 Watt σε κατανάλωση ισχύος - αναφέρεται στο τεχνικό εγχειρίδιο)
- Ρύθμιση λειτουργίας των πυρήνων του επεξεργαστή σε κατάσταση μέγιστης απόδοσης
- Ρύθμιση λειτουργίας της μονάδας GPU σε κατάσταση μέγιστης απόδοσης

Στην συνέχεια εγκαταστάθηκαν οι βιβλιοθήκες CUDA και cuDNN για να είναι δυνατή η εκμετάλλευση της υπολογιστικής ισχύς της μονάδας GPU²⁰.

Ένα μεγάλο μέρος της συνεισφοράς της παρούσας διπλωματικής εργασίας ήταν να γίνουν βελτιστοποιήσεις σε επίπεδο λογισμικού των εργαλείων που χρησιμοποιήθηκαν για το συγκεκριμένο μηχάνημα.

Αρχικά, ακολουθήθηκαν οι οδηγίες εγκατάστασης των εργαλείων *numpy*, *scipy*, *Theano* και *Keras*. Όλα τα προαναφερθέντα πακέτα εγκαταστάθηκαν μέσα από τον επίσημο διαχειριστή πακέτων της Python, *pip*. Τα πρώτα αποτελέσματα σε χρόνους εκτέλεσης πράξεων γραμμικής άλγεβρας ήταν απαγορευτικά (σχεδόν μία τάξη μεγέθους κάτω από το αναμενόμενο). Τόσο η βιβλιοθήκη Theano, όσο και η *numpy*, χρησιμοποιούν εξωτερικές βιβλιοθήκες για πράξεις γραμμικής άλγεβρας, οι οποίες ονομάζονται BLAS (Basic Linear Algebra Subprograms).

Αναγνωρίστηκε πρόβλημα στους υπολογισμούς πράξεων γραμμικής άλγεβρας, αφού σχεδόν όλες οι μαθηματικές εκφράσεις που εκτελούνται σε ένα νευρωνικό δίκτυο είναι πράξεις πινάκων και εκτελέστηκε profiling των ρουτινών BLAS χρησιμοποιώντας και τις τρεις βιβλιοθήκες που είναι διαθέσιμες για επεξεργαστές ARM οι οποίες είναι οι εξής:

- libblas-dev + liblapack3 + libumfpack5.6.2: Προ-εγκατεστημένες στην διανομή Linux4Tegra
- ATLAS: Ανοικτού κώδικα
- OpenBLAS: Ανοικτού κώδικα

Τόσο η βιβλιοθήκη ATLAS, όσο και η OpenBLAS, μεταγλωτίστηκαν από πηγαίο κώδικα, λαμβάνοντας έτσι υπόψη τις απαραίτητες βελτιστοποιήσεις κατά την διαδικασία της μεταγλώτισης. Συγκεκριμένα ορίστηκε στον μεταγλωτιστή η χρήση της αρχιτεκτονικής του συγκεκριμένου επεξεργαστή (armv7 - Cortex-A15) και των μονάδων NEON²¹ που διαθέτει για τον υπολογισμό πράξεων με αριθμούς κινητής

¹⁹SSH: Secure Shell <https://tools.ietf.org/html/rfc4254>

²⁰Οδηγός εγκατάστασης εργαλείων CUDA και cuDNN: http://elinux.org/Jetson_TK1

²¹url`https://www.arm.com/products/processors/technologies/neon.php`

ΚΕΦΑΛΑΙΟ 4. ΥΛΟΠΟΙΗΣΕΙΣ

υποδιαστολής. Επίσης ορίστηκε η χρηση τεσσάρων νημάτων (threads) για την εκτέλεση των ρουτινών BLAS.

Στην συνέχεια μετρήθηκε η απόδοση σε χρόνο εκτέλεσης των εξής ρουτινών BLAS:

- Εσωτερικό γινόμενο δύο διανυσμάτων ($\vec{x} \odot \vec{x}$) με 1000 στοιχεία το καθένα
- Εσωτερικό γινόμενο δύο πινάκων ($X \odot X$) διαστάσεων 1000×1000
- Αντιστροφή πίνακα (X^{-1}) διαστάσεων 1000×1000
- Υπολογισμός διακρίνουσας πίνακα ($|X|$) διαστάσεων 1000×1000
- Υπολογισμός ιδιοτιμών πίνακα ($SVD(X)$) διαστάσεων 2000×1000
- Υπολογισμός ιδιοδιανυσμάτων πίνακα ($Eigen(X)$) διαστάσεων 1500×1500

Κάθε μία από τις πιο πάνω ρουτίνες εκτελέστηκε 1000 φορές και λήφθηκε η μέση τιμή του χρόνου εκτέλεσης. Τα συγχριτικά αποτελέσματα των τριών βιβλιοθηκών BLAS παρουσιάζονται στον [πίνακα 4.3](#),

Πίνακας 4.3: Σύγκριση χρόνου εκτέλεσης βασικών πράξεων γραμμικής άλγεβρας μεταξύ των βιβλιοθηκών ATLAS και OpenBLAS

| | libblas+liblapack+libumfpack | ATLAS | OpenBLAS |
|-------------------------|------------------------------|------------|------------|
| $\vec{x} \odot \vec{x}$ | 11.21 us | 10.99 us | 11.49 us |
| $X \odot X$ | 2602.7 ms | 209.2 ms | 148.8 ms |
| X^{-1} | 3420.279 ms | 570.630 ms | 296.545 ms |
| $ X $ | 972.815 ms | 283.391 ms | 101.563 ms |
| $SVD(X)$ | 94.985 s | 51.635 s | 11.200 s |
| $Eigen(X)$ | 98.585 s | 38.157 s | 28.708 s |

όπου φαίνεται ότι οι καλύτεροι χρόνοι λήφθηκαν με χρήση της βιβλιοθήκης OpenBLAS. Στην συνέχεια μεταγλωττίστηκαν από πηγαίο κώδικα οι βιβλιοθήκες numpy, scipy και Theano, με χρήση της βιβλιοθήκης OpenBLAS (shared library link).

Στο Jetson TK1 εγκαταστάθηκε και η βιβλιοθήκη *CNMeM*²², η οποία έχει αναπτυχθεί από την Nvidia και χρησιμοποιείται για την εκ των προτέρων δέσμευση μνήμης σε μονάδες GPU που υποστηρίζουν CUDA. Όπως θα φανεί στο επόμενο κεφάλαιο, η εκ των προτέρων δέσμευση μνήμης επιταχύνει την διαδικασία εκτέλεσης των μαθηματικών υπολογισμών στην μονάδα GPU. Ωστόσο απαιτεί σωστή ρύθμιση για να αποφευχθούν περιπτώσεις δέσμευσης περισσότερης μνήμης από αυτή που το πρόγραμμα εκτέλεσης απαιτεί.

