Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Παραλληλοποίηση αλγορίθμων εκπαίδευσης νευρωνικών δικτύων σε μαζικά πολυπύρηνες αρχιτεκτονικές

Διπλωματικη Εργασια

του

ΑΝΔΡΕΑ ΤΡΙΑΝΤΑΦΥΛΛΟΥ

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

Εργαστηριο Υπολογιστικων Συστηματων
Αθήνα, Ιούλιος 2018.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Παραλληλοποίηση αλγορίθμων εκπαίδευσης νευρωνικών δικτύων σε μαζικά πολυπύρηνες αρχιτεκτονικές

Διπλωματικη Εργασια

του

ΑΝΔΡΕΑ ΤΡΙΑΝΤΑΦΥΛΛΟΥ

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

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Ιουλίου 2018

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Γεώργιος Γκούμας	Νεκτάριος Κοζύρης	Νικόλαος Παπασπύρου
Επίκουρος Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπληρωτής Καθηγητής Ε.Μ.Π.

Αθήνα, Ιούλιος 2018

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

(Υπογραφή)

.....

Ανδρεας Τριανταφυλλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η παραλληλοποίηση της εκπαίδευσης ενός αναδρομικού νευρωνικού δικτύου. Ειδικότερα, επιχειρείται η επιτάχυνση της εκπαίδευσης ενός long short-term memory (LSTM) δικτύου που χρησιμοποιείται για την οπτική αναγνώριση χαρακτήρων. Η δυσκολία της παραλληλοποίησης τέτοιου είδους νευρωνικών δικτύων εντοπίζεται στη σειριακή φύση της εκπαίδευσης και της λειτουργίας τους καθώς αντιμετωπίζουν το εκάστοτε δείγμα και είσοδο αντίστοιχα ως μία ακολουθία δεδομένων. Στα πλαίσια της εργασίας παρουσιάζεται η διαδικασία της εκπαίδευσης, υλοποιούνται τρία στάδια παραλληλοποίησης του αλγορίθμου εκπαίδευσης του νευρωνικού δικτύου και αξιολογείται η επίδοσή τους από τα πειραματικά αποτελέσματα που τους αντιστοιχούν.

Λέξεις Κλειδιά

Παραλληλοποίηση, Νευρωνικά Δίκτυα, Αναδρομικά Νευρωνικά Δίκτυα, LSTM, Οπτική Αναγνώριση Χαρακτήρων, OCR, Εκπαίδευση Νευρωνικών Δικτύων, Intel Xeon Phi, OpenMP.

Abstract

This diploma thesis addresses the parallelization of the training of a recurrent neural network. More specifically, it is attempted to accelerate the training of a LSTM (long short-term memory) network. which is used for OCR (optical character recognition) purposes. Training and running of such neural networks manage each sample or input correspondingly as a sequence of data and as a consequence parallelization becomes a difficult task. In the course of this thesis, the procedure of the training is analyzed, three stages of training parallelization are implemented and the performance for each one of the aforementioned stages is evaluated.

Keywords

Parallelization, Neural Networks, Recurrent Neural Networks, Long Short-Term Memory, LSTM, Optical Character Recognition, OCR, Neural Network Training, Intel Xeon Phi, OpenMP.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Γεώργιο Γκούμα ο οποίος μου έδωσε την ευκαιρία να εκπονήσω τη διπλωματική μου εργασία στο εργαστήριο υπολογιστικών συστημάτων σε ένα πολύ ενδιαφέρον θέμα. Επίσης, ευχαριστώ ιδιαίτερα το μετα-διδάκτορα ερευνητή του εργαστηρίου κ. Ιωάννη Βενέτη για την πολύτιμη καθοδήγηση καθώς και για το χρόνο που αφιέρωσε παρέχοντάς μου σημαντικές ιδέες κατά τη διάρκεια της παρούσας εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου, για τη στήριξη που μου παρέχει και τους συμφοιτητές και τους φίλους μου χάρη στους οποίους η ολοκλήρωση των σπουδών μου έγινε πολύ πιο ευχάριστη και παραγωγική.

Περιεχόμενα

Περίληψη **1**

Abstract **2**

Ευχαριστίες **3**

Περιεχόμενα **6**

1 Εισαγωγή **7**

1.1 Αντικείμενο της διπλωματικής 8

1.2 Οργάνωση του τόμου 9

2 Συστήματα παράλληλης επεξεργασίας **11**

2.1 Εισαγωγή 11

2.1.1 Ταξινόμηση του Flynn 12

2.1.2 Οργάνωση μνήμης στις παράλληλες αρχιτεκτονικές 12

2.1.3 Μετρικές αξιολόγησης επίδοσης 15

2.1.4 Νόμος του Amdahl 15

2.2 Παράλληλα προγραμματιστικά μοντέλα 16

2.2.1 Μοντέλο κοινού χώρου διευθύνσεων 16

2.2.2 Μοντέλο ανταλλαγής μηνυμάτων 16

2.2.3 Υβριδικό μοντέλο 17

2.3 Βιβλιοθήκες, γλώσσες και εργαλεία παραλληλοποίησης 17

2.4 Vectorization 19

2.5 Συνεπεξεργαστές 20

2.6 Περιγραφή του συστήματος 21

3 Νευρωνικά δίκτυα **23**

3.1 Τεχνητή νοημοσύνη και νευρωνικά δίκτυα 23

3.2 Είδη νευρωνικών δικτύων 24

3.3 LSTMs, πλεονεκτήματα και χρήσεις 26

4 Οπτική αναγνώριση χαρακτήρων με LSTM νευρωνικά δίκτυα 31
4.1 Οπτική αναγνώριση χαρακτήρων 31
4.2 Ανάλυση της λειτουργίας και του δικτύου της εφαρμογής 32
4.3 Ανάλυση του αλγόριθμου εκπαίδευσης της εφαρμογής 35

5 Παραλληλοποίηση εκπαίδευσης LSTM νευρωνικών δικτύων 39
5.1 Περιγραφή προγραμματιστικού περιβάλλοντος 39
5.2 Εμπόδια και προϋποθέσεις 39
5.3 Πρώτο επίπεδο παραλληλοποίησης 40
5.4 Δεύτερο επίπεδο παραλληλοποίησης 43
5.5 Τρίτο επίπεδο παραλληλοποίησης 47

6 Επίλογος 55
6.1 Σύνοψη και συμπεράσματα 55
6.2 Μελλοντικές επεκτάσεις 55

Βιβλιογραφία 57

Κεφάλαιο 1

Εισαγωγή

Μία από τις βασικές επιδιώξεις της επιστήμης των υπολογιστών είναι η αύξηση της επίδοσης και της διεκπεραιωτικής ικανότητας (throughput) των υπολογιστικών συστημάτων. Στην κατεύθυνση αυτή, τα προηγούμενα χρόνια, σημαντικό ρόλο έπαιξαν η διαρκής αύξηση της συχνότητας του επεξεργαστή και της ταχύτητας της κύριας μνήμης των υπολογιστών. Ωστόσο, εμφανίστηκαν εμπόδια λόγω της προσέγγισης κβαντικών αποστάσεων μεταξύ των transistors που αποτελούν ένα ολοκληρωμένο, της κατανάλωσης ενέργειας και της υπερβολικής παραγωγής θερμότητας, με αποτέλεσμα η ταχύτητα κάθε σειριακού επεξεργαστή (πυρήνα) να αυξάνει πλέον πολύ αργά. Για τη μείωση του χρόνου εκτέλεσης των εφαρμογών κρίνεται, συνεπώς, αναγκαία η αξιοποίηση περισσότερων πυρήνων (παραλληλία σε επίπεδο επεξεργαστών).

Τα σύγχρονα υπολογιστικά συστήματα έχουν στη διάθεσή τους πολλούς επεξεργαστές (έως και χιλιάδες) με περισσότερους από έναν πυρήνες ο καθένας που σε συνδυασμό με τη αύξηση της ταχύτητας της μεταξύ τους επικοινωνίας και το μετασχηματισμό αλγορίθμων ώστε να εκμεταλλεύονται το πλήθος των επεξεργαστών επιτυγχάνουν σημαντικά μικρότερους χρόνους εκτέλεσης. Ωστόσο, η ανάπτυξη παράλληλων αλγορίθμων και ο συγχρονισμός των διεργασιών των διαφορετικών πυρήνων ώστε οι εφαρμογές να κλιμακώνουν αποδοτικά με την αύξηση του αριθμού των πυρήνων συγκαταλέγονται στα κύρια προβλήματα των σύγχρονων συστημάτων παράλληλης επεξεργασίας.

Σημαντική ώθηση στις επιδόσεις των πολυπύρηνων συστημάτων επεξεργασίας αλλά και των προσωπικών υπολογιστών έχουν δώσει οι συνεπεξεργαστές ή αλλιώς επιταχυντές. Οι συνεπεξεργαστές (coprocessors) είναι επεξεργαστές οι οποίοι χρησιμοποιούνται για να βοηθούν τον κεντρικό επεξεργαστή (CPU), εκτελώντας λειτουργίες όπως πράξεις αριθμών κινητής υποδιαστολής, επεξεργασία γραφικών, κρυπτογράφηση δεδομένων και επεξεργασία σημάτων. Αναθέτοντας υπολογιστικά απαιτητικές εργασίες από τον κύριο επεξεργαστή σε έναν ή περισσότερους συνεπεξεργαστές επιτυγχάνεται η επιτάχυνση της επίδοσης του συστήματος. Η GPU ανήκει στην κατηγορία αυτή καθώς είναι υπεύθυνη για την εκτέλεση των πράξεων που απαιτούνται για τη γραφική απεικόνιση δεδομένων. Ακόμα, με τη χρήση FPGAs μπορούν να δημιουργηθούν επιταχυντές για ιδιαίτερες εργασίες όπως η ψηφιακή ανάλυση σήματος. Το 2012 η Intel ανακοίνωσε την κυκλοφορία του Xeon Phi, ενός συνεπεξεργαστή που χρησιμοποιείται ευρέως από υπερυπολογιστές οι οποίοι βρίσκονται στις υψηλότερες θέσεις της

κατάταξης TOP500 [1] για τους ισχυρότερους υπερυπολογιστές διεθνώς. Ο σχεδιασμός του βασίστηκε σε προηγούμενη μελέτη της Intel για τη δημιουργία GPU, ωστόσο έχει από 54 έως 72 πυρήνες ανάλογα με την έκδοση του και με τη δεύτερη γενιά πλέον μπορεί να λειτουργεί και ως αυτόνομος επεξεργαστής [2].

Μία εξίσου σημαντική επιδίωξη της επιστήμης των υπολογιστών είναι η μελέτη και η ανάπτυξη της τεχνητής νοημοσύνης, η οποία αποτελεί ένα από τα ταχύτερα εξελισσόμενα πεδία της επιστήμης αυτής. Η έρευνα στο συγκεκριμένο κλάδο επικεντρώνεται στην απόκτηση λογικής, γνώσης, και αντίληψης ταχύτερης και ποιοτικά ανώτερης από την ανθρώπινη. Κύριος εκφραστής της τεχνητής νοημοσύνης είναι τα τεχνητά νευρωνικά δίκτυα [3]. Τα νευρωνικά δίκτυα είναι υπολογιστικά συστήματα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα, τα οποία συστήνουν τους ανθρώπινους εγκεφάλους και σταδιακά βελτιώνουν την επίδοσή τους στην εκτέλεση συγκεκριμένων διαδικασιών μέσα από την παρατήρηση παραδειγμάτων. Για εφαρμογές όπως η μετάφραση από μία γλώσσα σε μία δεύτερη και η οπτική αναγνώριση χαρακτήρων χρησιμοποιούνται αναδρομικά νευρωνικά δίκτυα, τα οποία περιγράφονται αναλυτικά στο τρίτο κεφάλαιο. Τα αναδρομικά δίκτυα εφαρμόζονται με υψηλά ποσοστά επιτυχίας σε εφαρμογές διαχωρισμού προτύπων όπου οι είσοδοι κι οι έξοδοι των προγραμμάτων είναι ακολουθίες. Ωστόσο, εξαιτίας του γεγονότος ότι η εκμάθηση των νευρωνικών δικτύων γίνεται με τη χρήση παραδειγμάτων η εκπαίδευση τους σε μερικές περιπτώσεις, όπου απαιτείται μεγάλη ακρίβεια ή το πλήθος των διαφορετικών εισόδων είναι εξαιρετικά μεγάλο, μπορεί να διαρκέσει από ώρες μέχρι και μέρες. Τη λύση στο πρόβλημα αυτό μπορούν να δώσουν τα υπάρχοντα πολυπύρηνα συστήματα με την ανάπτυξη των κατάλληλων αλγορίθμων που απαιτούνται για τη παραλληλοποίηση της εκπαίδευσής τους.

1.1 Αντικείμενο της διπλωματικής

Αντικείμενο της διπλωματικής αποτελεί η παραλληλοποίηση της εκπαίδευσης ενός LSTM νευρωνικού δικτύου. Ειδικότερα, στόχοι της παρούσας εργασίας είναι η μελέτη της λειτουργίας των LSTMs, η ανάπτυξη παράλληλων αλγόριθμων για την εκπαίδευση ενός LSTM δικτύου το οποίο χρησιμοποιείται σε εφαρμογή οπτικής αναγνώρισης χαρακτήρων (OCR) και η πειραματική αξιολόγησή των αλγόριθμων αυτών με τη χρήση των συνεπεξεργαστών Intel® Xeon Phi™ 7250 και Intel® Xeon Phi™ 3120P.

Κυρίαρχο κίνητρο για την υλοποίηση των προαναφερθέντων αποτελεί η επιτάχυνση της εκπαίδευσης του συγκεκριμένου τύπου νευρωνικού δικτύου με τη διατήρηση του ίδιου ρυθμού σφάλματος σε σχέση με τη σειριακή έκδοση. Απαραίτητη προϋπόθεση για την υλοποίηση της παράλληλης εκπαίδευσης και τη διεξαγωγή των πειραμάτων είναι η μεταγλώττιση των βιβλιοθηκών που χρησιμοποιούνται από την εφαρμογή ώστε να είναι συμβατές με το λειτουργικό σύστημα των συνεπεξεργαστών. Ακόμα, το γεγονός ότι η εκπαίδευση του συγκεκριμένου τύπου νευρωνικού δικτύου είναι σειριακή ως προς τα δείγματα εκπαίδευσης και ότι κάθε δείγμα θεωρείται ως μία ακολουθία δεδομένων από την οποία το αποτέλεσμα εξαρτάται άμεσα μη επιτρέποντας την κατάτμησή της καθιστούν την παραλληλοποίηση μια δύσκολη διαδικασία.

1.2 Οργάνωση του τόμου

Η παρούσα εργασία αποτελείται από έξι κεφάλαια, το περιεχόμενο των οποίων παρουσιάζεται συνοπτικά στις επόμενες παραγράφους.

Στο τρέχον κεφάλαιο γίνεται μια εισαγωγή στο πρόβλημα και παρατίθενται οι στόχοι της παρούσας διπλωματικής εργασίας.

Στο Κεφάλαιο 2 γίνεται εισαγωγή στα συστήματα παράλληλης επεξεργασίας με την παράθεση του θεωρητικού υποβάθρου που αφορά την παραλληλοποίηση. Ακόμα, παρουσιάζονται οι γλώσσες, οι βιβλιοθήκες και τα εργαλεία που έχουν αναπτυχθεί και χρησιμοποιούνται για το σκοπό αυτό. Στο τέλος του κεφαλαίου αυτού γίνεται και η παρουσίαση των επεξεργαστικών συστημάτων που χρησιμοποιούνται για τους σκοπούς της εργασίας.

Το Κεφάλαιο 3 αφορά τα νευρωνικά δίκτυα. Παρέχει τις απαραίτητες γνώσεις σχετικά με τεχνητά νευρωνικά δίκτυα και τα είδη των διάφορων παραλλαγών τους. Επιπλέον, αναλύονται τα LSTMs (long short-term memory) δίκτυα και οι εφαρμοσμένες χρήσεις τους.

Στο Κεφάλαιο 4 περιέχεται η ανάλυση της εφαρμογής που επιχειρείται να παραλληλοποιηθεί, δηλαδή η οπτική αναγνώριση χαρακτήρων με LSTM δίκτυο. Ακόμα, αναφέρονται κάποια γενικά στοιχεία για την οπτική αναγνώριση χαρακτήρων.

Στο Κεφάλαιο 5 αναλύεται ο τρόπος με τον οποίο επιτυγχάνεται ο σκοπός της διπλωματικής αυτής εργασίας. Πιο συγκεκριμένα, περιγράφεται το προγραμματιστικό περιβάλλον που χρησιμοποιήθηκε και τα βασικά εμπόδια που αντιμετωπίστηκαν. Παρουσιάζονται τα τρία στάδια παραλληλοποίησης που εφαρμόστηκαν στην διαδικασία εκπαίδευσης του νευρωνικού δικτύου καθώς και η επίδοση τους μέσα από πειραματικά αποτελέσματα.

Το Κεφάλαιο 6 περιέχει τα συμπεράσματα και τη συνεισφορά αυτής της διπλωματικής εργασίας καθώς και προτάσεις για εξέλιξη της παραλληλοποίησης του αλγορίθμου εκπαίδευσης που χρησιμοποιείται.

Τέλος, παρατίθεται η σχετική βιβλιογραφία που χρησιμοποιήθηκε για τις ανάγκες της παρούσας εργασίας.

Κεφάλαιο 2

Συστήματα παράλληλης επεξεργασίας

2.1 Εισαγωγή

Από το 1965 ο Gordon Moore είχε προβλέψει πως ο αριθμός των transistors που μπορούν να τοποθετηθούν σε δεδομένη επιφάνεια θα διπλασιάζεται ανά δύο χρόνια. Εκτός από την αύξηση της πυκνότητας των transistors αυξανόταν σταδιακά και η συχνότητα λειτουργίας τους, γεγονός που οδήγησε τον David House στη διαπίστωση ότι η απόδοση των μικροεπεξεργαστών θα διπλασιάζεται κάθε 18 μήνες. Σαν αποτέλεσμα αυτών, άκμασε η παραλληλοποίηση σε επίπεδο εντολών επεξεργαστή (ILP) με τη χρήση βαθύτερων pipelines, out of order εκτέλεσης εντολών και της υπερβαθμωτής οργάνωσης υπολογιστών, παράχθηκαν μεγαλύτερες μνήμες cache, αυξήθηκε η συχνότητα του ρολογιού των επεξεργαστών, αναπτύχθηκαν καλύτερες μεθόδοι πρόβλεψης διακλαδώσεων και εντάχθηκαν στις CPUs διανυσματικές μονάδες επεξεργασίας, λύνοντας τα χέρια των προγραμματιστών αφού παράγονταν ολοένα και πιο ταχείς πυρήνες επεξεργασίας, με συνέπεια οι ίδιες εφαρμογές να εκτελούνται πιο γρήγορα χωρίς προγραμματικό κόστος. Ωστόσο, το 2004 περίπου φτάσαμε στο φράγμα του ILP με την έννοια ότι η απόδοση των επεξεργαστών βελτιωνόταν ελάχιστα συγκριτικά με την κατανάλωση ενέργειας και δεν ήταν δυνατό να αυξηθεί η πυκνότητα των transistors λόγω της προσέγγισης κβαντικών αποστάσεων και της υπερβολικής παραγωγής θερμότητας. Τη λύση στο πρόβλημα αυτό έδωσε η παραλληλοποίηση σε επίπεδο επεξεργαστών, με τη δημιουργία υπολογιστικών συστημάτων που χρησιμοποιούν πολυπύρηνους επεξεργαστές για να βελτιώσουν την απόδοση τους. Η καινοτομία αυτή, βέβαια, δεν ήρθε χωρίς τη δημιουργία νέων εμποδίων και προκλήσεων όπως είναι η απαίτηση χρήσης παράλληλων αλγορίθμων για την πλήρη αξιοποίηση των συστημάτων αυτών, το κόστος συγχρονισμού και επικοινωνίας μεταξύ των επεξεργαστών και ο σχεδιασμός για την αποφυγή αναμονής δεδομένων από την κύρια μνήμη.

2.1.1 Ταξινόμηση του Flynn

Η ταξινόμηση του Flynn [4] είναι ένας διαχωρισμός των αρχιτεκτονικών υπολογιστών, που προτάθηκε από τον Michael J. Flynn το 1966. Οι τέσσερις διαφορετικές κατηγορίες που προέκυψαν σχηματίστηκαν με βάση το πλήθος των παράλληλων εντολών και ροών δεδομένων (data streams) που είναι διαθέσιμα στην εκάστοτε αρχιτεκτονική (σχήμα 2.1):

- **Single Instruction, Single Data stream (SISD)**
  Ένας σειριακός υπολογιστής που δεν υποστηρίζει παραλληλία ούτε σε επίπεδο εντολών ούτε σε επίπεδο δεδομένων. Μία μονάδα ελέγχου που επεξεργάζεται μία ροή εντολών από τη μνήμη και δημιουργεί σήματα ελέγχου για να κατευθύνει τον επεξεργαστή να εκτελέσει μία εντολή κάθε φορά σε μία και μοναδική ροή δεδομένων. Στην κατηγορία αυτή ανήκει ο παλιός προσωπικός υπολογιστής με έναν επεξεργαστή.

- **Single Instruction, Multiple Data streams (SIMD)**
  Ένας υπολογιστής με μία ροή εντολών και πολλαπλές ροές δεδομένων, που έχει τη δυνατότητα να εκτελεί την ίδια εντολή σε πολλαπλά δεδομένα. Η πρώτη χρήση SIMD εντολών έγινε από τους διανυσματικούς υπερυπολογιστές (vector supercomputers) στις αρχές της δεκαετίας του 1970, οι οποίοι μπορούσαν να εκτελέσουν κάποιες πράξεις σε διανύσματα δεδομένων, δηλαδή σε μονοδιάστατους πίνακες από στοιχεία δεδομένων.

