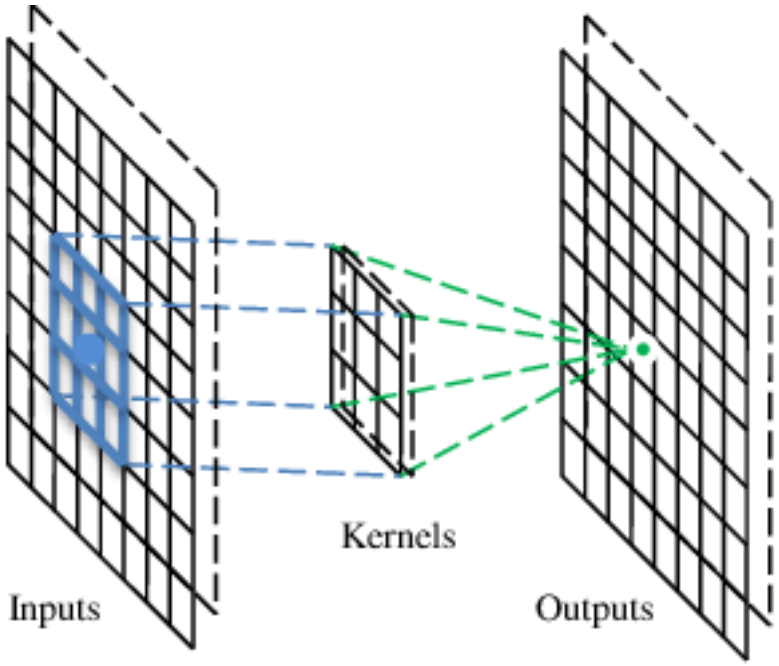
Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Υπολογιστικών Συστημάτων

”Αναλυση Επίδοσης Συνελικτικων Νευρωνικών Δικτύων σε Πολυπύρηνα Συστήματα”

Διπλωματική Εργασία

του

ΤΣΑΤΣΑΡΩΝΗ Γ. ΕΥΣΤΡΑΤΙΟΥ



Επιβλέπων: Γεώργιος Γκούμας
Επ. Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Υπολογιστικών Συστημάτων

”Αναλυση Επίδοσης Συνελικτικων Νευρωνικών Δικτύων σε Πολυπύρηνα Συστήματα”

Διπλωματική Εργασία

του

ΤΣΑΤΣΑΡΩΝΗ Γ. ΕΥΣΤΡΑΤΙΟΥ

Επιβλέπων: Γεώργιος Γκούμας
Επ. Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 6 Νοεμβρίου 2017.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Γκούμας	Ν. Κοζύρης	Ν. Παπασπύρου
Επ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ι.Π

Αθήνα, Οκτώβριος 2017

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Υπολογιστικών Συστημάτων

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

(Υπογραφή)

.....
Τ. Ευστρατίου

6 Νοεμβρίου 2017

0.1

Περίληψη

Η Βαθιά Μηχανική Μάθηση που εκφράζεται κυρίως μέσω των Βαθέων Νευρωνικών Δικτύων αποτελεί έναν πολύ διάσημο κλάδο της Επιστήμης Υπολογιστών. Η ικανότητα της να επιλύει προβλήματα που με τον συμβατικό προγραμματισμό θα ήταν αδύνατο να αντιμετωπιστούν την καθιστά ένα πολύ σπουδαίο εργαλείο. Όμως, τόσο η εφαρμογή όσο και η εκπαίδευσή τους αποτελούν πολύ κοπιαστικές και απαιτητικές διαδικασίες σε υπολογιστική ικανότητα και αποθηκευτική δυνατότητα. Παρόλο που, αρχικά, η χρήση Καρτών Γραφικών έδωσε πνοή στην υλοποίηση τους, τελικά το οικονομικό κόστος αυτών αλλά και η ανάγκη για εφαρμογή των Βαθέων Νευρωνικών Δίκτύων σε απλούστερες συσκευές μας οδήγησε στην περαιτέρω διερεύνηση των δυνατοτήτων που μας προσφέρουν οι επεξεργαστές.

Θέμα της παρούσας διπλωματικής εργασίας είναι η δημιουργία, εφαρμογή και σύγκριση μιας σειράς τεχνικών υλοποίησης των πλεόν απαιτητικών τμημάτων των Βαθέων Νευρωνικών Δικτύων: των Συνελικτικών Επιπέδων που βασίζονται στον δυσδιάστατο πολλαπλασιασμό πινάκων , με τελικό σκοπό την δημιουργία ενός συνόλου κανόνων επιλογής της κατάλληλης τεχνικής για κάθε επίπεδο του δικτύου, με βάση ορισμένα κριτήρια (ταχύτητα, μνήμη, αριθμός πράξεων). Ταυτόχρονα, θα διερευνήσουμε και τις δυνατότητες παραλληλοποίησης της Απευθείας Συνέλιξης ώστε να έχουμε μία εικόνα σύγκρισης με τις προηγούμενες τεχνικές.

Λέξεις Κλειδιά

Συνελικτικά Νευρωνικά Δίκτυα, Επίδοση, Caffe, πολυπύρηνα συστήματα, Gemm, Εφαρμογές Όρασης, Direct Convolution

Abstract

Deep Machine Learning, which is mainly expressed via the Deep Neural Networks, is a very popular field in Computer Science. The fact that they can solve many problems, which they would be very difficult to be solved using the conventional programming, makes it a very important tool. However, both their inference and their training are very tiring and demanding procedures in computing power and storage capability. Although, initially, the use of graphics cards gave impetus to implementing them, eventually their financial cost as well as the need for application of deep Neural Networks in simplest devices led us to further explore the possibilities offered by the processors.

Subject of this thesis is the creation, implementation and comparison of a series of technical realisation of the most demanding sections of deep Neural Networks: the Convolutional Layers, based on two-dimensional matrixes' multiplication, with the ultimate aim of creating a set of rules for the selection of appropriate technique for each layer of the network, based on certain criteria (speed, memory, number of operations). Simultaneously, we will investigate and the parallelism's possibilities of the Direct Convolution to an image comparison with the previous techniques.

Keywords

Convolutional Neuron Networks, Performance, Caffe, multicore systems, Gemm, Computer Vision Applications, Direct Convolution

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον καθηγητή κ. Νεκτάριο Κοζύρη για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία που μου έδωσε να την εκπονήσω στο εργαστήριο Υπολογιστικών Συστημάτων. Επίσης ευχαριστώ ιδιαίτερα την Δρ. Αθηνά Ελαφρού για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Τσατσαρώνη Ευστρατίου

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρα

Στον τομέα της Επιστήμης Υπολογιστών (Computer Science) υπάρχει ένας κλάδος που από την πρώτη στιγμή της δημιουργίας του, παρουσίασε εξαιρετικές προοπτικές και συνεπώς συγκέντρωσε όλα τα βλέμματα. Ο κλάδος αυτός είναι αυτός που ασχολείται με την **Μηχανική Μάθηση (Machine Learning)**. Σαν Μηχανική Μάθηση [1] ορίζουμε ένα σύνολο μεθόδων οι οποίες ”μαθαίνουν” σε μία εκάστοτε μηχανή (επεξεργαστική μονάδα) να εκτελεί μία συγκεκριμένη εργασία, όχι με βάση ένα πρόγραμμα, αλλά από τα ίδια τα δεδομένα. Ουσιαστικά πρόκειται για ένα σύνολο αλγορίθμων, οι οποίοι επιτρέπουν στους υπολογιστές να βρίσκουν πως πρέπει να εκτελούν μία εργασία ή να κάνουν κάποιες προβλέψεις ή παρατηρήσεις με βάση ένα σύνολο παραδειγμάτων , χωρίς την ανθρώπινη οδηγία (Ακριβές Πρόγραμμα Εντολών). Σαν αποτέλεσμα, σήμερα, με την επαρκή διάθεση δεδομένων και ισχυρών υπολογιστικών συστημάτων που μπορούν να υλοποιήσουν τις μεθόδους Μηχανικής Μάθησης, ο τομέας αυτός εξαπλώνεται σε ολοένα και περισσότερες πτυχές της καθημερινότητας. Πράγματι, οι μέθοδοι του Machine Learning χρησιμοποιούνται σε πληθώρα εφαρμογών, από την αναγνώριση ομιλίας (speech recognition)[2] και την πρόβλεψη του καιρού (weather prediction)[3], μέχρι και την ιατρική διάγνωση,[4],[5] και την εύρεση του καταναλωτικού προφίλ ενός πελάτη[6], προσδίδοντας, έτσι, στα δεδομένα, σήμερα, μεγάλη αξία.

Όσο, όμως, αυξάνεται η πολυπλοκότητα των προβλημάτων που θέλουμε να λύσουμε, οι απλές μέθοδοι της Μηχανικής Μάθησης χάνουν την αποδοτικότητα τους. Έτσι, αναπτύσσεται τα τελευταία χρόνια έντονα η τάση για την χρήση όλων και πιο περίπλοκων και υπολογιστικά απαιτητικών μεθόδων, οι οποίες ανήκουν στην λεγόμενη **Βαθιά Μηχανική Μάθηση (Deep Machine Learning)**[7]. Ιδιαίτερα σε έναν υποκλάδο της Μηχανικής Μάθησης τα **Νευρωνικά Δίκτυα (Neural Networks - ANNs)**[8], τα οποία είναι εμπνευσμένα από την δομή και λειτουργία του ανθρώπινου εγκεφάλου, αυτή η τάση βρίσκει μεγάλη εφαρμογή οδηγώντας με αυτό τον τρόπο στα **Βαθιά Νευρωνικά Δίκτυα (Deep Neural Networks - DNNs**)[9], την καρδιά της Βαθιάς Μηχανικής Μάθησης. Τα DNNs είναι πολύ διαδεδομένα και χρησιμοποιούνται σε μεγάλο αριθμό εφαρμογών, που ασχολούνται κυρίως με εικόνες και βίντεο. Για παράδειγμα, μπορούμε μέσω

μίας υποκατηγορίας του Deep Machine Learning, τα **Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs)**[10] να αναγνωρίσουμε το περιεχόμενο μίας εικόνας (image classification)[11] ή να προσδιορίσουμε ένα πρόσωπο σε μία εικόνα (face detection)[12], ενώ με μία άλλη υποκατηγορία τα **Περιοδικά Νευρωνικά Δίκτυα (Recurrent Neural Networks)**[13] μπορούμε να περιγράψουμε με λόγια τι γίνεται σε ένα βιντεο (Video to Word Translation)[14]. Αυτές οι εφαρμογές φαίνονται εύκολες για έναν άνθρωπο αλλά για έναν υπολογιστή είναι πολύ περίπλοκες και η Βαθιά Μηχανική Μάθηση έχει βελτιώσει κατά πολύ την επιδόση σε πληθώρα τέτοιων εφαρμογών. Βέβαια, πρέπει να τονιστεί ότι τονιστεί ότι πίσω από την επιτυχία της Βαθιάς Μηχανικής Μάθησης κρύβεται η ύπαρξη εκατομμυρίων δεδομένων για εκπαίδευση και πολύ εξελιγμένων επεξεργαστών (CPUs και GPUs) για να αναπτυχθεί.

Όλα τα παραπάνω προσδίδουν στα DNNs μεγάλες προοπτικές, όμως η εφαρμογή τους στην βιομηχανία θεωρείται ακόμα περιορισμένη. Και αυτό γιατί έχουν κάποια χαρακτηριστικά τα οποία περιορίζουν την **ευρωστία (robustness)** τους . Αρχικά, τα Νευρωνικά Δίκτυα αποτελούνται από ένα μεγάλο πλήθος παραμέτρων, οι οποίες κατά την φάση της **εκπαίδευσης (training phase)** προσαρμόζονται ώστε να οδηγούν στα σωστά αποτελέσματα κατά την φάση της **εφαρμογής (deployment phase)**. Στην περίπτωση, όμως, των DNNs ο αριθμός αυτός γιγαντώνεται, καθιστώντας δύσκολη και χρονοβόρα τόσο την εκπαίδευση όσο και την εφαρμογή. Για παράδειγμα, ένα δημοφιλές δίκτυο Βαθιάς Μηχανικής Μάθησης με μεγάλη ακρίβεια αποτελεσμάτων στην ταξινόμηση εικόνων σε κλάσεις, το VGG 16[15], χρησιμοποιεί συνολικά περισσότερους από 130.000.000 παραμέτρους. Ιδιαίτερα, στην φάση της εκπαίδευσης, όπου δοκιμάζουμε το δίκτυο για ένα μεγάλο μέγεθος δεδομένων εισόδου και επαναλαμβάνουμε την διαδικασία αρκετές φορές (training epochs), οι απαιτήσεις σε μνήμη είναι πολύ μεγάλες. Συγκεκριμένα, αν οι παράμετροι είναι αποθηκευμένες με τύπο κινητής υποδιαστολής απλής αρκίβειας 32-bit, τότε το μοντέλο αυτό χρειάζεται 496MB για την αποθήκευσή του.

Εκτός, βέβαια, από το μεγάλο μέγεθος των μοντέλων, τα DNNs απαιτούν και μηχανήματα με μεγάλη υπολογιστική ισχύ. Αναλυτικότερα, ένα DNN που ταξινομεί κάποιες εικόνες σε κλάσεις αναλόγως το περιεχόμενό τους μπορεί εύκολα να κληθεί να εκτελέσει πάνω από 90 δισεκατομμύρια πράξεις κατά την φάση της εκπαίδευσης. Συνεπώς, όταν οι σύγχρονοι υπολογιστές χρησιμοποιούν ρολόγια με το πολύ 5 δισεκατομμύρια κύκλους το δευτερόλεπτο, συμπεραίνουμε ότι απαιτούνται τουλάχιστον 18 δευτερόλεπτα για κάθε εικόνα. Όμως, για την εκπαίδευση του δικτύου χρειάζονται περίπου 1 εκατομμύριο εικόνες και συνήθως η διαδικασία επαναλαμβάνεται για 10 εποχές. Επομένως, συνολικά για την πλήρη εκπαίδευση ενός DNN θα απαιτούνταν 180 εκατομμύρια δευτερόλεπτα ή 4,7 χρόνια.

Παρόλο που οι παραπάνω υπολογισμοί γίνονται χωρίς κάποια παραλληλία και παρότι έχουν αναπτυχθεί αρκετοί αλγόριθμοι που βελτιώνουν την υπολογιστική απαίτηση των DNNs, είναι σαφές από τα παραπάνω ότι η εκτέλεση τους απαιτεί πολύ εξελιγμένες και αποδοτικές υπολογιστικές μονάδες. Την λύση σε αυτό το πρόβλημα ήρθαν να δώσουν οι κάρτες γραφικών (GPUs). Δεν θα ήταν υπερβολή να αναφέρουμε ότι χωρίς την χρήση των GPUs η εφαρμογή των DNNs θα ήταν ελάχιστη και ουσιαστικά ο όλος

τομέας του Deep Machine Learning θα ήταν ”νεκρός”. Αναλυτικότερα, η δομή των GPUs, που αποτελούνται από πολλούς μικρούς επεξεργαστές, τους καθιστά κατάλληλους για την εκτέλεση πολλαπλών μικρών εργασιών, όπως συμβαίνει στον πολλαπλασιασμό πινάκων. Από την στιγμή, βέβαια, που αρχίσε η εκτέλεση της συνέλιξης στα συνελικτικά δίκτυα μέσω του μετασχηματισμού σε πίνακες και στην συνέχεια του πολλαπλασιασμό αυτών[16], έκανε την χρήση των GPUs όλο και πιο αποτελεσματική. Σήμερα η εκπαίδευση ενός DNN με χρήση GPU διαρκεί κατά μέσο όρο από μερικές ημέρες έως και μερικές εβδομάδες, μία εξαιρετική βελτίωση σε σχέση με τα προηγουμένως αναφερθέντα νούμερα.

Παρόλα αυτά, το γεγονός ότι η εφαρμογή του Deep Machine Learning ολοένα και εξαπλώνεται και μάλιστα σε συσκευές όπου η χρήση των GPUs είναι αδύνατη, καθώς και ότι οι GPUs που μπορούν να υποστηρίξουν τα παραπάνω (General Purpose GPUs - GPGPUs [17]) είναι εξαιρετικά ακριβές, μας οδηγεί στο να αναζητήσουμε εναλλακτικές επιλογές. Όπως είναι λογικό, εκεί που τελειώνει το υλικό ξεκινά το λογισμικό. Έτσι, η προσπάθεια πλέον επικεντρώνεται στην εύρεση αποδοτικότερων αλγορίθμων πάνω σε CPUs που θα εκτελούν τα DNNs και κυρίως τα Συνελικτικά Επίπεδα αυτών, ώστε να επιτύχουμε ακριβέστερα αποτελέσματα με λιγότερους πόρους. Ιδιάιτερα τώρα με την εξέλιξη των επεξεργαστών αλλά και την αυξανόμενη χρήση των λεγόμενων Coprocessors[18], οι οποίοι προσφέρουν μεγαλύτερο αριθμό πυρήνων από τους κλασικούς CPUs και βοηθούν σημαντικά στην γρήγορη εκτέλεση παράλληλων τμημάτων κώδικα, λειτουργώντας επικουρικά στον κύριο επεξεργαστή, η εφαρμόγη των DNNs πάνω σε CPUs αποκτά μία ξεχωριστή προοπτική. Αξίζει, βέβαια, να αναφερθεί ότι στην λίστα με τους μεγαλύτερους υπερυπολογιστές παγκοσμίως (Top500 Supercomputer Sites List) υπάρχει μεγάλος αριθμός υπερυπολογιστών χωρίς την χρήση GPUs. Επομένως, η έρευνα πάνω σε αποδοτικότερους αλγορίθμους που να απευθύνονται κυρίως σε CPUs είναι επομένως κάτι παραπάνω από χρήσιμη.

1.2 Σχετικά με την διπλωματική

1.2.1 Στόχος της διπλωματικής

Στόχος της συγκεκριμένης διπλωματικής είναι να προσφέρει μία λεπτομερής ανάλυση και αξιολόγηση της επίδοσης διαφόρων τεχνικών υλοποιήσης των συνελικτικών επιπέδων, που αποτελούν το μεγαλύτερο ποσοστό του χρόνου εκτέλεσης των CNNs, καθώς και η βελτιστοποίηση τους πάνω σε σύγχρονα πολυπύρηνα υπολογιστικά συστήματα.

1.2.2 Περιβάλλον της διπλωματικής

Όπως αναφέραμε παραπάνω, η έρευνα στον τομέα αυτό είναι μεγάλη τόσο σε επίπεδο υλικού όσο και σε αυτό του λογισμικού. Ιδιαίτερα στο δεύτερο έχουν αναπτυχθεί και υλοποιηθεί σημαντικός αριθμός από frameworks και βιβλιοθηκών (libraries) που αφορούν την εκτέλεση CNNs (Convolutional Neural networks), όπως το Torch[19],

το Theano[20], το cuda-convnet (Krizhevsky (2014)), το TensorFlow[21]] αλλά και το Caffe[22]. Τα συγκεκριμένα frameworks υποστηρίζουν εκτέλεση των CNNs τόσο σε περιβάλλον CPU, όσο και GPU.

Στην διπλωματική αυτή, επικεντρωνόμαστε στο περιβάλλον των CPUs. Όλη η ανάλυση και οι τεχνικές που εφαρμόζουμε είναι κατάλληλα δομημένες, ώστε να εκμεταλλεύονται τα πλεονεκτήματα ενός πολυπύρηνου επεξεργαστή. Συγκεκριμένα, κάνουμε χρήση ενός πολυπύρηνου συστήματος 14 πυρήνων, που υποστηρίζει hyperthreading τον οποίο μας τον παρέχει το εργαστήριο όπου και εκτελείται η διπλωματική, ενώ παράλληλα λαμβάνουμε μετρήσεις από ένα σύστημα που υποστηρίζει πολύ μεγαλύτερη παραλληλία, όπως αυτό του Intel Xeon Phi KnL7250[23], το οποίο μπορεί να εκτελέσει 272 νήματα παράλληλα.

Η προσπάθεια μας υλοποιήθηκε πάνω στο framework του Caffe και συγκεκριμένα μία έκδοση αυτού που βελτιώνει την επίδοση του πάνω στον Coprocessor της Intel: τον INTEL XEON PHI, την intel-caffe[24], [25]. Η επιλογή αυτή δεν ήταν τυχαία. Το Caffe είναι ένα open-source framework, γραμμένο σε μία εύκολα διαχειρίσιμη γλώσσα: την C++, και οι δημιουργοί του είναι ανοιχτοί σε οποιαδήποτε προσπάθεια εξέλιξης του από οπουδήποτε και αν αυτή έρχεται[26]. Ακόμη, η έκδοση που στοχεύει στα προϊόντα της Intel μας προσφέρει μια πολύ καλή βάση για να εφαρμόσουμε τις βελτιώσεις μας, αλλά και την δυνατότητα εύκολης προσαρμογής στους επεξεργαστές ππου χρησιμοποιούμε. Το Caffe αυτή την στιγμή βρίσκεται στην κορυφή των προτιμήσεων για τους ερευνητές των CNNs.

1.2.3 Δομή Διπλωματικής

Η συγκεκριμένη διπλωματική εργασία αποσκοπεί στην κατανόηση σε βάθος πολλών ζητημάτων θεωρίας αλλά και πρακτικών θεμάτων, πριν ξεκινήσει η περιγραφή των δικών μας υλοποιήσεων αλλά και η εξαγωγή των συμπερασμάτων. Για αυτόν τον λόγο έγινε προσπάθεια να καλυφθούν όσον το δυνατό καλύτερα όλα τα θέματα και τα εργαλεία με τα οποία ασχοληθήκαμε, ώστε να μην υπάρχουν τυχόν κενά στον αναγνώστη.

Συγκεκριμένα, μετά από την σύντομη εισαγωγή στα κίνητρα και το περιβάλλον της εργασίας που προηγήθηκε, ακολουθεί στο κεφάλαιο 2 μια συνοπτική αλλά περιεκτική περιγραφή όλου του αναγκαίου θεωρητικού υπόβαθρου. Στην συνέχεια, στο κεφάλαιο 3 πραγματοποιείται μία ανάλυση των εργαλείων που χρησιμοποιήθηκαν αλλά και των Βαθιών Νευρωνικών Δικτύων (DNNs) που εκτελέστηκαν, ενώ στο κεφάλαιο 4 περιγράφουμε σε βάθος την λογική, τα πλεονεκτήματα και την διαδικασία κάθε μεθόδου που υλοποιήθηκε στο πλαίσιο αυτής της διπλωματικής. Ακολουθεί το κεφάλαιο 5 στο οποίο παρουσιάζονται οι μετρήσεις, τα αποτελέσματα τους καθώς και τα κύρια συμπεράσματα τα οποία εξάγονται, ενώ, τέλος, καταλήγουμε με το κεφάλαιο 6, στο οποίο συμπυκνώνουμε σαν επίλογο όλα τα αποτελέσματα και δημιουργούμε κάποιους κανόνες επιλογής της κατάλληλης μεθόδου για τις κατάλληλες συνθήκες.

Μέρος I

Θεωρητικό Μέρος

Κεφάλαιο 2

Θεωρητικό υπόβαθρο

Στο κεφάλαιο αυτό παρουσιάζεται συνοπτικά το θεωρητικό τμήμα της εργασίας πάνω στο οποίο βασιστήκαμε για την δημιουργία της. Συγκεκριμένα, θα αναφερθούμε γενικά σε Artificial Neural Networks (ANNs - Τεχνητά Νευρωνικά Δίκτυα), σε Convolutional Neural Networks (CNNs - Συνελικτικά Νευρωνικά Δίκτυα), σε Deep Neural Networks (DNNs) αλλά και θα περιγράψουμε τις εφαρμογές τους και τις προκλήσεις με τις οποίες είμαστε αντιμέτωποι.

2.1 Artificial Neural Networks (ANNs)

2.1.1 Τι είναι τα Artificial Neural Networks

Η έρευνα στον τομέα της Επιστήμης των Υπολογιστών (Computer Science) συνεχώς μεγαλώνει. Ο σύγχρονος ηλεκτρονικός υπολογιστής είναι σε θέση να επιλύει πληθώρα προβλημάτων και μάλιστα μέσω έξυπνων αλγορίθμων να το κάνει αυτό πολύ αποδοτικά. Ωστόσο, υπάρχουν αρκετά προβλήματα που για την κλασική λογική των υπολογιστών θεωρούνται περίπλοκα. Για παράδειγμα, ενώ για να υπολογίσει ο υπολογιστής την ρίζα του 1.522.756 χρειάζεται ελάχιστο χρόνο (είναι το 1.234!), ένα πρόβλημα εντόπισμού ενός ανθρώπου στο πλήθος είναι για τον υπολογιστή πάρα πολύ δύσκολο. Είναι προφανές ότι για τέτοιου είδους προβλήματα έπρεπε να στραφούμε σε κάτι πιο καινοτόμο και να χρησιμοποιήσουμε διαφορετική λογική.

Την έμπνευση για αυτήν την λογική την προσφέρει ένα πολύ πιο αναπτυγμένο σύστημα, ο ανθρώπινος εγκέφαλος, και ιδιαίτερα το σύστημα της ανθρώπινης όρασης. Το σύστημα αυτό δεν δομείται με βάση την εκτέλεση εντολών αλλά, αντιθέτως, δέχεται κάποια δεδομένα από το αισθητήριο όργανο της όρασης, το μάτι τα επεξεργάζεται και παράγει συμπεράσματα. Έτσι, στο παραπάνω παράδειγμα του εντοπισμού ενός προσώπου στο πλήθος, αναζητούμε σε κάθε άτομο στο πλήθος κάποια χαρακτηριστικά, π.χ. το χρώμα των ρούχων του ή κάποιο χαρακτηριστικό του προσώπου του, κ.τ.λ., που θα μας οδηγήσουν σε αυτό, και αποφασίζουμε αν το συγκεκριμένο άτομο είναι το επιθυμητό. Η συγκεκριμένη λογική είναι απλή για το σύστημα του εγκεφάλου.

Με βάση αυτή την λογική δημιουργήσαμε και τα νευρωνικά δίκτυα. Σύμφωνα με τον

ορισμό του Simon Haykin στο [27]:

Ένα νευρωνικό δίκτυο είναι ένας τεράστιος παράλληλος υπολογιστής με κατανεμημένη αρχιτεκτονική, ο οποίος αποτελείται από απλές μονάδες επεξεργασιας και έχει από την φύση του τη δυνατότητα να αποθηκεύει εμπειρική γνώση και να την καθιστά διαθέσιμη για χρήση. Μοιάζει με τον ανθρώπινο εγκέφαλο σε δύο σημεία:

1. **Το δίκτυο προσλαμβάνει τη γνώση από το περιβάλλον του, μέσω μίας διαδικασίας μάθησης.**
2. **Η ισχύς των συνδέσεων μεταξύ των νευρώνων, που αποκαλείται συνοπτικό βάρος, χρησιμοποιείται για την αποθήκευση της γνώσης που αποκτιέται.**

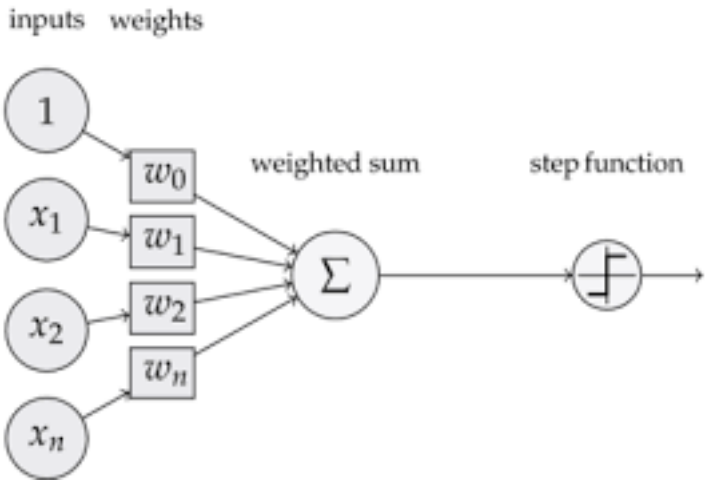
2.1.2 Η Δομή των ANNs

Για να γίνει πιο σαφής η περιγραφή της δομής ενός νευρωνικού δικτύου θα αναλύσουμε αρχικά την δομή του πιο απλού νευρωνικού, του Perceptron, και στην συνέχεια θα γενικέυσουμε και στα πιο περίπλοκα.

Απλό Perceptron

Το απλό Perceptron είναι η πρώτη προσπάθεια δημιουργίας ενός νευρωνικού δικτύου. Υλοποιήθηκε το 1957 από τον Frank Rosenblatt [28]. Αποτελεί μια σπουδαία προσπάθεια ομώς οι δυνατότητες του είναι αρκετά περιορισμένες.

Το Perceptron, ο οποίος θα μπορούσε να αντιστοιχηθεί με έναν νευρώνα του νευρικού μας δικτύου, αποτελείται από τα βάρη(weights), έναν αθροιστή (sum) και μια συνάρτηση ενεργοποίησης (activation function).



Σχήμα 2.1: *Perceptron*

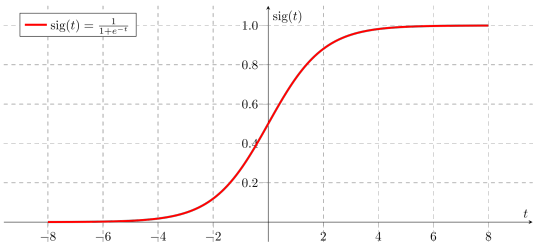
Ένα Perceptron, όπως αυτόν που φαίνεται στο παραπάνω σχήμα, παράγει μία τιμή εξόδου μέσω της εξής feed-forward λειτουργίας :

1. Δέχεται σαν είσοδο κάποια δεδομένα, έστω x1,x2,....xn, άλλα και μία σταθερά (bias), της οποίας το αντίστοιχο βάρος είναι πάντα ένα(1).

2. Στην συνέχεια πολλαπλασιάζει τις εισόδους αυτούς με τα βάρη που τους αντιστοιχούν και υπολογίζει το άθροισμα τους.

3. Ύστερα, το άθροισμα του προηγούμενου βήματος αποτελεί την είσοδο στην συνάρτηση ενεργοποίησης. Αναλόγως την τιμή του αθροίσματος η συνάρτηση αποφασίζει αν θα πυροδοτήσει (fire) το νευρώνας ή αν θα τον διατηρήσει ανενεργό. Συνήθως η τιμή της εξόδου ανήκει στα διαστήματα [-1,1] ή [0,1] αναλόγως της συνάρτησης ενεργοποίησης.

Σαν συνάρτηση ενεργοποίησης συνήθως χρησιμοποιούμε την σιγμοειδή, η οποία πυροδοτεί την έξοδο για θετική τιμή του αθροίσματος, ενώ για αρνητική την καθιστά μηδενική, όπως φαίνεται και στο σχήμα:



Σχήμα 2.2: *Σιγμοειδής Συνάρτηση (Sigmoid Function)*

Η σημαντικότερη ιδιότητα των νευρωνικών δικτύων, όμως, δεν είναι εμφανής στην παραπάνω διαδικασία. Πρόκειται, φυσικά για την προσαρμοστικότητα, την ιδιότητα, δηλαδή, των δικτύων να προσαρμόζονται πάνω στο πρόβλημα ώστε να το επιλύσουν. Με την feed-forward διαδικασία, ουσιαστικά, "τρέχουμε" το Perceptron ώστε να παρατηρήσουμε την τιμή εξόδου δεδομένων των εισόδων που του δώσαμε, είναι μία διαδικασία πειραματισμού για το Perceptron δηλαδή. Για να προσαρμόσουμε το Perceptron, και γενικότερα ένα νευρωνικό δίκτυο, χρησιμοποιούμε μία διαφορετική διαδικάσια που ονομάζεται Μάθηση (Learning). Αξίζει εδώ να σημειώσουμε ότι η προσαρμοστικότητα των νευρωνικών οφείλεται ξεκάθαρα στα βάρη. Αύτα είναι που τροποποιούνται δεδομένου του προβλήματος, ενώ η υπόλοιπη δομή παραμένει αμετάβλητη. Με άλλα λόγια, για να μπορεί να λύσει ένα Perceptron ένα συγκεκριμένο πρόβλημα πρέπει να βρούμε τις κατάλληλες τιμές για τα βάρη ώστε να δίνει τις σωστές εξόδους.

Γενικότερα, έχουμε τρία είδη μάθησης [29], [30]:

- **Επιβλεπόμενη μάθηση (Supervised Learning)**: Σε αυτήν την διαδικασία είναι απαραίτητη η ύπαρξη ενός "δασκάλου", ο οποίος θα παρέχει στο Perceptron κάθε φορά τις σωστές απαντήσεις που θα επρεπε αυτό να δώσει. Έτσι, με βάση αυτές υπολογίζουμε το σφάλμα (error) των απαντήσεων και μεταβάλλουμε τα βάρη σύμφωνα με αυτό. Η διαδικασία αυτή είναι η πλέον διαδεδομένη και θα την περιγράψουμε αναλυτικότερα παρακάτω.

- **Μη Επιβλεπόμενη μάθηση (Unsupervised Learning)**: Εφαρμόζεται σε περιπτώσεις όπου η ύπαρξη του των απαντήσεων δεν είναι δυνατή. Σαν συνέπεια, κατά

την μάθηση αυτή προσπαθεί το Perceptron να αναγνωρίσει κάποια όμοια πρότυπα ανάμεσα στα δεδομένα. Η μέθοδος αυτή είναι ιδιαίτερα χρήσιμη σε παραδείγματα ομαδοποίησης (clustering) των δεδομένων με βάση ορισμένες κοινές ιδιότητες, όμως δεν είναι πολύ δημοφιλής στα νευρωνικά δίκτυα.

- **Επιβαλλόμενη μάθηση (Reinforcement Learning)**: Εδώ η λογική είναι διαφορετική και βασίζεται στην παρατήρηση. Μαθαίνουμε, ουσιαστικά, στο Perceptron να αναγνωρίζει τι πρέπει να κάνει αναλόγως την είσοδο και το αποτέλεσμα. Ας σκεφτούμε ένα παιδί αρκετά μικρό σε ηλικία για να καταλάβει τι είναι το σωστό και το λάθος. Όταν το παιδί πλησιάζει έναν γκρεμό τότε θα παρατηρεί την αντίδραση των μεγαλύτερων που θα προσπαθήσουν να το απομακρύνουν και θα καταλάβει ότι αυτό είναι λάθος. Αντίθετα, όταν θα προσπαθήσει να παίξει με ένα άλλο παιδί και θα παρατηρήσει ότι κανείς δεν αντιδρά με αυτό, τότε θα συνεχίσει να το κάνει και θα το θεωρήσει καλό. Ακριβώς αντίστοιχη είναι και η κατάσταση σε αυτού του είδους την μάθηση. Βέβαια, απαιτείται και μία δομή που να παράγει την παρατήρηση.

Όπως προαναφέραμε, από τις παραπάνω μεθόδους περισσότερο διαδεδομένη είναι η Επιβλεπόμενη Μάθηση. Η εφαρμογή της σε έναν Perceptron είναι αρκετά απλή και βασίζεται σε μία διαδικασία που ονομάζεται Backward, καθώς πλεόν η ροή δεν γίνεται από την είσοδο στην έξοδο αλλά από την έξοδο στην πρώτη. Αναλυτικότερα, η διαδικασία αυτή για ένα Perceptron έχει ως εξής [30]:

1. Αρχικά απαιτείται μία βάση δεδομένων (δάσκαλος - training data set), η οποία θα περιέχει κάποια παραδείγματα τιμών εσόδου και τις αντίστοιχες σωστές εξόδους για αυτές.

2. Στην συνέχεια, αρχικοποιούμε τα βάρη τυχαία.

3. Εκτελούμε για όλα τα στοιχεία του ”δασκάλου ” τα βήματα 4, 5 και 6.

4. Εκτελείται για κάποιες τιμές εισόδου από την βάση δεδομένων του ”δασκάλου” ένα feed-forward run του Perceptron ώστε να βρούμε τις αντίστοιχες πειραματικές τιμές εξόδου.

5. Υπολογίζουμε το σφάλμα των πειραματικών τιμών με τις σωστές εξόδους μέσω της διαφοράς τους:

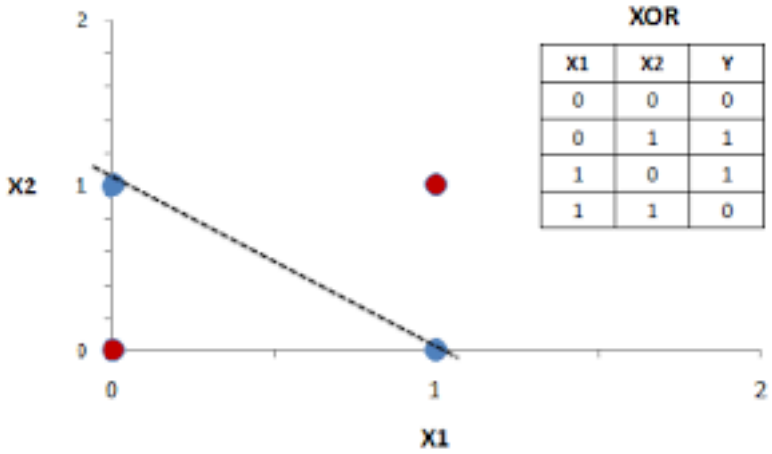
$$\text{Σφάλμα} = \text{Επιθυμητό Αποτέλεσμα} - \text{Παραγόμενο Αποτέλεσμα} \tag{2.1}$$

6. Μεταβάλουμε τις τιμές των βαρών κατάλληλα μέσω της σχέσης:

$$\begin{aligned} \text{Νέο Βάρος} &= \text{Παλιό Βάρος} + \text{Μεταβολή Βάρους} \\ \text{Μεταβολή Βάρους} &= \text{Learing Rate} * \text{Σφάλμα} * \text{Είσοδος} \end{aligned} \tag{2.2}$$

, όπου Learing rate είναι μια παράμετρος που καθορίζει πόσο γρήγορα θα μεταβάλουμε τα βάρη.

Με αυτό τον τρόπο εκπαιδεύονται τα Perceptron ώστε να επιλύουν συγκεκριμένα προβλήματα. Ωστόσο, οι δυνατότητες ενός Perceptron είναι πολύ περιορισμένες, καθώς είναι ικανά να αντιμετωπίσουν μόνο γραμμικώς διαχωριζόμενα προβλήματα. Χαρακτηριστικό τέτοιο παράδειγμα αποτελεί το πρόβλημα XOR. Σύμφωνα με αυτό για όλους τους συνδιασμούς των εισόδων η έξοδος θα πρέπει να μεταβάλλεται με βάση τον παρακάτω πίνακα:

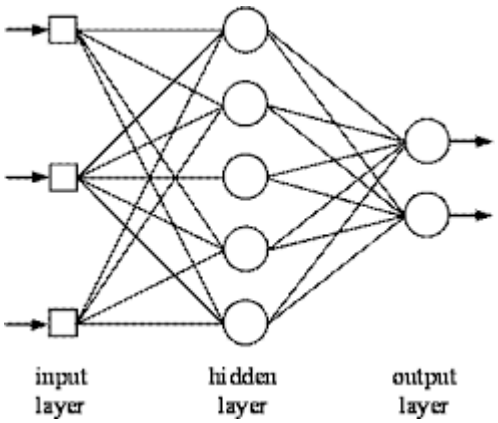


Σχήμα 2.3: *Πίνακας Τιμών Προβλήματος XOR*

Όπως είναι εμφανές και στο παραπάνω σχήμα, το πρόβλημα αυτό δεν είναι γραμμικώς διαχωριζόμενο καθώς δεν μπορούμε να ξεχωρίσουμε τις περιπτώσεις χρησιμοποιώντας μόνο μία ευθεία γραμμή. Αντιθέτως χρειάζονται δύο ευθείες ή έστω μία καμπυλοειδής γραμμή. Για να αντιμετωπίσουμε αυτού του είδους τα προβλήματα χρειαζόμαστε κάτι πιο περίπλοκο, το οποίο και περιγράφουμε στην συνέχεια.

2.1.3 Perceptron Πολλαπλών Επιπέδων (Κλασικά Νευρωνικά Δίκτυα)

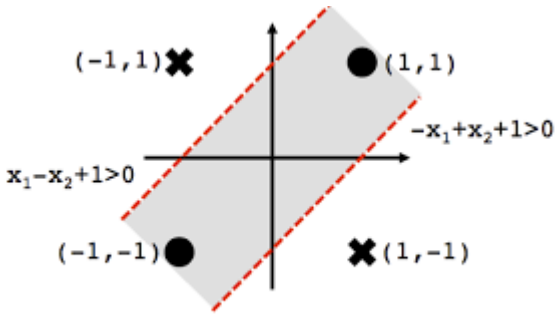
Όπως έχουμε ήδη αναφέρει ένα απλό Perceptron είναι ικανό μόνο σε πολύ απλά προβλήματα και συνεπως δεν έιναι και πάρα πολύ χρήσιμο. Για περισσοτερο σύνθετα προβλήματα (όπως το πρόβλημα XOR) χρειαζόμαστε πιο σύνθετες δομές, τα Perceptron Πολλαπλών Επιπέδων [31], τα συνήθη νευρωνικά δίκτυα δηλαδή.



Σχήμα 2.4: *Perceptron Δύο Επιπέδων*

Γραφικά μπορούμε να παραστήσουμε ένα Νευρωνικό Δίκτυο, όπως το παραπάνω σχήμα. Στην αρχή εμφανίζονται οι είσοδοι του νευρωνικού, ενώ οι κυκλικοί κόμβοι αντιπροσωπεύουν νευρώνες (Perceptrons). Στο τέλος πάντα υπάρχουν οι έξοδοι. Συνήθως τα Νευρωνικά είναι Fully-Connected, δηλαδή κάθε νευρώνας συνδέεται με όλους τους νευρώνες του επόμενου επιπέδου. Σε κάθε σύνδεσμο αντιστοιχεί ένα βάρος(weight), το οποίο και τροποποιείται κατά την μάθηση. Η ροή των δεδομένων είναι μόνο προς τα δεξιά κατά την feed-forward διαδικασία και μόνο προς τα αριστερά κατά την Backward. Επομένως, σε ένα Νευρωνικό Δίκτυο δεν επιτρέπεται να έχουμε κυκλικούς γράφους καθώς αυτό θα ευνοούσε μη πεπερασμένα (άπειρα) περάσματα. Οι νευρώνες που βρίσκονται στην ίδια κάθετη συστοιχία αποτελούν ένα επίπεδο νευρώνων. Φυσικά, όταν αναφερόμαστε σε ένα Νευρωνικό Δύο Επιπέδων, δεν θεωρούμε το επίπεδο της εισόδου σε αυτό, αλλά εννούμε μόνο τα επίπεδα που περιέχουν νευρώνες.