Είναι σημαντικό να αναφερθεί ότι όλες οι προαναφερθέντες διαδικασίες εγκατάστασης των εργαλείων λογισμικού καθώς και τα πειράματα που εκτελέσθηκαν περιγράφονται πλήρως στον ιστοχώρο:

<https://github.com/klpanagi/Thesis/tree/master/jetson-tk1>

²²<https://github.com/NVIDIA/cnmem>

5

Πειράματα - Αποτελέσματα

Για λόγους πληρότητας, πιο κάτω παρουσιάζονται τα βασικά χαρακτηριστικά των δύο συστημάτων που χρησιμοποιήθηκαν στην εκτέλεση των πειραμάτων:

| Μηχάνημα | # πυρήνων | # νημάτων | clock | Κατ. Ισχύος | RAM | Cache |
|----------|-----------|-----------|----------|-------------|--------------|--------|
| Host PC | 4 | 8 | 3.4 Ghz | 65Watts | 16GB | 8MB L3 |
| TK1 | 4 | 4 | 2.32 Ghz | 12Watts | 2GB (shared) | 2MB L2 |

Ο αριθμός νημάτων (8) που υποστηρίζει ο επεξεργαστής i7-6500 οφείλεται στην τεχνολογία *Hyper-Threading*²³. Επίσης, η τιμή της κατανάλωσης ισχύος για την πλατφόρμα Jetson TK1, αντιστοιχεί στην κατάσταση μέγιστης (ταυτόχρονης) λειτουργίας των μονάδων CPU και GPU.

5.1 ΠΡΩΤΗ ΦΑΣΗ ΠΕΙΡΑΜΑΤΩΝ

Οι εκδόσεις των εργαλείων λογισμικού που χρησιμοποιήθηκαν κατά την διάρκεια των πειραμάτων για τον επεξεργαστή Intel-i7-6700 είναι:

- Keras: v1.1.0
- Theano: v0.9.0-dev3
- numpy: v1.12.0
- scipy: v0.19.0-dev0

²³<http://www.intel.com/content/www/us/en/architecture-and-technology/hyper-threading/hyper-threading-technology.html>

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΑ - ΑΠΟΤΕΛΕΣΜΑΤΑ

Οι μετρήσεις κατανάλωσης ισχύος στον επεξεργαστή Intel-i7-6700 έγιναν χρησιμοποιώντας το εργαλείο *PowerAPI* [35]. Το τελευταίο, επιτρέπει την μέτρηση κατανάλωσης ισχύος σε επίπεδο διεργασίας (process). Για την βέλτιστη λειτουργία του, απαιτείται η ρύθμιση της παραμέτρου TDP (Thermal Design Power) για τον εκάστοτε επεξεργαστή στον οποίο λαμβάνονται οι μετρήσεις. Η τιμή αυτής της παραμέτρου για τον επεξεργαστή Intel-i7-6700, θεωρήθηκε στα 65W σύμφωνα με το αντίστοιχο τεχνικό εγχειρίδιο ²⁴. Επιπρόσθετα, για τις μετρήσεις που πραγματοποιήθηκαν λήφθηκαν υπόψη οι εξής παράγοντες:

5.1.1 AlexNet

Οι παράμετροι των πειραμάτων είναι οι εξής:

- Αριθμός επαναλήψεων: 1000
- Αριθμός Νημάτων: 1, 2, 4, 8

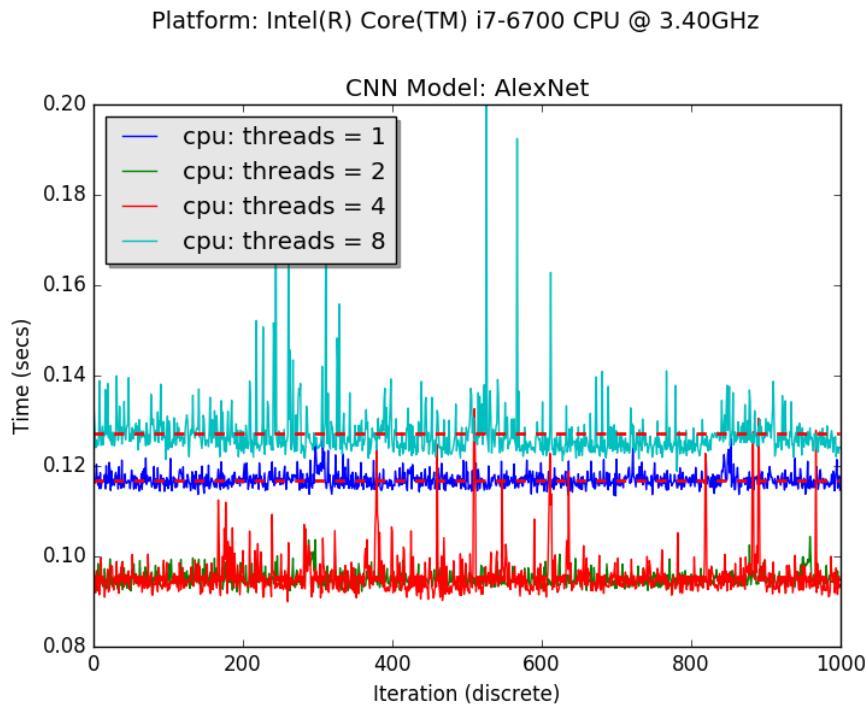
Στον [πίνακα 5.1](#) παρουσιάζονται τα αποτελέσματα της μέσης τιμής του χρόνου εκτέλεσης, ενώ στο [σχήμα 5.1](#) φαίνεται το διάγραμμα του χρόνου εκτέλεσης σε κάθε επανάληψη.

Πίνακας 5.1: Μετρήσεις πειραμάτων για το δίκτυο AlexNet σε επεξεργαστή Intel-i7-6700

| # νημάτων | Χρόνος Εκτέλεσης (sec) | Κατανάλωση Ισχύος (Watts) | Perf / Watt |
|-----------|------------------------|---------------------------|-------------|
| 1 | 0.117 | 5.7 | 1.5 |
| 2 | 0.095 | 11.48 | 0.916 |
| 4 | 0.095 | 22.86 | 0.46 |
| 8 | 0.127 | 41.4 | 0.19 |

Η μέγιστη τιμή μνήμης RAM που δεσμεύει το δίκτυο AlexNet μετρήθηκε στα 692MB.