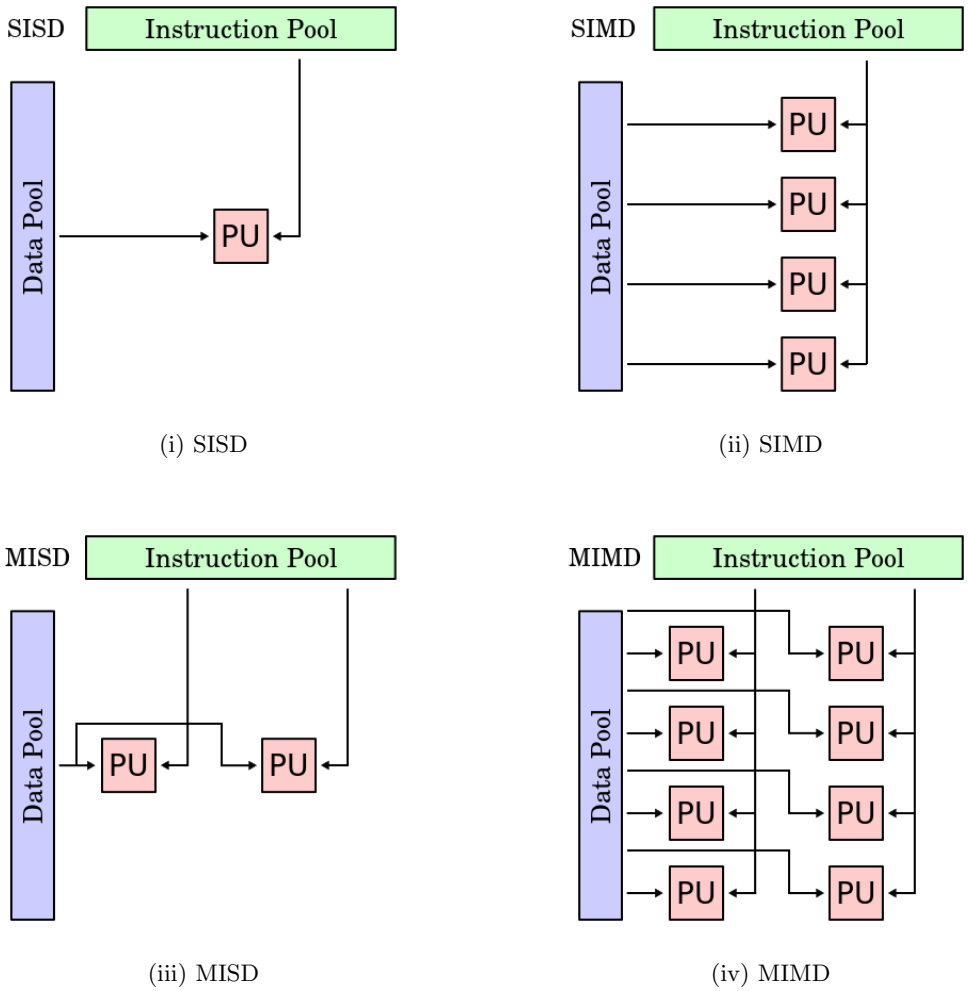
- **Multiple Instruction, Single Data stream (MISD)**
  Πολλαπλές εντολές εκτελούνται στην ίδια ροή δεδομένων. Η αρχιτεκτονική αυτή συναντάται σπάνια και χρησιμοποιείται κυρίως για τον έλεγχο της ανοχής σφαλμάτων των εφαρμογών, όπου διαφορετικά συστήματα επεξεργάζονται την ίδια ροή δεδομένων και πρέπει να έχουν το ίδιο αποτέλεσμα. Στην κατηγορία αυτή ανήκει το Space Shuttle Digital Flight Control System της NASA.

- **Multiple Instruction, Multiple Data streams (MIMD)**
  Πολλαπλοί αυτόνομοι επεξεργαστές εκτελούν εντολές σε διαφορετικά δεδομένα. Είναι η πιο συχνή και ενδιαφέρουσα αρχιτεκτονική για τα σύγχρονα συστήματα παράλληλης επεξεργασίας. Στην κατηγορία αυτή ανήκουν τα κατανεμημένα συστήματα και οι πολυπύρηνοι υπολογιστές.

2.1.2 Οργάνωση μνήμης στις παράλληλες αρχιτεκτονικές

Τα πολυπύρηνα συστήματα χωρίζονται με βάση την οργάνωση της κύριας μνήμης τους σε **κοινής μνήμης**, **κατανεμημένης μνήμης** και **υβριδικά** τα οποία συνδυάζουν και τις δύο αρχιτεκτονικές.

Τα συστήματα κοινής μνήμης δημιουργούνται με τη διασύνδεση πολλαπλών επεξεργαστών με την ίδια μνήμη και τη χρήση, συνεπώς, ενός κοινού χώρου διευθύνσεων. Αν όλα τα ξεχωριστά τμήματα μνήμης προσπελαύνονται σε ίσο χρονικό διάστημα από όλους τους επεξεργαστές τότε το σύστημα καλείται ομοιόμορφης πρόσβασης στη μνήμη (Uniform Memory Access -



(i) SISD (ii) SIMD

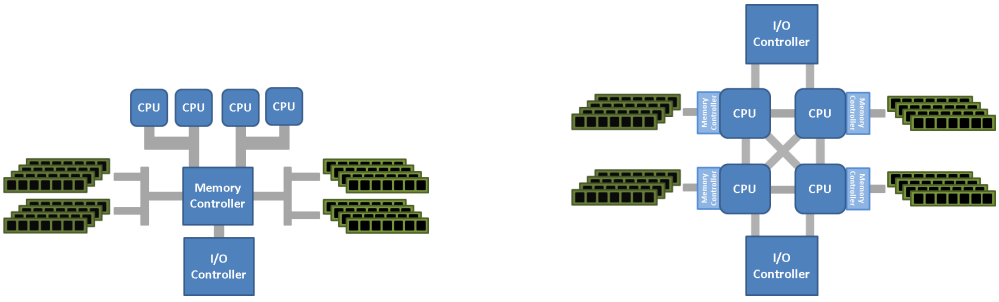
(iii) MISD (iv) MIMD

Σχήμα 2.1: Ταξινόμηση κατά Flynn.

UMA). Αν σε ορισμένα τμήματα μνήμης κάποιοι από τους επεξεργαστές έχουν γρηγορότερη πρόσβαση σε σχέση με τους υπόλοιπους τότε το σύστημα καλείται μη ομοιόμορφης πρόσβασης στη μνήμη (Non-Uniform Memory Access - **NUMA**). Ένα σύστημα NUMA το οποίο υποστηρίζει την επικοινωνία μεταξύ των επεξεργαστών ώστε όσες από τις cache μνήμες τους αναφέρονται σε κοινό τμήμα δεδομένων της κύριας μνήμης να έχουν διαρκώς ίδια δεδομένα, καλείται NUMA με συνάφεια κρυφής μνήμης (cache-coherent NUMA - **cc-NUMA**).

Χάρη στον κοινό χώρο διευθύνσεων τα συστήματα κοινής μνήμης θεωρείται ότι υποστηρίζουν καλύτερα και ευκολότερα τον παράλληλο προγραμματισμό καθώς υπάρχουν προγραμματιστικά μοντέλα για αυτού του είδους τα συστήματα τα οποία επιτρέπουν με σχετικά λίγες προσθήκες την μετατροπή των σειριακών προγραμμάτων σε παράλληλα. Ωστόσο, η αντιμετώπιση των καταστάσεων ανταγωνισμού (race conditions) και της αλληλοεξάρτησης των δεδομένων (data dependencies) για την ορθή λειτουργία της εφαρμογής η οποία επιχειρείται να παραλληλοποιηθεί είναι μία διαδικασία δύσκολη και επιρρεπής σε λάθη, η οποία απαιτεί ιδιαίτερη προσοχή.

Οι βιομηχανικές παράλληλες εφαρμογές, όμως, έχουν τεράστιες ανάγκες σε υπολογιστική

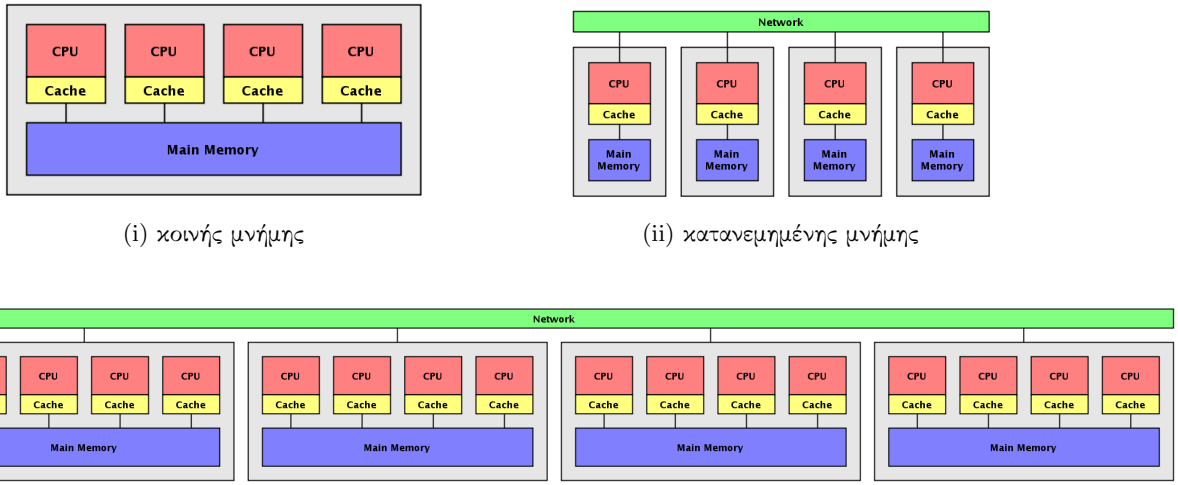


(i) UMA (ii) NUMA

Σχήμα 2.2: Κατηγορίες αρχιτεκτονικών κοινής μνήμης.

ισχύ και απαιτούν χιλιάδες έως και εκατομμύρια επεξεργαστές, μία απαίτηση που δεν μπορεί να ικανοποιηθεί από συστήματα κοινής μνήμης, καθώς η επικοινωνία μεταξύ των επεξεργαστών και της κύριας μνήμης θα γίνει πολύ πιο αργή και προϋποθέτει ιδιαίτερα περίπλοκο hardware. Για τους λόγους αυτούς, είναι προτιμότερο κάθε επεξεργαστής να έχει τη δική του ξεχωριστή μνήμη. Για τη δημιουργία τέτοιων συστημάτων, κατανεμημένης μνήμης, οι επεξεργαστές συνδέονται με διασυνδετικό δίκτυο μεταφοράς δεδομένων όπως φαίνεται στο σχήμα 2.3.ii. Στις αρχιτεκτονικές κατανεμημένης μνήμης δεν υπάρχει κοινός χώρος διευθύνσεων και συνεπώς οι επεξεργαστές επικοινωνούν διαμέσου ενός δικτύου κυρίως με τη μεταξύ τους ανταλλαγή μηνυμάτων, που υλοποιούνται με την κλήση βιβλιοθηκών επικοινωνίας οι οποίες βρίσκονται απαραίτητα και στον αποστολέα και στον παραλήπτη.

Η ευρεία διάδοση των πολυπύρηνων υπολογιστών σε συνδυασμό με την καλή κλιμακωσιμότητα των συστημάτων κατανεμημένης μνήμης έχει οδηγήσει στη δημιουργία μίας υβριδικής αρχιτεκτονικής παραλληλίας δύο επιπέδων, όπου συστήματα κοινής μνήμης διασυνδέονται και σχηματίζουν ένα μεγάλης κλίμακας παράλληλο σύστημα, όπως απεικονίζεται στο σχήμα 2.3.iii.



(i) κοινής μνήμης (ii) κατανεμημένης μνήμης

(iii) υβριδικό σύστημα

Σχήμα 2.3: Κατηγορίες αρχιτεκτονικών με βάση την οργάνωση μνήμης.

2.1.3 Μετρικές αξιολόγησης επίδοσης

Οι ακόλουθες μετρικές αξιολόγησης επίδοσης είναι σημαντικές για τον καθορισμό της αποδοτικότητας μίας παράλληλης εφαρμογής:

- **Επιτάχυνση (speedup)** $S = T_s/T_p$
  όπου T_s είναι ο χρόνος εκτέλεσης του καλύτερου σειριακού αλγορίθμου και T_p ο χρόνος του παράλληλου αλγορίθμου. Η μετρική αυτή δείχνει πόσες φορές πιο γρήγορο είναι το παράλληλο πρόγραμμα από το αντίστοιχο σειριακό.

  Γραμμική ή τέλεια επιτάχυνση έχουμε όταν $S = p$, όπου p ο αριθμός των επεξεργαστών στους οποίους εκτελείται η παράλληλη εφαρμογή. Συνήθως, ισχύει $S \leq p$, ωστόσο υπάρχουν κάποιες περιπτώσεις όπου $S > p$ (υπεργραμμική επιτάχυνση) και χρήζουν ιδιαίτερης ερμηνείας και προσοχής αν προκύψει στις μετρήσεις μας.

- **Αποδοτικότητα (efficiency)** $E = S/p$
  Η αποδοτικότητα δείχνει πόσο επιτυχημένη είναι η παραλληλοποίηση εκφράζοντας τι ποσοστό του χρόνου κάθε επεξεργαστής κάνει χρήσιμη δουλειά. Στις περισσότερες εφαρμογές ισχύει $E \leq 1$.

- **Κλιμακωσιμότητα (scalability)**
  Η κλιμακωσιμότητα εκφράζει ποιοτικά την ικανότητα ενός προγράμματος να βελτιώνει την επίδοσή του με την προσθήκη επιπλέον επεξεργαστών (πόρων). Δεν υπάρχει αυστηρός ορισμός για την κλιμακωσιμότητα, όταν όμως ένα πρόγραμμα έχει γραμμική επιτάχυνση λέγεται ότι κλιμακώνει καλά ενώ όταν ένα πρόγραμμα δεν βελτιώνει ή ακόμα μειώνει την απόδοση του για επεξεργαστές $p > p_0$ λέγεται ότι η κλιμακωσιμότητα τερματίζει στους p_0 επεξεργαστές.

2.1.4 Νόμος του Amdahl

Ο νόμος του Amdahl [5] είναι ένας θεμελιώδης νόμος για τα συστήματα παράλληλης επεξεργασίας και χρησιμοποιείται για να βρεθεί η μέγιστη αναμενόμενη βελτίωση σε ένα σύστημα όταν βελτιώνεται η επίδοση κάποιου μέρους του. Συχνά, στα παράλληλα συστήματα χρησιμοποιείται για τον υπολογισμό της μέγιστης θεωρητικής επιτάχυνσης ενός συστήματος με πολλαπλούς επεξεργαστές.

Πιο συγκεκριμένα, αν f είναι το κλάσμα ενός προγράμματος το οποίο δε μπορεί να παραλληλοποιηθεί και πρέπει να εκτελεστεί σειριακά και 1 - f το κλάσμα το οποίο μπορεί να παραλληλοποιηθεί τότε ο καλύτερος χρόνος εκτέλεσης του παράλληλου προγράμματος σε p επεξεργαστές δίνεται από τη σχέση:

$$T_p = f \cdot T_s + \frac{(1-f) \cdot T_s}{p}$$

και η μέγιστη επιτάχυνση από τη σχέση:

$$S_{max} = \frac{T_s}{T_p} = \frac{1}{f + \frac{1-f}{p}}$$

Παρ΄ όλη την απλότητα του, ο νόμος του Amdahl αποτελεί τον πιο σημαντικό οδηγό για την παραλληλοποίηση και γενικότερα για τη βελτιστοποίηση κώδικα. Πιο συγκεκριμένα, αρχικά πρέπει να παραλληλοποιούνται τα τμήματα του κώδικα που καταλαμβάνουν το μεγαλύτερο ποσοστό του χρόνου εκτέλεσης και να αναζητείται παραλληλία σε όλη την έκταση του κώδικα, καθώς, για παράδειγμα, αν 10% του κώδικα πρέπει να εκτελεστεί σειριακά τότε το μέγιστο speedup που μπορεί να επιτευχθεί είναι 10 ανεξάρτητα από τον αριθμό των επεξεργαστών που θα χρησιμοποιηθούν.

2.2 Παράλληλα προγραμματιστικά μοντέλα

Για κάθε μία από τις προαναφερθείσες κατηγορίες που προκύπτουν με βάση την οργάνωση της μνήμης των παράλληλων συστημάτων υπάρχουν αντίστοιχα προγραμματιστικά μοντέλα, με βασικό χαρακτηριστικό τον τρόπο ανταλλαγής δεδομένων μεταξύ των πολλαπλών οντοτήτων που συμμετέχουν στην εκτέλεση της παράλληλης εφαρμογής. Κάθε μοντέλο εφαρμόζεται καλύτερα στην αντίστοιχη αρχιτεκτονική, ωστόσο είναι δυνατό να χρησιμοποιηθεί και σε οποιαδήποτε άλλη είτε με περισσότερο προγραμματιστικό κόπο είτε με μειωμένη απόδοση.

2.2.1 Μοντέλο κοινού χώρου διευθύνσεων

Στο μοντέλο αυτό οι εκτελούμενες παράλληλες εργασίες (tasks) έχουν πρόσβαση σε κοινή μνήμη [6]. Κύριο πλεονέκτημα του αποτελεί το γεγονός πως η πρόσβαση σε κοινά δεδομένα γίνεται με απλές εντολές ανάγνωσης και εγγραφής στη μνήμη, διευκολύνοντας με τον τρόπο αυτό την ταχεία παραγωγή κώδικα. Ωστόσο, λόγω της ταυτόχρονης πρόσβασηςστην κοινή μνήμη δημιουργούνται καταστάσεις ανταγωνισμού αυξάνοντας την πιθανότητα σφαλμάτων στο αναπτυσσόμενο πρόγραμμα και επιβάλλοντας τη χρήση σχημάτων συγχρονισμού για την επίλυση τους, επιφέροντας επιβαρύνσεις στην απόδοση. Η εγγενής υποστήριξη των λειτουργικών συστημάτων μηχανισμών κοινής πρόσβασης στη μνήμη για την επικοινωνία μεταξύ των παράλληλων εργασιών καθιστά το συγκεκριμένο προγραμματιστικό μοντέλο ως το πλέον κατάλληλο για συστήματα κοινής μνήμης. Εξαιτίας της απαίτησης για κοινή μνήμη το συγκεκριμένο προγραμματιστικό μοντέλο έχει περιορισμένη δυνατότητα κλιμάκωσης.

2.2.2 Μοντέλο ανταλλαγής μηνυμάτων

Το μοντέλο ανταλλαγής μηνυμάτων υποθέτει ότι δεν υπάρχουν κοινά δεδομένα μεταξύ των διεργασιών και συνεπώς κάθε διεργασία έχει τη δική της μνήμη, όπου έχει αποκλειστική πρόσβαση στα δεδομένα [7]. Η διαδιεργασιακή επικοινωνία γίνεται μόνο με την ανταλλαγή μηνυμάτων και μπορεί να είναι είτε σύγχρονη είτε ασύγχρονη. Η επικοινωνία διαχωρίζεται επίσης σε επικοινωνία σημείο-προς-σημείο, κατά την οποία δύο διεργασίες ανταλλάσσουν κάποιο μήνυμα και συλλογική επικοινωνία, όπου περισσότερες από δύο διεργασίες λαμβάνουν μέρος στην ανταλλαγή μηνυμάτων. Το μοντέλο ανταλλαγής μηνυμάτων υλοποιείται τόσο σε συστήματα κοινής όσο και σε συστήματα κατανεμημένης μνήμης. Η απουσία κοινής μνήμης έχει

σαν αποτέλεσμα την παραγωγή προγραμμάτων με λιγότερα και πιο εμφανή σφάλματα κάνοντας την εκσφαλμάτωση πολύ πιο εύκολη. Ωστόσο, ο προγραμματισμός με τη χρήση του συγκεκριμένου μοντέλου είναι μία δύσκολη διαδικασία, καθώς ο διαμοιρασμός των δεδομένων, η συλλογή αποτελεσμάτων και ο σχεδιασμός για την επικοινωνία μεταξύ των διεργασιών απαιτούν ιδιαίτερη προσοχή και προσπάθεια.

2.2.3 Υβριδικό μοντέλο

Για τις εφαρμογές που προορίζονται να εκτελεστούν σε υπερυπολογιστές ή συστοιχίες πολλαπλών επεξεργαστών χρησιμοποιείται το υβριδικό προγραμματιστικό μοντέλο, το οποίο συνδυάζει τα δύο προηγούμενα. Η επικοινωνία μεταξύ των διεργασιών που ανήκουν στον ίδιο υπολογιστικό κόμβο, συστοιχία επεξεργαστών με κοινή μνήμη, υλοποιείται μέσω της διαμοιραζόμενης μνήμης, ενώ οι διεργασίες που βρίσκονται σε διαφορετικούς κόμβους επικοινωνούν μέσω του δικτύου διασύνδεσης, με ανταλλαγή μηνυμάτων [8].

2.3 Βιβλιοθήκες, γλώσσες και εργαλεία παραλληλοποίησης

Καθώς η ανάγκη για την παραγωγή παράλληλων προγραμμάτων μεγάλωνε άρχισαν να δημιουργούνται γλώσσες, βιβλιοθήκες και εργαλεία για να διευκολυνθεί η πιο γρήγορη και ποιοτική παραγωγή παράλληλου κώδικα. Τα βασικότερα από αυτά παρουσιάζονται παρακάτω.

- **POSIX Threads**
  Τα POSIX Threads (Pthreads) [9] είναι ένα πρότυπο POSIX για νήματα που προσδιορίζει μια προγραμματιστική διεπαφή (Application Programming Interface ή API) για τη δημιουργία και τη διαχείριση νημάτων. Οι υλοποιήσεις του συγκεκριμένου προγραμματιστικού περιβάλλοντος είναι διαθέσιμες σε πολλά λειτουργικά συστήματα τύπου UNIX. Ο προγραμματισμός με τη χρήση των Pthreads μοιάζει με τον προγραμματισμό λειτουργικών συστημάτων, καθώς κάνει κλήσεις συστήματος χαμηλού επιπέδου που παρέχονται από το λειτουργικό. Τα προγράμματα με Pthreads ακολουθούν το προγραμματιστικό μοντέλο κοινής μνήμης και η ανάθεση εργασιών στα νήματα είναι αποκλειστική ευθύνη του προγραμματιστή. Συνολικά, τα Pthreads είναι ένα ισχυρό εργαλείο για τον προγραμματισμό συστημάτων κοινής μνήμης που δίνει τον πλήρη έλεγχο στον προγραμματιστή, όμως η ανάπτυξη προγραμμάτων με αυτά είναι αρκετά αντιπαραγωγική.

