ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αποδοτική πρόβλεψη ταχύτητας εκτέλεσης τμημάτων κώδικα με τεχνικές μηχανικής μάθησης.

Διπλωματική Εργασία
του Αϊβαλή Ν.Θεόδωρου

Επιβλέπων καθηγητής: Διονύσιος Ν. Πνευματικάτος
Καθηγητής Ε.Μ.Π.

Εργαστήριο Υπολογιστικών Συστημάτων
Αθήνα, Ιούλιος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Αποδοτική πρόβλεψη ταχύτητας εκτέλεσης τμημάτων κώδικα με τεχνικές μηχανικής μάθησης.

Διπλωματική Εργασία
του Αϊβαλή Ν.Θεόδωρος

Επιβλέπων καθηγητής: Διονύσιος Ν. Πνευματικάτος
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Ιουλίου 2023

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....
Διονύσιος Πνευματικάτος	**Νεκτάριος Κοζύρης**	**Γεώργιος Γκούμας**
Καθηγητής Ε.Μ.Π.	**Καθηγητής Ε.Μ.Π.**	**Αναπλ. Καθηγητής Ε.Μ.Π.**

Αθήνα, Ιούλιος 2023

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

(Υπογραφή)

.....
Αϊβαλής Θεόδωρος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

Περίληψη

Η επιστήμη που ασχολείται με την ανάπτυξη και τη χρήση των νευρωνικών δικτύων αναπτύσσεται τα τελευταία χρόνια με πολύ γρήγορου ρυθμούς. Ο συγκεκριμένος τομέας παρότι μελετάται εδώ και πολλά χρόνια, έμεινε πρακτικά στάσιμος και προχώρησε τα τελευταία χρόνια λόγω της ραγδαίας αύξησης της ποσότητας των δεδομένων αλλά και των υπολογιστικών πόρων. Η επιστήμη της μηχανικής μάθησης χρησιμοποιείται στην όραση υπολογιστών, την ιατρική και γενικότερα όσους τομείς ασχολούνται με την επεξεργασία και την ανάλυση των δεδομένων.
Βασική ενασχόληση της εργασίας αυτής είναι η μελέτη της επίδοσης διαφορετικών επεξεργαστών μέσω της παρουσίασης και της πειραματικής αξιολόγησης εργαλείων πρόβλεψης - εκτίμησης της απόδοσης για ένα συγκεκριμένο τμήμα κώδικα με χρήση μεθόδων μηχανικής μάθησης. Η πρόβλεψη του αριθμού των κύκλων ρολογιού που απαιτεί η εκτέλεση ενός τμήματος κώδικα (throughput) μπορεί να χρησιμοποιηθεί από σχεδιαστές μεταγλωττιστών με σκοπό την δημιουργία του λογισμικού αλλά και για τους ίδιους τους προγραμματιστές που θέλουν να βρουν μια βέλτιστη υλοποίηση.
Αρχικά γίνεται η παρουσίαση τόσο για το βασικό μοντέλο με το οποίο ασχολείται η διπλωματική, δηλαδή το Ithemal, όσο και για τα άλλα κλασικά αλλά και σύγχρονα εργαλεία με τα οποία αργότερα θα συγκριθεί το βασικό μας μοντέλο. Στη συνέχεια αναλύονται τα θέματα των νευρωνικών δικτύων και συγκεκριμένα τόσο τα συνήθη χρησιμοποιούμενα νευρωνικά όσο και αυτά που χρησιμοποιεί το μοντέλο του Ithemal. Γίνεται επίσης ανάλυση ενός σύγχρονου νευρωνικού δικτύου, του Transformer το οποίο δοκιμάστηκε και στην πράξη με σκοπό να παρουσιάσει μια εναλλακτική προσέγγιση. Έπειτα, αναλύεται το σύνολο δεδομένων που χρησιμοποιείται για την αξιολόγηση του μοντέλου μαζί με τους λόγους για τους οποίους επιλέχθηκε και καταλήγουμε στην ανάλυση του μοντέλου και των πειραμάτων καθώς και στην αξιολόγηση των αποτελεσμάτων που προέκυψαν. Τα πειράματα αυτά αφορούν τόσο τη δοκιμή του μοντέλου στους διάφορους διαθέσιμους επεξεργαστές όσο και την αξιολόγηση των επεξεργαστών κατά την μεταφορά χωρίς επανεκπαίδευση. Όμοια διαδικασία έγινε και για τους Transformer, που επιλέχθηκε με σκοπό της βελτίωσης της επίδοσης του Ithemal. Καταλήγουμε στην ανακεφαλαίωση και την παρουσίαση πιθανών μελλοντικών επεκτάσεων.

Λέξεις Κλειδιά

Τεχνητά νευρωνικά δίκτυα, μηχανική μάθηση, αρχιτεκτονική υπολογιστών, πρόβλεψη throughput, Ithemal, BHive Dataset, Transformers, uiCA, nanoBench.

Abstract

Machine learning and neural networks have made great progress in recent years. Although scientists have made multiple studies in this field, the last few years have evolved. The main reason is the rise of the available data and computer resources. The science of machine learning uses its results in computer vision, medicine and other fields of data analysis.
In this thesis, a neural-network-based model is used to predict throughput of different basic blocks. This throughput estimation problem is a basic field of work for both researchers and compiler designers.
First of all, the basic model of this thesis, ithemal, is presented and there are given details for traditional models that were used all the previous years. Afterwards, aspects of neural networks are analyzed. Except for commonly used networks, there are also presented those of Ithemal. A modern neural network, called Transformer, is presented as an alternative version to predict the throughput of basic blocks. The dataset used to train and evaluate each model is the next issue analyzed. Finally, the main model is presented with the experiments made to evaluate the extracted results. Ithemal was installed and trained over the different CPUs and then was made an attempt to move the trained model without re-training it. The same experiments were made using Transformer instead of Ithemal model.
Concluding, we summarize the research findings and propose future studies.

Key Words

Artificial neural networks, machine learning, computer architecture, throughput estimation, Ithemal, BHive Dataset, Transformers, uiCA, nanoBench.

Ευχαριστίες

Θα ήθελα αρχικά να ευχαριστήσω τον επιβλέποντα καθηγητή κ. Διονύσιο Πνευματικάτο για την επίβλεψη της συγκεκριμένης διπλωματικής εργασίας καθώς και για την δυνατότητα που μου έδωσε να την εκπονήσω στο εργαστήριο Τεχνολογίας Πληροφορικής και Υπολογιστών (CSLAB). Επίσης, ευχαριστώ ιδιαίτερα τον κ. Κωνσταντίνο Νίκα για την συνεχή καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε σε όλη τη διάρκεια της διπλωματικής. Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια των σπουδών μου.

Αϊβαλής Θεόδωρος

Περιεχόμενα

Περίληψη **7**
Abstract **9**
Ευχαριστίες **11**
Περιεχόμενα **13**
Κατάλογος Σχημάτων **16**
Κατάλογος Πινάκων **17**
Κεφάλαιο 1 - Εισαγωγή **18**
1.1 Η εκτίμηση της απόδοσης του κώδικα. 18
1.2 Αντικείμενο διπλωματικής 18
1.3 Διάρθρωση της Διπλωματικής 19
Κεφάλαιο 2 - Παρουσίαση τεχνικών και εργαλείων εκτίμησης της απόδοσης ενός τμήματος κώδικα. **20**
2.1 Κλασικές τεχνικές εκτίμησης απόδοσης κώδικα 20
2.1.1 IACA 20
2.1.1.1 Λειτουργία και modes λειτουργίας 20
2.1.2 LLVM-MCA 21
2.1.2.1 Βήματα - Στάδια 21
2.1.3 Open Source Architecture Code Analyzer (OSACA) 21
2.2 Παρουσίαση Σύγχρονων Εργαλείων Εκτίμησης Απόδοσης 22
2.2.1 uiCA 22
2.2.1.1 Πλεονεκτήματα και παραδοχές του μοντέλου 22
2.2.1.2 Αρχιτεκτονική των Intel επεξεργαστών 22
2.2.1.3 Δεδομένα και τρόποι αξιολόγησης 23
2.2.2 Ithemal 23
2.2.2.1 Αρχιτεκτονική του Ithemal 24
2.2.2.2 Αποτελέσματα του Ithemal 24
2.2.3 DiffTune 25
2.2.3.1 Προσέγγιση DiffTune και αποτελέσματα 25
2.2.3.2 Αλγόριθμος DiffTune 25
2.2.4 SimNet 26
2.2.4.1 Αρχιτεκτονική Νευρωνικού δικτύου 26
2.2.4.2 Αξιολόγηση Μοντέλου SimNet και σύγκριση με το Ithemal 27
2.2.4.3 Παράλληλο Simulation 28
Κεφάλαιο 3 - Μηχανική Μάθηση και Τεχνητά Νευρωνικά Δίκτυα **29**
3.1 Τύποι Αλγορίθμων Μηχανικής Μάθησης 29
3.1.1 Επιβλεπόμενη Μάθηση (Supervised learning) 29
3.1.1.1 Λογιστική παλινδρόμηση (Logistic Regression) 29
3.1.1.2 Κατηγοριοποίηση (Classification) 29
3.1.2 Μη Επιβλεπόμενη Μάθηση (Unsupervised learning) 30
3.1.3 Ενισχυτική μάθηση (Reinforcement learning) 30

3.2 Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks) 30
3.2.1 Τεχνητοί νευρώνες - Perceptrons 31
3.2.2 Πολυεπίπεδα Perceptron - Multi Layer Perceptron 31
3.3 Εκπαίδευση Νευρωνικών Δικτύων 32
3.3.1 Συνάρτηση Ενεργοποίησης (Activation Function) 32
3.3.2 Συνάρτηση Κόστους (Cost Function) 34
3.3.3 Αλγόριθμος Βελτιστοποίησης (Optimization Algorithm) 34
3.3.3.1 Αλγόριθμος Καθόδου με βάση την Κλίση (Gradient Descent) 35
3.4 Συνελικτικά νευρωνικά δίκτυα 36
3.5 Επαναλαμβανόμενα νευρωνικά δίκτυα 36
3.5.1 Δίκτυα μακράς βραχυπρόθεσμης μνήμης 37
3.5.3 Μηχανισμός Προσοχής - Transformer 38
3.5.3.1 Regression Transformer 39
Κεφάλαιο 4 - Ανάλυση Dataset 40
4.1 Περιγραφή ανάγκης δημιουργίας BHive Dataset 40
4.1.1 Απαιτήσεις και δημιουργία του BHive Dataset 40
4.1.2 Ομαδοποίηση του BHive Dataset 41
4.2 Αξιολόγηση του συνόλου δεδομένων BHive 43
4.2.1 Παραδοχές του BHive Dataset 44
4.3 Παραλλαγή του BHive Dataset(BHiveU & BHiveL) 44
Κεφάλαιο 5 - Αναλυτική Παρουσίαση του Ithemal 45
5.1 Εισαγωγή και Ανάγκες για την ανάπτυξη του Ithemal 45
5.2 Αρχιτεκτονική Μοντέλου Ithemal 46
5.2.1 Στάδια Αρχιτεκτονικής Μοντέλου Ithemal 46
5.2.1.1 Στάδιο κανονικοποίησης (Canonicalization Stage) 46
5.2.1.2 Στάδιο Εμφύτευσης (Embedding Stage) 46
5.2.1.3 Στάδιο Πρόβλεψης (Prediction Stage) 47
5.3 Δεδομένα και εκπαίδευση 47
5.4 Αξιολόγηση Μοντέλου 48
5.5 Αρχιτεκτονικές Νευρωνικού για το Ithemal 50
Κεφάλαιο 6 - Πειραματική Αξιολόγηση και Παρουσίαση Αποτελεσμάτων 52
6.1 Μετρικές αξιολόγησης αποτελεσμάτων 52
6.1.1 Ακρίβεια (Accuracy) 52
6.1.2 Γεωμετρικός μέσος (Geometric mean) 52
6.1.3 Kendall's Tau 53
6.1.4 Spearman’s Rank Correlation 53
6.1.5 Μέση απόλυτη διαφορά (MAD) 53
6.2 Διαδικασία εύρεσης Παράταιρων Τιμών (Outliers) 54
6.3 Περιγραφή πειραματικής διαδικασίας 54
6.3.1 Παρουσίαση επεξεργαστών και εργαλείων μέτρησης 54
6.3.1.2 Πλατφόρμες-επεξεργαστές μελέτης 54
6.3.1.2 Παρουσίαση εργαλείων μέτρησης 55
6.3.2 Μελέτη επίδοσης Ithemal 58
6.3.2.1 Διαδικασία εκπαίδευσης 58

6.3.2.2 Αποτελέσματα επίδοσης Ithemal 59
6.3.3 Αξιολόγηση επίδοσης ενός επεξεργαστή με προεκπαιδευμένο μοντέλο 61
6.3.3.1 Περιγραφή διαδικασίας αξιολόγησης της επίδοσης ενός επεξεργαστή με μοντέλο ενός άλλου 61
6.3.3.2 Παρουσίαση αποτελεσμάτων και εξαγωγή συμπερασμάτων 61
6.4 Χρήση Transformer για την βελτίωση επίδοσης του Ithemal 63
6.4.1 Διαδικασία εκπαίδευσης Transformer 63
6.4.2 Παρουσίαση αποτελεσμάτων και συμπεράσματα 63
6.4.3 Αξιολόγηση επίδοσης ενός επεξεργαστή με προεκπαιδευμένο Transformer 64
Κεφάλαιο 7 - Επίλογος και Μελλοντικές επεκτάσεις 67
7.1 Συμπεράσματα 67
7.2 Μελλοντικές επεκτάσεις 68
7.2.1 Βελτιώσεις σε επίπεδο διαχείρισης κώδικα 68
68
7.2.2 Βελτιώσεις σε επίπεδο υλικού 68
7.2.3 Βελτιώσεις σε επίπεδο αρχιτεκτονικής 68
Βιβλιογραφία 69

Κατάλογος Σχημάτων

Σχήμα 2.1 : Βασική λειτουργία llvm-mca 21
Σχήμα 2.2: Αρχιτεκτονική του Ithemal 24
Σχήμα 2.3: Ροή εργασιών στο μοντέλο της μηχανικής μάθησης. 26
Σχήμα 2.4: Αναπαράσταση αρχιτεκτονικής Συνελικτικού Νευρωνικού δικτύου. 27
Σχήμα 2.5: Παράδειγμα εκτέλεσης ML-Based Προσομοίωσης 27
Σχήμα 2.6: Παράλληλο Simulation. 28
Σχήμα 3.1: Στοιχεία βιολογικών και τεχνητών νευρωνικών δικτύων 30
Σχήμα 3.2 : Διάταξη Perceptron 31
Σχήμα 3.3: Διάταξη Perceptron πολλαπλών επιπέδων. 32
Σχήμα 3.4: Σιγμοειδής Συνάρτηση (Sigmoid Function) 33
Σχήμα 3.5: Υπερβολική Εφαπτομένη (Hyperbolic Tangent) 33
Σχήμα 3.6: Rectified Linear Unit (ReLu) 33
Σχήμα 3.7: Γράφημα Gradient Descent (Κόστος - Βάρη) 35
Σχήμα 3.8: Μικρό και μεγάλο Learning rate 35
Σχήμα 3.9: Παράδειγμα συνελικτικού νευρωνικού δικτύου 36
Σχήμα 3.10: Παράδειγμα επαναλαμβανόμενου νευρωνικού δικτύου 37
Σχήμα 3.11:Παράδειγμα δικτύου μακράς βραχυπρόθεσμης μνήμης. 37
Σχήμα 3.12:Παραλλαγή δικτύου μακράς βραχυπρόθεσμης μνήμης. 37
Σχήμα 3.13: Βασική Δομή Transformer 38
Σχήμα 5.1: Αρχιτεκτονική μοντέλου Ithemal. 46
Σχήμα 5.2: Heatmaps για τις μετρήσεις και τις προβλέψεις των throughput στα διάφορα μοντέλα. 49
Σχήμα 5.3: Αρχιτεκτονική DAG-RNN δικτύου για throughput estimation. 50
Σχήμα 5.4: Αρχιτεκτονική του Token RNN μοντέλου για throughput estimation. 51
Σχήμα 5.5: Σύγκριση μοντέλων νευρωνικού για throughput estimation. 51
Σχήμα 6.1: Διαδικασία αφαίρεσης Outliers. 54
Σχήμα 6.2: Ιστογράμματα σύγκρισης των δύο εργαλείων μέτρησης 56
Σχήμα 6.3: Εξέλιξη σφάλματος με το πέρασμα του χρόνου(εποχών). 61

Κατάλογος Πινάκων

Πίνακας 2.1: Παράδειγμα Throughput Estimation για x-86 ακολουθίες. 24
Πίνακας 2.2: Πίνακας αξιολόγησης απόδοσης για x86 ακολουθίες. 25
Πίνακας 4.1: Δεδομένα διαφορετικών κατηγοριών για τη σύνθεση του BHive Dataset. 41
Πίνακας 4.2: Περιγραφή κατηγοριών basic blocks. 42
Πίνακας 4.3: Παραδείγματα basic blocks από κάθε κατηγορία. 43
Πίνακας 4.4: Αποτελέσματα αξιολόγησης του Bhive dataset. 43
Πίνακας 5.1: Εφαρμογές dataset εκπαίδευσης Ithemal. 48
Πίνακας 5.2: Μέσο σφάλμα κάθε μοντέλου στις διάφορες αρχιτεκτονικές. 49
Πίνακας 5.3: Εκτίμηση απόδοσης για τους διάφορους estimators σε εντολές κάθε δευτερόλεπτο. 49
Πίνακας 6.1: Χαρακτηριστικά συστημάτων που χρησιμοποιήθηκαν. 55
Πίνακας 6.2: Απόκλιση μετρήσεων των δύο εργαλείων. 55
Πίνακας 6.3: Γεωμετρικός Μέσος σύγκρισης των δύο εργαλείων. 56
Πίνακας 6.4: Πίνακας απόκλισης μετρήσεων για τους διάφορους επεξεργαστές με τις timing-harness μετρήσεις. 57
Πίνακας 6.5: Πίνακας απόκλισης μετρήσεων για τους διάφορους επεξεργαστές με τις nanoBench μετρήσεις. 58
Πίνακας 6.6: Πίνακας γεωμετρικού μέσου μετρήσεων για τους διάφορους επεξεργαστές για τις timing-harness μετρήσεις. 58
Πίνακας 6.7: Πίνακας γεωμετρικού μέσου μετρήσεων για τους διάφορους επεξεργαστές με τις nanoBnech μετρήσεις. 58
Πίνακας 6.8: Πίνακας μετρικών για τα διάφορα μοντέλα στους διάφορους επεξεργαστές με τις timing-harness μετρήσεις. 60
Πίνακας 6.9: Πίνακας μετρικών για τα διάφορα μοντέλα στους διάφορους επεξεργαστές με τις nanoBench μετρήσεις. 60
Πίνακας 6.10: Πίνακας σφάλματος για τη μεταφορά εκπαιδευμένων μοντέλων με timing-harness μετρήσεις. 62
Πίνακας 6.11: Πίνακας σφάλματος για τη μεταφορά εκπαιδευμένων μοντέλων με nanoBench μετρήσεις. 63
Πίνακας 6.12: Πίνακας Spearman Correlation για τη μεταφορά εκπαιδευμένων μοντέλων με timing-harness μετρήσεις. 63
Πίνακας 6.13: Πίνακας Spearman Correlation για τη μεταφορά εκπαιδευμένων μοντέλων με nanoBench μετρήσεις. 63
Πίνακας 6.14: Πίνακας μετρικών για τους Transformers με timing-harness μετρήσεις. 64
Πίνακας 6.15: Πίνακας μετρικών για τους Transformers με nanoBench μετρήσεις. 65
Πίνακας 6.16: Πίνακας σφάλματος για μεταφορά Transformers σε άλλους επεξεργαστές με timing-harness μετρήσεις. 66
Πίνακας 6.17: Πίνακας σφάλματος για μεταφορά Transformers σε άλλους επεξεργαστές με nanoBench μετρήσεις. 66
Πίνακας 6.18: Πίνακας Spearman Correlation για μεταφορά Transformers σε άλλους επεξεργαστές με timing-harness μετρήσεις. 67
Πίνακας 6.19: Πίνακας Spearman Correlation για μεταφορά Transformers σε άλλους επεξεργαστές με nanoBench μετρήσεις. 67

Κεφάλαιο 1 - Εισαγωγή

Ο τομέας των νευρωνικών δικτύων τα τελευταία χρόνια αναπτύσσεται με πολύ γρήγορους ρυθμούς. Πρόκειται για μια ιδέα και έναν τομέα που αν και υπάρχει εδώ και πολλά χρόνια, έμεινε πρακτικά στάσιμος και προχώρησε τα τελευταία χρόνια λόγω της μεγάλης έξαρσης της ποσότητας των δεδομένων αλλά και των υπολογιστικών πόρων. Η μηχανική μάθηση έχει βρεί ευρεία χρήση σε πολλούς τομείς, όπως στην όραση υπολογιστών, ιατρική και γενικά όπου απαιτείται επεξεργασία και ανάλυση δεδομένων. Στη συγκεκριμένη εργασία εξετάζεται η χρήση των πλεονεκτημάτων των νευρωνικών δικτύων σε ένα μοντέλο με σκοπό την πρόβλεψη του αριθμού των κύκλων ρολογιού που απαιτείται από τον επεξεργαστή για την εκτέλεση ενός κομματιού κώδικα. Μέσω του μοντέλου αυτού δείχνονται οι διαφορές της επίδοσης μεταξύ των διαφόρων επεξεργαστών.

1.1 Η εκτίμηση της απόδοσης του κώδικα.

Ένα καίριο και πολύ βασικό θέμα τόσο για τους σχεδιαστές μεταγλωττιστών όσο και για τους προγραμματιστές και ερευνητές αποτελεί η πρόβλεψη του αριθμού των κύκλων ρολογιού που απαιτεί η εκτέλεση ενός κομματιού κώδικα (throughput). Ο μεταγλωττιστής μετατρέπει ουσιαστικά έναν κώδικα από γλώσσα υψηλού επιπέδου σε γλώσσα χαμηλότερου επιπέδου (γλώσσα μηχανής). Για κάθε γλώσσα απαιτείται κι ένας διαφορετικός μεταγλωττιστής. Για την υλοποίηση ενός τέτοιου λογισμικού είναι απαραίτητη η αξιόπιστη εκτίμηση της απόδοσης για την καλύτερη υλοποίηση κάθε τμήματος εντολών. Με τον όρο απόδοση (throughput) παριστάνουμε το πόσο γρήγορα οι εντολές στις οποίες αναφερόμαστε μπορούν να επεξεργαστούν τα δεδομένα. Γνωρίζοντας αυτό το μέγεθος είναι εφικτή η πρόβλεψη και η εκτίμηση της απόδοσης του συστήματος (runtime performance). Για παράδειγμα, η κατανομή καταχωρητών (register allocation) και η δρομολόγηση εντολών (instruction scheduling) βασίζονται σε μια ακριβή εκτίμηση της απόδοσης. Η μέτρηση αυτή βρίσκει επίσης εφαρμογή σε θέματα γενετικών αλγορίθμων που πάλι χρειάζονται γνώσεις πάνω στην απόδοση του συστήματος αλλά και στη χρονοδρομολόγηση εντολών (για τον τομέα της ενισχυτικής μάθησης). Η εναλλακτική, η μέτρηση δηλαδή εκτελώντας τον κώδικα κάθε φορά, αποτελεί μια υπολογιστικά πολύ κοστοβόρα λύση και προφανώς δεν αποτελεί μια δημοφιλή προσέγγιση. Τόσο η Intel όσο και άλλες εταιρίες έχουν αναπτύξει μοντέλα για την πρόβλεψη αυτή ώστε να αποφύγουν την πλήρη εκτέλεση του κώδικα (IACA, llvm-mca, OSACA). Τα εργαλεία που κάνουν αυτή την πρόβλεψη θα πρέπει να είναι αξιόπιστα και να κάνουν γρήγορα τις προβλέψεις. Επίσης θα πρέπει να υπάρχει η δυνατότητα να χρησιμοποιηθούν τα εργαλεία αυτά σε όλους τους επεξεργαστές ανεξάρτητα από την τεχνολογία και την αρχιτεκτονική τους.