Παρατηρούμε ότι έχουμε προσθέσει ένα παραπάνω επίπεδο νευρώνων ανάμεσα στην είσοδο και στην έξοδο, το κρυφό (hidden) επίπεδο. Αυτό το επίπεδο μας δίνει ορισμένες πρόσθετες δυνατότητες. Στο πρόβλημα XOR, για παράδειγμα, με αυτό το πρόσθετο επίπεδο ποιοτικά μπορούμε να χρησιμοποήσουμε μία ακόμα ευθεία στο διαχωρισμό των κλάσεων, όπως φαίνεται παρακάτω:

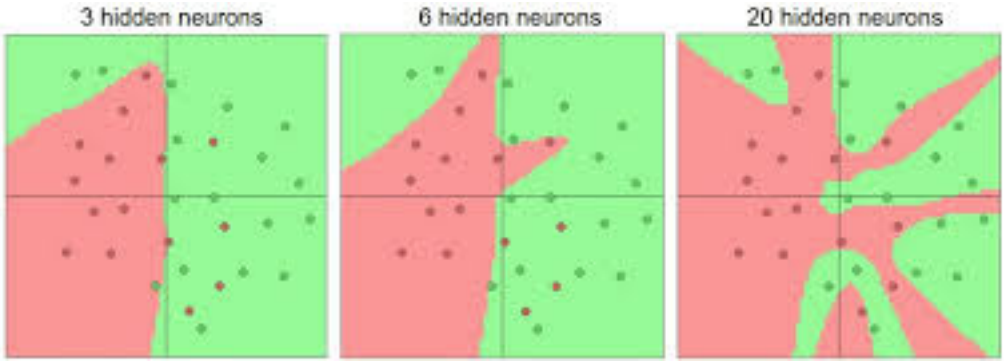


Σχήμα 2.5: *Διαχωρισμός των κλάσεων εξόδου στο πρόβλημα XOR με χρήση Perceptron Δύο Επιπέδων*

Αντίστοιχα, σε ένα πρόβλημα που απαιτούνται περισσότερες ευθείες για τον διαχωρισμό των εξόδων προσθέτουμε αντίστοιχο αριθμό κρυφών επιπέδων. Βέβαια, αυτή η περιγραφή είναι ποιοτική και έχει σαν στόχο την κατανόηση της σημασίας των κρυφών επιπέδων.

Με την προσθήκη, όμως, όλο και περισσότερων επιπέδων εγκυμωνεί ένας μεγάλος κίνδυνος. Στο training data set που θα χρησιμοποιήσουμε για την μάθηση του νευρωνικού, μπορεί να υπάρχουν δείγματα τα οποία να μην είναι τόσο σωστά, είτε επειδή είναι μακρυά από αυτά που θεωρούμε αντιπροσωπευτικά, είτε επειδή περιέχουν πολύ θόρυβο που τα αλλοιώνει. Έτσι είναι πολύ πιθανό να αποκλίνουν γραφικά από την περιοχή που αυτά ανήκουν, και να αυξάνουν έτσι την πολυπλοκότητα του προβλήματος. Σε αυτή την περίπτωση αν προσθέσουμε παραπάνω επίπεδα στο νευρωνικό ή αν το εκπαιδεύσουμε για παραπάνω εποχές(ο αριθμός των εποχών δηλώνει πόσες φορές επαναλαμβάνουμε την εκπαίδευση του νευρωνικού σε όλο το training data set) ώστε να εντάξουμε και τα ”λάθος” δείγματα, τότε το αποτέλεσμα δεν θα είναι σωστό. Αυτό

το πρόβλημα ονομάζεται πρόβλημα υπερεκπαίδευσης (overfitting) και για αυτό συνήθως σταματούμε την διαδικασία της μάθησης σε σημείο όπου υπάρχει μία γραφική συνεκτικότητα μεταξύ των εξόδων, όπως φαίνεται και στο σχήμα:



Σχήμα 2.6: *Πρόβλημα Overfitting. Για 20 επίπεδα έχουμε πλήρη αντιπροσώπευση αλλά το πρόβλημα γίνεται πολύ περίπλοκο.*

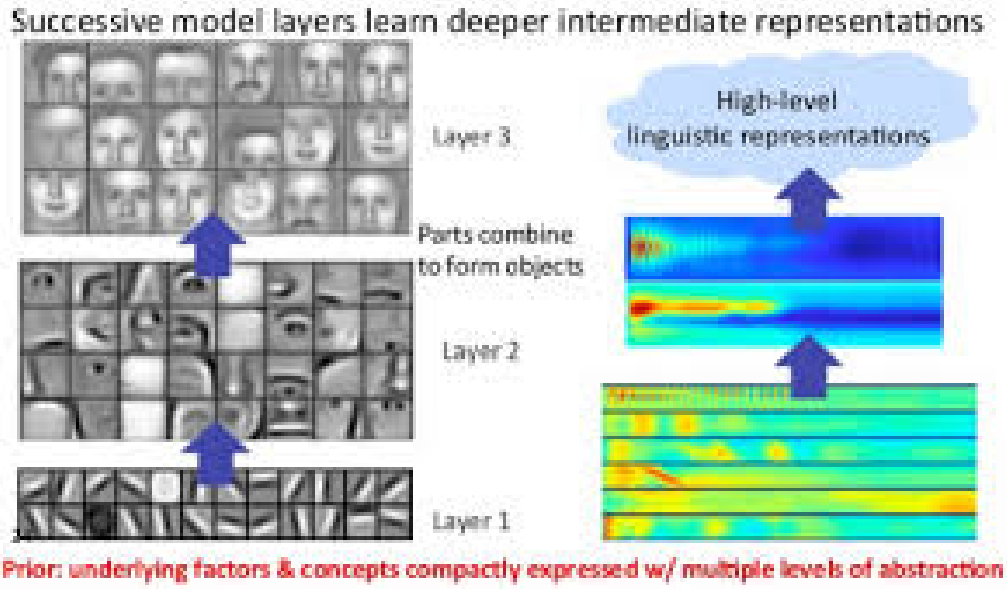
Τέλος, όσον αφορά την μάθηση των Νευρωνικών Δικτύων, αυτή είναι αρκετά πιο περίπλοκη από εκείνη στο απλό Perceptron, καθώς χάνει την φυσική σημασία της λόγω των πολλαπλών επιπέδων. Για αυτήν είναι πολύ διαδεδομένη η μέθοδος του backpropagation [32].

2.2 Βαθιά Νευρωνικά Δίκτυα (Deep Neural Networks - DNNs)

Όπως προκύπτει και από την ίδια την έννοια ένα Βαθύ Νευρωνικό Δίκτυο (DNN) διαφέρει από ένα απλό Νευρωνικό λόγω της ύπαρξης του βάθους (Depth). Σαν βάθος ορίζουμε στα Νευρωνικά Δίκτυα τον αριθμό των κρυφών επιπέδων, και, επομένως, σαν DNN ένα Νευρωνικό με μεγαλύτερο από ένα αριθμό κρυφών επιπέδων.

Μία σημαντική ιδιότητα των DNNs είναι η λεγόμενη ιεραρχία χαρακτηριστικών (features hierarchy). Σύμφωνα με αυτή κάθε κρυφό επίπεδο νευρώνων εκπαιδεύει πάνω σε ένα συγκεκριμένο είδος χαρακτηριστικών, όπως οι ακμές, τα μάτια αν εκπαιδεύουμε εικόνες με πρόσωπα,..,κ.τ.λ,[33], [34]. Όσο πιο βαθύ είναι ένα επίπεδο τόσο πιο σύνθετα είναι τα χαρακτηριστικά πάνω στα οποία αυτό εκπαιδεύεται. Συνεπώς, υπάρχει μια διαδικασία επεξεργασίας και ανάλυσης των εισόδων από απλά χαρακτηριστικά αρχικά (ακμές,γωνίες,...,κ.τ.λ.) σε όλο και πιο σύνθετα (περιοχές της εικόνας με συγκεκριμένες ομοιότητες,...,κ.τ.λ.). Αυτό δίνει την δυνατότητα στα DNNs να χειρίζονται μεγάλα training data sets χωρίς απαραίτητα αυτά να έχουν δεχθεί κάποια προεπεξεργασία. Το πιο θαυμαστό από όλα είναι πως αυτήν την εξαγωγή χαρακτηριστικών (feature extraction) το DNN είναι ικανό να το κάνει αυτόματα, χωρίς να χρειάζεται να του το υποδείξει κάποιος ανθρώπινος παράγοντας, κάτι που δεν συμβαίνει σε καμία άλλη μέθοδο αναγνώρισης προτύπων.

Σαν συνέπεια, τα DNN είναι ικανά να διαχειριστούν πολύ μεγάλα training sets και μάλιστα με πολύ θετικά αποτελεματα. Για παράδειγμα ένα DNN μπορεί να πάρει 1.000.000 εικόνες και να τις διαχωρίσει σε τάξεις αναλόγως το περιεχόμενο τους. Έτσι, μία εικονα



Σχήμα 2.7: *Η ιεραρχία των χαρακτηριστικών σε ένα DNN*

μίας γάτας θα παέι στην κλάση με αυτές των αιλουροειδών, ενώ μία εικόνα ενός αυτοκινήτου θα πάει σε μία πολύ διαφορετική κλάση από την προηγούμενη.

Ένα ακόμη σπουδαίο χαρακτηριστικό των DNN αποτελεί το γεγονός ότι, ενώ όλες οι άλλες μέθοδοι εμφανίζουν κορεσμό όταν εκπαιδεύονται σε πολύ μεγάλο αριθμό εικόνων, αυτό δεν συμβαίνει με τα DNNs. Σε αυτά όσο μεγαλύτερο είναι το training set τόσο πιο καλά εκπαιδεύεται το δίκτυο και τόσο καλύτερα τα αποτελέσματα.

Από όλα αυτά συμπεραίνουμε ότι τα DNNs είναι ένα πολύ χρήσιμο εργαλείο σε πολλούς τομείς. Για αυτό και οι εφαρμογές τους είναι πολλαπλές. Ενδεικτικά αναφέρουμε τα εξής:

- **Image Classification**: Στην ταξινόμηση εικόνων σε κλάσεις η χρήση των DNNs είναι ιδανική, όπως αποφαίνεται και από τα παραπάνω. Σε αυτήν κάθε νευρώνας του επιπέδου εξόδου αντιστοιχεί στην πιθανότητα η εικόνα να αντιστοιχεί στην κάθε κλάση. Άρα θα πρέπει να υπάρχει ένας νευρώνας εξόδου για κάθε κλάση. Αξίζει να αναφερθεί ότι για τον σκοπό αυτό έχει δημιουργηθεί και το ImageNet[35], μια τεράστια βάση δεδομένων με εικόνες προς ταξινόμηση για οποιοδήποτε ενδιαφερόμενο.

- **Digit Recognition**: Η αναγνώριση ψηφίου είναι πολύ χαρακτηριστικό παράδειγμα, και παρουσιάζει σπουδαία αποτελέσματα. Εδώ στο επίπεδο εξόδου υπάρχουν 9 νευρώνες, ένας για κάθε ψηφίο (MNIST[36])

- **Εφαρμογές στην Φυσική Γλώσσα**: Τα DNNs έχουν συμβάλει καθοριστικά στην ανάπτυξη εφαρμογών σε αυτόν τον τομέα, και ειδικότερα τα LSTM [37],τα οποία έχουν την δυνατότητα να επεξεργάζονται ροή δεδομένων [38]. Τα LSTM είναι μία ειδική περίπτωση DNN με την οποία δεν θα ασχοληθούμε σε αυτήν την εργασία, αλλά παρουσιάζουν μεγάλο ενδιαφέρον.

- **Αναγνώριση Χειρονομιών Σε Βίντεο**: Και σε αυτή την περίπτωση απαιτούνται ειδικότερα είδη DNN από αυτά που έχουμε ήδη περιγράψει.

,και πολλές ακόμα.

2.3 Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks)

2.3.1 Εισαγωγή στα CNNs

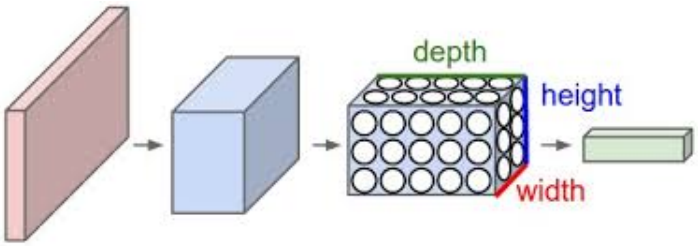
Τα Συνελικτικά Νευρωνικά Δίκτυα, ή απλά CNNs (Convolutional Neural Networks)[39], [40], [10] αποτελούν μία ειδική κατηγορία DNN και έχουν σπουδαία εφαρμόγη ιδιαίτερα όταν έχουμε σαν είσοδο δεδομένα εικόνας. Το ιδιαίτερο χαρακτηριστικό των CNN είναι ένα διαφορετικό είδος επιπέδου από τα προηγούμενα: τα συνελικτικά επίπεδα.

Αναλυτικότερα τα CNN υποθέτουν ότι δέχονται σαν είσοδο δεδομένα εικόνας. Μία εικόνα μπορεί να μοντελοποιηθεί σαν μία τρισδιάστατη (3D) βάση δεδομένων, δηλαδή με έναν πίνακα τριών διαστάσεων: το πλάτος (width), το ύψος (height) και το βάθος (depth - channels). Οι δύο πρώτες προσδιορίζουν το μέγεθος της εικόνας ενώ η τρίτη εξαρτάται από τον τύπο μοντελοποίησης της εικόνας , π.χ. RGB και άλλα. Όπως είναι εμφανές ένα DNN που επεξεργάζεται εικόνες θα έχει έναν τεράστιο αριθμό παραμέτρων. Πράγματι για το προτυπο CIFAR-10[41] που χρησιμοποιεί εικόνες διαστάσεων 32x32x3 και μόνο για το πρώτο επίπεδο των κρυφών νευρώνων θα χρειαζόμασταν 32*32*3 = 3.072 βάρη, ενώ για τα επόμενα επίπεδα ο αριθμός αυτός υπερπολλαπλασιάζεται. Σαν αποτέλεσμα η εφαρμογή των DNN σε εικόνες είναι μία πρόκληση, την οποία και επέλυσαν τα CNN.

2.3.2 Δομή και Λειτουργία των CNN

Τα CNNs αποτελούν, ουσιαστικά, μία ακολουθία επιπεδων όπου σε κάθε επίπεδο γίνεται ένας μετασχηματισμός του ”όγκου” δεδομένων. Όπως προαναφέραμε τα CNNs δέχονται κυρίως σαν είσοδο εικόνες, οι οποίες μοντελοποιούνται μέσω τρισδιάστατων δομών. Αυτές τις περιγράφουμε σαν ”όγκο” δεδομένων. Επομένως, στα CNNs υπάρχει μία σειρά από συγκεκριμένα επίπεδα που έχουν την ιδιότητα να μετασχηματίζουν τον ”όγκο”, ώστε να γίνεται όλο και πιο διαχειρίσιμος. Συγκεκριμένα, στην αρχή ο όγκος έχει συνήθως μεγάλες τις δύο πρώτες διαστάσεις και μικρότερη την τρίτη, δηλαδή μεγάλο ύψος και πλάτος και μικρό βάθος, ενώ μετά στο τελευταίο επίπεδο του CNN έχει μετασχηματιστεί σε έναν όγκο με μεγάλο βάθος και πολύ μικρότερο ύψος και πλάτος (συνήθως μοναδιαίο). Σχηματικά όλη αυτή η διαδικασία φαίνεται και στην παρακάτω εικόνα (Σχήμα 2.8).

Η φυσική ερμηνεία της λειτουργίας των CNNs που περιγράψαμε είναι η ακόλουθη: Ο αρχικός όγκος δεδομένων εισόδου είναι μία εικόνα και επομένως θα έχει τις διαστάσεις αυτής (πλάτος x ύψος x βάθος). Η παραπάνω δομή όμως δεν είναι εύκολα διαχειρίσιμη μιας και απαιτεί μεγάλο αριθμό παραμέτρων. Έτσι, στα επόμενα στάδια του CNN πραγματοποιείται από την μία, εξαγωγή χαρακτηριστικών από την εικόνα κάνοντας χρήση



Σχήμα 2.8: *Σχηματική Περιγραφή της Λειτουργίας ενός CNN*

κάποιων φίλτρων (συνέλιξη - convolution), αλλά και εν συνεχεία περικοπή των δύο πρώτων διαστάσεων (pooling). Σαν αποτέλεσμα με την συνέλιξη αυξάνεται το βάθος των όγκων όπου έχουμε πλεόν σε αυτό δεδομένα επεξεργασίας της εικόνας μέσω διαφορετικών κριτηρίων (φίλτρων), ενώ με το pooling μείωση των διαστάσεων, ώστε να μειωθούν και οι παράμετροι που πρέπει να εκπαδεύσουμε. Αυτό επαναλαβάνεται για κάποιο αριθμό που ορίζει το εκάστοτε δίκτυο, έτσι ώστε να προκύψει μία δομή με μεγάλο βάθος και πολύ μικρό συγκριτικά πλάτος και ύψος. Με άλλα λόγια συντελείται μία τρισδιάστατη μετατροπή δεδομένων, όπου από μία πλατιά αρχική δομή καταλήγουμε σε μία πολύ μακρόστενη.

Τα CNNs υλοποιούν τα παραπάνω κάνοντας χρήση πολλών ειδών επιπέδων, τα πλέον σημαντικά όμως είναι τρία: 1) **Τα Συνελικτικά Επίπεδα (Convolutional Layers)**, 2) **Τα Συγκεντρωτικά Επίπεδα (Pooling Layers)** και 3) **Τα Πλήρως Συνδεδεμένα Επίπεδα (Fully-Connected Layers)**. Για τα τελευταία μιλήσαμε εκτενώς παραπάνω, καθώς αποτελούν την κλασική περίπτωση νευρωνικών δικτύων, και στα CNNs χρησιμοποιούνται συνήθως στο τέλος των δικτύων όπου και έχει πραγματοποιηθεί ήδη η απαραίτητη προεπεξεργασία των δεδομένων από τα υπόλοιπα επίπεδα. Για αυτό και δεν θα ασχοληθούμε και εδώ με αυτά. Τα δύο πρώτα είδη επιπέδων θα τα περιγράψουμε εκτενώς παρακάτω:

Συνελικτικά Επίπεδα (Convolutional Layers)

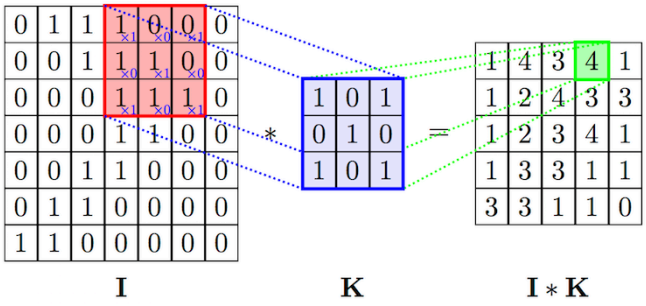
Τα Συνελικτικά Επίπεδα είναι η ουσία των CNNs, για αυτό τον λόγο και τους έδωσαν το όνομά τους. Αυτά δίνουν στα CNNs την διαφορετικότητα και τις σπουδαίες ιδιότητες τους.

Οι εκπαιδεύσιμοι παράμετροι των Συνελικτικών Επιπέδων αποτελούνται από μία σειρά φίλτρων συγκριτικά μικρών σε σχέση με την δομή της εισόδου. Η δομή που περιέχει τις παραμέτρους των φίλτρων έχει τέσσερεις (4) διαστάσεις: το πλάτος (filter ή kernel width - kw), το ύψος (filter ή kernel height - kh), τα κανάλια εισόδου (input channels - c) και, τέλος, τα κανάλια εξόδου (output channels ή number of filters- o). Τα input channels είναι τα ίδια με το βάθος της δομής εισόδου, ενώ τα κανάλια εξόδου με αυτή της δομής εξόδου. Με άλλα λόγια μετά το πέρας των Συνελικτικών Επιπέδων η δομή που θα προκύψει θα έχει βάθος όσο και τα κανάλια εξόδου. Για αυτό τον λόγο και λέμε ότι ένα Συνελικτικό Επίπεδο έχει τόσα φίλτρα όσα και τα κανάλια εξόδου.

Κατά το feed-forward πέρασμα το κάθε φίλτρο (kw x kh x c) συνελίσσεται κατά μήκος και πλάτους της δομής εισόδου. Για αυτό και οι διαστάσεις του φίλτρου θα πρέπει

να είναι αρκετά μικρότερες από τις αντίστοιχες της εισόδου, ώστε να έχουμε αρκετές συνελίξεις. Τα kw x kh πεδία στην είσοδο όπου γίνεται η συνέλιξη με το φίλτρο ονομάζονται ευαίσθητα (**receptive fields**).

Για να γίνει περισσότερο κατανοητή η πράξη της συνέλιξης θέτουμε τα εξής: Έστω έχουμε σαν είσοδο έναν πίνακα I (7x7), σαν φίλτρο έναν πίνακα K (3x3), και σαν αποτέλεσμα έναν πίνακα N = I * K(5x5) όπως φαίνεται και στο σχήμα:



Σχήμα 2.9: *Σχηματική Περιγραφή της 2D Συνέλιξης*

Για να παραχθεί το αποτέλεσμα της συνέλιξης I*K του σχήματος θα ακολουθηθεί η εξής διαδικασία:

- Θα πάρουμε το πρώτο receptive field της εισόδου, δηλαδη το I[0:2,0:2] και θα το πολλαπλασιάσουμε στοιχείο-στοιχείο με το φίλτρο. Δηλαδή:

$$I[0,0] * K[0,0], I[0,1] * K[0,1], ..., I[2,2] * K[2,2]$$

- Στην συνέχεια προσθέτουμε όλα τα παραπάνω γινόμενα και προκύπτει το πρώτο στοιχείο της συνέλιξης, δηλαδή:

$$N[0,0] = [0,0] * K[0,0] + I[0,1] * K[0,1] + ... + I[2,2] * K[2,2]$$

- Για τα υπόλοιπα στοιχεία του N θα επαναλάβουμε την παραπάνω διαδικασία μετατοπίζοντας τα receptive fields κατά ένα δεξία (δηλαδή το επόμενο θα είναι το I[0:2,1:3]). Όταν ολοκληρωθεί η δυνατή μετατόπιση κατά πλάτος τότε μετατοπίζουμε κατα ύψος (δηλαδή το receptive field θα είναι το I[1:3,0:2])

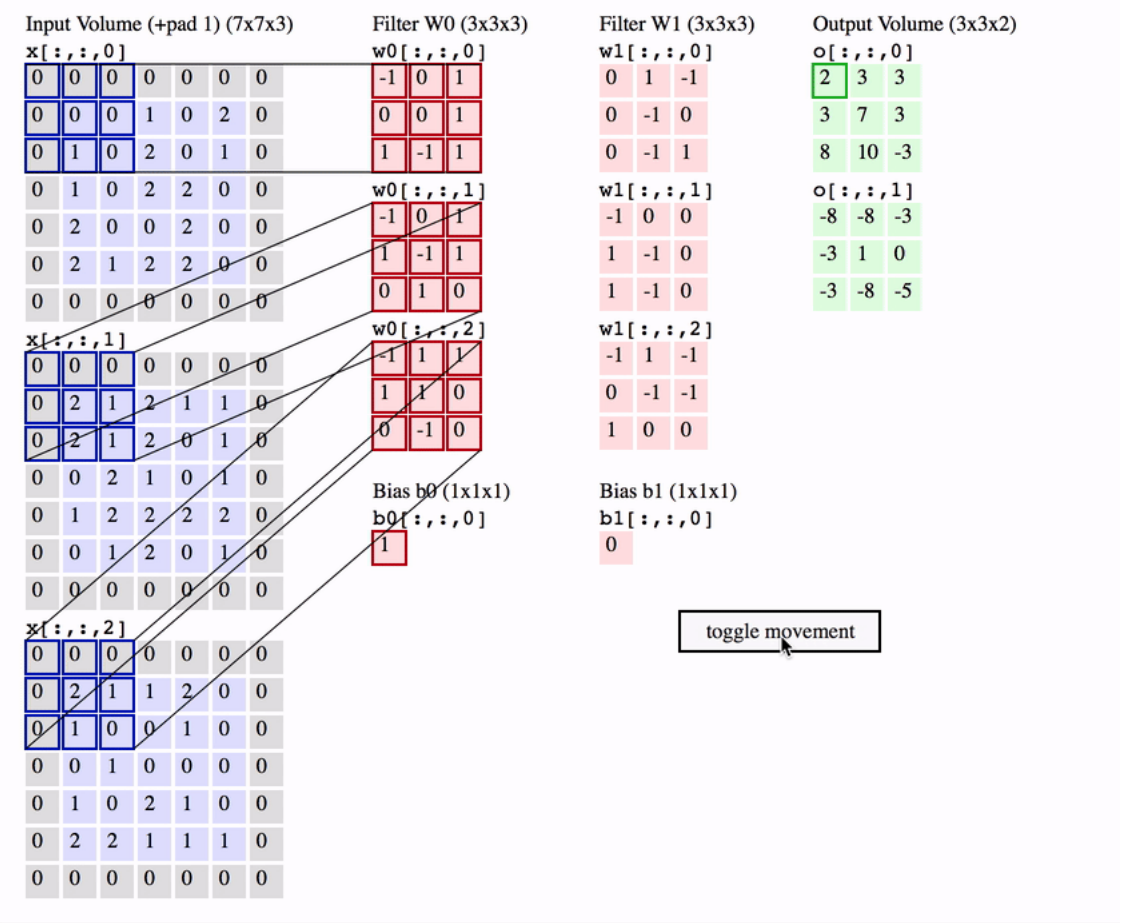
Έτσι, όπως περιγράφει και το σχήμα για να βρούμε το στοιχείο N[0,3] θα πράξουμε:

$$N[0,3] = 1*1 + 0*0 + 0*1 + 1*0 + 1*1 + 0*0 + 1*1 + 1*0 + 1*1 = 4$$

Στα συνελικτικά επίπεδα των CNNs η συνέλιξη είναι τρισδιάστατη, όμως δεν διαφέρει πολύ από την δυσδιάστατη παρά μόνο στο γεγονός ότι τώρα τα receptive fields και τα φίλτρα είναι τρισδιάστατα. Έτσι αν είχαμε βάθος (κανάλια εισόδου) ίσο με τρία (3) στο παραπάνω παράδειγμα τότε αντί για 9 γινόμενα, θα προσθέταμε

$$3 * 9 = 27$$

γινόμενα, 9 για κάθε κανάλι.**Το παρακάτω σχήμα είναι πλέον κατατοπιστικό:**



Σχήμα 2.10: *Σχηματική Περιγραφή της 3D Συνέλιξης*

Τα Συνελικτικά Επίπεδα, όμως, έχουν και κάποιες σταθερές παραμέτρους που τους καθορίζει το δίκτυο και διαφοροποιούν την συνέλιξη. Αυτές είναι:

1. **Stride** (default stride = 1): Το Stride καθορίζει το βήμα με το οποίο γίνεται η μετατόπιση κάθε φορά του receptive field. Για παράδειγμα, στα παραπάνω είχαμε stride = 1, ενώ για stride = 2 μετακινούμαστε κατά 2 θέσεις τόσο κατά πλάτος όσο και κατά ύψος, δηλαδή από το I[0:2,0:2] -> I[0:2,2:4] -> I[0:2,4:6] -> ..., κ.τ.λ. και I[0:2,0:2] -> I[2:4,0:2] -> I[4:6,0:2] -> ..., κ.τ.λ., αντίστοιχα.

2. **Dilation** (default dilation = 1): Το Dilation μεταβάλει τα receptive fields. Ουσιαστικά, διευρήνει χωρικά την εφαρμογή των φίλτρων στα δεδομένα εισόδου, ώστε να εφαρμόζονται σε μεγαλύτερα παράθυρα χωρίς ταυτόχρονα να αυξάνει το πλήθος των παραμέτρων. Στα παραπάνω είναι dilation = 1, ενώ για dilation = 2 τρο-

ποποιούμε τα φίλτρα ως εξής:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} --> \begin{bmatrix} 1 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 1 \end{bmatrix}$$

και υλοποιούμε την συνέλιξη με αυτό.

3. **Pad** (default pad = 0): Σε περιπτώσεις όπου οι χωρικές διαστάσεις των φίλτρων και αυτές της εισόδου δεν συμπίπτουν ακριβώς, ώστε να συνελισθούν τα πρώτα ομοιόμορφα στην δομή της εισόδου τότε προσθέτουμε στην είσοδο κάποιες σειρές και στήλες ώστε το πρόβλημα να ληθεί. Μαθηματικά αυτό θα εξηγηθεί και αργότερα, ενώ σχηματικά για pad = 1 προσθέτουμε στην δομή εισόδου σειρές και στήλες ωw εξής:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} --> \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 0 \\ 0 & 4 & 5 & 6 & 0 \\ 0 & 7 & 8 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Βέβαια χρησιμοποιούμε το pad και για έναν ακόμη λόγο. Όταν έχουμε μη μηδενικό Pad κάνουμε ορθότερη χρήση των δεδομένων εισόδου μιας και τα φίλτρα εφαρμόζονται καλύτερα στα ακραία στοίχεια τους.

Με βάση τις κυριότερες παραμέτρους των Συνελικτικών Επιπέδων που μόλις αναφέραμε οι διαστάσεις της δομής εξόδου υπολογίζονται ως εξής:

$$OutputWidth(oh) = (w(width) + 2 * pad - [(kw - 1) * dilation + 1])/stride + 1$$

$$OutputHeight(ow) = (h(height) + 2 * pad - [(kh - 1) * dilation + 1])/stride + 1$$

$$OutputDepth(od) = o(OutputChannels)$$

Είναι προφανές ότι για να μπορέσει να εφαρμοστεί η συνέλιξη θα πρέπει οι παραπάνω διαστάσεις να έχουν ακέραιες τιμές κάτι που δεν συμβαίνει για όλους τους συνδιασμούς παραμέτρων. Αυτός είναι και ο λόγος ύπαρξης της παραμέτρου pad. Παραδείγματος χάριν, για: height = width = 23, fheight = fwidth = 4, stride = 3, pad = 0 και dilation = 1 προκύπτει:

$$oh = (23 + 2 * 0 - [(4 - 1) * 1 + 1])/3 + 1 = 22/3$$

,και:

$$ow = (23 + 2 * 0 - [(4 - 1) * 1 + 1])/3 + 1 = 22/3$$

, ενώ αν για τις ίδιες παραμέτρους θέσουμε pad = 1 τότε:

$$oh = (23 + 2 * 1 - [(4 - 1) * 1 + 1])/3 + 1 = 8$$

,και:

$$ow = (23 + 2 * 1 - [(4 - 1) * 1 + 1])/3 + 1 = 8$$

. Επομένως, με την διόρθωση αυτή μπορούμε να υλοποιήσουμε την συνέλιξη με χρήση των συγκεκριμένων παραμέτρων.

Σύμφωνα με τα παραπάνω είναι σαφές ότι τα Συνελικτικά Επίπεδα χρησιμοποιούν λιγότερους παραμέτρους από τα Πλήρως Συνδεδεμένα. Ουσιαστικά το γεγονός ότι χρησιμοποιούνται μικρά φίλτρα τα οποία εφαρμόζονται σε όλο το μέγεθος της εικόνας (**parameter sharing**) προσφέρει **τοπικότητα (locality)**,η οποία είναι σημαντική αφού ένα χαρακτηριστικό θα εμφανιστεί σε μία περιοχή της εικόνας και έτσι δεν έχει νόημα να το αναζητούμε pixel-pixel σε όλο το μεγεθός της, αυξάνοντας ταυτόχρονα και τον αριθμό των παραμέτρων.

Συγκεντρωτικά Επίπεδα (Pooling Layers)

Τα Συγκεντρωτικά Επίπεδα έχουν ως στόχο την μείωση των χωρικών διαστάσεων της εισόδου (ύψος και πλάτος), ώστε από την μία να μειώνονται σημαντικά οι απαραίτητες παράμετροι του δικτύου, και από την άλλη να μην χάνεται σπουδαία πληροφορία από αυτή. Για να το επιτύχει αυτό χρησιμοποιεί δύο παραμέτρους: Α) **την διάσταση των τετράγωνων φίλτρων (F)** που εφαρμόζουμε στην είσοδο ώστε να την υποδειγματίσουμε και Β) **το Stride (S)**, που είναι το βήμα που εφαρμόζονται τα φίλτρα στην είσοδο. Με άλλα λόγια εφαρμόζουμε ένα φίλτρο διαστάσεων FxF με βήμα S στην είσοδο ώστε να την υποδειγματίζουμε με αναλογια

$$1 : F^2$$

.

Οι διαστάσεις της δομής εξόδου που θα προκύψει μετά την εκτέλεση του Συγκεντρωτικού Επιπέδου είναι οι ακόλουθες:

$$OutputWidth = (Width - F)/S + 1$$

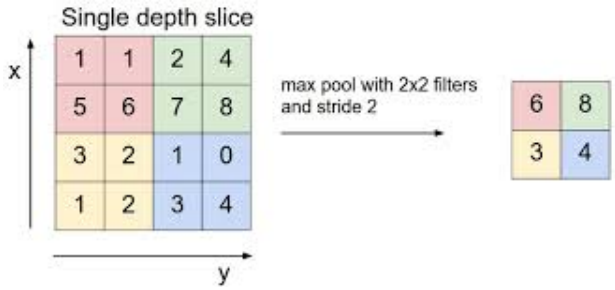
$$OutputHeight = (Height - F)/S + 1$$

$$OutputDepth = InputChannels$$

Όπως προείπαμε τα Συγκεντρωτικά Επίπεδα δεν επιφέρουν καμία αλλαγή στην διάσταση του βάθους της εισόδου. Για να επιτύχουν αυτή την υποδειγματοληψία, τα Συγκεντρωτικά Επίπεδα χρησιμοποιούν συνηθως μία από τις παρακάτω μεθόδους:

- **Max Pooling**: Η μέθοδος αυτή είναι η πλέον δημοφιλής και επιφέρει και τα καλύτερα αποτελέσματα. Σύμφωνα με αυτή, από όλα τα στοιχεία εισόδου στο φίλτρο

επιλέγεται αυτό με την μεγαλύτερη τιμή, στην δομή εξόδου, όπως περιγράφει και το σχήμα:



Σχήμα 2.11: *MAX POOLING OPERATION.*

- **Average Pooling**: Η λογική είναι όμοια με πριν, αλλά αυτήν την φορά η έξοδος παίρνει την τιμή του μέσου όρου των στοιχείων μέσα στο φίλτρο.
- **L2-Norm Pooling**: Εδώ η έξοδος είναι ίση με την Ευκλείδια Νόρμα όλων των στοιχείων μέσα στο φίλτρο.

2.3.3 Προβλήματα και Προκλήσεις κατά την Εκπαίδευση και Εφαρμογή των DNNs

Λαμβάνοντας υπόψη όλα τα παραπάνω προκύπτει ότι τα DNNs και ειδικότερα τα CNNs αποτελούν ένα πολύ χρήσιμο εργαλείο στην όραση υπολογιστών. Ανοίγει τον δρόμο για την επεξεργασία εικόνων και βίντεο αλλά και μειώνει το πλήθος των παραμέτρων σε σχέση με τα παραδοσιακά νευρωνικά δίκτυα. Ωστόσο, υπάρχουν ακόμα αρκετά προβλήματα που τα συνοδεύουν και που απαιτούν επίλυση για την πιο ευρεία χρήση τους.

Ζητήματα Μνήμης

Το σημαντικότερο ζήτημα με την χρήση των DNNs αφορά την **διαχείριση της μνήμης** [42]. Παρόλο που πλέον οι επεξεργαστές έχουν εξελιχθεί σημαντικά και ιδιαίτερα στην σειριακή εκτέλεση και οι DRAMs που τους συνοδεύουν μπορούν να αποθηκεύσουν μεγάλο αριθμό δεδομένων, τα DNNs εμφανίζουν προβλήματα. Τα προβλήματα αύτα οφείλονται στην δυσκολία διεπαφής και επικοινωνίας μνήμης - επεξεργαστών, όπου και εμφανίζεται μποτιλιάρισμα (bottleneck). Από την μία το μεγάλο πλήθος δεδομένων που απαιτούνται και από την άλλη το εύρος μεταφοράς του δίαυλου που ενώνει την μνήμη με την μονάδα επεξεργασίας (memory bandwidth) προκαλούν μεγάλη χρονική καθυστέρηση αλλά και κατανάλωση ισχύος κατά την εκτέλεση των DNNs.

Όμως γιατί τα DNNs απαιτούν τόσο μεγάλη μνήμη; Τα DNNs χρησιμοποιούν:

1. **Δεδομένα Εισόδου. (Input Data)**: Δέχονται κάποια δεδομένα εισόδων τα οποία και προφανώς πρέπει κάπου να αποθηκευτούν. Μάλιστα, σε περιπτώσεις όπου η είσοδος είναι εικόνα ή βίντεο, πρέπει να αποθηκευτούν δεδομένα τάξης MB.

2. **Δεδομένα Εξόδου. (Activations)**: Τα DNNs κατά το feed-forward πέρασμα δέχονται τα δεδομένα εισόδου και παράγουν κάποιες έξοδους. Αυτές πρέπει να αποθηκευτούν, φυσικά, είτε γιατί θα χρησιμοποιηθούν σαν είσοδο σε κάποιο επόμενο επίπεδο, είτε επειδή αποτελούν το τελικό αποτέλεσμα του DNN. Επιπροσθέτως, πρέπει να αποθηκευτούν και για τον λόγο ότι θα χρησιμοποιηθούν και στο backward πέρασμα, κατά την εκπαίδευση του δικτύου.

3. **Εκπαιδεύσιμοι Παράμετροι. (Βάρη - Weights)**: Αποτελούν την ουσία του Δικτύου, όμως η αποθήκευσή τους είναι μεγάλο εμπόδιο. Είναι απαραίτητοι τόσο στο feed-forward, όσο και στο backward πέρασμα, όπου και εκπαιδεύονται. Συνεπώς από το μεγεθός τους εξαρτάται ο χρόνος αλλά και η δυνατότητα εφαρμογής και εκπαίδευσης του δικτύου.

4. **Ποίκιλοι άλλοι Παράμετροι**: Αφορά τις σταθερές παραμέτρους που ρυθμίζουν την εκτέλεση κάθε επιπέδου του DNN και συγκριτικά με τους παραπάνω δεν επιφέρουν μεγάλη επιβάρυνση.

Για να γίνει καλύτερα κατανοητά τα μεγέθη θα χρησιμοποιήσουμε ένα παράδειγμα DNN το LesNet, το οποίο αποτελείται από 50 επίπεδα. Το LesNet, λοιπόν, χρησιμοποιεί 26 εκατομμύρια (26.000.000) εκπαιδεύσιμους παραμέτρους (βάρη) και 16 εκατομμύρια (16.000.000) τιμές για τα δεδομένα εξόδου. Έτσι σε ένα σύστημα με 32-bit floating point στοιχεια θα απαιτούνταν 168MB. Το μέγεθος αυτό μπορεί να μην τρομάζει όμως στην πραγματικότητα δεν είναι έτσι. Η εφαρμογή των DNNs σε συμβατικούς επεξεργαστες (CPUs) δεν αποδίδει χρονικά και για αυτό το λόγο χρησιμοποιούνται GPUs. Εκεί, όμως, εμφανίζεται το πρόβλημα της οργάνωσης μνήμης. Οι GPUs αποτελούνται από μεγάλο αριθμό SIMD (Simple Instructions Multiple Data) επεξεργαστικών πυρήνων, οι οποίοι υλοποιούν μικρές σε πολυπλοκότητα εργασίες αλλά με μεγάλο παραλληλισμό και οι οποίες απαιτούν μία συγκεκριμένη οργάνωση της μνήμης σε πυκνά διανύσματα. Για να χρησιμοποιηθεί μία GPU πρέπει να γίνει μεταφορά της προς επεξεργασίας μνήμης από την μνήμη του CPU σε αυτή της GPU με την μορφή των διανυσμάτων αυτών. **Η μεταφορά σε αυτά γίνεται μέσω ενός δίαυλου με Bandwidth 1024-bits συνηθως.** Η οργάνωση αυτή όμως επιβαρύνει αρκετά την μνήμη η οποία πλέον φτάνει τυπικά τα 2GB για το LesNet. Επιπροσθέτως, μία GPU δεν μπορεί να υλοποιήσει την συνέλιξη καθώς αυτή είναι αρκετά πολύπλοκη για την δομή της. Για να το αντιμετωπίσουμε αυτό μετατρέπουμε όλες τις δομές (εισόδου, εξόδου και φίλτρων-βαρών) κατάλληλα σε δυσδιάστατες ώστε να εκτελούμε την συνέλιξη σαν πολλαπλασιασμό πινάκων (matrix multiplication) στον οποίο μία GPU είναι πολύ ικανή. Όμως αυτοί οι δυσδιάστατοι τύποι που προκύπτουν περιέχουν πολλά επαναλαμβανόμενα στοιχεία ακόμα και στοιχεία που δεν χρειάζονται (κυρίως για την δομή εξόδου). Έτσι, το μέγεθος της απαιτούμενης μνήμης μπορεί ακόμα και να τριπλασιαστεί (στο LesNet γίνεται 7,5GB!!!), θέτοντας και θέμα χωρητικότητας στην μνήμη της GPU. Τέλος, αν λάβουμε υπόψη και το γεγονός ότι τα δεδομένα δεν μπορούν να μένουν συνεχώς στην μνήμη της GPU και ότι σε κάθε επίπεδο του DNN θα πρέπει να φορτώνουμε τα δεδομένα εισόδου και τα βάρη και να αποθηκεύουμε τα αντίστοιχα εξόδου και τα βάρη πάλι αν πρόκειται για backward

πέρασμα, τότε καταλαμβαίνουμε την σπουδαιότητα του ζητήματος.

Το ζήτημα της μνήμης θα μπορούσε να επιλυθεί μέσω χρήσης των εσωτερικών μνημών των επεξεργαστών. Από την μία, όμως, αυτές δεν έχουν μεγάλο μέγεθος και από την άλλη η αύξηση της χωρητικότητας θα αύξανε πάρα πολύ το κόστος κατασκευής τους. Επομένως, κάτι τέτοιο δεν θα είχε και πολύ επιτυχία.

Ζητήματα Χρόνων Εκτέλεσης

Βέβαια, εκτός από τα θέματα μνήμης, πολύ σπουδαία είναι και τα αντίστοιχα που σχετίζονται με τον χρόνο εκτέλεσης. Αναλυτικότερα, ιδιαίτερα στην περίπτωση της εκπαίδευσης ενός DNN, όπου έχουμε και feed-forward και backward περάσματα, τα οποία μάλιστα επαναλαμβάνονται για αρκετές εποχές, ώστε να προσαρμοστούν τα βάρη στο πρόβλημα, οι χρόνοι είναι πολύ μεγάλοι. Αυτό οφείλεται στους πολλούς παραμέτρους (βάρη) που πρέπει να εκπαιδευτούν, στο μεγάλο βάθος των DNN, αλλά και στα θέματα μνήμης που προαναφέραμε, τα οποία καθυστερούν σημαντικά την εκτέλεση.