- **OpenMP**
  Το OpenMP (Open Multi-Processing) [10] είναι ένα προγραμματιστικό περιβάλλον το οποίο υποστηρίζει τον προγραμματισμό εφαρμογών για παράλληλα συστήματα κοινής μνήμης. Προγράμματα που χρησιμοποιούν το OpenMP γράφονται σε C, C++ ή Fortran και υποστηρίζονται από τις περισσότερες αρχιτεκτονικές επεξεργαστών και λειτουργικά συστήματα. Σε αντίθεση με τα Pthreads όπου ο παράλληλος κώδικας διαφέρει ριζικά

από τον αντίστοιχο σειριακό, ένα μεγάλο πλεονέκτημα του OpenMP είναι ότι ο παράλληλος κώδικας διαφέρει ελάχιστα από το σειριακό στον οποίο έχουν προστεθεί κάποιες οδηγίες (directives), οι οποίες αγνοούνται από τον compiler όταν δεν τις υποστηρίζει καθώς είναι σε μορφή σχολίων στην εκάστοτε γλώσσα. Το OpenMP λειτουργεί με τη δημιουργία νημάτων τα οποία μπορούν να κατανεμηθούν σε έναν ή περισσότερους πυρήνες και είναι υπεύθυνα για την εκτέλεση των εντολών που τους ανατίθενται και την εξασφάλιση των απαραίτητων πόρων. Επιπλέον, το συγκεκριμένο προγραμματιστικό περιβάλλον υποστηρίζει την παραλληλοποίηση με τη χρήση έργων (tasks) καθώς και την εμφωλευμένη παραλληλοποίηση. Συνολικά, είναι ένα εργαλείο που χρησιμοποιείται ευρέως στη βιομηχανία καθώς απλοποιεί τον προγραμματισμό για συστήματα κοινής μνήμης αλλά χρειάζεται προσοχή στο συγχρονισμό και στις καταστάσεις ανταγωνισμού των νημάτων.

- **Cilk**
  Η Cilk [11] είναι μία επέκταση της C για την υποστήριξη παράλληλου προγραμματισμού σε συστήματα κοινής μνήμης. Αναπτύχθηκε αρχικά το 1994 στο εργαστήριο της επιστήμης υπολογιστών του Massachusetts Institute of Technology, αλλά στη συνέχεια απέκτησε τα δικαιώματα χρήσης της η Intel και παρήγαγε την εμπορική της έκδοση, η οποία υποστηρίζει τις γλώσσες C και C++ και διανέμεται με την ονομασία Intel Cilk Plus [12]. Η συγκεκριμένη γλώσσα παρέχει δύο βασικές λέξεις-κλειδιά για τη σηματοδότηση της παραλληλίας σε ένα πρόγραμμα, το spawn για να δείξει διαδικασίες που μπορούν να εκτελεστούν παράλληλα με άλλες και το sync που είναι ένα σημείο για το συγχρονισμό των διαδικασιών που εκτελούνται παράλληλα. Η Cilk δημιουργεί και διαχειρίζεται παράλληλα έργα (tasks), υποστηρίζοντας το μηχανισμό work-stealing για τη δυναμική εξισορρόπηση του φόρτου των επεξεργαστών [13]. Συνηθισμένη τεχνική για τα προγράμματα σε Cilk είναι η αναδρομική παραλληλοποίηση, όπου ένα πρόβλημα διασπάται αναδρομικά σε μικρότερα παράλληλα υποπροβλήματα.

- **Threading Building Blocks**
  Τα Threading Building Blocks ή αλλιώς TBBs [14] είναι μια βιβλιοθήκη της C++ που δημιουργήθηκε από την Intel για την παραγωγή πολυνηματικών προγραμμάτων. Τα TBBs παρέχουν ένα πλούσιο σύνολο εντολών για την υπόδειξη παραλληλίας σε διαφορετικές μορφές και επίπεδα. Βασικές εντολές είναι το parallel for, το parallel reduce και το parallel scan. Ωστόσο, η συγκεκριμένη βιβλιοθήκη παρέχει και πιο προχωρημένες εντολές και δομές δεδομένων και υιοθετεί τη φιλοσοφία της Cilk για τη διαχείριση των tasks και την ισοκατανομή του φόρτου εργασίας μεταξύ αυτών.

- **MPI**
  Το MPI (Message Passing Interface) [15] είναι μία βιβλιοθήκη που αναπτύχθηκε με τη συμβολή τόσο της βιομηχανίας όσο και ακαδημαϊκών ερευνητών με σκοπό την εύρωστη λειτουργία της σε μεγάλη ποικιλία παράλληλων συστημάτων. Το MPI ακολουθεί το προγραμματιστικό μοντέλο ανταλλαγής μηνυμάτων και χρησιμοποιείται εκτεταμένα για

την παραγωγή παράλληλων προγραμμάτων. Ακόμα, η πλειονότητα των βιομηχανικών εφαρμογών υψηλών απαιτήσεων που εκτελούνται σε μεγάλης κλίμακας παράλληλα συστήματα υλοποιούνται με MPI. Η βιβλιοθήκη αυτή υποστηρίζεται στη Fortran, τη C και τη C++. Οι MPI διεργασίες δημιουργούνται σε διαφορετικούς υπολογιστικούς κόμβους, εκτελούνται αυτόνομα σε ξεχωριστό χώρο διευθύνσεων και επικοινωνούν αποκλειστικά με τη χρήση μηνυμάτων.

- **CUDA**
  Όλα τα προαναφερθέντα εργαλεία παραλληλοποίησης προγραμμάτων χρησιμοποιούνται για την ανάπτυξη παράλληλων προγραμμάτων που εκτελούνται σε CPUs και συνεπεξεργαστές. Ωστόσο, το 2001 έγιναν οι πρώτες προσπάθειες παραλληλοποίησης απλών εφαρμογών με τη χρήση GPUs, καθώς υποστηρίζουν την παράλληλη εκτέλεση πολλαπλών νημάτων. Το 2007 η nVIDIA δημιούργησε την CUDA [16], μία πλατφόρμα ανάπτυξης παράλληλων εφαρμογών που επιτρέπει σε μία επεξεργαστική μονάδα γραφικών να εκτελεί παράλληλες εφαρμογές. Η χρήση της μπορεί να γίνει στις πιο διαδεδομένες γλώσσες προγραμματισμού όπως C, C++, Python και MATLAB μέσω επεκτάσεων αυτών που υποδεικνύουν την ικανότητα παραλληλοποίησης με τη χρήση κάποιων βασικών εντολών. Με τον τρόπο αυτό δημιουργήθηκε μια νέα τάση στην παράλληλη επεξεργασία με τη χρήση καρτών γραφικών, η οποία καλείται GPGPU (General-Purpose computing on Graphics Processing Units).

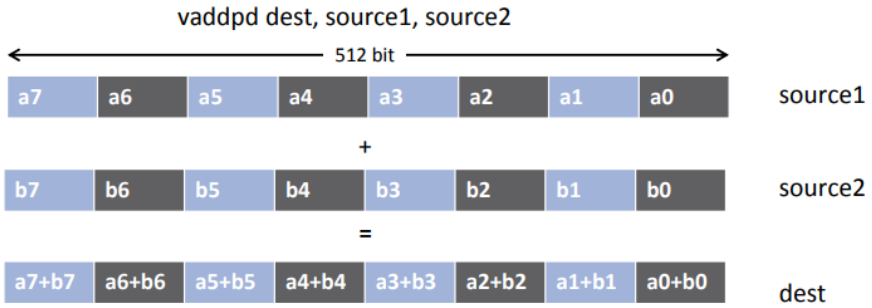
2.4 Vectorization

Τα σύγχρονα συστήματα επεξεργασίας έχουν πολλά επίπεδα παραλληλίας με σκοπό την ταχύτερη εκτέλεση των προγραμμάτων. Συνεπώς, γίνονται προσπάθειες για εκμετάλλευση παραλληλίας όπου αυτό είναι δυνατό, από την παράλληλη εκτέλεση εντολών του επεξεργαστή μέχρι την παράλληλη εκτέλεση πολλαπλών νημάτων σε περισσότερους του ενός επεξεργαστών. Στην κατεύθυνση αυτή έχουν προστεθεί όλο και μεγαλύτερης χωρητικότητας καταχωρητές και επεξεργαστικές μονάδες σε κάθε επεξεργαστή με σκοπό την εκτέλεση πράξεων όχι σε βαθμωτά μεγέθη αλλά σε διανύσματα (vectors). Για παράδειγμα, με τη βοήθεια του σχήματος 2.4, η εκτέλεση της πρόσθεσης 8 αριθμών κινητής υποδιαστολής διπλής ακρίβειας των 64 bits με άλλους 8 γίνεται σε ένα χρόνο σε αντίθεση με την παραδοσιακή τακτική, όπου η κάθε πράξη θα εκτελούνταν ξεχωριστά 8 φορές, μία για κάθε ζεύγος.

Με τον όρο διανυσματοποίηση (vectorization), λοιπόν, αποκαλείται η εύρεση και εκμετάλλευση δεδομένων με την ομαδοποίηση των οποίων επιτυγχάνεται ο ενιαίος χειρισμός τους. Για την επίτευξή του είναι αναγκαία η μεταβολή του υλικού με την προσθήκη εξειδικευμένων εντολών, επεξεργαστικών μονάδων και καταχωρητών του επεξεργαστή. Μερικές από τις πιο γνωστές επεκτάσεις εντολών επεξεργαστή (instruction sets) που υποστηρίζουν Single Instruction Multiple Data (SIMD) εντολές είναι η FMA3 και οι SSE4 και AVX-512 από την Intel. Ακόμα, οι VPUs (Vector Processing Units ή μονάδες επεξεργασίας διανυσμάτων) είναι μονάδες επεξεργασίας που υποστηρίζουν ένα σύνολο εντολών επεξεργαστή μερικές από τις

οποίες αφορούν πράξεις μεταξύ διανυσμάτων δεδομένων, σε αντίθεση με τις παραδοσιακές μονάδες επεξεργασίας οι εντολές των οποίων εκτελούνται σε μοναδικά στοιχεία δεδομένων.

Για την αποδοτική αξιοποίηση των προηγούμενων τα δεδομένα στα οποία εκτελούνται οι πράξεις πρέπει να βρίσκονται σε συνεχόμενες θέσεις μνήμης και να μην υπάρχουν μεταξύ αυτών εξαρτήσεις. Ακόμα, η αυτόματη παραγωγή διανυσματικού κώδικα υποστηρίζεται από κάποιους μεταγλωττιστές για απλές εντολές. Όμως για πιο περίπλοκους αλγόριθμους τόσο οι μεταγλωττιστές όσο και κάποια εργαλεία παραλληλοποίησης, όπως το OpenMP και η Cilk, προσφέρουν εξειδικευμένες εντολές (intrinsics), για την υπόδειξη της δυνατότητας χρήσης SIMD εντολών, τις οποίες μπορεί με προσοχή να χρησιμοποιεί ο προγραμματιστής.



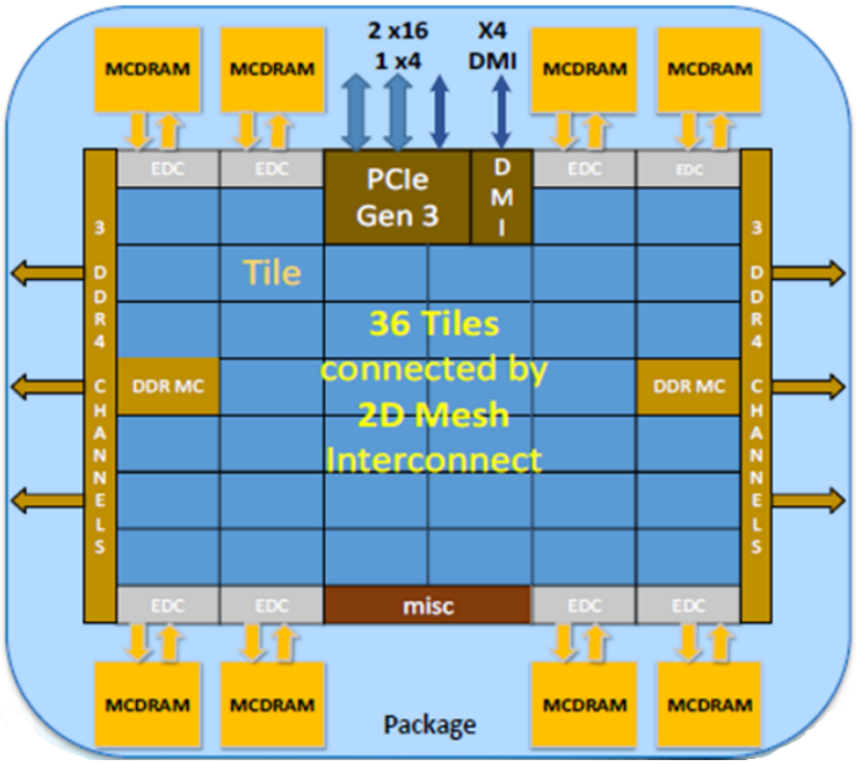
Σχήμα 2.4: Οκτώ ταυτόχρονες προσθέσεις αριθμών κινητής υποδιαστολής διπλής ακρίβειας 64 bits.

2.5 Συνεπεξεργαστές

Ως συνεπεξεργαστής (coprocessor) καλείται ένας επεξεργαστής ο οποίος υποβοηθά της λειτουργίες του κεντρικού επεξεργαστή (CPU). Οι συνεπεξεργαστές συμβάλλουν στην επιτάχυνση λειτουργιών όπως οι πράξεις αριθμών κινητής υποδιαστολής, η επεξεργασία σήματος, η υποστήριξη γραφικών, η επεξεργασία λέξεων, η κρυπτογράφηση δεδομένων και η εξυπηρέτηση περιφερειακών συσκευών. Ακόμα, πρόσφατα έχουν κυκλοφορήσει και ανακοινωθεί συνεπεξεργαστές που εξειδικεύονται και επιταχύνουν εφαρμογές τεχνητής νοημοσύνης [17] [18] [19]. Η μονάδα επεξεργασίας γραφικών (GPU) είναι ίσως ο πιο γνωστός συνεπεξεργαστής ο οποίος με τη χρήση εκατοντάδων ως και χιλιάδων πυρήνων επεξεργασίας [20] επιταχύνουν τον υπολογισμό των απαραίτητων πράξεων για την απεικόνιση γραφικών υψηλής ποιότητας. Επιπλέον, συνεπεξεργαστές χρησιμοποιούνται και για τη σύνθεση υπερυπολογιστών όπου εκτελούνται υπολογιστικά απαιτητικές εφαρμογές. Για παράδειγμα, ο δεύτερος πιο ισχυρός υπερυπολογιστής Tianhe-2 (MilkyWay-2) [21] αποτελείται εκτός των άλλων και από 48.000 συνεπεξεργαστές Intel® Xeon Phi™ 31S1P [22].

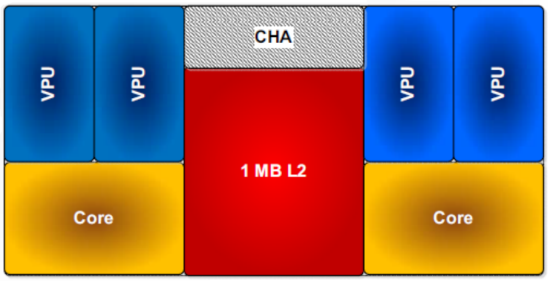
2.6 Περιγραφή του συστήματος

Για τους σκοπούς της παρούσας εργασίας χρησιμοποιήθηκαν οι συνεπεξεργαστές Intel® Xeon Phi™ 7250 και Intel® Xeon Phi™ 3120P. Πιο συγκεκριμένα ο Xeon Phi 7250 [23] ανήκει στη δεύτερη γενιά της συγκεκριμένης σειράς και βασίζεται στην αρχιτεκτονική MIC (Many Integrated Core) σε τεχνολογία 14nm. Ο συγκεκριμένος επιταχυντής (accelerator) έχει 68 πυρήνες οι οποίοι υποστηρίζουν 4 νήματα έκαστος προσφέροντας συνολικά 272 νήματα. Η συχνότητα επεξεργασίας των πυρήνων είναι 1.40 GHz. Με τη βοήθεια του σχήματος 2.5, παρατηρούμε ότι οι πυρήνες είναι ομαδοποιημένοι σε tiles. Τα tiles του μοντέλου αυτού είναι 34 σε αντίθεση με τα 36 που απεικονίζονται στο σχήμα καθώς αυτό δείχνει την οργάνωση ενός άλλου μοντέλου (Intel® Xeon Phi™ 7290) της ίδιας σειράς.



Σχήμα 2.5: Οργάνωση του συνεπεξεργαστή Xeon Phi.

Κάθε ένα από τα 34 tiles, όπως φαίνεται στο σχήμα 2.6, περιέχει 2 πυρήνες και 1 MB L2 cache. Σε κάθε πυρήνα αντιστοιχούν επίσης 2 VPUs μεγέθους 512 bits. Ακόμα, η νέα γενιά του Xeon Phi με κωδική ονομασία Knights Landing υποστηρίζει μία νέα τεχνολογία μνήμης, τη Multi-Channel DRAM (MCDRAM). Η MCDRAM είναι μία 3D-stacked DRAM μνήμη [25] η οποία μέσω των πολλαπλών καναλιών της προσφέρει στους πυρήνες του Xeon Phi και τις αντίστοιχες VPUs τους πάνω από 4 φορές μεγαλύτερο εύρος ζώνης (bandwidth) σε σύγκριση με τη DDR4 και η χωρητικότητα της στο συγκεκριμένο μοντέλο είναι 16 GB. Επιπλέον, παρέχεται DDR4 μνήμη μεγέθους 96 GB.



Σχήμα 2.6: Tile του Xeon Phi 7250.

Ο Xeon Phi 3120P [24] ανήκει στην πρώτη γενιά της προαναφερθείσας σειράς και βασίζεται στην αρχιτεκτονική MIC (Many Integrated Core) τεχνολογίας 22nm. Έχει 57 πυρήνες με συχνότητα επεξεργασίας 1.10 GHz οι οποίοι υποστηρίζουν 4 νήματα έκαστος προσφέροντας συνολικά 228 νήματα. Σε κάθε πυρήνα αντιστοιχεί 1 VPU μεγέθους 512 bits. Συνολικά για όλους τους πυρήνες η L2 κρυφή μνήμη του είναι 28.5 MB και η κύρια μνήμη του, η οποία είναι τύπου DDR5, έχει μέγεθος 6 GB.

Σε αντίθεση με τους συνεπεξεργαστές της πρώτης γενιάς, οι οποίοι λειτουργούν με τη χρήση ενός κύριου επεξεργαστή (host), οι επεξεργαστές της δεύτερης δεν υπόκεινται στον αυστηρό ορισμό των συνεπεξεργαστών καθώς έχουν τη δυνατότητα να λειτουργούν και ως αυτόνομοι επεξεργαστές. Γενικότερα, οι Xeon Phi συνεπεξεργαστές προορίζονται για χρήση σε υπερυπολογιστές, servers και σε σταθμούς εργασίας με υψηλές απαιτήσεις επεξεργαστικής ισχύος και μεγάλα περιθώρια παραλληλοποίησης των εφαρμογών που εκτελούν. Η αρχιτεκτονική τους επιτρέπει την εκτέλεση προγραμμάτων στις κυριότερες γλώσσες προγραμματισμού ενώ η Intel προσφέρει μεταγλωττιστές για προγράμματα σε C ή C++ [27] και Fortran [26] που εξάγουν εκτελέσιμα σε συμβατή μορφή.

Η κοινή μνήμη των πυρήνων ευνοεί τον προγραμματισμό παράλληλων εφαρμογών με τη χρήση του μοντέλου κοινής μνήμης χωρίς αυτό να είναι περιοριστικό για τη χρήση του μοντέλου ανταλλαγής μηνυμάτων ή του υβριδικού. Ακόμα, οι συγκεκριμένοι συνεπεξεργαστές υποστηρίζουν δύο μοντέλα εκτέλεσης το offload και το native [28]. Κατά την offload λειτουργία εκτέλεσης ο κύριος επεξεργαστής (host) εκτελεί το κυρίως μέρος του κώδικα και εναποθέτει στο συνεπεξεργαστή τις επεξεργαστικά απαιτητικές διαδικασίες. Στη native λειτουργία όλο το πρόγραμμα εκτελείται στους πυρήνες του συνεπεξεργαστή. Η native λειτουργία προτιμάται όταν στόχος είναι η όσο το δυνατό μεγαλύτερη παραλληλοποίηση του προγράμματος ενώ η offload όταν το μεγαλύτερο μέρος της εκάστοτε εφαρμογής εκτελείται σειριακά και εξαρτάται από κλήσεις εισόδου-εξόδου δεδομένων από περιφερειακές συσκευές.

Κεφάλαιο 3

Νευρωνικά δίκτυα

Στο κεφάλαιο αυτό γίνεται μία εκτενής παρουσίαση του θεωρητικού υποβάθρου που κρίνεται χρήσιμο για την κατανόηση της εργασίας. Αρχικά παρουσιάζονται τα τεχνητά νευρωνικά δίκτυα, οι σημαντικότερες κατηγορίες στις οποίες διαχωρίζονται και οι κυριότερες εφαρμογές τους. Τέλος, αναλύονται τα LSTM νευρωνικά δίκτυα καθώς χρησιμοποιούνται από την εφαρμογή η οποία επιχειρείται να παραλληλοποιηθεί.

3.1 Τεχνητή νοημοσύνη και νευρωνικά δίκτυα

Με τον όρο **τεχνητή νοημοσύνη** εννοείται η νοημοσύνη που επιδεικνύεται από τους υπολογιστές και γενικότερα τις μηχανές επεξεργασίας, σε αναλογία με τη φυσική νοημοσύνη, η οποία αφορά τη νοημοσύνη των ανθρώπων και των ζώων. Στην επιστήμη των υπολογιστών η έρευνα στην τεχνητή νοημοσύνη ορίζεται από τη μελέτη, τη σχεδίαση και την υλοποίηση υπολογιστικών συστημάτων που μιμούνται στοιχεία της ανθρώπινης συμπεριφοράς, τα οποία υπονοούν έστω και στοιχειώδη ευφυΐα όπως η μάθηση, η προσαρμοστικότητα και η επίλυση προβλημάτων, αντιλαμβάνονται το περιβάλλον τους και ενεργούν με σκοπό την επίτευξη του εκάστοτε στόχου τους.