1.2 Αντικείμενο διπλωματικής

Βασική ενασχόληση της συγκεκριμένης εργασίας είναι η παρουσίαση και η σύγκριση των διαφόρων συστημάτων (με διαφορετικούς επεξεργαστές) μέσω της πειραματικής αξιολόγησης εργαλείων που κάνουν πρόβλεψη - εκτίμηση των κύκλων ρολογιού που απαιτεί για να εκτελεστεί ένα συγκεκριμένο τμήμα κώδικα με χρήση μεθόδων μηχανικής μάθησης. Η

πρόβλεψη αυτή δεν αφορά σε πλήρη κώδικα αλλά σε ένα τμήμα του κάθε φορά (σε επίπεδο basic block). Παρά την ύπαρξη ορισμένων παραδοσιακών εργαλείων που επιτελούσαν αυτή την πρόβλεψη, τα τελευταία χρόνια παρουσιάζονται νέα και καινοτόμα εργαλεία. Τα εργαλεία αυτά δεν είναι μόνο αναλυτικά αλλά και ορισμένα που βασίζονται σε μεθόδους τεχνητής νοημοσύνης και έχουν στο επίκεντρο την μάθηση του μοντέλου που αναπτύσσεται ώστε να γίνει η ζητούμενη πρόβλεψη της απόδοσης. Συγκεκριμένα μελετήθηκε το μοντέλο Ithemal το οποίο αρχικά είχε αναπτυχθεί για συγκεκριμένες γενιές επεξεργαστών της Intel. Δοκιμάστηκε σε ένα ευρύ φάσμα επεξεργαστών με ειδικά προσαρμοσμένο σύνολο δεδομένων ώστε να είναι πιο άρτια και δίκαιη η σύγκριση του με άλλα ανταγωνιστικά εργαλεία. Επίσης, έγινε προσπάθεια για δοκιμή και χρήση του μοντέλου σε επεξεργαστές AMD καθώς και προσπάθεια αντικατάστασης του βασικού νευρωνικού του Ithemal με ένα νέο καινοτόμο και πολύ δημοφιλές νευρωνικό, τον Transformer. Και στις δύο περιπτώσεις (στο Ithemal και στον Transformer) έγινε προσπάθεια αξιολόγησης της επίδοσης των διάφορων επεξεργαστών με μοντέλα εκπαιδευμένα σε άλλους επεξεργαστές ώστε να φανεί η συμπεριφορά τους.

1.3 Διάρθρωση της Διπλωματικής

Ξεκινώντας, στο κεφάλαιο 2 παρουσιάζονται τόσο το βασικό μοντέλο με το οποίο ασχολείται η διπλωματική. δηλαδή το Ithemal, όσο και παραδοσιακά αλλά και σύγχρονα εργαλεία - μοντέλα με τα οποία αργότερα θα συγκριθεί το βασικό μοντέλο. Στο κεφάλαιο 3 αναλύονται τα θέματα των νευρωνικών δικτύων και συγκεκριμένα τόσο τα συνήθη χρησιμοποιούμενα νευρωνικά όσο και αυτά που περιέχονται στο μοντέλο του Ithemal. Γίνεται επίσης ανάλυση ενός σύγχρονου νευρωνικού, του Transformer το οποίο δοκιμάστηκε και στην πράξη. Στο κεφάλαιο 4 αναλύεται το σύνολο των δεδομένων που χρησιμοποιείται για την αξιολόγηση του μοντέλου μαζί με τους λόγους για τους οποίους επιλέχθηκε. Στα κεφάλαια 5 και 6 γίνεται ανάλυση του μοντέλου και των πειραμάτων καθώς και αξιολόγηση των αποτελεσμάτων που προέκυψαν. Τα πειράματα και οι δοκιμές που έγιναν αρχικά αφορούσαν τη μελέτη της επίδοσης του Ithemal στους διάφορους επεξεργαστές καθώς και τη δυνατότητα μεταφοράς του χωρίς επανάληψη της εκπαίδευσης ώστε να φανεί η συμπεριφορά των διάφορων επεξεργαστών. Η ίδια διαδικασία ακολουθήθηκε και με τον Transformer. Στο τελευταίο κεφάλαιο γίνεται ανακεφαλαίωση και παρουσιάζονται πιθανές μελλοντικές προεκτάσεις.

Κεφάλαιο 2 - Παρουσίαση τεχνικών και εργαλείων εκτίμησης της απόδοσης ενός τμήματος κώδικα.

2.1 Κλασικές τεχνικές εκτίμησης απόδοσης κώδικα

2.1.1 IACA

Ένα από τα βασικά εργαλεία που χρησιμοποιήθηκε για την ανάλυση εξαρτήσεων του κώδικα, του κόστους και των καθυστερήσεων του είναι το IACA. Πρόκειται για ένα εργαλείο που αναπτύχθηκε από την Intel.
Ο αναλυτής αυτός προσφέρει δυνατότητα εκτίμησης της απόδοσης σε διαφορετικές Intel 64-bit μικροαρχιτεκτονικές με χρήση σε Windows, Linux και Mac OS. Μέσω των διαφορετικών εκδόσεων του εργαλείου δόθηκε η δυνατότητα χρήσης του από τις περισσότερες αρχιτεκτονικές της Intel (Ivy Bridge, Haswell, Broadwell, Skylake). Το αποτέλεσμά του είναι σε ASCII μορφή και η χρήση του γίνεται μέσω του τερματικού του υπολογιστή (terminal).
Δεν απαιτεί το εκάστοτε υλικό (hardware) για να κάνει την πρόβλεψη αφού κάνει στατική ανάλυση και δεν τρέχει ουσιαστικά τον κώδικα. Μαζί με τον κώδικα που δίνεται σαν παράμετρος στο εργαλείο δίνεται σαν είσοδος και ο επεξεργαστής για τον οποίο θα γίνει η πρόβλεψη. Στο δυαδικό αρχείο που περιέχει τον κώδικα περιέχονται και οι θέσεις μνήμης που θα εστιάσει ο επεξεργαστής. Περιορισμός του εργαλείου αποτελεί το ότι δεν υποστηρίζει ορισμένες εντολες που αγνοούνται κατά την ανάλυση. Το 2019 η Intel σταμάτησε την χρήση του αναζητώντας νέες τεχνικές και εργαλεία για την συγκεκριμένη πρόβλεψη της απόδοσης.[1]

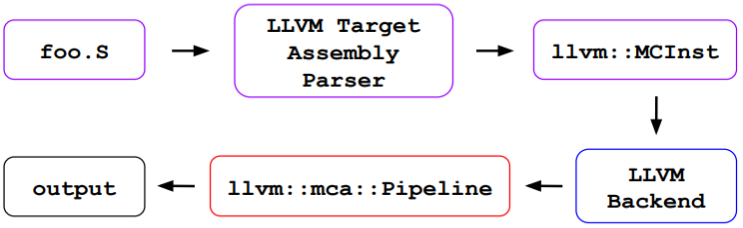
2.1.1.1 Λειτουργία και modes λειτουργίας

Με είσοδο έναν κώδικα σε δυαδική μορφή, αρχικά δημιουργεί μια στατική ανάλυση του kernel throughput καθώς και των καθυστερήσεων που προκύπτουν υπό ιδανικές συνθήκες. Επίσης βρίσκει τη διασύνδεση της κάθε εντολής με τις σχετικά πύλες (ports) αλλά και την κρίσιμη διαδρομή του κώδικα (critical path). Το κυρίως μέρος του κώδικα αντιμετωπίζεται ως ένας ατέρμονας βρόχος από το οποίο προκύπτει η απόδοση καθώς και τα “κολλήματα” (bottlenecks) του κώδικα.
Η αναφορά που προκύπτει από το Throughput Analysis περιέχει την απόδοση σε αριθμό κύκλων καθώς και το πλήθος των μικροεντολών για κάθε εντολή. Επίσης, περιέχει το μέσο πλήθος κύκλων για κάθε port του επεξεργαστή και για κάθε επανάληψη του βρόχου. Τέλος έχει μια ένδειξη σχετικά με το αν βρισκόμαστε στο κρίσιμο μονοπάτι και το disassembling για κάθε εντολή.
Το IACA μπορεί να χρησιμοποιηθεί με τους εξής τρεις τρόπους:

- Throughput Mode που βρίσκει τους μέγιστους κύκλους ρολογιού (απόδοση).
- Latency Mode για την εύρεση της ελάχιστης καθυστέρησης.
- Trace Mode που παρουσιάζει την πρόοδο της διαδικασίας μέσω του pipeline. [1]

2.1.2 LLVM-MCA

Το συγκεκριμένο αναλυτικό εργαλείο δημιουργήθηκε αρχικά από την εταιρεία Sony για να να γίνεται εύρεση λαθών σε δικά τους εσωτερικά προγραμματιστικά μοντέλα. Πρόκειται για έναν compiler που βασίστηκε στο εργαλείο IACA της Intel. Χρησιμοποιούνταν για την μέτρηση της απόδοσης συγκεκριμένων κομματιών κώδικα. Βοηθούσε στην εύρεση κολλημάτων (bottlenecks) του κώδικα και πιθανές καθυστερήσεις, οπότε πρόκειται για ένα ανταγωνιστικό εργαλείο με αυτό της Intel που αναφέρθηκε παραπάνω.



Σχήμα 2.1 : Βασική λειτουργία llvm-mca

Τα αποτελέσματα προκύπτουν από επαναληπτική εκτέλεση του εκάστοτε κώδικα (100 επαναλήψεις default) ώστε να προκύψει μια αναφορά με την απόδοση του εκάστοτε κώδικα. Χρησιμοποιώντας το συγκεκριμένο εργαλείο βγαίνουν ορισμένα συμπεράσματα, τόσο θετικά όσο και αρνητικά. Αρχικά η βασική μονάδα λειτουργίας είναι αρκετά απλοϊκή και δεν χρησιμοποιεί πληροφορίες για την ιεραρχία των κρυφών (cache) μνημών. Η προσομοίωση πολλές φορές δεν είναι ρεαλιστική αφού δεν είναι το ίδιο με μια πραγματική εκτέλεση, καθώς επίσης δεν λαμβάνει υπόψη το front-end. Ακόμα πρόκειται για ένα πρόσφατο εργαλείο το οποίο δεν έχει τα απαραίτητα documentations, tests και benchmarks ώστε να είναι σωστή η σύγκριση με το IACA. Επίσης κάποιες στατιστικές και απόψεις δεν είναι καθαρές στους developers. [2],[3],[4],[5],[6],[7]
Αξιοσημείωτο είναι ότι υπάρχει μια διαδικτυακή εκδοχή του εργαλείου.[8]

2.1.2.1 Βήματα - Στάδια

Αρχικά έχουμε το Dispatch stage όπου δεσμεύονται όλοι οι απαιτούμενοι πόροι (resources) για κάθε μια εντολή. Ακολουθεί το Issue stage, όπου μπλοκάρεται η εντολή μέχρι οι τελεστές και οι πύλες να είναι όλες διαθέσιμες. Έπειτα, το Execute stage δεσμεύει όλες τις απαραίτητες πύλες για όσο χρόνο ορίζεται από το mapping. Τέλος, στο Retire stage απελευθερώνονται όλοι οι δεσμευμένοι πόροι κάθε εντολής.

2.1.3 Open Source Architecture Code Analyzer (OSACA)

Το συγκεκριμένο εργαλείο δίνει τη δυνατότητα στους χρήστες να έχουν πρόσβαση στον κώδικα (open source) σε αντίθεση με το IACA. Μπορεί να χρησιμοποιηθεί εκτός από Intel επεξεργαστές και σε AMD αλλά και ARM επεξεργαστές με την κατάλληλη παραμετροποίηση στην είσοδο από την γραμμή εντολών. Πρόκειται για ένα εργαλείο βασισμένο στην Python (python module) που και πάλι προβλέπει την απόδοση ενός τμήματος κώδικα. Ο συγκεκριμένος μεταγλωττιστής μπορεί να χρησιμοποιηθεί χωρίς να είναι απαραίτητη η εγκατάσταση του αφού υπάρχει και μια διαδικτυακή εκδοχή του. Στην είσοδο δίνεται το αρχείο κώδικα και ένα yaml αρχείο με πληροφορίες για τον επεξεργαστή. Θεωρεί ένα

χρονοδιάγραμμα για τις εντολές και ότι ο επεξεργαστής μπορεί να το εκτελέσει ώστε να βρει τη βέλτιστη απόδοση.

2.2 Παρουσίαση Σύγχρονων Εργαλείων Εκτίμησης Απόδοσης

Τα τελευταία χρόνια έχουν αναπτυχθεί πολλά εργαλεία - μοντέλα για την πρόβλεψη της απόδοσης. Ορισμένα είναι αναλυτικά εργαλεία, όπως το uiCA με εξαιρετικά καλά αποτελέσματα καθώς και άλλα με εστίαση στα δεδομένα και τις ιδιότητές τους. Μάλιστα ορισμένα από τα νέα αυτά εργαλεία χρησιμοποιούν τεχνικές μηχανικής μάθησης ώστε να βελτιώσουν τα αποτελέσματά τους.

2.2.1 uiCA

Πρόκειται για ένα αναλυτικό μοντέλο που μπορεί να κάνει πρόβλεψη της απόδοσης ενός τμήματος κώδικα. Το εργαλείο αυτό ουσιαστικά προσομοιώνει την επαναληπτική εκτέλεση του κώδικα (τουλάχιστον 10 επαναλήψεις και 500 κύκλους) μέχρι να βγάλει αποτελέσματα για την απόδοση. Μπορεί να χρησιμοποιηθεί σε όλους τους Intel επεξεργαστές που βγήκαν από το 2011 έως το 2021. Μια από τις βασικές παραδοχές που γίνεται στην συγκεκριμένη προσέγγιση είναι ότι θα έχουμε συνεχώς cache hits κατά τη χρήση δεδομένων από την μνήμη. Η αξιολόγηση του συγκεκριμένου μοντέλου γίνεται με χρήση του συνόλου δεδομένων BHive (μια τροποποιημένη έκδοση).[11],[12]

2.2.1.1 Πλεονεκτήματα και παραδοχές του μοντέλου

Συχνά παρατηρούνται ορισμένες ασυνέπειες μεταξύ των προβλέψεων και των πραγματικών μετρήσεων λόγω ελλιπούς ανάλυσης των προηγούμενων εργαλείων και ορισμένων μη κατάλληλα διαμορφωμένων benchmarks. Εδώ είναι πολύ καλύτερα ορισμένο το pipeline model. Το θέμα με κάποια μη ορισμένα χαρακτηριστικά, κάτι το οποίο λύθηκε με τεχνικές reverse engineering. Με σκοπό να είναι πιο σωστή και δίκαιη η σύγκριση έγιναν κάποιες αλλαγές στα benchmarks του συνόλου δεδομένων.
Οι παραδοχές που έγιναν κατά την ανάπτυξη του συγκεκριμένου μοντέλου είναι οι παρακάτω. Αρχικά, δεν έχουμε TLB και cache misses αλλά ούτε κολλήματα (conflicts) κατά την εκτέλεση του κώδικα. Επίσης, γίνεται ταίριασμα όλων των απαραίτητων αποθηκεύσεων. Ακόμα, γίνεται η υπόθεση ότι δεν υπάρχουν λάθος προβλέψεις σε διακλαδώσεις του κώδικα και δεν υπάρχει memory aliasing. Τέλος, ο κώδικας είναι γενικότερα καλά ορισμένος και άρα δεν υπάρχουν εξαιρέσεις ή διακοπές κατά την εκτέλεση. [11],[12]

2.2.1.2 Αρχιτεκτονική των Intel επεξεργαστών

Η αρχιτεκτονική του uiCA βασίστηκε στο pipeline των Intel επεξεργαστών. Συνεπώς απαιτεί αναλυτική γνώση της αρχιτεκτονικής του εκάστοτε επεξεργαστή, γεγονός που δεν βοηθά στην μεταφερσιμότητά του. Ξεκινώντας από το front end, αρχικά, υπάρχει ένας προαποκωδικοποιητής (predecoder) ο οποίος φέρει από την προσωρινή μνήμη εντολών κάθε μια από τις εντολές και εντοπίζει το σημείο που αρχίζει κάθε εντολή. Οι εντολές που έχουν περάσει από το συγκεκριμένο στάδιο μπαίνουν σε μια άλλη δομή που ονομάζεται ουρά εντολών (instruction queue). Σε έναν κύκλο μπορεί να γίνει αποκωδικοποίηση 5 εντολών (αν είχα 6 εντολές στον έναν κύκλο θα γίνοντας η αποκωδικοποίηση των 5 εντολών και στον επόμενο κύκλο της 6ης εντολής). Υπάρχει ακόμα μια μονάδα κωδικοποίησης (decoding) η οποία μπορεί να φέρει (fetching) μέχρι 4 εντολές ανά κύκλο από την ουρά των εντολών και

μετά την ολοκλήρωση της κωδικοποίησης τις στέλνει σε μια νέα ουρά (instruction decode queue). Η μονάδα κωδικοποίησης αποτελείται από έναν σύνθετο κωδικοποιητή που κωδικοποιεί εντολές με μια μικροεντολή καθώς και 3 απλούς κωδικοποιητές οι οποίοι διαλέγουν την πρώτη από τις εντολές που έχουν έρθει και κωδικοποιεί εντολές αποτελούμενες από μέχρι και 4 μικροεντολές. Αν υπάρχει κάποια εντολή που αποτελείται από πάνω από 4 μικροεντολές τότε αυτή οδηγείται στον Microcode Sequencer (MS).

Περνώντας τώρα στο back end κομμάτι, υπάρχει ένας μετονομαστή (renamer) που κάνει αντιστοίχιση των architectural σε physical registers. Εντολές έρχονται από την ουρά που έχουν τοποθετηθεί μετά τη διαδικασία της κωδικοποίησης (instruction decode queue). Αυτές οι εντολές αποθηκεύονται σε έναν reorder buffer και οργανώνονται από έναν χρονοσχεδιαστή (scheduler). Ο τρόπος που ο μετονομαστής (renamer) διαλέγει την πύλη που θα ανατεθεί η κάθε εντολή δεν είναι γνωστός αφού έχει υλοποιηθεί με τεχνικές reverse engineering. Σχετικά με τον χρονοσχεδιαστή (scheduler) είναι αυτός που παρακολουθεί τις εξαρτήσεις των μικροεντολών. Όταν οι διάφοροι τελεστές είναι έτοιμοι τότε στέλνονται στην κατάλληλη πύλη, όπου κάθε ένα από αυτά είναι συνδεδεμένα με μια μονάδα επεξεργασίας (functional unit). [11],[12]

2.2.1.3 Δεδομένα και τρόποι αξιολόγησης

Για την αξιολόγηση του μοντέλου και τη σύγκρισή του με τους άλλους προβλέπτες χρησιμοποιούμε τα BHive benchmarks. Συγκεκριμένα χρησιμοποιήθηκε μια τροποποιημένη έκδοση του συνόλου δεδομένων BHive όπου χωρίζονται τα τμήματα κώδικα που περιέχουν διακλάδωση από αυτά που δεν περιέχουν. Οι δημιουργοί του uiCA διαπίστωσαν ότι ενώ ήθελαν να τρέχουν επαναληπτικά τις εντολές έπρεπε να εισάγουν μια διαφοροποίηση ανάλογα με την ύπαρξη ή όχι εντολών διακλάδωσης. Πιο συγκεκριμένα, όταν υπάρχει τέτοια εντολή, δεδομένου ότι η διακλάδωση είναι πάντα taken και στέλνει την ροή του κώδικα πάλι στην αρχή οδηγεί στην δημιουργία ενός ατέρμονου βρόχου. Άρα για πιο αποτελεσματική αξιολόγηση όταν δεν υπάρχει εντολή διακλάδωσης κάνουν υπολογισμό του TP_U που εκτελεί τον κώδικα επαναλαμβάνοντας τον συγκεκριμένες φορές. Αντίθετα όταν υπάρχει εντολή διακλάδωσης γίνεται υπολογισμός του TP_L.