²⁴Intel i7 6700 TDP: http://ark.intel.com/products/88196/Intel-Core-i7-6700-Processor-8M-Cache-up-to-4_00-GHz



Σχήμα 5.1: Χρόνοι εκτέλεσης για το δίκτυο AlexNet σε επεξεργαστή i7

5.1.2 VGG16

Οι παράμετροι των πειραμάτων είναι οι εξής:

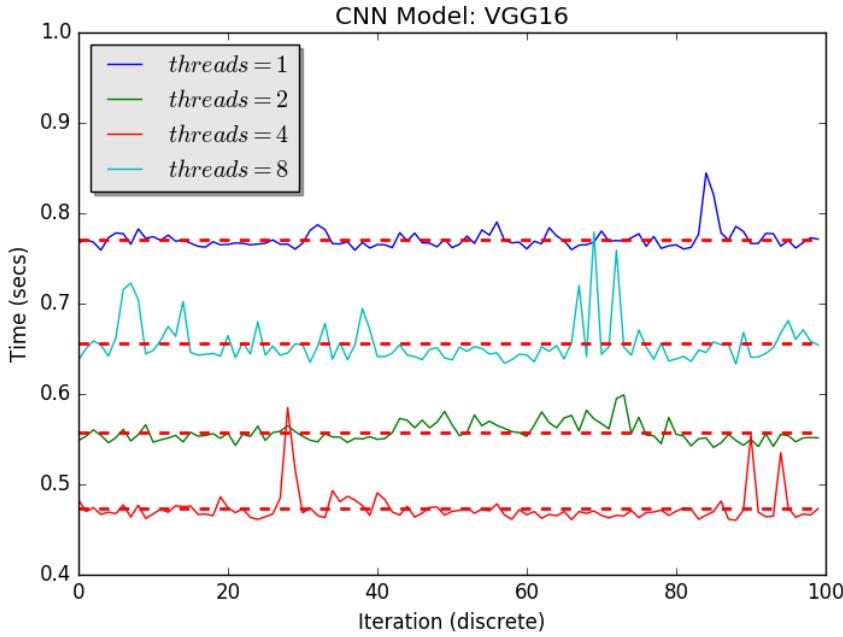
- Αριθμός επαναλήψεων: 100
- Αριθμός Νημάτων: 1, 2, 4, 8

Η μέσες τιμές του χρόνου εκτέλεσης των πειραμάτων παρουσιάζονται στον πίνακα 5.2. Η απαιτούμενη τιμή μνήμης RAM μετρήθηκε στα 710MB.

Πίνακας 5.2: Μετρήσεις πειραμάτων για το δίκτυο VGG16 σε επεξεργαστή Intel-i7-6700

| # νημάτων | Χρόνος Εκτέλεσης (sec) | Κατανάλωση Ισχύος (Watts) | Perf / Watt |
|-----------|------------------------|---------------------------|-------------|
| 1 | 0.7709 | 5.76 | 0.225 |
| 2 | 0.5577 | 11.13 | 0.161 |
| 4 | 0.4734 | 21.05 | 0.1 |
| 8 | 0.6555 | 41.56 | 0.036 |

Platform: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz



Σχήμα 5.2: Χρόνοι εκτέλεσης για το δίκτυο VGG16 σε επεξεργαστή i7

5.1.3 Tiny-YOLO

Οι παράμετροι των πειραμάτων είναι οι εξής:

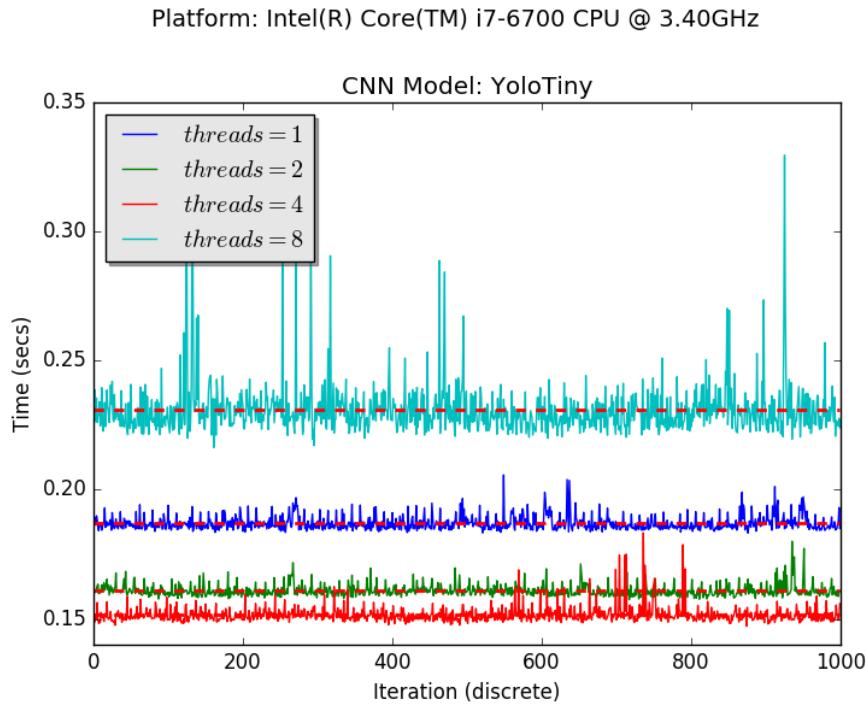
- Αριθμός επαναλήψεων: 1000
- Αριθμός Νημάτων: 1, 2, 4, 8

Η αντίστοιχη μέση τιμή των χρόνων εκτέλεσης των πειραμάτων παρουσιάζονται στον πίνακα 5.3. Σημαντικό να αναφερθεί ότι ο χρόνος εκτέλεσης της αντίστοιχης υλοποίησης της ερευνητικής ομάδας που σχεδίασε το δίκτυο Tiny-YOLO, μετρήθηκε στα 1.096 δευτερόλεπτα (δεν χρησιμοποιούνται νήματα).

Πίνακας 5.3: Μετρήσεις πειραμάτων για το δίκτυο Tiny-YOLO σε επεξεργαστή Intel-i7-6700

| # νημάτων | Χρόνος Εκτέλεσης (sec) | Κατανάλωση Ισχύος (Watts) | Perf / Watt |
|-----------|------------------------|---------------------------|-------------|
| 1 | 0.1871 | 5.74 | 0.93 |
| 2 | 0.1609 | 11.5 | 0.54 |
| 4 | 0.1516 | 22.86 | 0.288 |
| 8 | 0.2309 | 42.77 | 0.1 |

Η μνήμη (μέγιστη) που απαιτείται κατά την διαδικασία προς-τα-εμπρός εκτέλεσης μετρήθηκε στα 379MB.



Σχήμα 5.3: Χρόνοι εκτέλεσης για το δίκτυο Tiny-YOLO σε επεξεργαστή i7

5.2 ΔΕΥΤΕΡΗ ΦΑΣΗ ΠΕΙΡΑΜΑΤΩΝ

Η μέτρηση της ισχύος που καταναλώνει η πλακέτα έγινε ακολουθώντας την παρακάτω μεθοδολογία:

- Από το έγγραφο που περιγράφει τα σχηματικά της πλακέτας²⁵ βρέθηκε η αντίσταση (R5C11, 5mΩ) από την οποία περνάει το ρεύμα που παρέχεται από το τροφοδοτικό.
- Βρέθηκε πάνω στην πλακέτα η θέση της αντίστασης και χρησιμοποιήθηκε ένα πολύμετρο για λήψη της τιμής της τάσης στα άκρα της. Από την τιμή της αντίστασης και της τάσης στα άκρα της υπολογίζεται το ρεύμα που διέρχεται από αυτήν, το οποίο ισοδυναμεί με το ρεύμα που λαμβάνει η πλακέτα από το τροφοδοτικό.
- Γνωρίζοντας την τιμή της τάσης τροφοδοσίας (12.15Volts) και την τιμή του ρεύματος που λαμβάνεται από το τροφοδοτικό, υπολογίζεται η παρεχόμενη στην πλακέτα ισχύ.