Η εφαρμογή και εκπαίδευση των DNNs σε συμβατικούς CPUs κατά αρχάς δεν αποδίδει. Ακόμα και με χρήση ανεπτυγμένων CPUs η εκπαίδευση ενός DNN μπορεί να διαρκέσει μήνες, καθώς ο φόρτος εργασίας είναι μεγάλος και η παραλληλία μικρή. Αυτό καθιστά την χρήση του CPUs μη δημοφιλή επιλογή.

Από την άλλη μεριά, στις GPUs βελτιώνεται πολύ η κατάσταση. Το γεγονός ότι η συνέλιξη μετατρέπεται σε πολλαπλασιασμό πινάκων καθιστά τις GPUs εξαιρετικό εργαλείο. Πλέον οι χρόνοι εκπαίδευσης με GPUs διαρκούν ορισμένες εβδομάδες, μία πολύ σπουδαία πρόοδος. Βέβαια, υπάρχουν τα προβλήματα με την μνήμη που περιορίζουν τις δυνατότητες μίας κάρτας γραφικών, όμως η βελτίωση είναι αισθητή σε σχέση με τις CPUs. Δεν θα ήταν υπερβολή να πούμε ότι πλεόν τα νευρωνικά δίκτυα έχουν ταυτιστεί με την χρήση των GPUs.

Τα DNNs, όμως, έχουν τεράστιες δυνατότητες και δεν θα πρέπει να περιοριστούν για χρήση μόνο σε συσκευές με GPUs. Επιπρόσθετα, οι GPUs που είναι ικανές να εκτελέσουν ένα DNN είναι λίγες και μάλιστα πολύ ακριβές. Για αυτό τον λόγο τα τελευταία χρόνια αναπτύσσεται η τάση έρευνας των DNNs για χρήση σε μη συμβατικές CPUs στους λεγόμενους Coprocessors. Η συγκεκριμένη τάση παρουσιάζει εξαιρετικό ενδιαφέρον και σε αυτό το αντικείμενο κινείται και η παρούσα εργασία.

Τέλος αξίζει να αναφερθεί ότι το πιο χρονοβόρο τμήμα ενός DNN είναι τα Convolutional Layers. Πειράματα έχουν δείξει ότι η εκτέλεση του τμήματος αυτού περιλαμβάνει από 85%, έως και 98%!!! και για αυτό και επιλέξαμε να ασχοληθούμε κατά βάση με αυτό.

Μέρος II

Πρακτικό Μέρος

Κεφάλαιο 3

Περιγραφή Περιβάλλοντος - Επιμέρους Στοιχείων

Στο κεφάλαιο αυτό παρουσιάζονται όλα εκείνα τα εργαλεία τα οποία χρησιμοποιήθηκαν για την πραγματοποίηση της εργασίας. Γίνεται μια συνοπτική περίγραφη αυτών και αιτιολόγηση της συγκεκριμένης επιλογής. Αναλύονται και περιγράφονται τόσο τα προγραμματιστικά εργαλεία όσο και Βαθιά Νευρωνικά Δίκτυα πάνω στα οποία υλοποιήθηκαν οι πειραματισμοί.

3.1 Λογισμικό Εργαλείο

Για να γίνει πράξη η παρούσα εργασία χρησιμοποίηθηκε σαν λογισμικό εργαλείο το **Caffe**[22]. Το Caffe είναι ένα πολύ διάσημο framework πάνω σε Βαθιά Νευρωνικά Δίκτυα, που αναπτύχθηκε από τον **Yangqing Jia** κατά την διάρκεια του Phd του στο Πανεπιστήμιο του **UC Berkeley**. Συγκεκριμένα, κάναμε χρήση μίας έκδοσης του Caffe, η οποία αφοσιώνεται στην προσαρμογή του σε CPU της Intel. Η έκδοση αυτή είναι η **Intel-Caffe** [25].

3.1.1 Γιατί να επιλέξω το Caffe;

Ο λόγος της επιλογής του Caffe βασίστηκε στους παρακάτω άξονες [26]:

- **Συμβατότητα σε πολλές αρχιτεκτονικές.** Το Caffe προσφέρει την δυνατότητα επιλογής της αρχιτεκτονικής αλλά και πληθώρας άλλων επιλογών όπως μοντέλων και άλλων βελτιώσεων μέσω είτε αρχικής δήλωσης αυτών πριν την εγκατάστασή του, είτε μέσω ρύθμισης των κατάλληλων παραμέτρων κατά την διάρκεια της εκτέλεσης. Αξίζει, βέβαια, να σημειωθεί ότι το Caffe μπορεί να λειτουργήσει τόσο σε περιβάλλον CPU όσο και σε GPU.
- **Ανοιχτό σε προοπτικές Βελτίωσης.** Το Caffe είναι open-source framework, κάτι που σημαίνει ότι ο κώδικας του είναι διαθέσιμος στον καθένα. Αυτό γίνεται στο πλάισιο της ενίσχυσης και αναζήτησης περαιτέρων βελτιώσεων που θα κάνουν το πρόγραμμα ακόμα καλύτερο. Σαν αποτέλεσμα, μόνο την πρώτη χρονιά ύπαρξης του Caffe έγιναν πάνω από 1.000 σημαντικές αλλάγες από τρίτους.

- **Ταχύτητα.** Είναι αλήθεια ότι το Caffe είναι μία από τις πλέον γρηγορότερες εφαρμογές, καθώς κάνοντας χρήση μίας GPU γενικού σκοπού (GPGPU) μπορεί να επεξεργαστεί πάνω από 60 εκατομμύρια εικόνες την ημέρα! Δηλαδή, θα μπορούσαμε να πούμε πως στις παραπάνω συνθήκες χρειάζονται 1 msec ανά εικόνα για απλό testing και 4msec για εκπαίδευση. Βέβαια, τελευταίες βελτιώσεις αλλά και χρήση νεώτερων βιβλιοθηκών αυξάνουν περαιτέρω την ταχύτητα.

- **Forum.** Το Caffe διαθέτει ένα μεγάλο forum για την επίλυση οποιοδήποτε προβλημάτων μπορεί να εμποδίζουν έναν προγραμματιστή.

- **Εύρος Εργασιών.** Ένα ακόμη πλεονέκτημα του Caffe είναι ότι καλύπτει όλο το εύρος του τομέα των Βαθιών Νευρωνικών Δικτύων. Αρχικά, μπορεί να υλοποιήσει τόσο μηχανική Μάθηση ενός Νευρωνικού όσο και Εφαρμογή αυτών, προσφέροντας όλες τις υπάρχουσες δυνατότητες ώστε να προσαρμοστούν οι διαδικασίες αυτές στα θέλω του εκάστοτε προγραμματιστή. Επιπρόσθετα, το Caffe μπορεί να εκπαιδεύσει ή να τεστάρει τόσο νέα Νευρωνικά Δίκτυα που μπορεί ο εκάστοτε προγραμματιστής να δημιουργήσει, όσο και ήδη προυπάρχοντα Νευρωνικά. Ιδιαιτέρως για τα τελευταία προσφέρεται μία λίστα μέ όλα τα διαθέσιμα Δίκτυα που έχουν κερδίσει σε διαγωνισμούς αλλά και άλλα. Μία λίστα που διαρκώς ανανεώνεται.

Όλα τα παραπάνω αιτιολογούν την επιλογή του Caffe για την παρούσα εργασία ως ένα πολύ διάσημο, γρήγορο και αποτελεσματικό framework πάνω σε Deep Learning Δίκτυα. Όσον αφορά, δε, την έκδοση του Intel-Caffe που χρησιμοποιείται συγκεκριμένα εδώ η επιλογή βασίστηκε στο γεγονός ότι εκτελέσαμε τα πειράματα σε αρχιτεκτονικές CPU της INTEL. Σαν αποτέλεσμα, είναι σημαντικό να έχουμε την μέγιστη απόδοση πάνω σε αυτές αλλά και τα κατάλληλα εργαλεία και βιβλιοθήκες για να χειριστούμε καλύτερα τις αλλάγές που θα επιφέρουμε.

3.1.2 Περιγραφή του Caffe

Για να κατανοήσουμε σε μεγάλο βαθμό τον τρόπο με τον οποίο το Caffe λειτουργεί θα πρέπει αρχικά εξοικειωθούμε με τις βασικές δομές και παραμέτρους που αυτό χρησιμοποιεί για να εκτελέσει και να εκπαιδεύσει κάποιο μηχάνημα. Συνεπώς, για να γίνουν περισσότερο κατανοητές και οι αλλαγές που επιφέραμε σε αυτό, αλλά και για να μην υπάρξουν τυχούσες απορίες για την υλοποίση, γίνεται παρακάτω μία εκτενής περιγραφή του τρόπου με τον οποίο το Caffe αντιμετωπίζει και υλοποιεί ένα Νευρωνικό Δίκτυο, δίνοντας βάση σε τρεις βασικές παραμέτρους: α) Τα Blobs, β) Τα Επίπεδα, και γ) Το Δίκτυο [43].

Blobs

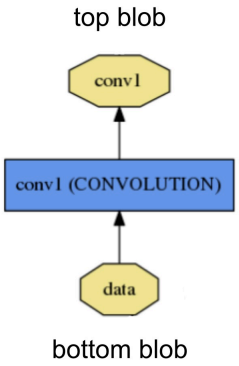
Μιας και η Βαθιά Μηχανική Μάθηση απευθύνεται κύριως σε δίκτυα που δέχονται σαν είσοδους εικόνες, το Caffe έρχεται αντιμέτωπο με δεδομένα πολύ μεγάλου μεγέθους τα οποία και καλείται να διαχειριστεί. Για τον λόγο απαιτείται μία δομή δεδομένων

που θα προσφέρει αρκετές δυνατότητες και ελευθερίες επικοινωνίας και επεξεργασίας. Αυτή η δομή δεδομένων αποτελεί το λεγόμενο Blob.

Το Blob μαθηματικά είναι ένας N-διάστατος πίνακας που είναι αποθηκευεμένος σε συνεχόμενες θέσεις μνήμης. Στην πραγματικότητα, όμως, οι διαστάσεις είναι μόνο στο μυαλό του προγραμματίστη, καθώς ένα Blob είναι ομοιάζει περισσότερο με ένα διάνυσμα που περιέχει όλα τα δεδομένα με προτεραιότητα γραμμής (row major). Οι διαστάσεις δηλώνονται κανονικά και αποθηκεύονται σε ξεχωριστά Blob αλλά έχουν σημασία μόνο για την διευκόλυνση των διαδικασιών χωρίς να ορίζουν στην πραγματικότητα κάποια δομή δεδομένων. Για να γίνει κατανοητός ο τρόπος αποθήκευσης των Blobs θα χρησιμοποιήσουμε ένα παράδειγμα ενός Blob τεσσάρων (4) διαστάσεων, όπου η πρώτη διάσταση θα είναι το πλάτος (W), δεύτερη το ύψος (H), η τρίτη τα κανάλια εισόδου (C) και η τέταρτη τα κανάλια εξόδου (O). Για αυτό το Blob η τιμή με θέση (w,h,c,o) θα αποθηκευτεί στην θέση:

$$((*C + c) * H + h) * W + w$$

Τα Blobs χρειάζονται για την αποθήκευση δεδομένων, παραμέτρους των μοντέλων, αλλά και παράγωγους που χρειάζονται στην μάθηση. Για την περίπτωση της αποθήκευσης δεδομένων οι διαστάσεις εξαρτώνται από το Επίπεδο (Layer) στο οποίο τα blobs απευθύνονται. Πάντως, στην περίπτωση της εισόδου του πρώτου επιπέδου που συναντάται στο Δίκτυο (Net) τα Blobs αντιστοιχούν σε δεδομένα εικόνων που σημαίνει ότι οι δύο πρώτες διαστάσεις αποτελούν τις χωρικές (πλάτος και ύψος) της εικόνας ενώ η τρίτη αντιστοιχεί σε αυτή των καναλιών, π.χ. για εικόνα με κωδικοποίηση RGB ή YCrCb η τρίτη διάσταση είναι τρία (3). Γενικότερα τα Blobs λειτουργούν ως απομονωτές (buffers) τα οποία έχουν σκοπό την σύνδεση των επιπέδων των δικτύου και την μετάβαση των δεδομένων εξόδου ενός επιπέδου σαν δεδομένα εισόδου στο επόμενο επίπεδο. Για αυτό τον λόγο και παρεμβάλονται μεταξύ των επιπέδων, όπως φαίνεται και στο παρακάτω σχήμα:



Σχήμα 3.1: *Παρεμβολή των Blobs μεταξύ των επιπέδων*

Εκτός βέβαια από τα δεδομένα, αποθηκεύουν και παραμέτρους των μοντέλων αλλά και των διαφόρων επιπέδων. Οι παράμετροι αυτές δηλώνονται κατά την δημιουργία ενός δικτύου από το αρχείο με κατάληξη .prototxt που περιγράφει το δίκτυο με κάθε

λεπτομέρεια. Έτσι, κάθε φορά που καλούνται να εκτελεστούν τα εκάστοτε επίπεδα δημιουργούνται και τα αντίστοιχα Blobs που περιέχουν όλες τις απαραίτητες παραμέτρους, όπως αυτές δηλώθηκαν στο .prototxt αρχείο. Χαρακτηριστικό παράδειγμα είναι αυτό των Συνελικτικών επιπέδων που εκτός πολλών αλλών απαιτούν και ένα Blob που να αποθηκεύει τα βάρη της συνέλιξης. Αυτό το Blob είναι τεσσάρων (4) διαστάσεων με κάθε διάσταση και οι αρχικές του τιμές να δηλώνονται από το .prototxt αρχείο του δικτύου. Συγκεκριμένα, οι δύο πρώτες διαστάσεις αντιστοιχούν στις χωρικές διαστάσεις (πλάτος και ύψος) του blob των βαρών, η τρίτη στα κανάλια του Blob με τα δεδομένα εισόδου (input channels) και η τέταρτη στα κανάλια του Blob με τα αντίστοιχα εξόδου (output channels).

Σε προσθήκη όλων αυτών, οι διαφορικές τίμες που απαιτούνται κατά την εκπαίδευση του Δικτύου αποθηκεύονται επίσης σε δομές Blobs. Σημειώνουμε ότι τα Blobs σαν δομές δεδομένων προσφέρουν εύκολη επικοινωνία και σύνδεση των επιπέδων όχι μόνο σε περάσματα από την αρχή προς το τέλος (forward pass) κατά την διάρκεια του testing δηλαδή, αλλά και την διάρκεια της εκπαίδευσης όπου τα περάσματα είναι από το τέλος προς την αρχή (backward pass).

Τέλος αξίζει να σημειωθεί ένα ακόμα πλεονέκτημα των Blobs, το οποίο στην παρούσα εργασία δεν θα χρησιμοποιηθεί αλλά παρόλα αυτά παραμένει σημαντικό: Τα Blobs προσφέρουν εύκολο συγχρονισμό μεταξύ CPU (host machine) και GPU (device), καθώς δεν είναι αυστηρά αρχικοποιήμενη σε κάποια από τα δύο μηχανήματα. Επίσης, καθιστούν δυνατή την μεικτή λειτουργία του Caffe σε CPU και GPU ταυτόχρονα, κάτι εξαιρετικά ενδιαφέρον για περαιτέρω πειραματισμούς.

Επίπεδα (Layers)

Τα Επίπεδα αποτελούν καθοριστικό ρόλο σε ένα Δίκτυο και είναι αυτά που ορίζουν την δομή και λειτουργία του αλλά και τις απαιτήσεις του για να λειτουργήσει. Το Caffe διαθέτει μία πληθώρα από τα επίπεδα που απαρτίζουν τα πλεόν σύνθετα και αποτελεσματικά Δίκτυα Βαθίας Μάθησης, όπως:

- Συνελικτικά Επίπεδα (Convolutional Layers)
- Πλήρως Συνδεδεμένα Επίπεδα (Inner Product Layers)
- Συγκεντρωτικά Επίπεδα (Pooling Layers)
- Επίπεδα Κανονικοποίησης Τιμών (Normalization Layers)
- Συνενωτικα Επίπεδα (Concat Layers)
- Επίπεδα Φιλτραρίσματος (Filter Layers)
- Επαναληπτικά Επίπεδα (Recurrent Layers)
- Επίπεδα Βραχυχρόνιας Μνήμης (Long-Short Term Memory Layers)

,και πολλά άλλα, ώστε να έχει ο προγραμματιστής πολλαπλές δυνατότητες.

Πολλά από τα παραπάνω επίπεδα τα έχουμε αναλύσει νωρίτερα, όμως σε αυτό το σημείο θα περιγράψουμε το πως το Caffe τα διαχειρίζεται:

Ένα επίπεδο για το Caffe αποτελεί ένα κούτι που δέχεται σαν είσοδο ένα Blob δεδομένων και παράγει ένα Blob με δεδομένα εξόδου τα οποία είτε θα είναι τα τελικά αποτελέσματα του δικτύου είτε θα χρησιμοποιηθούν σαν είσοδο για το επόμενο επίπεδο. Προγραμματιστικά, ένα επίπεδο αποτελείται από τέσσερεις (4) κύριες συναρτήσεις:

1. **SetUp συνάρτηση.** Η συνάρτηση αυτή αποσκοπεί στο να λάβει όλες τις απαραίτητες παραμέτρους από το αρχείο .prototxt, να δημιουργήσει οποιαδήποτε νέα και υλοποιήσει το κατάλληλο προγραμματιστικό περιβάλλον για την εκτέλεση του επιπέδου. Καλείται αρχικά με τον σχηματισμό και την σύνδεση των επιπέδων του Δικτύου.

2. **Reshape συνάρτηση.** Αύτη η συνάρτηση προσαρμόζει τα ήδη υπάρχοντα δεδομένα στις συνθήκες του επιπέδου. Έτσι μεταβάλει τις διαστάσεις του Blob εξόδου και αυτών που διατηρούν κάποιες ενδιάμεσες παραμέτρους ώστε να ταιριάζουν με τις διαστάσεις του Blob εισόδου. Καλείται και αυτή αρχικά μαζί με την SetUp.

3. **Forward συνάρτηση.** Αποτελεί την πλέον σημαντική συνάρτηση που εκτελεί την λειτουργία του επιπέδου. Αφορά βέβαια το forward πέρασμα που χρησιμοποιείται κατά την εφαρμογή του δικτύου και είναι αυτό που δέχεται το Blob εισόδου (bottom Blob) και παράγει τα αποτελέσματα στο Blob εξόδου (Top Blob).

4. **Backward συνάρτηση.** Τέλος, η συνάρτηση αυτή εκτελεί το backward πέρασμα του επιπέδου, το οποίο είναι απαραίτητο κατά την διάρκεια της εκπαίδευσης του, ώστε να προσαρμοστούν τα βάρη του δικτύου και να παράγεται η ελάχιστη απόκλιση από τις επιθυμητές τιμές. Σε αυτή την συνάρτηση δωσμένου του σφάλματος και του Top Blob (Blob εξόδου) παράγεται το Bottom Blob (εισόδου).

Ιδιαίτερα, για τις δύο τελευταίες συναρτήσεις κάθε επίπεδο οφείλει να δύο αντίστοιχες συναρτήσεις, μία που θα αφορά την εκτέλεση σε CPU και μία σε GPU, καθώς έτσι διατηρείται η απλότητα στον κώδικα και ρυθμίζονται καλύτερα οι ιδιαίτερες συνθήκες σε κάθε περίπτωση.

Δίκτυο (Net)

Ένα δίκτυο αποτελεί ουσιαστικά ένα σύνολο από επίπεδα και Blobs κατάλληλα συνδεδεμένα μεταξύ τους. Συνήθως η σύνδεση των δικτύων γίνεται σε ένα κατευθυνόμενο μη κυκλικό γράφο (Directed Acyclic Graph - DAG) αλλά δεν είναι λίγα τα δίκτυα που περιέχουν κυκλικούς γράφους, όπως, για παράδειγμα, τα δίκτυα με Recurrent Layers. Σκοπός του Δικτύου είναι να εκτελέσει μία εργασία, όπως η αναγνώριση ψηφίων, η ταξινόμηση εικόνων, κ.τ.λ ανάλογα τον στόχο του προγραμματιστή και για αυτό τροποποιεί την δομή του κατάλληλα. Το Δίκτυο εκπαιδεύεται και εφαρμόζεται όπως περιγράφεται και στο δεύτερο κεφάλαιο της εργασίας.

Ουσιαστικά το δίκτυο αντιπροσωπεύει μία συνάρτηση (function) αλλά και το σφάλμα (gradient) από τα επιθυμητά αποτελέσματα. Κατά την εφαρμογή του Δικτύου εκτελείται η συνάρτηση που θα υλοποιήσει την επιθυμητή εργασία και κατά την εκπαίδευση υπολογίζεται η κλίση του σφάλματος, η οποία αργότερα θα χρησιμοποιηθεί για να μάθει το μηχάνημα να εκτελεί την συγκεκριμένη διαδικασία.

Κάθε δίκτυο στο Caffe ξεκινά με ένα Επίπεδο Δεδομένων (Data Layer), στο οποίο φορτώνει από την μνήμη τα δεδομένα εισόδου και καταλήγει με ένα Επίπεδο το οποίο υλοποιεί την τελική ταξινόμηση, ή επιλογή ψηφίου, ή κάποια οποιαδήποτε άλλη εργασία έχει σκοπό το δίκτυο (Loss Function). Αρχικά, το Caffe αρχικοποιεί το Δίκτυο φτιάχνοντας όλες τις απαραίτητες συνδέσεις όπως τις διαβάζει από το αρχείο .prototxt, δηλαδή τα Επίπεδα και τα Blobs και στην συνέχεια καλεί τις συναρτήσεις SetUp και Reshape κάθε Δικτύου, ώστε να είναι έτοιμο το Δίκτυο για οποιαδήποτε εργασία. Κατά την διάρκεια τον παραπάνω γίνονται βέβαια και συνεχείς έλεγχοι ώστε να εξασφαλιστεί ότι το Δίκτυο έχει σχεδιαστεί άρτια. Τέλος, σημειώνουμε ότι ο ορισμός του Δικτύου είναι ανεξάρτητος από την αρχιτεκτονική στην οποία θα εκτελεστεί, αλλά πρέπει μέσα στο Δίκτυο να διευκρινιστεί αν πρόκειται για CPU ή GPU ώστε να κληθούν οι κατάλληλες Forward και Backward συναρτήσεις κάθε Επιπέδου.

3.2 Περιγραφή Χρησιμοποιούμενων Δικτύων

Όπως αναφέραμε και παραπάνω το Caffe μας προσφέρει την δυνατότητα να χρησιμοποιήσουμε ορισμένα δημοφιλή δίκτυα, για τα οποία εκτός από την δομή τους μας παραχωρούνται και ήδη προεκπαιδευμένοι παράμετροι (βάρη). Βέβαια, πρέπει να τονιστεί ότι το ενδιαφέρον της παρούσας εργασίας εστιάζεται κυρίως σε θέματα επίδοσης και χρόνου εκτέλεσης ενός Νευρωνικού Δικτύου, και όχι αποτελεσματικότητας και ακρίβειας που αυτό επιτυγχάνει. Επομένως, είναι ιδιαίτερα σημαντικό να έχουμε πρόσβαση στα συγκεκριμένα δημοφιλή Δίκτυα ώστε τα αποτελέσματά μας να έχουν και φυσική σημασία.

Σαν συνέπεια, σε αυτό το σημείο θα αναφερθούμε σε αυτά ακριβώς τα δίκτυα που χρησιμοποιήσαμε αναλύοντας τα χαρακτηριστικά τους αλλά και τους λόγους της επιλογής τους. Συγκεκριμένα, στην συνέχεια θα γίνει αναφορά στα εξής Βαθιά Νευρωνικά Δίκτυα[44]:

- **Alexnet** [45]
- **Googlenet** [46]
- **Caffenet** [47]
- **VGG 16-Layers** [48]
- **ResNet 50-Layers** [49]
- **SqueezeNet** [50]
- **MobileNet** [51]

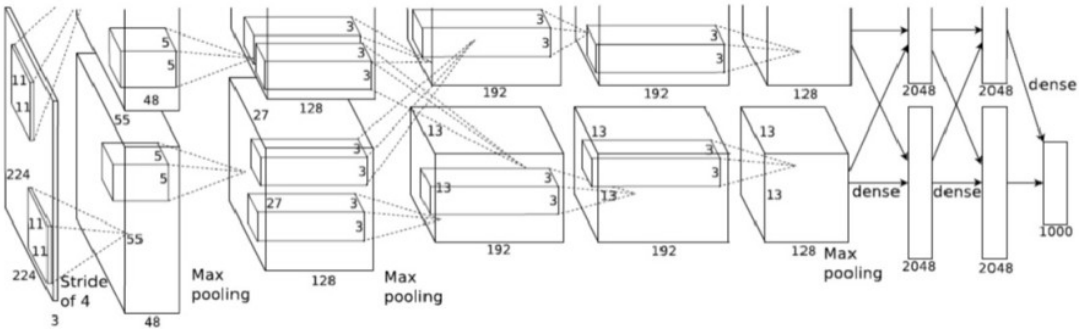
3.2.1 Alexnet

Το **AlexNet** δημιουργήθηκε το 2012 από τους Alex Krizhevsky, Sutskever, and Hinton και πήρε το όνομά του από το όνομα του πρώτου. Αποτελεί ίσως το πλέον σημαντικό Δίκτυο Βαθιάς Νευρωνικής Μάθησης από πολλές απόψεις. Αρχικά, είναι το Δίκτυο το οποίο άνοιξε το δρόμο στον συγκεκριμένο τομέα καθώς αποτελεί την πρώτη επιτυχή προσπάθεια για την δημιουργία ενός Βαθέως Νευρωνικού Δικτύου. Θεωρείται η βάση και η πηγή έμπνευσης για όλες τις επόμενες προσπάθειες. Το γεγονός αυτό γίνεται αντιληπτό και από το ότι το άρθρο με το οποίο παρουσιάστηκε το AlexNet: *"ImageNet Classification with Deep Convolutional Networks"*[45] έχει πάνω από 6.000 παραπομπές.

Από την πλευρά των επιδόσεων το AlexNet συμμετείχε και αναδείχθηκε στον διεθνή διαγωνισμό **2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)**[52], στον οποίο ομάδες από όλον τον κόσμο διαγωνίζονται με τα Βαθιά Δίκτυα (DNNs) τους πάνω σε δημοφιλείς εργασίες στον τομέα αυτό, όπως την ταξινόμηση εικόνων σε κλάσεις ή την ανίχνευση χαρακτηριστικών σε εικόνα, κ.τ.λ. Το AlexNet κέρδισε τον συγκεκριμένο διαγωνισμό επιτυγχάνοντας την ταξινόμηση μιας μεγάλης βάσης δεδομένων σε 1.000 κλάσεις περιεχομένου (πρόγραμμα Imagenet) με **Top-5 σφάλμα 15,4%**, όπου σαν Top-5 σφάλμα είναι η μη κατάταξη μίας εικόνας στις 5 πιο πιθανές κλάσεις. Για να γίνει ακόμα πιο εμφανής η επίδοση αυτή αξίζει να αναφερθεί ότι η ακριβώς επόμενη επίδοση στον διαγωνισμό αυτό είχε αρκετά μεγαλύτερο Top-5 σφάλμα και συγκεκριμένα 26,2%.

Η δομή του AlexNet

Το AlexNet αποτελείται από 8 επίπεδα, από τα οποία τα 5 πρώτα είναι Συνελικτικά (Convolutional) και τα υπόλοιπα Πλήρως Συνδεδεμένα (Fully Connected ή Inner Product Layers). Ανάμεσα σε αυτά υπάρχουν κάποια ακόμη Συγκεντρωτικά (Pooling) και Ενεργοποιητικά (Activation Layers) Επίπεδα. Αναλυτικότερα,η δομή αυτή είναι εμφανής και στο *Σχήμα 4.2*.



Σχήμα 3.2: *Alexnet. Η δομή του Δικτύου*

Το πρώτο επίπεδο του Δικτύου είναι ένα Συνελικτικό Επίπεδο, το οποιό συνελίσσει δεδομένα εικόνων, διαστάσεων 224x224x3 με την βοήθεια 11x11x3x48 φίλτρου ώστε να παράγει μία Blob διαστάσεων 55x55x96. Το Επίπεδο αυτό είναι η πρώτη φάση της

επεξεργασίας της εικόνας και για αυτό τον λόγο έχει ως σκοπό από την μία να λάβει όσον το δυνατό περισσότερη πληροφορία από τα δεδομένα εισόδου αλλά και ταυτόχρονα να μειώσει αρκετά το μέγεθος των δεδομένων, χρησιμοποιώντας βήμα (stride) 4 στην συνέλιξη ώστε αυτά να είναι ευκολότερα διαχειρίσιμα από τα επόμενα Επίπεδα.

Στην συνέχεια, έπεται ένα δεύτερο Συνελικτικό Επίπεδο, που δέχεται σαν είσοδο ένα Blob διαστάσεων 55x55x96, το συνελίσσει μέσω ενός φίλτρου διαστάσεων 5x5x48x256 για να παράγει ένα 27x27x256 Blob εξόδου. Αν παρατηρήσουμε προσεκτικά τις παραπάνω διαστάσεις θα συνειδητοποιήσουμε ότι αυτές απομακρύνονται από το κλασικό συνδιασμό αυτών καθώς παρατηρούμε ότι το blob εισόδου και αυτό του φίλτρου δεν έχουν την ίδια τρίτη διάσταση, αυτή των Καναλιών Εισόδου (Input Channels). Η εξήγηση αυτού του φαινομένου είναι εμφανής και στο Σχήμα 4.2. Συγκεκριμένα, είναι εμφανές ότι μετά το πρώτο Επίπεδο τα Blobs χωρίζονται σε δύο όπου το κάθε ένα έχει την μισή τρίτη διάσταση από την ολική. Θα λέγαμε ότι η διαδικασία μοιράζεται σε δύο, οι οποίες υλοποιούνται παράλληλα μέχρι και το τελευταίο Συνελικτικό Επίπεδο όπου ενώνονται και συγκροτούν μία ενιαία που εκτελεί τα Πλήρως Συνδεδεμένα Δίκτυα. Η λογική αυτή προήλθε για πρακτικούς και μόνο σκοπούς καθώς παρατήρησαν ότι με αυτό τον τρόπο η εκτέλεση παρουσιάζει μεγάλη επιτάχυνση στην παρούσα αρχιτεκτονική που χρησιμοποιούνταν για το AlexNet (2 GTX 580 GPUs) με αμελητέα απώλεια ακρίβειας. Σαν αποτέλεσμα δημιουργήθηκε μία νέα Συνελικτική παράμετρος η οποία εκφράζει αυτήν ακριβώς την λογική και επαχρησιμοποιήθηκε σε πολλά νεότερα Δίκτυα, το **group** . Το group ορίζει τον αριθμό των διαδικασιών που θα εκτελεστούν παράλληλα χωρίζοντας τα Blob εισόδου, εξόδου και φίλτρου κατάλληλα. Έτσι, στην περίπτωση του AlexNet είναι κατανοητό ότι ισχύει:

$$group = 2$$

Μετά το δεύτερο Συνελικτικό Επίπεδο ακολουθούν άλλα 3(τρία) Συνελικτικά ,τα οποία συνελίσσουν κατάλληλα τα δεδομένα οδηγόντας τα στα Τρία τελευταία Πλήρως Συνδεδεμένα Επίπεδα, όπου πραγματοποιείται η ταξινόμηση των εικόνων στις κλάσεις. Συγκεκριμένα, στα 2 προτελευταία Επίπεδα τα Blobs δεδομένων μετατρέπονται μέσω αυτών σε διανύσματα (vectors) διαστάσεων 1x2048 για να ακολουθήσει το τελευταίο Πλήρως Συνδεδεμένο Επίπεδο το οποίο παράγει ένα διάνυσμα 1x1000 που αντιπροσωπεύει την πιθανότητα της αρχικής εικόνας να ανήκει σε κάθε μία από τις 1000 κλάσεις. Προφανώς ”νικήτρια” κλάση είναι αυτή με την μεγαλύτερη πιθανότητα στο τελικώς παραγόμενο διάνυσμα.

Σημασία του AlexNet

Όπως αναφέραμε και παραπάνω το AlexNet είναι ένα πολύ σημαντικό Δίκτυο για τον κλάδο της Βαθιάς Μηχανικής Μάθησης. Αυτό δεν οφείλεται μόνο στο γεγονός ότι ήταν το πρώτο Βαθύ Νευρωνικό Δίκτυο αλλά και σε μια σειρά άλλων παραγόντων. Αρχικά, εφαρμόστηκε στο **ImageNet**, ένα Δίκτυο από 15 εκατομμύρια (15.000.000) εικόνες από πάνω από 22.000 κατηγορίες. Επομένως, η εργασία την οποία προσπάθησε

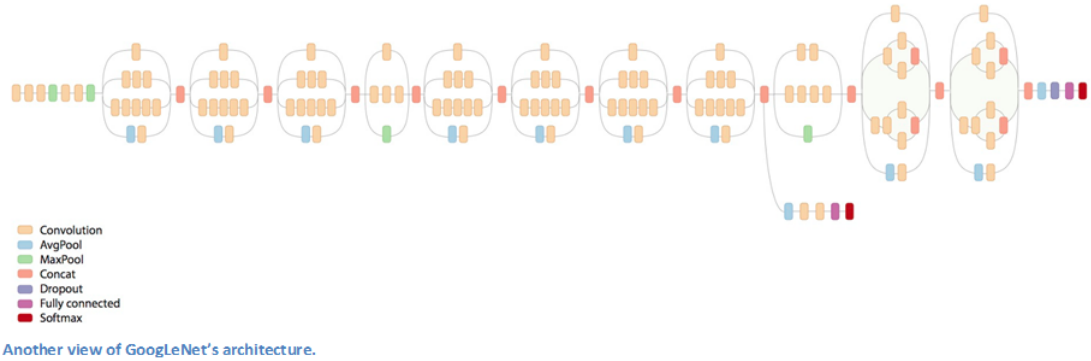
να υλοποιήσει είναι εξαιρετικά δύσκολη και το ότι κατάφερε να την φέρει εις πέρας για πρώτη φορά τόσο αποτελεσματικά όσο και γρήγορα, καθώς χρειάστηκε μόνο 5 με 6 ημέρες εκπαίδευσης, αποτελεί σημείο αναφοράς. Επιπρόσθετα, για να το επιτύχουν αυτό έκαναν χρήση ορισμένων τεχνικών που μέχρι και σήμερα θεωρούνται πολύ αποτελεσματικές και χρησιμοποιούνται ευρέως, υποδεικνύοντας έτσι τα οφέλη και τις δυνατότητες που προσφέρει η Βαθιά Μηχανική Μάθηση. Για παράδειγμα, για να επιταχύνουν την εκπαίδευση χρησιμοποίησαν συναρτήσεις ενεργοποίησης ReLU, στα Πλήρως Συνδεδεμένα Επίπεδα, οι οποίες είναι πολύ πιο γρήγορες από την συνάρτηση υπερβολικής εφαπτόμενης (tanh), ενώ, ταυτόχρονα αύξησαν τα δεδομένα εκπαίδευσης εκμεταλλεύοντας τεχνικές βασισμένες σε περιστροφή εικόνων, οριζόντιους καθρεπτισμούς, κ.τ.λ..

3.2.2 GoogleNet

Το **GoogleNet** δημιούργηθηκε το 2014 από την Google. Όπως και το AlexNet, έτσι και αυτό κέρδισε στον διαγωνισμό *ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)* [53], επιτυγχάνοντας μάλιστα πολύ υψηλά ποσοστά. Συγκεκριμένα, εμφάνιζοντας Top-5 σφάλμα ύψους 6,7% έγινε ένα από τα πιο αποτελεσματικά Δίκτυα μέχρι και σήμερα, ενώ ταυτόχρονα χρησιμοποιεί μικρό σκετικά μέγεθος μνήμης και προκαλεί μικρή κατανάλωση ενέργειας. Προκείται για ένα πολύ σημαντικό Δίκτυο με πρωτότυπη δομή, η οποία παρουσιάζεται παρακάτω.

Η δομή του GoogleNet

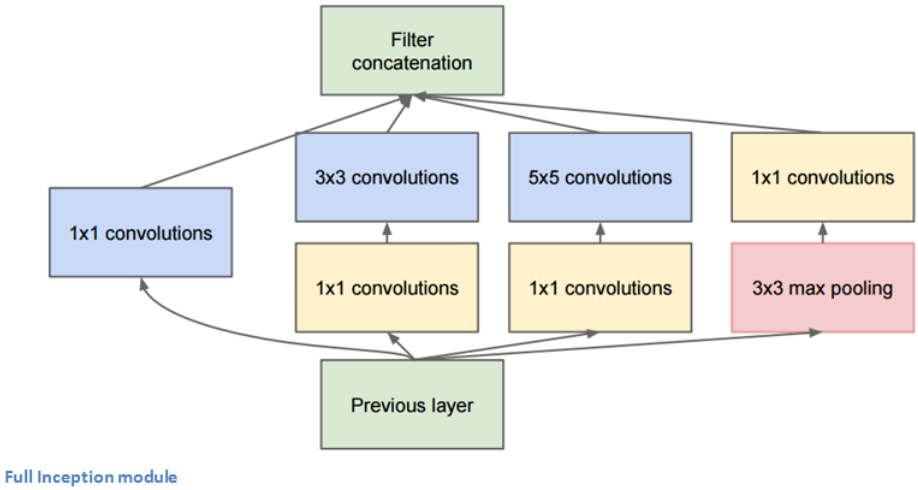
Το GoogleNet αποτελεί ένα από τα πιο μεγάλα σε μέγεθος Δίκτυα. Αποτελείται από 22 Επίπεδα από τα οποία μόνο το ένα είναι Πλήρως Συνδεδεμένο. Όπως είναι εμφανές και στο Σχήμα 4.3,πρόκειται για ένα Δίκτυο πολύ μεγάλου μεγέθους, στο οποίο όμως είναι διακριτά ορισμένα περιοδικά μοτίβα καθώς και το γεγονός ότι το Δίκτυο δεν ακολουθεί σειριακή ροή επεξεργασίας των δεδομένων. Οι δύο παραπάνω παρατηρήσεις αποτελούν και το μυστικό της επιτυχίας του GoogLeNet.



Σχήμα 3.3: *GoogLenet. Η δομή του Δικτύου*

Αρχικά, το επαναλαμβανόμενο πρότυπο σύνδεσης των Επιπέδων που εμφανίζεται στο Σχήμα 4.3 αποτελεί το Inception Module, όπως ονομάστηκε, και είναι παρουσιά-

ζεται σαφέστερα παρακάτω:



Σχήμα 3.4: *Inception Module: Η βασική Δομική Μονάδα του GoogLeNet.*

Η βασική ιδέα πίσω από το **Inception** είναι η εξής: Σε ένα οποιοδήποτε σημείο ενός Συνελικτικού Δικτύου (CNN) ερχόμαστε με το δίλημμα αν θα πρέπει να εμφαρμόσουμε Συνέλιξη με φίλτρο διαστάσεων 1x1, 3x3 ή 5x5 ή αν θα προσπαθήσουμε να μειώσουμε το μέγεθος των δεδομένων για την συνέχεια. Σημειώνουμε ότι το μέγεθος του φίλτρου στην Συνέλιξη υποδηλώνει το είδος της πληροφορίας που θέλουμε να αντλήσουμε κάθε φορά. Έτσι με μία 1x1 Συνέλιξη λαμβάνουμε πληροφορίες που απευθύνονται σε λεπτομέρειες τις εικόνας, όπως μία γωνία αυτής, ενώ με ένα μεγαλύτερο μέγεθος φίλτρου, π.χ. 5x5, εκμεταλλευόμαστε τοπικά κομμάτια της εικόνας. Επομένως, κάθε φορά πρέπει να αποφασίζουμε αν θα πάρουμε πιο γενικές πληροφορίες από την εικόνα εφαρμόζοντας ένα φίλτρο μεγαλύτερων διαστάσεων ή αν θα πρέπει να επικεντρώσουμε σε πιο λεπτομερή χαρακτηριστικά. Αυτό το δίλημμα παύει να υπάρχει στο GoogLeNet χάριν στο Inception. Όπως φαίνεται και στο Σχήμα 4.4 και υποδηλώνουν και τα βέλη εκεί το Inception δέχεται σαν είσοδο το αποτέλεσμα του προηγούμενου Επιπέδου. Στην συνέχεια, εκτελεί παράλληλα και τις τέσσερεις επιλογές του παραπάνω διλήμματος. Έτσι, σε κάθε σημείο διαθέτουμε πληροφορία τόσο για λεπτομερή χαρακτηριστικά όσο και για πιο γενικά, ενώ ταυτόχρονα πραγματοποιούμε και μείωση των διαστάσεων των δεδομένων. Στο τέλος οι τέσσερεις παράλληλες ροές επεξεργασίας συνενωννόνται (**Filter concatenation**) σε μία δομή για να παραχθεί το αποτέλεσμα του Inception. Σαν αποτέλεσμα η παραγόμενη δομή δεδομένων θα είναι τρισδιάστατη και θα διαχωρίζει τα αποτελέσματα της παράλληλης διαδικασίας στην τρίτη διάσταση, αυτήν των καναλιών, δηλαδή στα πρώτα κανάλια θα αποθηκεύονται τα αποτελέσματα της 1x1 Συνέλιξης, στα επόμενα της 3x3 κ.ο.κ.. Τέλος είναι κατανοητό ότι, καθώς εφαρμόζονται τρεις συνελίξεις ταυτόχρονα και τα αποτελέσματά τους αποθηκεύονται σε μία δομή, υπάρχει μεγάλο ζήτημα με το μεγάλο και μη διαχειρίσιμο μέγεθος της τελευταίας. Για την αντιμετώπιση αυτού, πραγματοποιούνται 1x1 συνελίξης πριν τις 3x3 και 5x5, ώστε να μειωθεί το μέγεθος των καναλιών των δεδομένων.

Επανερχόμενοι πάλι στο Σχήμα 4.3 παρατηρούμε ότι το GoogLeNet διαθέτει κάποια

αρχικά Συνελικτικά επίπεδα τα οποία υλοποιούν μία αρχική εξαγωγή χαρακτηριστικών της εικόνας και στην συνέχεια ακολουθούν 9 Inception Επίπεδα τα οποία ακολουθούν με μικρές αποκλίσεις το πρότυπο που αναλύσαμε. Τέλος, υπάρχει ένα Πλήρως Συνδεδεμένο Επίπεδο το οποίο και ταξινομέι τις εικόνες του ImageNet στις ζητούμενες κλάσεις.