Άρα, σχετικά με τις μετρικές αξιολόγησης του μοντέλου, χρησιμοποιούνται το μέσο σφάλμα (MAPE) και το Kendall's coefficient. Η πρόβλεψη της απόδοσης για τις δύο μορφές του BHive φαίνεται παρακάτω. [11],[12]

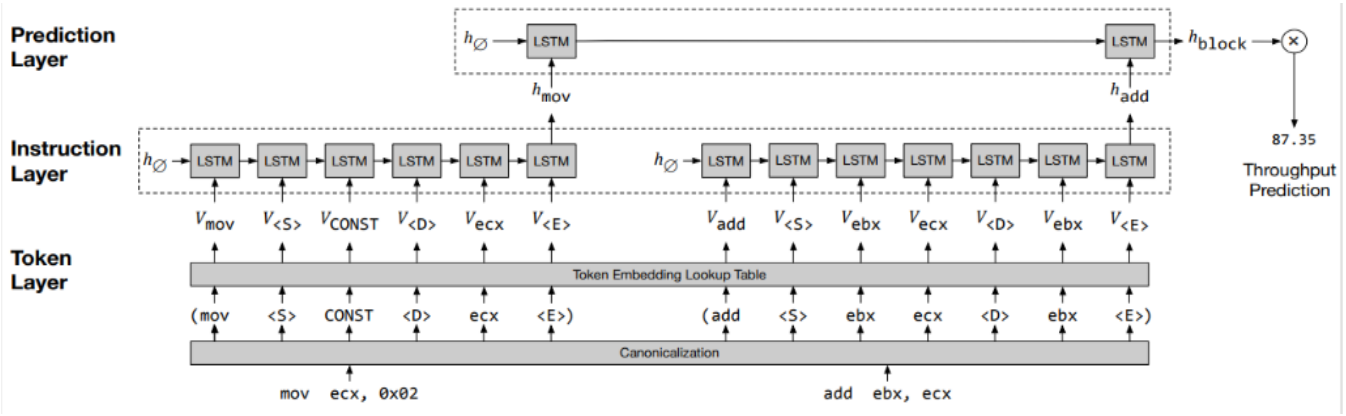
$$TP_{baseline,U} = max(\frac{n}{4}, \frac{mr}{2}, \frac{mw}{w}) \qquad TP_{baseline,L} = max(1, \frac{n-1}{i}, \frac{mr}{2}, \frac{mw}{w})$$

2.2.2 Ithemal

Το Ithemal αποτελεί το πρώτο εργαλείο που μαθαίνει να προβλέπει την απόδοση ενός συνόλου εντολών. Η νέα αυτή προσέγγιση που χρησιμοποιεί μια ιεραρχική LSTM μέθοδο είναι πολύ αποτελεσματική δίνοντας έως και το μισό σφάλμα από τα συνήθη μοντέλα.
Η σημασία του εργαλείου έγκειται στην αξία της πρόβλεψης των κύκλων που θα χρειαστεί ένα κομμάτι κώδικα τόσο από τους σχεδιαστές όσο και από τους προγραμματιστές. Τα μοντέλα που χρησιμοποιούνται πρέπει να συνδυάζουν ταχύτητα, προσαρμοστικότητα σε διάφορες αρχιτεκτονικές καθώς και μεγάλη ακρίβεια αφού αρχικά κάνουν μετατροπή σε μικροεντολές και μετά βελτιστοποίηση ώστε να επιτευχθεί η μέγιστη παραλληλία. Με τον τρόπο αυτό θα αντιμετωπιστούν τα μειονεκτήματα των κλασσικών εργαλείων.[9],[10]

2.2.2.1 Αρχιτεκτονική του Ithemal

Το συγκεκριμένο μοντέλο αποτελείται από 3 στάδια - επίπεδα. Στο πρώτο στάδιο έχουμε την κανονικοποίηση των δεδομένων όπου δημιουργούνται τα tokens. Στο δεύτερο στάδιο γίνεται η επεξεργασία των tokens και η πληροφορία ουσιαστικά οδηγείται στο νευρωνικό μέρος του μοντέλου. Το τρίτο στάδιο είναι η εφαρμογή ενός διανύσματος βαρών στην είσοδο και η χρήση ενός bias για να προκύψει η τελική πρόβλεψη.[9],[10]



Σχήμα 2.2: Αρχιτεκτονική του Ithemal

2.2.2.2 Αποτελέσματα του Ithemal

Στον παρακάτω πίνακα βλέπουμε ορισμένα χαρακτηριστικά παραδείγματα προβλέψεων κύκλων για ορισμένα τμήματα κώδικα. Παρουσιάζεται η σύγκριση των παραδοσιακών εργαλείων με το Ithemal, του οποίου η έξοδος είναι πιο κοντά στην τιμή από την εκτέλεση του κάθε κώδικα.

	vxorps xmm0, xmm0,xmm0	mov [rbp+0x70], rax mov rax, 0x01	shl rbx, 0x02 mov rdi, rbx
Actual	**32**	**103**	**83**
llvm - mca	100	100	50
IACA	24	84	96
Ithemal	**35**	**102**	**83**

Πίνακας 2.1: Παράδειγμα Throughput Estimation για x-86 ακολουθίες.

Στον παρακάτω πίνακα φαίνονται αναλυτικά τα αποτελέσματα από την αξιολόγηση του Ithemal στις τρεις αρχιτεκτονικές που αρχικά δοκιμάστηκαν (Ivy Bridge, Haswell, Skylake). Σε όλες τις περιπτώσεις το Ithemal παρουσιάζει αξιόπιστα μοντέλα και μετρικές που επιβεβαιώνουν τα καλά αποτελέσματα, συγκριτικά με τα παραδοσιακά εργαλεία πρόβλεψης. Παρότι το uiCA έχει ακόμα καλύτερα αποτελέσματα στην πρόβλεψη της απόδοσης μπορεί να χρησιμοποιηθεί μόνο από Intel επεξεργαστές. Επίσης, απαιτούν πολλές πληροφορίες σχετικά με την αρχιτεκτονική και το pipeline του εκάστοτε επεξεργαστή ώστε να υπάρχει

συμβατότητα, γεγονός που περιορίζει πολύ τους χρήστες.

μArch	Predictor	BHiveU	BHiveL
IVY	**uiCA** **Ithemal** IACA llvm-mca	**1.51%** **7.08%** 13.94% 22.79%	**1.12%** - 11.54% 20.76%
HSW	**uiCA** **Ithemal** IACA llvm-mca	**0.76%** **7.38%** 15.04% 20.29%	**0.59%** - 12.00% 18.97%
BDW	**uiCA** **Ithemal** IACA llvm-mca	**1.08%** - 14.69% 14.23%	**0.61%** - 11.47% 16.71%
SKL	**uiCA** **Ithemal** IACA llvm-mca	**0.45%** **8.28%** 13.49% 15.61%	**0.38%** - 13.66% 12.01

Πίνακας 2.2: Πίνακας αξιολόγησης απόδοσης για x86 ακολουθίες.

2.2.3 DiffTune

Το συγκεκριμένο εργαλείο έχει να κάνει με τη βελτιστοποίηση των παραμέτρων ενός προσομοιωτή με χρήση ενός αντικαταστάτη (surrogate). Γενικά η διαδικασία ξεκινά με την συλλογή μετρήσεων για κάθε παράμετρο στην πραγματική μηχανή. Όταν οι μετρήσεις του προσομοιωτή και της πραγματικής μηχανής συμπέσουν τότε έχουν βρεθεί οι ζητούμενες - σωστές τιμές. Επίσης μπορεί να γίνει κατά προσέγγιση μέτρηση των παραμέτρων και με ένα σύνολο από benchmarks και να βρεθούν οι τιμές που ελαχιστοποιούν το σφάλμα. [13],[14]

2.2.3.1 Προσέγγιση DiffTune και αποτελέσματα

Στην προσέγγιση του DiffTune, η είσοδος αποτελείται από ένα πρόγραμμα, μια περιγραφή των παραμέτρων και ένα σύνολο δεδομένων ενώ σαν αποτέλεσμα εξάγει τις παραμέτρους που ελαχιστοποιούν το σφάλμα του προγράμματος. Οι συγκεκριμένες παράμετροι προκύπτουν από έναν αντικαταστάτη (surrogate) που παράγει τις ζητούμενες παραμέτρους. Για να επιτευχθεί η εύρεση του ελάχιστου σφάλματος βρίσκουμε το gradient του αντικαταστάτη. Αυτός ο αντικαταστάτης προσομοιώνεται με ένα νευρωνικό δίκτυο που μιμείται τη συμπεριφορά του προσομοιωτή. Το νευρωνικό αυτό βασίζεται στο Ithemal. [13],[14]

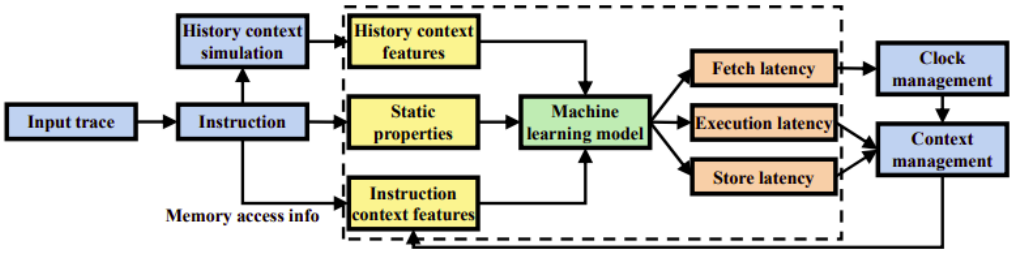
2.2.3.2 Αλγόριθμος DiffTune

Έστω πρόγραμμα f με παραμέτρους θ, είσοδο X και έξοδο Y και μια συνάρτηση f* που βρίσκει τις παραμέτρους που ελαχιστοποιούν το σφάλμα. Στο πρώτο στάδιο (Train Surrogate) γίνεται η εκτίμηση της f-hat που θα ελαχιστοποιήσει το σφάλμα. Στο δεύτερο

στάδιο (Train Parameters) γίνεται η εκτίμηση των παραμέτρων ώστε να ελαχιστοποιηθεί το σφάλμα μεταξύ των f* και f-hat. Τέλος, στο τρίτο στάδιο (Extract Parameters) βρίσκουμε τις παραμέτρους και τις βάζουμε στο αρχικό στο πρόγραμμα f. [13],[14]

2.2.4 SimNet

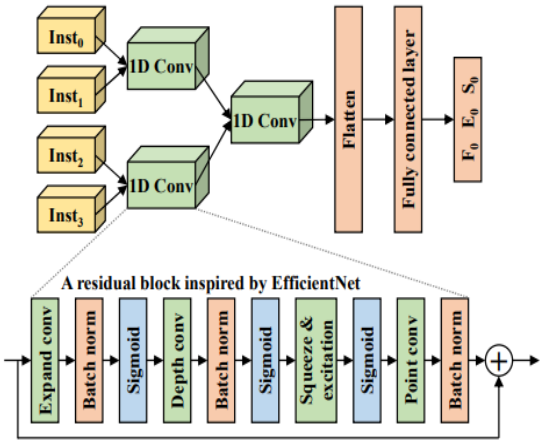
Πρόκειται για μια προσέγγιση που κάνει χρήση της μηχανικής μάθησης με σκοπό τη διευκόλυνση της προσομοίωσης σε επίπεδο μικροαρχιτεκτονικής.
Οι DES Simulators (Discrete-event simulation) που χρησιμοποιούνται εδώ οδηγούν σε νέες αρχιτεκτονικές ιδέες βρίσκοντας αρκετές εφαρμογές όπως εξερεύνησης του διαστήματος. Η εκτέλεση μιας προσομοίωσης απαιτεί περισσότερο χρόνο από την πραγματική εκτέλεση του προγράμματος. Μια λύση στο πρόβλημα αυτό θα ήταν η παραλληλοποίηση, όμως η δομή, οι αλληλεπιδράσεις και η μη αναμενόμενη συμπεριφορά ορισμένων τμημάτων δεν το επιτρέπει. Έτσι η λύση βασίστηκε σε βαθιά νευρωνικά δίκτυα τα οποία μπορούν να βοηθήσουν σε υπολογισμούς καθυστερήσεων που είναι απαραίτητοι για την προσομοίωση.
Οι βασικοί παράγοντες που καθορίζουν το Instruction Latency είναι οι εξής:
Τα Static Instruction Properties που αφορούν τα βασικά στοιχεία μιας εντολής όπως είναι το είδος της εντολής αλλά και οι καταχωρητές που θα καθορίσουν τις μονάδες επεξεργασίας που θα χρειαστούν καθώς και τα θέματα μνήμης.
Τα Dynamic Processor States καθώς η καθυστέρηση μιας εντολής εξαρτάται από την κατάσταση των components και τον χρόνο εκτέλεσης (execution time). Στην προσέγγιση αυτή υπάρχει ένας προβλέπτης καθυστέρησης που βασίζεται με μηχανική μάθηση που είναι το κέντρο του framework και βρίσκει την επιρροή των χαρακτηριστικών εισόδου. [15],[16]



Σχήμα 2.3: Ροή εργασιών στο μοντέλο της μηχανικής μάθησης.

2.2.4.1 Αρχιτεκτονική Νευρωνικού δικτύου

Μια προσέγγιση αποτελούν τα Sequence Oriented Models όπου σαν είσοδο έχουν μια ακολουθία από εντολές παρόμοια με την φυσική γλώσσα. Εκεί ουσιαστικά βασίζεται η δράση του Ithemal όπου χρησιμοποιεί ένα LSTM για την πρόβλεψη.
Μια διαφορετική προσέγγιση είναι τα Deep CNN Models. Έχουν ευρεία χρήση στην όραση υπολογιστών αλλά βρίσκουν εφαρμογή και στην συγκεκριμένη περίπτωση, δηλαδή σε θέματα καθυστέρησης εντολών. Τα συγκεκριμένα μοντέλα έχουν λιγότερες υπολογιστικές ανάγκες και είναι πλήρως συνδεδεμένα. Ένα ακόμα πλεονέκτημα είναι ότι η εκπαίδευση είναι πιο εύκολη και γρήγορη αφού απαιτούνται λιγότεροι παράμετροι ακόμα και για βαθύτερα δίκτυα. Παρακάτω φαίνεται η αναπαράσταση ενός τέτοιου συνελικτικού δικτύου.



Σχήμα 2.4: Αναπαράσταση αρχιτεκτονικής Συνελικτικού Νευρωνικού δικτύου.

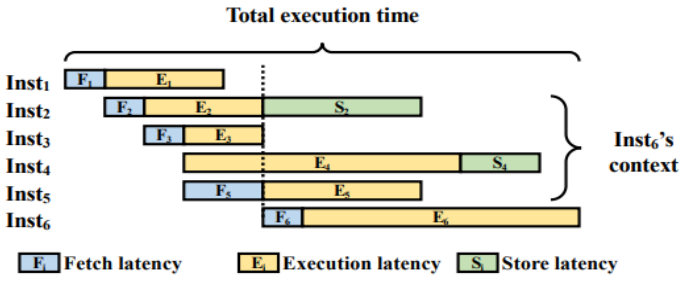
Η εντολή για την οποία θα γίνει πρόβλεψη Inst0 και θα έχει fetch, execution και store εξόδους. Οι είσοδοι οργανώνονται σε μονοδιάστατους πίνακες και υπάρχουν για κάθε εντολή. Αυτή η οργάνωση βοηθά στην εύρεση των σχέσεων μεταξύ των εντολών. Στο σχήμα 5 φαίνεται η σύνδεση των εντολών Inst0 με την Inst1 και αντίστοιχα των Inst2 με την Inst3. Η έξοδος από το τελευταίο συνελικτικό δίκτυο διαπλατύνεται (Flatten layer) και πάει στο επόμενο επίπεδο. Στο τέλος του μοντέλου προκύπτουν οι καθυστερήσεις από τα fetch, execution και store στάδια. Σαν συνάρτηση ενεργοποίησης χρησιμοποιείται η ReLu. [15],[16]

2.2.4.2 Αξιολόγηση Μοντέλου SimNet και σύγκριση με το Ithemal

Γενικά στα CNN το σφάλμα αυξάνει με βάση το πλήθος των επιπέδων του δικτύου. Στη συγκεκριμένη περίπτωση το σφάλμα ορίζεται ως εξής:

$$E = \frac{|f_\theta(x_i) - y_i|}{y_i + 1}$$

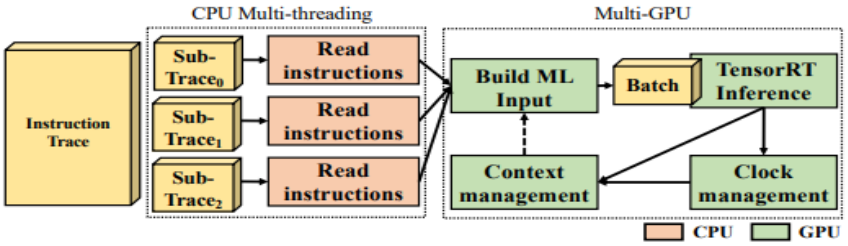
Συνολικά το SimNet επιτυγχάνει καλύτερα throughputs σε σχέση με άλλα εργαλεία και μάλιστα με καλύτερη ενεργειακή απόδοση (power efficiency). Συγκρίνοντας τώρα το Ithemal με το SimNet, το πρώτο χρησιμοποιεί σταθερό αριθμό εντολών σαν είσοδο ενώ το SimNet διαλέγει ανάλογα με το ποιές είναι ενεργές στον επεξεργαστή.
Ένα παράδειγμα της διαδικασίας του Simulation φαίνεται στο παρακάτω σχήμα. [15],[16]



Σχήμα 2.5: Παράδειγμα εκτέλεσης ML-Based Προσομοίωσης

2.2.4.3 Παράλληλο Simulation

Μια άλλη ιδέα είναι να σπάσει η εντολή σε μικρότερες εντολές και να γίνει η προσομοίωσή τους παράλληλα. Το πρόβλημα που προκύπτει στην συγκεκριμένη περίπτωση είναι ότι προκύπτουν επιπλέον σφάλματα κατά την προσομοίωση λόγω ανακριβειών του κώδικα. Από το σχήμα 7 φαίνεται ότι η CPU σπάει την κάθε εντολή και έπειτα τη στέλνει στην GPU που κάνει την υπόλοιπη δουλειά. Σχετικά με τη GPU κάθε νήμα έχει τις δικές του ουρές και μετρητή για το πλήθος των κύκλων. Μετά το μοντέλο μηχανικής μάθησης τρέχει για κάθε εντολή και στο τέλος συνδυάζονται σε μια συνολική είσοδο που πάει στο επόμενο επίπεδο. [15],[16]



Σχήμα 2.6: Παράλληλο Simulation.

Κεφάλαιο 3 - Μηχανική Μάθηση και Τεχνητά Νευρωνικά Δίκτυα

Η μηχανική μάθηση είναι υποπεδίο της επιστήμης των υπολογιστών και αναπτύχθηκε από τη μελέτη της αναγνώρισης προτύπων και της υπολογιστικής θεωρίας μάθησης στην τεχνητή νοημοσύνη. Διερευνά τη μελέτη και την κατασκευή αλγορίθμων που μπορούν να μαθαίνουν από τα δεδομένα και να κάνουν προβλέψεις σχετικά με αυτά. Τέτοιοι αλγόριθμοι λειτουργούν κατασκευάζοντας μοντέλα από πειραματικά δεδομένα, προκειμένου να κάνουν προβλέψεις ή να εξάγουν αποφάσεις βασισμένες στα δεδομένα.
Ο υπολογιστής παρατηρεί κάποια δεδομένα, δημιουργεί ένα μοντέλο με βάση τα δεδομένα και χρησιμοποιεί το μοντέλο τόσο ως μια υπόθεση για τον κόσμο όσο και ως ένα πακέτο λογισμικού που μπορεί να επιλύει προβλήματα.
Οι βασικοί λόγοι που έχει αξία να μαθαίνει μια μηχανή είναι δύο. Πρώτον, οι σχεδιαστές δεν μπορούν να προβλέψουν όλες τις δυνατές μελλοντικές καταστάσεις και δεύτερον, οι σχεδιαστές δεν έχουν ιδέα πως να προγραμματίσουν οι ίδιοι μια λύση.[17],[18]

3.1 Τύποι Αλγορίθμων Μηχανικής Μάθησης

Ακριβώς όπως υπάρχουν διαφορετικοί τρόποι με τους οποίους μαθαίνουν οι άνθρωποι από το περιβάλλον τους, το ίδιο ισχύει και στην μηχανική μάθηση. Οι εργασίες μηχανικής μάθησης συνήθως ταξινομούνται σε τρεις μεγάλες κατηγορίες, ανάλογα με τη φύση του εκπαιδευτικού «σήματος» ή την «ανατροφοδότηση» που είναι διαθέσιμα σε ένα σύστημα εκμάθησης. Αυτές είναι:[19]

3.1.1 Επιβλεπόμενη Μάθηση (Supervised learning)

Στην επιβλεπόμενη μάθηση μπορούμε να θεωρήσουμε ότι ο εκπαιδευτής έχει γνώση του περιβάλλοντος και αυτή η γνώση αντιπροσωπεύεται από ένα σύνολο παραδειγμάτων εισόδου - εξόδου. Ωστόσο το περιβάλλον είναι άγνωστο στο νευρωνικό δίκτυο. Η επιθυμητή απόκριση αντιπροσωπεύει τη βέλτιστη ενέργεια που πρέπει να εκτελείται από το νευρωνικό δίκτυο. Οι παράμετροι του δικτύου προσαρμόζονται υπό τη συνδυασμένη επιρροή του διανύσματος εκπαίδευσης και του σήματος σφάλματος (διαφορά μεταξύ της επιθυμητής απόκρισης και της πραγματικής απόκρισης του δικτύου). [19]

3.1.1.1 Λογιστική παλινδρόμηση (Logistic Regression)

Εδώ σκοπός είναι η πρόβλεψη μιας τιμής με μια δεδομένη είσοδο. Η έξοδος του μοντέλου είναι συνεχής. [19]

3.1.1.2 Κατηγοριοποίηση (Classification)

Σκοπός σε τέτοια προβλήματα είναι η επιλογή μιας κατηγορίας για κάποια δεδομένα. Οι τιμές εξόδου σε δυαδική αναζήτηση είναι 0 ή 1, ενώ αν έχουμε πλήθος κατηγοριών, σε κάθε κατηγορία αντιστοιχεί μια ετικέτα - τιμή. [19]

3.1.2 Μη Επιβλεπόμενη Μάθηση (Unsupervised learning)

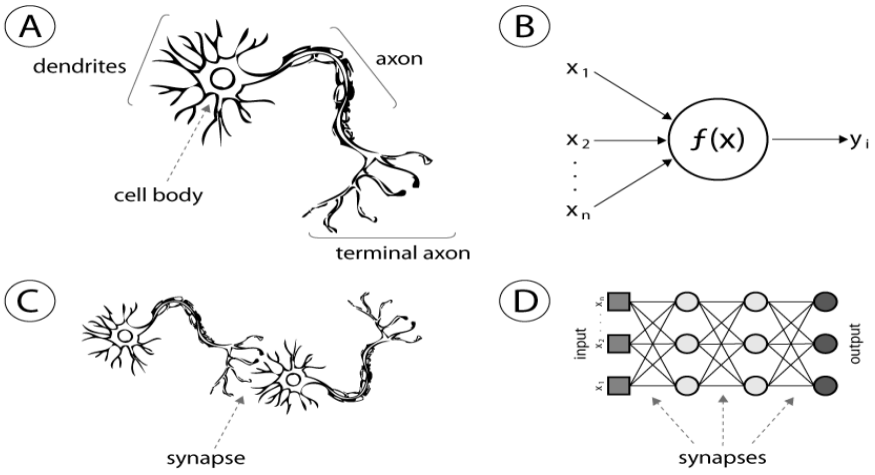
Σε αυτή την κατηγορία δεν υπάρχει εξωτερικός εκπαιδευτής που να επιβλέπει τη διαδικασία. Υπάρχει ένα ανεξάρτητο από την εργασία μέτρο της ποιότητας της αναπαράστασης που καλείται να μάθει το δίκτυο και οι ελεύθερες παράμετροι βελτιστοποιούνται με βάση αυτό το μέτρο. [19]

3.1.3 Ενισχυτική μάθηση (Reinforcement learning)

Στην ενισχυτική μάθηση, η εκμάθηση της αντιστοίχισης εισόδου - εξόδου γίνεται με συνεχή αλληλεπίδραση με το περιβάλλον. Στόχος εδώ είναι να ελαχιστοποιήσει μια συνάρτηση τρέχοντος κόστους. Για κάθε απόφαση που παίρνει, το μοντέλο επιβραβεύεται ή τιμωρείται ανάλογα με τον στόχο. [19]

3.2 Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks)

Ένα τεχνητό νευρωνικό δίκτυο είναι ένας συμπαγής παράλληλος κατανεμημένος επεξεργαστής που έχει τη φυσική κλίση να αποθηκεύει εμπεριστατωμένη γνώση και να την κάνει διαθέσιμη για χρήση. Το νευρωνικό ονομάζεται δίκτυο καθώς αποτελείται από υπολογιστικούς κόμβους που συνδέονται μεταξύ τους. Κάθε υπολογιστικός κόμβος δέχεται ένα σύνολο αριθμητικών εισόδων, εκτελεί έναν υπολογισμό με βάση αυτές τις εισόδους και παράγει μία έξοδο. Η έξοδος αυτή μπορεί είτε να αποτελέσει μέρος της συνολικής εξόδου είτε να διοχετευθεί σε άλλους κόμβους.
Αυτή η προσέγγιση παρουσιάζει ανοχή σε δεδομένα εκπαίδευσης με θόρυβο, δηλαδή δεδομένα που περιστασιακά έχουν λανθασμένες τιμές αλλά αδυνατεί να εξηγήσει ποιοτικά τη γνώση που μοντελοποιεί.