Προτού γίνουν μετρήσεις ισχύος, χρησιμοποιήθηκε το εργαλείο *powertop*²⁶, το οποίο δίνει πληροφορίες σχετικά με τις ενεργοποιημένες μονάδες ενός συστήματος και την σχετική απόδοσή τους σε κατανάλωση ισχύος (όχι σε τιμές Watt). Οι

²⁵TK1 Compact Development Module, 602-7R375-0000-D10, σελίδα 27

²⁶PowerTOP: <https://github.com/fenrus75/powertop>

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΑ - ΑΠΟΤΕΛΕΣΜΑΤΑ

πιο κάτω προτάσεις του συγκεκριμένου εργαλείου για το ενσωματωμένο σύστημα Jetson TK1 λήφθηκαν υπόψη:

- Απενεργοποίηση των ελεγκτών USB (EHCI και xHCI).
- Ενεργοποίηση λειτουργίας διαχείρισης ισχύος για τον ελεγκτή SATA (SATA link power management)
- Ενεργοποίηση λειτουργίας διαχείρισης ισχύος της μονάδας Audio Codec (Audio Codec power management)
- Ενεργοποίηση λειτουργίας διαχείρισης ισχύος διαύλου PCIe (PCIe bridge runtime power management)

Στην συνέχεια μετρήθηκε η τιμή της κατανάλωσης ισχύος της πλακέτας σε κατάσταση αεργίας (idle), στα 3.4Watts. Σημαντικό επίσης να αναφερθεί ότι ο ανεμιστήρας τον οποίο διαθέτει το συγκεκριμένο ενσωματωμένο, για την ψύξη του SOC, καταναλώνει 0.85Watts ²⁷.

Ο λόγος της απόδοσης ανά μονάδα ισχύος (performance per watt) υπολογίστηκε με βάση τον αριθμό των εικόνων που μπορεί να επεξεργαστεί το εκάστοτε νευρωνικό δίκτυο ανά δευτερόλεπτο (frames per second):

$$\text{performance/watt} = (1/\text{executiontime}) / (\text{powerconsumption})$$

Η μεταγλώττιση των δικτύων AlexNet και VGG16 απαιτεί περισσότερη μνήμη από αυτή που διαθέτει το ενσωματωμένο σύστημα Jetson TK1 (1888MB). Το πρόβλημα αυτό το αντιμετωπίστηκε προσθέτοντας 1GB μνήμη Swap²⁸.

Οι εκδόσεις των εργαλείων λογισμικού που χρησιμοποιήθηκαν κατά την διάρκεια των πειραμάτων για τον επεξεργαστή Intel-i7-6700 είναι:

- Keras: v1.1.0
- Theano: v0.9.0-dev3
- numpy: v1.11.0
- scipy: v0.19.0-dev0

5.2.1 AlexNet

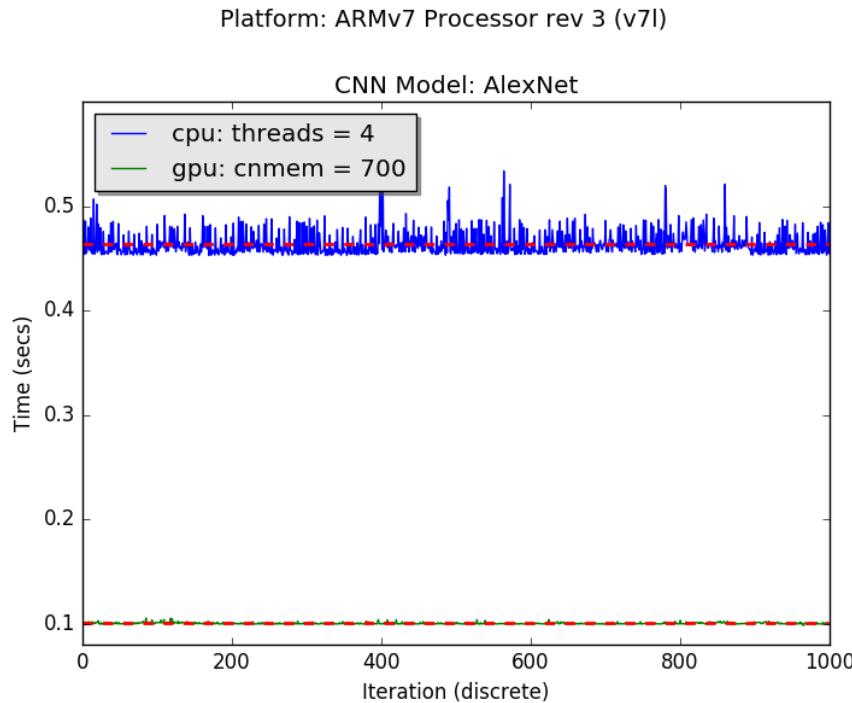
Οι παράμετροι των πειραμάτων είναι οι εξής:

- Αριθμός επαναλήψεων: 1000
- Υπολογιστική μονάδα:
 - CPU: Αριθμός Νημάτων (1, 2, 4, 8)

²⁷Κατανάλωσης ισχύος του ανεμιστήρα που διαθέτει το Jetson TK1 http://elinux.org/Jetson/Graphics_Performance#Test_Methodology

²⁸Δημιουργία αρχείου μνήμης swap: <http://www.jetsonhacks.com/2014/10/04/creating-swapfile-ubuntu-nvidia-jetson-tk1/>

- GPU: Με και χωρίς εκ των προτέρων δέσμευση μνήμης



Σχήμα 5.4: Χρόνοι εκτέλεσης για το δίκτυο AlexNet στο Jetson TK1

Στον πίνακα 5.4 παρουσιάζονται τα συγκριτικά αποτελέσματα της εκτέλεσης τόσο στην μονάδα CPU (με μεταβλητό αριθμό νημάτων), όσο και στην GPU (με και χωρίς εκ των προτέρων δέσμευση μνήμης).

Πίνακας 5.4: Μετρήσεις πειραμάτων για το δίκτυο AlexNet στο Jetson TK1

| Μονάδα | # νημάτων | cnmem | Χρόνος εκτέλεσης (sec) | Κατ. Ισχύος (Watts) | Perf/Watt |
|--------|-----------|-------|------------------------|---------------------|-----------|
| CPU | 1 | N/A | 0.62 | 6.318 | 0.254 |
| CPU | 2 | N/A | 0.4845 | 8.019 | 0.2574 |
| CPU | 4 | N/A | 0.4634 | 10.692 | 0.202 |
| CPU | 8 | N/A | 0.4646 | 10.692 | 0.2013 |
| GPU | N/A | None | 0.1049 | 8.5 | 1.1215 |
| GPU | N/A | 700MB | 0.1 | 8.5 | 1.1764 |

5.2.2 VGG16

Οι παράμετροι των πειραμάτων είναι οι εξής:

- Αριθμός επαναλήψεων: 100
- Υπολογιστική μονάδα:
 - CPU: Αριθμός Νημάτων (1, 2, 4, 8)