Συγκριτικά με τον άνθρωπο η εφαρμογή της τεχνητής νοημοσύνης μπορεί, σε ορισμένες περιπτώσεις, να φέρει σε πέρας πιο δύσκολες και πολύπλοκες εργασίες σε πολύ λιγότερο χρόνο, με λιγότερα σφάλματα και μεγαλύτερη ανοχή σε ανωμαλίες [29]. Μερικοί από τους κύριους κλάδους έρευνας της τεχνητής νοημοσύνης είναι η θεωρία παιγνίων, οι γλώσσες προγραμματισμού, η όραση υπολογιστών, η ρομποτική και τα νευρωνικά δίκτυα.

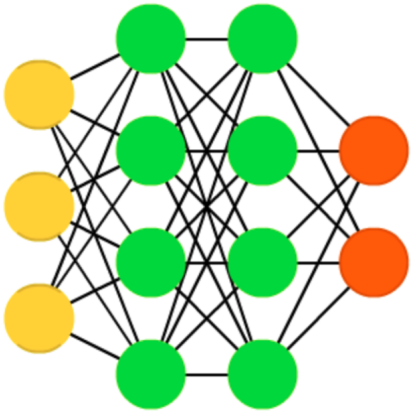
Τα **νευρωνικά δίκτυα** είναι υπολογιστικά συστήματα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα [30], τα οποία συστήνουν τους ανθρώπινους εγκεφάλους, και σταδιακά βελτιώνουν την επίδοσή τους στην εκτέλεση συγκεκριμένων εργασιών μέσω της παρατήρησης παραδειγμάτων. Ένα νευρωνικό δίκτυο είναι ένας τεράστιος επεξεργαστής με κατανεμημένη αρχιτεκτονική αποτελούμενος από απλές μονάδες επεξεργασίας (νευρώνες) συνδεδεμένες μεταξύ τους, έχοντας την ικανότητα να αποθηκεύει εμπειρική γνώση και να την καθιστά διαθέσιμη για χρήση. Τα δίκτυα αυτά έχουν την εγγενή δυνατότητα να προσαρμόζονται ανάλογα με τις μεταβολές που επιδέχεται το περιβάλλον τους. Ακόμα, λόγω της κατανεμημένης φύσης της

πληροφορίας που αποθηκεύεται στο δίκτυο ένα νευρωνικό δίκτυο είναι ανεκτικό σε βλάβες, με την έννοια ότι η απόδοση του μειώνεται βαθμιαία και ομαλά υπό αντίξοες συνθήκες λειτουργίας. Επιπροσθέτως, τα δίκτυα αυτά μπορούν να διαχειρίζονται δεδομένα μεγάλου όγκου και με θόρυβο. Ωστόσο, ένα νευρωνικό δίκτυο είναι σχεδιασμένο μόνο για μία συγκεκριμένη λειτουργία, όπως η αναγνώριση φωνής και άλλων προτύπων, η όραση υπολογιστών, η ιατρική διάγνωση και η ανάπτυξη νικητήριων στρατηγικών σε επιτραπέζια και περίπλοκα παιχνίδια.

3.2 Είδη νευρωνικών δικτύων

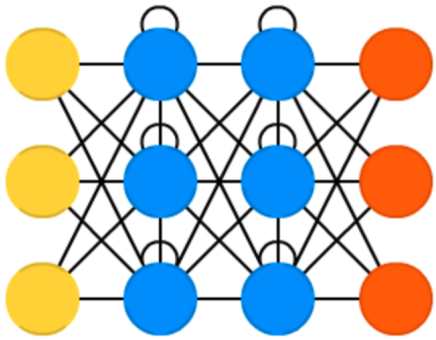
Υπάρχουν πολλά είδη νευρωνικών δικτύων και διαχωρίζονται κυρίως ανάλογα με τον τρόπο που συνδέονται μεταξύ τους οι νευρώνες, την τοπολογία της εσωτερικής τους οργάνωσης, την ύπαρξη μνήμης και τον τρόπο εκπαίδευσής τους. Οι βασικές κατηγορίες είναι οι εξής:

- **Feedforward (Πρόσθιας τροφοδότησης)**



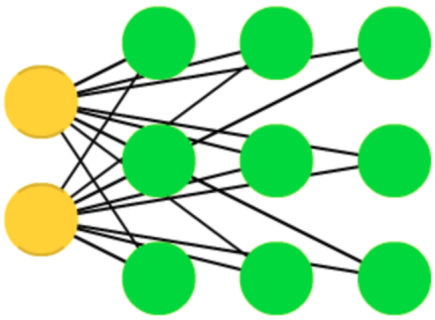
Σε ένα νευρωνικό δίκτυο οι νευρώνες οργανώνονται σε μορφή επιπέδων. Στην απλούστερη δυνατή μορφή ενός δικτύου έχουμε ένα επίπεδο εισόδου το οποίο συνδέεται απευθείας με ένα επίπεδο νευρώνων εξόδου αλλά όχι αντίστροφα. Ακόμα, στα παραπάνω δίκτυα μπορούν να προστεθούν ένα ή περισσότερα κρυφά επίπεδα ανάμεσα στα δύο προαναφερθέντα επίπεδα εισόδου και εξόδου, δίνοντας τη δυνατότητα στο νευρωνικό να εξαγάγει στατιστικά υψηλότερης τάξης από την είσοδο του. Στα δίκτυα αυτά ως είσοδοι για τα κρυφά επίπεδα χρησιμοποιούνται τα σήματα εξόδου του προηγούμενου επιπέδου και μόνο, δηλαδή η ροή της πληροφορίας δεν περιέχει βρόχους ανάδρασης και πηγαίνει μόνο προς τα επόμενα επίπεδα, εμπρός, και προφανώς εκεί οφείλεται και η ονομασία τους [32]. Ίσως το πιο γνωστό δίκτυο της κατηγορίας αυτής είναι το convolutional neural network (CNN ή ConvNet) [31], το οποίο χρησιμοποιείται ευρέως σε εφαρμογές αναγνώρισης εικόνας και βίντεο, ανάπτυξης συστημάτων προτάσεων και επεξεργασίας φυσικής γλώσσας.

- **Recurrent Neural Networks (RNNs)**



Ένα αναδρομικό νευρωνικό δίκτυο (recurrent neural network) [33] διαφέρει από ένα δίκτυο πρόσθιας τροφοδότησης στο ότι έχει έναν τουλάχιστον βρόχο ανάδρασης, όπου δηλαδή ένας νευρώνας τροφοδοτεί με το σήμα εξόδου του κάποιους νευρώνες από προηγούμενα ή το ίδιο επίπεδο. Είναι κατάλληλα για εφαρμογές όπου τα δεδομένα εισόδου και εξόδου είναι ακολουθίες, καθώς μπορούν να ενσωματώνουν πληροφορίες από τα συμφραζόμενα με έναν ευέλικτο τρόπο και ανεκτικό σε τοπικές διαστρεβλώσεις των δεδομένων εισόδου. Η σημαντικότητα ανάπτυξης μεθόδων αντιστοίχισης ακολουθιών εισόδου σε ακολουθίες εξόδου αντικατοπτρίζεται από τις εφαρμογές που επιχειρούν να φέρουν εις πέρας όπως είναι η αναγνώριση και η σύνθεση φωνής, η συντακτική ανάλυση προτάσεων και η αυτοματοποιημένη μετάφραση. Από τα πιο υποσχόμενα RNNs είναι τα δίκτυα long short-term memory (LSTMs), καθώς έχουν τη δυνατότητα να διαχειρίζονται μεγάλα διαστήματα καθυστέρησης μεταξύ των δεδομένων εισόδου που σχετίζονται με την έξοδο και την επηρεάζουν.

- **Self-organizing maps (SOMs)**



Οι αυτο-οργανούμενοι χάρτες (self-organizing maps) [34] είναι ένας τύπος νευρωνικού δικτύου που εκπαιδεύεται με μη επιβλεπόμενη μάθηση για την παραγωγή κατηγοριοποιήσιμης αναπαράστασης των δεδομένων εισόδου. Με την ιδιότητα αυτή τα SOMs είναι χρήσιμα για την απεικόνιση πολυδιάστατων δεδομένων εισόδου σε λίγες διαστάσεις, επιτυγχάνοντας, δηλαδή, τη μείωση της διαστατικότητας ενός φαινομένου. Επιπλέον, τα

self-organizing maps διαφέρουν από τα άλλα νευρωνικά δίκτυα αφού δε χρησιμοποιούν τεχνικές μάθησης διόρθωσης σφάλματος, αλλά εφαρμόζουν ανταγωνιστική μάθηση, όπου οι νευρώνες εξόδου του δικτύου ανταγωνίζονται μεταξύ τους για το δικαίωμα ενεργοποίησης. Σε έναν αυτο-οργανούμενο χάρτη οι νευρώνες τοποθετούνται στους κόμβους ενός πλέγματος το οποίο είναι συνήθως μονοδιάστατο ή διδιάστατο.

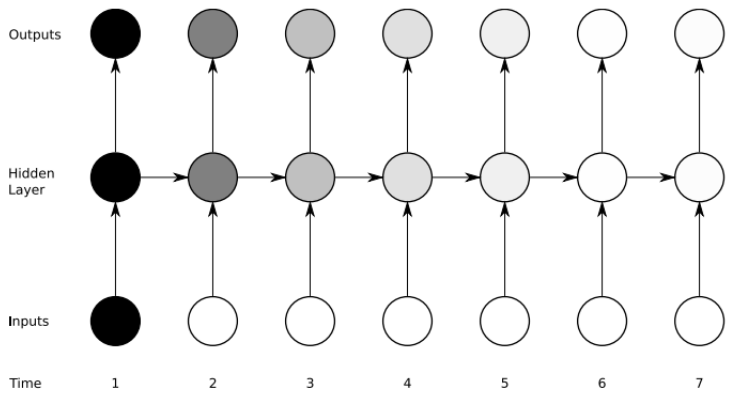
Κάποιες κατηγορίες νευρωνικών δικτύων ακόμα είναι τα **Radial Basis Function (RBF)** [35], τα **Auto-Associative** [36] και τα δίκτυα **Hopfield** [37]. Ακόμα, διαφορετικού τύπου νευρωνικά δίκτυα συνδυάζονται με σκοπό την καλύτερη αποτελεσματικότητα, όπως τα Convolutional LSTM (ConvLSTM) που αποτελούν μία μίξη Convolutional και LSTM νευρωνικών δικτύων.

3.3 LSTMs, πλεονεκτήματα και χρήσεις

Τα δίκτυα long short-term memory είναι απλά αναδρομικά νευρωνικά δίκτυα τα οποία είναι ικανά να μαθαίνουν μακρινές σε χρονική απόσταση εξαρτήσεις. Τα LSTMs εφευρέθηκαν το 1997 από τους Sepp Hochreiter και Jürgen Schmidhuber [38] και εξελίχθηκαν το 2000 από τους Felix Gers, Jürgen Schmidhuber και Fred Cummins [39]. Ανάμεσα σε άλλες διακρίσεις, εφαρμογές που χρησιμοποιούν το συγκεκριμένο είδος νευρωνικού δικτύου έχουν πετύχει κορυφαία αποτελέσματα σε διαγωνισμούς συμπίεσης κειμένου φυσικής γλώσσας [40] και αναγνώρισης φωνής και χειρόγραφων κειμένων [41], με συνέπεια από το 2016 τεχνολογικοί κολοσσοί, όπως οι Google, Apple και Microsoft να χρησιμοποιούν LSTM νευρωνικά ως βασικά συστατικά των καινοτομιών τους [42][43][44].

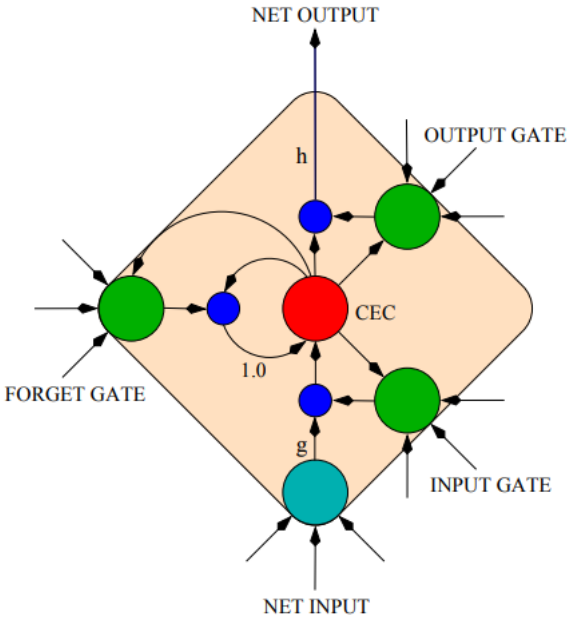
Το όνομα του συγκεκριμένου τύπου νευρωνικών δικτύων δηλώνει την ικανότητα τους να έχουν μικρή μνήμη η οποία μπορεί να διαρκέσει μεγάλες χρονικές περιόδους. Για το λόγο αυτό είναι τα κατάλληλα νευρωνικά για να κατηγοριοποιούν, να διαχειρίζονται και να προβλέπουν ακολουθίες που περιέχουν εξαρτημένα γεγονότα στα οποία παρεμβάλλονται αγνώστου μεγέθους και διάρκειας δεδομένα. Η ανακάλυψη τους οφείλεται στην αδυναμία των RNNs να διαχειρίζονται εξαρτήσεις μεταξύ των δεδομένων εισόδου σε μεγάλο εύρος.

Μία σημαντική ιδιότητα των αναδρομικών δικτύων είναι η δυνατότητά τους να χρησιμοποιούν πληροφορίες από τα συμφραζόμενα όταν καλούνται να επεξεργαστούν ακολουθίες δεδομένων. Ωστόσο, το εύρος των συμφραζομένων για τις κλασικές αρχιτεκτονικές RNN δικτύων είναι περιορισμένο. Το πρόβλημα εντοπίζεται στο γεγονός ότι η επίδραση μιας εισόδου στα κρυφά επίπεδα και συνεπώς στο επίπεδο εξόδου του δικτύου είτε φθίνει είτε αυξάνεται εκθετικά καθώς διέρχεται από τους βρόχους του δικτύου. Η ατέλεια αυτή, η οποία αναφέρεται στη βιβλιογραφία ως το πρόβλημα των μηδενιζόμενων κλίσεων (vanishing gradient problem), κάνει δύσκολη τη διαδικασία μάθησης διαδικασιών που περιέχουν καθυστερήσεις μεγαλύτερες των δέκα χρονικών στιγμών μεταξύ συσχετιζόμενων εισόδων και εξόδων. Το πρόβλημα των μηδενιζόμενων κλίσεων, που αναπαρίσταται στο σχήμα 3.1, λύνει πιο αποτελεσματικά η χρήση long short-term memory νευρωνικών δικτύων.



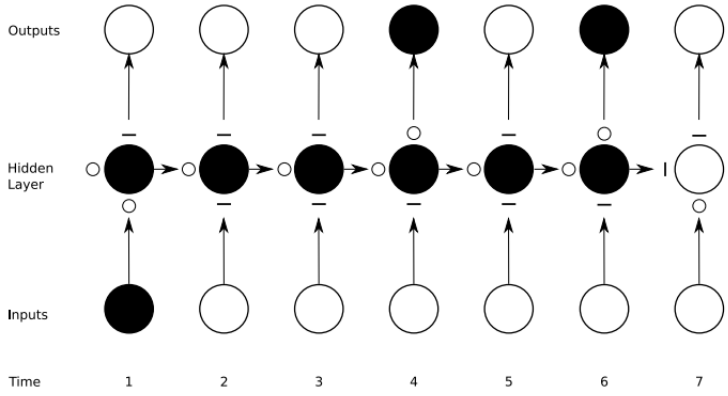
Σχήμα 3.1: **Το πρόβλημα μηδενιζόμενων κλίσεων των RNNs.** Η σκίαση των κόμβων αναπαριστά την ευαισθησία των κόμβων του δικτύου συναρτήσει του χρόνου, όσο πιο έντονη τόσο μεγαλύτερη είναι η ευαισθησία. Με την πάροδο του χρόνου η ευαισθησία φθίνει εκθετικά καθώς νέες είσοδοι ανανεώνουν την κατάσταση του κρυφού επιπέδου και το δίκτυο ξεχνά την αρχική είσοδο.

Ένα block του LSTM συντίθεται από τέσσερα βασικά στοιχεία: το κύτταρο μνήμης (memory cell), τη θύρα εισόδου (input gate), τη θύρα εξόδου (output gate) και τη θύρα λήθης (forget gate). Το κύτταρο είναι υπεύθυνο να θυμάται την κατάσταση του για απροσδιόριστα χρονικά διαστήματα με τη χρήση μίας αναδρομικής σύνδεσης με τον εαυτό του. Επιπλέον, η θύρα εισόδου είναι υπεύθυνη για τις τιμές της εισόδου και κατά πόσο αυτές θα επηρεάσουν την ενημέρωση της κατάστασης του κυττάρου. Η θύρα λήθης αποφασίζει ποιες τιμές θα σταματήσουν να κρατούνται στην μνήμη του κυττάρου. Τέλος, η θύρα εξόδου αποφασίζει για το αν και σε τι ποσοστό θα επηρεαστεί η έξοδος από την εσωτερική κατάσταση του κυττάρου και την είσοδο. Οι μεταξύ τους συνδέσεις και γενικότερα η αρχιτεκτονική ενός LSTM block φαίνεται στο σχήμα 3.2. Για περισσότερες λεπτομέρειες για τον ακριβή τρόπο λειτουργίας του LSTM block, ο αναγνώστης μπορεί να μελετήσει το [39].



Σχήμα 3.2: Αρχιτεκτονική ενός LSTM block.

Οι τρεις θύρες που προαναφέρθηκαν επιτρέπουν στο κύτταρο του δικτύου να αποθηκεύει και να έχει πρόσβαση σε πληροφορίες για μεγάλα χρονικά διαστήματα, λύνοντας με τον τρόπο αυτό το πρόβλημα των μηδενιζόμενων κλίσεων. Στο σχήμα 3.3 που ακολουθεί αναπαρίσταται η διατήρηση της πληροφορίας σε ένα LSTM δίκτυο.



Σχήμα 3.3: **Διατήρηση της πληροφορίας από LSTM δίκτυο.** Οι σκιασμένοι κόμβοι δείχνουν την ευαισθησία του block στην αρχική είσοδο, της χρονικής στιγμής ένα. Οι καταστάσεις των θυρών εισόδου, λήθης και εξόδου απεικονίζονται κάτω, αριστερά και πάνω αντίστοιχα του κόμβου του κρυφού επιπέδου (κύτταρο). Για απλότητα είναι είτε πλήρως ανοιχτές ('O') είτε πλήρως κλειστές ('—'). Το κύτταρο μνήμης απομνημονεύει την αρχική είσοδο όσο η θύρα λήθης είναι ανοιχτή και η θύρα εισόδου κλειστή. Η ευαισθησία του επιπέδου εξόδου εξαρτάται από την πόρτα εξόδου αφήνοντας ανεπηρέαστη την κατάσταση του κυττάρου.

Όπως σε όλα τα νευρωνικά δίκτυα έτσι και στα LSTMs κάθε σύνδεση μεταξύ των συστατικών στοιχείων του δικτύου έχει ένα βάρος (weight) το οποίο καθορίζει την ένταση της επιρροής του δέκτη από τον παραλήπτη, αφού η έξοδος εξαρτάται από το γινόμενο του βάρους με την τιμή της εισόδου. Όταν η είσοδος είναι διανυσματική τότε και το βάρος είναι διανυσματικό (πίνακας βαρών) το γινόμενο με την είσοδο προκύπτει με τον πολλαπλασιασμό στοιχείο προς στοιχείο (pointwise). Επίσης, κάθε δομικό στοιχείο, είτε είναι θύρα είτε είναι κύτταρο μνήμης, έχει μία πόλωση (bias) η οποία καθορίζει την τιμή της εξόδου του ανεξάρτητα από την είσοδο. Συνολικά, λοιπόν, η έξοδος ισούται με το άθροισμα της πόλωσης και του γινομένου του βάρους με την είσοδο ($\text{out} = \text{w} \cdot \text{in} + \text{b}$). Αφού υπολογιστεί η έξοδος τότε υπολογίζεται η απόκλισή της από την επιθυμητή με τη χρήση μίας συνάρτησης, η οποία καλείται συνάρτηση απωλειών (loss function).

Η σχέση που περιγράφει την ενημέρωση των βαρών των LSTMs δικτύων είναι η ακόλουθη:

$$w_t \leftarrow w_{t-1} - \eta \partial_w c(w_{t-1}),$$

όπου η ο εκάστοτε ρυθμός μάθησης (learning rate) και c η συνάρτηση απωλειών του δικτύου.

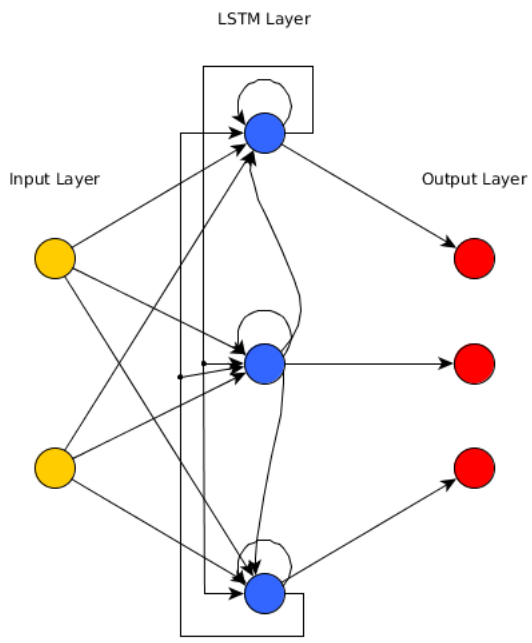
Όπως μαρτυρά και η παραπάνω σχέση ο ρυθμός μάθησης είναι μία σταθερά του δικτύου που καθορίζει την ένταση με την οποία το βάρος ανανεώνεται. Όσο πιο μεγάλος ο ρυθμός μάθησης τόσο πιο "νευρική" είναι η ανανέωση των βαρών.