Σημασία του GoogLeNet

Το GoogLeNet είναι μαζί με το AlexNet ένα ακόμα Βαθύ Νευρωνικό Δίκτυο που αποτελεί σημείο αναφοράς για την Μηχανική Μάθηση. Η σημασία και η συμβολή του στον κλάδο αυτό είναι σπουδαία και πολύπλευρη, αλλά θα μπορούσαμε να την περιορίσουμε στα εξής σημεία:

- Έχοντας ορίσει ένα Βαθύ Νευρωνικό Δίκτυο σαν ένα Νευρωνικό με περισσότερα από 2 Νευρωνικά Επίπεδα, κατανοούμε ότι το GoogLeNet οδηγεί αυτόν τον ορισμό σε ένα άλλο επίπεδο. Χρησιμοποιώντας πάνω από 100 Επίπεδα συνολικά, επιλύει κάθε απορία σχετικά με το πόσο Βαθιά μπορούν να γίνουν τα Δίκτυα αλλά και ποια θα είναι η αποτελεσματικότητα τους. Το GoogLeNet ήταν το πρώτο Νευρωνικό Δίκτυο που χρησιμοποίησε τέτοιον αριθμό επιπέδων και έδειξε ότι μπορούν να παραχθούν μεγάλα και ταυτόχρονα αποτελεσματικά Δίκτυα.

- Παρόλο το μεγάλο μέγεθός Επιπέδων που διαθέτει, διατηρεί μικρό αριθμό παραμέτρων! Συγκεκριμένα, διαθέτει 12 φορές λιγότερες παραμέτρους από το πολύ μικρότερο σε μέγεθος AlexNet, κάτι το οποίο οφείλεται στο γεγονός ότι διαθέτει ένα μόνο Πλήρως Συνδεδεμένο Επίπεδο. Ταυτόχρονα, απαιτεί και τον ίδιο χρόνο εκπαίδευσης (1 εβδομάδα) συγκριτικά με το AlexNet.

- Το GoogLeNet εισήγαγε την ιδέα της μη σειριακής επεξεργασίας των δεδομένων καθώς μέσω του Inception διοχέτευε τα δεδομένα εισόδου σε τέσσερεις παράλληλες ροές επεξεργασίας, όπως περιγράψαμε και προηγουμένως.

- Ακόμη είναι το πρώτο που εφάρμοσε συνεχόμενη διοχέτευση δεδομένων από Συνελικτικά επίπεδα σε Συγκεντρωτικά (Pooling) χωρίς κάποιο βοηθητικό επίπεδο ανάμεσά τους.

- Τέλος, η αποτελεσματικότητα και η ακρίβεια που επιτύγχανε αποτελεί σημείο αναφοράς και μέχρι και σήμερα δεν υπάρχουν πολλά Δίκτυα που να την προσεγγίζουν, ενώ το ίδιο δεν αποτελεί ένα ολοκληρωμένο πρόγραμμα μιας και συνεχώς παράγονται βελτιωμένες εκδόσεις του Inception (Version 6 ή 7) που προσφέρουν ακόμα καλύτερα χαρακτηριστικά.

3.2.3 CaffeNet

Το CaffeNet δεν αποτελεί ένα τόσο πολύ σημαντικό Δίκτυο συγκριτικά με τα παραπάνω. Ουσιαστικά πρόκειται για ένα αντίγραφο του AlexNet το οποίο δημιουργήθηκε από την ομάδα των δημιουργών του Caffe και διαφέρει ελάχιστα από το πρώτο.

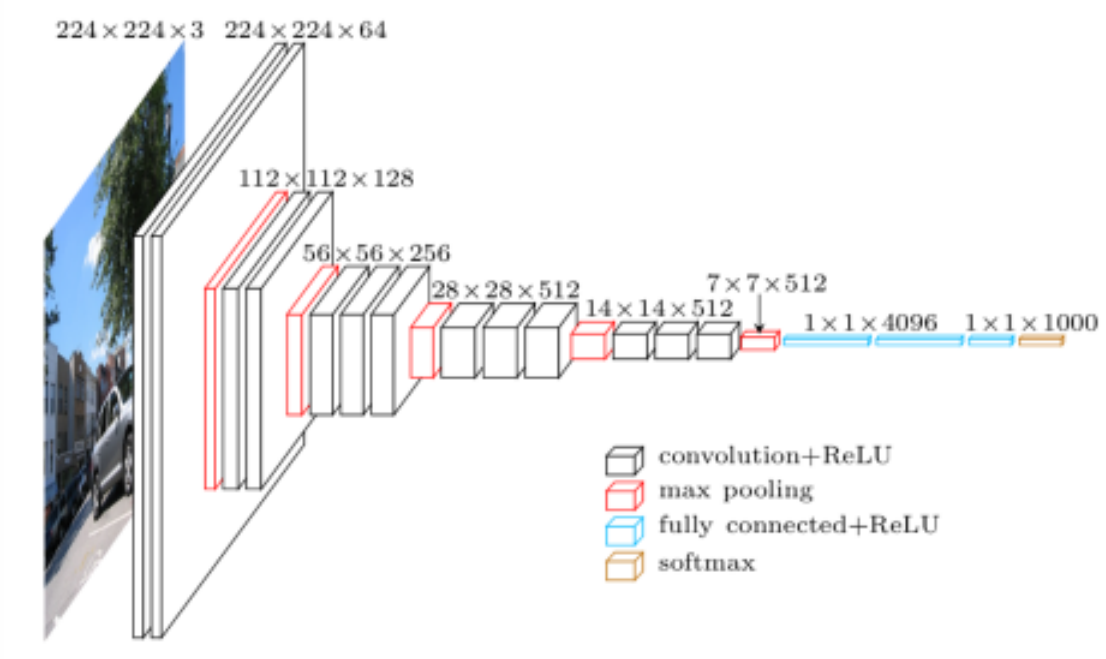
Οι ελάχιστες διαφορές τους επικεντρώνονται στην αλλαγή της σειράς εκτέλεσης κάποιων Συγκεντρωτικών (Pooling) και Επιπέδων Κανονικοποίησης (Normalization), με το CaffeNet να εκτελεί πρώτα το πρώτο και ύστερα να κανονικοποιεί τα δεδομένα, κάτι που προσφέρει ένα μικρό κέρδος μνήμης και υπολογισμών. Ωστόσο, το γεγονός ότι προσφέρεται από το Caffe σαν εναλλακτική επιλογή δεν μας άφησε αδιάφορους.

3.2.4 VGG - 16 Επιπέδων

Το VGG είναι ένα Δίκτυο, το οποίο δημιουργήθηκε ταυτόχρονα με το GoogLeNet το 2014 από τους Simonyan and Zisserman. Συμμετείχε και αυτό στον 2014 *ILSVRC (ImageNet Large-Scale Visual Recognition Challenge)*[53] και παρότι δεν κέρδισε σε αυτό αποτελεί ένα σπουδαίο Βαθύ Νευρωνικό Δίκτυο από πολλές πλευρές. Το VGG υπάρχει σε έξι παραλλαγές, οι οποίες διαφέρουν κυρίως στον αριθμό των κύριων Επιπέδων (Συνελικτικών και Πλήρως Συνδεδεμένων). Στην παρούσα εργασία έγινε επιλογή του VGG των 16 επιπέδων καθώς αποτελεί μάλλον την πλέον αποδοτική.

Η Δομή του VGG-16 Επιπέδων

Το VGG είναι ένα Δίκτυο το οποίο συνδίαζε από την μία το μεγάλο Βάθος για την εποχή του και από την άλλη την **απλότητα**. Όπως αποκαλύπτει και το όνομα του αποτελείται από 16 Επίπεδα: 13 Συνελικτικά και 3 Πλήρως Συνδεδεμένα. Τα Συνελικτικά συνελίσουν συνεχόμενα τα δεδομένα μετασχηματίζοντας τις δομές τους από την αρχική των διαστάσεων 224x224x3 σε μία αντίστοιχη διαστάσεων 14x14x512, ενώ τα 3 τελευταία Πλήρως Συνδεδεμένα Επίπεδα υλοποιούν την ταξινόμηση και παράγουν το επιθυμητό 1x1000 διάνυσμα πιθανοτήτων, σχηματίζοντας, έτσι, ένα αρκετά Βαθύ Νευρωνικό Δίκτυο.



Σχήμα 3.5: *VGG 16 Layers: Η δομή του Δικτύου*

Εκτός, όμως από το βάθος, αυτό που χαρακτηρίζει κυρίως το VGG είναι η απλότητα. Η απλότητα στην δομή του εκφράζεται από το γεγονός ότι κάθε Συνελικτικό Επίπεδο διαθέτει κοινές παραμέτρους με τα υπόλοιπα. Συγκεκριμένα, όλα τα Συνελικτικά Επίπεδα κάνουν χρήση 3x3 φίλτρα και συνελίσσουν με βήμα (stride) και pad ίσο με 1. Εκτός αυτού στο Σχήμα 4.5 είναι εμφανές οτι στην δομή του Δικτύου επαναλαμβάνονται συνεχόμενα όμοια σε παραμέτρους Συνελικτικά Επίπεδα. Αυτό οφείλεται στο ότι οι δημιουργοί του δικτύου παρατήρησαν ότι ο συνδιασμός δύο συνεχόμενων Συνελικτικών Επιπέδων με 3x3 φίλτρα αντιστοιχεί σε ένα Συνελικτικό Επίπεδο με 5x5 φίλτρο, ενώ αν συνδιαστούν τρία όμοια 3x3 Επίπεδα τότε έχουμε παρόμοια αποτελέσματα με την χρήση ενός 7x7 επιπέδου. Σαν συνέπεια, κατάφερανε να προσομοιώσουν 5x5 και 7x7 Συνελίξεις διατηρώντας τα υπολογιστικά οφέλη των μικρότερων φίλτρων.

Σημασία VGG

Παρόλο το γεγονός ότι δεν ήταν ποτέ το καλύτερο Δίκτυο από άποψη ακρίβειας στην εποχή του, το VGG αποτελεί ένα πολύ σημαντικό Δίκτυο, γιατί ανέπτυξε πολλές ιδέες και αντιλήψεις πάνω στην Βαθιά Μηχανική Μάθηση. Αρχικά, εξέφραζε την ιδέα της **Ιεραρχικής Αντιπροσώπευσης** της δομής ενός Βαθέως Νευρωνικού Δικτύου, υπό την έννοια ότι η εξαγωγή των χαρακτηριστικών που υλοποιείται από τα Συνελικτικά Επίπεδα ξεκινά από τα πιο λεπτομερή χαρακτηριστικά και με την πορεία ασχολείται με χαρακτηριστικά που αφορούν μεγαλύτερα και γενικότερα τμήματα της εικόνας. Η ιδέα αυτή είναι θεμελειώδης για τον τομέα της Βαθιάς Μηχανικής Μάθησης. Σε προσθήκη, αυτού ανέπτυξε την ιδέα ότι η απλότητα οδηγεί σε εξαιρετικά αποτελέσματα ακόμα και όταν πρόκειται για ένα βαθύ Νευρωνικό Δίκτυο με όλη την περιπλοκότητα που μπορεί να το συνοδεύει.

Επιπρόσθετα, η αποτελεσματικότητα του Δικτύου είναι αξιοσημείωτη καθώς στον 2014 ILSVRC εμφάνισε πολύ μεγάλη ακρίβεια, πολύ κοντά σε αυτή του GoogLeNet και συγκεκριμένα πέτυχε 7,3% Top-5 σφάλμα. Ταυτόχρονα, το γεγονός ότι μπορεί αποτελεσματικά να χρησιμοποιηθεί και σε διαφορετικές εργασίες από την ταξινόμηση εικόνων, προσδίδει ιδιαίτερη σημασία στο VGG. Βέβαια, αξίζει να τονισθεί ότι το VGG παρουσίαζει ένα μεγάλο ελάττωμα: Διαθέτει πολύ μεγάλο αριθμό παραμέτρων, κυρίως λόγω των τρίων Πλήρως Συνδεδεμένων Επιπέδων, κάτι που το καθιστά αρκετά ”δυσκίνητο” και αργό στην εκπαίδευση (χρείαστηκε 3 με 4 εβδομάδες πάνω σε πανίσχυρες GPUs) αλλά και πολύ ενδιαφέρουσα πρόκληση για την παρούσα εργασία. Τέλος, είναι πολύ σημαντικό το γεγονός για την παρούσα εργασία ότι οι δημιουργοί έκαναν χρήση του Caffe για την ανάπτυξη του Δικτύου.

3.2.5 Residual Net - ResNet

Το Residual Net δημιουργήθηκε από την Microsoft και ήταν ο νικητής του **2015 ILSVRC**[54] επιτυγχάνοντας την καλύτερη μέχρι στιγμής ακρίβεια, έχοντας **Top-5 σφάλμα 3.6%** !! Γνωρίζοντας ότι η διακριτική ικανότητα του ανθρώπου κυμαίνεται μεταξύ 5 και 10 %, κατανούμε ότι πρόκειται για μία επίδοση που ξεπερνά ακόμα και την ανθρώπινη δυνατότητα. Για να το επιτύχει αυτό ξεπέρασε κατά πολύ την αντίληψη που υπήρχε

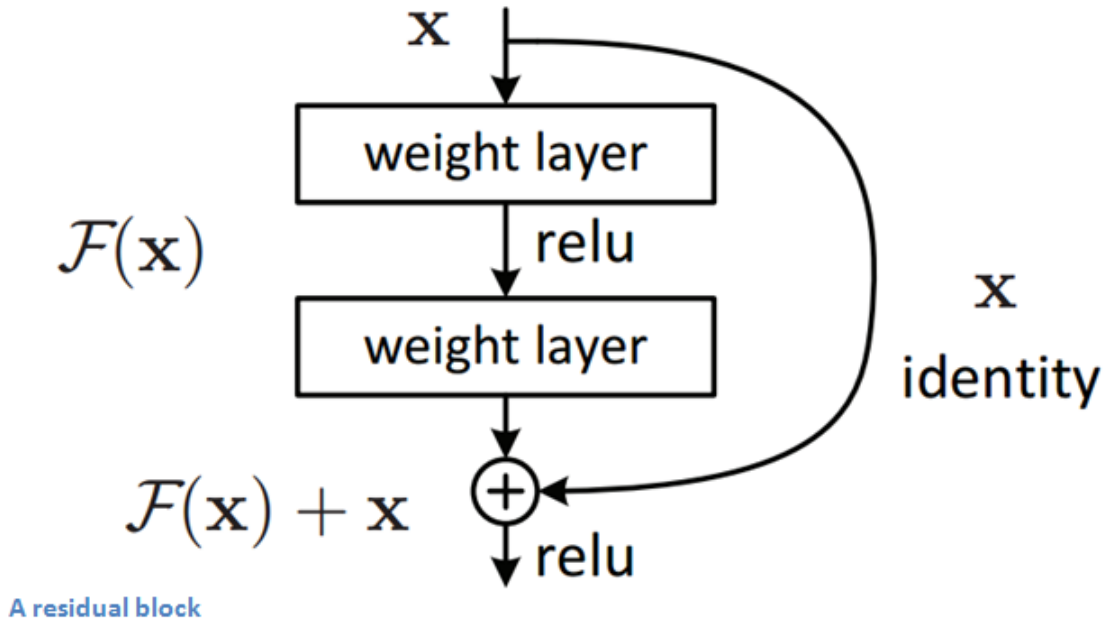
μέχρι τότε για το Βάθος των Νευρωνικών Δικτύων. Συγκεκριμένα, μία από τις τρεις εκδόσεις του ResNet διαθέτει **152 Παραμετρικά Επίπεδα**!!! Το ResNet αποτελεί σήμερα το κορυφαίο Βαθύ Νευρωνικό Δίκτυο μέχρι και σήμερα.

Η δομή του Δικτύου

Η δομή του Residual Net βασίστηκε στην δομική μονάδα που ονομάζεται Residual Block και είναι εμφανές στο Σχήμα 4.6. Σε αυτό παρατηρούμε ότι ξεφεύγουμε από την συμβατική ροή επεξεργασίας των Δικτύων στην οποία το κάθε επίπεδο λαμβάνει μία x είσοδο και παράγει μία έξοδο F(x), με την F() να εξαρτάται από το είδος του Επιπέδου. Εδώ, το κάθε Residual Block παράγει μία έξοδο:

$$H(x) = F(x) + x$$

Έτσι, με αυτόν τον τρόπο αντί μετά από κάθε επίπεδο να παράγουμε μία αντιπροσώπευση των δεδομένων η οποία δεν συνδέεται καθόλου με την προηγούμενη, τώρα κάθε Block παράγει μία μεταβολή της εισόδου, η οποία προστίθεται στην πρώτη για να παραχθεί τροποποιήμενη έκδοση της εισόδου. Ουσιαστικά, πρόκειται για μία εξέλιξη των δεδομένων καθώς περνούν μέσα από τα Blocks για νακαταλήξουν στο τελικό Πλήρως Συνδεδεμένο Επίπεδο. Σύμφωνα με τους δημιουργούς του Δικτύου "είναι ευκόλοτερη η βελτιστοποίηση του συνδιασμού των επιπέδων με αυτό τον τρόπο παρά με τον συμβατικό τρόπο".



Σχήμα 3.6: *ResNet: Residual Block*

Αναλυτικότερα, το Residual Net έχει τρεις εκδόσεις, οι οποίες διαφέρουν κυρίως στον αριθμό των επιπέδων τους: α) Το **ResNet των 50 επιπέδων**, Β) Το **ResNet των 101**, και Γ) Το **ResNet των 152 Επιπέδων**. Στην παρούσα εργασία γίνεται χρήση του πρώτου, δηλαδή του ResNet των 50 Επιπέδων, μιας και παρουσιάζει ακρίβεια που προ-

σεγγίζει αυτή των υπολοίπων, ενώ είναι πιο κατάλληλη για εφαρμογή σε επεξεργαστές λόγω του μικρότερου μεγέθους της.

Η δομή του ResNet των 50 Επιπέδων που χρησιμοποιούμε ξεκινά με ένα εισαγωγικό Συνελικτικό και ένα Συγκεντρωτικό Επίπεδο, τα οποία δέχονται σαν είσοδο μία εικόνα διαστάσεων 224x224x3 και παράγουν 55x55x64, μειώνοντας σε μεγάλο βαθμό τις χωρικές διαστάσεις των Blobs, κάτι αρκετά ασυνήθιστο καθώς χάνεται αρκετή πληροφορία, αλλά αναγκαίο για να είναι διαχειρίσιμο παραμετρικά το δίκτυο. Στην συνέχεια, έπονται 16 Residual Blocks τα οποία τροποποιούν κατάλληλα τα δεδομένα, όπως εξηγήσαμε παραπάνω και παράγουν μία δομή 7x7x2048. Τέλος, υπάρχει ένα ακόμη Συγκεντρωτικό Επίπεδο το οποίο μειώνει την χωρκή διάσταση των δεδομένων από 7 σε 1 και καταλήγει το δίκτυο με το Πλήρως Συνδεδεμένο Δίκτυο που ταξινομεί το 1x1x2048 διάνυσμα χαρακτηριστικών στις κλάσεις παράγοντας το 1x1x1000 διάνυσμα πιθανοτήτων.

Σημασία του ResNet

Για να γίνει αντιληπτή η σημασία του Residual Net μόνο και μόνο το ποσοστό σφάλματος 3,6% αρκεί. Αποτελεί το καλύτερο από άποψη ακρίβειας Βαθύ Νευρωνικό Δίκτυο που έχει εμφανιστεί μέχρι τώρα και η επίδοση του είναι πολύ δύσκολο να ξεπεραστεί. Επίσης, είναι το πρώτο Νευρωνικό Δίκτυο που διέθετε τέτοιο αριθμό επιπέδων ενισχύοντας την άποψη ότι: ”The Deeper, the Better!!”, δηλαδή ότι Δίκτυα με μεγαλύτερο Βάθος εμφανίζουν και μεγαλύτερη αποτελεσματικότητα. Σημειώνεται ότι η ομάδα του ResNet δοκίμασε ένα δίκτυο με 1202 επίπεδα(!), όμως τα αποτελέσματα δεν ήταν καλύτερα και έτσι παράτησαν την ιδέα. Επιπρόσθετα, δημιούργησε και εφάρμοσε την ιδέα της Διαφορικής (Residual) Μάθησης, η οποία χρησιμοποιήθηκε από πολλά μετεγενέστερα δίκτυα και η οποία είναι αυτή που προφυλάσσει ένα τόσο βαθύ δίκτυο από την υπερεκπαίδευση (overfitting). Από την άλλη πλευρά, είναι προφανές ότι πρόκειται για ένα μεγάλο δίκτυο από άποψη μεγέθους παραμέτρων, η εκπαίδευση των οποίων απαιτεί μεγάλη υπολογιστική δύναμη αλλά και χρόνο. Είναι ενδεικτικό ότι για να πρωτοεκπαιδευτέι απαιτήθηκε χρόνος 2 με 3 εβδομάδων πάνω σε πολύ ισχυρά υπολογιστικά συστήματα, τα οποία διέθεταν 8 GPUs.

3.2.6 SqueezeNet

Το Squeezenet δημιουργήθηκε από τους *Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally and Kurt Keutzer* και αποτελεί ένα δίκτυο, το οποίο αντλεί την σημασία του, όχι από την αποτελεσματικότητα τόσο των αποτελεσμάτων του, όσο από το μικρό μέγεθός του. Ουσιαστικά, πρόκειται για ένα μικρό Συνελικτικό Δίκτυο, το οποίο επιτυγχάνει αξιόλογη ακρίβεια συγκρίσιμη με αυτή του AlexNet, διαθέτοντας όμως 50x λιγότερες παραμέτρους από αυτό.

Η δομή του SqueezeNet

Για να γίνει εφικτή η τόσο μεγάλη σμίκρυνση του μεγέθους του Δικτύου χρησιμοποιήθηκε πληθώρα τεχνικών:

- **Αντικατάσταση των 3x3 Συνελικτικών Επιπέδων με 1x1** αντίστοιχων, εξοικονομούντας, με αυτό τον τρόπο, σχεδόν 9x λιγότερες παραμέτρους. Σημειώνουμε ότι η 1x1 Συνέλιξη εκμεταλλεύεται μόνο την σχέση μεταξύ των καναλιών, αδιαφορώντας για τις σχέσεις κάθε άλλης χωρικής διάστασης μιας και εκτελείται σε μόνο ένα pixel της εικόνας κάθε φορά.

- **Χρήση επιπέδων συμπίεσης (Squeeze Layers)**. Με τον όρο Επιπέδων Συμπίεσης εννοούμε ένα σύνολο από 1x1 Συνελικτικά Επίπεδα τα οποία προηγούνται των Συνελικτικών Επιπέδων με φίλτρα μεγαλύτερων χωρικών διαστάσεων (π.χ. 3x3) με σκοπό την μείωση των παραμέτρων των δεύτερων. Τονίζεται ότι τα επίπεδα που ακολουθούν τα Επίπεδα Συμπίεσης (Squeeze) ονομάζονται Εκτεταμένα (Expand Layers), καθώς και ότι από τα επίπεδα Συμπίεσης προήλθε το ονόμα του Δικτύου.

- Για να είναι πρακτικά χρήσιμο το Δίκτυο μετά την τόσο μεγάλη μείωση του μεγέθους του, θα πρέπει να εξασφαλίσουμε την μέγιστη εκμετάλλευση των παραμέτρων που απέμειναν. Σε αυτή την κατεύθυνση οι δημιουργοί του Δικτύου υποστηρίζουν ότι πρέπει να **αυξήσουμε τον χάρτη ενεργοποίησης της Συνέλιξης** και, επομένως, να χρησιμοποιούμε κατάλληλα τις συνελικτικές παραμέτρους (π.χ. μικρό βήμα (stride)), ώστε να εκμεταλλευτούμε στο μέγιστο την αλληλοσυσχέτιση μεταξύ των εναπομείναντων δεδομένων.

Σημασία του Δικτύου

Όπως προείπαμε, το SqueezeNet είναι ένα Δίκτυο του οποίου η σπουδαιότητα δεν πηγάζει στην αποτελεσματικότητα του αλλά σε άλλους παραμέτρους. Συγκεκριμένα, η ακρίβεια που παρουσιάζει είναι αντίστοιχη του AlexNet, όμως το γεγονός ότι εφάρμοσε πολλές καινοτόμες εφαρμογές συμπίεσης, όπως οι παραπάνω, το καθιστά εξαιρετικά σημαντικό. Έδειξε ότι για την επίτευξης μίας ικανοποιητικής ακρίβειας δεν απαιτείται τόσο μεγάλο πλήθος υπολογισμών, ενώ εγκαινίασε έναν νέο τρόπο σκέψης και μία στροφή του ενδιαφέροντος στην απλοποίηση των Βαθέων Νευρωνικών Δικτύων, ώστε να γίνει η εφαρμογή τους πιο προσιτή σε συσκευές με μικρότερη υπολογιστική δύναμη, κάτι που ίσως να αποδειχθεί πιο χρήσιμο σε τομείς όπως η βιομηχανία.

3.2.7 MobileNet

Το ResNet, που περιγράψαμε νωρίτερα, αποτελεί το καλύτερο Βαθύ Νευρωνικό Δίκτυο από άποψη ακρίβειας και είναι πολύ δύσκολο να ξεπερασθεί. Σαν συνέπεια, οι ερευνητές έπαυσαν να προσπαθούν να βελτιώσουν την αποτελεσματικότητα των Νευρωνικών Δικτύων και έστρεψαν το ενδιαφέρον τους στην έυρεση τεχνικών με τις

οποίες αυτά θα γίνονταν πιο προσιτά στην βιομηχανία, και όχι μόνο. Οι τεχνικές αυτές σκόπευαν στην μείωση της υπολογιστικής ισχύος που απαιτείται για ένα Δίκτυο και κατά συνέπεια στην εφαρμογή τους ακόμα και σε φόρητές συσκεύες. Σε αυτήν την κατεύθυνση ανήκει το SqueezeNet, που περιγράψαμε προηγουμένως, αλλά κυρίως το **MobileNet**.

Το MobileNet δημιουργήθηκε τον Απρίλιο του 2017 από την Google και αποτελέι ένα αρκετά αποτελεσματικό Νευρωνικό Δίκτυο που στοχεύει σε εφαρμογή σε Κινητά (Mobile) και Ενσωματωμένα Συστήματα (Embedded Systems). Τα συγκεκριμένα συστήματα διαθέτουν ισχυρούς περιορισμούς σε ζητήματα μνήμης και υπολογιστικής ικανότητας. Επομένως, τα συμβατικά Συνελικτικά Δίκτυα θα ήταν πολύ δύσκολο όχι μόνο να εκπαιδευτούν αλλά και να εφαρμοστούν στα συστήματα αυτά.

Η Δομή του Δικτύου

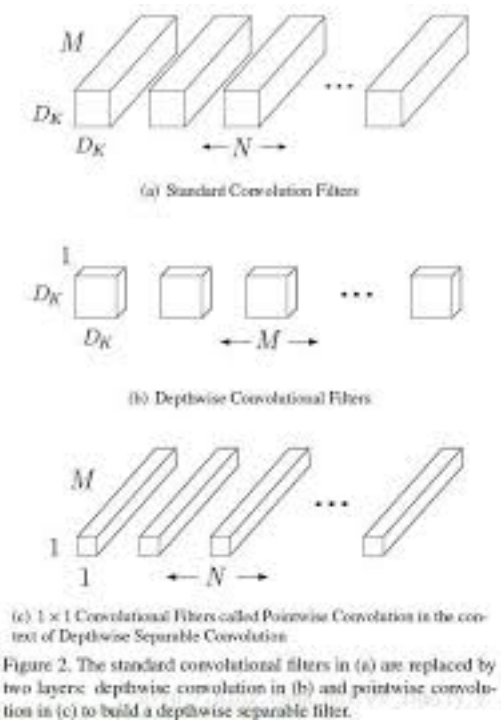
Το MobileNet για να αντιμετωπίσει τους περιορισμούς των Ενσωματωμένων Συστημάτων διαχωρίζει τα Συνελικτικά Επίπεδα σε δύο αντίστοιχα, τα οποία υλοποιούν ακριβώς την ίδια λειτουργία χρησιμοποιώντας λιγότερες παραμέτρους. Ένα συμβατικό Συνελικτικό Επίπεδο υλοποιεί δύο ενέργειες: 1) Φιλτράρει την Είσοδο πολλές φορές και 2) Συνεννώνει όλα τα Φιλτραρίσματα σε μία Δομή Εξόδου. Εδώ αυτές τις δύο ενέργειες τις αναλαμβάνουν δύο διαφορετικά Επίπεδα. Το πρώτο επίπεδο που υλοποιεί το φιλτράρισμα εκτελεί μία σειρά από Συνελίξης με φίλτρα διαστάσεων Kh x Kw x 1, και το δεύτερο δέχεται τα αποτελέσματα από το παραπάνω δυσδιάστατο φιλτράρισμα και τα ενώνει, παράγοντας το τελικό Blob εξόδου. Για την συνέννωση χρησιμοποιείται ένα Συνελικτικό Επίπεδο με φίλτρο διαστάσεων 1 x 1 x O, όπου O ο αριθμός των 1 x 1 φίλτρων του πρώτου επιπέδου. Οι διαφορές των παραπάνω επιπέδων είναι εμφανείς και στο Σχήμα 4.7. Τέλος, αν θεωρήσουμε τα δύο αυτά επίπεδα σαν ξεχωριστά Συνελικτικά Επίπεδα, τότε το MobileNet διαθέτει συνολικά 28 Επίπεδα.

Η σημασία του MobileNet

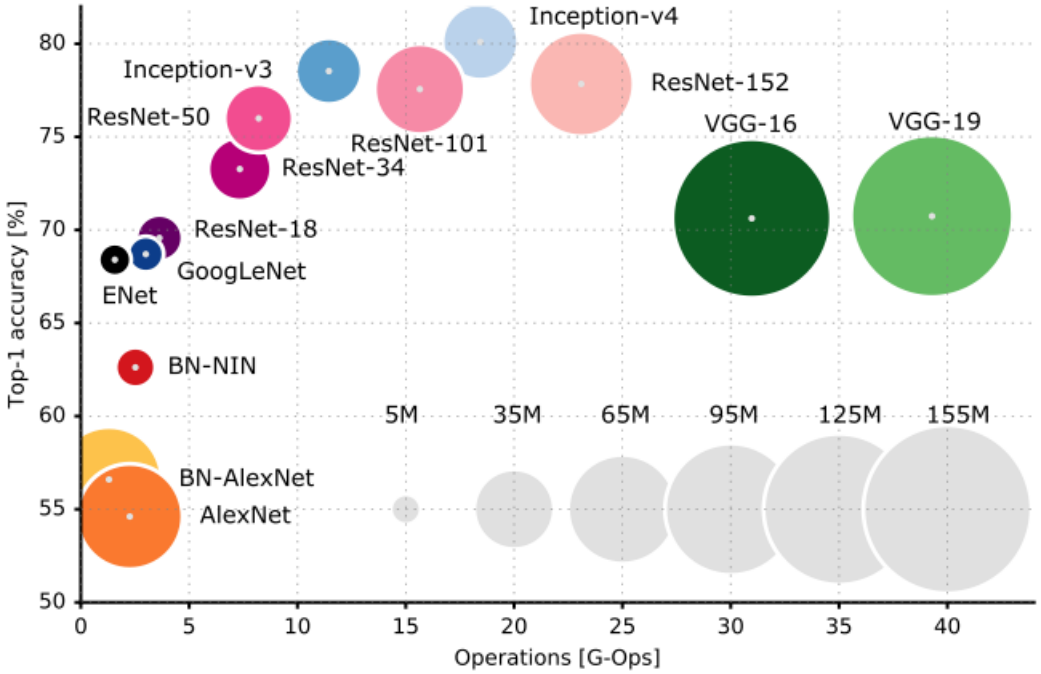
Η σημασία του MobileNet έγκειται στο γεγονός ότι κατάφερε να μειώσει σε μεγάλο βαθμό το μέγεθος των δικτύων και να δείξει ότι η χρήση της Βαθιάς Μηχανικής Μάθησης σε Ενσωματωμένα Συστήματα είναι προσιτός στόχος. Ταυτόχρονα, με τις τεχνικές που χρησιμοποίει αυξάνει την ταχύτητα εκτέλεσης κάτι που είναι πολύ σημαντικό για πολλές εφαρμογές των συγκεκριμένων συστημάτων.

3.2.8 Σύγκριση Δικτύων

Καταλήγοντας στο κεφάλαιο αυτό θα παραθέσουμε ένα εξαιρετικό συγκριτικό διάγραμμα (Σχήμα 3.8) όλων των δικτύων που θα χρησιμοποίσουμε και όχι μόνο, το οποίο λάμβαμε από το εξής άρθρο: [55]. Στο διάγραμμα αυτό γίνεται μία σύγκριση των δικτύων αυτών όσο αφορά τον αριθμό των πράξεων με την ακρίβεια που πετυχαίνουν αλλά και με την μνήμη που απαιτούν για την αποθήκευση των παραμέτρων τους.



Σχήμα 3.7: *Διαχωρισμός των Συνελικτικών επιπέδων σε 2 αντίστοιχα.*



Σχήμα 3.8: *Συγκριτικό Διάγραμμα Δημοφιλών Νευρωνικών Δικτύων*

Κεφάλαιο 4

Υλοποίηση

Σε αυτό το κεφάλαιο θα προσπαθήσουμε να αναλύσουμε την υλοποιήση της διπλωματικής εργασίας δίνοντας έμφαση στους λόγους κάθε ενέργειας. Θα ασχοληθούμε με το περιεχόμενο της εργασίας, το οποίο αποτελεί η ανάλυση της επίδοσης διαφόρων μεθόδων εκτέλεσης της Συνέλιξης στα Βαθιά Νευρωνικά Δίκτυα και το οποίο χωρίζεται σε δύο άξονες:

1. Την υλοποίηση της Συνέλιξης με χρήση των GEMM συναρτήσεων, και
2. Την υλοποίηση της Συνέλιξης απευθείας, χωρίς την χρήση επιπρόσθετων εργαλείων (Direct Convolution).

Σε κάθε περίπτωση θα γίνει τόσο η περιγραφή της εκάστοτε μεθόδου, όσο και η ανάλυση των πλεοκτημάτων και μειονεκτημάτων που αυτή παρουσιάζει καθώς και των κινήτρων που μας τράβηξε το ενδιαφέρον σε αυτή.

4.1 Εισαγωγή - Κριτήρια

Στην προηγούμενα κεφάλαια της εργασίας κάναμε λόγο για την σπουδαιότητα της Μηχανικής Μάθησης και ιδιαίτερα ενός κλάδου αυτής: την Βαθιά Μηχανική Μάθηση. Ακόμη, αναλύσαμε τις δυσκολίες που εμφανίζονται τόσο κατά την εκπαίδευση όσο και κατά την εφαρμογή των Νευρωνικών Δικτύων που ανήκουν σε αυτό τον κλάδο. Τονίσαμε ότι αυτές οι δυσκολίες πηγάζουν κυρίως στο μέγεθος των εκπαιδεύσιμων παραμέτρων των δικτύων, και κυρίως στις παραμέτρους δύο επίπέδων των δικτύων: 1. **Των Συνελικτικών (Convolutional Layers)**, και 2) **των Πλήρως Συνδεδεμένων (Fully Connected Layers)**, καθώς αυτές αποτελούν και την μεγάλη πλειοψηφία του συνόλου των παραμέτρων. Εμείς, στην παρούσα εργασία, θα ασχοληθούμε κυρίως με τα πρώτα, μιας και σε αυτά βρίσκεται η ουσία των δικτύων και αυτά αποτελούν το 95% του χρόνου εφαρμογής και εκπαίδευσης ενός Βαθέως Νευρωνικού δικτύου.

Το μεγάλο πλήθος των παραμέτρων ενός Συνελικτικού επιπέδου απαιτεί την ύπαρξη συσκευών με τόσο μεγάλη υπολογιστική δύναμη, όσο και μεγάλη αποθηκευτική ικανότητα. Μέχρι σήμερα, παρ'όλη την τεχνολογική ανάπτυξη, συσκευές με ικανά χαρακτηριστικά για την εφαρμογή των Συνελικτικών δικτύων είναι δυσεύρετες και κατά συνέπεια

έχουν μεγάλο κόστος. Σαν αποτέλεσμα, το ενδιαφέρον των ερευνητών στράφηκε στην υλοποίηση της Συνέλιξης στα Βαθιά Νευρωνικά δίκτυα με εναλλακτικές μεθόδους.

Οι εναλλακτικές αυτές μέθοδοι βασίστηκαν στην χρησιμοποίηση κάποιων τρίτων εργαλείων τα οποία είχαν αναπτυχθεί σε τέτοιο βαθμό, ώστε να έχει σταματήσει η περαιτέρω εξέλιξή τους. Σύμφωνα με τον δημιουργό του Caffe: **Yangqing.Jia** [56], για την επίλυση του προβλήματος της συνέλιξης προσπάθησε να το μετατοπίσει σε ένα άλλο πρόβλημα που έχει σήμερα αναπτυχθεί πάρα πολύ: τον πολλαπλασιασμό πινάκων. Συγκεκριμένα, το εργαλείο το οποίο έπαιξε καθοριστικό ρόλο στην βελτίωση των απαιτήσεων για την εκπαίδευση και την εφαρμογή των Συνελικτικών δικτύων, είναι οι **GEMM (General Matrix-Matrix Multiplication) συναρτήσεις**. Οι συναρτήσεις αυτές ανήκουν στην ευρύτερη ομάδα των BLAS (Basic Linear Algebra Subprograms) συναρτήσεων [57] και αποτελούν βελτιστοποιημένες ρουτίνες χαμηλού επιπέδου προσφέροντας μεγάλη ταχύτητα υλοποίησης. Ταυτόχρονα, το γεγονός ότι είναι προγραμματισμένες σε χαμηλού επιπέδου γλώσσα, μειώνει την υπολογιστική δύναμη που απαιτείται, ανοίγοντας τον δρόμο για την χρήση όχι μόνο GPU αλλά και CPU για την εφαρμογή των Συνελικτικών δικτύων.

Το αρνητικό των μεθόδων που χρησιμοποιούν τις GEMM συναρτήσεις είναι ότι απαιτούν περισσότερες θέσεις μνήμης από ότι η συμβατική υλοποίηση της Συνέλιξης. Συγκεκριμένα, η επιπρόσθετη μνήμη απατείται κατά την μετατροπή των τρισδιάστατων δομών δεδομένων σε κατάλληλες δυσδιάστατες ώστε να μπορούν να επεξεργαστούν από τις GEMM συναρτήσεις, καθώς στις δυσδιάστατες δομές που προκύπτουν παρατηρούνται πολλές επαναλήψεις ίδιων στοιχείων της πρωταρχικής τρισδιάστατης δομής μιας και αυτά χρησιμοποιούνται πολλές φορές κατά την Συνέλιξη. Το γεγονός αυτό δεν είναι πολύ ευνοϊκό για την εφαρμογή των Βαθιών Νευρωνικών δικτύων σε συσκευές με μικρές αποθηκευτικές δυνατότητες, στις οποίες η παραμικρή προσθήκη περαιτέρω μνήμης αποτελεί σημαντικό πρόβλημα. Για τον σκοπό αυτό, είναι ιδιαιτέρως σημαντική η διερεύνηση των συμβατικών ή απευθείας υλοποιήσεων της Συνέλιξης (Direct Convolution), οι οποίες εμφανίζουν προφανώς πολύ μικρότερες επιδόσεις από τις μη συμβατικές υλοποιήσεις αλλά κάνουν χρήση λιγότερης μνήμης από αυτές.

4.2 GEMM υλοποιήσεις

4.2.1 Εισαγωγή

Η χρήση των GEMM συναρτήσεων για την υλοποίηση της Συνέλιξης έδωσε νέα πνόη στα Βαθιά Νευρωνικά Δίκτυα, καθώς όπως προαναφέραμε μείωσε κατά πολύ την απαιτούμενη υπολογιστική δύναμη για την εφαρμογή τους [58]. Παρόλα αυτά, η συγκεκριμένη ιδέα διαθέτει και ένα σημαντικό μειονέκτημα, το οποίο πηγάζει από την ανάγκη ύπαρξης μιας διαδικασίας προσαρμογής τόσο πριν την εκτέλεση των GEMM συναρτήσεων όσο και μετά. Αυτή που προηγείται της εκτέλεσης των GEMM, αποσκοπεί στον κατάλληλο μετασχηματισμό της δομής που το Συνελικτικό επίπεδο δέχεται σαν είσοδο σε μία δομή η οποία έχει δύο διαστάσεις. Η διαδικάσία αυτή ονομάζεται **Υποβιβασμός Διαστάσεων (Lowering)**. Από την άλλη η διαδικασία που έπεται της εκτέλεσης των

Πίνακας 4.1: *Πίνακας Συνελικτικών Παραμέτρων για την περιγραφή των μεθόδων*

Παράμετροι	Τιμές
PAD	1
STRIDE	1
DILATION	1

GEMM συναρτήσεων ονομάζεται **Αύξηση Διαστάσεων (Lifting)**, Και στοχεύει στην μετατροπή της δυσδιάστατης δομής που προκύπτει από την εκτέλεση του πολλαπλασιασού πινάκων σε μία τρισδιάστατη ή τετραδιάστατη δομή, αναλόγως την εφαρμογή ή όχι δέσμης εικόνων, η οποία θα έχει την κατάλληλη μορφή ώστε να γίνει είσοδος στα επόμενα επίπεδα.

Σε αυτό το σημείο θα αναλύσουμε όλες τις μεθόδους που υλοποιήσαμε και χρησιμοποιούν GEMM συναρτήσεις. Οι μέθοδοι αυτές διαφέρουν τόσο στο σκεπτικό, όσο και στην διαφορετική υλοποίηση των Lowering Και Lifting διαδικάσιων αλλά στην απαιτούμενη υπολογιστική δύναμη και αποθηκευτική ικανότητα. Σε κάθε περίπτωση θα γίνεται αναλυτική περιγραφή όλων των στοιχείων κάθε μεθόδου.

4.2.2 Μέθοδος 1: Ακριβό (Expensive) Lowering

Η συγκεκριμένη μέθοδος προήλθε από μία ομάδα του Πανεπιστημίου του Wisconsin-Madison και περιέχεται στο CaffeConTroll [59] , μία προσπάθεια να διερευνητηθούν οι δυνατότητες της χρήσης καρτών γραφικών (GPUs) ή επεξεργαστικών μονάδων (CPUs) για την εφαρμογή του Caffe. Σε αυτήν την προσπάθεια ανήκουν και οι δύο επόμενες τεχνικές που θα αναφέρουμε. Τονίζεται ότι η μέθοδος του Ακριβού Lowering βρίσκει μεγάλης αναγνώρισης σε σημείο ώστε να είναι και αυτή που χρησιμοποιεί το ίδιο το Caffe για την εκτέλεση των Συνελικτικών Επιπέδων του.

Περιγραφή Μεθόδου

Όπως αποκαλύπτει και το όνομα της, η μέθοδος αυτή παρέχει μια GEMM εκτέλεση στην οποία η μετατροπή της αρχικής δομής εισόδου σε μία δυσδιάστατη (Lowering) είναι πιο περίπλοκη και συνεπώς περισσότερο απαιτητική. Αντιθέτως, η μετατροπή του αποτελέσματος του πολλαπλασιασμού πινάκων στην κατάλληλη τρισδιάστατη δομή (Lifting) είναι μία αμελητέα διαδικασία και συγκεκριμένα στην παρούσα μέθοδο δεν απαιτείται, όπως θα δούμε και παρακάτω.