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΑ - ΑΠΟΤΕΛΕΣΜΑΤΑ

- GPU: Με και χωρίς εκ των προτέρων δέσμευση μνήμης (cnmem)

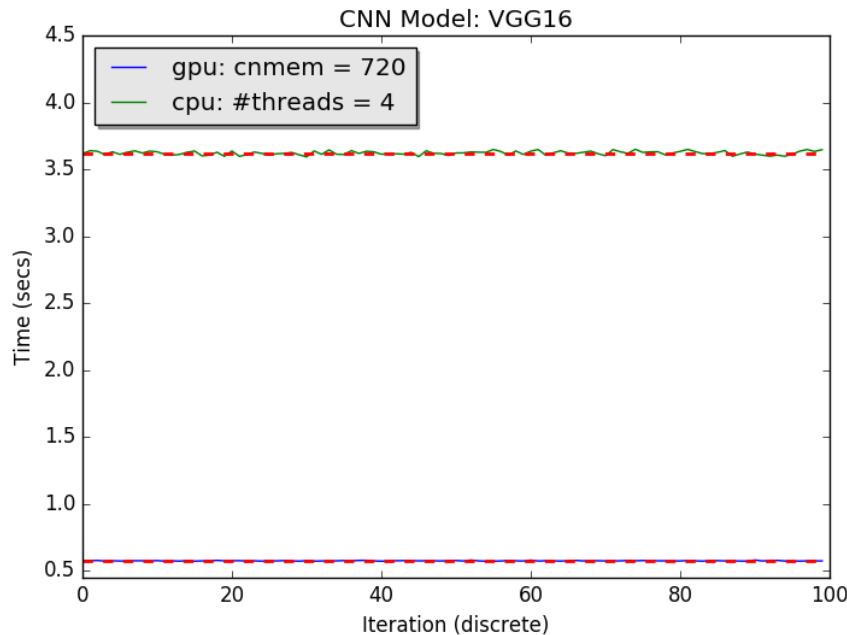
Στον παρακάτω [πίνακα 5.5](#) βρίσκονται τα συγχριτικά αποτελέσματα της εκτέλεσης τόσο στην μονάδα CPU (με μεταβλητό αριθμό νημάτων), όσο και στην GPU (με και χωρίς εκ των προτέρων δέσμευση μνήμης).

Πίνακας 5.5: Μετρήσεις πειραμάτων για το δίκτυο VGG16 στο Jetson TK1

| Μονάδα | # νημάτων | cnmem | Χρόνος εκτέλεσης (sec) | Κατ. Ισχύος (Watts) | Perf/Watt |
|--------|-----------|-------|------------------------|---------------------|-----------|
| CPU | 1 | N/A | 9.431 | 6.318 | 0.0167 |
| CPU | 2 | N/A | 5.4476 | 8.748 | 0.021 |
| CPU | 4 | N/A | 3.6224 | 13.122 | 0.021 |
| CPU | 8 | N/A | 3.6308 | 13.122 | 0.021 |
| GPU | N/A | None | 0.6983 | 11.907 | 0.1207 |
| GPU | N/A | 720MB | 0.5761 | 11.907 | 0.1457 |

Στο [σχήμα 5.5](#) που ακολουθεί, παρουσιάζεται το διάγραμμα των χρόνων εκτέλεσης στην μονάδα CPU με χρήση τεσσάρων νημάτων και στην μονάδα GPU με εκ των προτέρων δέσμευση 720MB μνήμης.

Platform: Jetson TK1



Σχήμα 5.5: Χρόνοι εκτέλεσης για το δίκτυο VGG16 στο Jetson TK1

Παρατηρούμε ότι η εκτέλεση στην μονάδα GPU είναι 5.187 φορές πιο γρήγορη (χωρίς εκ των προτέρων δέσμευση μνήμης). Επίσης, με εκ των προτέρων δέσμευση μνήμης για την μονάδα GPU, ο χρόνος εκτέλεσης μειώνεται ακόμη περισσότερο (περίπου 120ms).

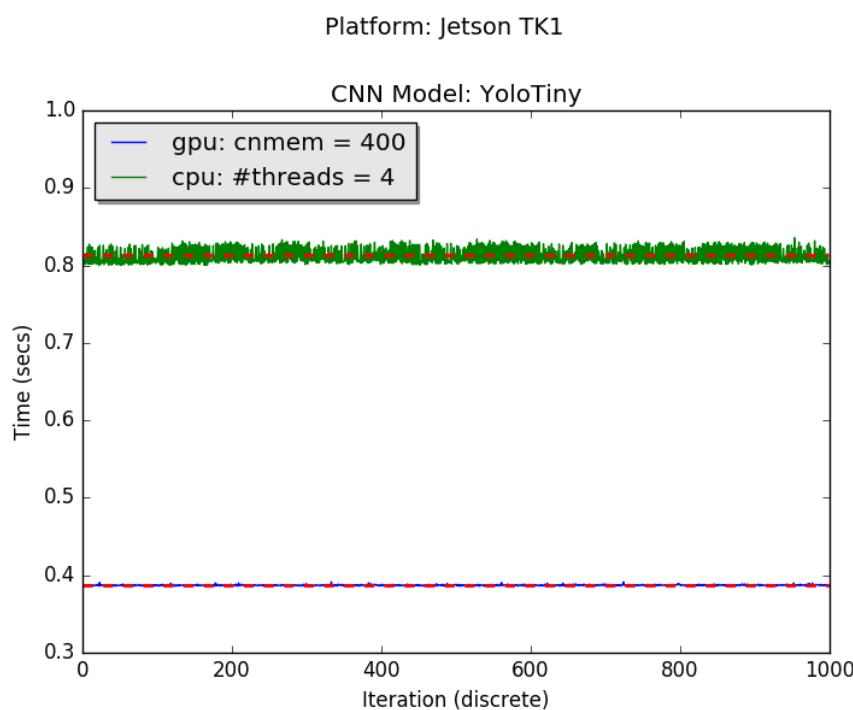
5.2.3 Tiny-YOLO

Οι παράμετροι των πειραμάτων είναι οι εξής:

- Αριθμός επαναλήψεων: 1000
- Υπολογιστική μονάδα:
 - CPU: Αριθμός Νημάτων (1, 2, 4, 8)
 - GPU: Με και χωρίς εκ των προτέρων δέσμευση μνήμης (cnmem)

Πίνακας 5.6: Μετρήσεις πειραμάτων για το δίκτυο Tiny-YOLO στο Jetson TK1

| Μονάδα | # νημάτων | cnmem | Χρόνος εκτέλεσης (sec) | Κατ. Ισχύος (Watts) | Perf/Watt |
|--------|-----------|-------|------------------------|---------------------|-----------|
| CPU | 1 | N/A | 1.5902 | 6.318 | 0.1 |
| CPU | 2 | N/A | 1.0276 | 8.504 | 0.1144 |
| CPU | 4 | N/A | 0.8137 | 11.9 | 0.103 |
| CPU | 8 | N/A | 0.8126 | 11.9 | 0.1034 |
| GPU | N/A | None | 0.5543 | 8 | 0.2255 |
| GPU | N/A | 400MB | 0.3872 | 8 | 0.323 |