Η διαδικασία ανανέωσης των παραμέτρων ενός δικτύου η οποία περιγράφεται από την παραπάνω σχέση ονομάζεται Stochastic Gradient Descent (SGD).

Παρακάτω παρουσιάζονται αναλυτικά οι σχέσεις των συστατικών στοιχείων ενός LSTM δικτύου για το προς τα εμπρός πέρασμα της εκπαίδευσης:

- **Θύρα λήθης (forget gate):** $\boldsymbol{f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)}$, όπου
  σ: σιγμοειδής συνάρτηση, W_f : πίνακας βαρών της θύρας, h_{t-1} : η αμέσως προηγούμενη έξοδος, x_t: η είσοδος και b_f: η πόλωση της θύρας.
- **Θύρα εισόδου (input gate):** $\boldsymbol{i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)}$.
- **Κύτταρο μνήμης (memory cell):** $\boldsymbol{C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t}$, όπου
  $\boldsymbol{\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)}$.
- **Θύρα εξόδου (output gate):** $\boldsymbol{o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)}$.
- **Έξοδος (output):** $\boldsymbol{h_t = o_t \cdot tanh(C_t)}$.

Στο σχήμα που ακολουθεί παρουσιάζεται αφαιρετικά η μορφή ενός LSTM δικτύου με δύο εισόδους και τρία LSTM κύτταρα.



Σχήμα 3.4: Νευρωνικό δίκτυο LSTM δύο εισόδων και τριών εξόδων.

Με τη βοήθεια του παραπάνω σχήματος επισημαίνεται ότι ο αριθμός των εξόδων του δικτύου είναι όσο και το πλήθος των LSTM blocks. Στο παραπάνω σχήμα το δίκτυο δέχεται ως είσοδο ένα διάνυσμα 2 x 1. Ωστόσο, η είσοδος των LSTM κυττάρων είναι 5 x 1 καθώς εκτός από τις δύο τιμές της πραγματικής εισόδου περιλαμβάνονται και οι τρεις έξοδοι των κυττάρων που προέκυψαν από την προηγούμενη επανάληψη της εκπαίδευσης, όπως άλλωστε αναμενόταν από τους παραπάνω τύπους ανανέωσης των βαρών των θυρών και του κυττάρου μνήμης.

Κεφάλαιο 4

Οπτική αναγνώριση χαρακτήρων με LSTM νευρωνικά δίκτυα

Στο κεφάλαιο αυτό παρουσιάζεται συνοπτικά η έννοια της οπτικής αναγνώρισης χαρακτήρων και αναλύεται ο τρόπος λειτουργίας της εφαρμογής που επιχειρείται να παραλληλοποιηθεί.

4.1 Οπτική αναγνώριση χαρακτήρων

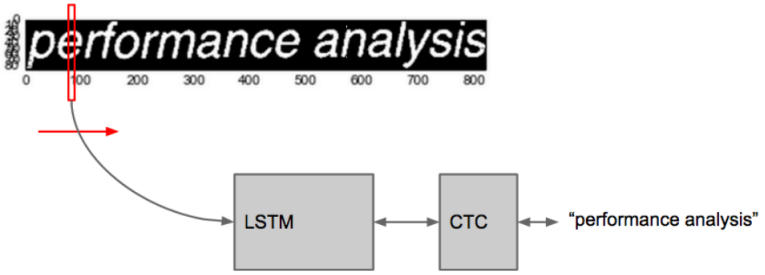
Η οπτική αναγνώριση χαρακτήρων (Optical Character Recognition ή OCR) είναι η διαδικασία μετατροπής εικόνων χειρογράφων ή έντυπων κειμένων σε κωδικοποιημένο κείμενο αναγνωρίσιμο από ηλεκτρονικό υπολογιστή [45]. Η οπτική αναγνώριση χαρακτήρων αποτελεί μια κοινή πρακτική ψηφιοποίησης εκτυπωμένου ή χειρόγραφου κειμένου ώστε να είναι δυνατή ηλεκτρονικά η επεξεργασία, η αναζήτηση, η αποθήκευση και η χρήση του κειμένου αυτού από οποιαδήποτε εφαρμογή όπως η μετατροπή κειμένου σε φωνή ή η αυτόματη μετάφραση του. Η οπτική αναγνώριση χαρακτήρων έχει αποτελέσει πεδίο έρευνας αρκετών κλάδων της επιστήμης των υπολογιστών όπως η τεχνητή νοημοσύνη, η αναγνώριση προτύπων και η όραση υπολογιστών [46] [47]. Αρχικά, οι εφαρμογές OCR εκπαιδεύονταν με διαφορετικές εικόνες του ίδιου χαρακτήρα για κάθε χαρακτήρα του εκάστοτε αλφαβήτου και λειτουργούσαν αποκλειστικά για μία γραμματοσειρά. Τα σύγχρονα συστήματα είναι ικανά να παρέχουν οπτική αναγνώριση με μεγάλη ακρίβεια για τις περισσότερες γραμματοσειρές και τους περισσότερους τύπους αρχείων ψηφιακών εικόνων. Στην πρόοδο αυτή έχει συμβάλει σημαντικά η χρήση LSTM νευρωνικών δικτύων.

4.2 Ανάλυση της λειτουργίας και του αλγόριθμου εκπαίδευσης της εφαρμογής

Η εφαρμογή η οποία επιχειρείται να παραλληλοποιηθεί αφορά την οπτική αναγνώριση χαρακτήρων από εικόνες κειμένου. Ειδικότερα, γίνεται προσπάθεια για την παραλληλοποίηση της εκπαίδευσης του νευρωνικού δικτύου στο οποίο βασίζεται η οπτική αναγνώριση και για το λόγο αυτό κρίνεται αναγκαία η παρουσίαση του τρόπου λειτουργίας της εφαρμογής και η ανάλυση της εκπαίδευσης του δικτύου.

Ο κώδικας που χρησιμοποιείται στο πλαίσιο της διπλωματικής αυτής είχε δημιουργηθεί για τους σκοπούς της δημοσίευσης [48], στην οποία διερευνώνται η καλύτερη αρχιτεκτονική και οι παράμετροι της εκπαίδευσης LSTM δικτύων για OCR εφαρμογές[1].

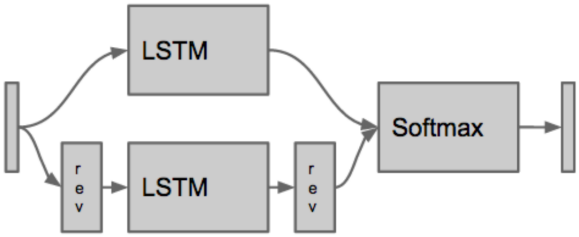
Τα δεδομένα που χρησιμοποιούνται ως είσοδοι για την εκπαίδευση, την επαλήθευση της ορθής λειτουργίας αλλά και την κανονική λειτουργία του νευρωνικού δικτύου είναι εικόνες κειμένου σε συνδυασμό με το κείμενο που τους αντιστοιχεί σε μορφή αναγνώσιμη από τον υπολογιστή και προέρχονται από τη βάση δεδομένων UW-III [49], η οποία χρησιμοποιείται για την αξιολόγηση OCR εφαρμογών. Το πρόβλημα της οπτικής αναγνώρισης χαρακτήρων μετατρέπεται σε ένα πρόβλημα κατηγοριοποίησης ακολουθιών σαρώνοντας την ψηφιακή εικόνα από αριστερά προς τα δεξιά σε κάθετες λωρίδες, όπως φαίνεται στο σχήμα 4.1. Οι εικόνες UW-III είναι μεταβλητού μεγέθους ωστόσο κανονικοποιούνται ως προς το ύψος ούτως ώστε αυτό να ισούται με 48 pixels.



Σχήμα 4.1: Εφαρμογή LSTM σε εικόνα για οπτική αναγνώριση χαρακτήρων.

Το συνολικό νευρωνικό δίκτυο αποτελείται από ένα LSTM διπλής κατεύθυνσης (bidirectional) το οποίο ακολουθεί ένα επίπεδο εξόδου Softmax. Η αρχιτεκτονική του δικτύου απεικονίζεται στο σχήμα 4.2.

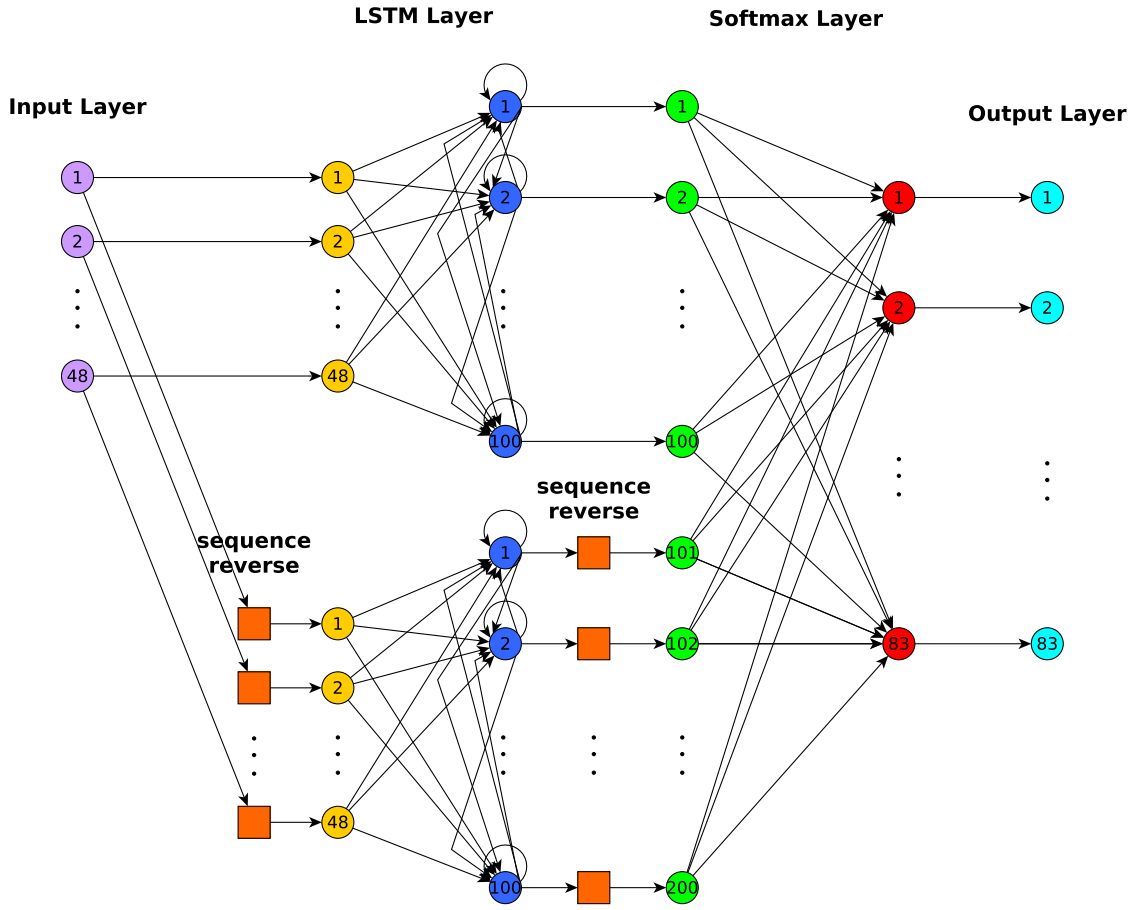
[1] Η εφαρμογή είναι διαθέσιμη στο `https://github.com/tmbdev/clstm`.



Σχήμα 4.2: Αρχιτεκτονική του δικτύου.

Όπως φαίνεται και στο παραπάνω σχήμα ένα bidirectional LSTM συντίθεται από δύο LSTMs. Το πρώτο δέχεται την ακολουθία εισόδου που προκύπτει από την αριστερά προς τα δεξιά σάρωση της εικόνας και τροφοδοτεί το επόμενο επίπεδο με τις εξόδους που προκύπτουν ενώ το δεύτερο αντιστρέφει την ακολουθία εισόδου, σαν να είχε σαρωθεί η εκάστοτε εικόνα από τα δεξιά προς τα αριστερά, την επεξεργάζεται και η ακολουθία εξόδου αντιστρέφεται εκ νέου πριν προωθηθεί στο επόμενο επίπεδο.

Στο σχήμα που ακολουθεί εμφανίζεται το συνολικό νευρωνικό δίκτυο που χρησιμοποιείται από την εφαρμογή με περισσότερες λεπτομέρειες.



Σχήμα 4.3: Λεπτομερής απεικόνιση της αρχιτεκτονικής του δικτύου.

Το επίπεδο εισόδου του δικτύου έχει 48 εισόδους, όσο δηλαδή είναι και το ύψος της κάθε εικόνας, οι οποίες προωθούνται στα δύο LSTM δίκτυα τα οποία έχουν 48 εισόδους και 100 εξόδους έκαστο. Ο αριθμός των εξόδων των LSTM δικτύων, δηλαδή ο αριθμός των LSTM κυττάρων του κάθε δικτύου, επιλέχθηκε να είναι 100 καθώς με αυτή τη ρύθμιση το συνολικό δίκτυο δίνει τα καλύτερα αποτελέσματα.

Το επίπεδο εξόδου τύπου Softmax λαμβάνει ως εισόδους τις εξόδους και των δύο επιμέρους LSTM δικτύων και συνεπώς έχει 200 εισόδους. Οι έξοδοι του επιπέδου αυτού είναι όσες και οι διαφορετικές κλάσεις στις οποίες μπορεί να ανήκει ένας χαρακτήρας των κειμένων που εξετάζονται και στην προκειμένη περίπτωση είναι 83. Δηλαδή, αφού οι εικόνες περιέχουν γράμματα της αγγλικής γλώσσας οι 52 κλάσεις είναι τα πεζά και τα κεφαλαία γράμματα των αγγλικών και σε αυτές προστίθεται το κενό (space), τα σημεία στίξης και η κενή κλάση, η οποία έχει ιδιαίτερη σημασία. Η κενή κλάση χρησιμοποιείται για τις περιπτώσεις όπου η είσοδος δεν αντιστοιχεί σε κανέναν από τους 82 χαρακτήρες και η έξοδος πρέπει να είναι κενή.

Το επίπεδο Softmax [50] είναι υπεύθυνο για την κατηγοριοποίηση της εξόδου. Εφαρμόζοντας τη συνάρτηση softmax στο διάνυσμα εισόδου του το επίπεδο αυτό υπολογίζει την πιθανότητα της εισόδου να ανήκει σε μία από τις 83 κλάσεις. Η συνάρτηση softmax εφαρμόζεται σε διανύσματα και περιγράφεται από την εξής σχέση:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad for\ j = 1,\ ..., K.$$

Κάθε κύτταρο του Softmax επιπέδου, λοιπόν, αντιπροσωπεύει μία κλάση, υπολογίζει την παραπάνω συνάρτηση και το προωθεί το αποτέλεσμα ως έξοδό του. Η έξοδος με τη μεγαλύτερη τιμή θεωρείται η επικρατούσα και συμπεραίνεται ότι η είσοδος ανήκει στην κλάση αυτή.

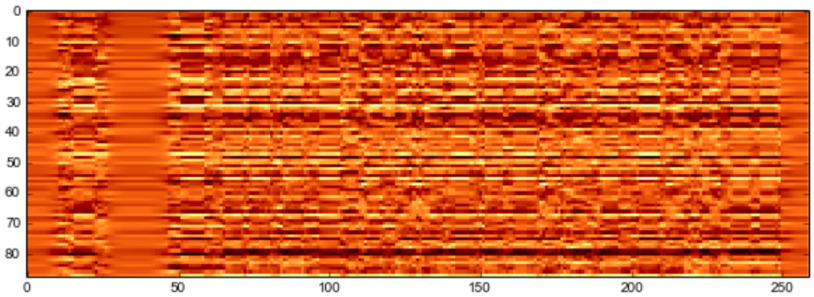
4.3 Ανάλυση του αλγόριθμου εκπαίδευσης της εφαρμογής

Για την εκπαίδευση του νευρωνικού δικτύου χρησιμοποιούνται 450 εικόνες μαζί με το κείμενο που απεικονίζουν ενώ για την επαλήθευση και την εύρεση του σφάλματος της λειτουργίας του 50.

Κάθε εικόνα διασπάται στα κάθετα τμήματα που τη συνθέτουν. Συνεπώς, δημιουργείται μία ακολουθία από διανύσματα διαστάσεων 48x1. Σε ότι αφορά στο δεύτερο LSTM η ακολουθία αυτή αντιστρέφεται και είναι όπως θα ήταν αν εικόνα σαρωνόταν από τα αριστερά προς τα δεξιά. Ύστερα, ακολουθεί το προς τα εμπρός πέρασμα στα δύο LSTM υποδίκτυα και η ακολουθία εξόδου του δεύτερου αντιστρέφεται εκ νέου. Συγχρονισμένα, τότε προωθούν τις εξόδους τους στο επίπεδο Softmax το οποίο παράγει με τη σειρά του την ακολουθία εξόδων.

Έχοντας ως είσοδο μία εικόνα όπως αυτή του σχήματος 4.4.α το αποτέλεσμα που λαμβάνεται από το προς τα εμπρός πέρασμα (forward pass) όταν το δίκτυο είναι στους αρχικούς γύρους της εκπαίδευσης μοιάζει με αυτή που απεικονίζεται στο σχήμα 4.4.β.

(α)

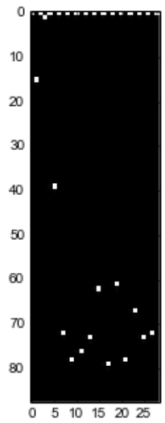


(β)

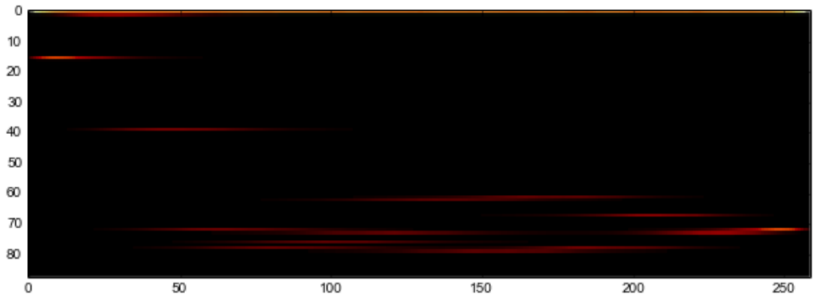
Σχήμα 4.4: (α) Εικόνα εισόδου. (β) Αποτέλεσμα ύστερα από το προς τα εμπρός πέρασμα της παραπάνω εισόδου.

Το πέρασμα προς τα πίσω (backward pass) το οποίο είναι υπεύθυνο για τη διάδοση του σφάλματος και την ανανέωση των χαρακτηριστικών του δικτύου με βάση αυτό προϋποθέτει τη δημιουργία των επιθυμητών στόχων για την εκάστοτε εικόνα και τον υπολογισμό του σφάλματος. Ειδικότερα για κάθε χαρακτήρα του κειμένου-στόχου αποθηκεύεται στη θέση που του αντιστοιχεί ένας άσος και αμέσως μετά άλλος ένας στη θέση του μηδενικού δείκτη για τη διευκόλυνση της μετέπειτα αποκωδικοποίησης. Συνεπώς, οι στόχοι είναι ένας διδιάστατος πίνακας μήκους διπλάσιου του κειμένου, σαν αυτόν που παρουσιάζεται στο σχήμα 4.5. Ωστόσο, όπως είναι προφανές οι διαστάσεις των στόχων και της εξόδου δε συμφωνούν και συνεπώς

για τον υπολογισμό του σφάλματος απαιτείται η ευθυγράμμιση των στόχων με την έξοδο το αποτέλεσμα της οποίας απεικονίζεται για το προηγούμενο παράδειγμα εισόδου στο σχήμα 4.6. Η μεταβολή αυτή των διαστάσεων του στόχου γίνεται με CTC ευθυγράμμιση [51], η οποία μεταβάλλει το στόχο σύμφωνα με την έξοδο που έχει προκύψει από το προς τα εμπρός πέρασμα.



Σχήμα 4.5: Στόχοι για την εικόνα του παραδείγματος.

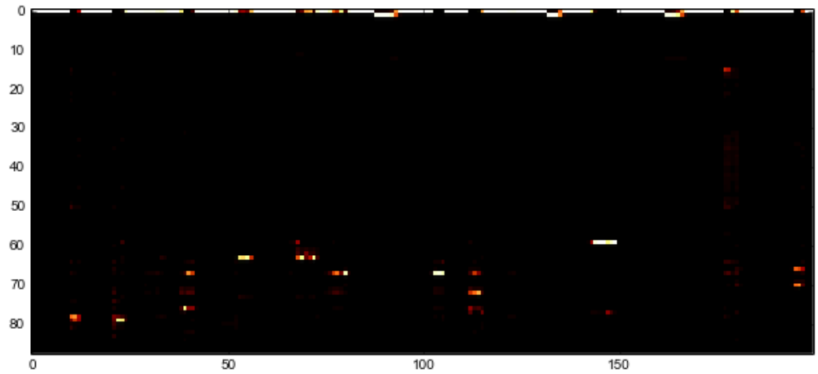


Σχήμα 4.6: CTC ευθυγράμμιση των στόχων.

Έπειτα από τον καθορισμό των στόχων προσδιορίζεται το σφάλμα με τη χρήση της συνάρτησης απωλειών, η οποία στην προκείμενη περίπτωση είναι η διαφορά της εξόδου από το στόχο. Ακολούθως με την προς τα πίσω διάδοση του σφάλματος ανανεώνονται οι πίνακες βαρών των συστατικών στοιχείων των κυττάρων καθώς και οι αντίστοιχες πολώσεις αυτών. Με τον τρόπο αυτό το δίκτυο έχει ανανεωθεί και είναι έτοιμο να δεχθεί την επόμενη ακολουθία (δείγμα).

Ύστερα από αρκετές επαναλήψεις η έξοδος από το πέρασμα προς τα εμπρός, για παράδειγμα της εικόνας του σχήματος 4.7, είναι αρκετά πιο καθαρή από την πρόβλεψη που γινόταν στα αρχικά στάδια της εκπαίδευσης του δικτύου και στο προκείμενο παράδειγμα παρουσιάζεται στο σχήμα 4.8.

Σχήμα 4.7: Εικόνα εισόδου.