Σχήμα 3.1: Στοιχεία βιολογικών και τεχνητών νευρωνικών δικτύων

Υπάρχουν τρία είδη νευρώνων:

- Οι νευρώνες εισόδου που διοχετεύουν στους υπολογιστικούς νευρώνες την είσοδο του προβλήματος.
- Οι νευρώνες εξόδου που παρουσιάζουν στο περιβάλλον την απόκριση του νευρωνικού.

- Οι υπολογιστικοί νευρώνες που πολλαπλασιάζουν κάθε είσοδό με μια τιμή που ονομάζεται βάρος. Το τελικό αποτέλεσμα εισάγεται σε μια συνάρτηση που θα ονομάζουμε συνάρτηση ενεργοποίησης και είτε παρουσιάζεται στην έξοδο είτε δίνεται σε κάποιον άλλο νευρώνα επεξεργασίας.

Αν θέλουμε να εκφράσουμε την έξοδο ενός νευρώνα επεξεργασίας k θα μπορούσαμε να χρησιμοποιήσουμε την παρακάτω εξίσωση:[20],[21]

$$y_k = f(\sum_{i=1}^{d} w_i x_i + \theta_k)$$

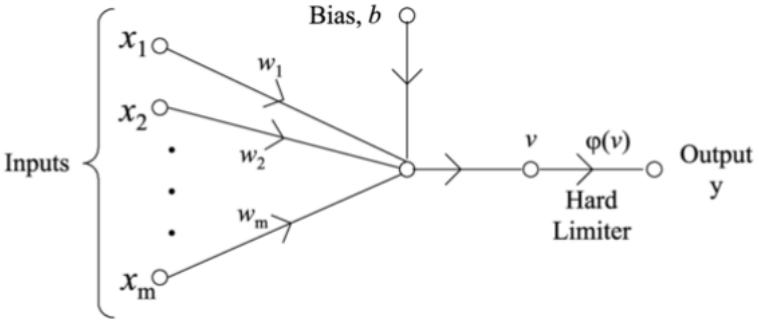
3.2.1 Τεχνητοί νευρώνες - Perceptrons

Το Perceptron βασίζεται στο μοντέλο ενός μη γραμμικού νευρώνα και είναι το βασικότερο δομικό στοιχείο ενός νευρωνικού δικτύου.

Αποτελείται από ένα γραμμικό συνδυαστή, ακολουθούμενο από έναν απότομο περιοριστή. Ο κόμβος άθροισης υπολογίζει ένα γραμμικό συνδυασμό των εισόδων που δέχεται, ενσωματώνοντας μια πόλωση (bias). Τα συναπτικά βάρη του νευρώνα συμβολίζονται με w1, w2, ..., wm και οι αντίστοιχες είσοδοι που εφαρμόζονται στο perceptron ως x1, x2, ..., xm, ενώ την εξωτερική πόλωση ως b. Άρα η είσοδος του απότομου περιοριστή προκύπτει να είναι:

$$u = \sum_{i=1}^{m} w_i * x_i + b$$

Το παραγόμενο αποτέλεσμα, εφαρμόζεται έπειτα στον απότομο περιοριστή, που παράγει ως έξοδο απόκριση +1 ή -1, εάν η είσοδός του είναι θετική ή αρνητική αντίστοιχα. [19]

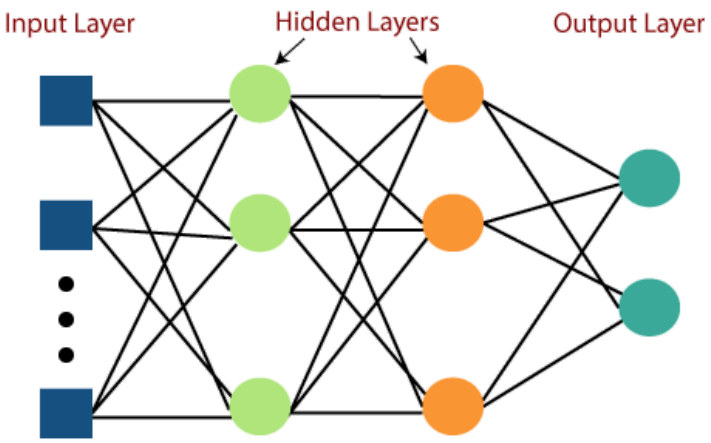


Σχήμα 3.2 : Διάταξη Perceptron

3.2.2 Πολυεπίπεδα Perceptron - Multi Layer Perceptron

Τα perceptron πολλών επιπέδων αποτελούνται από πολλαπλά επίπεδα νευρώνων και είναι η αρχή των βαθιών νευρωνικών δικτύων (Deep Neural Networks).

Αποτελείται από τρία επίπεδα: το επίπεδο εισόδου (input layer), το κρυφό επίπεδο (hidden layer) και το επίπεδο εξόδου (output layer). Τα επίπεδα εισόδου και εξόδου αποτελούνται από ένα στρώμα, ενώ το κρυφό επίπεδο αποτελείται από πολλαπλά στρώματα. Κάθε στρώμα, αποτελείται από ένα σύνολο νευρώνων, το πλήθος των οποίων ονομάζεται πλάτος του επιπέδου (layer width). Κατά κανόνα, πολλαπλά κρυφά επίπεδα χρησιμοποιούνται μονάχα κατά την επίλυση σύνθετων και πολύπλοκων προβλημάτων, καθώς καθιστούν δυσκολότερη και τη διαδικασία της εκπαίδευσης. [19]



Σχήμα 3.3: Διάταξη Perceptron πολλαπλών επιπέδων.

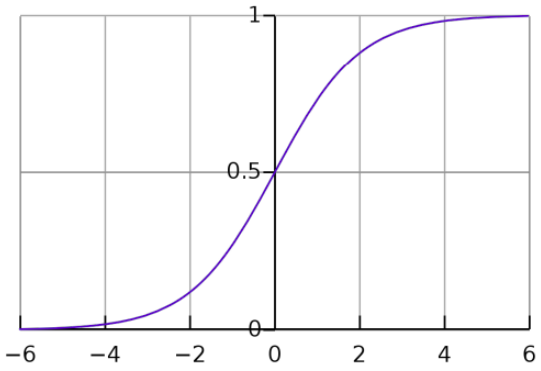
3.3 Εκπαίδευση Νευρωνικών Δικτύων

Εκπαίδευση είναι η διαδικασία που μεταβάλλουμε τις παραμέτρους του νευρωνικού δικτύου, με σκοπό να επιτυγχάνει σωστή πρόβλεψη. Σημαντική παράμετρος της εκπαίδευσης είναι τα βάρη του νευρώνα. ΄Ενα αποτελεσματικό νευρωνικό πρέπει να εκπαιδεύεται γρήγορα με όσο λιγότερο υπολογιστικούς πόρους.
Η γενικότερη ιδέα είναι πως αφού εκπαιδευτεί με βάση κάποια γνωστά δεδομένα, θα μπορεί να κάνει σωστές προβλέψεις για νέα δεδομένα. Η κάθε επανάληψη της διαδικασίας ονομάζεται εποχή (epoch) και ο αριθμός τον εποχών που χρησιμοποιούμε παίζει καθοριστικό ρόλο στην απόδοση του νευρωνικού δικτύου. Σπουδαίο ρόλο εδώ παίζει και η συνάρτηση κόστους καθώς και ο αλγόριθμος βελτιστοποίησης. [19]

3.3.1 Συνάρτηση Ενεργοποίησης (Activation Function)

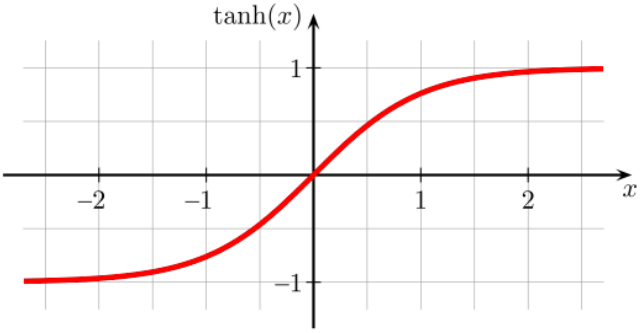
Η συνάρτηση ενεργοποίησης αφορά τον τρόπο που οι είσοδοι κάθε κόμβου μετατρέπονται σε έξοδο. Κάθε επίπεδο του δικτύου χρησιμοποιεί την ίδια μη γραμμική συνάρτηση ενεργοποίησης. Η μόνη απαίτηση που πρέπει να ικανοποιεί μια συνάρτηση ενεργοποίησης είναι η διαφορισιμότητα.
Οι βασικότερες συναρτήσεις ενεργοποίησης είναι:

A. Σιγμοειδής Συνάρτηση (Sigmoid Function): η συνάρτηση αυτή αντιστοιχεί κάθε τιμή εισόδου σε μια τιμή στο διάστημα (0, 1), με τις μικρότερες τιμές να πλησιάζουν στο 0, και τις μεγαλύτερες να πλησιάζουν στο 1 (χωρίς να παίρνει τις τιμές αυτές).



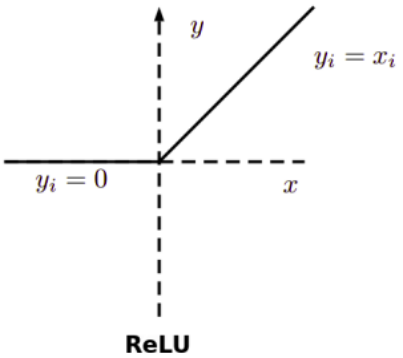
Σχήμα 3.4: Σιγμοειδής Συνάρτηση (Sigmoid Function)

B. Υπερβολική Εφαπτομένη (Hyperbolic Tangent): Η συνάρτηση αυτή αντιστοιχίζει τις τιμές εισόδου στο διάστημα (-1, 1). Οι τιμές κοντά στο 0 δεν μεταβάλλονται σημαντικά και άρα δεν συμβάλλουν πολύ στη διαδικασία της εκπαίδευσης.



Σχήμα 3.5: Υπερβολική Εφαπτομένη (Hyperbolic Tangent)

C. Rectified Linear Unit (ReLu): Εδώ οι θετικές τιμές δεν μεταβάλλονται, ενώ οι τιμές που είναι μικρότερες του μηδενός μηδενίζονται και δεν συμμετέχουν στην εκπαίδευση.



Σχήμα 3.6: Rectified Linear Unit (ReLu)

D. Συνάρτηση Softmax: Η συνάρτηση Softmax διαμορφώνει το σύνολο των τιμών ώστε το άθροισμά να ισούται με 1 και να εκφράζονται με κατάλληλο τρόπο οι πιθανότητες που αντιστοιχούν σε κάθε κλάση. [19]

3.3.2 Συνάρτηση Κόστους (Cost Function)

Η συνάρτηση κόστους μετρά την απόδοση ενός μοντέλου για ένα σύνολο δεδομένων. Ποσοτικοποιεί το σφάλμα μεταξύ προβλεπόμενων και αναμενόμενων τιμών και παρουσιάζει αυτό το σφάλμα με τη μορφή ενός μοναδικού πραγματικού αριθμού. Ανάλογα με το πρόβλημα, η συνάρτηση κόστους μπορεί να διαμορφωθεί με πολλούς διαφορετικούς τρόπους. Ο σκοπός της συνάρτησης αυτής είναι είτε να ελαχιστοποιηθεί είτε να μεγιστοποιηθεί. Κάθε τέτοια συνάρτηση πρέπει να είναι διαφοροποιήσιμη.
Οι βασικές συναρτήσεις κόστους είναι :

A. Μέσο Απόλυτο Σφάλμα (Mean Absolute Error)
   Το μέσο απόλυτο σφάλμα μετράει το μέσο σφάλμα σε μια ομάδα προβλέψεων, χωρίς να λαμβάνει υπόψη τις κατευθύνσεις τους (είναι ο μέσος όρος των απόλυτων διαφορών μεταξύ των προβλέψεων και των αναμενόμενων αποτελεσμάτων).

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |\hat{y}^{(i)} - y^{(i)}|$$

B. Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error)
   Το μέσο τετραγωνικό σφάλμα αντιπροσωπεύει τη μέση τετραγωνική διαφορά μεταξύ των προβλέψεων και των αναμενόμενων αποτελεσμάτων. Πρόκειται για μια αλλαγή του MAE όπου, αντί να πάρουμε την απόλυτη τιμή των διαφορών, τετραγωνίζουμε αυτές τις διαφορές. Κάθε μερικό σφάλμα είναι ισοδύναμο με το εμβαδόν του τετραγώνου που δημιουργείται από τη γεωμετρική απόσταση μεταξύ των μετρούμενων σημείων.

$$MSE = \frac{1}{2m}\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^{2}$$

C. Απώλεια Εγκάρσιας Εντροπίας (Cross Entropy Loss)
   Χρησιμοποιείται για μοντέλα ταξινόμησης, όπου η έξοδος είναι μεταξύ 0 και 1. [22]

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i * log(\hat{y}_i)$$

3.3.3 Αλγόριθμος Βελτιστοποίησης (Optimization Algorithm)

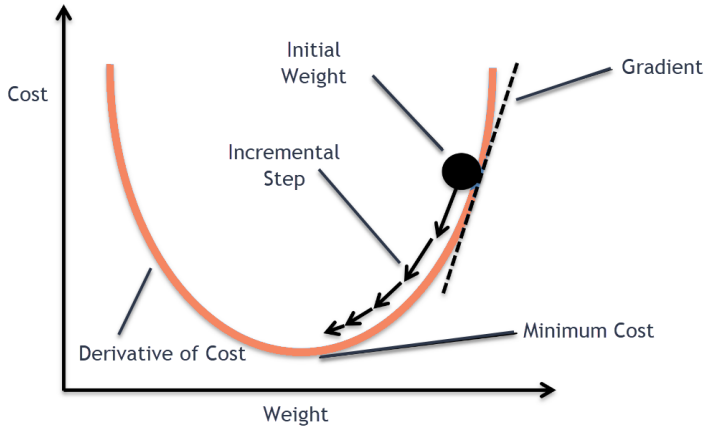
Ο αλγόριθμος βελτιστοποίησης σε ένα νευρωνικό δίκτυο αφορά την εύρεση βέλτιστων βαρών των κόμβων με σκοπό την ελαχιστοποίηση της τιμής της συνάρτησης κόστους. Η επιλογή αυτής της συνάρτησης έχει σημαντικές επιπτώσεις στην ακρίβεια και την ταχύτητα εκπαίδευσης του μοντέλου.
Οι αλγόριθμοι αυτοί κάνουν χρήση της μερικής παραγώγου. Η συχνότερα χρησιμοποιούμενη μέθοδος για τον υπολογισμό των παραγώγων, είναι η μέθοδος της οπισθοδιάδοσης (Backpropagation). Οι αλγόριθμοι αυτοί έπονται του υπολογισμού της εξόδου και μεταφέρουν την πληροφορία από την έξοδο προς την είσοδο, κάνοντας χρήση του κανόνα αλυσίδας.

3.3.3.1 Αλγόριθμος Καθόδου με βάση την Κλίση (Gradient Descent)

Ένας από τους γνωστότερους αλγόριθμους βελτιστοποίησης και συχνότερα χρησιμοποιούμενος για τη βελτιστοποίηση νευρωνικών δικτύων, είναι ο αλγόριθμος καθόδου

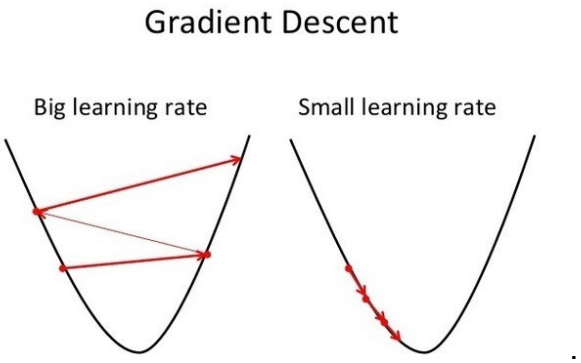
με βάση την κλίση (Gradient Descent). Βασίζεται σε μια κυρτή συνάρτηση και τροποποιεί τις παραμέτρους της επαναληπτικά για να ελαχιστοποιήσει μια δεδομένη συνάρτηση στο τοπικό της ελάχιστο. Το αλγόριθμος καθόδου χρησιμοποιείται απλώς για να βρεθούν οι τιμές των παραμέτρων μιας συνάρτησης που ελαχιστοποιούν μια συνάρτηση κόστους (να βρεθεί κάποιο τοπικό ελάχιστο της συνάρτησης).

Στόχος της συνάρτησης είναι να φτάσουμε στο κάτω μέρος του γραφήματος ή σε ένα σημείο όπου δεν μπορούμε πλέον να κινηθούμε κατηφορικά (ένα τοπικό ελάχιστο).



Σχήμα 3.7: Γράφημα Gradient Descent (Κόστος - Βάρη)

Το πόσο μεγάλες αποκλίσεις έχει η κλίση προς την κατεύθυνση του τοπικού ελάχιστου καθορίζεται από τον ρυθμό εκμάθησης, ο οποίος υπολογίζει πόσο γρήγορα ή πόσο αργά θα κινηθούμε προς τα βέλτιστα βάρη. Για να φτάσει στο τοπικό ελάχιστο, πρέπει να ορίσουμε τον ρυθμό εκμάθησης σε μια κατάλληλη τιμή, η οποία δεν είναι ούτε πολύ χαμηλή ούτε πολύ υψηλή. [23]

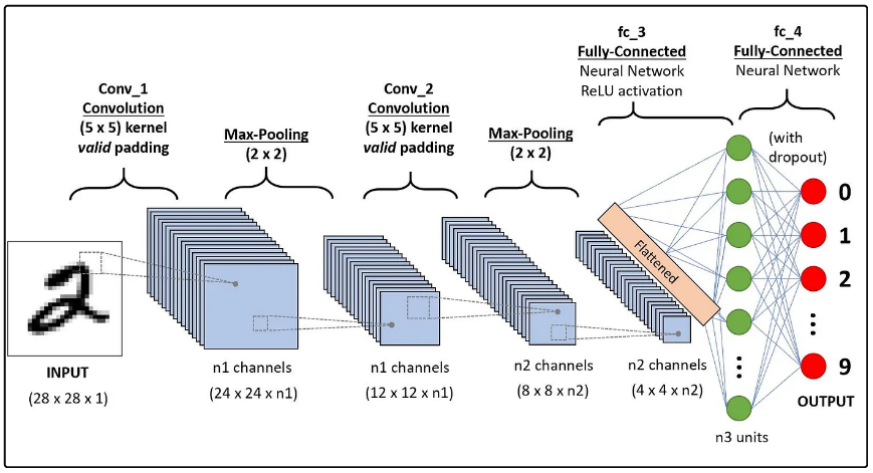


Σχήμα 3.8: Μικρό και μεγάλο Learning rate

3.4 Συνελικτικά νευρωνικά δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (CNN) αποτελούν μια κατηγορία νευρωνικών που χρησιμοποιούνται σε προβλήματα με εικόνες. Είναι μια εκδοχή του Perceptron πολλαπλών επιπέδων. Η αρχιτεκτονική τους είναι ανάλογη με αυτή του μοτίβου συνδεσιμότητας των

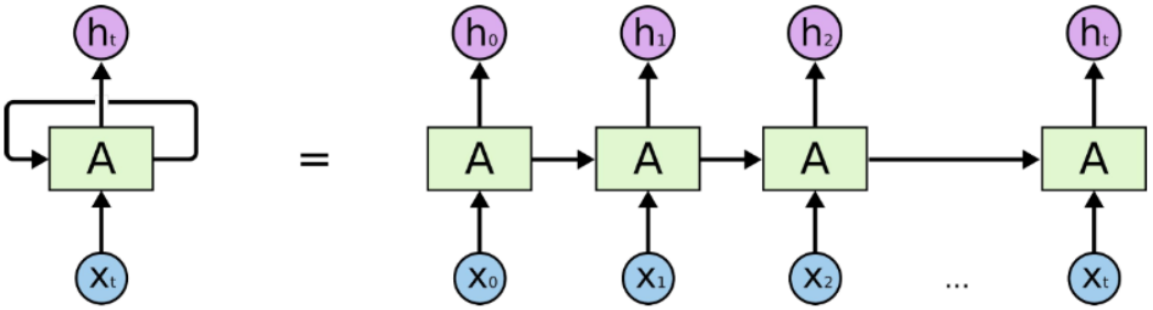
Νευρώνων στον Ανθρώπινο Εγκέφαλο και εμπνεύστηκε από την οργάνωση του Οπτικού Φλοιού.
Τα κύρια επίπεδα ενός συνελικτικού νευρωνικού δικτύου είναι το στρώμα συνέλιξης (Convolution Layer), το στρώμα ομαδοποίησης (Pooling Layer) και ένα πλήρως συνδεδεμένο στρώμα (Fully Connected Layer). Το πρώτο στρώμα εκτελεί εσωτερικά γινόμενα μεταξύ δύο πινάκων. Πρόκειται για το εσωτερικό γινόμενο μεταξύ του συνόλου των βαρών (1ος πίνακας), γνωστά και ως kernel, και ενός υποσυνόλου δεδομένων (2ος πίνακας). Το kernel πρέπει να έχει το ίδιο βάθος με την εικόνα.
Με τον όρο βάθος εννοούμε τα κανάλια που ορίζουν την εικόνα (για παράδειγμα τα 3 RGB κανάλια μιας έγχρωμης εικόνας). Το kernel θα διασχίσει την εικόνα κατά ύψος και πλάτος και θα δημιουργηθεί ένας νέος πίνακας (Activation Map). Το βήμα μετακίνησης του kernel ονομάζεται stride. Στο δεύτερο επίπεδο του δικτύου αντικαθίστανται οι έξοδοι του δικτύου με τη μέση τιμή των γειτονικών κόμβων. Στο τελευταίο στρώμα γίνεται η αντιστοίχιση εισόδου-εξόδου.[24]



Σχήμα 3.9: Παράδειγμα συνελικτικού νευρωνικού δικτύου

3.5 Επαναλαμβανόμενα νευρωνικά δίκτυα

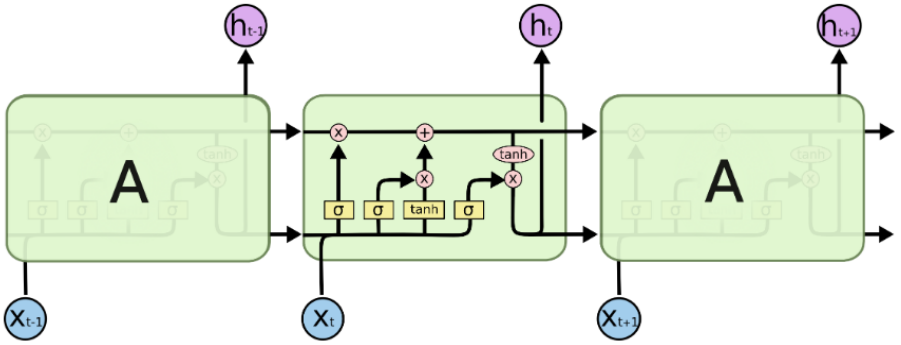
Τα επαναλαμβανόμενα νευρωνικά δίκτυα (RNN) είναι μια κατηγορία τεχνητών νευρωνικών δικτύων όπου οι συνδέσεις μεταξύ κόμβων μπορούν να δημιουργήσουν έναν κύκλο, επιτρέποντας σε κόμβους να κρατούν προηγούμενες πληροφορίες που έπειτα χρησιμοποιούν σε συνδυασμό με τις επόμενες εισόδους, δημιουργώντας εξαρτήσεις μεταξύ των δεδομένων. Τα RNN μπορούν να χρησιμοποιήσουν την εσωτερική τους κατάσταση (μνήμη) για να επεξεργαστούν ακολουθίες μεταβλητού μήκους εισόδων. Μοιάζει με μία αλυσίδα που κάθε κελί μεταφέρει κάποια πληροφορία στο επόμενο. Τα RNN δεν αποδίδουν ικανοποιητικά σε ακολουθίες όπου η πληροφορία κάποιου σταδίου, μπορεί να βρίσκεται αρκετά βήματα μακριά. [25]



Σχήμα 3.10: Παράδειγμα επαναλαμβανόμενου νευρωνικού δικτύου

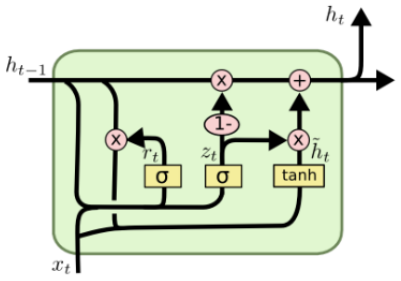
3.5.1 Δίκτυα μακράς βραχυπρόθεσμης μνήμης

Τα δίκτυα μακράς βραχυπρόθεσμης μνήμης (Long Short-Term Memory - LSTM) κάνουν κάποιες επιπλέον πράξεις σε σχέση με τα RNN (πολλαπλασιασμούς και προσθέσεις). Το δίκτυο πλέον μπορεί να ξεχνά ή να θυμάται δεδομένα ανάλογα με τη σημασία τους.
Κάθε κελί δέχεται την είσοδο xt, καθώς και δεδομένα από την προηγούμενη κατάσταση, τα επεξεργάζεται και παράγει μια έξοδο ht και πληροφορίες που περνάνε στο επόμενο κελί.