Σχήμα 5.6: Χρόνοι εκτέλεσης για το δίκτυο Tiny-YOLO στο Jetson TK1

Ο χρόνος εκτέλεσης της αντίστοιχης υλοποίησης της ερευνητικής ομάδας που σχεδίασε το δίκτυο Tiny-YOLO μετρήθηκε, για την μονάδα CPU του Tegra K1, στα

ΚΕΦΑΛΑΙΟ 5. ΠΕΙΡΑΜΑΤΑ - ΑΠΟΤΕΛΕΣΜΑΤΑ

3.475 δευτερόλεπτα (δεν χρησιμοποιούνται νήματα). Η μέτρηση του χρόνου εκτέλεσης της αντίστοιχης υλοποίησης στην μονάδα GPU ήταν αδύνατη. Η εκτέλεση στην μονάδα GPU απαιτούσε περισσότερη μνήμη από αυτή που διαθέτει η πλατφόρμα Jetson TK1 (2GB) και έτσι το εκτελέσιμο τερματίζει με σφάλμα *CUDA Error: out of memory*. Ο λόγος που αυτό συμβαίνει μόνο στην περίπτωση εκτέλεσης στην μονάδα GPU δεν είναι ευφανές. Πιθανόν να οφείλεται σε σφάλμα τύπου διαρροής μνήμης (memory leak).

5.3 Συγκριτικά Αποτελέσματα

Στον πίνακα 5.7 συνοψίζονται τα αποτελέσματα που λήφθηκαν στις δύο φάσεις των πειραμάτων (εκτέλεση σε επεξεργαστή Intel i7-6700 και εκτέλεση στο Jetson TK1).

Πίνακας 5.7: Συγκριτικά αποτελέσματα των πειραμάτων σε επεξεργαστή Intel i7-6700 και Jetson TK1

| CNN | # νημάτων | Μονάδα | cnmem | Χρόνος εκτέλεσης (sec) |
|-----------|-----------|----------------|-------|------------------------|
| AlexNet | 1 | i7-6700 | N/A | 0.117 |
| | | ARM Cortex A15 | N/A | 0.62 |
| | 2 | i7-6700 | N/A | 0.095 |
| | | ARM Cortex A15 | N/A | 0.4845 |
| | 4 | i7-6700 | N/A | 0.095 |
| | | ARM Cortex A15 | N/A | 0.4634 |
| | 8 | i7-6700 | N/A | 0.127 |
| | | ARM Cortex A15 | N/A | 0.4646 |
| | N/A | GK20a GPU | 0 | 0.1049 |
| | | | 700MB | 0.1 |
| VGG16 | 1 | i7-6700 | N/A | 0.7709 |
| | | ARM Cortex A15 | N/A | 9.431 |
| | 2 | i7-6700 | N/A | 0.5577 |
| | | ARM Cortex A15 | N/A | 5.4476 |
| | 4 | i7-6700 | N/A | 0.4734 |
| | | ARM Cortex A15 | N/A | 3.6224 |
| | 8 | i7-6700 | N/A | 0.6555 |
| | | ARM Cortex A15 | N/A | 3.6308 |
| | N/A | GK20a GPU | 0 | 0.6983 |
| | | | 720MB | 0.5761 |
| Tiny-YOLO | 1 | i7-6700 | N/A | 0.18709 |
| | | ARM Cortex A15 | N/A | 1.5902 |
| | 2 | i7-6700 | N/A | 0.1609 |
| | | ARM Cortex A15 | N/A | 1.0276 |
| | 4 | i7-6700 | N/A | 0.1516 |
| | | ARM Cortex A15 | N/A | 0.8137 |
| | 8 | i7-6700 | N/A | 0.2309 |
| | | ARM Cortex A15 | N/A | 0.8126 |
| | N/A | GK20a GPU | 0 | 0.5543 |
| | | | 400MB | 0.3872 |

6

Συμπεράσματα

Στο κεφάλαιο αυτό παρουσιάζονται συνοπτικά οι υλοποιήσεις των μοντέλων CNN και τα συμπεράσματα που προέκυψαν από την έκβαση των πειραμάτων. Γίνεται σύγκριση της απόδοσης των μοντέλων αυτών στα δύο συστήματα (PC με επεξεργαστή Intel i7-6700 και Jetson TK1), έχοντας ως παραμέτρους αξιολόγησης τον χρόνο εκτέλεσης και την κατανάλωση ισχύος. Στην συνέχεια αναφέρονται τα προβλήματα που παρουσιάστηκαν κατά την διάρκεια των υλοποιήσεων και των πειραμάτων.

6.1 ΓΕΝΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ

Στα πλαίσια της παρούσας διπλωματικής εργασίας υλοποιήθηκε το δίκτυο Tiny-YOLO χρησιμοποιώντας την βιβλιοθήκη Keras. Το δίκτυο AlexNet υλοποιήθηκε με βάση την αντίστοιχη υλοποίηση στο Caffe²⁹, και το μοντέλο του δικτύου VGG16 υπάρχει υλοποιημένο στα παραδείγματα του Keras.

Με βάση τα αποτελέσματα των πειραμάτων (βλ. [κεφάλαιο 5](#)) που αφορούν στους χρόνους εκτέλεσης και κατανάλωσης ισχύος παρατηρήθηκαν τα εξής:

- Οι καλύτεροι χρόνοι εκτέλεσης στον επεξεργαστή Intel-i7-6700 (και για τα 3 μοντέλα CNN) λήφθηκαν με χρήση τεσσάρων (4) νημάτων. Παρόλο που ο συγκεκριμένος επεξεργαστής υποστηρίζει (φευδο)παράλληλη εκτέλεση 8 νημάτων (Hyper-Threading), παρατηρήθηκε στα πειράματα σημαντική καθυστέρηση. Το γεγονός αυτό οφείλεται στην εναλλαγή των νημάτων στους πυρήνες του επεξεργαστή (context switch³⁰).

²⁹Υλοποίηση του δικτύου AlexNet στο Caffe: https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet

³⁰http://wiki.osdev.org/Context_Switching

- Ο χρόνος εκτέλεσης στην μονάδα CPU και των δύο συστημάτων παρουσιάζει αυξητικές διακυμάνσεις, ανάλογες προς τον αριθμό των νημάτων που εισέρχονται στον επεξεργαστή. Η αύξηση του αριθμού των νημάτων προσδίδει αβεβαιότητα στον χρόνο εκτέλεσης. Αντίθετα, ο χρόνος εκτέλεσης στην μονάδα GPU είναι γενικά σταθερός, χωρίς σημαντικές διακυμάνσεις και επομένως ενδείκνυται για εφαρμογές πραγματικού χρόνου.
- Με χρήση της βιβλιοθήκης CNMeM για εκ των προτέρων δέσμευση μνήμης ο χρόνος εκτέλεσης στην μονάδα GPU μειώνεται αισθητά (100-150ms).
- Η ισχύς που καταναλώνεται από την μονάδα GPU είναι σε κάθε περίπτωση χαμηλότερη σε σχέση με αυτή που καταναλώνει η μονάδα CPU σε κατάσταση μέγιστης λειτουργίας (χρήση τεσσάρων νημάτων).
- Με χρήση της μονάδας GPU ο χρόνος εκτέλεσης μειώνεται κατά 2-5 φορές σε σχέση με την εκτέλεση στην μονάδα CPU του Tegra K1.
- Η εκτέλεση στην μονάδα GPU πέρα από το γεγονός ότι είναι πιο γρήγορη, αφήνει 3.5 πυρήνες της μονάδας CPU ελεύθερες στο σύστημα. Αυτό είναι ιδιαίτερα σημαντικό αφού επιτρέπει την εκτέλεση άλλων διεργασιών (processes) στην μονάδα CPU.