Lowering

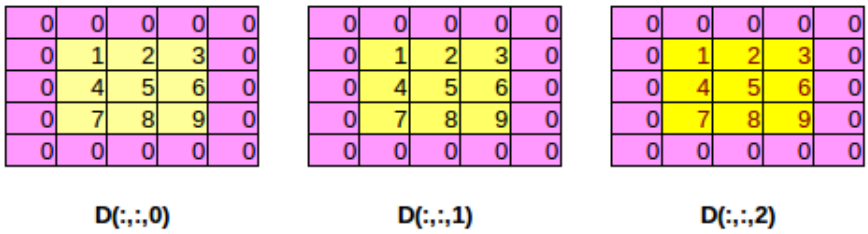
Για την πιο περιγραφική ανάλυση της μεθόδου ορίζουμε τις εξής δομές και παραμέτρους της Συνέλιξης:
Παράμετροι Συνέλιξης:

Δομή Εισόδου (Input Blob):

$$D = h \times w \times c \tag{4.1}$$

Σε αυτή την δομή εφαρμόζουμε zero padding ίσο με ένα και προκύπτει μία δομή:

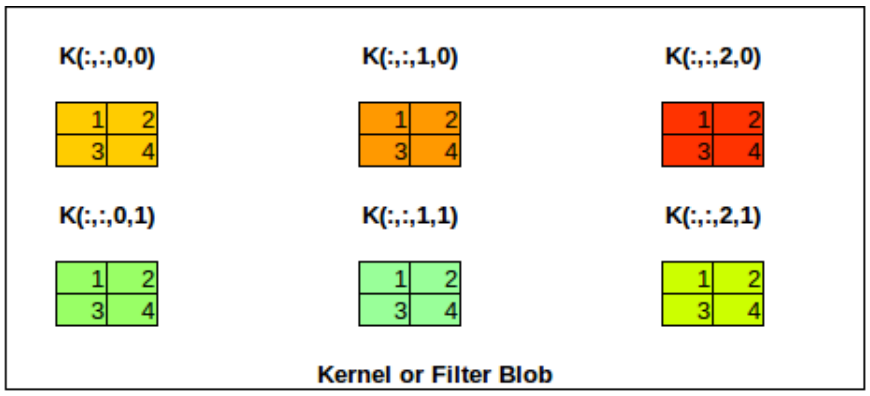
$$D = (h + 1) \times (w + 1) \times c \tag{4.2}$$



Σχήμα 4.1: *Blob Εισόδου*

Δομή Φίλτρου (Kernel or Filter Blob):

$$K = kh \times kw \times c \times n \tag{4.3}$$



Σχήμα 4.2: *Blob Φίλτρου*

Επομένως παρατηρούμε ότι διαθέτουμε μία τρισδιάστατη δομή για τα δεδομένα εισόδου, όπου το ύψος (h) και πλάτος (w) ίσο με 3 και βάθος (channels - c) ίσο με 3. Μετά την προσθήκη του zero padding οι χωρικές διαστάσεις (h και w) της εισόδου μετατρέπονται σε

$$3 + 2 = 5$$

. Παράλληλα η δομή που αφορά τα βάρη των φίλτρων είναι τετρασδιάστατη και έχει χώρικες διαστάσεις (kh = ύψος και kw = πλάτος) ίσες με 2, ενώ το βάθος (c) είναι όμοιο με το βάθος της εισόδου και ίσο με 3. Η τέταρτη διάσταση αφορά τον αριθμό των φίλτρων και στην περίπτωση μας είναι 2. Συνοπτικά, τα παραπάνω είναι εμφανή

στον πίνακα που ακολουθεί.

Blob	*Height*	*Width*	*Depth*	*NumberBlob*
Input	$3+2=5$	$3+2=5$	3	–
Kernel	2	2	3	2

Είναι εμφανές ότι οι παραπάνω δομές δεν διαθέτουν την κατάλληλη μορφή για την εκτέλεση ενός δυσδιάστατου πολλαπλασιασμού πινάκων μέσω των GEMM συναρτήσεων. Για αυτό και θα πρέπει να μετατραπούν σε μία κατάλληλη, ή οποία θα εκτελεί ταυτόχρονα και την Συνέλιξη. Αυτό θα γίνει μέσω του Lowering.
Στην συγκεκριμένη μέθοδο το Lowering είναι μία αρκετά περίπλοκη διαδικασία. Η ιδέα σε αυτή είναι απλή: Παράγουμε μία καινούργια δομή Εισόδου η οποία θα έχει διαστάσεις:

$$\hat{D} = kh * kw * c \times oh * ow \tag{4.4}$$

όπου oh και ow είναι οι διαστάσεις του Blob εξόδου που θα προκύψει μετά τον πολλαπλασιασμό και υπολογίζονται ως εξής:

$$\begin{aligned} oh &= [(h + 2 * pad_h) - (kh * dilation_h - 1) + 1]/stride_h + 1, \\ ow &= [(w + 2 * pad_w) - (kw * dilation_w - 1) + 1]/stride_w + 1 \end{aligned} \tag{4.5}$$

Στις παραπάνω σχέσεις τονίζεται ότι pad_h, pad_w είναι το zero padding που εφαρμόζεται στην είσοδο στο ύψος και στο πλάτος αντίστοιχα, $dilation_h$, $dilation_w$ είναι η διαστολή του φίλτρου στον άξονα του ύψους και του πλάτους αντίστοιχα, και το $stride_h$, $stride_w$ αφορά το βήμα στο ύψος και στο πλάτος με το οποίο εφαρμόζονται τα φίλτρα στην είσοδο κατά την συνέλιξη. Με βάση τα δεδομένα που θέσαμε προηγουμένως, έχουμε ότι:

$$oh = 4, ow = 4 \tag{4.6}$$

Στην συγκεκριμένη δυσδιάστατη δομή γεμίζουμε κάθε στήλη με τις τιμές της εισόδου στις οποίες θα εφαρμοστεί το φίλτρο κατά την συνέλιξη. Για αυτόν τον λόγο η δομή έχει **kh*kw*c** αριθμό σειρών, καθώς τόσα είναι τα στοιχεία ενός φίλτρου. Ο αριθμός των στηλών υποδηλώνει πόσες φορές θα εφαρμοστεί ένα φίλτρο στην δομή εισόδου. Έτσι, δικαιολογείται και η δεύτερη διάσταση του Blob που είναι ίση με **n*oh*ow**, μιας και το κάθε ένα από τα n σε αριθμό φίλτρα εφαρμόζεται τόσες φορές όσες και οι διαστάσεις του Blob εξόδου, δηλαδή oh*ow. Στην περίπτωση των δεδομένων που χρησιμοποιούμε είναι:

$$\hat{D} = 2 * 2 * 3 \times 4 * 4 = 12 \times 16 \tag{4.7}$$

Για να γίνει πιο κατανοητή η παραπάνω διαδικάσία απεικονίζεται και στο παρακάτω Σχήμα 4.3.
Σε αυτό τα τμήματα της δομής με το ίδιο χρώμα αντιστοιχούν στα τμήματα με τα οποία προήλθαν από τα τμήματα της δομής εισόδου με το αντίστοιχο χρώμα. Δηλαδή, χωρίζεται σε 3 (όσο και το βάθος της εισόδου) 4×16. Είναι εμφανές ότι το πρώτο τμήμα θα πολλαπλασιαστεί μόνο με τα πρώτα κανάλια των φίλτρων, το δεύτερο με μόνο με

0	0	0	0	0	1	2	3	0	4	5	6	0	7	8	8
0	0	0	0	1	2	3	0	4	5	6	0	7	8	9	9
0	1	2	3	0	4	5	6	0	7	8	9	0	0	0	0
1	2	3	0	4	5	6	0	7	8	9	0	0	0	0	0
0	0	0	0	0	1	2	3	0	4	5	6	0	7	8	8
0	0	0	0	1	2	3	0	4	5	6	0	7	8	9	9
0	1	2	3	0	4	5	6	0	7	8	9	0	0	0	0
1	2	3	0	4	5	6	0	7	8	9	0	0	0	0	0
0	0	0	0	0	1	2	3	0	4	5	6	0	7	8	8
0	0	0	0	1	2	3	0	4	5	6	0	7	8	9	9
0	1	2	3	0	4	5	6	0	7	8	9	0	0	0	0
1	2	3	0	4	5	6	0	7	8	9	0	0	0	0	0

Modified Input Blob

Σχήμα 4.3: *Τροποποιημένο Blob Εισόδου*

τα δεύτερα κανάλια των φίλτρων, κ.ο.κ.. Η παραπάνω περιγραφή μας δίνει μια ιδέα για το πως θα πρέπει να μετασχηματιστεί αντίστοιχα και η δομή του φίλτρου ώστε να έχουμε τον δυσδιάστατο πολλαπλασιασμό με την προηγούμενη. Το θετικό είναι ότι παρόλο για την δομή εισόδου το Lowering είναι πολύ περίπλοκο, για την δομή του φίλτρου τα πράγματα είναι πολύ πιο απλά. Συγκεκριμένα, η τροποποιημένη δομή του φίλτρου έχει διαστάσεις:

$$\hat{K} = n \times kh * kw * c \tag{4.8}$$

,δηλαδή για τις τιμές του παραδείγματος:

$$\hat{K} = 2 \times 2 * 2 * 3 = 2 \times 12 \tag{4.9}$$

Έτσι η δομή που προκύπτει θα είναι:

1	2	3	4	1	2	3	4	1	2	3	4
1	2	3	4	1	2	3	4	1	2	3	4

Modified Kernel Blob

Σχήμα 4.4: *Τροποποιημένο Blob Φίλτρου*

Παρατηρούμε ότι πρόκειται για μία απλή αλλαγή των διαστάσεων της δομή του φίλτρου. Αναλυτικότερα, συνεννώνουμε τις πρώτες τρεις διαστάσεις της αρχικής δομής (ύψος, πλάτος και βάθος) σε μία φτιάχνοντας ουσιαστικά μία σειρά-φίλτρο. Για κάθε ένα φίλτρο της Συνέλιξης προσθέτουμε και μία ακόμα σειρά στην δομή. Ένας ακόμα καλύτερος παρατηρητής θα συνειδητοποιούσε για το Lowering των φίλτρων δεν απαιτείται κάποια περαιτέρω διαδικασία μιας και η μορφή των blob που χρησιμοποιεί το Caffe αλλά και αυτά αποθηκεύονται με row-major τρόπο.

Ολοκληρώνοντας το τμήμα του Lowering, για την υλοποίηση αυτού χρησιμοποιήθηκαν

οι εξής μαθηματικές σχέσεις, θέτοντας pad = 0:

$$\hat{D}[c * m + r, :] = vec(D[:, r : r + kh, c : c + kw]) \tag{4.10}$$

$$\hat{K} = vec(K) \tag{4.11}$$

, όπου ο συμβολισμός ":" υποδηλώνει το εύρος. Έτσι το "r:r+kh" περιέχει όλα τα στοιχεία από το r μέχρι το r+kh.

Gemm Εκτέλεση Στην εκτέλεση της GEMM ρουτίνας υλοποιείται τον παρακάτω πολλαπλασιασμό πινάκων:

$$\hat{R} = \hat{K} * \hat{D} \tag{4.12}$$

Οι διαστάσεις των πινάκων, οι οποίοι πολλαπλασιάζονται είναι οι εξής:

$$(2 \times 12) * (12 \times 16) = (2 \times 16) \tag{4.13}$$

και το αποτέλεσμα του πολλαπλασιασμού:

12	33	54	27	54	111	141	63	108	201	231	99	42	69	78	78
12	33	54	27	54	111	141	63	108	201	231	99	42	69	78	78

Matrix Multiplication Result

Σχήμα 4.5: *Αποτέλεσμα GEMM*

Επομένως θα μπορούσαμε να πούμε ότι εκτελούνται **2 * 12 * 16 = 384** πολλαπλασιασμοί.

Lifting

Όσον αφορά την διαδικασία του Lifting στην παρούσα μέθοδο τα πράγματα είναι πολύ ευνοϊκά. Συγκεκριμένα, παρατηρούμε ότι το αποτέλεσμα της GEMM υλοποίησης είναι μία δομή με αριθμό στοιχείων όσο και η επιθυμητή δομή εξόδου. Πιο αναλυτικά, η δομή που προκύπτει έχει 16 * 2 = 32 στοιχεία, ενώ η επιθυμητή δομή εξόδου πρέπει να έχει:

$$oh * ow * n = 4 * 4 * 2 = 32 \tag{4.14}$$

Ακόμα παρατηρούμε ότι στην προκύπτουσα δομή τα δεδομένα είναι τοποθετημένα στην μνήμη με row-major τρόπο έχοντας ως πιο εσωτερική διάσταση αυτή του ow και ακολουθούν αυτές των oh και n, όπως δηλαδή θα έπρεπε να ήταν αποθηκευμένα και στην επιθυμητή δομή. Επομένως, για την παρούσα μέθοδο δεν απαιτείται κάποια πρόσθετη εργασία για το Lifting.

4.2.3 Μέθοδος 2: Ακριβό (Expensive) Lifting

Η μέθοδος αυτή έχει αντίστροφη λογική από την μέθοδο που προηγήθηκε. Εκεί όλο το βάρος έπεφτε στην διαδικασία του Lowering, ενώ το Lifting δεν απαιτούνταν καν. Εδώ, όμως, τώρα γίνεται ακριβώς το αντίθετο: Η μέθοδος διαθέτει μία πολύ απλή διαδικασία για το Lowering και επικεντρώνεται στο Lifting, όπου υλοποιεί μεγαλύτερο έργο, παίρνοντας ακόμα και τμήμα του έργου της GEMM εκτέλεσης. Σημειώνεται ότι αυτή η μέθοδος έχει την ίδια πηγή με την πρώτη [59].

Περιγραφή Μεθόδου

Για την περιγραφή της μεθόδου θα χρησιμοποιήσουμε τις ίδιες δομές εισόδου και φίλτρων που χρησιμοποίησαμε και προηγουμένως.

Lowering

Το Lowering της συγκεκριμένης μεθόδου αποτελεί ένα απλό μετασχηματισμό διαστάσεων του Blob εισόδου. Ουσιαστικά, μετατρέπουμε τις διαστάσεις του Blob από c x h x w (δηλαδή από μία δομή όπου η πιο εσωτερική διάσταση είναι το w, στην συνέχεια το h, κ.ο.κ.) σε ένα Blob με διαστάσεις **w * h x c**. Αντίστοιχα, για το Lowering του Φίλτρου υλοποίούμε πάλι έναν μετασχηματισμό διαστάσεων. Έτσι από την αρχική δομή των n x c x kh x kw διαστάσεων οδηγούμαστε σε μία δομή με διαστάσεις **c x n * kh * kw**. Η συγκεκριμένη διαδικασία είναι αρκετά απλή μιας και παρατηρούμε ότι δεν υπάρχει καμία επανάληψη κάποιας τιμής στις νέες δομές, οι οποίες για αυτόν τον λόγο έχουν και ίδιο αριθμό στοιχείων με τις αρχικές. Πιο αναλυτικά, τα Τροποποιημένα Blobs που προκύπτουν είναι εμφανή στο Σχήμα 4.6, όπου αποκαλύπτονται και οι διαστάσεις τους για τα δεδομένα του παραδείγματος που πήραμε από την περιγραφή της πρώτης μεθόδου, δηλαδή ότι:

$$\begin{aligned} \hat{D} &= (5 * 5) \times 3 = 15 \times 3, \\ \hat{K} &= 3 \times (2 * 2 * 2) = 3 \times 8 \end{aligned} \tag{4.15}$$

Θέτοντας πάλι pad=0, χρησιμοποιήσαμε για το Lowering τις εξής απλοποιημένες μαθηματικές σχέσεις:

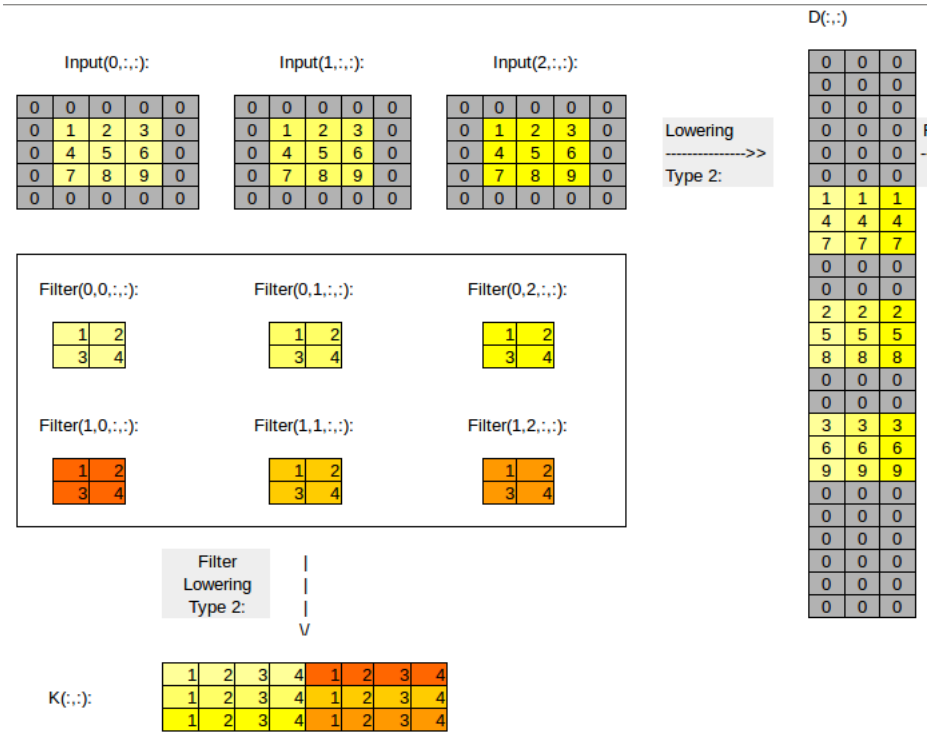
$$\hat{D}[c * w + r, :] = vec(D[:, r, c] \tag{4.16}$$

$$\hat{K}[:, i * kw + j + p * kw * kh] = vec(K[p, :, i, j], \tag{4.17}$$

, όπου $r \in 1...kh - 1$, $c \in 1...kw$, $i \in 1...kh$, $j \in 1...kw$ και $p \in 1...n - 1$

Εκτέλεση GEMM

Στην φάση της εκτέλεσης της GEMM συνάρτησης πραγματοποιείται ο εξής πολλα-



Σχήμα 4.6: *Τροποποιημένο Blob Εισόδου και Φίλτρου*

πλασιασμός:

$$\hat{R} = \hat{D} * \hat{K} \tag{4.18}$$

Και προκύπτει μία δομή με διαστάσεις:

$$\hat{R} : w * h \times c * c \times n * kh * kw \tag{4.19}$$

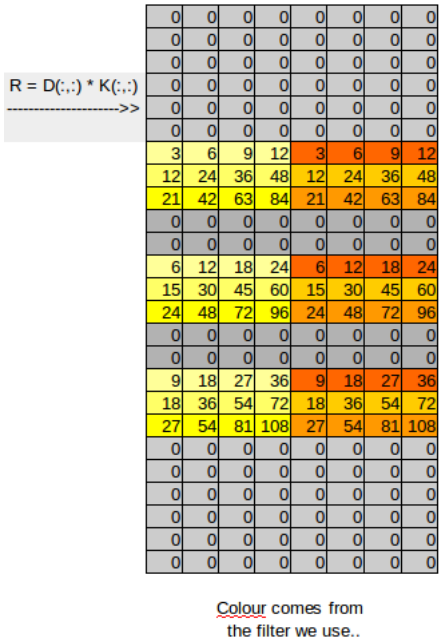
, η οποία για τα δεδομένα που χρησιμοποιούμε είναι:

$$\hat{R} : 15 \times 3 * 3 \times 8 = 15 \times 8 \tag{4.20}$$

Το αποτέλεσμα του πολλαπλασιασμού παρουσιάζεται στο Σχήμα 4.7. Παρατηρούμε ότι το αποτελέσμα δεν έχει σε καμία περίπτωση τις διαστάσεις της επιθυμητής δομής εξόδου κάτι που μας φανερώνει την σημασιά του Lifting που ακολουθεί.

Lifting

Το Lifting στην συγκεκριμένη μέθοδο είναι πολύ σημαντικό. Όπως αναφέραμε και παραπάνω, σε αυτό λαμβάνουν χώρα εργασίες που σε κανονικές συνθήκες θα άνηκαν στην δικαιοδοσία της GEMM συνάρτησης. Σκοπός της διαδικάσίας είναι να γίνει ο κατάλληλος συνδιασμός των στοιχείων του Blob $\hat{R}$ ώστε να παραχθεί ένα νέο Blob το οποίο θα περιέχει την έξοδο στην κατάλληλη μορφή. Για να γίνει αυτό χρησιμοποιείται



Σχήμα 4.7: *Τροποποιημένο Blob Εισόδου και Φίλτρου*

η εξής μαθηματική σχέση:

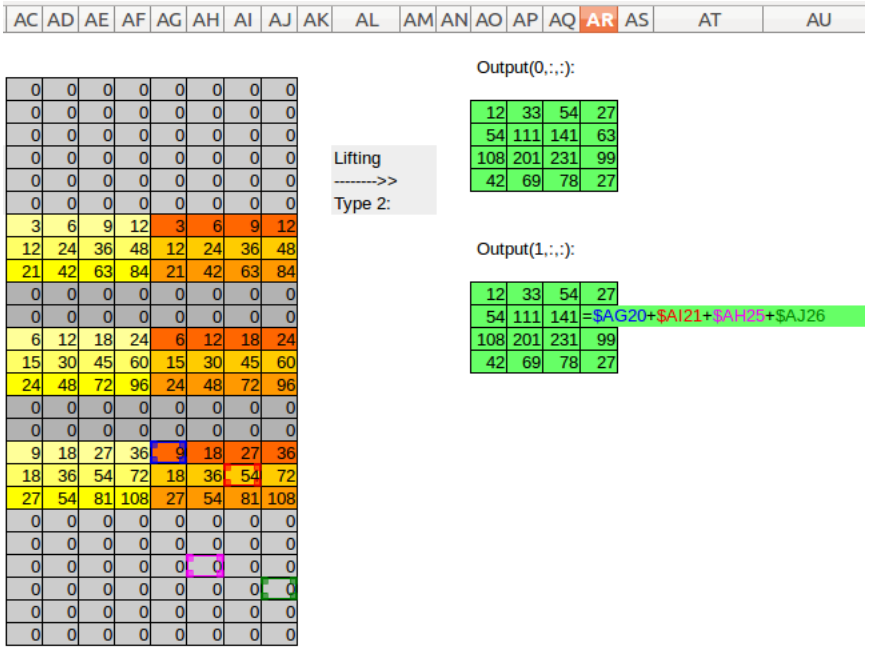
$$R[p,r,c] = \sum_{i=0}^{kh-1} \sum_{j=0}^{kw-1} \hat{R}[(c+j)*w+r+i, i*kw+j+p*kh*kw] \tag{4.21}$$

, όπου :

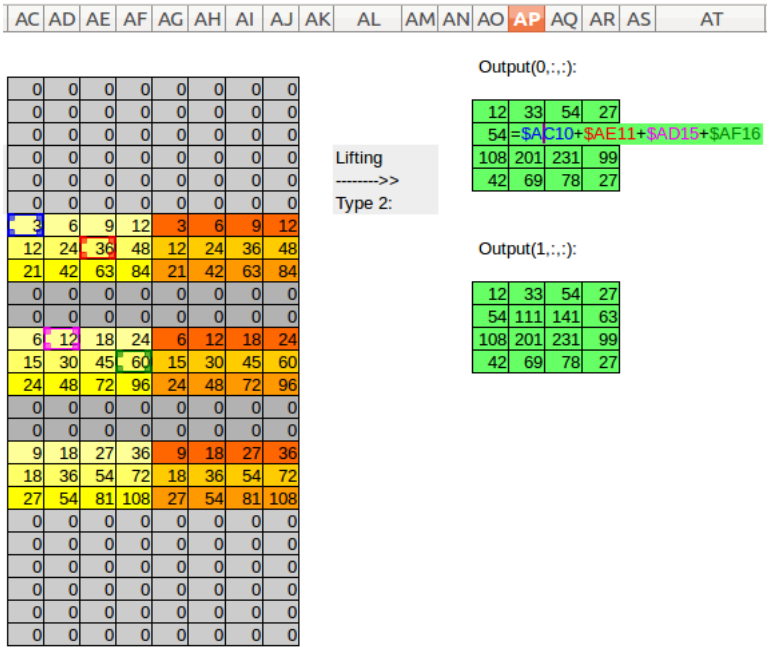
$$\begin{aligned} i &\in 0,...,kh-1, \\ j &\in 0,...,kw-1, \end{aligned} \tag{4.22}$$

Για να γίνει λίγο πιο κατανοητή η παραπάνω σχέση ακολουθούν τα Σχήματα 4.8 και 4.9, στα οποία περιγράφεται πως γίνεται η επιλογή των στοιχείων που θα επελεχθούν από την δομή $\hat{R}$ ώστε αυτά να προστεθούν μεταξύ τους και να παράξουν την τελική τιμή του Blob εξόδου.
Έτσι για το στοιχείο στην θέση (1,1,3) του Blob εξόδου έχουμε την εξής επιλογή:

, ενώ για το στοιχείο στην θέση (0,1,1):



Σχήμα 4.8: *Παράδειγμα για τον υπολογισμό του στοιχείου (1,1,3) του Blob εξόδου.*



Σχήμα 4.9: *Παράδειγμα για τον υπολογισμό του στοιχείου (0,1,1) του Blob εξόδου.*

4.2.4 Μέθοδος 3: Ισορροπημένο (Balanced) Lowering - Lifting

Η μέθοδος του Ισορροπημένου Lowering-Lifting αποτελεί μία προσπάθεια να συνδιάσει τις δύο προηγούμενες με σκοπό να εκμεταλλευτεί σε μεγάλο βαθμό τα πλεονεκτήματα αμφότερων. Προφανώς, και αυτή προήλθε από την ίδια ομάδα [59] που υλοποίησε τις προηγούμενες μεθόδους και, όπως το φανερώνει και το όνομά της προσπαθεί να μοιράσει τον φόρτο εργασίας της Συνέλιξης τόσο στο Lowering όσο και στο Lifting.

Περιγραφή Μεθόδου

Για ακόμα μία φορα θα κάνουμε χρήση του συστήματος που δημιουργήσαμε προηγουμένως, δηλαδή θα βασιστούμε πάνω στις ίδιες δομές εισόδου και φίλτρων. Έτσι:

Lowering Στην συγκεκριμένη μέθοδο το Lowering είναι εξαιρετικής σημασίας. Παρόλο που δεν αναλαμβάνει όλο το φόρτο εργασίας, είναι ρυθμιστική διαδικασία για τις υπόλοιπες που ακολουθούν. Τονίζεται ότι η υλοποίηση που χρησιμοποιούμε έχει ελαφρώς διαφοροποιηθεί από την αρχική του Paper [59], ώστε να καλυφθούν ορισμένα κενά σε αυτή αλλά και να βελτιωθεί η επίδοσή της. Αναλυτικότερα, επικεντρωνόμαστε στο Lowering της δομής εισόδου. Εκεί οι διαστάσεις της νέας δομής εξαρτόνται από δίαφορους παραμέτρους, όπως το stride, το dilation κτλ, και συγκεκριμένα ισχύει:

$$\hat{D} : (H * ow) \times (c * kh)$$
$$\text{,όπου } H = \begin{cases} h, \text{ if } stride_h < kh \\ oh, \text{ if } stride_h >= kh \end{cases} \tag{4.23}$$

Στα δεδομένα που χρησιμοποιούμε η δομή $\hat{D}$ παίρνει τις εξής διαστάσεις:

$$\hat{D} : (5 * 4) \times (3 * 2) = 2 \times 6 \tag{4.24}$$

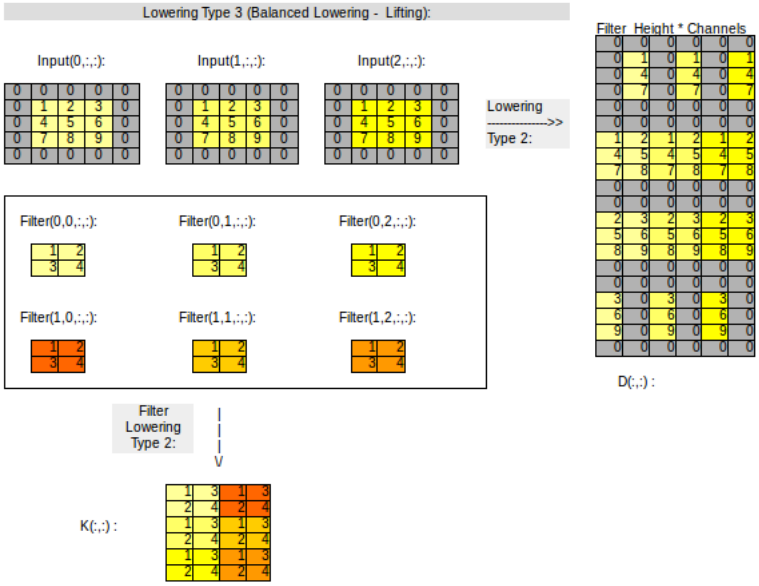
,μιας και $stride_h$1 και kh = 2.
Όσον δε αφορά το Lowering του φίλτρου εκεί τα πράγματα είναι πολύ πιο απλά, μιας και πάλι γίνεται ένας απλός μετασχηματισμός των διαστάσεων. Έτσι, από την αρχική δομή η οποία είχε διαστάσεις : $n \times c \times kh \times kw$, τώρα δημιουργούμε μία νέα δομή με διαστάσεις $(c * kh) \times (n * kw)$, δηλαδή στην περίπτωσή μας: $3 * 2 \times 2 * 2 = 6 \times 4$. Οι δομές που προκύπτουν από το Loweirng της μεθόδου Balanced Lowering-Lifting απεικονίζονται στο Σχήμα 4.10, ενώ οι μαθηματικές σχέσεις πάνω στις ποίες βασίστηκαν είναι οι εξής:

$$\hat{D}[c * n + r, :] = vec(D[r, c : c + k, :]),$$
$$\hat{K}[:, p * kh + i] = vec(K[p, i, :, :)) \tag{4.25}$$

, όπου :

$$i \in 0, ..., kh - 1,$$
$$p \in 0, ..., n - 1,$$
$$r \in 0, ..., h - 1, and$$
$$c \in 0, ..., w - 1 \tag{4.26}$$

Εκτέλεση GEMM



Σχήμα 4.10: *Τα τροποποιημένα Blobs μετά το Lowering της Balanced Lowering-Lifting μεθόδου.*

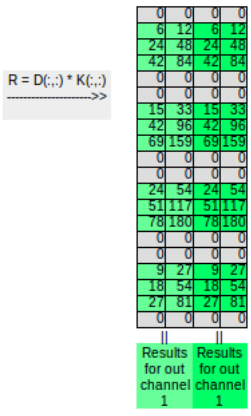
Σε αυτό το στάδιο υλοποιείται ο πολλαπλασιασμός:

$$\hat{R} : \hat{D} * \hat{K} \tag{4.27}$$

, ο οποίος θα παράξει μία δομή διαστάσεων :

$$\hat{R} : (H * ow) \times (n * kw) \tag{4.28}$$

Η δομή αυτή απεικονίζεται στο Σχήμα 4.11:



Σχήμα 4.11: *Το αποτέλεσμα της εκτέλεσης της GEMM συνάρτησης.*

Lifting

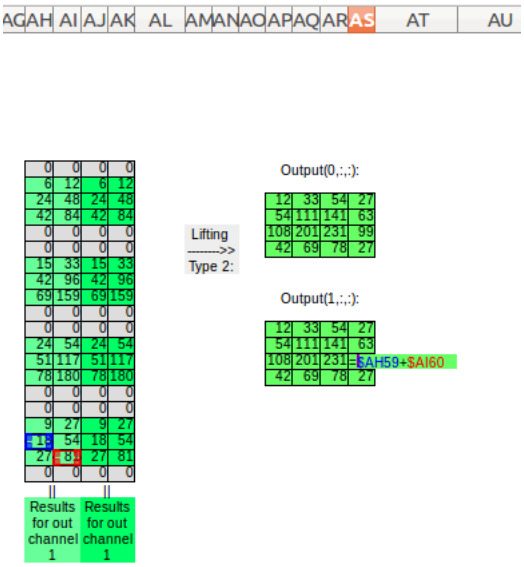
Όπως στην περίπτωση της μεθόδου Expensive Lifting, έτσι και εδώ η διαδικασία του Lifting έχει παρόμοια εφαρμογή. Συγκεκριμένα, εκτός από τον μετασχηματισμό της δομής εξόδου της GEMM συνάρτησης σε μία δομή η οποία θα είναι σε θέση να χρησιμοποιηθεί από τα επόμενα επίπεδα, σε μία τρισδιάστατη δομή δηλαδή, κατά την διάρκεια του Lifting πραγματοποιείται και κάποια επιλογή και συνδιασμός των στοιχείων για την παραγωγή του τελικού Blob εξόδου. Η τελευταία εργασία αποτελεί και αυτή τμήμα της GEMM εκτέλεσης και την αναλαμβάνει το Lifting ώστε να μειωθεί ο φόρτος εργασίας της. Συγκεκριμένα, για το Lifting αυτής της μεθόδου βασιζόμαστε στην εξής μαθηματική σχέση:

$$R[p,r,c] = \sum_{j=0}^{kw-1} \hat{R}[c*h + p*kw + r + j, j] \tag{4.29}$$

, όπου :

$$j \in 0,\ldots,kw-1 \tag{4.30}$$

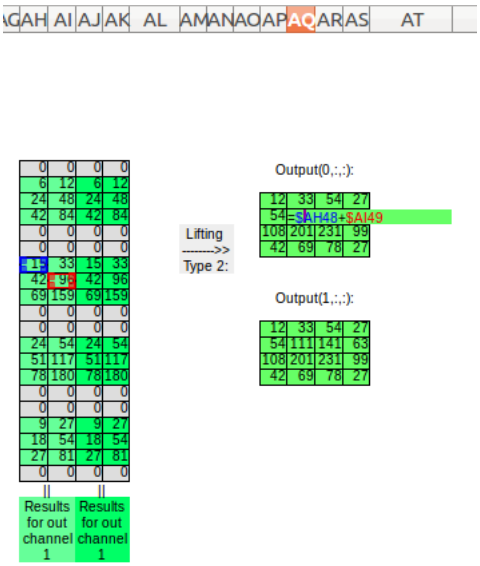
Με αυτόν τον τρόπο παράγεται η τελική δομή εξόδου. Παρακάτω ακολουθούν δύο γραφικά παραδείγματα της επιλογής των στοίχειων από την δομή της εξόδου της GEMM συνάρτησης, σύμφωνα με την παραπάνω σχέση, ώστε αυτή να γίνει πεισσότερο κατανοήτή. Τα παραδείγματα αφορούν τα στοιχεία των θέσεων (1,1,3) (Σχήμα 4.12) και (0,1,1) (Σχήμα 4.13)



Σχήμα 4.12: *Παράδειγμα για τον υπολογισμό του στοιχείου (1,1,3) του Blob εξόδου.*

4.2.5 Μέθοδος 4: Ker2row-NACC

Η μέθοδος που θα περιγραφεί σε αυτό το σημείο δημιουργήθηκε από το Πανεπιστήμιο του Δουβλίνου (School of Computer Vision & Statistics, Trinity College Dublin) και έχει εντελώς διαφορετική λογική από τις προηγούμενες. Σε αυτές στην διαδικασία



Σχήμα 4.13: *Παράδειγμα για τον υπολογισμό του στοιχείου (0,1,1) του Blob εξόδου.*

του Lowering καλούμασταν να μεταβάλλουμε σε μεγάλο βαθμό την δομή της Εισόδου, εδώ, όμως, δίνουμε μεγαλύτερη σημασία στην δομή του Φίλτρου, διατηρώντας την δομή της Εισόδου όπως έχει. Με αυτόν τον τρόπο μειώνουμε σε μεγάλο βαθμό τον φόρτο εργασίας του Lowering καθώς στην συντριπτική τους πλειονότητα οι δομές των Φίλτρων έχουν μικρότερες διαστάσεις από τις αντίστοιχες των Εισόδων.

Περιγραφή Μεθόδου

Για την περιγραφή της συγκεκριμένης μεθόδου δεν θα χρησιμοποίησουμε το παράδειγμα συστήματος των προηγούμενων μεθόδων, αλλά θα βασιστούμε στην περιγραφή του Paper [60] από το οποίο προήλθε η μέθοδος. Αναλυτικότερα: Στο Σχήμα

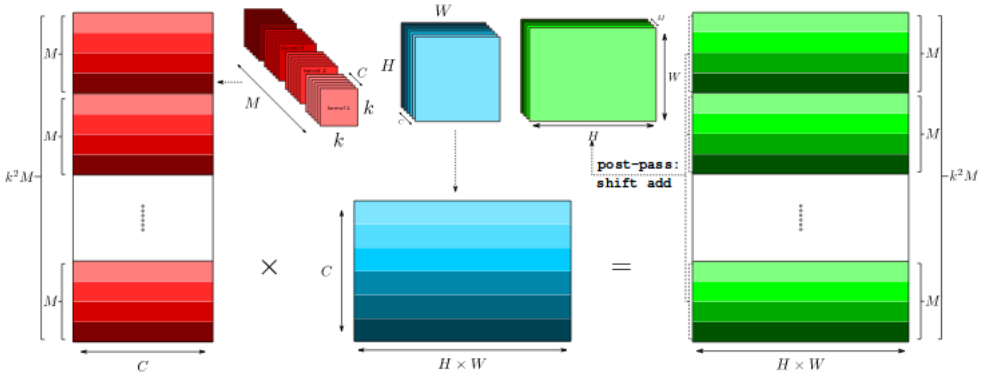


Fig. 4: MCMK using the "Kernel to Row (kn2row)" method

Σχήμα 4.14: *Η μέθοδος Ker2row-NACC.*

4.14 εμφανίζονται όλα τα στάδια της μεθόδου. Ας ξεκινήσουμε να τα εξηγήσουμε ένα ένα:

Lowering Όπως προείπαμε, στο Lowering της μεθόδου Ker2row-NACC η δομή της Εισόδου διατηρείται ως έχει, ενώ επικεντρωνόμαστε στην μετατροπή της δομής του Φίλτρου. Ξέρουμε ότι η δομή του φίλτρου έχει διαστάσεις $n \times c \times kh \times kw$. Σκοπός του Lowering είναι να μετατρέψει την αρχική δομή του φίλτρου σε μία δομή με διαστάσεις:

$$\hat{K} : (kh * kw) * n \times c \tag{4.31}$$

Παρατηρούμε ότι στην δομή $\hat{K}$ η πιο εσωτερική διάσταση είναι πλέον η διάσταση του βάθους, ακολουθεί αυτή του αριθμού των φίλτρων και στην συνέχεια οι χωρικές διαστάσεις του φίλτρου. Η λογική της μεθόδου βασίζεται στην εκτέλεση της $kh \times kw$ συνέλιξης ως ένα σύνολο 1×1 συνελίξεων, ώστε να εκμεταλλευτούμε το γεγονός οτί στις τελευταίες δεν παρατηρείται κάποια επανάληψη δεδομενων. Έτσι, μειώνουμε αρκετά το μέγεθος της επιπρόσθετης μνήμης που προκύπτει στις GEMM υλοποιήσεις. Αν παρατηρήσουμε προσεκτικά την τροποποιημένη δομή του Φίλτρου στο Σχήμα 4.14, θα συνειδητοποιήσουμε ότι σε αυτήν τα στοιχεία παρατάσσονται με τέτοιο τρόπο ώστε να συμβάλλουν στην εκτέλεση της 1×1 Συνελίξης, δηλαδή θα μπορούσαμε να πούμε ότι η δομή είναι χωρισμένη σε kh * kw τον αριθμό, $n \times c$ τμήματα, το κάθε ένα από τα οποία φιλοξενεί όλα τα στοιχεία [:,:,i,j] δηλαδή όλα τα στοιχεία μιας συγκεκριμένης χωρικής θέσης της αρχικής δομής του Φίλτρου. Συνεπώς, παρόλο που η εκτέλεση του πολλαπλασιασμού θα γίνει σε ένα στάδιο, η εκτελέση χωρίζεται λογικά σε kh * kw πολλαπλασιασμούς, δηλαδή σε kh * kw 1×1 Συνελίξεις.

GEMM Εκτέλεση

Η εκτέλεση του GEMM πολλαπλασιασμού γίνεται σε ένα στάδιο, όπως φαίνεται και στο Σχήμα 4.14. Ο πολλαπλασιασμός που προκύπτει είναι ο εξης:

$$\hat{R} : \hat{K} \times D \tag{4.32}$$

, δηλαδή θα έχει διαστάσεις **για stride = 1, όπου oh = h και ow = w**:

$$\hat{R} : ((kh * kw) * n \times c) * (c \times h * w) = ((kh * kw) * n \times h * w) \tag{4.33}$$

Lifting

Το στάδιο του Lifting έχει την ευθύνη και εδώ να παράξει την τελικά δομή εξόδου, προσθέτοντας όλα τα αποτελέσματα των 1×1 Συνελίξεων. Συγκεκριμένα, πρόκειται για μία μετατοπισμένη πρόσθεση αφού κάθε ένα από τα kh * kw τον αριθμό, $oh * ow \times n$ την διάσταση τμήματα, δηλαδή κάθε αποτέλεσμα της 1×1 Συνέλιξης, προστίθεται αρχικά με ένα offset, το οποίο είναι ανάλογο με την χωρική θέση του στοιχείου του φίλτρου από τον οποίο συνελίχθει. Πιο απλά, κάθε ένα από τα $oh * ow \times n$ τμήματα

αρχικά προστίθεται σε ένα offset:

$$\text{Lifting offset} = i * kw + j, \\ \text{, όπου } i \in 0, ..., kh - 1 \text{ και } j \in 0, ..., kw - 1 \tag{4.34}$$

Στην συνέχεια, το κάθε μετατοπισμένο τμήμα προστίθεται στην δομή εξόδου ώστε να προκύψει, έτσι, η τελευταία. Το τελευταίο άθροισμα γίνεται φύσικα μόνο για τις τιμές για τις οποιές ισχύει:

$$R_l[i,j] + offset_l <= (oh * ow * n), \\ \text{, όπου } R_l \text{ το εκάστοτε τμήμα, και i, j οι δείκτες που το διατρέχουν} \tag{4.35}$$

Μετά το πέρας αυτής της διαδικασίας θα έχουμε εκτελέσει επιτυχώς την Συνέλιξη.