Σχήμα 3.11:Παράδειγμα δικτύου μακράς βραχυπρόθεσμης μνήμης.

Μια παραλλαγή των κλασσικών LSTM είναι τα Gated Recurrent Unit (GRU). Αυτή η παραλλαγή χρησιμοποιεί μια πύλη ενημέρωσης με πληροφορία από το προηγούμενο κελί συνδυάζοντας τις δύο εισόδους της προηγούμενης προσέγγισης.



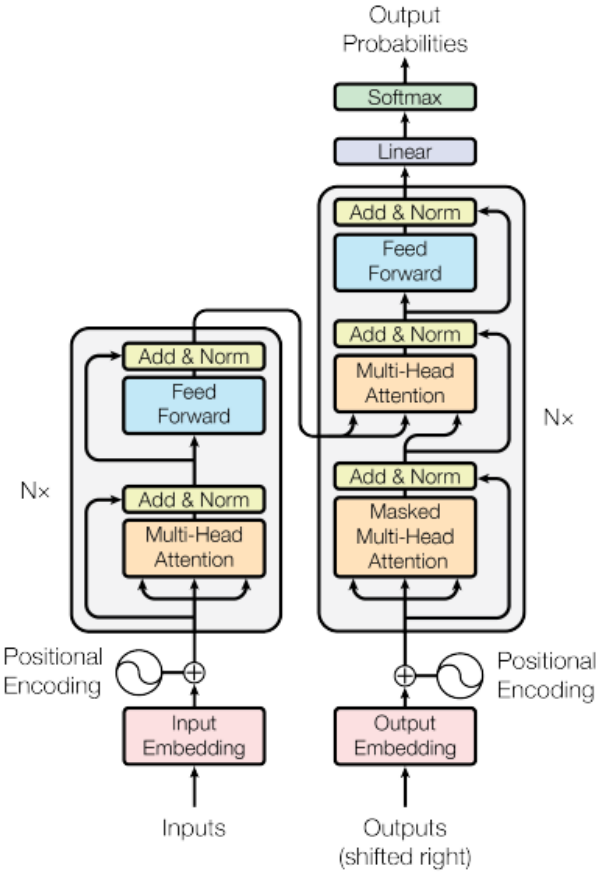
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Σχήμα 3.12:Παραλλαγή δικτύου μακράς βραχυπρόθεσμης μνήμης.

Αυξάνοντας το μήκος των ακολουθιών το μοντέλο δυσκολεύεται να ξεχωρίσει ποια δεδομένα είναι σημαντικά και ποια όχι.[26],[27]

3.5.3 Μηχανισμός Προσοχής - Transformer

Ένας νέος μηχανισμός που προτάθηκε είναι ο μηχανισμός προσοχής. Μέσω αυτού του μηχανισμού η κάθε έξοδος δεν προκύπτει μόνο από την προηγούμενη κατάσταση, αλλά είναι συνδυασμός από όλες τις προηγούμενες καταστάσεις. Έτσι αξιολογείται η σημασία κάθε μέρους των δεδομένων εισόδου. Κάθε έξοδος προκύπτει από το sum-product των βαρών του μηχανισμού προσοχής (attention weights) και όλων των κρυφών καταστάσεων. Αυτή είναι και η βασική διαφοροποίηση από την προσέγγιση των LSTM.
Ο παραπάνω μηχανισμός βελτιώνει την απόδοση των Επαναλαμβανόμενων Νευρωνικών Δικτύων (RNN) με την διαδικασία αυτή όμως να είναι αργή στην εκπαίδευση μεγάλων ακολουθιών δεδομένων.
Ένα νέο μοντέλο νευρωνικού δικτύου που παρουσιάστηκε είναι ο Transformer που βασίζεται αποκλειστικά στον μηχανισμό προσοχής και όχι στη χρήση RNNs. Αυτά τα μοντέλα χρειάζονται πολύ λιγότερο χρόνο για την εκπαίδευση τόσο σε μικρά όσο και μεγάλα σύνολα δεδομένων λόγω της μεγάλης παραλληλίας που επιτρέπει.
Σχετικά με την αρχιτεκτονική του μοντέλου αυτή συνήθως αποτελείται από μια δομή κωδικοποιητή - αποκωδικοποιητή (encoder - decoder).



Σχήμα 3.13: Βασική Δομή Transformer

Εσωτερικά κάθε μέρος αποτελείται από πολλούς κωδικοποιητές (και αποκωδικοποιητές αντίστοιχα). Ο κάθε κωδικοποιητής αποτελείται από έναν μηχανισμό προσοχής και ένα τροφοδοτούμενο προς τα εμπρός δίκτυο (feed forward network). Μεταξύ των δύο αυτών υποεπιπέδων υπάρχει σύνδεση καθώς και ένα επίπεδο κανονικοποίησης. Σκοπός του κωδικοποιητή είναι να αναγνωρίσει την συσχέτιση του κάθε στοιχείου της ακολουθίας με τα υπόλοιπα. Όμοια είναι και η δομή του αποκωδικοποιητή αναζητά συσχετίσεις των δεδομένων της ακολουθίας εξόδου, ώστε σε συνδυασμό με τα δεδομένα που λαμβάνει από τον κωδικοποιητή να προβλέψει την επόμενη έξοδο.
Σκοπός του κωδικοποιητή είναι να εντοπίσει τη συσχέτιση κάθε στοιχείου της ακολουθίας με τα υπόλοιπα. Όμοια και ο αποκωδικοποιητής αναζητά συσχετίσεις στα δεδομένα εξόδου ώστε σε συνδυασμό με τα δεδομένα του κωδικοποιητή να προβλέψει την έξοδο.[28]

3.5.3.1 Regression Transformer

Οι Transformers παλινδρόμησης (regression) αποτελούν ένα από τα βασικότερα και ευρέως χρησιμοποιούμενα είδη αυτής της οικογένειας νευρωνικών. Σε τέτοια προβλήματα (παλινδρόμησης) τα δεδομένα είναι σε μορφή πινάκων, είναι δηλαδή μια σειρά (tuple) από αριθμούς ή κατηγορίες που τοποθετούνται σε διαφορετικά πεδία (fiends). Για να γίνει πιο κατανοητό παρακάτω φαίνεται ένα παράδειγμα ενός δείγματος - δεδομένου που ανήκει στο σύνολο δεδομένων που διαχειρίζεται το μοντέλο του Ithemal.

{"id": 4889de4889c24c89ff4889de4889c24c89ff , "text": "mov rsi,rbx; mov rdx,rax; mov rdi,r15; mov rsi,rbx; mov rdx,rax; mov rdi,r15; dec r10; jne 0x0 ", "uuid": "4889de4889c24c89ff4889de4889c24c89ff ", "score": 175}

Όπου στην παραπάνω πλειάδα στο id καθώς και στο uuid αντιστοιχείται το hex code του εκάστοτε κώδικα, στο πεδίο text τοποθετείται η assembly αναπαράσταση του κώδικα και στο πεδίο score η μέτρηση που έχει βρεθεί από ένα από τα διαθέσιμα εργαλεία.
Αυτού του είδους τα νευρωνικά έχουν ευρεία χρήση σε προβλήματα με εικόνες, κείμενο και ομιλία αλλά η περίπτωση που βασιζόμαστε εδώ είναι αυτή του κειμένου, όπου με βάση αυτό το δίκτυο που αναπτύσσεται αναλύει το κείμενο-εντολές και προβλέπει το throughput.[29]

Κεφάλαιο 4 - Ανάλυση Dataset

4.1 Περιγραφή ανάγκης δημιουργίας BHive Dataset

Οι σχεδιαστές μεταγλωττιστών και οι μηχανικοί απόδοσης τη σημερινή εποχή χρησιμοποιούν μοντέλα υπολογισμού απόδοσης με σκοπό να έχουν μια αξιόπιστη πρόβλεψη των κύκλων ρολογιού για την εκτέλεση του κώδικά τους. Η δημιουργία αυτών των μοντέλων είναι κοπιαστική. Τα μοντέλα τα οποία αναπτύσσονται έχουν σκοπό την πρόβλεψη χωρίς την εκτέλεση του εκάστοτε κώδικα. Οι πρώτες προσεγγίσεις για τέτοια εργαλεία δεν έχουν αξιολογηθεί με συστηματικό τρόπο ώστε να είναι αξιόπιστα τα αποτελέσματα τα οποία προκύπτουν. Μια πρόκληση αποτελεί η συλλογή κατάλληλων δεδομένων που θα βοηθήσουν στην ορθή αξιολόγηση. Οι δημιουργοί του Ithemal με τη χρήση ενός αναλυτή (profiler) δημιούργησαν το σύνολο δεδομένων BHive που αποτελεί ένα benchmark κατάλληλο για συστηματική αξιολόγηση της απόδοσης μοντέλων x86-64. [30],[31]

4.1.1 Απαιτήσεις και δημιουργία του BHive Dataset

Υπάρχουν δύο είδη από τέτοια μοντέλα εκτίμησης της απόδοσης, οι microarchitectural προσομοιωτές και τα per instruction latency ή throughput lookup tables. Βασική στρατηγική των εργαλείων αυτών που κάνουν εκτίμηση απόδοσης είναι να κάνουν unroll τον κώδικα πολλές φορές ώστε να διαιρέσουν την προκύπτουσα καθυστέρηση με τον εκάστοτε unrolling factor. Μια εναλλακτική προσέγγιση είναι η παραπάνω διαδικασία να γίνει με δύο unrolling factors. Το σύνολο δεδομένων το οποίο θα δημιουργηθεί θα πρέπει προφανώς να είναι εφικτό να χρησιμοποιηθεί σε διαφορετικές αρχιτεκτονικές. Για να γίνει αυτό πρέπει να γίνει αντιστοίχηση των virtual σε physical pages, να εντοπιστούν cache misses και να εντοπιστούν τα floating-points. Με βάση τα παραπάνω AMD, Intel και ARM αρχιτεκτονικές είναι αποδεκτές.
Παρακάτω, στο σχήμα 1 παρουσιάζονται οι εφαρμογές από τις οποίες έγινε η εξαγωγή των δεδομένων για το BHive. Οι εφαρμογές αυτές πρέπει να καλύπτουν ένα ευρύ φάσμα εφαρμογών ώστε να αντιπροσωπεύουν επαρκώς τις ανάγκες του σύγχρονου κόσμου και τα αποτελέσματα να είναι καλύτερα. Επίσης, τα κύρια τμήματα κώδικα που απαρτίζουν το σύνολο των δεδομένων θα πρέπει να αντικατοπτρίζουν τα θέματα των τυπικών χρηστών ενός μοντέλου εκτίμησης απόδοσης (performance model).

Application	Domain	#Basic Blocks
OpenBLAS	Scientific Computing	19032
Redis	Database	9343
SQLite	Database	8871
GZip	Compression	2272
Tensorflow	Machine Learning	71988
Clang/LLVM	Compiler	212758
Eigen	Scientific Computing	4545
Embree	Ray Tracing	12602
FFmpeg	Multimedia	17150
Total		330018

Πίνακας 4.1: Δεδομένα διαφορετικών κατηγοριών για τη σύνθεση του BHive Dataset.

Πιο αναλυτικά οι εφαρμογές Clang/LLVM (compiler), Redis (database), SQLite (database) και Gzip (compression) προκειμένου να συλλεχθούν δεδομένα (basic blocks) τα οποία είναι αντιπροσωπευτικά εφαρμογών που είναι κατά βάση γραμμένα σε C και C++ γλώσσες. Πρόκειται για ευρείας χρήσης εφαρμογές που χρησιμοποιούν σύνθετους αλγορίθμους (sophisticated algorithms) και δομές δεδομένων (data structures). Έπειτα επιλέγονται εφαρμογές που χρησιμοποιούν hand-optimized high performance kernels. Οι εφαρμογές αυτές είναι οι OpenSSL (cryptography), OpenBLAS, Eigen (scientific computing), TensorFlow (machine learning), Embree (rendering) και FFmpeg (multimedia), όπου ορισμένες από αυτές χρησιμοποιούν assembly για τους κρίσιμους εσωτερικούς βρόχους και είναι γραμμένες σε γλώσσα που βοηθά την παραλληλοποίηση των δεδομένων με εστίαση στα vector extensions της Intel. Η εξαγωγή των τμημάτων κώδικα έγινε με τη χρήση του εργαλείου DynamoRio, το οποίο επιτρέπει την καταγραφή όλων των τμημάτων κώδικα που εκτελούνται. [30],[31]

4.1.2 Ομαδοποίηση του BHive Dataset

Ορισμένα από τα βασικά κομμάτια κώδικα είναι πιο δύσκολο να διαμορφωθούν σε σχέση με άλλα, κυρίως λόγω των εξαρτήσεων μνήμης. Για το λόγο αυτό παρουσιάζεται μια τεχνική ομαδοποίησης βασιζόμενο στις ανάγκες υπολογιστικών πόρων (processor resources). Με τον τρόπο αυτό επιτρέπεται στους προγραμματιστές και γενικά στους χρήστες να αντιληφθούν καλύτερα τη συμπεριφορά του μοντέλου, επιτρέποντάς τους να εστιάσουν σε κατηγορίες δεδομένων που δημιουργούν μεγαλύτερες προκλήσεις. Η τεχνική της ομαδοποίησης ακολουθεί τα παρακάτω βήματα. Αρχικά γίνεται αντιστοίχηση του κάθε μέρους κώδικα με μια αναπαράσταση η οποία αντικατοπτρίζει τις ανάγκες υλικού (hardware resources) ώστε να γίνει η εκάστοτε εκτέλεση, ενώ έπειτα με βάση την αναπαράσταση αυτή

γίνεται και η ζητούμενη ομαδοποίηση. Οι κατηγορίες οι οποίες προκύπτουν μαζί με την πληθικότητά τους φαίνονται στο παρακάτω πίνακα.

Category	Description	#Basic Blocks
Scalar	Scalar ALU operations	85208
Vec	Purely vector instructions	1267
Scalar/Vec	Scalar and vector arithmetic	7710
Ld	Mostly loads	121412
St	Mostly stores	55879
Ld/St	Mix of loads and stores	58540

Πίνακας 4.2: Περιγραφή κατηγοριών basic blocks.

Στο παρακάτω σχήμα φαίνονται παραδείγματα κάθε μιας κατηγορίας ώστε να γίνει ξεκάθαρο το είδος των τμημάτων κώδικα που ανήκουν σε κάθε κατηγορία.

Scalar

```
movzbl 2(%rdi), %eax
shrb $2, %al
andl $31, %eax
cmpl $1, %eax
```

Vec

```
vmovss   -4(%rax), %xmm2
vmovaps  %xmm2, %xmm3
vmovss   (%rax), %xmm1
vxorps   %xmm4, %xmm3,
                   %xmm3
vucomiss %xmm3, %xmm1
```

Scalar/Vec

```
movsd   (%rcx), %xmm1
movsd   (%rsi), %xmm0
movaps  %xmm1, %xmm2
movaps  %xmm0, %xmm3
mulps   %xmm14, %xmm0
          ...
addps   %xmm1, %xmm0
subps   %xmm3, %xmm2
movlps  %xmm0, (%rsi)
movlps  %xmm2, (%rcx)
addq    $8, %rsi
addq    $8, %rcx
subq    $2, %rdi
```

Ld

```
movq (%rbp), %rax
movq %rbx, %rsi
movq %rbp, %rdi
popq %rbx
popq %rbp
popq %r12
movq 32(%rax), %rax
```

St

```
pushq %r14
movq  %rdi, %r14
leaq  8(%rdi), %rdi
pushq %r13
pushq %r12
pushq %rbp
pushq %rbx
```

Ld/St

```
movq (%rbp), %rax
movq %rbx, %rsi
movq %rbp, %rdi
popq %rbx
popq %rbp
popq %r12
movq 32(%rax), %rax
```

Πίνακας 4.3: Παραδείγματα basic blocks από κάθε κατηγορία.

Βασικός στόχος αυτής της ομαδοποίησης είναι να βοηθήσει στην γενίκευση και σε νέα blocks τα οποία δεν έχουν μελετηθεί νωρίτερα από το εκάστοτε μοντέλο. [30],[31]

4.2 Αξιολόγηση του συνόλου δεδομένων BHive

Η αρχική αξιολόγηση του συνόλου δεδομένων έγινε σε σύγχρονες αρχιτεκτονικές της Intel. Τα μοντέλα τα οποία χρησιμοποιήθηκαν για την αξιολόγηση είναι τα IACA, llvm-mca, OSACA και Ithemal. Για την παρουσίαση των αποτελεσμάτων της αξιολόγησης χρησιμοποιήθηκαν οι εξής μετρικές:

- Το συνολικό σφάλμα (Overall error) που αφορά το μέσο σφάλμα όλων των τμημάτων κώδικα σε μια συγκεκριμένη μικροαρχιτεκτονική.
- Το Per-application σφάλμα το οποίο αναφέρεται στο μέσο σφάλμα όλων των τμημάτων κώδικα μιας δοσμένης εφαρμογής σε μια μικροαρχιτεκτονική.
- Το Per-category σφάλμα που αφορά το μέσο σφάλμα όλων των τμημάτων κώδικα μιας κατηγορίας (όπως αναλύθηκαν παραπάνω) για μια συγκεκριμένη μικροαρχιτεκτονική.
- Το Kendall's tau που χρησιμοποιείται ευρέως για αξιολόγηση συστημάτων.

Τα αποτελέσματα φαίνονται αναλυτικά στον παρακάτω πίνακα.

Microarchitecture	Model	Average Error	Kendall's Tau
Ivy Bridge	IACA llvm - mca **Ithemal** OSACA	0.1664 0.2813 **0.0973** 0.3299	0.8888 0.7544 **0.8442** 0.6197
Haswell	IACA llvm - mca **Ithemal** OSACA	0.1790 0.2511 **0.0926** 0.3566	0.8043 0.7829 **0.8544** 0.6067
Skylake	IACA llvm - mca **Ithemal** OSACA	0.1566 0.2683 **0.0980** 0.3573	0.8121 0.7745 **0.8516** 0.6111

Πίνακας 4.4: Αποτελέσματα αξιολόγησης του Bhive dataset.