Επιπλέον, ο χρόνος εκτέλεσης της διαδικασίας forward propagation του δικτύου Tiny-YOLO στο Jetson TK1 (0.3sec), είναι αποδεκτός για χρήση του σε ρομποτικές εφαρμογές όπου δεν απαιτείται επεξεργασία περισσότερων των τριών εικόνων το δευτερόλεπτο (3 frames per second). Αυτό φυσικά εξαρτάται από τις απαιτήσεις της εκάστοτε εφαρμογής.

6.2 ΠΡΟΒΛΗΜΑΤΑ

Ένα από τα αρχικά προβλήματα που παρουσιάστηκαν κατά την διάρκεια των υλοποιήσεων ήταν η ασυμβατότητα μεταξύ των διαφόρων εργαλείων λογισμικού και του ενσωματωμένου συστήματος. Συγκεκριμένα, τα εργαλεία για DL υποστηρίζουν σήμερα την 5η έκδοση της βιβλιοθήκης cuDNN, ενώ η τελευταία έκδοση που υποστηρίζεται για την μονάδα GPU που διαθέτει το ενσωματωμένο σύστημα Jetson TK1 είναι η 2η. Αυτό έχει σαν αποτέλεσμα τη μη ενσωμάτωση και χρήση της cuDNN από την βιβλιοθήκη Theano.

Ένα δεύτερο πρόβλημα ήταν η περιορισμένη μνήμη RAM που διαθέτει το ενσωματωμένο σύστημα Jetson TK1. Παρόλο που το Tegra K1 SOC υποστηρίζει μέχρι και 8GB μνήμη RAM η συγκεκριμένη πλακέτα διαθέτει μόνο 2GB. Το γεγονός αυτό περιορίσε τον αριθμό των δικτύων που εκτελέστηκαν στο Jetson TK1 αφού δίκτυα με πολλά επίπεδα απαιτούν πολύ περισσότερη μνήμη. Συγκεκριμένα, δοκιμάστηκε η εκτέλεση του δικτύου ResNet σε PC και η μνήμη που απαιτούσε μετρήθηκε στα 3.7GB. Η περιορισμένη μνήμη RAM έφερε προβλήματα και κατά την διαδικασία μεταγλώτισης του δικτύου VGG16, τα οποία ωστόσο αντιμετωπίστηκαν προσθέτοντας μνήμη τύπου Swap.

7

Μελλοντικές επεκτάσεις

Είναι σημαντικό να ερευνηθεί η εκτέλεση των πειραμάτων και στο ενσωματωμένο σύστημα Jetson TX1. Το Jetson TX1 είναι το τελευταίο ενσωματωμένο σύστημα της Nvidia, με αρχιτεκτονική επεξεργαστή 64bit και μονάδα GPU αρχιτεκτονικής Maxwell με 256 πυρήνες, το οποίο έχει σχεδιαστεί για χρήση σε εφαρμογές βαθιάς μηχανικής μάθησης. Η αρχιτεκτονική Maxwell της μονάδας GPU είναι συμβατή με την 5η έκδοση της βιβλιοθήκης cuDNN, η οποία ενσωματώνει ρουτίνες εκτέλεσης των διαφόρων επιπέδων ενός CNN κατευθείαν πάνω στην GPU.

Επίσης, θα μπορούσαν να εκτελεστούν τα πειράματα με χρήση της βιβλιοθήκης Tensorflow έτσι ώστε να ληφθούν αποτελέσματα τα οποία αφορούν στην απόδοση του χρόνου εκτέλεσης μεταξύ των βιβλιοθηκών Theano και Tensorflow. Η επιλογή χρήσης μίας εκ των 2 αυτών βιβλιοθηκών στο Keras ορίζεται μέσα από ένα αρχείο ρύθμισης. Δηλαδή, δεν απαιτούνται αλλαγές στον υπάρχον πηγαίο κώδικα των υλοποιήσεων που έγιναν κατά την διάρκεια της παρούσας διπλωματική εργασίας.

Επιπρόσθετα θα μπορούσαν να γίνουν μετρήσεις της κατανάλωσης ισχύος στον επεξεργαστή Intel i7-6700 και να συγκριθούν με τις αντίστοιχες που παρουσιάστηκαν για το ενσωματωμένο σύστημα Jetson TK1. Οι μετρήσεις αυτές θα μπορούσαν να χρησιμοποιηθούν στην συνέχεια για σύγκριση της απόδοσης (σε χρόνο εκτέλεσης) των 2 συστημάτων σε σχέση με την κατανάλωση ισχύος (performance per watt - GFlops/Watt).

Είναι σημαντικό να ερευνηθεί το κομμάτι της εκπαίδευσης των CNN και η χρήση τους για την ανάπτυξη στοχευμένων εφαρμογών. Οι υλοποιήσεις των δικτύων που παρουσιάστηκαν στο [κεφάλαιο 4](#) επιτρέπουν την αποσύνδεση των τελευταίων (πλήρως συνδεδεμένων) επιπέδων από το εκάστοτε CNN και την επικόλληση άλλων επιπέδων. Τα τελευταία αυτά επίπεδα έχουν τον ρόλο ενός ταξινομητή. Είναι δηλαδή δυνατή η σχεδίαση και επικόλληση ταξινομητών για συγκεκριμένες εφαρμογές. Για παράδειγμα, θα μπορούσε να χρησιμοποιηθεί ένας δυαδικός ταξινομητής για την επίλυση προβλημάτων αναγνώρισης και εντοπισμού ενός αντικειμένου. Σε αυτή την περίπτωση η έξοδος από το τελευταίο επίπεδο του CNN θα αποτελείται από 2

νευρώνες. Ο ένας νευρώνας ενεργοποιείται σε περίπτωση εντοπισμού του συγκεκριμένου αντικειμένου (True Positive), ενώ ο δεύτερος νευρώνας ενεργοποιείται σε αντίθετη περίπτωση (True Negative). Η συγκεκριμένη μεθοδολογία θα μπορούσε να χρησιμοποιηθεί για την επίλυση διαφόρων προβλημάτων αναγνώρισης και εντοπισμού αντικειμένων που έχει να αντιμετωπίσει το ρομπότ-διασώστης PANDORA, όπως:

- Αναγνώριση θυμάτων
- Αναγνώριση πινακίδων κινδύνου (hazards)
- Αναγνώριση βαρελιών