4.2.6 Μέθοδος 5: Ker2row-ACC

Η συγκεκριμένη μέθοδος αποτελεί βελτίωση της προηγούμενης και οδηγεί σε υλοποίηση της Συνέλιξης με πολύ λιγότερη χρήση επιπρόσθετης μνήμης. Φυσικά προέρχεται από την ίδια πηγή από την οποία προήλθε και η προηγούμενη: [60].

Περιγραφή της Μεθόδου Για να επιτύχει την μείωση της επιπρόσθετης μνήμης που κάνει χρήση η μέθοδος ώστε να υλοποιήσει την συνέλιξη μοιράζει την $kh \times kw$ Συνέλιξη σε 1×1 συνελίξεις, όπως ακριβώς έκανε και η προηγούμενη μέθοδο με την διαφορά ότι εκεί ο διαχωρισμός αυτός ήταν νοητός, ενώ τώρα έχει πραγματική σημασία. Αναλυτικότερα, εδώ καλούμε την GEMM συνάρτηση kh * kw φορές, ώστε να υλοποιήσει κάθε φορά την 1×1 συνέλιξη. Με τον τρόπον αυτόν μειώνουμε σε μεγάλο βαθμό το μέγεθος της απαιτούμενης μνήμης καθώς από εκεί που χρησιμοποιούσαμε $((kh * kw) * n \times h * w$ κατά το Lifting, τώρα χρειαζόμαστε μόνο έναν απομονωτή (buffer) διαστάσεων $n \times h * w$ για το Lifting, οι οποίες θα είναι οι είσοδοι και έξοδοι της GEMM συνάρτησης. Συγκεκριμένα, για κάθε τμήμα $n \times c$ που μετατρέπαμε στην παραπάνω μέθοδο την αποθηκεύουμε κάθε φορά στον απομονωτή εισόδου, αυτό πολλαπλασιάζεται με την δομή εισόδου μέσω της GEMM συνάρτησης, το αποτέλεσμα της αποθηκεύεται στον απομονωτή εξόδου και αυτός με το κατάλληλο offset που χρησιμοποιούνταν και πριν, αποθηκεύεται στην δομή εξόδου. Στην συνέχεια, ακόλουθεί το επόμενο $n \times c$ τμήμα, ώσπου η διαδικασία να ολοκληρωθεί για όλα τα τμήματα και να προκύψει το τελικό Blob εισόδου. Η παραπάνω διαδικασία που περιγράψαμε απεικονίζεται και στο Σχήμα 4.15.

Τονίζεται ότι στις Ker2row μεθόδους οι μαθηματικές σχέσεις που εμφανίζονται ισχύουν για: stride = 1 και dilation = 1. Για διαφορετικές τιμές οι σχέσεις εμφανίζουν ορισμένες αλλαγές οι οποίες δεν περιγράφονται εδώ χάριν της απλότητας.

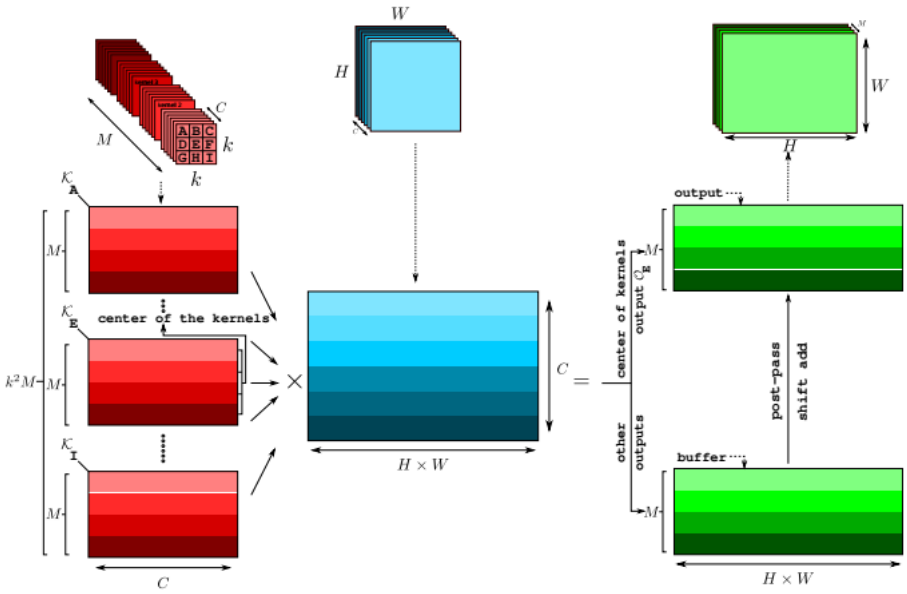


Fig. 8: Kernel to Row ($kn2row$) – Solving the "tube of toothpaste" problem from $kn2row - nacc$

Σχήμα 4.15: *Η μέθοδος Ker2row-ACC.*

4.2.7 Σύνοψη GEMM υλοποιήσεων

Παρακάτω ακολουθεί ένας συνοπτικός πίνακας με κάποια από τα χαρακτηριστικά της κάθε GEMM μεθόδου:

	Layers				
Feature	Expensive Lowering	Expensive Lifting	Balanced	Ker2row-NACC	Ker2row-ACC
Flops	$2*kh*kw*c*oh*ow*o$	$2*w*h*c*o*kh*kw$	$2*H*ow*c*kh*o*kw$	$2*c*h*w*kh*kw*o$	$2*c*h*w*o*kh*kw$
$\hat{D}$ Dimensions	$kh*kw*c \times oh*ow$	$w*h \times c$	$(H*ow) \times (c*kh)$	$c \times h*w$	$c \times h*w$
$\hat{K}$ Dimensions	$o \times kh*kw*c$	$c \times o*kh*kw$	$(c*kh) \times (o*kw)$	$(kh*kw)*o \times c$	$o \times c*$
$\hat{R}$ Dimensions	$o \times oh*ow$	$w*h \times o*kh*kw$	$(H*ow) \times (o*kw)$	$(kh*kw)*o \times h*w$	$o \times h*w$
Additional Memory	$oh*ow \times kh*kw*c$	$h*w \times kh*kw*o$	$h*w \times kh*(o+c)$	$h*w \times kh*kw*o$	$h*w \times o$

* Πρόκειται για (kh * kw) μικρούς απομονωτές διαστάσεων $o \times c$.

4.3 Απευθείας (Direct) Υλοποιήσεις

4.3.1 Εισαγωγή

Σύμφωνα με όλα τα παραπάνω παρατηρούμε ότι, ενώ οι GEMM υλοποιήσεις είναι αρκετά αποδοτικές, προκαλούν την αύξηση της απαιτούμενης μνήμης καθώς χρειάζονται την δημιουργία δυσδιάστατων δομών διαφορετικών από τις υπάρχουσες. Η δυνατότητα για αυτήν την αύξηση της απαιτούμενης μνήμης αποτελεί πολυτέλεια που δεν υπάρχει σε όλα τα συστήματα. Αντιθέτως, υπάρχουν συστήματα και συσκευές, όπως τα Ενσωματωμένα συστήματα (Embedded Systems) στα οποία οι αποθηκευτικές ικανότητες είναι πολύ μικρές. Για αυτόν τον λόγο, αποτελεί μεγάλη ανάγκη η διερεύνηση των δυνατοτήτων μεθόδων που εκτελούν Απευθείας Συνέλιξη (Direct Convolution), παρόλο το γεγονός ότι γνωρίζουμε εκ των προτέρων ότι η επιδόσεις τους υπολείπονται πολύ των GEMM υλοποιήσεων.

Η Απευθείας Συνέλιξη είναι μία πολύ κοπιαστική και περίπλοκη διαδικασία. Αποτε-

λείται από μία σειρά 6 εμφωλευμένων επαναληπτικών βρόχων (loops), οι οποιές κάνουν οποιοδήποτε έργο βελτίωσης ή και παραλληλισμού των βρόχων πολύ δύσκολο. Είναι χαρακτηριστικό ότι κάποιες από τις μεθόδους που θα περιγράψουμε παρακάτω εμφανίζουν πολύ άσχημες επιδόσεις, αλλά τις αναλύουμε γιατί θεωρούμε χρήσιμο να κατανοήσουμε τους λόγους για τους οποίους αυτές αποτυγχάνουν. Σε κάθε περίπτωση η βασική εργασία που προσπαθούμε να εκτελέσουμε είναι η παραλληλοποίηση των εξωτερικών κάθε φορά βρόχων.
Τονίζεται ότι στους αλγορίθμους που περιγράφονται παρακάτω έχουμε απλοποιήσει τις εντολές, θέτοντας τις παραμέτρους stride = 1, dilation = 1 και pad = 0.

4.3.2 Μέθοδος 1: Βασισμένη στην Είσοδο (Input Based)

Η μέθοδος αυτή ονομάζεται Input Based για τον λόγο ότι βασιζόμαστε στην δομή Εισόδου ώστε να βρούμε τον τρόπο υλοποίησης της Συνέλιξης. Συγκεκριμένα, ακολουθούμε την δομή της Εισόδου, παίρνοντας κάθε φορά ένα στοιχείο αυτής και κάνοντας όλες τις πράξεις που αυτό συμμετέχει. Αυτό πρακτικά σημαίνει ότι οι πιο εξωτερικοί βρόχοι θα είναι αυτοί της δομής Εισόδου, όπως φαίνεται και στον παρακάτω αλγόριθμο.
Τονίζεται ότι η συγκεκριμένη μέθοδος απαιτεί την ύπαρξη συγχρονισμού στις προ-

Αλγόριθμος 4.1: *Input Based*

```
1:  for c = 0 to 1...InputChannels do
2:      for oh = 0 to 1...OutputRows do
3:          for kh = 0 to 1...KernelRows do
4:              for ow = 0 to 1...OutputColumns do
5:                  for kw = 0 to 1...KernelColumns do
6:                      for n = 0 to 1...OutputChannels do
7:                          Output[n, oh, ow] += Input[c, oh+kh, ow+kw]*Filter[n, c, kh, kw]
8:                      end for
9:                  end for
10:             end for
11:         end for
12:     end for
13: end for
```

σβάσεις στην δομή της Εξόδου, κάτι που αποτελεί πολύ μεγάλο εμπόδιο στην παραλληλοποίηση της μεθόδου.

4.3.3 Μέθοδος 2: Βασισμένη στην Έξοδο (Output Based)

Η λογική σε αυτή την μέθοδο ομοιάζει με αυτήν της προηγούμενης μεθόδου με την διαφορά ότι τώρα έχουμε σαν οδηγό την δομή εξόδου, δηλαδή παίρνουμε το κάθε στοιχείο αυτής και εκτελούμε κάθε πράξη που αυτό συμμετέχει. Πρακτικά, εδώ, οι εξωτερικοί βρόχοι είναι αυτοί της δομής εξόδου, κάτι το οποίο είναι εμφανές και στον αλγόριθμο που ακολουθεί:

Αλγόριθμος 4.2: *Output Based*

```
1: for n = 0 to 1...OutputChannels do
2:     for oh = 0 to 1...OutputRows do
3:         for ow = 0 to 1...OutputColumns do
4:             if n % 2 = 0  then
5:                 for c = 0 to 1...InputChannels do
6:                     for kh = 0 to 1...KernelRows do
7:                         for kw = 0 to 1...KernelColumns do
8:                             Output[n,oh,ow] += Input[c,oh+kh,ow+kw]*Filter[n,c,kh,kw]
9:                         end for
10:                    end for
11:                end for
12:            else
13:                for c = 0 to InputChannels...1 do
14:                    for kh = 0 to 1...KernelRows do
15:                        for kw = 0 to 1...KernelColumns do
16:                            Output[n,oh,ow] += Input[c,oh+kh,ow+kw]*Filter[n,c,kh,kw]
17:                        end for
18:                    end for
19:                end for
20:            end if
21:        end for
22:    end for
23: end for
```

Όπως είναι εμφανές και από τον αλγόριθμο γίνεται ένας διαχωρισμός στην υλοποίηση της μεθόδου καθώς για άρτιο αριθμό Φίλτρου η υλοποίηση πραγματοποιείται κανονίκα, ενώ για περιττό υλοποιείται ανάποδα στο επίπεδο του Βάθους εισόδου. Αυτό έγινε για να μην υπάρχουν τυχών ανταγωνισμός στις προσβάσεις των δομών από νήματα που παραλληλοποιούν την υλοποίηση στον διάσταση των n.

4.3.4 Μέθοδος 3: Βασισμένη στο Φίλτρο (Filter Based)

Η μέθοδος αυτή είναι βασισμένη στην δομή του Φίλτρου, καθώς η λογική που εφαρμόζεται ξεκινά από κάθε στοιχείο του Φίλτρου και εκτελεί οποιαδήποτε πράξη αυτό συμμετέχει. Φυσικά, εδώ πρακτικά εξωτερικοί βρόχοι αποτελούν αυτοί της δομής του Φίλτρου και ο αλγόριθμος που υλοποιείται:

4.3.5 Μέθοδος 4: Βελτιστοποιημένη Υλοποίηση Βασισμένη στο Εξόδο (Cache Blocking Output Based)

Η μέθοδος αυτή προσπαθεί βελτιώσει την επίδοση της προηγούμενης Output Based μεθόδου με την χρήση εναλλακτικών τεχνικών από τον παραλληλισμό και τις υπόλοιπες που χρησιμοποιήθηκαν προηγούμενα. Αναλυτικότερα, γίνεται μία προσπάθεια

Αλγόριθμος 4.3: *Filter Based*

```
1:  for n = 0 to 1...OutputChannels do
2:      for c = 0 to 1...InputChannels do
3:          for kh = 0 to 1...KernelRows do
4:              for kw = 0 to 1...KernelColumns do
5:                  for oh = 0 to 1...OutputRows do
6:                      for ow = 0 to 1...OutputColumns do
7:                          Output[n, oh, ow] += Input[c, oh+kh, ow+kw]*Filter[n, c, kh, kw]
8:                      end for
9:                  end for
10:             end for
11:         end for
12:     end for
13: end for
```

να επιλυθούν κάποιες βελτιστοποιήσεις σε θέματα προσβάσεων στην Κρυφή Μνήμη (Cache) του Συστήματος. Έτσι, προσπαθούμε να μειώσουμε τον αριθμό των μεταφορών από την κύρια μνήμη στην Cache μοιράζοντας τον υπολογισμό του κάθε στοιχείου της δομής εξόδου σε μικρότερα τμήματα. Τα τμήματα αυτά τα ονομάζουμε **Chunks**. Έτσι, για παράδειγμα αν έχουμε 3 chunks, τότε θα υπολογιστεί κάθε ένα από τα (oh * ow) στοιχεία της δομής εξόδου, αρχικά εκτελώντας πρώτα τις πράξεις που αφορούν το ένα τρίτο (1/3) των δομών εισόδου και φίλτρου, μετά το δεύτερο και τέλος το τρίτο. Με αυτόν τον τρόπο εκμεταλλευόμαστε σε μεγάλο βαθμό τα στοιχεία που είναι ήδη μέσα στην Cache, προσπαθώντας να εκτελέσουμε όσο το δυνατό περισσότερες πράξεις στις οποίες αυτά συμμετέχουν.

Ένας απλοποιημένος αλγόριθμος της συγκεκριμένης μεθόδου, όπου θεωρείται ότι ο αριθμός των Chunks είναι ακέραιος διαιρέτης του Βάθους Εισόδου, περιγράφεται παρακάτω: Τονίζεται ότι στον παραπάνω αλγόριθμο θεωρούμε ότι ο αριθμός των chunks είναι ακέραιος διαιρέτης των καναλιών εισόδου (C). Αυτό όμως δεν ισχύει πάντα και για αυτόν τον λόγο θα πρέπει να λάμβουμε υπόψην αυτές τις περιπτώσεις και να προσαρμόσουμε το τελευταίο chunk ώστε να εκτελεί το υπόλοιπο των δεδομένων. Ακόμα παρατηρούμε ότι ο παραπάνω αλγόριθμος ομοιάζει περισσότερο με τον Filter Based, όμως δεν τον ονομάζουμε ως Output Based μιας και η λογική του αλγορίθμου επικεντρώνεται στην κατανομή του φόρτου εργασίας της παραγωγής της εξόδου στο επίπεδο των καναλιών εισόδου.

Αλγόριθμος 4.4: *Cache Blocking Output Based*

```
1: NumOfThreads = Αριθμός των διαθέσιμων Νημάτων
2: Tid = Αριθμός που υποδηλώνει την ταυτότητα του Νήματος
3: ChunkSize = InputChannels/Chunks
4: OutChannels = OutputChannels/NumOfThreads
5: for ch = 0 to 1...Chunks do
6:     for  = 0 to 1...OutChannels do
7:         for c = 0 to 1...ChunkSize do
8:             for kh = 0 to 1...KernelRows do
9:                 for kw = 0 to 1...KernelColumns do
10:                    for oh = 0 to 1...OutputRows do
11:                        for ow = 0 to 1...OutputColumns do
12:                            Output[o+Tid*OutChannels,oh,ow] +=
                               Input[c,oh+kh,ow+kw]*Filter[o+Tid*OutChannels,c,kh,kw]
13:                        end for
14:                    end for
15:                end for
16:            end for
17:        end for
18:    end for
19: end for
```

Κεφάλαιο 5

Αξιολόγηση

Στο κεφάλαιο αυτό πραγματοποιείται μία εκτενής αξιολόγηση των μεθόδων που περιγράψαμε στο κεφάλαιο "Υλοποίηση" προσπαθώντας να εξάγουμε συμπεράσματα λαμβάνοντας μετρήσεις υπό πολλές οπτικές γωνίες. Υπό αυτό το σκεπτικό, θα περιγράψουμε αρχικά τις αρχιτεκτονικές υλικού πάνω στις οποίες πραγματοποιήθηκαν οι μετρήσεις μας δίνοντας έμφαση στις παραμέτρους που μπορούν να μεταβάλλουν την επίδοση των μεθόδων. Στην συνέχεια, θα περιγράψουμε το είδος των μετρήσεων που λάβαμε, αλλά και κάποιες παραδοχές που έπρεπε να ληφθούν καθώς και τις τιμές των παραμέτρων της αρχιτεκτονικής που μας οδηγούν στα καλύτερα αποτελέσματα. Καταλήγοντας θα ασχοληθούμε με την αξιολόγηση των μεθόδων παρουσιάζοντας μέσω γραφικών παραστάσεων τα αποτελέσματα των μετρήσεων και προσπαθώντας να προβάλουμε τα αίτια της εκάστοτε συμπεριφοράς που παρουσιάζεται σε αυτές.

5.1 Ανάλυση - περιγραφή αρχιτεκτονικής

Όπως αναφέρθει και προηγουμένως η συγκεκριμένη μελέτη πραγματοποιήθηκε πάνω σε δύο αρχιτεκτονικές:

- **Intel Haswell.** Ο πειραματισμός πάνω στην συγκεκριμένη αρχιτεκτονική κατέστη δυνατός μέσω του εργαστηρίου (cslab), και αποτελείται από τα χαρακτηριστικά που περιγράφονται στον πίνακα 5.1. Σε αυτόν τον πίνακα παρατηρούμε ότι πρόκειται για έναν επεξεργαστή [61] με υψηλή συχνότητα ρολογιού (2.60GHz) και δυνατότητα παράλληλης εκτέλεσης μέχρι και 56 νημάτων χάρης των 28 πυρήνων που διαθέτει αλλά και της υπερνημάτωσης που δίνει την δυνατότητα σε κάθε πυρήνα να εκτελέσει 2 νήματα παράλληλα.

- **Intel Xeon Phi Knights Landing (KnL7250) [23].** Ο επεξεργαστής αυτός αποτελεί ένα από τα πιο ενδιαφέροντα Projects της Intel και αποτελεί την ναυαρχίδα στην προσπάθειά της να ασχοληθεί με τα πολυπύρηνα συστήματα. Παρότι η σειρά Intel Xeon Phi αποτελείται κυρίως από Coprocessors, ο συγκεκριμένος είναι από τους πρώτους που μπορεί να καλύψει και λειτουργίες των συμβατικών Processors με αξιοσημείωτα αποτελέσματα. Αναλυτικότερα, τα χαρακτηριστικά του Intel Xeon Phi Knl7250 εμφανίζονται στον πίνακα 5.2:

Πίνακας 5.1: *Πίνακας Intel Haswell Αρχιτεκτονικής*

Χαρακτηριστικά Πολυπύρηνης Αρχιτεκτονικής	
Όνομα Επεξεργαστή	Intel Xeon CPU E5-2697
Πυρήνες	28
Sockets	2
Νήματα ανά Πυρήνα	2
Ταχύτητα Επεξεργαστή	2.60GHz
L1 Cache	32K
L2 Cache	256K
L3 Cache	35.84M
NUMA nodes	2
NUMA node 0 CPUs	0-13,28-41
NUMA node 1 CPUs	14-27, 42-55

Πίνακας 5.2: *Πίνακας Χαρακτηριστικών Intel Xeon Phi Knl7250*

Χαρακτηριστικά Knl7250	
Όνομα Επεξεργαστή	Intel Xeon Phi Knl7250
Πυρήνες	68
Sockets	1
Νήματα ανά Πυρήνα	4
Ταχύτητα Επεξεργαστή	1.40GHz
L2 Cache	34MB
NUMA nodes	2
Ιδιαιτερότητα Μνήμης	**Μνήμη Υψηλού Εύρους 16GB (HBM) σαν MCDRAM**
Μέγεθος DDR4 Μνήμης	384GB

Παρατηρούμε ότι ο Intel Xeon Phi KnL7250 μπορεί να διαχειριστεί προβλήματα μεγάλης παραλληλισιμότητας καθώς μπορεί να εκτελέσει μέχρι και **68 * 4 = 272** νήματα παράλληλα. Ο αριθμός αυτός είναι συγκρίσιμος τόσο με έναν Coprocessor όσο και με μία κάρτα Γραφικών Γενικού Σκοπού (GPGPU). Βέβαια οι πυρήνες που διαθέτει δεν είναι αρκετά περίπλοκοι και "δυνατοί", όπως είναι εμφανές και από την συχνότητα τους, αλλά είναι σε θέση να φέρουν εις πέρας αρκετά αποτελεσματικά και σειριακές διεργασίες.

Εκτός των άλλων πρέπει να δωθεί ιδιαίτερη προσοχή στο τελευταίο χαρακτηριστικό [62]. Η **Hιgh Bandwidth Memory(HBM)** αποτελεί ένα μοναδικό χαρακτηριστικό της συγκεκριμένης σειράς επεξεργαστών. Ιδιαίτερα σε μια σειρά από προγράμματα-εφαρμογές (APIs) τα οποία αποτελούνται από αλγόριθμους με εντολές με πολλές μεταβλητές κινητής υποδιαστολής (floating point operations) σε κάθε πρόσβαση στην μνήμη., οι οποίες προκαλούν μεγάλη καθυστέρηση στην υλοποίηση (low arithmetic intensity). Παραδείγματα τέτοιων αλγορίθμων αποτελούν ο Γρήγορος Μετασχηματισμός Fourier (Fast Fourier Transformation) [63], και ο Πολλαπλασιασμός Πινάκων (Matrix Multiplication), με τον δεύτερο να μας ενδιαφέρει πολύ και στην παρούσα εργασία. Όπως είναι κατανοητό σε τέτοιου είδους εφαρμογές απαιτούνται οι προσβάσεις στην μνήμη να είναι όσο τον δυνατόν πιο γρήγορες και μικρές σε αριθμό, και αυτό είναι που μπορεί να μας προσφέρει η HBM. Έτσι ο KNL7250 προσφέρει **μία HBM μεγέθους 16GB και 5x μεγαλύτερης επίδοσης από την 384GB DDR4 μνήμη** που επίσης χρησιμοποιεί σαν κύρια μνήμη. Επιπρόσθετα υπάρχει και η δυνατότητα επιλογής του πως θα χρησιμοποιηθεί η HBM. Συγκεκριμένα, μπορεί να γίνει χρήση της HBM σε:

- **Flat Mode**, όπου στην ουσία χρησιμοποιείται σαν γενική μνήμη, μοιράζεται τον χώρο διευθύνσεων της DDR4, είναι L2 cached και βρίσκεται σε NUMA node χωρίς κάποιον επεξεργαστή σε αυτό, σε
- **Cache Mode**, όπου χρησιμοποιείται σαν ένα επιπλέον επίπεδο cache, και σε
- **Hybrid Mode**, όπου και ένα τμήμα της HBM χρησιμοποιείται σε Cache mode και το υπόλοιπο σε Float mode.

Κάθε επιλογή παρουσιάζει τα αντίστοιχα πλεονεκτήματα και μειονεκτήματα, έτσι:

- Όταν γίνεται χρήση της HBM σε **Float Mode** ο προγραμματιστής έχει πολύ μεγαλύτερο έλεγχο και με τον κατάλληλο συντονισμό των προσβάσεων στην μνήμη μπορεί να οδηγήσει στην μέγιστη επίδοση, ωστόσο αυτή η επιλογή αυξάνει σε μεγάλο βαθμό την πολυπλοκότητα και το φόρτο εργασίας του προγραμματιστή.
- Αντίθετα όταν η HBM χρησιμοποιείται σε **Cache Mode**, τα πράγματα για τον προγραμματιστή είναι σαφώς πιο απλά, καθώς δεν θα χρειαστέι να επέμβει καθόλου στις προσβάσεις στην μνήμη, αλλά σε προβλήματα που παρουσιάζουν μεγάλα ποσοστά αποτυχιών (misses) στην cache, αυξάνεται σε μεγάλο βαθμό το κόστος πρόσβασης (Access Latency).

– Τέλος, η **Hybrid** χρήση της HBM συνδιάζει τα πλεονεκτήματα των δύο προηγουμένων επιλογών, όμως μειώνει σε μεγάλο βαθμό το μέγεθος και στις δύο.

5.2 Περιγραφή Μετρήσεων

Οι αρχιτεκτονικές που περιγράψαμε παραπάνω μας προσφέρουν την δυνατότητα να εκτελέσουμε πολλαπλών ειδών μετρήσεις αλλά και να λάβουμε σημαντικά συμπεράσματα πάνω σε αυτές. Σε αυτό το σημείο θα περιγράψουμε αναλύτικα τις μετρήσεις που λάβαμε καθώς και τις όλες τις παραμέτρους που διαφοροποιήσαμε ώστε να αξιοποιήσουμε το σύνολο των δυνατοτήτων που μας προσφέρουν οι διαθέσιμοι επεξεργαστές. Έτσι πήραμε τις εξής μετρήσεις:

- **Μετρήσεις για Διαφορετικό Πλήθος Νημάτων - Προδιαγραφές Υλικού (Hardware Specifications)**:
  Η βάση των μετρήσεων μας σε κάθε παράμετρο ήταν η μεταβολή του πλήθους των νημάτων που καθιστούσαμε διαθέσιμα σε κάθε εκτέλεση. Φυσικά εκτός την αλλαγή του αριθμού των νημάτων ορίζουμε και τους πυρήνες του επεξεργαστή που θα αναλάμβουν την εκτέλεση κάθε νήματος. Αυτό είναι δυνατό μέσω της ρύθμισης δύο παραμέτρων : της Συγγένειας (**Affinity**) και του Υλικού Υποσυνόλου (**Hardware SubSet**) στον Knl7250 [64] και μέσω μόνο της παραμέτρου **Affinity** στον E5 [65]. Ο τρόπος που ρυθμίζουμε τις παραμέτρους αυτές αποσκοπεί στο να αποφύγουμε όσον το δυνατό την χρήση της υπερνημάτωσης **(**HyperThreading). Συγκεκριμένα, όταν ο αριθμός των νημάτων είναι μικρότερος από τον αντίστοιχο των πυρήνων μοιράζουμε το κάθε νήμα σε έναν πυρήνα, και μόνο όταν δεν μπορούμε να το αποφύγουμε, όταν, δηλαδή, ο αριθμός των νημάτων γίνει μεγαλύτερος από τους πυρήνες τότε αναθέτουμε σε κάθε πυρήνα περισσότερα από ένα νήματα. Ο λόγος που το επιδιώκουμε αυτό, έγκειται στο γεγονός ότι ο Υπερνηματισμός δίνει την εντύπωση στον επεξεργαστή ότι διαθέτει περισσότερους πυρήνες από ότι στην πραγματικότητα, ώστε να αυξηθεί η ικανότητα παραλληλισμού. Έτσι, όταν ένας πυρήνας αναλαμβάνει περισσότερα από ένα νήματα τότε δημιουργούνται καταστάσεις ανταγωνισμόυ σε κάποιους μοιραζόμενους πόρους όπως οι Μονάδες Κινητής Υποδιαστολής (Floating Point Units) με αποτέλεσμα η επίδοση να είναι χαμηλότερη από εκείνη που θα είχε το σύστημα αν κάθε νήμα είχε όλους τους πόρους διαθέσιμους, αν, δηλαδή, κάθε πυρήνας αναλάμβανε ένα μόνο νήμα.

- **Μετρήσεις για την Επιλογή Node**:
  Το συγκεκριμένο είδος μέτρησεων αφορά μόνον τον Knl7250, όπου εκεί μπορούμε να χρησιμοποιήσουμε την Μνήμη Υψηλού Εύρους Ζώνης (High Bandwidth Memory) τόσο σαν Cache (node=1) όσο και κύρια μνήμη σαν συνέχεια της υπάρχουσας DDR4 (node=0). Στον E5 θέτουμε node=0 ώστε να αποφύγουμε την χρήση και των δύο sockets του επεξεργαστή, γεγονός που θα είχε αρνητικές συνέπειες στην επίδοση μιας και ελαττώνεται σε μεγάλο βαθμό η τοπικότητα (locality).

- **Μετρήσεις για Διαφορετικά Δίκτυα**:
  Όπως αναλύσαμε και στο κεφάλαιο "Εργαλεία" λαμβάνουμε μετρήσεις για επτά Δίκτυα:

  - AlexNet [45]
  - GoogLeNet [46]
  - CaffeNet [47]
  - VGG-16 [48]
  - ResNet-50 [49]
  - SqueezeNet v1.0 [50]
  - MobileNet [51]

- **Μετρήσεις για τις Διαφορετικές Τεχνικές**:
  Όπως περιγράψαμε και στο προηγούμενο κεφάλαιο "Υλοποίηση" λάμβαμε μετρήσεις για τις εξής τεχνικές GEMM υλοποιήσεων:

  - Expensive Lowering (Im2col - Προεπιλογή του Caffe)
  - Expensive Lifting
  - Balanced Lowering - Lifting
  - Ker2row Non ACCumulate (NACC)
  - Ker2row ACCumulate (ACC)

  Ταυτόχρονα μετρήσαμε και την επίδοση των Δικτύων και για τις παρακάτω τεχνικές Απευθείας Συνέλιξης (Direct Convolution):

  - Input Based
  - ZigZag Output Based
  - Kernel Based
  - Cache Blocking Output Based

- **Μετρήσεις για τις Διαφορετικές Αρχιτεκτονικές**:
  Μετρήσεις λήφθηκαν και για τις δύο διαθέσιμες Αρχιτεκτονικές που περιγράψαμε παραπάνω, δηλαδή για τον Intel Xeon E5 (πολυπύρηνη Αρχιτεκτονική) [61] και για τον Intel Xeon Phi Knl7250 [23].

Αξίζει να σημειωθεί ότι συνολικά στο πλαίσιο αυτής της εργασίας λήφθηκαν περισσότερες από 300.000 μετρήσεις και πραγματοποιήθηκαν πάνω από 15000 εκτελέσεις Συνελικτικών Δικτύων μέσω του Caffe, μόνο για την λήψη των τελικών μετρήσεων.

5.3 Έλεγχος

Σε αυτό το σημείο θα αναλύσουμε όλες τις απαραίτητες παραδοχές και αποφάσεις που πάρθηκαν ώστε να έχουμε τις καλύτερες δυνατές μετρήσεις αλλά και να καταλήξουμε σε ορθά συμπεράσματα μεσω της διαδικασίας σύγκρισης των μετρήσεων που θα ακολουθήσει. Αναλυτικότερα, οι παραδοχές που πήραμε περιγράφονται παρακάτω:

- **Μέγεθος Δέσμων Εικόνας (Images' Batch)**:
  Σε όλες τις μετρήσεις χρησιμοποίησαμε ως τέταρτη διάσταση στα Input Blobs ίση με 1, δηλαδή δοκιμάζαμε μία εικόνα ανά εκτέλεση και όχι δέσμη εικόνων (Batch). Ο λόγος που δεν έγινε κάποια δοκιμή με μεγαλύτερη είσοδο εικόνων ήταν το γεγονός ότι δεν υλοποιούμε εκπαίδευση του δικτύου αλλά απλή εκτέλεση (Testing) και σε αυτήν την διαδικασία το μέγεθος της δέσμης εικόνων που εκτελούμε δεν παίζει τόσο σπουδαίο ρόλο.

- **Αριθμός Επαναλήψεων (Run Loops)**:
  Για να είμαστε απόλυτα σίγουροι για την ακρίβεια των μετρήσεων μας εκτελέσαμε κάθε μέτρηση σε 10 επαναλήψεις και λάμβαμε σαν τελική τιμή τον μέσο όρο όλων των μετρήσεων από κάθε επανάληψη.

- **Προδιαγραφές Μετρήσεων GEMM Υλοποιήσεων**:
  Έχουμε αναφέρει και προηγουμένως ότι το ίδιο το Caffe χρησιμοποιεί σαν προεπιλεγμένη υλοποίηση αυτή του Expensive Lowering (ή Im2col όπως το ίδιο το ονομάζει). Σε αυτήν την κατεύθυνση το Caffe έχει προσαρμόσει όλην την δομή του, έτσι ώστε να συμβάλλει στην αύξηση της επίδοσής του. Για παράδειγμα, η διάταξη των διαστάσεων των Blobs είναι τέτοια ώστε να μην απαιτούνται κάποιες επιπρόσθετες ενέργειες για το Weight Lowering αλλά και το Lifting σύμφωνα με την μέθοδο του Expensive Lowering. Σε όλες τις άλλες τεχνικές όμως η συγκεκριμένη διάταξη δεν βοηθάει και μάλιστα δημιουργεί επιπρόσθετες εργασίες που πρέπει να πραγματοποιηθούν. Σαν συνέπεια οποιαδήποτε σύγκριση μεταξύ των τεχνικών Lowering καθίσταται άδικη. Για να έχουμε συνεπή αποτελέσματα οδηγηθήκαμε στην εξής παραδοχή: **" Οποιαδήποτε ενέργεια μεταβολής της διάταξης των διαστάσεων των Blobs δεν θα καταλογίζεται στις τελικές μετρήσεις."** Σημειώνεται ότι η μεταβολή της διάταξης του Caffe δεν είναι πολύ δύσκολή, όμως ο σχεδιασμός συστήματος εναλλαγής της διάταξης για κάθε τεχνική ξεπερνά τα όρια μίας διπλωματικής εργασίας.

- **Καλύτερη Τεχνική Απευθείας Συνέλιξης (Direct Convolution)**:
  Σύμφωνα με το κεφάλαιο "Υλοποίηση" υλοποιήσαμε τέσσερεις εκδοχές για την εκτέλεση της Απευθείας Συνέλιξης. Σε αυτές οι τρεις μπορούν να θεωρηθούν σαν "Τυπικές" καθώς ο μόνος παραλληλισμός που πραγματοποιείται είναι στο επίπεδο του αριθμού των φίλτρων. Μιλούμε, φυσικά για τις υλοποιήσεις με βάση την δομή Εισόδου, Έξοδο και Φίλτρου (Input Based - Output Based - Filter Based). Από αυτές η Υλοποίηση με βάση το Φίλτρο είναι η πλέον γρήγορη μιας και εμφανίζει καλύτερη τοπικότητα και λιγότερη μεταφορά από την μνήμη, και ως εκ τούτου

θα θεωρείται και ως η Τυπική μας Τεχνική Απευθείας Συνέλιξης. Αυτήν την Τυπική τεχνική προσπαθούμε να την παραλληλοποιήσουμε με την διαμοιρασμό των δεδομένων σε νήματα με βάση κάποια κομμάτια (Blocks) και έτσι δημιουργείται η Cache Blocking Υλοποίηση την οποία από εδώ και έπειτα θα χρησιμοποιούμε στις παρακάτω μετρήσεις για την σύγκριση της Απευθείας Συνέλιξης με τις GEMM υλοποιήσεις.

- **Βέλτιστες Παράμετροι για την Cache Blocking Τεχνική Απευθείας Συνέλιξης**:
  Στην τεχνική αυτή χρησιμοποιείται η μέθοδος του Cache Blocking στο επίπεδο των καναλιών εισόδου (input channels) για τον επιπλέον παραλληλισμό της διαδικασίας. Μετά από αρκετό πειραματισμό καταλήξαμε στην παραδοχή ότι η μέθοδος εμφανίζει βέλτιστη απόδοση για αριθμό Blocks ίσο με 10 για μεγάλες εικόνες, αυτές δηλαδή των δύο πρώτων επιπέδων και ίσο με 8 για τις υπόλοιπες μικρότερες. Επίσης στο πλαίσιο της επίτευξης μεγαλύτερης επαναχρησιμοποίησης των δεδομένων ώστε να αποφευχθούν οι μεταφορές από την μνήμη η υλοποίηση τροποποιείται από αυτήν της κλασικής Output Based και τείνει περισσότερο σε αυτήν της Filter Based, αλλά διατηρεί το όνομα ”Cache Blocking Output Based” μιας και η λογική του Cache Blocking εφαρμόζεται στην δομή Εξόδου.

- **Προδιαγραφές Συγγένειας (Affinity)**:
  Η συγκεκριμένη παραδοχή αναφέρθηκε και προηγουμένως και είναι και εμφανής και στα αποτελέσματα. Πρόκειται για τον τρόπο ανάθεσης των νημάτων στους πυρήνες και προαναφέρθηκε επιζητούμε την αποφυγή χρήσης της Υπερνημάτωσης. Έτσι πήραμε μετρήσεις για δύο τρόπους ανάθεσης στον Knl7250: α) ”compact” ανάθεση, δηλαδή ανάθεση των νημάτων στους πυρήνες με βάση τον τρόπο που αυτοί αριθμούνται, και β) ”scatter” ανάθεση, η οποία ξεκινά μοιράζοντας τα νήματα σε κάθε ένα πυρήνα και αν χρειαστεί χρησιμοποίει και τα υπερνήματα. Στην παρούσα εργασία ελέγξαμε την συμπεριφορά και για τις δύο μεθόδους και καταλήξαμε στο συμπέρασμα που περιμάναμε ότι δηλαδή η ”scatter” ανάθεση είναι αρκετά πιο αποτελεσματική και για αυτό χρησιμοποιήθηκε στην εξαγωγή των παρακάτων αποτελεσμάτων.

- **Προδιαγραφή Επιλογής Node**:
  Η προδιαγραφή αυτή αφορά μόνο την περίπτωση της αρχιτεκτονικής Knl7250. Πρόκειται για την επιλογή της χρήσης της Μνήμης Υψηλού Εύρους Ζώνης (High Bandwidth Memory - HBM) σαν κρυφή μνήμη (Cache) ή σαν κύρια μνήμη σαν συνέχεια της υπάρχουσας DDR4. Εδώ η απόφαση δεν είναι πολύ απλή, αλλά τα αποτελέσματα μας δίνουν μία εικόνα. Όπως περιγράφει και ο Πίνακας 5.3 παρατηρούμε το εξής: για Blobs με μεγάλες χωρικές διαστάσεις, όπως είναι οι εικόνες στα αρχικά επίπεδα του δικτύου, είναι σοφότερο να χρησιμοποιήσουμε την HBM μνήμη σαν κύρια μνήμη, ενώ σε εικόνες με μικρότερες χωρικές διαστάσεις και μεγαλύτερη διάσταση βάθους η χρήση της HBM σαν κρυφή μνήμη αποδίδει καλύτερα. Συνεπώς μιας και εικόνες με μεγαλύτερη την διάσταση του βάθους είναι πλειοψηφία στα συνελικτικά δίκτυα λαμβάνουμε την απόφαση να χρησιμοποιήσουμ την

Πίνακας 5.3: *Περιγραφή της Επίδρασης του Τρόπου Χρήσης της HBM μνήμης στην Επίδοση των Επιπέδων του VGG16*

VGG16 Layer	Input Blob's Dimensions	Χρήση της HBM σαν Cache	Χρήση της HBM σαν Κύρια Μνήμη
L1	224x224x3	0.52 Gflops/s	0.69 Gflops/s
L3	112x112x64	2019 Gflops/s	1470 Gflops/s
L6	56x56x256	1230 Gflops/s	944 Gflops/s
L8	28x28x256	989 Gflop/s	1205 Gflops/s

HBM μνήμη σαν Cache για την εξαγωγή των αποτελεσμάτων μας. Βέβαια επειδή οι υλοποιήσεις του εκάστοτε επιπέδου διαρκούν κάποια εκατοστά του δευτερολέπτου υπάρχουν και φορές που τα παραπάνω δεν επαληθεύονται για άλλα αίτια, όπως το επίπεδο 8 του VGG16 στον Πίνακα 5.3.

5.4 Περιγραφή Αποτελεσμάτων - Αξιολόγηση

Σε αυτό το σημείο θα προσπαθήσουμε να παρουσιάσουμε τα αποτελέσματα των μετρήσεων και να εξηγήσουμε την συμπεριφορά τους. Σε αυτήν την κατεύθυνση θα εμφανίσουμε μια σειρά από γραφήματα που θα μας δώσουν καλύτερη εικόνα για τα αποτελέσματα σε κάθε αρχιτεκτονική. Έτσι:

5.4.1 Intel Xeon Phi Knl7250

Σύγκριση Τεχνικών Lowering

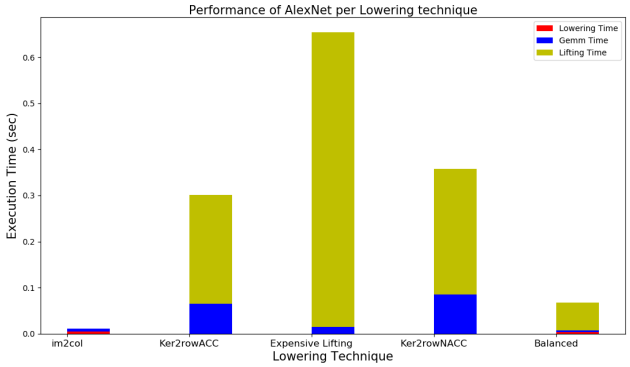
Στην εικόνα 5.1 παρουσιάζεται ο Συνολικός Χρόνος Εκτέλεσης κάθε επιπέδου του δικτύου AlexNet σε δευτερόλεπτα με την μορφή μπαρών. Κάθε μπάρα διαθέτει το πολύ τρία χρώματα ένα για κάθε τμήμα κάθε τεχνικής (Lowering - GEMM - Lifting) και το ύψος κάθε χρώματος υποδηλώνει την συμβολή κάθε τμήματος στον τελικό συνολικό χρόνο του κάθε επιπέδου.