Τα συμπεράσματα που μπορούν να βγουν από την αξιολόγηση των μοντέλων είναι αρχικά ότι το IACA είναι το δεύτερο πιο ακριβές μοντέλο για τις περισσότερες κατηγορίες. Επίσης το llvm-mca είναι σημαντικά χειρότερο από τα IACA και Ithemal ενώ το OSACA είναι καλύτερο από το llvm-mca αλλά όχι από τα IACA και Ithemal. Τέλος, το Ithemal υπερνικά σαφώς τα υπόλοιπα μοντέλα με εξαίρεση τα blocks τα οποία περιέχουν vectors. [30],[31]

4.2.1 Παραδοχές του BHive Dataset

Κατά την προσπάθεια αξιολόγησης του εργαλείου uiCA προτάθηκε αρχικά ως σύνολο δεδομένων το BHive που χρησιμοποιούνταν όπως αναφέρεται και παραπάνω για αξιολόγηση παρόμοιων εργαλείων. Αποτελείται από 300.000 διαφορετικά κομμάτια κώδικα που έχουν συλλεχθεί από κατάλληλες εφαρμογές. Παρόλα αυτά παρατηρήθηκαν πολλές ανακρίβειες και κακές προβλέψεις. Οπότε προέκυψε η ανάγκη να αντιμετωπιστούν τέτοια θέματα. Τα θέματα

αυτά συχνά πηγάζουν από τις παραδοχές που έχουν γίνει κατά την προσομοίωση του κώδικα. Οι κύριες παραδοχές στις οποίες πρέπει να προσαρμοστεί το BHive είναι οι παρακάτω:
Δεν υπάρχουν TLB Misses και συγκεκριμένα όλα οι προσβάσεις στην μνήμη οδηγούν σε TLB και cache hits και δεν υπάρχουν καθόλου αστοχίες. Υπάρχουν ορισμένες “μη ισορροπημένες” εντολές όπου σε πολλά τμήματα εντολών περιέχεται μεγάλο πλήθος από push και pop στην x86 floating-point stack. Τα σωστά προγράμματα δεν εκτελούν σε επανάληψη τέτοιες εντολές καθώς μπορεί να οδηγήσουν σε καθυστερήσεις εκατοντάδων κύκλων ρολογιού. Ακόμα υπάρχουν ορισμένες μη υποστηριζόμενες εντολές που έχουν μη υποστηριζόμενο πρόθεμα όπως είναι η TZCNT εντολή που προστέθηκε στην μικροαρχιτεκτονική του Haswell. Μια ακόμα παραδοχή είναι ότι οι χρόνοι του εκάστοτε κώδικα ενδέχεται να είναι input-dependent. Παρόλα αυτά κάτι τέτοιο δεν γίνεται από το BHive. Αντίθετα αυτό το οποίο κάνει είναι να αρχικοποιεί όλους τους καταχωρητές στην ίδια τιμή το οποίο συχνά οδηγεί σε “αλλοιώσεις” της μνήμης (memory aliasing). [30],[31]

4.3 Παραλλαγή του BHive Dataset(BHiveU & BHiveL)

Η απόδοση ενός τμήματος κώδικα συνήθως ορίζεται ως ο μέσος αριθμός των κύκλων ρολογιού ανά εντολή όταν εκτελείται επαναλαμβανόμενα και σε σταθερή κατάσταση. Όμως υπάρχει ένα σημείο αναφοράς ανάλογα με το είδος της κάθε εντολής. Συγκεκριμένα, στην περίπτωση των εντολών που τελειώνουν με διακλάδωσης (και άρα η εκτέλεση μεταπηδά πάλι στην αρχή του κώδικα) και με δεδομένο ότι η διακλάδωση είναι always taken θα οδηγήσει στην δημιουργία ενός ατέρμονου βρόχου. Αντίθετα αυτά που δεν τελειώνουν με εντολή διακλάδωσης μπορούν να εκτελεστούν με χρήση του unrolling factor και να εκτελεστούν επαναλαμβανόμενα σε σταθερή κατάσταση ώστε να βγάλουν ένα ασφαλές αποτέλεσμα. Η παραπάνω διαφοροποίηση οδηγεί στην ύπαρξη και χρήση δύο διαφορετικών τιμών απόδοσης, του TPL (για αυτά που τελειώνουν με διακλάδωσης) και του TPU (για αυτά που δεν τελειώνουν με εντολή διακλάδωσης).
Τα benchmarks του BHive δεν τελειώνουν με εντολές διακλάδωσης οπότε ο αναλυτής (profiler) του συνόλου δεδομένων κάνει χρήση του TPU για την εκτίμηση του χρόνου εκτέλεσης. Άρα ορισμένα μοντέλα χρησιμοποιούν το TPU για την αξιολόγηση τους (Ithemal, uiCA) και κάποια άλλα χρησιμοποιούν το TPL (IACA, OSACA). Η διαφοροποίηση αυτή δεν καθιστά δίκαιη τη σύγκριση των διαφόρων μοντέλων αφού βασίζονται σε διαφορετικό ορισμό της απόδοσης. Για να είναι πιο συνεπής και με περισσότερο νόημα η σύγκριση αυτή, έχει δημιουργηθεί μια παραλλαγή του BHive όπου τα benchmarks τελειώνουν με μια εντολή διακλάδωσης. Το νέο αυτό σύνολο δεδομένων ονομάζεται BHiveL ενώ τα αρχικά δεδομένα ονομάζονται πλέον BHiveU. [30],[31]

Κεφάλαιο 5 - Αναλυτική Παρουσίαση του Ithemal

Η πρόβλεψη των κύκλων ρολογιού που θα χρειαστεί ένας επεξεργαστής για να εκτελέσει ένα κομμάτι από εντολές assembly είναι μια μέτρηση σημαντική για προγραμματιστές - ερευνητές αλλά και για σχεδιαστές μεταγλωττιστών. Αυτή η μέτρηση μπορεί να προκύψει από την χρήση ενός αναλυτικού μοντέλου που θα δίνει τη μέτρηση αυτή. Το Ithemal αποτελεί ένα τέτοιο εργαλείο και μάλιστα είναι το πρώτο εργαλείο που κάνει αυτή την πρόβλεψη με χρήση τεχνικών μηχανικής μάθησης. Συγκεκριμένα το εργαλείο αυτό μαθαίνει να προβλέπει την απόδοση ενός συνόλου εντολών χρησιμοποιώντας μια προσέγγιση βασισμένη σε ένα LSTM-δίκτυο βασισμένο στα opcodes και στους τελεστές του κάθε τμήματος κώδικα.
Πρόκειται για ένα μοντέλο πολύ πιο ακριβές από τα πιο σύγχρονα μοντέλα (state-of-the-art models) και το οποίο είναι εφικτό να εφαρμοστεί σε πλήθος μικροεπεξεργαστών. [9],[10]

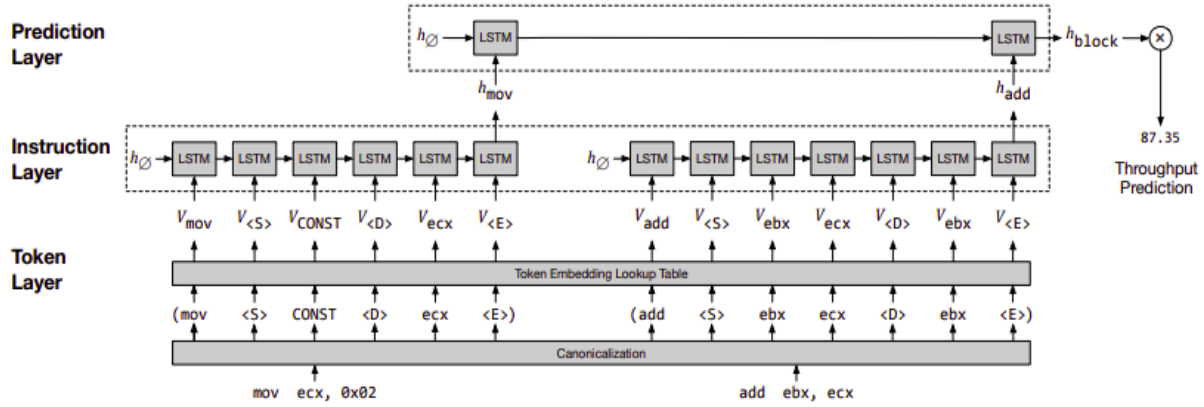
5.1 Εισαγωγή και Ανάγκες για την ανάπτυξη του Ithemal

Η απόδοση παριστάνει το πόσο γρήγορα οι εντολές επεξεργάζονται τα δεδομένα και είναι απαραίτητο ώστε να μπορεί αν γίνει εκτίμηση της απόδοσης του συστήματος. Η εκτίμηση αυτή είναι πολύ κοστοβόρα να γίνει με κανονική εκτέλεση του εκάστοτε κώδικα και έτσι πρέπει να υπάρχει ένα μοντέλο το οποίο θα κάνει εκτίμηση της τιμής αυτής. Το κάθε τέτοιο μοντέλο το οποίο και αναπτύσσεται πρέπει να έχει ορισμένες βασικές προδιαγραφές.
Αρχικά πρέπει να υπάρχει ακρίβεια (Accuracy). Πιο συγκεκριμένα οι σύγχρονοι επεξεργαστές έχουν πλήθος από βελτιστοποιήσεις υλικού (hardware optimizations) γεγονός που κάνει πιο δύσκολη την ανάπτυξη αναλυτικών μοντέλων. Οι επεξεργαστές “μεταφράζουν” (translate) κάθε εντολή από το ISA (Instruction Set Architecture) σε μικροεντολές, τις οποίες έπειτα εκτελούν. Όλο αυτό κάνει πιο δύσκολη την πρόβλεψη και απαιτεί μεγάλη ακρίβεια στην πρόβλεψη.
Κάθε νέο μοντέλο το οποίο αναπτύσσεται θα πρέπει να είναι συμβατό σε πλήθος συστημάτων (Portability) ώστε να βρίσκουν χρήση σε πολλές μικροαρχιτεκτονικές και να εξυπηρετούν περισσότερους χρήστες. Αναπτύσσοντας έναν εκτιμητή απόδοσης (throughput estimator) που θα υποστηρίζει πολλές αρχιτεκτονικές προφανώς απαιτεί επανεγγράψιμο των πινάκων εντολών, τις λίστες αξιοποίησης πόρων και τις βελτιστοποιήσεις κάθε μικροαρχιτεκτονικής, κάτι το οποίο είναι περίπλοκο λόγω των διαφόρων documentations. Αυτές οι αλλαγές - προσαρμογές πρέπει να γίνονται με όσο το δυνατόν μικρότερη ανθρώπινη παρέμβαση.
Τέλος, απαραίτητο χαρακτηριστικό κάθε μοντέλου είναι η ταχύτητα καθώς κάθε μεταγλωττιστής πρέπει να ψάχνει μεταξύ πλήθους τμημάτων κώδικα. [9],[10]

5.2 Αρχιτεκτονική Μοντέλου Ithemal

Στο παρακάτω σχήμα φαίνεται μια προσέγγιση του σχεδιασμού του Ithemal όπου φαίνεται ότι πρόκειται για ένα πρόβλημα παλινδρόμησης (regression problem). Συγκεκριμένα, δοθέντος ενός κώδικα assembly το μοντέλο κάνει πρόβλεψη της απόδοσης μιας ακολουθίας εντολών σαν έναν πραγματικό αριθμό. Στον πυρήνα του Ithemal υπάρχει ένα πολυεπίπεδο RNN δίκτυο που επεξεργάζεται κάθε εντολή του κώδικα και εξάγει ένα embedding το οποίο

χρησιμοποιείται έπειτα για την εύρεση της απόδοσης. Η ροή πληροφορίας μέσα στο μοντέλο του Ithemal ακολουθεί τα εξής 3 στάδια, Canonicalization, Embedding, Prediction. [9],[10]



Σχήμα 5.1: Αρχιτεκτονική μοντέλου Ithemal.

5.2.1 Στάδια Αρχιτεκτονικής Μοντέλου Ithemal

5.2.1.1 Στάδιο κανονικοποίησης (Canonicalization Stage)

Το πρώτο στάδιο είναι αυτό της κανονικοποίησης (Canonicalization Stage) όπου ο κώδικας assembly που δίνεται ως είσοδος μετατρέπεται σε μια πιο συγκροτημένη μορφή που προκύπτει με βάση τις εκάστοτε εντολές. Το Ithemal αυτό που κάνει είναι να παίρνει μια σειρά εντολών assembly, να τις κάνει disassembly και να τις χαρτογραφεί σε μια λίστα εντολών. Η κάθε εντολή αποτελείται από ένα πλήθος tokens τα οποία παριστάνουν το είδος της εντολής (operation code) και όλους τους απαραίτητους τελεστές (source και destination operands).
Πιο κατανοητή η μετατροπή μιας εντολής σε token μπορεί να γίνει με το παρακάτω παράδειγμα. Έστω η εντολή **mul ecx** με την οποία πολλαπλασιάζονται οι καταχωρητές **ecx** και **eax**, με το αποτέλεσμα να τοποθετείται στους καταχωρητές **edx** και **eax**. Το κανονικοποιημένο σετ από tokens που θα προκύψει από το μοντέλο του Ithemal θα είναι το εξής:
(mul, <S>, eax, ecx, <D>, edx, eax, <E>)
όπου τα tokens που βρίσκονται μέσα σε εισαγωγικά παριστάνουν τα “διαχωριστικά” μεταξύ opcode, source και destination operands. [9],[10]

5.2.1.2 Στάδιο Εμφύτευσης (Embedding Stage)

Στο δεύτερο στάδιο επεξεργασίας του Ithemal είναι το στάδιο εμφύτευσης (Embedding Stage). Σε αυτό το σημείο λαμβάνεται ως είσοδος η κανονικοποιημένη ακολουθία από tokens και για κάθε εντολή παράγεται ένα embedding, μια αναπαράσταση δηλαδή της εντολής σαν διάνυσμα (vector) σε ένα χώρο μεγάλων διαστάσεων (high-dimensional space). Το στάδιο αυτό αποτελείται από δύο υποστάδια, το token layer και το instruction layer. Στο token layer γίνεται η αντιστοίχιση κάθε token στην ακολουθία ενός n-διαστάσεων διανύσματος, δηλαδή του embedding. Στο δεύτερο υποστάδιο, το instruction layer γίνεται η αντιστοίχιση της ακολουθίας των token embeddings με ένα συνολικό embedding για τη συνολική εντολή. Δεδομένου ότι κάθε εντολή ενδέχεται να ποικίλει σε πλήθος τελεστών το μέγεθος της εισόδου είναι μεταβλητό κάθε φορά. [9],[10]

5.2.1.3 Στάδιο Πρόβλεψης (Prediction Stage)

Στο τρίτο στάδιο του Ithemal περιλαμβάνεται η διαδικασία της πρόβλεψης (Prediction Stage). Πρόκειται για το τελικό στάδιο όπου γίνεται η πρόβλεψη και στο σημείο αυτό αντιστοιχίζεται κάθε τμήμα κώδικα (που πλέον αντιστοιχεί σε μια ακολουθία από embeddings) με μια τιμή απόδοσης. Εδώ πάλι γίνεται χρήση ενός RNN δικτύου με LSTM κόμβους του οποίου τα βάρη προκύπτουν από το instruction layer. Η έξοδος του LSTM παριστάνεται με h_{block} και ο τελικός υπολογισμός είναι το $w*h_{block} + b$ όπου w είναι ένα διάνυσμα βαρών και b είναι το bias.
Ο κλιμακωτός συνδυασμός των RNN τόσο στο instruction όσο και το prediction layer έχει πολλά πλεονεκτήματα έναντι του μη ιεραρχικού. Αρχικά, η χρήση μνήμης και τα μονοπάτια οπισθοδρόμησης είναι σημαντικά μικρότερα σε αυτή την περίπτωση. Επίσης, οι εντολές ενσωματώνονται ατομικά (are embedded atomically) το οποίο σημαίνει ότι το σύστημα δεν υποχρεούται να παράγει εκτίμηση της απόδοσης μεταξύ των εκάστοτε εντολών. [9],[10]

5.3 Δεδομένα και εκπαίδευση

Τα δεδομένα που χρησιμοποιήθηκαν από αυτούς που αρχικά ανέπτυξαν και χρησιμοποίησαν το μοντέλο προέκυψαν από πλήθος εφαρμογών που φαίνονται στον παρακάτω πίνακα.

Benchmark suite	Description	#Total Blocks	#Unique Blocks
Linux Shared Libraries	linux loaders, standard library and other utilities	313846	103977
SPEC2006 (SPEC, 2006)	benchmark suite with compilers, chess engines, video compression and various simulation applications. Commonly used for benchmarking compilers	247047	141051
SPEC2017 (SPEC, 2017)	similar to SPEC2006, but with a larger variety	616899	234588
NAS (NASA, 1991-2014)	benchmarks with stencil computations(dense loops)	3935	1813
polybench-3.1(Pouchet,2012)	polyhedral compilation test suite(dense loops)	1900	859
TSVC(Maleki et al.,2011)	suite for testing compiler auto-vectorization	5129	2350
cortexsuite(Venkata et al.,2009)	computer vision workloads including neural networks	6582	3968
simd(Ihar et al.,2018)	heavily hand vectorized image processing library(exposes lot of SSE2,AVX,AVX2 variants)	212544	25462
compilers/interpreters	clang(Lattner & Adve,2004) and different versions of python (2.7, 3.5)	2746275	924663
end user applications	gimp filters, firefox, open-office, rhythmbox, etc	83555	35513
Full Dataset		4237712	1416473

Πίνακας 5.1: Εφαρμογές dataset εκπαίδευσης Ithemal.

Η εξαγωγή των εκάστοτε τμημάτων κώδικα κάθε εφαρμογής ακολουθεί την παρακάτω διαδικασία. Αρχικά γίνεται compile κάθε εφαρμογής με χρήση GCC 4.9.4 με χρήση -O3 optimization σε έναν επεξεργαστή. Στη συνέχεια με χρήση του εργαλείου Dynamorio (dynamic binary instrumentation tool) ώστε να σώσουν τα κωδικοποιημένα bytes των εκτελεσμένων x86-64 blocks. Τα benchmarks εκτελούνται με τις κατάλληλες προϋποθέσεις. Τέλος αφαιρούνται οι επαναλαμβανόμενοι κώδικες, αυτοί με την ίδια κωδικοποίηση δηλαδή.

Τα εργαλεία IACA και llvm-mca κάνουν μια σταθερή πρόβλεψη με δεδομένο ότι έχουμε L1 Cache hit και το περιβάλλον εκτέλεσης είναι non-preemptive. Στοχεύοντας στη συλλογή συμβατών με τα άλλα εργαλεία προβλέψεις χρησιμοποιείται ένας βρόχος εκτέλεσης του κώδικα 100 φορές (πλήθος επαναλήψεων που χρησιμοποιείται και από το llvm). Για την μέτρηση της τιμής γίνεται χρήση ενός timing script ανεπτυγμένο στα πρότυπα του Agnet Fog. Πρόκειται για ένα εγχειρίδιο που περιέχει όλες τις πληροφορίες σχετικά με τις διαδικασίες βελτιστοποίησης σε Intel και AMD μηχανήματα και συνήθως χρησιμοποιείται για την επιβεβαίωση τα throughput των εκάστοτε εντολών. Η επιπλέον προδιαγραφή του νέου κώδικα που αναπτύχθηκε βασιζόμενο στο Agnet Fog είναι η εξασφάλιση ότι σε όλες τις προσβάσεις μνήμης θα υπάρχει L1 Cache hit. [9],[10]

5.4 Αξιολόγηση Μοντέλου

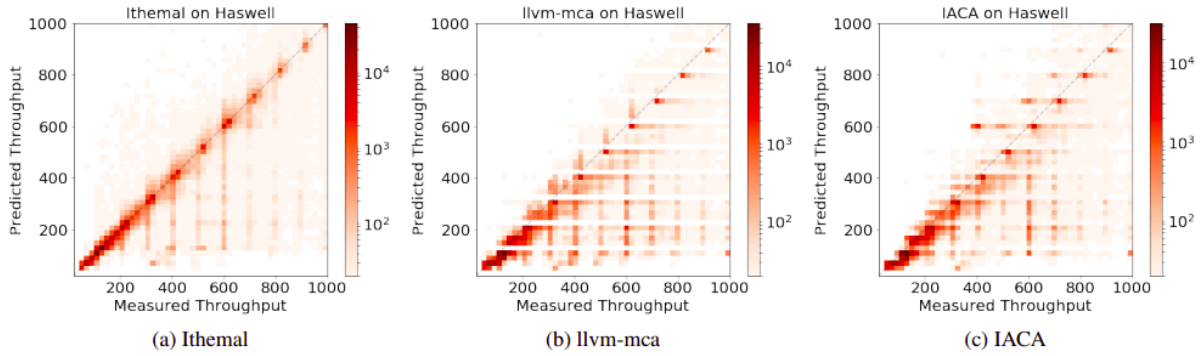
Η αρχική αξιολόγηση του μοντέλου έγινε συγκρίνοντάς το με τα εργαλεία IACA και llvm-mca τα οποία προσομοιώνουν όλα τα αξιώματα των σύγχρονων επεξεργαστών. Η αξιολόγηση έγινε με βάση τους τρεις άξονες στους οποίους πρέπει να βασίζονται τα σύγχρονα συστήματα, δηλαδή ακρίβεια, ταχύτητα και συμβατότητα.
Αρχικά η αξιολόγηση της ακρίβειας του Ithemal έναντι των άλλων μοντέλων έγινε σε Ivy Bridge, Haswell, και Skylake μικροαρχιτεκτονικές της Intel. Στα αποτελέσματα εκτός από το μέσο σφάλμα περιλαμβάνονται τα Spearman και Pearson correlations.

Microarchitecture	Method	Error	Spearman Correlation	Pearson Correlation
Ivy Bridge	llvm-mca **Ithemal**	0.181 **0.089**	0.902 **0.955**	0.777 **0.913**
Haswell	llvm-mca IACA **Ithemal**	0.200 0.209 **0.089**	0.890 0.917 **0.960**	0.790 0.833 **0.918**
Skylake	llvm-mca IACA **Ithemal**	0.239 0.167 **0.079**	0.852 0.926 **0.960**	0.729 0.835 **0.895**

Πίνακας 5.2: Μέσο σφάλμα κάθε μοντέλου στις διάφορες αρχιτεκτονικές.

Προκύπτει ότι το Ithemal έχει μεγαλύτερη ακρίβεια έναντι όλων των άλλων εργαλείων και στις τρεις μικροαρχιτεκτονικές. Επιπλέον το νέο αυτό μοντέλο έχει καλύτερη συσχέτιση σε σχέση με τα IACA και llvm. Δεδομένου ότι η Spearman correlation αντιστοιχεί σε υψηλότερη

χρησιμότητα μέσα σε έναν optimizing compiler φαίνεται καλύτερα το πλεονέκτημα του Ithemal. Αναλυτικότερα, τυπικά οι μεταγλωττιστές χρειάζεται να καθορίσουν ποιές από τις διάφορες παραμετροποιήσεις του κώδικα είναι η ταχύτερη και άρα δεν υπολογίζουν την ακριβή απόδοση κάθε κώδικα.



(a) Ithemal (b) llvm-mca (c) IACA

Σχήμα 5.2: Heatmaps για τις μετρήσεις και τις προβλέψεις των throughput στα διάφορα μοντέλα.

Στις παραπάνω μετρικές γίνεται η συσχέτιση των μετρήσεων και των προβλέψεων της απόδοσης. Από αυτές φαίνεται ότι τόσο σε εντολές με μικρή όσο και με μεγάλη απόδοση το Ithemal έχει καλύτερη ανταπόκριση σε σχέση με τα άλλα μοντέλα, τα οποία σε υψηλές αποδόσεις έχουν χειρότερα αποτελέσματα.

Σχετικά με την αξιολόγηση στον τομέα της ταχύτητας ο παρακάτω πίνακας δείχνει το πλήθος των εντολών που είναι ικανό να μετρηθεί κάθε δευτερόλεπτο από τον εκάστοτε εκτιμητή.

Method	Throughput(Instructions / Second)
llvm - mca	492
IACA	541
Ithemal	560
Empirical execution	13

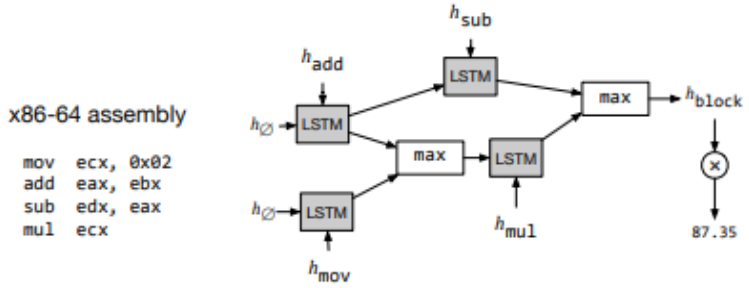
Πίνακας 5.3: Εκτίμηση απόδοσης για τους διάφορους estimators σε εντολές κάθε δευτερόλεπτο.