Μία ακόμα ενδιαφέρουσα προοπτική είναι να αναπτυχθούν οι συγκεκριμένες υλοποιήσεις στο *Cloud*, να ληφθούν μετρήσεις που αφορούν τον συνολικό χρόνο εκτέλεσης κλήσης της υπηρεσίας αυτής και να συγκριθούν με τους χρόνους εκτέλεσης στο Jetson TK1 που παρουσιάστηκαν στο [κεφάλαιο 5](#). Αυτό υπονοεί και την μέτρηση των καθυστερήσεων που εισάγονται λόγω της μεταφοράς των δεδομένων στο εκάστοτε δίκτυο (network delays). Θα πρέπει να ληφθούν υπόψη η μορφή του δικτύου (WiFi, Ethernet, local, public) και το πρωτόκολλο επικοινωνίας που θα χρησιμοποιηθεί (HTTP, WebSockets, κτλ). Προτείνεται η χρήση της πλατφόρμας *RAPP Platform*³¹. Η συγκεκριμένη πλατφόρμα έχει σχεδιαστεί για χρήση ανάπτυξης ρομποτικών (χυρίων) εφαρμογών στο *Cloud*. Τα κύρια χαρακτηριστικά της πλατφόρμας αυτής είναι τα εξής:

- Υποδομή για διασύνδεση κόμβων ROS (ROS Nodes) με υπηρεσίες διαδικτύου (Web Services).
- Παρέχει εργαλεία για εύκολη και γρήγορη ανάπτυξη υπηρεσιών διαδικτύου.
- Παρέχει client API (σε γλώσσες προγραμματισμού Python, JavaScript και C++) για κλήση των υπηρεσιών που προσφέρει η πλατφόρμα.
- Παρέχει εργαλεία δοκιμής (testing tools) των υπηρεσιών που αναπτύσσονται στην πλατφόρμα.

³¹The RAPP Platform: <https://github.com/rapp-project/rapp-platform>

Βιβλιογραφία

- [1] Avron Barr, Paul R Cohen, and Edward A Feigenbaum. “*The handbook of Artificial Intelligence, volume IV*“, 1989.
- [2] Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu. “*Deep blue*“. Artificial intelligence, 134(1):57–83, 2002.
- [3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “*ImageNet Classification with Deep Convolutional Neural Networks*“. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, “*Advances in Neural Information Processing Systems 25*“, pages 1097–1105. Curran Associates, Inc., 2012.
- [4] Firas Abuzaid. “*Optimizing CPU Performance for Convolutional Neural Networks*“.
- [5] Masahiro Mori, Karl F MacDorman, and Norri Kageki. “*The uncanny valley [from the field]*“. IEEE Robotics & Automation Magazine, 19(2):98–100, 2012.
- [6] Ryszard S Michalski, Jaime G Carbonell, and Tom M Mitchell. “*Machine learning: An artificial intelligence approach*“. Springer Science & Business Media, 2013.
- [7] Pierre Baldi. “*Autoencoders, unsupervised learning, and deep architectures.*“. ICML unsupervised and transfer learning, 27(37-50):1, 2012.
- [8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. “*Deep Learning*“. Book in preparation for MIT Press, 2016. URL <http://www.deeplearningbook.org>.
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “*Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*“. CoRR, abs/1502.01852, 2015.
- [10] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. “*Maxout networks.*“. ICML (3), 28:1319–1327, 2013.
- [11] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “*Learning representations by back-propagating errors*“. Cognitive modeling, 5(3):1, 1988.
- [12] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “*Representation learning: A review and new perspectives*“. IEEE transactions on pattern analysis and machine intelligence, 35(8):1798–1828, 2013.
- [13] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. “*Gradient-based learning applied to document recognition*“. Proceedings of the IEEE, 86(11):2278–2324, 1998.

- [14] Matthew D. Zeiler and Rob Fergus. “*Visualizing and Understanding Convolutional Networks*“. CoRR, abs/1311.2901, 2013.
- [15] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. “*Deep neural networks for object detection*“. In “*Advances in Neural Information Processing Systems*“, pages 2553–2561, 2013.
- [16] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. “*Overfeat: Integrated recognition, localization and detection using convolutional networks*“. arXiv preprint arXiv:1312.6229, 2013.
- [17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “*Deep Residual Learning for Image Recognition*“. CoRR, abs/1512.03385, 2015.
- [18] Ross B. Girshick. “*Fast R-CNN*“. CoRR, abs/1504.08083, 2015.
- [19] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. “*You Only Look Once: Unified, Real-Time Object Detection*“. CoRR, abs/1506.02640, 2015.
- [20] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. “*Caffe: Convolutional Architecture for Fast Feature Embedding*“. arXiv preprint arXiv:1408.5093, 2014.
- [21] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zhang. “*TensorFlow: A system for large-scale machine learning*“. CoRR, abs/1605.08695, 2016.
- [22] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu,

Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. “*Theano: A Python framework for fast computation of mathematical expressions*“. arXiv e-prints, abs/1605.02688, May 2016.

- [23] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. “*Theano: a CPU and GPU Math Expression Compiler*“. In “*Proceedings of the Python for Scientific Computing Conference (SciPy)*“, June 2010. Oral Presentation.
- [24] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. “*Theano: new features and speed improvements*“. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.
- [25] Ronan Collobert, Samy Bengio, and Johnny Mariéthoz. “*Torch: a modular machine learning software library*“. Technical report, Idiap, 2002.
- [26] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. “*Torch7: A matlab-like environment for machine learning*“. In “*BigLearn, NIPS Workshop*“, number EPFL-CONF-192376, 2011.
- [27] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. “*Implementing neural networks efficiently*“. In “*Neural Networks: Tricks of the Trade*“, pages 537–557. Springer, 2012.
- [28] François Chollet. “*Keras*“. <https://github.com/fchollet/keras>, 2015.
- [29] Frank Seide and Amit Agarwal. “*CNTK: Microsoft’s Open-Source Deep-Learning Toolkit*“. In “*Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*“, KDD ’16, pages 2135–2135, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2.
- [30] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan

- Yu, and Xiaoqiang Zheng. “*TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*“. CoRR, abs/1603.04467, 2016.
- [31] Andrej Karpathy. “*Convnetjs-deep learning in your browser, 2015*“. <https://github.com/karpathy/convnetjs>, 2015.
- [32] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “*Deep learning*“. Nature, 521 (7553):436–444, 2015.
- [33] K. Simonyan and A. Zisserman. “*Very Deep Convolutional Networks for Large-Scale Image Recognition*“. CoRR, abs/1409.1556, 2014.
- [34] Joseph Redmon. “*Darknet: Open Source Neural Networks in C*“. <http://pjreddie.com/darknet/>, 2013–2016.
- [35] R. E. Grant, M. Levenhagen, S. L. Olivier, D. DeBonis, K. T. Pedretti, and J. H. Laros III. “*Standardizing Power Monitoring and Control at Exascale*“. Computer, 49(10):38–46, Oct 2016. ISSN 0018-9162.