Αν παρατηρήσουμε καλύτερα την Εικόνα 5.1 θα μπορέσουμε να εφαρμόσουμε έναν πρώτο έλεγχο ορθότητας των μετρήσεων μας για κάθε τεχνική. Αναλυτικότερα, στην πρώτη μπάρα απεικονίζεται και στα πέντε γραφήματα η Expensive Lowering Τεχνική που αποτελεί και την προεπιλεγμένη του Caffe. Σε όλα τα γραφήματα η μπάρα αυτή διαθέτει μεγάλο τμήμα Lowering ενώ το τμήμα του Lifting δεν υφίσταται λόγω της συγκεκριμένης διάταξης των διαστάσεων των Blobs, που εξηγήσαμε και νωρίτερα. Αντίθετα η τρίτη μπάρα που απεικονίζει την τεχνική του Expensive Lifting διαθέτει μεγάλο τμήμα του χρόνου στο Lifting ενώ το Lowering τμήμα δεν μετράται καθώς δραστηριοποιείται κυρίως στην αλλαγή της διάταξης των Blobs Εισόδου και Φίλτρου. Επιπρόσθετα η πέμπτη μπάρα, η οποία και παρουσιάζει τους χρόνους της Τεχνικής Balanced Lowering - Lifting είναι μοιρασμένη και στα τρία τμήματα της μεθόδου και μάλιστα με σχετικά ίσα μεγέθη κάτι που επιβεβαιώνει το όνομα της. Όσον αφορά τις υπόλοιπες δύο τεχνικές οι οποίες έχουν διαφορετική λογική από τις προηγούμενες καθώς ασχολούνται κυρίως με το Blob Φίλτρου παρατηρούμε ότι το τμήμα Lowering δεν υφίσταται

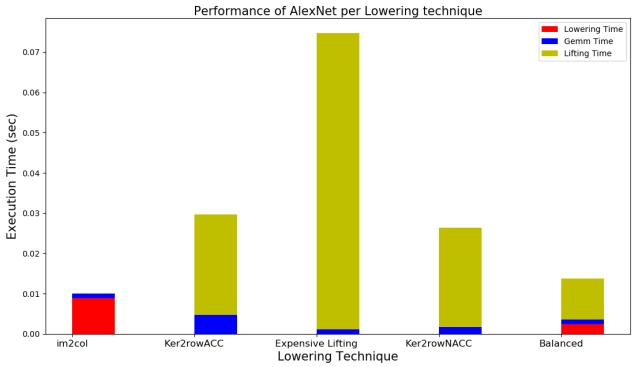
μιας και ασχολείται με αλλαγή της διάταξης του blob φίλτρου. Ακόμα και στις δύο αυτές τεχνικές (Ker2rowNACC και Ker2rowACC) το τμήμα του Lifting είναι πάντα μεγαλύτερο από αυτό της GEMM εκτέλεσης, με την δεύτερη τεχνική (Ker2rowACC) να απαιτεί λιγότερο χρόνο Lifting από την τέταρτη (Ker2rowNACC) κάτι που είναι λογικό και από την υλοποίηση θεωρητικά.

Όσον αφορά τον χρόνο GEMM εκτέλεσης παρατηρούμε ότι το μικρότερο χρόνο απαιτεί η μέθοδος του Expensive Lifting ή οποία και υλοποιεί τους μικρότερους δυνατούς υπολογισμούς μέσω της GEMM συνάρτησης. Αντιθέτως παρατηρούμε ότι, η προεπιλεγμένη του Caffe, η Expensive Lowering απαιτεί τον μεγάλο χρόνο GEMM εκτέλεσης κάτι που οφείλεται στις πολλές επαναλήψεις των στοιχείων των Blobs εισόδου τα οποία οδηγούν στην αύξηση των πολλαπλασιασμών κατα την εκτέλεση της GEMM συνάρτησης. Στην περίπτωση, της τεχνικής Balanced Lowering-Lifting ο GEMM χρόνος που απαιτείται κείται ανάμεσα στα όρια των δύο προηγούμενων τεχνικών μιας και αποτελεί συνδιασμό τους. Επιπρόσθετα, στην τεχνική Ker2rowNACC παρατηρούμε ότι ο GEMM χρόνος είναι αρκετά μεταβαλλόμενος κάτι που οφείλεται στις ιδιότητες της GEMM συνάρτησης, σύμφωνα με τις οποίες συμφέρει οι πίνακες εισόδου να έχουν, αν όχι τετραγωνική, ”μακρόστενη” μορφή για την δομή που προηγείται στον πολλαπλασιασμό, δηλαδή εδώ του Φίλτρου. Τέλος, ο GEMM χρόνος της τεχνικής Ker2rowNACC είναι πάντα μικρότερος του αντίστοιχου της Ker2rowACC, καθώς εκεί εκτελούμε πολλούς πολλαπλασιασμούς μικρότερων πινάκων κάτι που απαιτεί συχνότερη μεταφορά δεδομένων για την εκτέλεση του πολλαπλασιασμού.

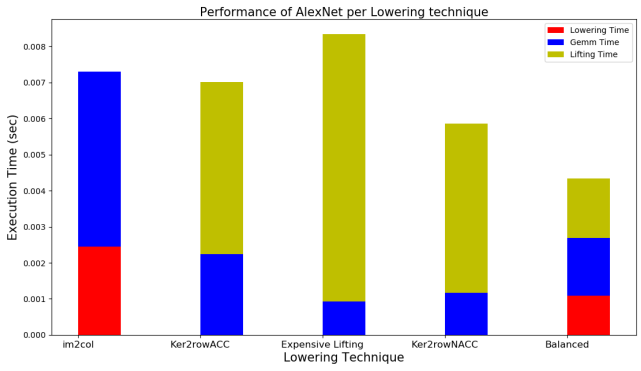
Τέλος, αν δούμε την συμπεριφορά των αποτελεσμάτων με την αλλάγη των επιπέδων μπορούμε να εξάγουμε κάποια πολύ χρήσιμα συμπεράσματα. Συγκεκριμένα, στα δύο πρώτα επίπεδα, όπου οι χωρικές διαστάσεις των BLobs είναι συντριπτικά μεγαλύτερες του Βάθους παρατηρούμε ότι παρουσιάζουν εξαιρετική επίδοση η τεχνικές Expensive Lowering και Balanced Lowering - Lifting, ενώ, αντίθετα, τα αποτελέσματα είναι πολύ άσχημα κυρίως για την Expensive Lifting τεχνική. Αυτό οφείλεται στο γεγονός οτι στην τεχνική αυτή η έξοδος της GEMM συνάρτησης έχει μεγαλύτερες διαστάσεις από οποιαδήποτε άλλη τεχνική, όπως είναι σαφές και από τις διαστάσεις των πολλαπλασιαστικών πινάκων που παρουσιάζονται πάνω από κάθε μπάρα, και για αυτόν τον λόγο ερχόμαστε αντιμέτωποι με την επεξεργασία ενός μεγάλου όγκου δεδομένων κατά του περίπλοκου Lifting, το οποίο αποτελεί και τα κύρια αίτια της καθυστέρησης. Σε αντίθεση, όσο οι χωρικές διαστάσεις των Blobs μειώνονται τόσο και αυξάνονται οι επιδόσεις όλων των τεχνικών σε σημείο που να ξεπερνούν και την επίδοση της προεπιλεγμένης έκδοσης του Caffe. Αναλυτικότερα, πολύ καλή επίδοση παρουσιάζει οι Balanced και Ker2row τεχνικές ενώ αρκετά βελτιωμένη εμφανίζεται η Expensive Lifting τεχνική η οποία πλέον έρχεται αντιμέτωπη με μικρότερες δομές. Στα τρία τελευταία επίπεδα είναι σαφές ότι η Expensive Lowering δεν είναι η καλύτερη επιλογή μας.



(α) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 1 του AlexNet ανά Τεχνική GEMM

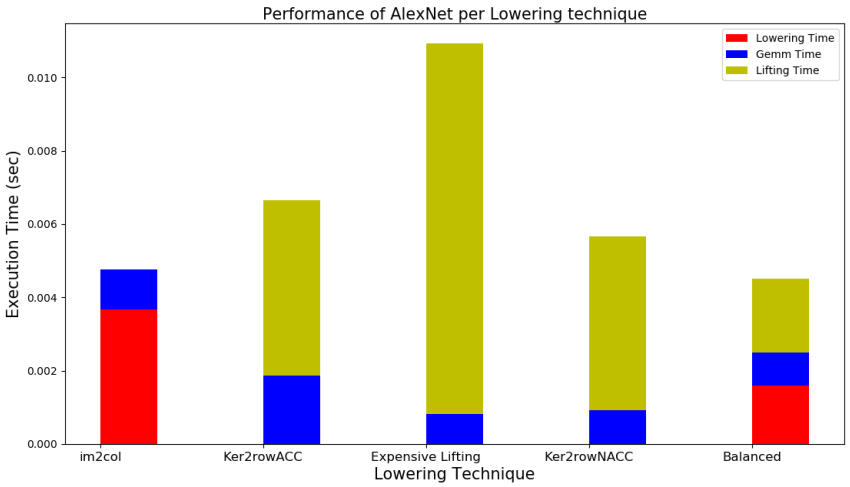


(β) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 2 του AlexNet ανά Τεχνική GEMM

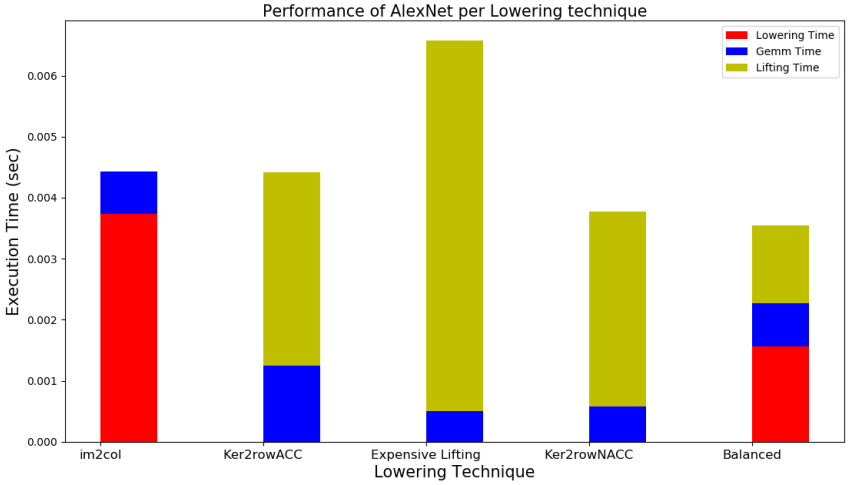


(γ) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 3 του AlexNet ανά Τεχνική GEMM

Σχήμα 5.1: *Παρουσίαση Επίδοσης της Εκτέλεσης για τα Επίπεδα 1,2 και 3 του AlexNet. Σε κάθε γράφημα παρουσιάζεται το τμήμα του χρόνου που αφορά το κάθε τμήμα της τεχνικής (Lowering - GEMM - Lifting) για την αρχιτεκτονική Knl7250*



(δ) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 4 του AlexNet ανά Τεχνική GEMM

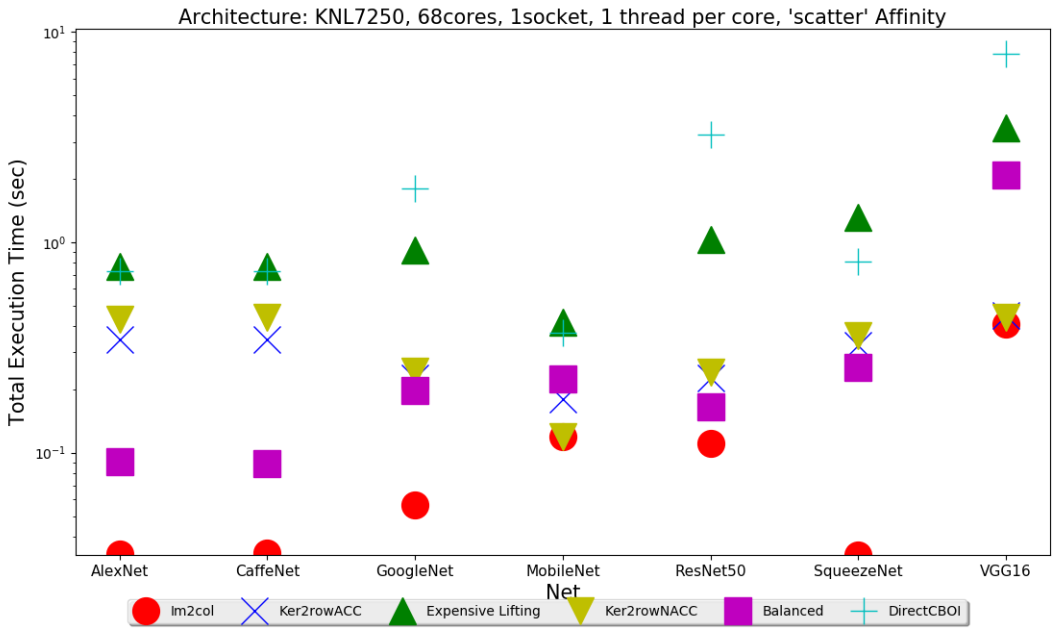


(ε) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 5 του AlexNet ανά Τεχνική GEMM

Σχήμα 5.2: *Παρουσίαση Επίδοσης της Εκτέλεσης για τα Επίπεδα 4 και 5 του AlexNet. Σε κάθε γράφημα παρουσιάζεται το τμήμα του χρόνου που αφορά το κάθε τμήμα της τεχνικής (Lowering - GEMM - Lifting) για την αρχιτεκτονική Knl7250*

Σύγκριση Συνολικού Χρόνου Υλοποίησης ανά Τεχνική και Δίκτυο

Η Εικόνα 5.2 παρουσιάζει τον συνολικό χρόνο εκτέλεσης κάθε δικτύου συναρτήσει κάθε τεχνικής. Σημειώνεται ότι εκτός από τις GEMM τεχνικές έχουμε προσθέσει και την καλύτερη τεχνική Απευθείας Συνέλιξης, ώστε να έχουμε μία εικόνα σύγκρισης.



Σχήμα 5.3: *Γράφημα περιγραφής του Συνολικού Χρόνου Εκτέλεσης για κάθε Δίκτυο και Τεχνική*

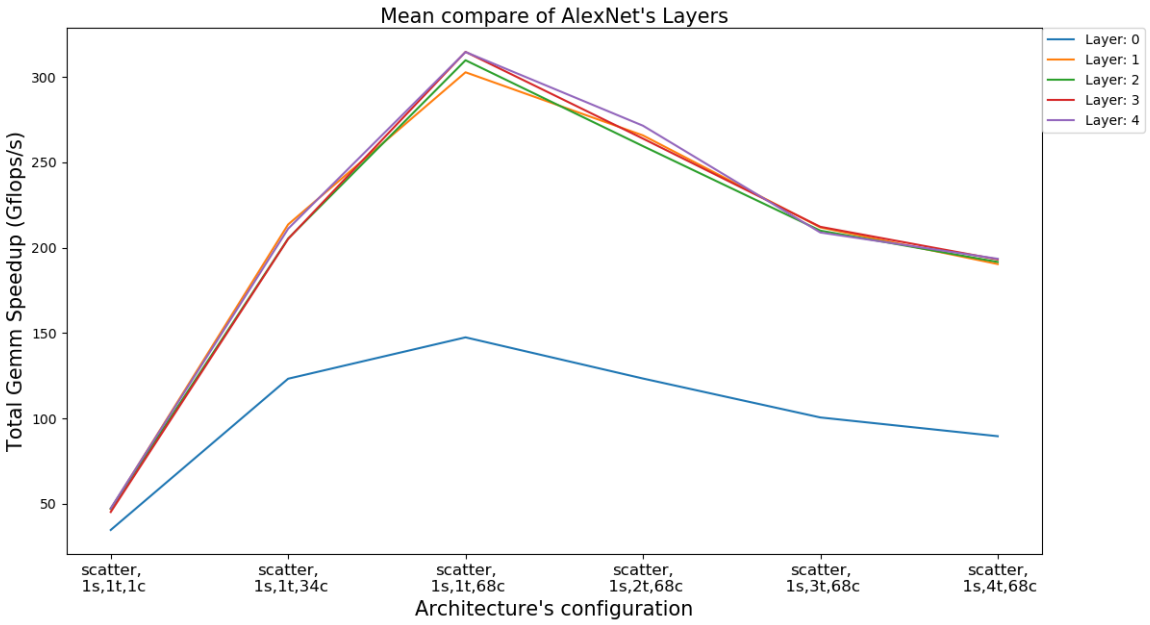
Το γράφημα της Εικόνας 5.3 μας προσφέρει κάποιες σημαντικές πληροφορίες. Αρχικά, παρατηρούμε ότι η τεχνική Απευθείας Συνέλιξης παρόλη την προσπάθεια βελτιστοποίησης της δεν μπορεί να συναγωνιστεί ακόμα και τις πλέον αργές GEMM, μιας και διατηρεί σταθερά την χειρότερη επίδοση.

Ακόμα, σε συνδιασμό με το γράφημα των Σχημάτων 5.1 και 5.2 παρατηρούμε ότι η επίδοση της μεθόδου Expensive Lowering στα πρώτα Επίπεδα του δικτύου Alexnet είναι ικανή για να την καταστήσει αρκετά πιο γρήγορη παρότι στα επόμενα επίπεδα δεν είναι καλύτερη από άλλες μεθόδους. Αυτό συμβαίνει και στα υπόλοιπα δίκτυα καθώς όπως θα δούμε και παρακάτω η μέθοδος Expensive Lowering παρουσιάζει μακράν την καλύτερη επίδοση στις περιπτώσεις όπου οι χωρικές διατάξεις της εισόδου είναι μεγάλες, στα αρχικά επίπεδα δηλαδή, δίνοντας έτσι στην μέθοδο αυτή ένα σηματνικό πλεονέκτημα έναντι των υπολοίπων. Επομένως θα μπορούσαμε να καταληξουμε στο συμπέρασμα ότι η Expensive Lowering τεχνική αποτελεί μια εξαιρετική επιλόγη όσον αφορά την επίδοση στην περίπτωση που θέλουμε να περιοριστούμε στην χρήση μόνο μίας τεχνικής για την εκτέλεση ενός δικτύου. Στην συνέχεια, βλέπουμε ότι ακολουθεί η μέθοδος Balanced, η οποία διαθέτει και αυτή το συγκεκριμένο πλεονέκτημα σε μικρότερο βαθμό βέβαια, ενώ η τεχνική του Expensive Lifting αντιμετωπίζει αρκετά προβλήματα από άποψη συνολικού χρόνου εκτέλεσης μιας και όπως είδαμε στα προηγούμενα σχήματα επιφέρει μεγάλη καθυστέρηση στα αρχικά επίπεδα.

Όσον αφορά την σύγκριση μεταξύ δικτύων παρατηρούμε ότι το πιο αργό δίκτυο είναι αυτό του VGG καθώς περιέχει και τις περισσότερες παραμέτρους. Αντίθετα αρκετά γρηγορότερα δίκτυα είναι τα AlexNet-CaffeNet και Squeezenet κάτι που δικαιολογείται απόλυτα από το ότι τα πρώτα έχουν μικρό αριθμό παραμέτρων, ενώ το SqueezeNet, όπως είδαμε, δημιουργήθηκε με τον στόχο να είναι ένα ελαφρύ δίκτυο.

Έλεγχος κλιμακωσιμότητας

Σε αυτό το σημείο θα ελέγξουμε την δυνατότητα κλιμακωσιμότητας **(Scalability)** της ταχύτητας εκτέλεσης κάθε Επιπέδου του δικτύου AlexNet στην αύξηση των νημάτων της αρχιτεκτονικής. Έτσι στο γράφημα της Εικόνας 5.4 έχουμε στον οριζόντιο άξονα τον αριθμό των νημάτων και την αντίστοιχη ρύθμιση της αρχιτεκτονικής που τα συνοδεύει και στον κάθετο άξονα την ταχύτητα εκτέλεσης της GEMM συνάρτησης. Κάθε γραμμή διαφορετικού χρώματος υποδηλώνει την συμπεριφορά καθενός από τα πέντε συνολικά συνελικτικά επίπεδα του AlexNet. Παρατηρούμε ότι με την άυξηση των νημά-



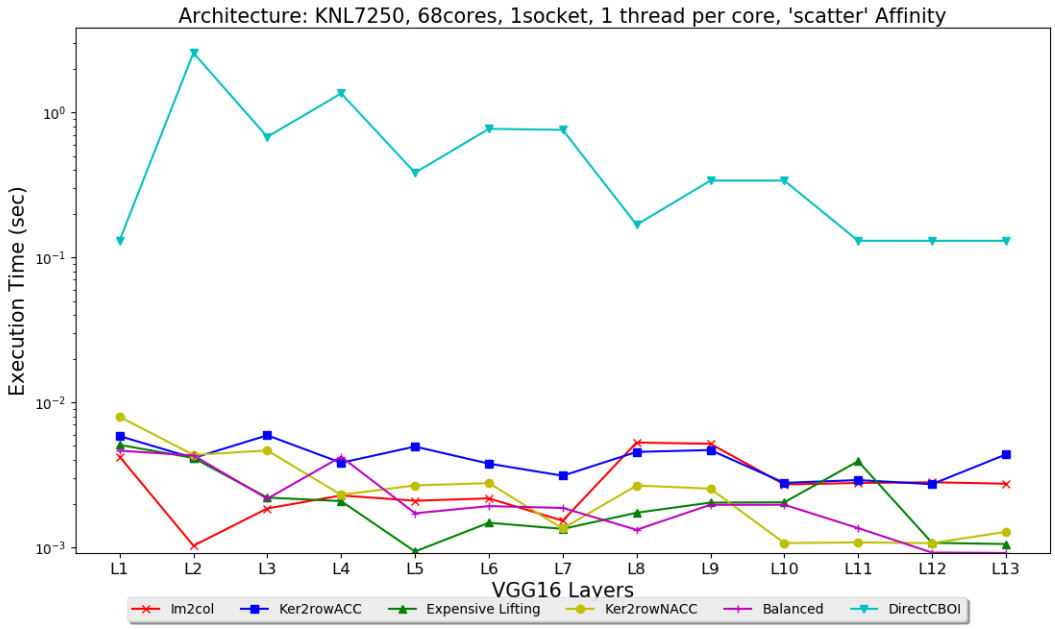
Σχήμα 5.4: *Περιγραφή της κλιμακωσιμότητας της Ταχύτητας για κάθε Επίπεδο του AlexNet για την αρχιτεκτονική Knl7250*

των από 1 σε 68 η ταχύτητα αυξάνεται σταθερά. Υπενθυμίζουμε ότι, με την παράμετρο "Affinity" ρυθμισμένη στην τιμή "scatter", σε αυτό το διάστημα γίνεται ανάθεση ενός μόνο νήματος σε κάθε πυρήνα, αποφεύγοντας έτσι την χρήση του υπερνηματισμού. Το γεγονός αυτό είναι σημαντικό μιας και όπως παρατηρούμε για αριθμό νημάτων μεγαλύτερο από αυτό των πυρήνων (δηλαδή > 68) η επίδοση αρχίζει να μειώνεται. Μάλιστα όσο αυξάνουμε τον αριθμό των νημάτων τόσο χειροτερεύει η επίδοση καθώς χρησιμοποιούμε όλο και περισσότερο τον υπερνηματισμό και δημιουργούμε περισσότερες καταστάσεις συναγωνισμού στις διαμοιραζόμενες (shared) δομές του πυρήνα, όπως η Μονάδα Επερξεργασίας Κινητής Υποδιαστολής (Floating Point Processing Units). Επί-

σης, είναι εμφανές στην Εικόνα 5.4 ότι το Επίπεδο που παρουσιάζει πάντα την χαμηλότερη ταχύτητα είναι το πρώτο καθως σε εκείνο ερχόμαστε αντιμέτωποι με εικόνες υψηλών χωρικών διαστάσεων, οι οποίες είναι πολύ δύσκολα διαχειρίσιμες και απαιτούν αρκετές μεταφόρες από την μνήμη στις Επεξεργαστικές Μονάδες.

Σύγκριση Χρόνου Εκτέλεσης ανά Επίπεδο και Τεχνική Υλοποίηση

Ιδιαίτερα χρήσιμη είναι και η απεικόνιση του χρόνου εκτέλεσης ανά Επίπεδο και τεχνική υλοποίησης. Για αυτόν τον σκοπό στο Σχήμα 5.5 παρουσιάζουμε τον χρόνο εκτέλεσης για κάθε επίπεδο του δικτύου VGG16 και για κάθε τεχνική υλοποίησης (όλες τις GEMM και την καλύτερη τεχνική Απεθείας Συνέλιξης δηλαδή την Cache Blocking Output Based).



Σχήμα 5.5: *Παρουσίαση του Χρόνου Εκτέλεσης κάθε Συνελικτικού Επιπέδου του δικτύου VGG16 ανά τεχνική υλοποίησης για την αρχιτεκτονική Knl7250*

Το πρώτο πράγμα που μπορούμε να παρατηρήσουμε από το γράφημα του Σχήματος 5.5 είναι ότι η τεχνική Απευθείας Συνέλιξης απαιτεί δύο τάξεις μεγέθους μεγαλύτερο χρόνο εκτέλεσης, κάτι το οποίο το γνωρίζαμε.

Όσον αφορά τις υπόλοιπες τεχνικές μπορούμε να παρατηρήσουμε ότι για τα επίπεδα όπου τα Blobs εισόδου έχουν μεγάλες χωρικές διαστάσεις αρκετά καλή επίδοση παρουσιάζει η τεχνική του Expensive Lowering ενώ την ακολουθεί από πολύ κοντά και η Balanced Lowering - Lifting και η Expensive Lowering. Οι τεχνικές αυτές πλεονεκτούν έναντι των υπολοίπων στην διαχείριση χωρικά μεγάλων δομών εισόδων, καθώς είναι το κέντρο της λογικής τους. Αντίθετα σχετικά μη αποδοτικές είναι η Ker2row τεχνικές καθώς αυτές επικεντρώνονται στην διαχείριση της δομής του Φίλτρου, η οποία έχει μικρό βάθος και συνεπώς είναι εύκολα διαχειρίσιμη, ακυρώνοντας έτσι την πλεονεκτήματα της μεθόδου.

Όσο προχωρούμε σε επίπεδα με δομές μικρότερων χωρικών και μεγαλύτερων διαστάσεων βάθους τόσο τα πράγματα μεταβάλλονται. Στα επίπεδα με μία σχετική ισορροπία μεταξύ διαστάσεων χώρου(ύψος και πλάτος) και βάθους παρατηρούμε ότι εξαιρετικά γρήγορη είναι η τεχνική του Expensive Lifting, κάτι το οποίο οφείλεται στο γεγονός ότι αποτελεί την GEMM τεχνική που εκτελεί τον μικρότερο αριθμό πράξεων κατά την διάρκεια της GEMM εκτέλεσης. Η Balanced Τεχνική, η οποία υιοθετεί έναν συνδιασμό της λογικής της Expensive Lowering και Expensive Lifting τεχνικής, παρατηρούμε ότι συνεχίζει να εμφανίζει καλή επίδοση όσο οι χωρικές διαστάσεις των δομών των επιπέδων μειώνονται, σε αντίθεση με την Expensive Lowering η οποία βλέπει την απόδοση της να μειώνεται όσο προχωρούν τα επίπεδα. Είναι γεγονός ότι όταν οι διαστάσεις βάθους είναι πολύ μεγαλύτερες από τις χωρικές τότε οι ενδιάμεσες δομές που δημιουργούνται κατά το Lowering της τεχνικής Expensive Lowering τείνουν σε δύσμορφα διανύσματα για την υλοποίηση του πολλαπλασιασμού και αυτό είναι που δημιουργεί την κακή επίδοση της συγκεκριμένης τεχνικής για αυτού του είδους τα επίπεδα. Σε αντίθεση με αυτά, με την πάροδο των επιπέδων οι Ker2row τεχνικές συνεχώς αυξάνουν την επιδοσή τους. Αυτό οφείλεται στο ότι όσο προχωρούμε τόσο αυξάνεται το βάθος των δομών Φίλτρου και τόσο η λογική των τεχνικών αυτών εμφανίζει χρησιμότητα, αφού δημιουργούν δομές με πιο εσωτερική διάσταση αυτή του βάθους αυξάνοντας έτσι την τοπικότητα.

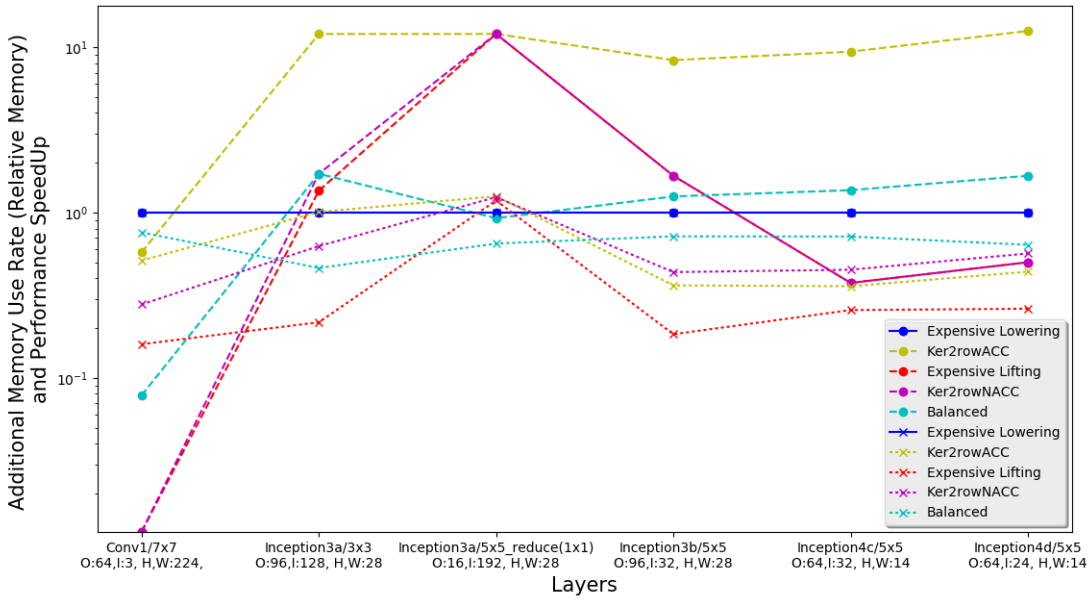
Σύγκριση Απαιτήσεων Μνήμης και Ταχύτητας για κάθε Τεχνική και Δίκτυο

Είναι πολύ ενδιαφέρον να συγκρίνουμε τις απαιτήσεις μνήμης με την επίδοση για κάθε τεχνική και δικτύο, ώστε να λάβουμε συμπεράσματα σχετικά με το που συμφέρει να χρησιμοποιείται η καθεμία. Για αυτόν τον σκοπό στο γράφημα του Σχήματος 5.6 απεικονίζεται η σχετική επιτάχυνση ή επιβράδυνση (speedup ή slowdown) που εμφανίζει κάθε τεχνική σε ζητήματα μνήμης και χρόνου εκτέλεσης σε σχέση με την τεχνική Im2col (Expensive Lowering) η οποία χρησιμοποιείται σαν βάση των μετρήσεων. Στο συγκεκριμένο γράφημα η επιτάχυνση της ταχύτητας κάθε τεχνικής απεικονίζεται με 'o', η σχετική απαίτηση για περισσότερη ή λιγότερη μνήμη με 'x', ενώ με συνεχή γραμμή είναι η βάση των μετρήσεων. Επομένως, όσο περισσότερο απέχει μια μέτρηση που βρίσκεται στην χώρο πάνω από την βάση τόσο μεγαλύτερο είναι το μεγαλύτερο είναι το speedup ή τόσο λιγότερη μνήμη απαιτεί για την υλοποίηση της, ενώ το αντίθετο συμβαίνει όταν η μέτρηση βρίσκεται κάτω από την βάση. Τονίζεται ότι στο παραπάνω σχήμα δεν υπολογίζουμε την συνολική μνήμη που απαιτεί η κάθε τεχνική για κάθε επίπεδο αλλά η **επιπλέον μνήμη** που χρειάζεται. Τα επίπεδα είναι κατάλληλα επιλεγμένα από το δίκτυο GoogLeNet ώστε να διαφέρουν σε κάποια χαρακτηριστικά κάθε φορά ώστε να δούμε την επίδραση των στις μετρήσεις.

Συγκεκριμένα, στην Εικόνα 5.5 θεωρήσαμε το SpeedUp της Μνήμης και της Ταχύτητας για κάθε τεχνική, σύμφωνα με τις παρακάτω σχέσεις:

$$\text{Σχετική Επιπλέον Μνήμη(Relative Memory)} = \frac{\text{Επιπλέον Μνήμη Im2col}}{\text{Επιπλέον Μνήμη Τεχνικής}} \quad (5.1)$$

$$\text{Επιτάχυνση (Time SpeedUp)} = \frac{\text{Χρόνος Εκτέλεσης Im2col}}{\text{Χρόνος Εκτέλεσης Τεχνικής}} \tag{5.2}$$



Σχήμα 5.6: *Γράφημα Σύγκρισης Απαιτήσεων Μνήμης και Επίδοσης για κάθε τεχνική και δίκτυο*

Ας αρχίσουμε όμως να επεξεργαζόμαστε τα δεδομένα του Σχήματος 5.6. Αρχικά, παρατηρούμε όπως και αναμέναμε βέβαια ότι σε θέματα επίδοσης η τεχνική του Expensive Lowering αποτελεί μία πάρα πολύ γρήγορη επιλογή, καθώς είναι σε πολύ καλύτερο επίπεδο από τις υπόλοιπες με μία μικρή εξαίρεση την περίπτωση του επιπέδου που εκτελεί 1x1 Συνέλιξη, η οποία όμως δεν συναντάται συχνά στα δίκτυα και δεν είναι ρυθμιστικός παράγοντας. Πάντως, στην περίπτωση της 1x1 συνέλιξης είναι αλήθεια ότι όλες οι τεχνικές χάνουν τα χαρακτηριστικά τους και είναι σαν να ευτελίζονται σε μία κοινή υλοποίηση. Όσον αφορά τα υπόλοιπα επίπεδα παρατηρούμε ότι την Expensive Lowering τεχνική ακολουθέι σε επίδοση η Balanced η οποία είναι ανικάνη να βελτιωθεί παραπάνω μιας και χρησιμοποιεί και την λογική της Expensive Lifting, η οποία είναι σταθερά η πιο αργή μέθοδος. Ακόμα, παρατηρούμε ότι για τις Ker2row μεθόδους παίζουν ρόλο δύο παράμετροι, α) για μικρές χωρικές διαστάσεις της δομής φίλτρου οι τεχνικές αυτές βελτιώνονται σε μεγάλο βαθμό και ιδιαίτερα η Ker2rowACC, η οποία για 3x3 φίλτρα ξεπερνά και την Expensive Lowering για λίγο, και β) για μεγάλες χωρικές διαστάσεις της δομής εισόδου παρατηρούμε ότι η Ker2rowACC είναι πιο γρήγορη της Ker2rowNACC, γεγονός το οποίο είναι αντιστρέφεται στα επίπεδα όπου οι χωρικές διαστάσεις εισόδου μειώνονται.

Σε θέματα απαιτήσεων μνήμης, όμως, τα δεδομένα αντιστρέφονται. Αν και στο πρώτο επίπεδο όπου οι χωρικές διαστάσεις της εικόνας εισόδου είναι πολύ μεγαλύτερες όλων των υπολοίπων διαστάσεων, η Expensive Lowering τεχνική απαιτεί την

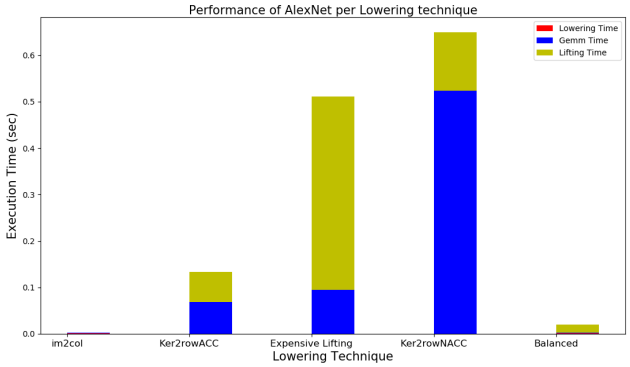
λιγότερη μνήμη, σε όλα τα υπόλοιπα η συγκεκριμένη μέθοδος χρειάζεται περισσότερη επιπλεόν μνήμη για την εκτέλεσή της. Συγκεκριμένα, στο πρώτο επίπεδο, το οποίο λαμβάνει σαν είσοδο μία εικόνα ανεπεξέργαστη με μεγάλες συνεπώς χωρικές διαστάσεις και παράγει σαν έξοδο μία δομή με αρκετά μικρότερες διαστάσεις, η τεχνική Expensive Lowering είναι η μόνη που ξεχωρίζει καθώς η ενδιάμεση δομή που χρησιμοποιεί στο Lowering βασίζεται στις χωρικές διαστάσεις εξόδου και όχι εισόδου, όπως όλες οι άλλες τεχνικές. Όμως, αυτό συμβαίνει μόνο στο εισαγωγικό πρώτο συνελικτικό επίπεδο των δικτύων το οποίο έχει και σαν στόχο την μεγάλη μείωση των διαστάσεων. Έτσι, στα επόμενα επίπεδα η εικόνα αλλάζει. Αρχικά, η Ker2rowACC τεχνική, η οποία χρησιμοποιεί σαν επιπρόσθετη μνήμη μόνον έναν απομονωτή διαστάσεων $h \times w \times o$ αποτελεί ότι καλύτερο σε άποψη χρήσης λιγότερης μνήμης. Μόνο στην περίπτωση της 1x1 συνέλιξης χρησιμοποιεί την ίδια ποσότητα μνπμης με την Ker2rowNACC και Expensive Lifting, οι οποίες απαιτούν την ίδια επιπρόσθετη μνήμη σε κάθε περίπτωση και για αυτόν τον λόγο δεν εμφανίζεται κάποια διαφορά στο γράφημα. Στην περίπτωση της Balanced, η οποία είναι η μόνη τεχνική που απαιτεί επιπλέον μνήμη τόσο στο Lowering όσο και στο Lifting, παρατηρούμε ότι βρίσκεται πάντα σε καλύτερη θέση από την Expensive Lowering. Τέλος, εξαιρετικά ενδιαφέρον είναι το γεγονός ότι στα επίπεδα όπου οι χωρικές διαστάσεις των φίλτρων είναι μεγάλες οι τεχνικές Expensive Lifting και Ker2rowNACC χρειάζονται αρκετή επιπρόσθετη μνήμη, ενώ στην περίπτωση που έχουμε πολλά φίλτρα (κανάλια εξόδου) η μνήμη που απαιτείται είναι μεγαλύτερη και από αυτή της προεπιλεγμένης μεθόδου του Caffe.

5.4.2 Πολυπύρηνη Αρχιτεκτονική - Intel Xeon E5

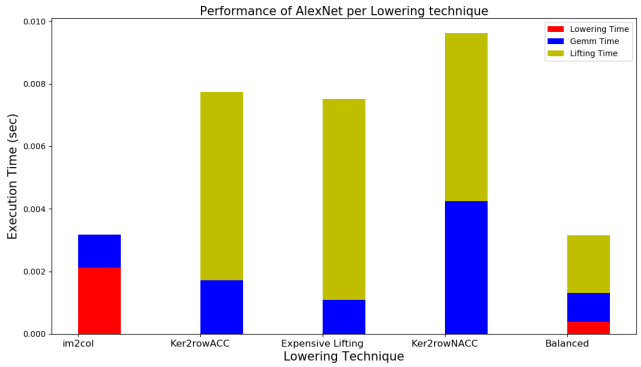
Σύγκριση Τεχνικών Lowering

Όπως και στην παρουσίαση των αποτελεσμάτων και στην προηγούμενη αρχιτεκτονική, έτσι και σε αυτήν θα ξεκινήσουμε με την απεικόνιση των ολικών χρόνων που απαιτεί κάθε τεχνική για την εκτέλεση του δικτύου AlexNet δίνοντας έμφαση στο ποσοστό που λαμβάνει το κάθε τμήμα της μεθόδου (Lowering - GEMM - Lifting). Τα αποτελέσματα απεικονίζονται στα Σχήματα 5.7 και 5.8.

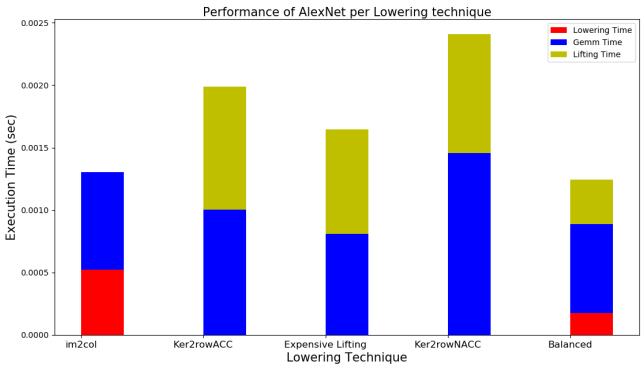
Παρατηρούμε ότι τα αποτελέσματα ομοιάζουν με αυτά της προηγούμενης τεχνικής με μία σημαντική διαφορά: Σε αυτήν την αρχιτεκτονική δεν υπάρχουν οι διαφόρες της προηγούμενης, καθώς το Lifting των τεχνικών Expensive Lifting και των Ker2row διαρκεί πολύ λιγότερο στα αρχικά επίπεδα από ότι στον Knl7250. Επίσης παρατηρούμε ότι απαιτεί πολύ χρόνο η GEMM εκτέλεση της τεχνικής Ker2rowNACC πολύ περισσότερο από στην προηγούμενη αρχιτεκτονική και πολύ περισσότερο από οποιαδήποτε άλλη τεχνική. Για τα υπόλοιπα δεν παρατηρούνται ουσιαστικές διαφορές με την προηγούμενη αρχιτεκτονική καθώς και εδώ το κυριότερο πρόβλημα των τεχνικών Expensive Lifting και Ker2rowACC είναι το Lifting, και εδώ η Expensive Lowering τεχνική δεν είναι πάντα πιο γρήγορη, αλλά και εδω οι τεχνικές έχουν θεωρητικά ορθές δομές.