Ο υπολογισμός αυτός γίνεται μετρώντας το πλήθος των τμημάτων κώδικα που μπορεί να μετρήσει κάθε εργαλείο το δευτερόλεπτο πολλαπλασιάζοντάς το με το μέσο πλήθος εντολών ανά κώδικα. Με βάση τις παραπάνω τιμές το Ithemal και σε αυτόν τον τομέα είναι αποδοτικότερο, με τα IACA και llvm να επιτυγχάνουν ομοίως καλές τιμές και να είναι και τα τρία αυτά εργαλεία πολύ καλύτερα από την εμπειρική μέθοδο. Σημαντική λεπτομέρεια είναι επίσης το γεγονός ότι IACA και llvm προσφέρουν διαγνωστικές πληροφορίες σχετικά με τον

κώδικα καθώς και ότι η εμπειρική αξιολόγηση των δεδομένων μπορεί να επιταχυνθεί τρέχοντας λιγότερες επαναλήψεις των μετρήσεων.
Όσον αφορά τον τομέα της φορητότητας σε πολλές μικροαρχιτεκτονικές, η αρχική προσέγγιση είχε δοκιμαστεί και αξιολογηθεί σε Ivy Bridge, Haswell και Skylake. Άξιο αναφοράς είναι το γεγονός ότι το Ithemal δεν απαιτεί από το χρήστη να παρέχει πληροφορίες σχετικά με την μικροαρχιτεκτονική του επεξεργαστή σε σχέση με τα υπόλοιπα μοντέλα.
Στο επόμενο κεφάλαιο όπου θα αναλυθεί εκτενώς το πλήθος και το είδος των πειραμάτων θα φανεί και πάλι η σύγκριση των εκάστοτε μοντέλων στους τρεις τομείς που αναλύθηκαν και πιο πάνω, η ακρίβεια, η ταχύτητα και η συμβατότητα στις διάφορες αρχιτεκτονικές. [9],[10]

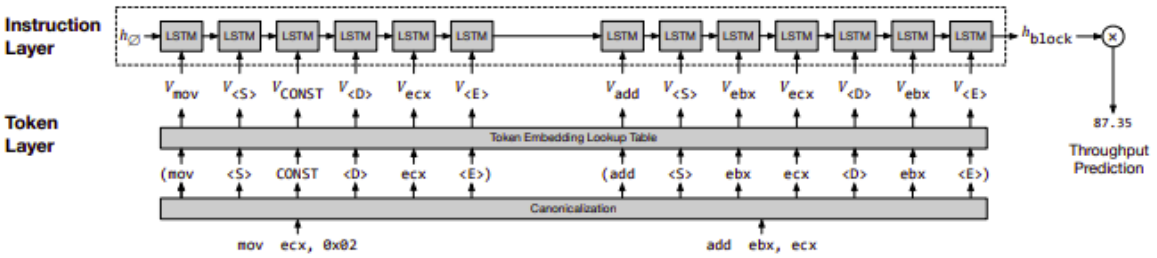
5.5 Αρχιτεκτονικές Νευρωνικού για το Ithemal

Έως ότου προκύψει η τελική μορφή της αρχιτεκτονικής του Ithemal δοκιμάστηκαν και αξιολογήθηκαν και άλλες αρχιτεκτονικές μέχρι να επιλεγεί η αποδοτικότερη. Ένα χαρακτηριστικό δίκτυο νευρωνικού που δοκιμάστηκε είναι το DAG-RNN δίκτυο που φαίνεται στο παρακάτω σχήμα.



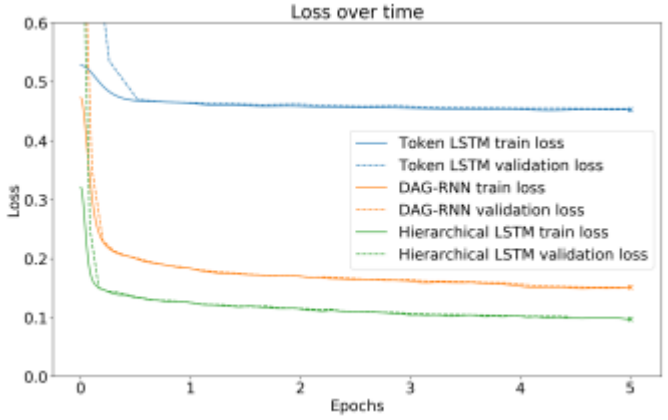
Σχήμα 5.3: Αρχιτεκτονική DAG-RNN δικτύου για throughput estimation.

Ουσιαστικά οι εντολές γίνονται embedded όπως στο Ithemal και άρα το token και το instruction layer παραμένει ίδιο. Αντί να τρέχει το RNN ακολουθιακά σε όλες τις εντολές στο prediction layer στην περίπτωση του DAG-RNN δικτύου δημιουργείται ένας εξαρτώμενος γράφος με κατευθυνόμενες ακμές μεταξύ δύο εντολών σε περίπτωση που μια από αυτές εξαρτάται από την δεύτερη. Δεδομένου ότι μια εντολή μπορεί να εξαρτάται από παραπάνω από μια εντολές, έχει προστεθεί ένας max κόμβος ώστε να μειωθούν οι καταστάσεις τροφοδότησης σε μια εντολή. Στην συνέχεια τοποθετείται ένας LSTM κόμβος που χρησιμοποιεί τα embeddings των εντολών σαν είσοδο καθώς και το αποτέλεσμα των max κόμβων. Με σκοπό τη γενίκευση της τελικής πρόβλεψης τοποθετείται ένας ακόμα max κόμβος με είσοδο όλες τις μη εξαρτώμενες πλέον εντολές (leaf instructions).
Ο DAG-RNN γράφος παραπάνω έχει εμπνευστεί από την θεωρητική συμπεριφορά ενός επεξεργαστή. Συγκεκριμένα η διαδικασία αυτή του επεξεργαστή βασίζεται στο ότι η απόδοση ενός τμήματος κώδικα ισούται με την απόδοση του μεγαλύτερου μονοπατιού που πρέπει σειριακά να εκτελεστεί από τον κώδικα.
Εκτός από το παραπάνω δίκτυο δοκιμάστηκε ένα token-level RNN δίκτυο με LSTM κόμβους με αρχιτεκτονική παρόμοια με αυτή του Ithemal αλλά χωρίς το prediction layer. Το μοντέλο αυτό ακολουθιακά προσθέτει όλα τα token χωρίς να κάνει διάκριση μεταξύ των εντολών. Το μοντέλο αυτό φαίνεται αναλυτικά στο παρακάτω σχήμα.



Σχήμα 5.4: Αρχιτεκτονική του Token RNN μοντέλου για throughput estimation.

Συγκρίνοντας αυτά τα διάφορα μοντέλα είναι εμφανές ότι το ιεραρχικό LSTM δίκτυο είναι το καλύτερο από τα τρία μοντέλα. Το ακολουθιακό LSTM έχει την χειρότερη ανταπόκριση από όλα λόγω της ανάγκης επεξεργασίας των tokens και των εντολών σε πολλαπλές κλίμακες. Το γεγονός ότι το DAG-RNN ανταποκρίνεται χειρότερα από το ιεραρχικό LSTM έγκειται στο γεγονός της σημασίας της σειράς των εντολών μέσα σε ένα block. [9],[10]



Σχήμα 5.5: Σύγκριση μοντέλων νευρωνικού για throughput estimation.

Κεφάλαιο 6 - Πειραματική Αξιολόγηση και Παρουσίαση Αποτελεσμάτων

Αντικείμενο του συγκεκριμένου κεφαλαίου είναι η παρουσίαση των αποτελεσμάτων των εκπαιδεύσεων και των αξιολογήσεων που εκτελέστηκαν στους διάφορους επεξεργαστές. Σε όλα τα σενάρια που εκτελέστηκαν οι προβλέψεις έγιναν σε επίπεδο τμήματος κώδικα (basic block) και όχι πλήρους κώδικα. Για την αξιολόγηση των διαφόρων αρχιτεκτονικών καθώς και το νόημα της μεταφορά των εκάστοτε μοντέλων σε άλλες γενιές επεξεργαστών και την εξαγωγή συμπερασμάτων, χρησιμοποιήθηκαν ορισμένες ευρέως χρησιμοποιούμενες μετρικές. Σε όλες τις αξιολογήσεις που έγιναν προηγήθηκε μια διαδικασία αφαίρεσης των παράταιρων τιμών (outliers) ώστε να είναι πιο αντιπροσωπευτικά τα αποτελέσματα.

6.1 Μετρικές αξιολόγησης αποτελεσμάτων

Οι μετρικές που αναλύονται παρακάτω χρησιμοποιήθηκαν με σκοπό την αξιολόγηση τόσο των διάφορων μοντέλων που εκπαιδεύτηκαν στα μηχανήματα αλλά και για τη μεταξύ τους σύγκριση. Για όλες τις μετρικές μπορούν να δημιουργηθούν πίνακες που θα δείχνουν τη σύγκριση τους με άλλες τιμές ώστε να εξαχθούν τα απαραίτητα συμπεράσματα. Οι υπολογισμοί μπορούν να γίνουν εύκολα με χρήση βιβλιοθηκών της Python όπου υπάρχουν έτοιμες συναρτήσεις που βρίσκουν τις συγκεκριμένες τιμές.[32]

6.1.1 Ακρίβεια (Accuracy)

Η πρώτη μετρική με βάση την οποία έγινε η αξιολόγηση των μοντέλων αλλά και της ορθότητας των μετρήσεων είναι αυτή της ακρίβειας (accuracy). Πρόκειται για την κύρια και συνηθέστερη μέθοδο αξιολόγησης των μοντέλων μηχανικής μάθησης. Το εύρος τιμών αυτής της μετρικής είναι μεταξύ του 0 και 1 και τα αποτελέσματα θεωρούνται τόσο καλύτερα όσο προσεγγίζουν την μονάδα. Η συγκεκριμένη τιμή εκφράζεται αλλιώς και ως συμπληρωματική του σφάλματος.

6.1.2 Γεωμετρικός μέσος (Geometric mean)

Ουσιαστικά με τη συγκεκριμένη μετρική συγκρίνουμε δύο τιμές. Εδώ οι τιμές αυτές είναι η πραγματική τιμή και η πρόβλεψη από το εκάστοτε μοντέλο. Το πηλίκο που προκύπτει από τη διαίρεση των δύο τιμών πρέπει να κινείται όσο το δυνατόν πιο κοντά στη μονάδα, γεγονός που δείχνει την εγγύτητα των δύο τιμών. Μπορεί επίσης να εξαχθεί ένα διάγραμμα με το σύνολο της συγκεκριμένης μετρικής για όλα τα ζεύγη τιμών που σε μια καλή περίπτωση τιμών θα πρόκειται για μια καμπύλη (καμπάνα) γύρω από το 1.

6.1.3 Kendall's Tau

Ένας ακόμα συνήθης τελεστής συσχέτισης είναι ο Kendall's Tau γνωστός και ως Kendall's τ coefficient από το ελληνικό γράμμα τ. Χρησιμοποιείται για να μετρήσει τη συσχέτιση μεταξύ δύο ποσοτήτων. Έστω ότι έχουμε ένα σετ παρατηρήσεων: (x_1, y_1), (x_2, y_2),..., (x_n, y_n). Θεωρείται ότι είναι αρμονικά (concordant) αν ισχύει $x_i < x_j$ και $y_i < y_j$ ή $x_i > x_j$ και

$y_i > y_j$ ενώ μη αρμονικά (discordant) αν ισχύει άλλη διάταξη. Με βάση τα παραπάνω το Kendall's τ coefficient ορίζεται ως:

$$\tau = \frac{(number\ of\ concordant\ pairs) - (number\ of\ discordant\ pairs)}{(number\ of\ pairs)} =$$

$$= 1 - \frac{2*(number\ of\ discordant\ pairs)}{\left(\frac{n}{2}\right)} \quad \text{όπου } \left(\frac{n}{2}\right) = \frac{n(n-1)}{2}$$

Όσο πιο κοντά στο 1 είναι η τιμή αυτή τόσο πιο κοντινές είναι οι ποσότητες που συγκρίνονται.[36],[37]

6.1.4 Spearman's Rank Correlation

Όπως και άλλοι συντελεστές συσχέτισης, αυτός κυμαίνεται μεταξύ -1 και +1 με το 0 να υποδηλώνει καμία συσχέτιση. Οι συσχετίσεις -1 ή +1 υποδηλώνουν μια ακριβή μονοτονική σχέση. Οι θετικές συσχετίσεις υποδηλώνουν ότι όσο αυξάνεται το x, αυξάνεται και το y. Οι αρνητικές συσχετίσεις υποδηλώνουν ότι καθώς το x αυξάνεται, το y μειώνεται.
Δοθέντος δύο συνόλων X και Y η συγκεκριμένη συσχέτιση αντί να χρησιμοποιεί τα κανονικά δεδομένα (όπως στην συνήθη συσχέτιση) χρησιμοποιεί την κατάταξη κάθε στοιχείου μέσα στο σύνολο. Η μετρική αυτή δίνεται από τον τύπο:

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2] * [n\sum y^2 - (\sum y)^2]}}$$

Η κατάταξη προκύπτει με την παρακάτω μέθοδο. Έχοντας έναν πίνακα N στοιχείων(με επιτρεπόμενα διπλότυπα) όλα τα στοιχεία ταξινομούνται σε αύξουσα σειρά. Εάν υπάρχουν x επαναλαμβανόμενα στοιχεία μιας συγκεκριμένης τιμής, τότε σε κάθε στοιχείο θα πρέπει να εκχωρηθεί μια κατάταξη ίση με τον αριθμητικό μέσο όρο των x διαδοχικών βαθμών.[38],[39]

6.1.5 Μέση απόλυτη διαφορά (MAD)

Η μέση απόλυτη απόκλιση είναι μια μετρική της διασποράς, δηλαδή παρουσιάζει τη διάσπαση του συνόλου δεδομένων. Είναι πολύ χρήσιμη αφού είναι λιγότερο ευαίσθητη στις παράταιρες τιμές σε σχέση με άλλες όπως η διασπορά (variance). Ο υπολογισμός γίνεται με τον παρακάτω τύπο.

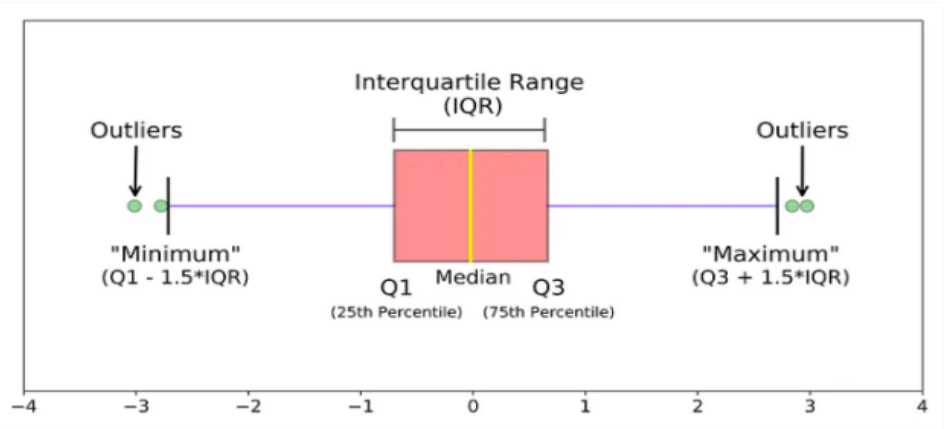
$$MAD = median(|X_i - \underline{X}|)$$

Όσο μικρότερη είναι η συγκεκριμένη τιμή τόσο πιο μικρή είναι η διασπορά των δεδομένων ή του εκάστοτε συνόλου τιμών-δεδομένων που θέλουμε να εξάγουμε πληροφορίες.[33]

6.2 Διαδικασία εύρεσης Παράταιρων Τιμών (Outliers)

Όλες τις μετρήσεις που έγιναν περιείχαν τιμές οι οποίες απείχαν σε μεγάλο βαθμό από τις πραγματικές τιμές. Οι τιμές αυτές ονομάζονται παράταιρες (outliers) και η χρήση τους στην εξαγωγή μετρικών και κατ' επέκταση συμπερασμάτων πρέπει να αποφεύγεται ώστε τα αποτελέσματα να είναι πιο αντιπροσωπευτικά. Υπάρχουν πολλοί και διαδεδομένοι αλγόριθμοι που βοηθούν στην εύρεση των τιμών αυτών και εδώ έγινε χρήση του IQR Algorithm. Η μέθοδος αυτή διαχωρίζει τα δεδομένα σε τεταρτημόρια. Υπολογίζουμε το

λεγόμενο διατεταρτημόριο εύρος (interquartile range - IQR) και προσδιορίζουμε τα δεδομένα που βρίσκονται εκτός ενός άνω και κάτω εύρους. Τα όρια αυτά υπολογίζονται ως εξής:
Κατώτερο όριο εύρους = Q1 – (1,5* IQR)
Ανώτερο όριο εύρους = Q3 + (1,5*IQR) [35]



Σχήμα 6.1: Διαδικασία αφαίρεσης Outliers.

6.3 Περιγραφή πειραματικής διαδικασίας

6.3.1 Παρουσίαση επεξεργαστών και εργαλείων μέτρησης

6.3.1.2 Πλατφόρμες - Επεξεργαστές Μελέτης

Παρακάτω παρουσιάζονται τα χαρακτηριστικά όλων των συστημάτων που χρησιμοποιήθηκαν καθ' όλη τη διάρκεια της εργασίας αυτής ώστε να αναπτυχθούν τα διάφορα μοντέλα. Συγκεκριμένα καταγράφονται το πλήρες όνομα του μοντέλου και η μικροαρχιτεκτονική.

System	Model name	Microarchitecture
Haswell	Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz	Haswell
Broadwell	Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz	Haswell
Skylake	Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz	Skylake
Icelake	Intel(R) Xeon(R) Silver 4314 CPU @ 2.40GHz	Sunny Cove
AMD	AMD EPYC 7402 24-Core Processor	Zen 2

Πίνακας 6.1: Χαρακτηριστικά συστημάτων που χρησιμοποιήθηκαν.

Από τα παραπάνω στοιχεία φαίνεται ότι Haswell, Broadwell και Skylake θα έχουν λιγότερες διαφορές σε σχέση με τον IceLake και την AMD. Συγκεκριμένα, μεταξύ Haswell και Broadwell υπάρχει αλλαγή τεχνολογίας ενώ μεταξύ Broadwell και Skylake υπάρχει και αλλαγή αρχιτεκτονικής.

Όσο κοντινότερες είναι οι γενιές επεξεργαστών, τα μοντέλα τα οποία θα προκύψουν αναμένεται να είναι παρόμοια και η μεταφορά μεταξύ αυτών των συστημάτων θα είναι πιο εύκολη και με λιγότερη επιβάρυνση.[41],[42],[43],[44],[45],[46]

6.3.1.2 Παρουσίαση εργαλείων μέτρησης

Για την εκπαίδευση του εκάστοτε μοντέλου του Ithemal απαιτείται για κάθε τμήμα κώδικα να υπάρχει μια τιμή που παριστάνει τον απαιτούμενο χρόνο για να εκτελεστεί ο συγκεκριμένος κώδικας. Για το σκοπό αυτό υπάρχει αρχικά το timing-harness εργαλείο που αναπτύχθηκε από τους δημιουργούς του Ithemal με πολύ ακριβείς και στοχευμένες μετρήσεις. Το εργαλείο αυτό δοκιμάστηκε για Intel επεξεργαστές και δοκιμάστηκε συγκεκριμένα σε Ivy Bridge, Haswell και Skylake γενιές. Στα πλαίσια της συγκεκριμένης εργασίας εγκαταστάθηκε και δοκιμάστηκε το εργαλείο αυτό σε Haswell, Broadwell, Skylake και Ice Lake επεξεργαστές. Ένα ακόμα εργαλείο που χρησιμοποιείται για την μέτρηση αυτή του πραγματικού χρόνου εκτέλεσης του εκάστοτε κώδικα είναι το nanoBench. Πρόκειται για ένα εργαλείο που αναπτύχθηκε από τους δημιουργούς του uiCA και έχει δοκιμαστεί τόσο σε Intel όσο και σε AMD επεξεργαστές. Το συγκεκριμένο εργαλείο δοκιμάστηκε αρχικά στο AMD μηχάνημα χρησιμοποιώντας το κατάλληλο αρχείο παραμετροποίησης (configuration file). Εν συνεχεία, έγινε δοκιμή του σε Broadwell, Skylake και Ice lake επεξεργαστές της Intel. Πρόκειται για ένα εξίσου αξιόπιστο εργαλείο μέτρησης με το επιπλέον πλεονέκτημα της συμβατότητας και σε AMD επεξεργαστές. Συλλέγοντας όλες τις παραπάνω μετρήσεις στους παρακάτω πίνακες βλέπουμε τις διάφορες μετρικές που συγκρίνουν τις μετρήσεις μεταξύ των διαφόρων αυτών επεξεργαστών, αλλά και των δύο διαφορετικών εργαλείων.
Παρακάτω φαίνονται ο γεωμετρικός μέσος και η απόκλιση μεταξύ των μετρήσεων για τα δύο εργαλεία μέτρησης.

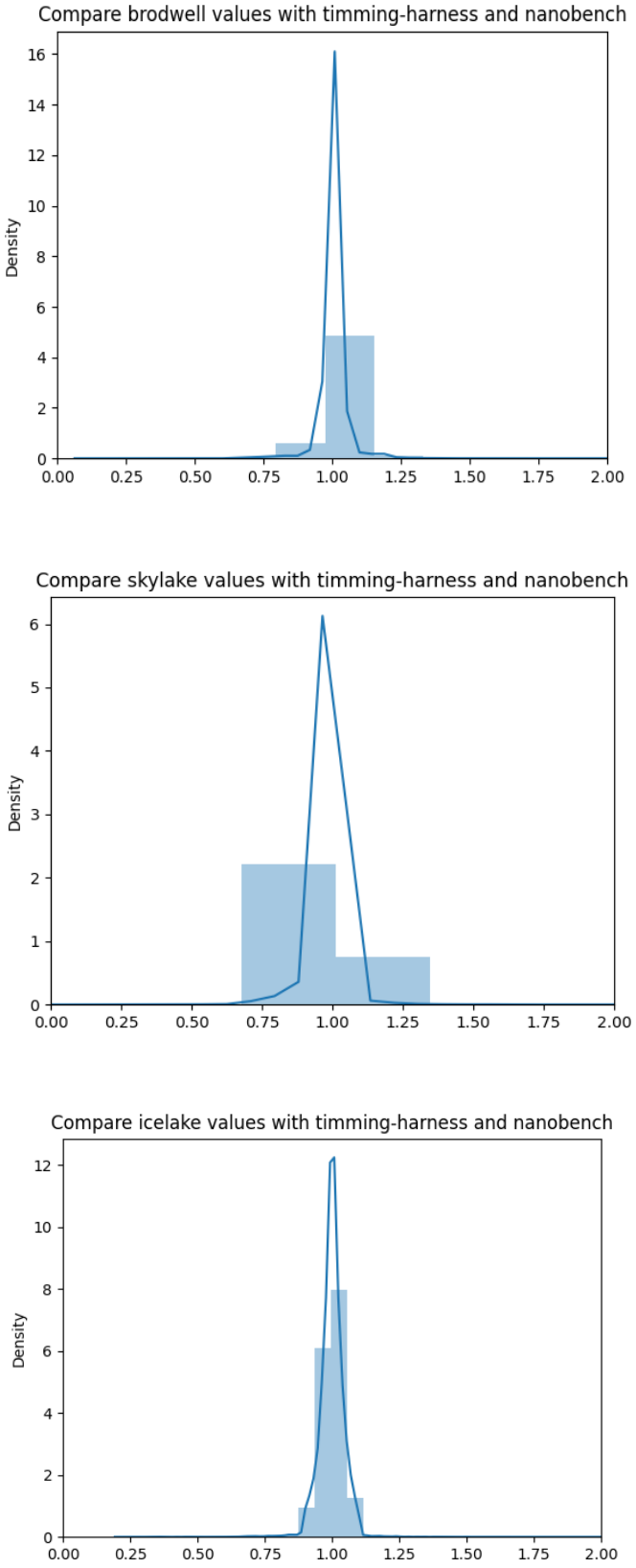
Απόκλιση Μετρήσεων(%)	Broadwell(Nano)	Skylake(Nano)	Icelake(Nano)
Broadwell	5.4	-	-
Skylake	-	6.9	-
Icelake	-	-	5.1

Πίνακας 6.2: Απόκλιση μετρήσεων των δύο εργαλείων.