Σχήμα 4.8: Αποτέλεσμα ύστερα από το προς τα εμπρός πέρασμα της παραπάνω εισόδου.

Με τη συγκεκριμένη αρχιτεκτονική το νευρωνικό δίκτυο επιτυγχάνει σχεδόν 0.9% ρυθμό σφάλματος. Το διάγραμμα που ακολουθεί απεικονίζει την εξέλιξη του ρυθμού σφάλματος σε συνάρτηση με τον αριθμό των επαναλήψεων της εκπαίδευσης.

Σχήμα 4.9: Μεταβολή του ρυθμού σφάλματος πρόβλεψης συναρτήσει του αριθμού των επαναλήψεων της εκπαίδευσης.

Παρατηρείται ότι σχεδόν μετά τις 140.000 επαναλήψεις ο ρυθμός σφάλματος αυξάνεται. Το γεγονός αυτό αποδίδεται στην υπερβολική προσαρμογή (υπερπροσαρμογή ή overfitting) [52] του νευρωνικού δικτύου στα δείγματα εκπαίδευσης. Όταν το νευρωνικό δίκτυο είναι υπερεκπαιδευμένο έχει εστιάσει αποκλειστικά στα δεδομένα εκπαίδευσης και χάνει τη δυνατότητα να γενικεύει την αποτελεσματικότητά του σε παρόμοια πρότυπα εισόδου-εξόδου, αυξάνοντας συνεπώς το σφάλμα του.

Κεφάλαιο 5

Παραλληλοποίηση εκπαίδευσης LSTM νευρωνικών δικτύων

Σε αυτήν την ενότητα παρουσιάζονται τα εργαλεία παραλληλοποίησης και ανάλυσης προγραμμάτων καθώς και οι μέθοδοι που χρησιμοποιήθηκαν για την παραλληλοποίηση της εκπαίδευσης του LSTM νευρωνικού δικτύου, η οποία περιγράφτηκε στο προηγούμενο κεφάλαιο. Ακόμα, γίνεται αξιολόγηση της επίδοσης του αλγόριθμου εκπαίδευσης στις δύο διαφορετικές εκδόσεις του Xeon Phi.

5.1 Περιγραφή προγραμματιστικού περιβάλλοντος

Για τη μετατροπή της εφαρμογής από σειριακό πρόγραμμα σε παράλληλο χρησιμοποιήθηκε το OpenMP καθώς υποστηρίζει πολλά επίπεδα παραλληλίας με πολύ προσιτό τρόπο για τον προγραμματιστή σε συστήματα κοινής μνήμης όπως ο Xeon Phi. Ο compiler που χρησιμοποιήθηκε είναι ο ICC 17.0.4 της Intel, ο οποίος υποστηρίζει μέχρι και την έκδοση 4.5 του OpenMP. Για την παρατήρηση, την ανάλυση και τη βελτίωση της επίδοσης του παράλληλου κώδικα χρησιμοποιήθηκε το VTune™ Amplifier της Intel. Το VTune είναι μία βοηθητική εφαρμογή ανάλυσης λογισμικού που επιτρέπει την αξιολόγηση και τη μέτρηση της επίδοσης παράλληλων προγραμμάτων που χρησιμοποιούν OpenMP ή TBBs για την παραλληλοποίησή τους.

5.2 Εμπόδια και προϋποθέσεις

Απαραίτητη προϋπόθεση για την εκτέλεση ενός προγράμματος στον Xeon Phi αποτελεί η μεταγλώττιση του ώστε το παραχθέν εκτελέσιμο να είναι συμβατό με το λειτουργικό του σύστημα. Συνεπώς, για τη μετατροπή αυτή του προγράμματος εκπαίδευσης του νευρωνικού δικτύου ήταν αναγκαία η μεταγλώττιση εκτός του πηγαίου κώδικα του προγράμματος και καθεμίας από τις βιβλιοθήκες που χρησιμοποιούνταν. Ακόμα, όπως αναμενόταν αναγκαία ήταν και η μεταγλώττιση των βιβλιοθηκών από τις οποίες εξαρτώνται οι εκάστοτε βιβλιοθήκες. Ειδικότερα, οι βιβλιοθήκες που μεταγλωττίστηκαν ήταν οι protolib, libpng και libz. Η διαδι-

κασία μεταγλώττισης των βιβλιοθηκών αποδείχθηκε ιδιαίτερα χρονοβόρα και επίπονη καθώς παρουσιάστηκαν αρκετά προβλήματα ασυμβατότητας μεταξύ αυτών και του λειτουργικού συστήματος με συνέπεια τον πειραματισμό με διάφορες εκδόσεις των βιβλιοθηκών και κάποιες παρεμβάσεις σε σημεία του κώδικά τους.

Η εκπαίδευση του συγκεκριμένου τύπου νευρωνικού δικτύου είναι σειριακή ως προς τα δείγματα εκπαίδευσης. Αυτό σημαίνει ότι για να λειτουργήσει σωστά χρειάζεται να εκτελείται η κάθε επανάληψη, να ενημερώνονται οι μεταβλητές του δικτύου και ύστερα να ακολουθεί η εκπαίδευση με τη χρήση του επόμενου δείγματος. Για το λόγο αυτό η παραλληλοποίηση ως προς τα δείγματα εκπαίδευση δεν είναι προφανής και εύκολη. Επιπλέον, κάθε δείγμα θεωρείται ως μία ακολουθία δεδομένων και όπως φαίνεται και από τους τύπους του τρίτου κεφαλαίου, οι οποίοι περιγράφουν τις σχέσεις μεταξύ των συστατικών στοιχείων ενός LSTM κυττάρου, το αποτέλεσμα και η ορθή λειτουργία του δικτύου εξαρτώνται άμεσα από αυτήν την ακολουθία μη επιτρέποντας τη κατάτμησή της και, συνεπώς, την παραλληλοποίηση ως προς τα κάθετα τμήματα της εικόνας εισόδου που απαρτίζουν την ακολουθία. Συμπερασματικά, λόγω της φύσης του δικτύου αυτού η παραλληλοποίηση της εκπαίδευσής του δεν είναι προφανής.

5.3 Πρώτο επίπεδο παραλληλοποίησης

Αρχικά, έγινε η προσπάθεια παραλληλοποίησης όλων των βοηθητικών συναρτήσεων που χρησιμοποιούνται από την εφαρμογή για την εκπαίδευση του νευρωνικού δικτύου. Οι περισσότερες από αυτές τις βοηθητικές συναρτήσεις εμπεριέχουν απλές πράξεις πρόσθεσης και πολλαπλασιασμού και σαν άμεση συνέπεια αυτού μπορούν με σχετική ευκολία να μετατραπούν σε συναρτήσεις οι οποίες εκτελούν μεγάλο μέρος των πράξεων που απαιτούνται για τη διεκπεραίωσή τους παράλληλα.

Επιπλέον, στο αρχικό αυτό στάδιο προστέθηκε και η παραλληλοποίηση ανεξάρτητων λειτουργιών, οι οποίες εκτελούνται σε κάθε επανάληψη της εκπαίδευσης του δικτύου. Τόσο το προς τα εμπρός όσο και το προς τα πίσω πέρασμα στα δύο LSTM δίκτυα είναι ανεξάρτητα, αρκεί βέβαια να επιτυγχάνεται ο συγχρονισμός όταν είναι έτοιμα να προωθήσουν τις ακολουθίες εξόδου στο επίπεδο Softmax. Η παραλληλοποίηση των δύο LSTMs έγινε με task-based προσέγγιση καθώς διευκολύνει την εξασφάλιση του συγχρονισμού που είναι απαραίτητος με μία απλή οδηγία του OpenMP (#pragma omp taskwait). Ακόμα, ανεξάρτητοι μεταξύ τους είναι και οι υπολογισμοί για τις θύρες εισόδου (g_i), εξόδου (g_o), λήθης (g_f) και του κυττάρου μνήμης (c_i), όπως άλλωστε μαρτυρούν και οι σχέσεις στο τέλος του τρίτου κεφαλαίου, επιτρέποντας την ταυτόχρονη εκτέλεση πράξεων για τον υπολογισμό τους. Ωστόσο, και σε αυτήν την περίπτωση υπάρχει ανάγκη για συγχρονισμό καθώς η συνέχεια της εκπαίδευσης προϋποθέτει την αποτίμηση τους, η οποία επιτυγχάνεται και πάλι με την ανάθεση των ανεξάρτητων λειτουργιών σε διαφορετικά tasks. Όταν η εκτέλεση του προγράμματος, λοιπόν, φτάνει σε αυτά τα σημεία του κώδικα, τα τμήματα της εκάστοτε λειτουργίας εκτελούνται ταυτόχρονα και συγχρονίζονται, με την έννοια ότι όταν έχει περατωθεί ο υπολογισμός όλων των ξεχωριστών και ανεξάρτητων τμημάτων της λειτουργίας η εκτέλεση της κανονικής ροής του προγράμματος συνεχίζεται κανονικά.

Ακόμα, αλλάχτηκε η δομή των δεδομένων σε κάποια σημεία του κώδικα, όπως οι πίνακες των βαρών, οι πίνακες στους οποίους αποθηκεύονταν οι τιμές των θυρών, οι τιμές των εισόδων και των εξόδων αυτών καθώς και οι πίνακες των στόχων. Στην αρχική έκδοση χρησιμοποιούνταν δομές δεδομένων της βιβλιοθήκης Eigen [54], καθώς η βιβλιοθήκη αυτή παρέχει πληθώρα συναρτήσεων που αφορούν πράξεις με πίνακες. Ειδικότερα, οι δομές Eigen::Tensor αντικαταστάθηκαν από απλούς πίνακες double ή float, ανάλογα με την περίσταση, της C++. Σαν αποτέλεσμα, στο στάδιο αυτό υλοποιήθηκαν και οι αντίστοιχες συναρτήσεις για τις νέες δομές δεδομένων. Οι νέες δομές είναι όσο πιο απλές γίνεται καθώς με τον τρόπο αυτό διευκολύνεται η προσπάθεια τόσο της παραλληλοποίησης όσο και της διανυσματοποίησης του προγράμματος.

Στο διάγραμμα που ακολουθεί απεικονίζεται ο χρόνος εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης του νευρωνικού δικτύου στον Xeon Phi 3120P σε σχέση με τον αριθμό των νημάτων που χρησιμοποιούνται.

Η εκτέλεση 30000 επαναλήψεων της εκπαίδευσης του δικτύου με την πρωτότυπη έκδοση του προγράμματος απαιτεί κατά μέσο όρο 5212 λεπτά στον Xeon Phi 3120P. Επιπλέον, διευκρινίζεται πως το πρόγραμμα τροποποιήθηκε ούτως ώστε η ακολουθία των εικόνων που επιλέγονται από το σύνολο των δειγμάτων εκπαίδευσης να είναι η ίδια σε κάθε εκτέλεση αντίθετα με την πρωτότυπη εκδοχή του, κατά την οποία η εικόνα σε κάθε επανάληψη επιλεγόταν τυχαία. Αυτό έγινε ώστε να είναι άμεσα συγκρίσιμα μεταξύ τους τα αποτελέσματα από τις διαδοχικές βελτιστοποιήσεις που εισήχθησαν στον κώδικα.

Σχήμα 5.1: Μεταβολή του χρόνου εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης σε συνάρτηση με τον αριθμό των OpenMP νημάτων που χρησιμοποιούνται για την έκδοση του πρώτου επιπέδου παραλληλοποίησης και της πρωτότυπης έκδοσης.

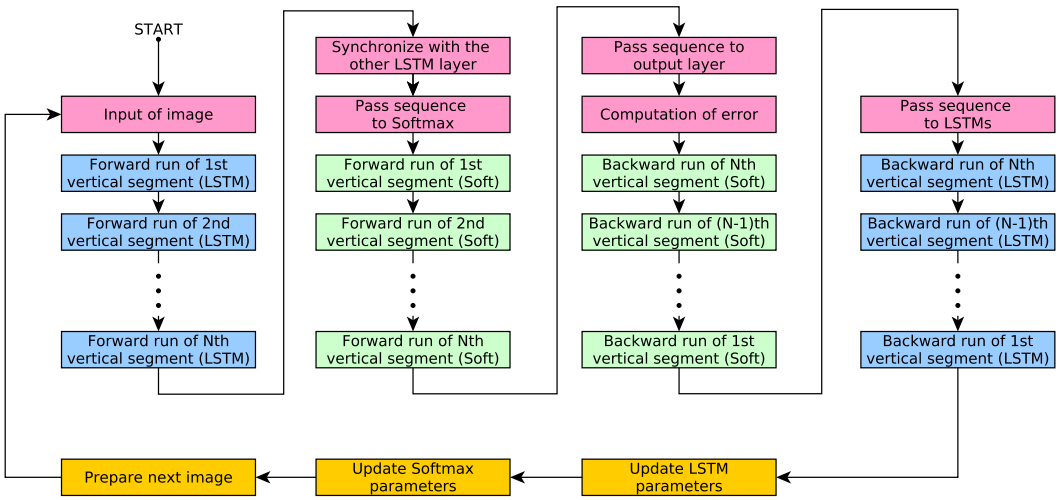
Παρατηρείται με τη βοήθεια του διαγράμματος ότι το πρόγραμμα δεν κλιμακώνει ομαλά και δεν αξιοποιείται ο αριθμός των διαθέσιμων νημάτων του Xeon Phi παρόλο που υπάρχει σημαντική βελτίωση στην ταχύτητα εκτέλεσης του κώδικα και για το λόγο αυτό επιχειρήθηκε η περαιτέρω παραλληλοποίησή του. Η κλιμάκωση δεν είναι η επιθυμητή καθώς έως αυτό

το επίπεδο εκτελούνται παράλληλα οι λειτουργίες των δύο LSTM επιπέδων, μερικοί υπολογισμοί των τιμών των συστατικών στοιχείων καθώς και βοηθητικές συναρτήσεις που όμως διαχειρίζονται δεδομένα μικρών διαστάσεων. Η ταχύτερη εκτέλεση σημειώνεται όταν χρησιμοποιούνται 20 νήματα του Xeon PHI και διαρκεί περίπου 585 λεπτά. Ακόμα, παρατηρείται ότι η βελτίωση από τη σειριακή έκδοση του προγράμματος σε αυτήν του πρώτου σταδίου είναι σημαντική ακόμα και με τη χρήση του ενός νήματος του επεξεργαστή. Αυτό συμβαίνει κυρίως λόγω της αντικατάστασης των δομών δεδομένων της Eigen με τις απλούστερες της C++.

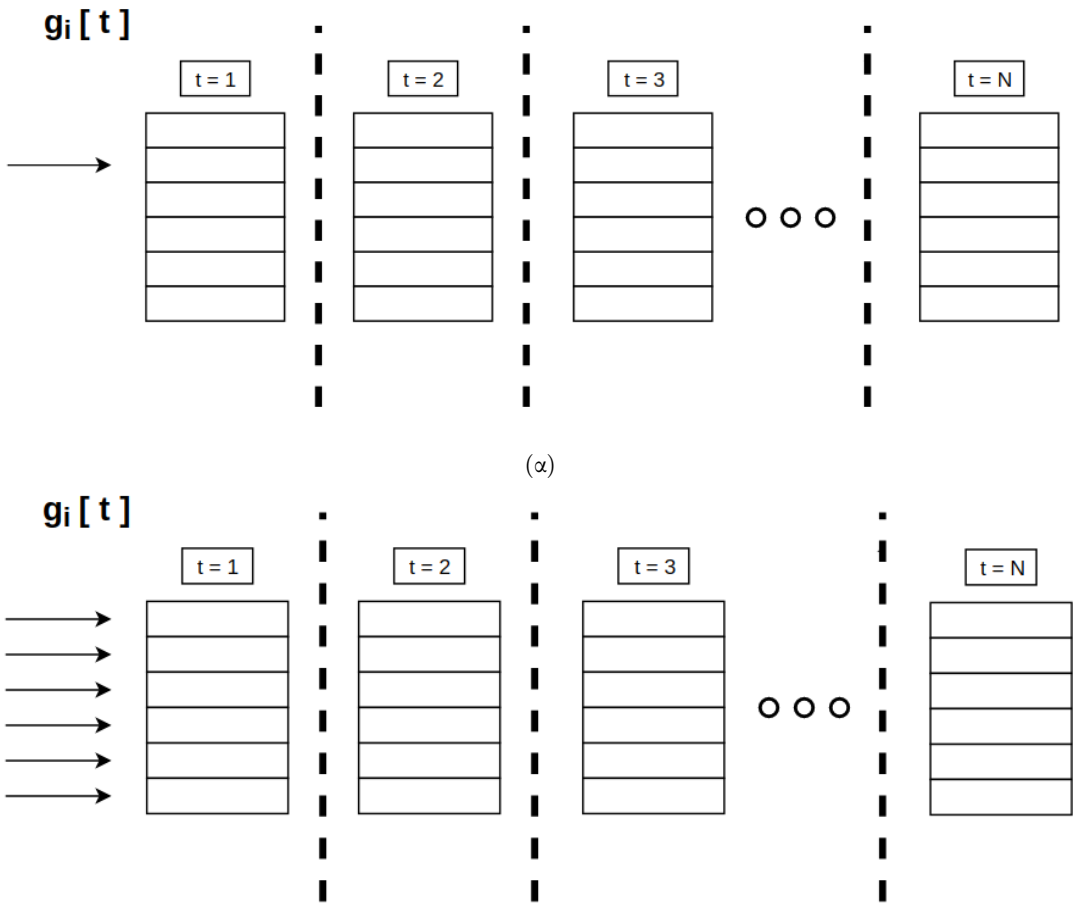
5.4 Δεύτερο επίπεδο παραλληλοποίησης

Εκτός από την εξάρτηση της κάθε επανάληψης της εκπαίδευσης με τις προηγούμενές της βασικό εμπόδιο για την προσπάθεια ανεύρεσης τμημάτων του κώδικα τα οποία είναι δυνατό να εκτελεστούν παράλληλα αποτέλεσε και η εξάρτηση των κάθετων τμημάτων της εκάστοτε εικόνας που χρησιμοποιείται ως δείγμα εκπαίδευσης. Η εφαρμογή συγκεντρώνει σε έναν πίνακα τις τιμές της εκάστοτε θύρας ή του κυττάρου μνήμης για όλα τα κύτταρα του επιπέδου για κάθε στοιχείο (κάθετο τμήμα) της ακολουθίας εισόδου (εικόνα), όπως φαίνεται και στο σχήμα 5.3.α. Για παράδειγμα, οι τιμές των κυττάρων της θύρας εισόδου για το κάθε LSTM υποδίκτυο συγκεντρώνονται σε έναν πίνακα 100 x N στοιχείων, όπου N είναι το μήκος της εκάστοτε ακολουθίας.

Όπως έχει αναφερθεί και στο προηγούμενο κεφάλαιο η έξοδος και κατά συνέπεια και η εκπαίδευση του νευρωνικού εξαρτώνται από την ακολουθία των κάθετων τμημάτων της εκάστοτε εικόνας. Σαν αποτέλεσμα αυτού κάθε τέτοιο τμήμα πρέπει να διέλθει από τις διαδικασίες της εκπαίδευσης αφού έχει τελειώσει το προηγούμενό του και έχει επηρεάσει στο βαθμό που του αντιστοιχεί την εκπαίδευση του δικτύου. Συνεπώς, ο υπολογισμός του $g_i[t]$, για παράδειγμα, δεν είναι δυνατό να παραλληλοποιηθεί ως προς τα κάθετα τμήματα του δείγματος. Ειδικότερα, καθώς το $g_i[t]$ εξαρτάται από το $g_i[t\text{-}1]$ δεν καθίσταται δυνατή η παράλληλη εκτέλεση πράξεων για τον υπολογισμό τους και αναγκαστικά η αποτίμηση τους γίνεται σειριακά. Ωστόσο, το κάθε κύτταρο του LSTM επιπέδου δεν εξαρτάται από τα υπόλοιπα κατά τη διάρκεια που αυτά επεξεργάζονται τα στοιχεία της ακολουθίας εισόδου παρά μόνο πρέπει να συγχρονίζονται με το πέρας αυτής, όπως φαίνεται και στο σχήμα 5.2, όπου απεικονίζεται σχηματικά η σειρά λειτουργιών της εκπαίδευσης. Δηλαδή, ο $g_i[t]$ εμπεριέχει τις τιμές της θύρας εισόδου για όλα τα LSTM κύτταρα για το συγκεκριμένο κάθετο τμήμα της ακολουθίας και σαν αποτέλεσμα αυτού ο υπολογισμός του $g_i^j[t]$ είναι ανεξάρτητος του $g_i^i[t]$, όπου $i \neq j$ καθώς αναφέρεται σε διαφορετικό LSTM κύτταρο. Επιτρέπεται, λοιπόν, η ομαδοποίηση τους και ο παράλληλος υπολογισμός του χαρακτηριστικού αυτού και των 100 κυττάρων για κάθε LSTM επίπεδο, όπως παρουσιάζεται και στο σχήμα 5.3.β.



Σχήμα 5.2: Σχηματική απεικόνιση του αλγορίθμου εκπαίδευσης.