(α) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 1 του AlexNet ανά Τεχνική GEMM

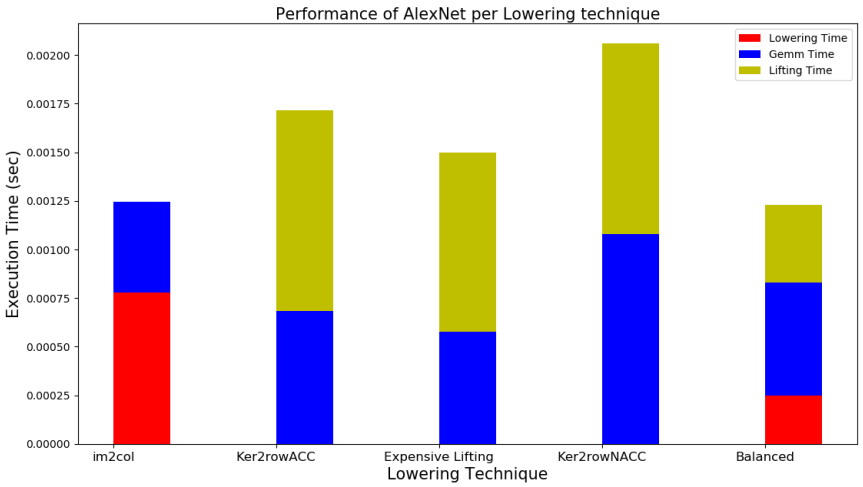


(β) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 2 του AlexNet ανά Τεχνική GEMM

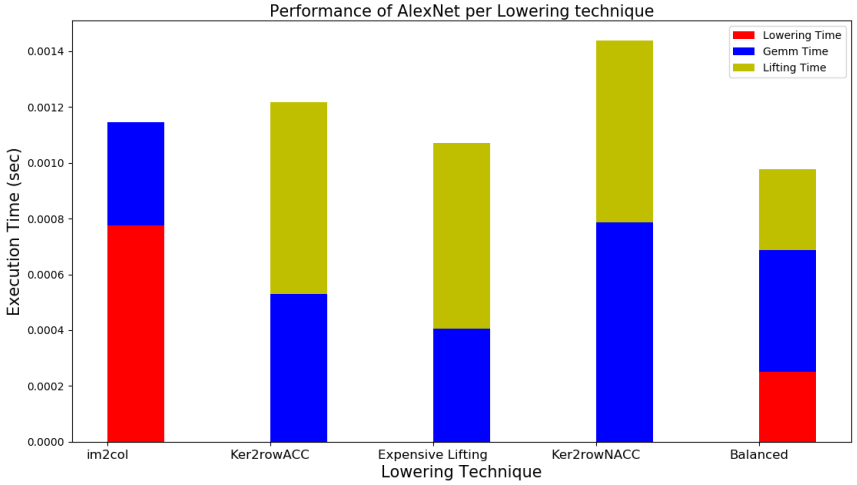


(γ) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 3 του AlexNet ανά Τεχνική GEMM

Σχήμα 5.7: *Παρουσίαση Επίδοσης της Εκτέλεσης για τα Επίπεδα 1,2 και 3 του AlexNet. Σε κάθε γράφημα παρουσιάζεται το τμήμα του χρόνου που αφορά το κάθε τμήμα της τεχνικής (Lowering - GEMM - Lifting) για την αρχιτεκτονική Intel Xeon E5*



(δ) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 4 του AlexNet ανά Τεχνική GEMM

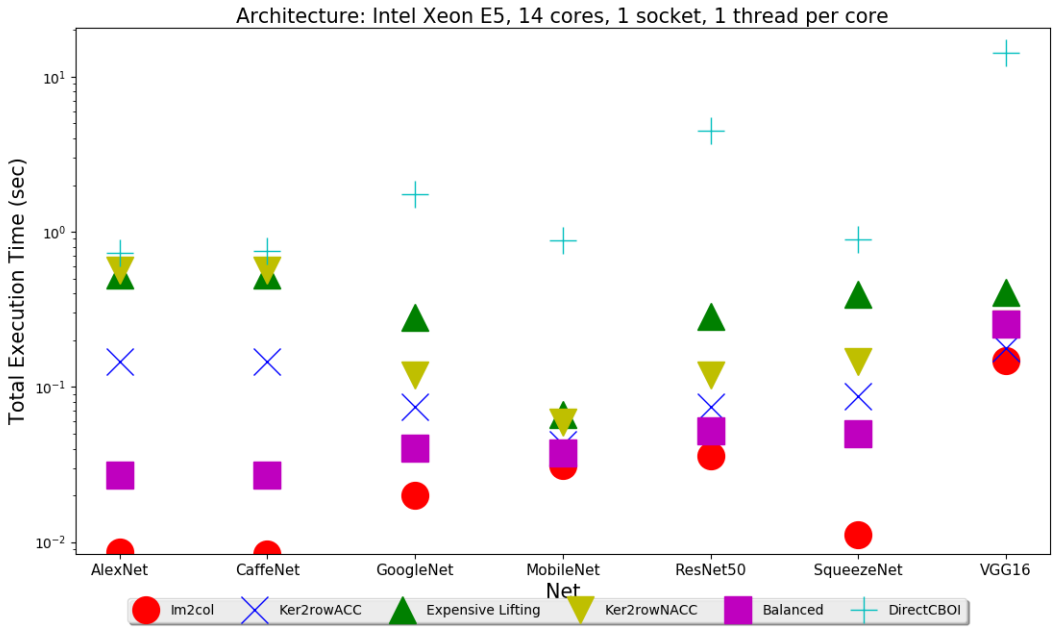


(ε) Μπάρα Χρόνου Εκτέλεσης Επιπέδου 5 του AlexNet ανά Τεχνική GEMM

Σχήμα 5.8: *Παρουσίαση Επίδοσης της Εκτέλεσης για τα Επίπεδα 1,2 και 3 του AlexNet. Σε κάθε γράφημα παρουσιάζεται το τμήμα του χρόνου που αφορά το κάθε τμήμα της τεχνικής (Lowering - GEMM - Lifting) για την αρχιτεκτονική Intel Xeon E5*

Σύγκριση Συνολικού Χρόνου Υλοποίησης ανά Τεχνική και Δίκτυο

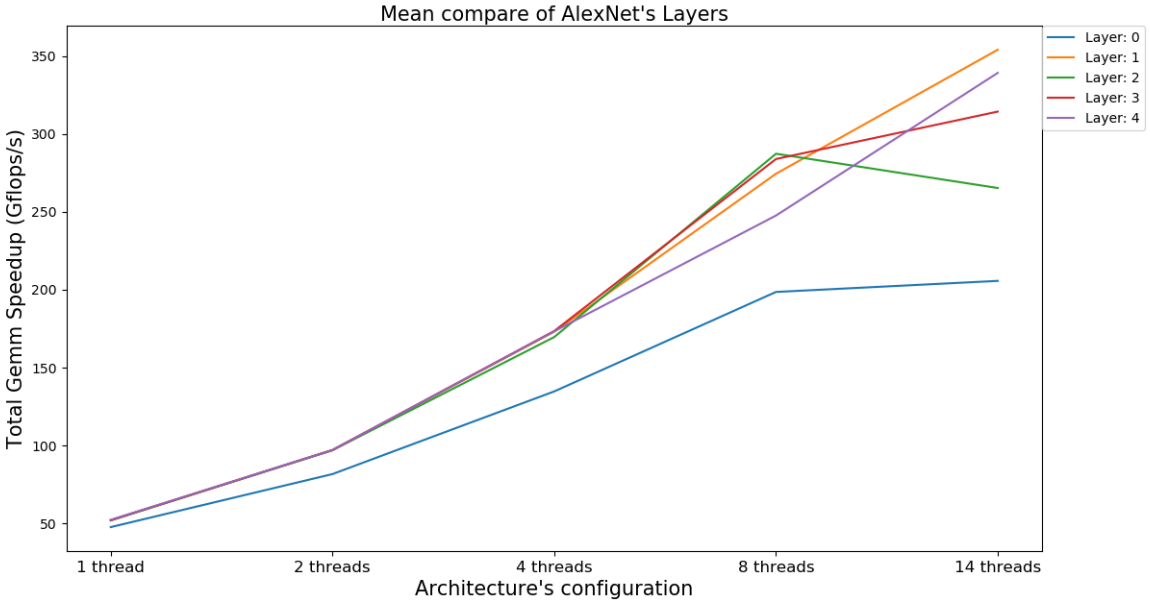
Αν τώρα πάμε να συγκρίνουμε τον συνολικό χρόνο υλοποίησης καθενός από τα διαθέσιμα δίκτυα για κάθε διαθέσιμη τεχνική θα καταλήξουμε σε και πάλι όμοια αποτελέσματα με την προηγούμενη αρχιτεκτονική. Παρατηρούμε ότι και πάλι η τεχνική Απευθείας Συνέλιξης: Cache Blocking Output Based (DirectCBOI) εμφανίζει πολύ αργή επίδοση συγκρινόμενη με τις αντίστοιχες των GEMM τεχνικών. Από την άλλη πλευρά και πάλι καλύτερη επίδοση εμφανίζει η Expensive Lowering τεχνική, η οποία βέβαια όπως εξηγήσαμε και προηγουμένως διαχειρίζεται πολύ καλύτερα τα αρχικά επίπεδα των δικτύων, και αυτό το πλεονέκτημα υπερνικά τις όποιες καθυστερήσεις της τεχνικής στα μετέπειτα επίπεδα. Όσον αφορά τις υπόλοιπες τεχνικές η Balanced ακολουθεί την Expensive Lowering, ενώ σε αρκετά καλά επίπεδα βρίσκονται και οι Ker2row τεχνικές και ιδιαίτερα η Ker2rowACC, η οποία απαιτεί λιγότερες μεταφορές δεδομένων από την μνήμη στο Lifting, καθώς εκεί διαχειρίζεται αρκετά μικρότερες δομές από ότι η Ker2rowNACC. Επίσης, η Expensive Lifting έχει ως μειονέκτημα της την ταχύτητα καθώς στοχεύει κυρίως στην εκτέλεση των λιγότερων δυνατών πράξεων κατά την εκτέλεση της GEMM συνάρτησης, φορτώνοντας μεγάλο φόρτο εργασίας στο Lifting. Παράλληλα, παρατηρούμε και πάλι ότι τα αποτελέσματα του Σχήματος 5.9 συμφωνούν με αυτά των 5.7 και 5.8 καθώς η πολύ καλή επίδοση της τεχνικής του Expensive Lowering στα αρχικά επίπεδα της δίνει ένα πολύ μεγάλο πλεονέκτημα που την καθιστά αρκετά πιο γρήγορη από τις υπόλοιπες. Τέλος, προφανώς, εξάγουμε τα ίδια αποτελέσματα με αυτά στην προηγούμενη αρχιτεκτονική όσον αφορά την σύγκριση των δικτύων.



Σχήμα 5.9: *Γράφημα περιγραφής του Συνολικού Χρόνου Εκτέλεσης για κάθε Δίκτυο και Τεχνική για την αρχιτεκτονική Intel Xeon E5*

Έλεγχος Κλιμακωσιμότητας

Σε αυτό το στάδιο ελέγχουμε την κλιμακωσιμότητα του συστήματος. Για αυτόν τον λόγο παρουσιάζουμε την ολική ταχύτητα εκτέλεσης του κάθε επιπέδου του δικτύου AlexNet μεταβάλλοντας τον αριθμό των νημάτων. Το αποτέλεσμα που λάμβαμε εμφανίζεται στο γράφημα του Σχήματος 5.10. Σε αυτό είναι εμφανής η κλιμακωσιμότητα καθώς αυξάνεται σχεδόν εκθετικά ο χρόνος εκτέλεσης των Επιπέδων, κάτι που είναι λογικό σε αντίθεση με την προηγούμενη αρχιτεκτονική καθώς εδώ έχουμε αποκλίσει την χρήση τόσο του υπερνηματισμού των πυρήνων όσο και την χρήση πυρήνων από διαφορετικό socket του επεξεργαστή κάτι που θα μείωνε σε μεγάλο βαθμό την τοπικότητα μεταξύ των πυρήνων. Επίσης, σε αντίθεση με την προηγούμενη αρχιτεκτονική τώρα η επίδοση του πρώτου επιπέδου δεν διαφέρει πολύ από την επίδοση των υπολοίπων επιπέδων.



Σχήμα 5.10: *Περιγραφή της Κλιμακωσιμότητας της Ταχύτητας για κάθε Επίπεδο του AlexNet για την αρχιτεκτονική Intel Xeon E5*

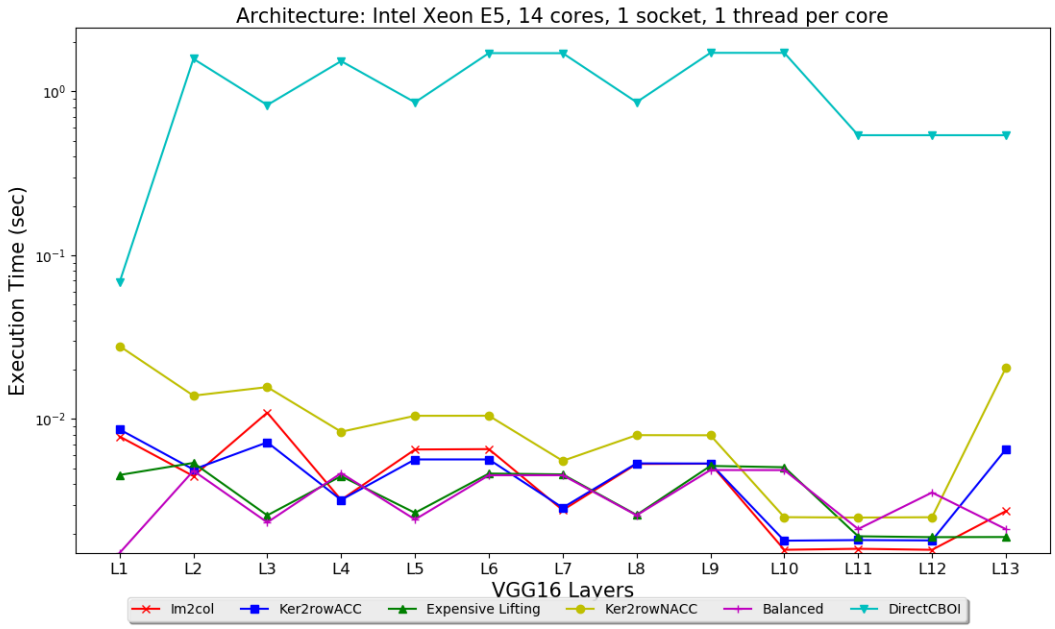
Σύγκριση Χρόνου Εκτέλεσης ανά Επίπεδο και Τεχνική Υλοποίηση

Διατηρώντας την ίδια λογική με πριν και παίρνοντας τις ίδιες μετρήσεις για την παραγωγή του γραφήματος του Σχήματος 5.11, φτάνουμε σε κάποια διαφορετικά αποτελέσματα σχετικά με την προηγούμενη αρχιτεκτονική. Αρχικά, η υλοποίηση της Απευθείας Συνέλιξης χρειάζεται σχεδόν δύο (2) τάξεις μεγέθους περισσότερο χρόνο για την υλοποίηση της, πράγμα σαφώς αναμενόμενο. Η διαφόρες με τον Knl7250 αφορά τις υπόλοιπες τεχνικες. Εκεί, για τα αρχικά συνελικτικά επίπεδα του δικτύου VGG16, ναι μεν οι καλύτερες τεχνικές είναι οι Balanced και η Expensive Lifting, αλλά τώρα έχει υποχωρήσει αρκετά η Expensive Lowering. Αυτό οφείλεται στον αριθμό των διαθέσι-

μων νημάτων, καθώς η τεχνική αυτή δημιουργεί μεγάλους ενδιάμεσους πίνακες και τα μόλις 14 νήματα που διαθέτουμε αδυνατούν να παραλληλοποιήσουν σε μεγάλο βαθμό τον πολλαπλασιασμό. Κατά τα άλλα οι μεγάλες χωρικές διαστάσεις επιφέρουν όμοια καθυστέρηση στις Ker2row τεχνικές με τον Knl7250.

Όσο όμως προχωρούμε στα βαθύτερα επίπεδα η συμπεριφορά αλλάζει, όπως και αναμέναμε. Αρχικά, οι Ker2row τεχνικές γίνονται όλο και πιο γρήγορες και μάλιστα η Ker2rowACC, η οποία καλεί πολλές φορές την GEMM συνάρτηση για την εκτέλεση πολλαπλασιασμών πινάκων μικρότερων διαστάσεων, απαιτεί λιγότερες μεταφορές από την μνήμη, και, συνεπώς, είναι σταθερά γρηγορότερη από την Ker2rowNACC. Στα ενδιάμεσα επίπεδα οι πλεόν γρήγορες είναι οι Expensive Lifting και Balanced τεχνικές, ενώ η Expensive Lowering παραμένει σχετικά πιο αργή.

Η μεγάλη όμως διαφορά με την προηγούμενη αρχιτεκτονική εμφανίζεται στα τελευταία Συνελικτικά επίπεδα του δικτύου. Εδώ αν και οι Ker2row τεχνικές γίνονται πιο γρήγορες, δεν αποτελούν την καλύτερη τεχνική για τα στάδια αυτά, καθώς εμφανίζονται πολύ γρήγορες η Expensive Lifting αλλά κυρίως η Expensive Lowering τεχνικές, οι οποίες ακολουθούν την Ker2rowNACC τεχνική.



Σχήμα 5.11: *Παρουσίαση του Χρόνου Εκτέλεσης κάθε Συνελικτικού Επιπέδου του δικτύου VGG16 ανά τεχνική υλοποίησης για την αρχιτεκτονική Intel Xeon E5*

Σαν τελικό συμπέρασμα από την σύγκριση των μετρήσεων μεταξύ αρχιτεκτονικών παρατηρούμε ότι τα συγκεκριμένα αποτελέσματα που αναφέραμε παραπάνω μπορεί να ποικίλουν ανάλογα τις προδιαγραφές της εκάστοτε αρχιτεκτονικής, και αυτό μας δίνει μεγαλύτερο ενδιαφέρον στην διερεύνηση όλων αυτών για διαφορετικές αρχιτεκτονικές και στην εξαγωγή κανόνων επιλογής της κατάλληλης τεχνικής με βάση διάφορετικά κριτήρια. Πάντως με βάση αυτές τις δύο διαθέσιμες αρχιτεκτονικές μπορούμε να λάμβουμε τα συμπεράσματα που ακολουθούν στο επόμενο κεφάλαιο.

Μέρος III

Επίλογος

Κεφάλαιο 6

Επίλογος

Σε αυτό το κεφάλαιο θα προσπαθήσουμε να συγκεντρώσουμε τα συμπεράσματα που εξάγαμε στα προηγούμενα κεφάλαια και να προβούμε στην δημιουργία μιας σειράς κανόνων για την χρησιμοποίηση της κατάλληλης τεχνικής για τις εκάστοτε συνθηκες. Τέλος, θα προτείνουμε κάποιους τρόπους περαιτέρω εργασίας για την βελτίωσης του έργου της παρούσας εργασίας.

6.1 Συμπεράσματα

Μέσω της μελέτης της επίδοσης της εφαρμογής των Συνελικτικών Δικτύων στις διαφορετικές αρχιτεκτονικές που πραγματοποιήσαμε σε αυτήν την εργασία, μπορούμε να προβούμε στην δημιουργία κάποιων κανόνων επιλογής των τεχνικών που πραγματοποιήσαμε. Οι κανόνες αυτοί θα βασίζονται σε τρείς άξονες:

- **Χρόνος Εκτέλεσης**
- **Μέγεθος Απαιτούμενης Μνήμης**
- **Αριθμός Εκτελέσιμων Πράξεων - Ταχύτητα Εκτέλεσης GEMM**

Ας πάρουμε τον κάθε άξονα ξεχωριστά:

6.1.1 Χρόνος Εκτέλεσης

Όσον αφορά τον χρόνο εκτέλεσης οι συμπεράσματα που μπορούμε να εξάγουμε είναι τα εξής:

- Σε θέματα χρόνου εκτέλεσης οποιαδήποτε σύγκριση μεταξύ των μεθόδων Απευθείας Συνέλιξης με τις αντίστοιχες GEMM είναι μάταιη. Όση προσπάθεια και να καταναλωθεί στην βελτίωση των πρώτων θα υπολείπονται πάντα των δεύτερων.
- Η τεχνική του Expensive Lowering αποτελεί γενικά την καλύτερη επιλόγη για την εφαρμογή των Συνελικτικών Δικτύων με την μεγαλύτερη δυνατή ταχύτητα. Το γεγονός ότι αποφεύγει την κοπιαστική και χρονοβόρα διαδικασία του Lifting της δίνει ένα πολύ σημαντικό πλεονέκτημα έναντι των υπολοίπων, ικάνο να ξεπεράσει

το γεγονός ότι είναι από τις πιο αργές κατά την εκτέλεση του πολλαπλασιασμού μέσω της GEMM συνάρτησης.

- Την Expensive Lowering ακολουθεί σε επίδοση η Balanced, η οποία όμως δεν αποφεύγει την διαχείριση μίας εκ των Lowering ή Lifting και επομένως συνήθως μειωνεκτεί από άποψη χρόνου.

- Η πλέον αργή τεχνική είναι αυτή του Expensive Lifting, η οποία καταναλώνει πολύ χρόνο στην εκτέλεση του πιο κοπιασικού Lifting από όλες τις τεχνικές. Ταυτόχρονα, οι Ker2row τεχνικές κυμαίνονται μεταξύ των χρόνων των τεχνικών Expensive Lifting και Balanced.

6.1.2 Μέγεθος Απαιτούμενης Μνήμης

Στον άξονα του Μεγέθους Απαιτούμενης Μνήμης τα πράγματα είναι αντίστροφα:

- Η τεχνική της Απευθείας Συνέλιξης δεν απαιτεί κάποια επιπρόσθετη μνήμη και, συνεπώς, χρειάζεται την λιγότερη επιπλέον μνήμη από τις υπόλοιπες. Το γεγονός όμως ότι υπολείπεται σε μεγάλο βαθμό από τις GEMM υλοποιήσεις μας βάζει σε σκέψη για την συγκεκριμένη τεχνική.

- Σε γενικές γραμμές, μετά την τεχνική της Απευθείας Συνέλιξης ακολουθεί σε απαιτήσεις μνήμης η τεχνική Ker2rowACC η οποία αξιώνει την δημιουργία μόνο ενός απομονωτή διαστάσεων (ύψος) $\times$ (πλάτος) $\times$ (Αριθμός Φίλτρων). Εξαιρετικά λιτές στις απαιτήσεις τους είναι οι τεχνικές Expensive Lifting και Ker2rowNACC που απαιτούν ακριβώς το ίδιο μέγεθος επιπλέον μνήμης, ενώ την περισσότερη μνήμη αξιώνουν η Balanced και ιδιαίτερα η Expensive Lowering τεχνική.

- Στην περίπτωση των Επίπεδων που έχουν ως σκοπό την μεγάλη μείωση των διαστάσεων της δομής εισόδου, όπως συνηθίζουν τα αρχικά συνελικτικά επίπεδα των δικτύων, καλύτερη επιλογή αποτελεί αυτή του Expensive Lowering καθώς είναι η μόνη που απαιτεί επιπρόσθετες δομές διαστάσεων που βασίζονται στις δομές εξόδου και όχι εισόδου. Αντίθετα στις περιπτώσεις των επιπέδων με μικρές χωρικές διαστάσεις εισόδου και μεγάλες χωρικές διαστάσεις φίλτρων, οι τεχνικές Expensive Lifting και Ker2rowNACC απαιτούν περισσότερη μνήμη και απο την τεχνική του Expensive Lowering.

6.1.3 Αριθμός Εκτελέσιμων Πράξεων - Ταχύτητα Εκτέλεσης GEMM

Μικρότερος αριθμός των εκτελέσιμων πράξεων σημαίνει λιγότερος χρόνος για τον GEMM πολλαπλασιασμό, άρα και λιγότερη χρήση κάποιων ιδιαίτερων δομών των επεξεργαστών, όπως αυτές της Εκτέλεσης Πράξεων Κινητής Υποδιαστολής. Επομένως, έχουν σημασία τα παρακάτω συμπεράσματα:

- Η Expensive Lifting τεχνική, καθώς και οι αντίστοιχες Ker2row, επιφορτίζουν τον επεξεργαστή με τις λιγότερες πράξεις, ενώ αντίθετα αρκετά επιβαρής μέθοδος για την GEMM συνάρτηση είναι η Expensive Lowering, από το Lowering της οποίας

Πίνακας 6.1: *Περιγραφή Εξαγόμενων Απλών Κανόνων*

Χαρακτηριστικά Επιπέδου					Βέλτιστη Επιλογή Τεχνικής με βάση τα κριτήρια		
Επίπεδο	**H,W**	**I**	**O**	**kh,kw**	**Χρόνος Εκτέλεσης**	**Μνήμη***	**Αριθμός Πράξεων****
Αρχικό	Μεγάλα	Μικρό	-	Μεγάλα	Im2col	Im2col	Im2col
Ενδιάμεσο	Μεσαία	Μεγάλο	-	Μικρό	Im2col ή Ker2rowACC	Ker2rowACC	Exp. Lifting
Ενδιάμεσο	-	-	-	-	Im2col	Ker2rowACC	Exp. Lifting
-	-	-	-	1	Όλες	Ker2row ή Exp.Lifting	Exp. Lifting
Τελικό	Μικρό	-	Μεγάλο	Μικρό	Im2col	Ker2rowACC	Ker2rowACC
Τελικό	Μικρό	Μεγάλο	Μεγάλο	-	Balanced	Ker2rowACC	Balanced ή Ker2rowNACC

προκύπτει μία δομή με πολλές επαναλαμβανόμενες τιμές της αρχικής δομής.

- Μεγάλο ρόλο παίζει και το σχήμα των δομών που πολλαπλασιάζονται στον χρόνο εκτέλεσης του πολλαπλασιασμού. Για παράδειγμα για μεγάλες χωρικές διαστάσεις της δομής εισόδου η τροποποιημένη δομή της τεχνικής Expensive Lowering έχει σφαιρικό σχήμα, κάτι που επιταγχύνει την GEMM εκτέλεση, ενώ αντίστοιχα το γεγονός αυτό εμφανίζεται στην τεχνική Expensive Lifting στα ενδιάμεσα επίπεδα όπου οι όλες οι διαστάσεις είναι λίγο πολύ ισορροπημένες. Ακόμα το γεγονός ότι η Ker2rowACC εκτελεί τον GEMM για κάθε ένα από τα χωρικά στοιχεία των φίλτρων προκαλεί σημαντική καθυστέρηση μιας και το σχήμα των δομών που πολλαπλασιάζονται δεν είναιτο επιθυμητό.

6.1.4 Εξαγόμενοι Κανόνες

Με βάση τα παραπάνω συμπεράσματα αλλά και αυτά του προηγούμενου κεφαλαίου μπορούμε να συνθέσουμε κάποιους κανόνες σχετικά με την επιλογή των διαφορετικών τεχνικών, οι οποίοι παρουσιάζονται στον Πίνακα 6.1.

Όπως παρατηρήσαμε υπάρχει μία ισορροπία (trade off) ανάμεσα στα κριτήρια με την έννοια ότι είναι πολύ σπάνιο να υπάρχει μία τεχνική που να αποτελεί την καλύτερη επιλογή για κάθε κριτήριο ταυτόχρονα. Για αυτό είναι ιδιαίτερα σημαντικό να προχωρήσουμε σε μία σειρά από κανόνες, οι οποία θα προσπαθήσουν να συνδιάσουν καλές επιδόσεις σε δύο ή περισσότερα κριτήρια ταυτόχρονα. Τα κριτήρια αυτά εμφανίζονται στον Πίνακα 6.2, στον οποίο έχουμε κάνει την κατηγοριοποίηση των επιπέδων αρκετά ποιοτικά με βάση την θέση τους στην δομή ενός δικτύου, η οποία συνεπάγεται και το μέγεθος των διαστάσεων που το χαρακτηρίζουν.

*Στους κανόνες έχουμε αποκλίσει την επιλογή της μεθόδου Απευθείας Συνέλιξης μιας και αυτή αξιώνει πάντα την λιγότερη μνήμη. **Σε αυτήν την στήλη το κριτήριο δεν είναι ο μικρότερος αριθμός πράξεων αλλά κυρίως η ταχύτητα GEMM εκτέλεσης

6.2 Μελλοντικές Επεκτάσεις

Η ανάλυση που πραγματοποιήθηκε στην παρούσα εργασία μπορεί να βοηθήσει στην ανάπτυξη της εφαρμογής των Συνελικτικών Επιπέδων σε πολλές αρχιτεκτονικές είτε πρόκειται για επεξεργαστές είτε για ενσωματωμένα συστήματα. Επίσης υμβάλει στην δημιουργία στην περαιτέρω εξέλιξη του Caffe το οποίο θα μπορούσε σε κάποια

Πίνακας 6.2: *Περιγραφή Εξαγόμενων Συνδιαστικών Κανόνων*

Συνδιασμός Κριτηρίων	Βέλτιστη Επιλογή Τεχνικής
Αρχικό Επίπεδο	
Όλα	Im2col
Ενδιάμεσο Επίπεδο	
Ταχύτητα + Μνήμη	Ker2rowACC
Ταχύτητα + Αριθμός Πράξεων**	Balanced
Μνήμη + Αριθμός Πράξεων**	Expensive Lifting
Τελικό Επίπεδο	
Ταχύτητα + Μνήμη	Im2col ή Balanced
Ταχύτητα + Αριθμός Πράξεων**	Ker2rowNACC
Μνήμη + Αριθμός Πράξεων**	Balanced

μελλοντική έκδοση να προσαρμόζεται ακόμα καλύτερα στην υπάρχουσα αρχιτεκτονική όπου εγκαθίσταται και με βάση τα παραπάνω κριτήρια να κάνει την σωστή επιλογή των τεχνικών υλοποίησης των Συνελικτικών Επιπέδων. Η συγκεκριμένη, βέβαια, θα απαιτούσε αρκετά μεγάλη προεργασία, μέρος της οποίας υλοποιήθηκε στο πλάισιο αυτής της εργασίας, και την επίλυση κάποιων σημαντικών προβλημάτων όπως η δυνατότητα εύκολης και χρονοβόρας εναλλαγής της διάταξης των δομών κατά το πέρασμα από το ένα επίπεδο στο επόμενο.

Σε προσθήκη των παραπάνω, εξαιρετικά μεγάλο ενδιαφέρον παρουσιάζει και η δοκιμή των παραπάνω τεχνικών και σε εναλλακτικές αρχιτεκτονικές, όπως αυτή των καρτών γραφικών (GPUs) και η διερεύνηση του κατά πόσο τα κριτήρια αυτά ικανοποιούνται και σε αυτές. Τέλος, η ανάλυση που υλοποιήθηκε δίνει μία ιδέα στον ερευνητή για το ποιές παράμετροι προκαλούν προβλήματα και καθυστέρηση σε κάθε περίπτωση, πληροφορία που μπορεί να τον στρέψει στην δημιουργία νέων δικτύων ειδικά προσαρμοσμένων στην αντιμετώπισή τους.

Βιβλιογραφία

[1] Sergios Theodoridis και Konstantinos Koutroumbas. *Pattern recognition*. Elsevier/Acad. Press, 2012.

[2] Xiu Qing Zhang και Shu Wang Chen. *Speech recognition system based on DSP and SVM*. *2010 International Conference on Machine Learning and Cybernetics*, 2010.

[3] Kan Li και Yu Shu Liu. *Fuzzy case-based reasoning: weather prediction*. *Proceedings. International Conference on Machine Learning and Cybernetics*, 2002.

[4] Abhilash Alexander Miranda, Olivier Caelen και Gianluca Bontempi. *Machine Learning for Automated Polyp Detection in Computed Tomography Colonography*. *Machine Learning*, σελίδα 830–850, 2009.

[5] Guorong Wu, Dinggang Shen και Mert Sabuncu. *Machine Learning and Medical Imaging*. *Machine Learning and Medical Imaging*, 2016.

[6] Michael Y. Hu, Murali Shanker και Ming S. Hung. *Predicting Consumer Situational Choice with Neural Networks*. *Neural Networks in Business Forecasting*, 2003.

[7] *What is Deep Learning?* https://machinelearningmastery.com/what-is-deep-learning/, 2016.

[8] Simon S. Haykin. *Neural networks and learning machines*. PHI Learning, 2011.

[9] Chris Nicholson Adam Gibson. *Introduction to Deep Neural Networks*. https://deeplearning4j.org/neuralnet-overview.

[10] http://cs231n.github.io/convolutional-networks/#layers.

[11] Evgeny A. Smirnov, Denis M. Timoshenko και Serge N. Andrianov. *Comparison of Regularization Methods for ImageNet Classification with Deep Convolutional Neural Networks*. *AASRI Procedia*, 6:89–94, 2014.

[12] A. Zisserman O. M. Parkhi, A. Vedaldi. *Deep Face Recognition*. *British Machine Vision Conference*, 2015.

[13] Chris Nicholson Adam Gibson. *A Beginner's Guide to Recurrent Networks and LSTMs*. https://deeplearning4j.org/lstm.html.

[14] J. Donahue S. Venugopalan, M. Rohrbach. *Sequence to Sequence - Video to Text*. *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[15] K. Simonyan και A Zisserman. *VGG-16 Trained on ImageNet Competition Data*. *Very Deep Convolutional Networks for Large-Scale Image Recognition*, 2014.

[16] Andrew Lavin και Scott Gray. *Fast Algorithms for Convolutional Neural Networks*. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Ogier Maitre. *Understanding NVIDIA GPGPU Hardware*. *Natural Computing Series Massively Parallel Evolutionary Computation on GPGPUs*, σελίδα 15–34, 2013.

[18] Rezaur Rahman. *Intel® Xeon Phi™ Coprocessor Architecture and Tools*. 2013.

[19] Clement Farabet Ronan Collobert, Koray Kavukcuoglu. *Torch7: A matlab-like environment for machine learning*. 2011.

[20] Frederic Bastien James Bergstra, Olivier Breuleux. *Theano: a cpu and gpu math expression compiler*. *SciPy*, 4:3, 2010.

[21] Nikhil Ketkar. *Introduction to Tensorflow*. *Deep Learning with Python*, σελίδα 159–194, 2017.

[22] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama και Trevor Darrell. *Caffe: Convolutional Architecture for Fast Feature Embedding*. *arXiv preprint arXiv:1408.5093*, 2014.

[23] *Intel® Xeon Phi™ Processor 7250 (16GB, 1.40 GHz, 68 core) Product Specifications*. https://ark.intel.com/products/94035/Intel-Xeon-Phi-Processor-7250-16GB-1_40-GHz-68-core.

[24] Vadim K. (Intel). *Caffe* Optimized for Intel® Architecture: Applying Modern Code Techniques*. https://software.intel.com/en-us/articles/caffe-optimized-for-intel-architecture-applying-modern-code-techniques, 2017.

[25] Intel. *intel/caffe*. https://github.com/intel/caffe, 2017.

[26] *Caffe*. http://caffe.berkeleyvision.org/.

[27] Simon S. Haykin. *Neural networks and learning machines*. PHI Learning, 2011.

[28] F. Rosenblatt. *The perceptron: A probabilistic model for information storage and organization in the brain.* *Psychological Review*, 65(6):386–408, 1958.

[29] Sergios Theodoridis και Konstantinos Koutroumbas. *Pattern recognition*. Elsevier/Acad. Press, 2012.

[30] *THE NATURE OF CODE*. http://natureofcode.com/book/chapter-10-neural-networks/.

[31] http://cs231n.github.io/neural-networks-1/.

[32] Eva Volna. *Multilayer Neural Networks & Backpropagation Rules*. *Backpropagation Neural Networks and Their Applications*, σελίδα 239–1284.

[33] Chris Nicholson Adam Gibson. *Introduction to Deep Neural Networks*. https://deeplearning4j.org/neuralnet-overview.

[34] Yoshua Bengio. *Deep Learning oDeep Learning of Representations for Unsupervised and Transfer Learningf Representations*. *Deep Learning of Representations for Unsupervised and Transfer Learning*, σελίδα 1–21, 2012.

[35] *ImageNet*. http://www.image-net.org/.

[36] *THE MNIST DATABASE*. http://yann.lecun.com/exdb/mnist/.

[37] F.a. Gers. *Learning to forget: continual prediction with LSTM*. *9th International Conference on Artificial Neural Networks: ICANN 99*, 1999.

[38] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney και Kate Saenko. *Translating Videos to Natural Language Using Deep Recurrent Neural Networks*. *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[39] Yann LeCunn, Leon Bottou και Yoshua Bengio. *GradientBased Learning Applied to Document Recognition*. *Intelligent Signal Processing*, 2009.

[40] Jürgen Schmidhuber. *Deep learning in neural networks: An overview*. *Neural Networks*, 61:85–117, 2015.

[41] https://www.cs.toronto.edu/~kriz/cifar.html.

[42] *How To Solve The Memory Challenges Of Deep Neural Networks*. http://www.topbots.com/how-solve-memory-challenges-deep-learning-neural-networks-graphcore/, 2017.

[43] *Caffe*. http://caffe.berkeleyvision.org/tutorial/net_layer_blob.html.

[44] Adit Deshpande. *The 9 Deep Learning Papers You Need To Know About (Understanding CNNs Part 3)*. https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html.

[45] Alex Krizhevsky, Ilya Sutskever και Geoffrey E Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. *Advances in Neural Information Processing Systems 25*F. Pereira, C. J. C. Burges, L. Bottou και K. Q. Weinberger, επιμελητές, σελίδες 1097–1105. Curran Associates, Inc., 2012.

[46] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke και Andrew Rabinovich. *Going Deeper with Convolutions*. *CoRR*, abs/1409.4842, 2014.

[47] Bvlc. *BVLC/caffe*. https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet, 2017.

[48] Karen Simonyan και Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. *CoRR*, abs/1409.1556, 2014.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren και Jian Sun. *Deep Residual Learning for Image Recognition*. *CoRR*, abs/1512.03385, 2015.

[50] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally και Kurt Keutzer. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size*. *CoRR*, abs/1602.07360, 2016.

[51] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto και Hartwig Adam. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. *CoRR*, abs/1704.04861, 2017.

[52] *Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)*. http://www.image-net.org/challenges/LSVRC/2012/.

[53] *Large Scale Visual Recognition Challenge 2014 (ILSVRC2014)*. http://image-net.org/challenges/LSVRC/2014/index.

[54] *Large Scale Visual Recognition Challenge 2015 (ILSVRC2015)*. http://image-net.org/challenges/LSVRC/2015/.

[55] Alfredo Canziani, Adam Paszke και Eugenio Culurciello. *An Analysis of Deep Neural Network Models for Practical Applications*. *CoRR*, abs/1605.07678, 2016.

[56] Yangqing. *Yangqing/caffe*. https://github.com/Yangqing/caffe/wiki/Convolution-in-Caffe:-a-memo.

[57] *BLAS (Basic Linear Algebra Subprograms)*. http://www.netlib.org/blas/.

[58] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro και Evan Shelhamer. *cuDNN: Efficient Primitives for Deep Learning*. *CoRR*, abs/1410.0759, 2014.

[59] Firas Abuzaid, Stefan Hadjis, Ce Zhang και Christopher Ré. *Caffe con Troll: Shallow Ideas to Speed Up Deep Learning*. *CoRR*, abs/1504.04343, 2015.

[60] A. Vasudevan, A. Anderson και D. Gregg. *Parallel Multi Channel convolution using General Matrix Multiplication*. *2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, σελίδες 19–24, 2017.

[61] *Intel® Xeon® Processor E5-2697 v2*. https://ark.intel.com/products/75283/Intel-Xeon-Processor-E5-2697-v2-30M-Cache-2_70-GHz.

[62] *MCDRAM as High-Bandwidth Memory (HBM) in Knights Landing Processors: Developer's Guide*. https://colfaxresearch.com/knl-mcdram/.

[63] James W. Cooley. *The re-discovery of the fast Fourier transform algorithm*. *Mikrochimica Acta*, 93(1-6):33–45, 1987.

[64] *Controlling Thread Allocation*. https://software.intel.com/en-us/node/694293.

[65] *Environment Variables*. https://gcc.gnu.org/onlinedocs/libgomp/Environment-Variables.html.

Συντομογραφίες - Αρκτικόλεξα - Ακρωνύμια

βλπ	βλέπε
κ.λπ.	και λοιπά
κ.ο.κ	και ούτω καθεξής
ΕΜΠ	Εθνικό Μετσόβιο Πολυτεχνείο
ACC	ACCumulate
ANN	Artificial Neural Network
CBLAS	C Basic Linear Algebra Subprograms
CNN	Convolutional Neural Network
DNN	Deep Neural Memory
GEMM	GEneral Matrix-matrix Multiplication
GPGPUs	General Purpose Graphic Processing
HBM	High Bandwidth Memory
KNL	KNight Landing
MCDRAM	Multi-Channel Dynamic Random Access Memory
NACC	Non ACCumulate
h	input Height
w	input Width
c	input Channels
kh	Kernel Height
kw	Kernel Width
oh	Output Height
ow	Output Width
o	Output Channels - Number of Filters

Απόδοση ξενόγλωσσων όρων

Απόδοση	Ξενόγλωσσος όρος
Αθροιστής	Sum
Αναγνώριση Ψηφίου	Digit Recognition
Ανίχνευση Προσώπου	Face Detection
Απευθείας Συνέλιξη	Direct Convolution
Απομονωτές	Buffers
Αύξηση Διαστάσεων	Lifting
Βαθιά Μηχανική Μάθηση	Deep Machine Learning
Βαθιά Νευρωνικά Δίκτυα	Deep Neural Networks
Βάθος	Depth
Βάρυ	Weights
Βήμα	Stride
Διαμοιρασμός Παραμέτρων	Parameter Sharing
Διαστολή	Dilation
Διαφορικά Επίπεδα	Residual Layers
Εξαγωγή Χαρακτηριστικών	Feature Extraction
Επιβαλλόμενη Μάθηση	Rainforcement Learning
Επιβλεπόμενη Μάθηση	Supervised Learning
Επίπεδα Βραχυχρόνιας Μνήμης	Long-Short Memory Layers LSTM
Επιστήμη Υπολογιστών	Computer Science
Ευαίσθητα Πεδία	Receptive Fields
Εύρος Ζώνης	Bandwidth
Ευρωστία	Robustness
Ιεραρχία Χαρακτηριστικών	Features' Hierarchy
Μη Επιβλεπόμενη Μάθηση	Unsupervised Learning
Μηχανική Μάθηση	Machine Learning
Νευρωνικά Δίκτυα	Neural Networks
Οπισθοδιαδρομή	Backpropagation
Περιοδικά Νευρωνικά Δίκτυα	Recurrent Neural Networks
Πλήρως Συνδεδεμένα Επίπεδο	Fully Connected Layer
Συγγένεια	Affinity
Συγκεντρωτικά Επίπεδα	Pooling Layers
Συνάρτηση Ενεργοποίησης	Activation Function
Συνελικτικά Επίπεδα	Convolutional Layers
Συνελικτικά Νευρωνικά Δίκτυα	Convolutional Neural Networks

Συνένωση Φίλτρων	Filter Concantenation
Ταξινόμηση Εικόνων	Image Classification
Τεχνιτά Νευρωνικά Δίκτυα	Artificial Neural Networks
τοπικότητα	locality
Φάση Εκπαίδευσης	Training Phase
Φάση Εφαρμογής	Inference - Deployment Phase
Φίλτρο	Filter
Υπερεκπαίδευση	Overfitting
Υπερνηματισμός	HyperThreading
υποβιβασμός Διαστάσεων	Lowering