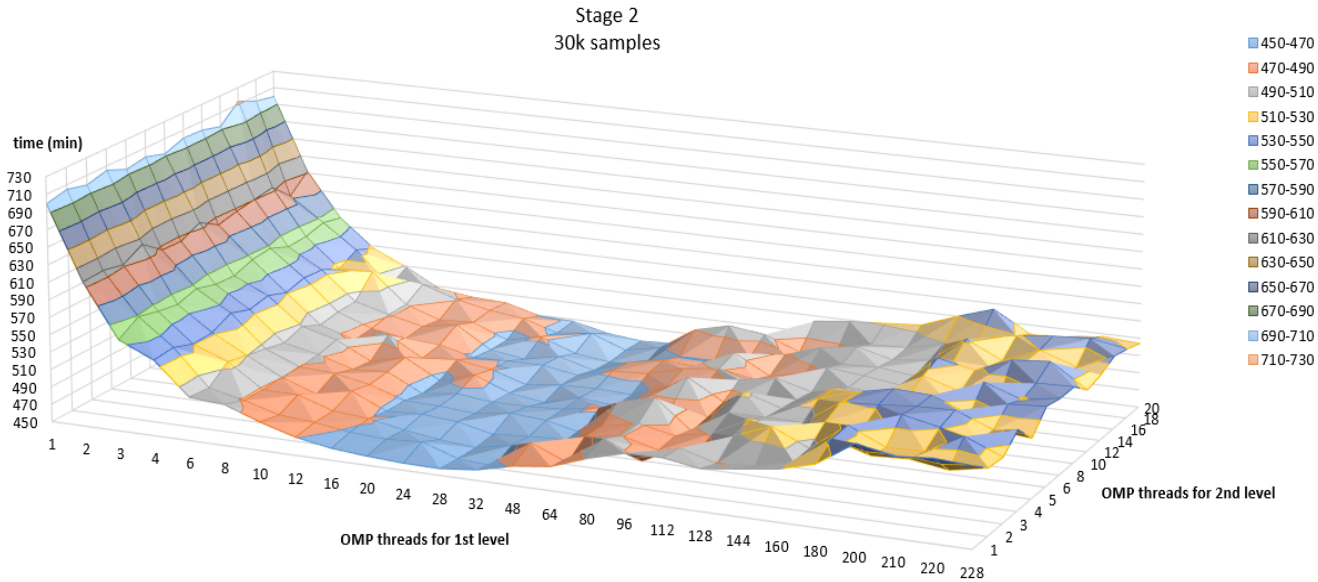


Σχήμα 5.3: (α) Ο υπολογισμός του $g_i[t]$ εξαρτάται από το $g_i[t-1]$ και για αυτό το λόγο δεν καθίσταται δυνατή η παραλληλοποίηση τους ως προς αυτή τη διάσταση. (β) Ο πίνακας $g_i[t]$ περιλαμβάνει τις τιμές της θύρας εισόδου για όλα τα LSTM κύτταρα του επιπέδου τα οποία μεταξύ τους είναι ανεξάρτητα επιτρέποντας την παράλληλη εκτέλεση των πράξεων που τα αφορούν.

Με τη βοήθεια του VTune που έχει αναφερθεί σε προηγούμενη ενότητα διαπιστώθηκε ότι τα σημεία στα οποία αναλωνόταν περισσότερο το πρόγραμμα και έχρηζε βελτίωσης ήταν τα σημεία της πρόσθιας και της οπίσθιας εκπαίδευσης του LSTM επιπέδου του νευρωνικού δικτύου και επιβεβαιώθηκε το γεγονός ότι δεν έγινε κάποια παράλειψη στην προσπάθεια παραλληλοποίησης όλων των χρονοβόρων συναρτήσεων.

Το OpenMP προσφέρει αρκετούς τρόπους για τον καθορισμό των OpenMP νημάτων που θα χρησιμοποιηθούν κατά την εκτέλεση του προγράμματος. Ένας από αυτούς είναι και η μεταβλητή περιβάλλοντος OMP_NUM_THREADS, η οποία παίρνει ως είσοδο μία λίστα αριθμών και επιτρέπει τη χρήση διαφορετικού πλήθους νημάτων σε κάθε επίπεδο παραλληλίας. Κάθε ένας από τους αριθμούς αυτούς καθορίζει τον αριθμό των νημάτων που θα χρησιμοποιηθούν στο επίπεδο παραλληλίας που του αντιστοιχεί. Αν τα επίπεδα παραλληλίας είναι περισσότερα από το πλήθος των αριθμών της λίστας τότε για όλα τα επίπεδα που δεν αντιστοιχίζονται με κάποιο πλήθος νημάτων ισχύει ο τελευταίος αριθμός της λίστας.

Στο διάγραμμα που ακολουθεί απεικονίζεται ο χρόνος εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης του νευρωνικού δικτύου στον Xeon Phi 3120P σε σχέση με τον αριθμό των OpenMP νημάτων που χρησιμοποιούνται για τα πρώτα δύο επίπεδα παραλληλίας.



Σχήμα 5.4: Μεταβολή του χρόνου εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης σε συνάρτηση με τον αριθμό των OpenMP νημάτων που χρησιμοποιούνται για το πρώτο και το δεύτερο επίπεδο παραλληλίας στον Xeon Phi 3120P.

Παρατηρείται ότι οι καλύτεροι χρόνοι εκτέλεσης παρουσιάζονται για πλήθος των OpenMP νημάτων του πρώτου επιπέδου από 16 έως 32 και για πλήθος από 2 έως 6 νήματα για το δεύτερο επίπεδο παραλληλίας. Πιο συγκεκριμένα ο καλύτερος χρόνος επιτυγχάνεται για 32 και 2 OpenMP νήματα για το πρώτο και το δεύτερο επίπεδο αντίστοιχα και μετρήθηκε ως 452 λεπτά.

Οι αντίστοιχοι χρόνοι στον Xeon Phi 7250 είναι αισθητά χαμηλότεροι (περίπου στο 20% των προαναφερθέντων). Για παράδειγμα για το ζευγάρι (32,2) το οποίο στον προηγούμενο συνεπεξεργαστή ήταν αυτό για το οποίο επιτυγχάνονταν η ταχύτερη επίδοση στα 452 λεπτά πλέον απαιτούνται 115 λεπτά για την εκτέλεση 30000 επαναλήψεων της εκπαίδευσης του δικτύου. Η πιο γρήγορη εκτέλεση στον Xeon Phi 7250 επιτυγχάνεται για 32 και 5 νήματα στο πρώτο και δεύτερο επίπεδο παραλληλίας και διαρκεί 96 λεπτά για 30000 επαναλήψεις.

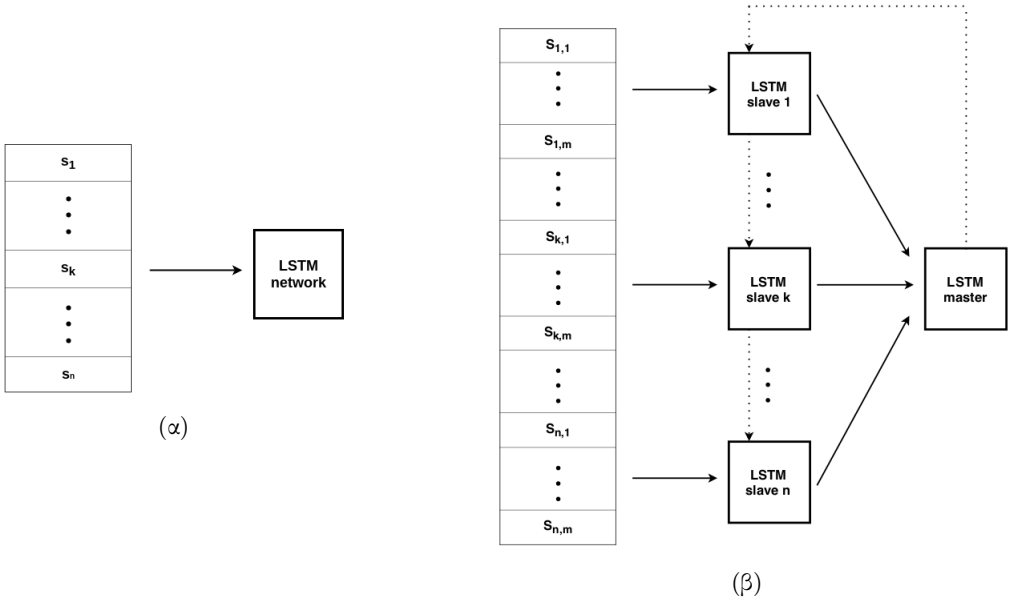
Στο επόμενο σχήμα παρουσιάζονται συνολικά οι χρόνοι εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης του δικτύου για τα δύο στάδια παραλληλοποίησης που περιγράφηκαν καθώς και ο χρόνος εκτέλεσης του σειριακού προγράμματος στον Xeon Phi 3120P.

Σχήμα 5.5: Διάρκεια εκπαίδευσης 30000 δειγμάτων για το σειριακό πρόγραμμα (stage 0) και τα δύο στάδια παραλληλοποίησης που περιγράφηκαν παραπάνω στον Xeon Phi 3120P.

Παρατηρείται ότι όσο μεγαλώνει ο αριθμός των νημάτων που χρησιμοποιούνται από κάποιο σημείο και έπειτα η βελτίωση είναι μικρή.

5.5 Τρίτο επίπεδο παραλληλοποίησης

Στο στάδιο αυτό έγινε η προσπάθεια παραλληλοποίησης της εκπαίδευσης του νευρωνικού δικτύου που μελετάται σε επίπεδο δειγμάτων. Αντίθετα με την έως τώρα εκπαίδευση όπου κάθε δείγμα επιλεγόταν διαδοχικά και διενεργούνταν παράλληλα μόνο οι εσωτερικές λειτουργίες της εκπαίδευσης του δικτύου πλέον τα δείγματα ομαδοποιούνται σχηματίζοντας ομάδες με μικρό αριθμό δειγμάτων (batches) οι οποίες χρησιμοποιούνται ως είσοδοι και βοηθούν στην περαιτέρω παραλληλοποίηση της εκπαίδευσης. Ειδικότερα, δημιουργούνται τόσα αντίγραφα του δικτύου όσες και οι διαφορετικές ομάδες που είναι επιθυμητό να εκτελούν ταυτόχρονα την εκπαίδευση ούτως ώστε με το πέρας της εκπαίδευσης του κάθε ξεχωριστού αντιγράφου με βάση την ομάδα δειγμάτων που του αντιστοιχεί τα αποτελέσματα, τα χαρακτηριστικά δηλαδή του εκάστοτε δικτύου, συνδυάζονται και αφού ενημερωθούν τα δίκτυα με τα νέα κοινά πλέον χαρακτηριστικά η εκπαίδευση τους συνεχίζεται ομοίως με νέες ομάδες δειγμάτων. Ο νέος τρόπος εκπαίδευσης του LSTM δικτύου παρουσιάζεται στο σχήμα 5.6.



Σχήμα 5.6: (α) Το δίκτυο όπως παρουσιάστηκε στα προηγούμενα στάδια. Η σειρά των δειγμάτων είναι τυχαία και προωθούνται σειριακά το ένα μετά το άλλο στο νευρωνικό δίκτυο. (β) Το σύστημα πλέον αποτελείται από n αντίγραφα του αρχικού δικτύου (slaves) τα οποία εκπαιδεύονται για συγκεκριμένο αριθμό δειγμάτων (batch size) ενημερώνουν το κυρίαρχο δίκτυο (master) και αφού ανανεωθούν με βάση τα νέα χαρακτηριστικά που προκύπτουν από το κυρίαρχο συνεχίζουν την εκπαίδευσή τους με τον ίδιο τρόπο.

Ο αλγόριθμος που επιτρέπει την παραπάνω διαδικασία βασίστηκε σε αυτόν που περιγράφεται στην εργασία [53]. Στη δημοσίευση αυτή παρουσιάζεται ο αλγόριθμος παραλληλοποίησης της SGD που χρησιμοποιείται και από το νευρωνικό δίκτυο της παρούσας εργασίας. Ο αλγόριθμος που ακολουθεί και χρησιμοποιήθηκε εν τέλει για την παρούσα εργασία είναι ελάχιστα παραλλαγμένος ώστε η ανταλλαγή πληροφοριών μεταξύ των δικτύων αντιγράφων να είναι συχνή κατά τη διάρκεια της εκτέλεσης.

Algorithm: Parallel SGD

Inputs: Examples $\{c^1, ..., c^m\}$, Learning Rate η, Machines κ, Batch Size β
Remaining examples $R = \{c^1, ..., c^m\}$;
$v = 0$ **while** $R \neq \emptyset$ **do**
  Randomly partition the examples in R, giving β examples to each machine;
  Update R;
  foreach $i \in \{1, ..., k\}$ **do**
    Randomly shuffle the data on machine i;
    Initialize $w_{i,0} = v$;
    foreach $t \in \{1, ..., \beta\}$ **do**
      Get the tth example on the ith machine (this machine), $c^{i,t}$;
      $w_{i,t} \leftarrow w_{i,t-1} - \eta \partial_w c^i(w_{i,t-1})$;
    end
  end
  Aggregate from all machines $v = \frac{1}{k} \sum_{i=1}^{k} w_{i,t}$;
end

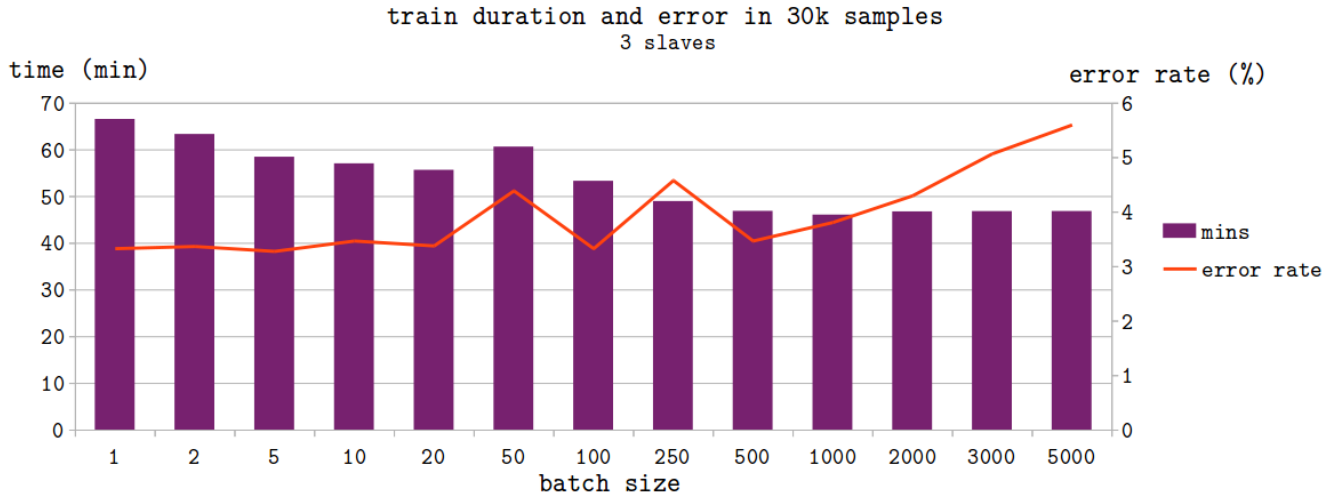
Για την επίτευξη του ίδιου ρυθμού σφάλματος με το αρχικό δίκτυο δεκαπλασιάστηκε ο ρυθμός εκμάθησης των δικτύων-αντιγράφων από 0.0001 σε 0.001.

Όπως ήταν αναμενόμενο το μέγεθος του batch επηρεάζει τόσο την ταχύτητα εκτέλεσης όσο και το ρυθμό σφάλματος του δικτύου. Παρατηρήθηκε ότι μετά από κάποιο σημείο η αύξηση του batch μειώνει όλο και περισσότερο το χρόνο που διαρκεί η εκπαίδευση και αυξάνει το ρυθμό σφάλματος. Όπως έχει αναφερθεί και σε προηγούμενες ενότητες η φύση τη εκπαίδευσης είναι σειριακή. Αυτό σημαίνει ότι το σφάλμα μειώνεται όλο και περισσότερο καθώς εκτελούνται οι επαναλήψεις της εκπαίδευσης η μία μετά την άλλη. Συνεπώς, όσο περισσότερο μειώνεται η μεταξύ τους ανταλλαγή δεδομένων τόσο λιγότερο βελτιώνεται η επίδοση του δικτύου.

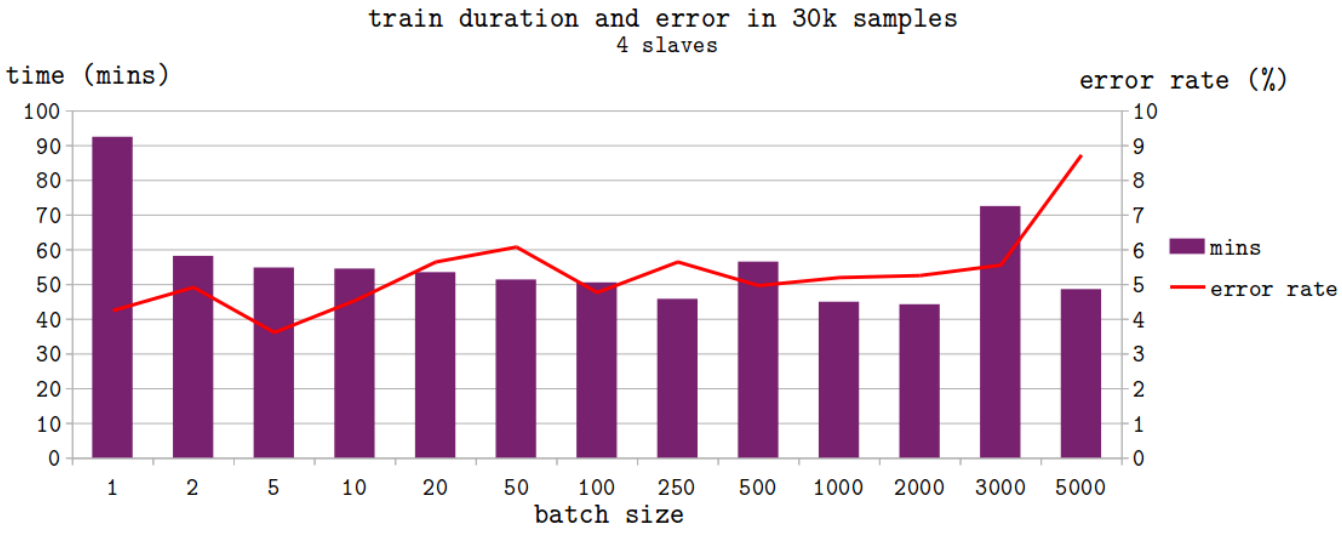
Στο σχήμα που ακολουθεί παρουσιάζονται οι χρόνοι εκπαίδευσης και οι ρυθμοί σφάλματος των εκάστοτε κυρίαρχων δικτύων ύστερα από 30000 επαναλήψεις συνολικά για διάφορα μεγέθη του batch για 2, 3, 4, 6 και 8 αντίγραφα του δικτύου (slaves). Ο καλύτερος συνδυασμός της διάρκειας της εκπαίδευσης και του ρυθμού σφάλματος του δικτύου προέκυψε για 2 αντίγραφα του νευρωνικού με batch μεγέθους 250 δειγμάτων. Με την επιλογή αυτή των παραμέτρων επιτεύχθηκε ρυθμός σφάλματος 2.5% σε 276 λεπτά για 30000 επαναλήψεις στον Xeon Phi 3120P. Ο αντίστοιχος χρόνος για τον Xeon Phi 7250 είναι 62 λεπτά.


train duration and error in 30k samples
2 slaves
time (mins)
error rate (%)
120
100
80
60
40
20
0
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
5
10
20
50
100
250
500
1000
2000
3000
5000
batch size
mins
error rate

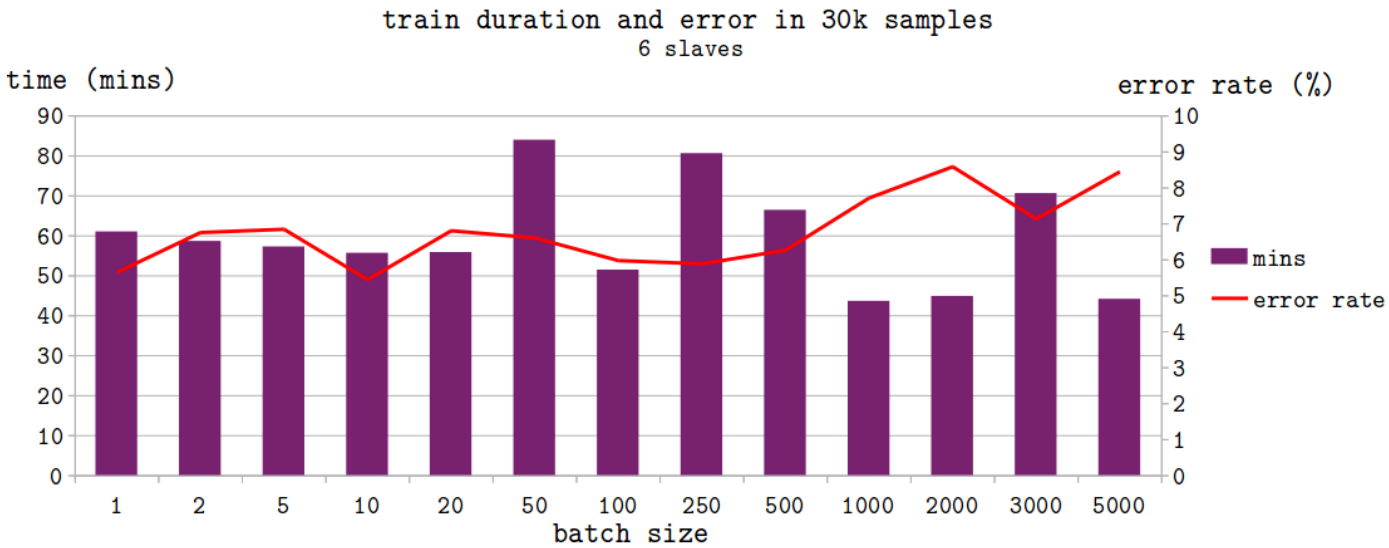
(α)


train duration and error in 30k samples
3 slaves
time (min)
error rate (%)
70
60
50
40
30
20
10
0
6
5
4
3
2
1
0
1
2
5
10
20
50
100
250
500
1000
2000
3000
5000
batch size
mins
error rate

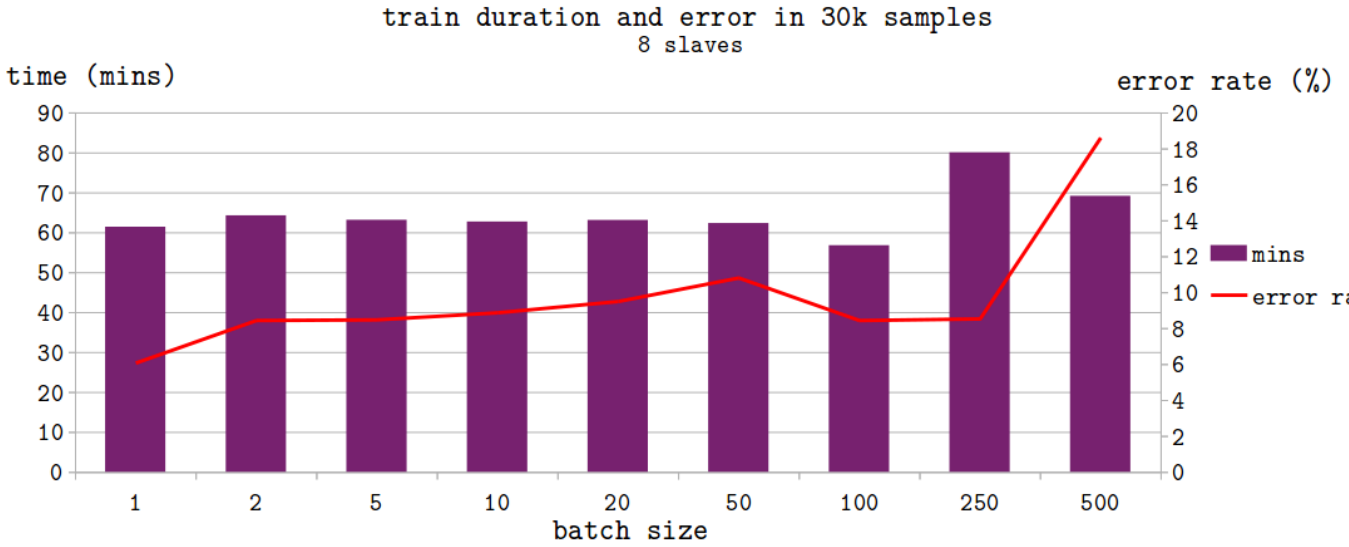
(β)


train duration and error in 30k samples
4 slaves
time (mins)
error rate (%)
100
90
80
70
60
50
40
30
20
10
0
10
9
8
7
6
5
4
3
2
1
0
1
2
5
10
20
50
100
250
500
1000
2000
3000
5000
batch size
mins
error rate

(γ)


train duration and error in 30k samples
6 slaves
time (mins)
error rate (%)
90
80
70
60
50
40
30
20
10
0
10
9
8
7
6
5
4
3
2
1
0
1
2
5
10
20
50
100
250
500
1000
2000
3000
5000
batch size
mins
error rate

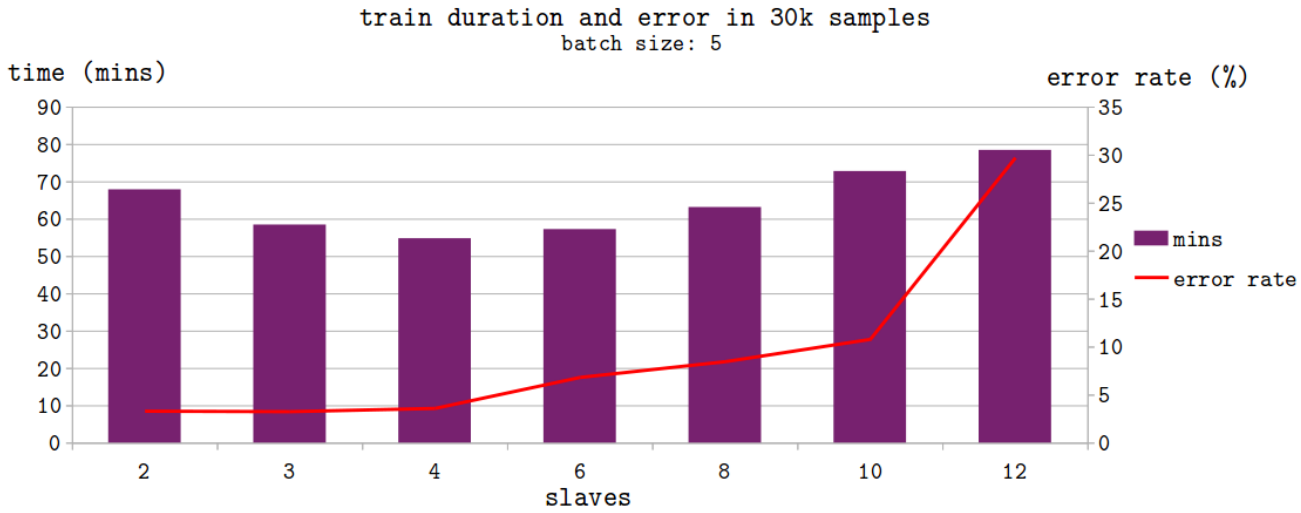
(δ)



(ε)

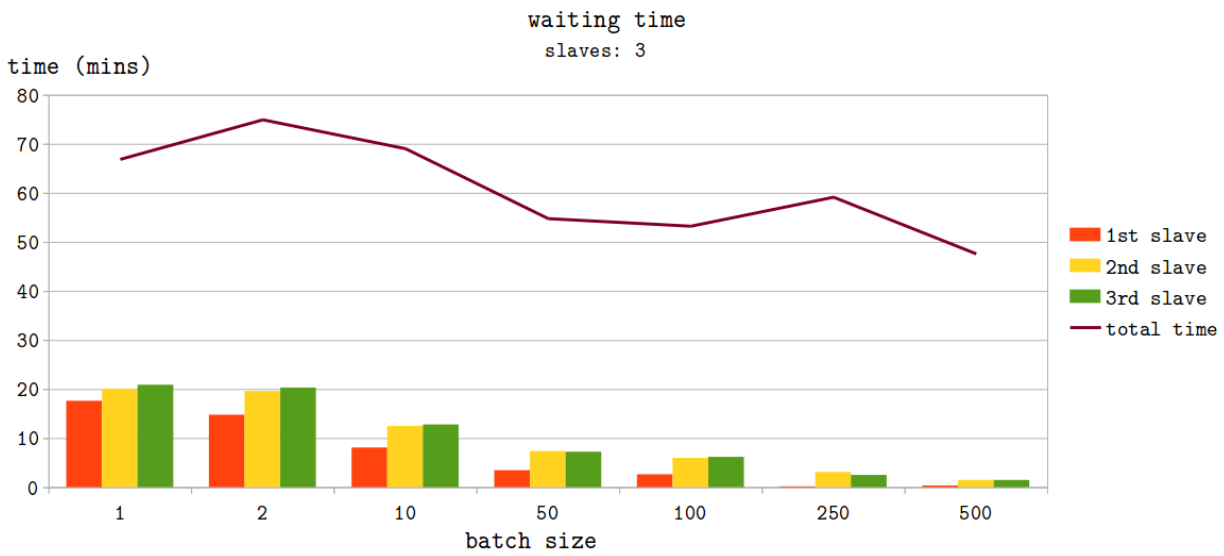
Σχήμα 5.11: Μεταβολή του ρυθμού σφάλματος και του χρόνου εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης του δικτύου σε συνάρτηση με το μέγεθος του batch διατηρώντας σταθερό τον αριθμό των slaves που χρησιμοποιούνται για τον Xeon Phi 7250.

Επίσης, παρατηρήθηκε ότι διατηρώντας σταθερό το μέγεθος του batch και αυξάνοντας τα αντίγραφα του δικτύου (slaves) που χρησιμοποιούνται ο χρόνος εκτέλεσης των επαναλήψεων μειώνεται διατηρώντας σταθερό το σφάλμα μέχρι ένα συγκεκριμένο αριθμό αντιγράφων. Ξεπερνώντας τον αριθμό αυτό των αντιγράφων αυξάνεται και ο χρόνος ολοκλήρωσης της εκπαίδευσης αλλά και ο ρυθμός σφάλματος του τελικού δικτύου (master). Στο διάγραμμα που ακολουθεί παρουσιάζονται η μεταβολή του ρυθμού σφάλματος και του χρόνου εκπαίδευσης του δικτύου συναρτήσει του πλήθους των slaves που χρησιμοποιούνται για batch μεγέθους πέντε δειγμάτων.



Σχήμα 5.12: Μεταβολή του ρυθμού σφάλματος και του χρόνου εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης του δικτύου σε συνάρτηση με τον αριθμό των slaves που χρησιμοποιούνται για batch μεγέθους πέντε δειγμάτων.

Ακόμα, μετρήθηκε η συνολική διάρκεια των διαστημάτων κατά τα οποία το κάθε επιμέρους δίκτυο (slave) ενημερώνει το κυρίαρχο δίκτυο (master) ύστερα από την εκπαίδευση της υποομάδας των δειγμάτων του, ανανεώνει τα χαρακτηριστικά του σύμφωνα με τα αντίστοιχα του κυρίαρχου και περιμένει ανενεργό με σκοπό το συγχρονισμό του με τα υπόλοιπα. Το πρώτο αντίγραφο αναλαμβάνει την ενημέρωση του κυρίαρχου δικτύου και έτσι οι χρόνοι αναμονής του είναι μειωμένοι. Στο παρακάτω σχήμα παρουσιάζονται οι χρόνοι αναμονής των τριών slaves για 30000 επαναλήψεις της εκπαίδευσης.



Σχήμα 5.13: Διάρκεια εκτέλεσης 30000 επαναλήψεων της εκπαίδευσης του δικτύου με 3 slaves και διάρκεια άεργης αναμονής του καθενός για την επίτευξη συγχρονισμού.

Παρατηρείται το γεγονός ότι όσο μικρότερο είναι το μέγεθος του batch τόσο μεγαλύτεροι είναι οι χρόνοι αναμονής των δικτύων-αντιγράφων. Αυτό συμβαίνει καθώς η διάρκεια εκπαίδευσης με τη χρήση του κάθε δείγματος εξαρτάται από το μέγεθός του (μήκος εικόνας). Συνεπώς, καθώς τα δείγματα επιλέγονται τυχαία, αυξάνοντας το μέγεθος του batch μειώνονται οι διαφορές της διάρκειας του γύρου εκπαίδευσης μεταξύ των slaves.

Κεφάλαιο 6

Επίλογος

Στο τελευταίο αυτό κεφάλαιο συνοψίζεται η μελέτη που εκπονήθηκε στα πλαίσια της παρούσας διπλωματικής εργασίας. Επίσης, προτείνονται κάποιες κατευθυντήριες γραμμές στις οποίες μπορεί να κινηθεί η μελλοντική έρευνα.

6.1 Σύνοψη και συμπεράσματα

Στο πλαίσιο της εκπόνησης της παρούσας εργασίας μελετήθηκε η λειτουργία των LSTM νευρωνικών δικτύων, τα παράλληλα συστήματα επεξεργασίας και η ανάπτυξη παράλληλων αλγορίθμων. Επιπλέον, αναπτύχθηκε ένας αλγόριθμος παράλληλης εκτέλεσης, που αποτελείται από τρία επίπεδα, για την εκπαίδευση ενός LSTM δικτύου το οποίο χρησιμοποιείται σε εφαρμογή οπτικής αναγνώρισης χαρακτήρων (OCR). Ακόμα, αξιολογήθηκαν πειραματικά τα τρία επίπεδα από τα οποία αποτελείται ο αλγόριθμος αυτός με τη χρήση σύγχρονων συνεπεξεργαστών. Σαν αποτέλεσμα αυτών, επιτεύχθηκε σημαντική επιτάχυνση της εκπαίδευσης ενός LSTM δικτύου. Οι ιδέες και οι τεχνικές που παρατίθενται στην παρούσα εργασία μπορούν να χρησιμοποιηθούν και πιθανώς να οδηγήσουν σε αποτελεσματικότερους αλγορίθμους.

6.2 Μελλοντικές επεκτάσεις

Η χρήση του LSTM δικτύου με κάποιες παραλλαγές όπως η χρήση ενός κοινού Softmax επιπέδου ή η από κοινού ανανέωσή τους είναι πιθανό να απέφερε αποτελέσματα τα οποία θα ήταν χρήσιμα σε συστήματα μεγαλύτερης κλίμακας. Ακόμα, ενδιαφέρον θα είχε η αξιολόγηση της επίδοσης του προγράμματος με τη χρήση ενός διαφορετικού αλγορίθμου παραλληλοποίησης της SGD από αυτούς που παρέχει η σχετική βιβλιογραφία ή ακόμα και η ανάπτυξη ενός. Η υλοποίηση των σταδίων παραλληλοποίησης που προτάθηκαν από την παρούσα εργασία μπορεί να γίνει και σε GPUs με τη χρήση των κατάλληλων εργαλείων. Τέλος, καθώς χρησιμοποιήθηκε το μοντέλο κοινής μνήμης για την ανάπτυξη και την εκτέλεση των παράλληλων αλγορίθμων δίνεται η δυνατότητα μετατροπής της εκπαίδευσης του νευρωνικού δικτύου ώστε να χρησιμοποιείται το μοντέλο ανταλλαγής μηνυμάτων με τη χρήση των απαραίτητων εργαλείων όπως το MPI και να εκτελείται σε συστήματα κατανεμημένης μνήμης.

Βιβλιογραφία

[1] List of TOP500 Supercomputers.
https://www.top500.org/list/2017/11/, [Online; accessed on 11/6/2018].

[2] Intel® Xeon Phi™ Processors.
https://www.intel.com/content/www/us/en/products/processors/xeon-phi/xeon-phi-processors.html, [Online; accessed on 11/6/2018].

[3] Simon Haykin. "Neural networks and learning machines", *Pearson Education*, 2009.

[4] Micahel J. Flynn. "Some Computer Organizations and Their Effectiveness", *IEEE Transactions on Computers* C–21 (9): 948–960, 1972.

[5] Gene M. Amdahl. "The validity of the single processor approach to achieving large scale computing capabilities", *AFIPS Spring Joint Computer Conference*, AFIPS Press, 1967.

[6] Computing System Laboratory of NTUA. "Parallel Processing Systems", *NTUA*, 2012.
http://www.cslab.ntua.gr/courses/pps/files/fall2014/pps-notes.pdf, [Online; accessed on 11/6/2018].

[7] Ian Foster. "A Parallel Programming Model", *Argonne National Laboratory*, "Designing and Building Parallel Programs", Section 1.3, 1995.
http://www.mcs.anl.gov/~itf/dbpp/text/node9.html, [Online; accessed on 11/6/2018].

[8] Blaise Barney. "Models", *Lawrence Livermore National Laboratory*, "Introduction to Parallel Computing", 2015.
https://computing.llnl.gov/tutorials/parallel_comp/\#Models, [Online; accessed on 11/6/2018].

[9] The Open Group Base Specifications Issue 7, IEEE Std 1003.1-2008, 2016 Edition.
http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html, [Online; accessed on 11/6/2018].

[10] Homepage of OpenMP.
http://www.openmp.org/, [Online; accessed on 11/6/2018].

[11] Homepage of Intel® Cilk™ Plus.
https://www.cilkplus.org/, [Online; accessed on 11/6/2018].

[12] The Cilk Project. *MIT CSAIL*, 2010.
http://supertech.csail.mit.edu/cilk/, [Online; accessed on 11/6/2018].

[13] Robert D. Blumofe, Christopher F. Joerg, Bradley C. Kuszmaul, Charles E. Leiserson, Keith H. Randall, Yuli Zhou. "Cilk: An efficient multithreaded runtime system", *Parallel and Distributed Computing* 37 (1): 55–69, 1996.

[14] Homepage of Intel® Threading Building Blocks (Intel® TBB).
https://www.threadingbuildingblocks.org/, [Online; accessed on 11/6/2018].

[15] Homepage of MPICH.
http://www.mpich.org/, [Online; accessed on 11/6/2018].

[16] Homepage of CUDA® Zone.
https://developer.nvidia.com/cuda-zone, [Online; accessed on 11/6/2018].

[17] Intel® Nervana™ Neural Network Processor.
https://ai.intel.com/intel-nervana-neural-network-processor/, [Online; accessed on 11/6/2018].

[18] Accelerating Deep Convolutional Neural Networks Using Specialized Hardware.
https://www.microsoft.com/en-us/research/publication/accelerating-deep-convolutional-neural-networks-using-specialized-hardware/, [Online; accessed on 11/6/2018].

[19] Bruno Ferreira. "Google boosts machine learning with its Tensor Processing Unit", *Techreport*, 2016.
https://techreport.com/news/30155/google-boosts-machine-learning-with-its-tensor-processing-unit, [Online; accessed on 11/6/2018].

[20] NVIDIA's GeForce GTX 1080.
https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080/, [Online; accessed on 11/6/2018].

[21] Top 10 supercomputers for November 2017.
https://www.top500.org/lists/2017/11/, [Online; accessed on 11/6/2018].

[22] Jack Dongarra. "Visit to the National University for Defense Technology Changsha, China", *Netlib*, 2013.
http://www.netlib.org/utk/people/JackDongarra/PAPERS/tianhe-2-dongarra-report.pdf, [Online; accessed on 11/6/2018].

[23] Intel® Xeon Phi™ 7250.
https://ark.intel.com/products/94035/Intel-Xeon-Phi-Processor-7250-16GB-1_40-GHz-68-core, [Online; accessed on 11/6/2018].

[24] Intel® Xeon Phi™ 3120P.
https://ark.intel.com/products/75798/Intel-Xeon-Phi-Coprocessor-3120P-6GB-1_100-GHz-57-core, [Online; accessed on 11/6/2018].

[25] Mike P. (sic). "An Intro to MCDRAM (High Bandwidth Memory) on Knights Landing", *software.intel.com*, 2016.
https://software.intel.com/en-us/articles/mcdram-high-bandwidth-memory-on-knights-landing-analysis-methods-tools, [Online; accessed on 11/6/2018].

[26] Intel® Fortran Compiler 18.0 Developer Guide and Reference.
https://software.intel.com/en-us/download/intel-fortran-compiler-180-developer-guide-and-reference, [Online; accessed on 11/6/2018].

[27] Intel® C++ Compiler 18.0 Developer Guide and Reference.
https://software.intel.com/sites/default/files/managed/03/3f/DevGuide_CPP_Compiler_18.3.pdf, [Online; accessed on 11/6/2018].

[28] Ronald W. Green. "Native and Offload Programming Models", *Intel*, 2012.
https://software.intel.com/en-us/articles/native-and-offload-programming-models, [Online; accessed on 11/6/2018].

[29] Homepage of AlphaGo.
https://deepmind.com/research/alphago/, [Online; accessed on 11/6/2018].

[30] Marcel van Gerven, Sander Bohte. "Artificial Neural Networks as Models of Neural Information Processing",*Frontiers in Computational Neuroscience*,2018.
https://www.frontiersin.org/research-topics/4817/artificial-neural-networks-as-models-of-neural-information-processing\#overview, [Online; accessed on 11/6/2018].

[31] Yann LeCun, Yoshua Bengio. "Convolutional networks for images, speech, and time-series",*The Handbook of Brain Theory and Neural Networks*, MIT Press, 1995.

[32] Feed-Forward networks. "Convolutional networks for images, speech, and time-series",*The Handbook of Brain Theory and Neural Networks*, MIT Press, 1995.
https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/feedforward.html, [Online; accessed on 11/6/2018].

[33] Danilo P. Mandic, Jonathon Chambers. "Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability", Wiley, 2001.

[34] Teuvo Kohonen. "Self-Organized Formation of Topologically Correct Feature Maps", *Biological Cybernetics*, 1982.

[35] D. S. Broomhead, David Lowe. "Radial basis functions, multi-variable functional interpolation and adaptive networks", RSRE, 1988.

[36] Guy Desjardins, Robert Proulx, Robert Godin. "An Auto-Associative Neural Network for Information Retrieval", International Joint Conference on Neural Networks, 2006.

[37] Kevin Gurney. "An Introduction to Neural Networks", Routledge, 2002.

[38] Sepp Hochreiter, Jürgen Schmidhuber. "Long short-term memory", *Neural Computation*, pages: 1735–1780, MIT Press, 2006.

[39] Felix A. Gers, Jürgen Schmidhuber, Fred Cummins. "Learning to Forget: Continual Prediction with LSTM", *Neural Computation*, pages: 2451–2471, MIT Press, 2000.

[40] Matt Mahoney. "Large Text Compression Benchmark", 2006. `http://www.mattmahoney.net/dc/text.html\#1218`, [Online; accessed on 11/6/2018].

[41] Alex Graves, Abdel-rahman Mohamed, Geoffrey Hinton. "Speech Recognition with Deep Recurrent Neural Networks", 2013. `https://arxiv.org/pdf/1303.5778.pdf`, [Online; accessed on 11/6/2018].

[42] Chris Smith. "iOS 10: Siri now works in third-party apps, comes with extra AI features",*BGR*, 2016. `http://bgr.com/2016/06/13/ios-10-siri-third-party-apps/`, [Online; accessed on 11/6/2018].

[43] Françoise Beaufays. "The neural networks behind Google Voice transcription",*Research Blog*, 2015. `https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html`, [Online; accessed on 11/6/2018].

[44] Amir Efrati. "Apple's Machines Can Learn Too", *The Information*, 2016. `https://www.theinformation.com/articles/apples-machines-can-learn-too`, [Online; accessed on 11/6/2018].

[45] Arindam Chaudhuri, Krupa Mandaviya, Pratixa Badelia, Soumya K. Ghosh. "Optical Character Recognition Systems for Different Languages with Soft Computing", Springer, 2017.

[46] Arnold Rokus, Miklos Poth. "Character recognition using neural networks", *CINTI*, 2010.

[47] Faisal Mohammad, Jyoti Anarase, Milan Shingote, Pratik Ghanwat. "Optical Character Recognition Implementation Using Pattern Matching", *IJCSIT*, Vol. 5 (2), 2014.

[48] Thomas M. Breuel. "Benchmarking of LSTM Networks", *Google*, 2015. `https://arxiv.org/pdf/1508.02774.pdf`, [Online; accessed on 11/6/2018].

[49] Robert M. Haralick. "UW-III English/Technical Document Image Database", Intelligent Systems Laboratory, University of Washington, 1995. `http://isis-data.science.uva.nl/events/dlia//datasets/uwash3.html`, [Online; accessed on 11/6/2018].

[50] Christopher M. Bishop. "Pattern Recognition and Machine Learning", Springer, 2006.

[51] Alex Graves, Santiago Fernandez, Faustino Gomez. "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks", *Proceedings of the International Conference on Machine Learning*, ICML 2006: 369–376, 2006.

[52] Douglas M Hawkins. "The Problem of Overfitting", *Journal of Chemical Information and Computer Sciences*, 44(1):1-12, 2004.

[53] Martin Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. "Parallelized stochastic gradient descent", *Advances in Neural Information Processing Systems* 23 : 2595–2603, 2010.

[54] Eigen library. `http://eigen.tuxfamily.org/index.php?title=Main_Page`, [Online; accessed on 11/6/2018].