Geometric Mean	Broadwell(Nano)	Skylake(Nano)	Icelake(Nano)
Broadwell	1.003	-	-
Skylake	-	0.992	-
Icelake	-	-	0.997

Πίνακας 6.3: Γεωμετρικός Μέσος σύγκρισης των δύο εργαλείων.

Από τις παραπάνω τιμές φαίνεται ότι οι μετρήσεις των δύο εργαλείων είναι κοντά και κατ' επέκταση θα οδηγήσουν και στην δημιουργία κοντινών και παρόμοιων μοντέλων. Παρακάτω φαίνονται ακόμα τα ιστογράμματα σύγκρισης των δύο εργαλείων επιβεβαιώνοντας και αυτά τις κοντινές τιμές με μια καμπύλη κοντά στη μονάδα.



Σχήμα 6.2: Ιστογράμματα σύγκρισης των δύο εργαλείων μέτρησης

Παρακάτω φαίνεται ακόμα η απόκλιση των μετρήσεων και ο γεωμετρικός μέσος μεταξύ των διαφορετικών επεξεργαστών και εργαλείων. Μπορούμε να συμπεράνουμε ότι μεγαλύτερες διαφορές μεταξύ των συγκρινόμενων επεξεργαστών οδηγεί σε μεγαλύτερη απόκλιση των συγκρινόμενων τιμών.

Απόκλιση Μετρήσεων(%) (timing-harness)	Haswell	Broadwell	Skylake	Icelake
Haswell	0	3.9	7.8	12.4
Broadwell	4	0	8.9	13.2
Skylake	7.3	8.2	0	13.2
Icelake	9.2	10.1	10.1	0

Πίνακας 6.4: Πίνακας απόκλισης μετρήσεων για τους διάφορους επεξεργαστές με τις timing-harness μετρήσεις.

Απόκλιση Μετρήσεων(%) (nanoBench)	Broadwell	Skylake	Icelake	AMD
Broadwell	0	2.4	8.6	25.2
Skylake	2.1	0	7.5	25.2
Icelake	5.8	4.3	0	23.1
AMD	18.3	16.6	20.1	0

Πίνακας 6.5: Πίνακας απόκλισης μετρήσεων για τους διάφορους επεξεργαστές με τις nanoBench μετρήσεις.

Geometric Mean (timing-harness)	Haswell	Broadwell	Skylake	Icelake
Haswell	1	0.998	1.013	1.025
Broadwell	1.0009	1	1.014	1.033
Skylake	0.985	0.983	1	1.002
Icelake	0.971	0.953	0.995	1

Πίνακας 6.6: Πίνακας γεωμετρικού μέσου μετρήσεων για τους διάφορους επεξεργαστές για τις timing-harness μετρήσεις.

Geometric Mean (nanoBench)	Broadwell	Skylake	Icelake	AMD
Broadwell	1	1.005	1.017	1.075
Skylake	0.994	1	1.013	1.075
Icelake	0.982	0.986	1	1.039
AMD	0.947	0.945	1.03	1

Πίνακας 6.7: Πίνακας γεωμετρικού μέσου μετρήσεων για τους διάφορους επεξεργαστές με τις nanoBnech μετρήσεις.

Παρατηρώντας τους παραπάνω πίνακες που παρουσιάζουν τις αποκλίσεις αλλά και τον γεωμετρικό μέσο συγκρίνοντας τις μετρήσεις και με τα δύο εργαλεία, μπορούμε να εξάγουμε ορισμένα συμπεράσματα. Αρχικά, από όλες τις μετρικές επιβεβαιώνεται ότι όσο πιο κοντινές είναι οι γενιές (της Intel) τόσο πιο κοντινές είναι και οι μετρήσεις και τόσο πιο παρόμοια η συμπεριφορά των συστημάτων. Συγκεκριμένα βλέπουμε ότι οι τιμές σε Haswell, Broadwell και Skylake είναι πιο κοντά σε σχέση με τον Ice Lake. Η αύξηση των διαφορών μεταξύ μη διαδοχικών και όμοιων γενεών οδηγεί σε μεγαλύτερες αποκλίσεις μεταξύ των μετρήσεων. Επίσης είναι φανερό, όπως άλλωστε θα περίμενε κανείς, ότι οι διαφορές είναι μεγαλύτερες συγκρίνοντας τις τιμές της Intel με αυτές της AMD μηχανής. Το γεγονός αυτό οφείλεται στις μεγαλύτερες διαφοροποιήσεις που παρουσιάζονται μεταξύ των μηχανών αυτών. Το μοτίβο αυτό ισχύει τόσο με το harness όσο και με το nanoBench εργαλείο συλλογής μετρήσεων.

6.3.2 Μελέτη επίδοσης Ithemal

Για τη δημιουργία και την ανάπτυξη ενός μοντέλου βασικό στάδιο είναι η διαδικασία της εκπαίδευσης. Πρόκειται για το στάδιο στο οποίο το εκάστοτε μοντέλο μαθαίνει να προβλέπει κάποια τιμή ανάλογα με την είσοδο που του δίνεται. Τα δεδομένα χωρίζονται σε δεδομένα εκπαίδευσης (80%), επαλήθευσης (10%) και αξιολόγησης (10%). Η εκπαίδευση γίνεται με τα αντίστοιχα δεδομένα (train). Έπειτα γίνεται η διαδικασία επαλήθευσης του μοντέλου (validation) και τέλος με τα δεδομένα αξιολόγησης (test) βγαίνουν τα αποτελέσματα ακρίβειας για το εκπαιδευμένο μοντέλο. Πλέον σε ότι νέα δεδομένα βλέπει το μοντέλο μπορεί και κάνει μια πρόβλεψη συγκρίνοντάς τα με αυτά τα οποία ξέρει την έξοδό τους από την εκπαίδευση.

6.3.2.1 Διαδικασία εκπαίδευσης

Αρχικά, θα πρέπει να γίνει η κατάλληλη επεξεργασία και διαμόρφωση των δεδομένων. Πιο συγκεκριμένα για κάθε block κώδικα θα πρέπει να δημιουργείται μια πλειάδα (tuple) με τη εξής μορφή:

(code_id, timing, code_intel, code_xml).

Το code_id είναι ένα μοναδικό αναγνωριστικό για κάθε κομμάτι κώδικα, που στην συγκεκριμένη περίπτωση είναι η δεκαεξαδική αναπαράσταση του. Το timing είναι η τιμή της

πραγματικής μέτρησης για το χρόνο εκτέλεσης που έχει προέλθει από τα εργαλεία που αναφέρθηκαν παραπάνω (timing-harness και nano Bench). Το code_intel αποτελεί μια λεκτική τιμή (string) που χρησιμοποιείται κυρίως για debugging και στην συγκεκριμένη περίπτωση ισούται με “None” σε αυτή την περίπτωση. Τέλος, το code_xml είναι η έξοδος ενός tokenizer που αναπτύχθηκε με σκοπό την δημιουργία της XML αναπαράστασης για κάθε κομμάτι κώδικα. Με χρήση τεχνικών pytorch γίνεται η δημιουργία και η αποθήκευση ενός τένσορα (tensor) με τα συνολικά δεδομένα.
Για την κύρια διαδικασία της εκπαίδευσης υπάρχει το αρχείο run_ithemal.py στο οποίο δίνονται οι απαραίτητες παράμετροι ώστε να ολοκληρωθεί η διαδικασία. Δίνονται ως παράμετροι τα δεδομένα που δημιουργήθηκαν με τον τρόπο που αναλύθηκε παραπάνω καθώς και τιμές που αφορούν τα ταυτόχρονα νήματα (threads), τους εκπαιδευτές (trainers) και το πλήθος των εποχών. Καθ’όλη τη διάρκεια της εκπαίδευσης φαίνεται το σφάλμα και το πόσο γρήγορα αυτό μειώνεται με το πέρας των εποχών. Μετά την ολοκλήρωση της εκπαίδευσης παράγονται κάποια αρχεία με τα αποτελέσματα. Συγκεκριμένα, το loss_report.log αρχείο που περιέχει μια λίστα από την εξέλιξη του σφάλματος στο πέρασμα του χρόνου και των εποχών μαζί με τους παράλληλους εκπαιδευτές. Επιπλέον παράγεται ένα αρχείο με το όνομα validation_results.txt που περιέχει τα δεδομένα επαλήθευσης με την παράθεση της πραγματικής τιμής μαζί με την πρόβλεψη. Οι μετρικές που χρησιμοποιούνται με αυτά τα αρχεία είναι και πάλι αυτές που αναφέρθηκαν πιο πάνω καθώς και κάποια διαγράμματα που φαίνεται η ταχύτητα σύγκλισης στο τελικό σφάλμα.

6.3.2.2 Αποτελέσματα Επίδοσης Ithemal

Χρησιμοποιώντας τα παραπάνω αρχεία για κάθε ένα από τα μοντέλα που εκπαιδεύτηκαν εξάγονται εκτός από το μέσο σφάλμα, οι μετρικές συσχέτισης (Kendall's Tau και Spearman Correlation) για τις πραγματικές τιμές και τις προβλέψεις. Για τις ίδιες τιμές εξάγεται η Μέση Απόλυτη Απόκλιση και ο γεωμετρικός μέσος ώστε να βγουν συμπεράσματα για τα δεδομένα αυτά.

timing-harness	Error (%)	Geometric Mean	MAD	Kendall's Tau	Spearman Correlation
Haswell	7.7	0.962	0.017	0.807	0.906
Broadwell	6.5	0.969	0.01	0.852	0.927
Skylake	10	0.954	0.026	0.77	0.874
Icelake	10	0.956	0.04	0.79	0.889

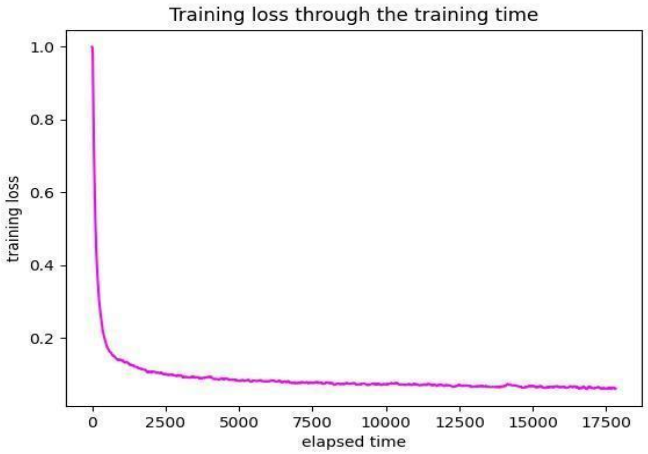
Πίνακας 6.8: Πίνακας μετρικών για τα διάφορα μοντέλα στους διάφορους επεξεργαστές με τις timing-harness μετρήσεις.

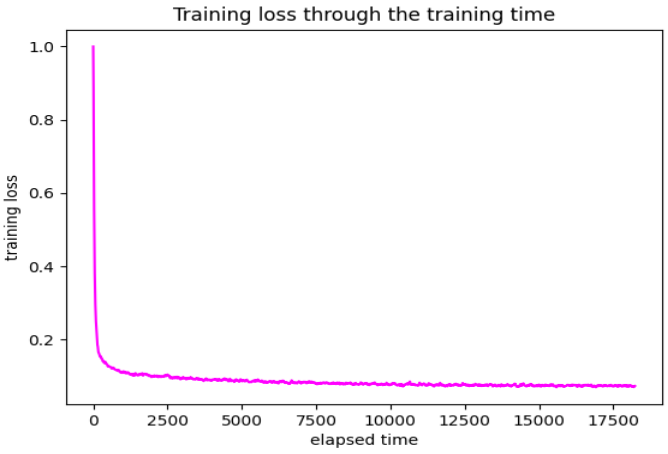
nanoBench	Error (%)	Geometric Mean	MAD	Kendall's Tau	Spearman Correlation
Broadwell	6.5	0.963	0.009	0.849	0.92
Skylake	5.7	0.974	0.002	0.862	0.924
Icelake	7	0.969	0.01	0.85	0.917
AMD	9.6	0.964	0.019	0.828	0.90

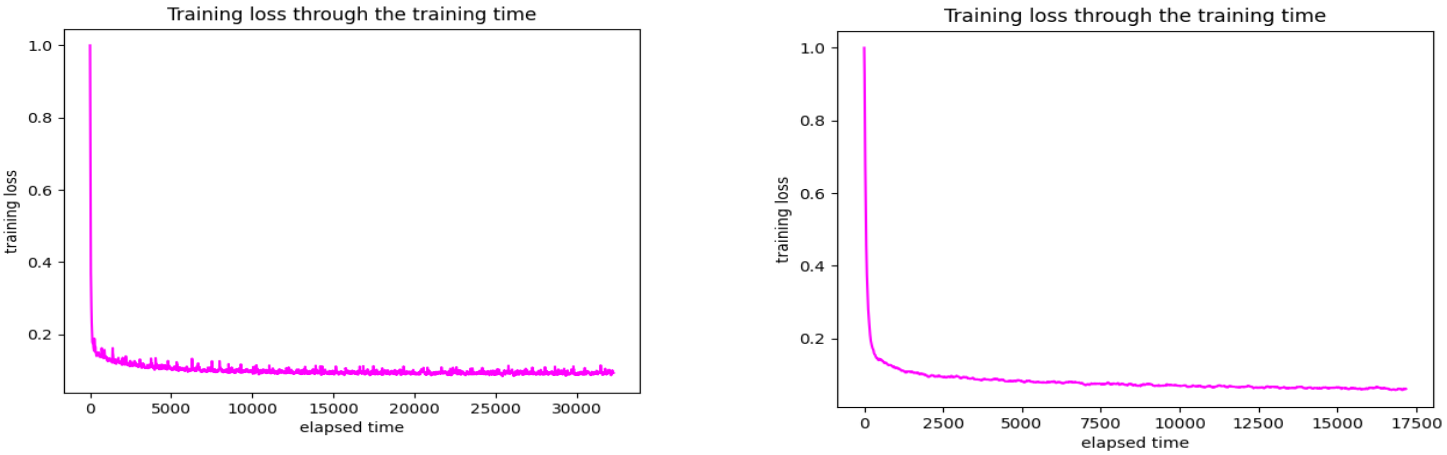
Πίνακας 6.9: Πίνακας μετρικών για τα διάφορα μοντέλα στους διάφορους επεξεργαστές με τις nanoBench μετρήσεις.

Παρατηρώντας τις παραπάνω τιμές είναι εμφανές ότι σε όλα τα συστήματα που έγιναν εκπαιδεύσεις τα αποτελέσματα είναι πολύ ικανοποιητικά αφού το σφάλμα (error) είναι αρκετά μικρό (κάτω από 10%) αλλά και η μέση γεωμετρική τιμή είναι πολύ κοντά στο 1. Εμφανές είναι ακόμα ότι συγκρίνοντας τα δύο εργαλεία μέτρησης στον Broadwell δεν παρατηρούνται διαφορές ενώ σε Skylake και Ice lake τα αποτελέσματα είναι λίγο καλύτερα με τις τιμές του nano Bench. Πολύ ικανοποιητικά είναι τα αποτελέσματα και στο μηχάνημα AMD.
Τα αποτελέσματα αυτά επιβεβαιώνονται και από τη συσχέτιση μεταξύ πραγματικών τιμών και προβλέψεων αλλά και του Kendall's Tau.

Παρακάτω φαίνονται ενδεικτικά για κάποιους από τους επεξεργαστές η εξέλιξη του σφάλματος με το πέρασμα του χρόνου (εποχών).







Σχήμα 6.3: Εξέλιξη σφάλματος με το πέρασμα του χρόνου(εποχών).

Παρατηρώντας τις παραπάνω γραφικές βλέπουμε ότι για όλα τα μοντέλα το σφάλμα κινείται σε μικρά επίπεδα και μάλιστα συγκλίνει αρκετά νωρίς. Συγκεκριμένα ανεξάρτητα από τις συνολικές εποχές που θα επιλεγούν για εκπαίδευση, μετά τη δεύτερη περίοδο το σφάλμα έχει σταθεροποιηθεί και κινείται κοντά στην τελική του τιμή.

6.3.3 Αξιολόγηση επίδοσης ενός επεξεργαστή με προεκπαιδευμένο μοντέλο

6.3.3.1 Περιγραφή διαδικασίας αξιολόγησης της επίδοσης ενός επεξεργαστή με μοντέλο ενός άλλου

Μια ακόμα ενδιαφέρουσα πειραματική διαδικασία που εκτελέστηκε ήταν αυτή κατά την οποία ελέγχθηκε η δυνατότητα μεταφοράς ενός προεκπαιδευμένου μοντέλου σε άλλο επεξεργαστή διαφορετικής γενιάς ώστε να αποφευχθεί η επανάληψη της διαδικασίας εκπαίδευσης. Πρόκειται για ένα ενδιαφέρον πείραμα αφού θα είχε τεράστια αξία για τους προγραμματιστές και τους σχεδιαστές να υπάρχει ένα μοντέλο που θα έχει εξίσου καλά αποτελέσματα σε διαφορετικά συστήματα χωρίς να απαιτείται επανεκπαίδευση. Έτσι θα εξοικονομούσαν χρόνο αλλά και υπολογιστικούς πόρους. Η διαδικασία που ακολουθήθηκε ήταν να συγκριθούν οι προβλέψεις του εκάστοτε μοντέλου με τις μετρήσεις από τα εργαλεία timing-harness και nano Bench και να συγκριθεί το σφάλμα κάθε φορά με το σφάλμα που θα προκύπτει από την εκπαίδευση ενός νέου μοντέλου. Οι ομοιότητες των διαφόρων γενεών που αναφέρθηκαν παραπάνω αναμένεται να οδηγήσουν και στην ευκολότερη μεταφορά μεταξύ των γενεών αυτών (Haswell, Broadwell, Skylake).

6.3.3.2 Παρουσίαση αποτελεσμάτων και εξαγωγή συμπερασμάτων

Στους παρακάτω πίνακες φαίνονται τόσο το μέσο σφάλμα για τα νέα μοντέλα που μεταφέρθηκαν στα διάφορα συστήματα καθώς και οι μετρικές συσχέτισης (Kendall's Tau και Spearman Correlation) για τις πραγματικές τιμές και τις προβλέψεις από τη μεταφορά των μοντέλων.

(timing-harness)	Haswell	Broadwell	Skylake	Icelake
Haswell Απόκλιση Μετρήσεων Error Μεταφοράς	0 **0**	3.9 **9.9**	7.8 **12.7**	12.4 **16.9**
Broadwell Απόκλιση Μετρήσεων **Error Μεταφοράς**	4 **10.6**	0 **0**	8.9 **14.2**	13.2 **17.4**
Skylake Απόκλιση Μετρήσεων **Error Μεταφοράς**	7.3 **11.6**	8.2 **12.2**	0 **0**	13.2 **15.7**
Icelake Απόκλιση Μετρήσεων **Error Μεταφοράς**	9.2 **14.1**	10.1 **15.1**	10.1 **14.9**	0 **0**

Πίνακας 6.10: Πίνακας σφάλματος για τη μεταφορά εκπαιδευμένων μοντέλων με timing-harness μετρήσεις.

(nanoBench)	Broadwell	Skylake	Icelake	AMD
Broadwell Απόκλιση Μετρήσεων **Error Μεταφοράς**	0 **0**	2.4 **7.9**	8.6 **13.8**	25.2 **18.8**
Skylake Απόκλιση Μετρήσεων **Error Μεταφοράς**	2.1 **8.3**	0 **0**	7.5 **12.7**	25.2 **17.7**
Icelake Απόκλιση Μετρήσεων **Error Μεταφοράς**	5.8 **12.3**	4.3 **11.7**	0 **0**	23.1 **19.3**
AMD Απόκλιση Μετρήσεων **Error Μεταφοράς**	18.3 **14.5**	16.6 **13.7**	20.1 **16.1**	0 **0**

Πίνακας 6.11: Πίνακας σφάλματος για τη μεταφορά εκπαιδευμένων μοντέλων με nanoBench μετρήσεις.

Spearman Correlation (timing-harness)	Broadwell	Haswell	Skylake	Icelake
Broadwell	1	0.886	0.851	0.86
Haswell	0.89	1	0.86	0.868
Skylake	0.884	0.89	1	0.872
Icelake	0.866	0.871	0.84	1

Πίνακας 6.12: Πίνακας Spearman Correlation για τη μεταφορά εκπαιδευμένων μοντέλων με timing-harness μετρήσεις.

Spearman Correlation (nanoBench)	Broadwell	Skylake	Icelake	AMD
Broadwell	1	0.909	0.886	0.897
Skylake	0.92	1	0.89	0.902
Icelake	0.889	0.869	1	0.868
AMD	0.891	0.88	0.864	1

Πίνακας 6.13: Πίνακας Spearman Correlation για τη μεταφορά εκπαιδευμένων μοντέλων με nanoBench μετρήσεις.

Παρατηρώντας τους παραπάνω πίνακες βλέπουμε όπως ήταν αναμενόμενο αύξηση του σφάλματος πάνω από 10% στις περισσότερες περιπτώσεις. Όπως αναμενόταν, μεταξύ κοντινότερων γενιών η μεταφορά οδηγεί σε μικρότερη αύξηση του σφάλματος αφού οι προβλέψεις είναι πιο κοντά σε σχέση με τις μετρήσεις και οι διαφορές των επεξεργαστών μικρότερες. Σε γενιές με μεγαλύτερη απόσταση το σφάλμα μεγαλώνει κι άλλο. Ιδιαίτερο ενδιαφέρον έχουν τα αποτελέσματα από τη μεταφορά του AMD μοντέλου σε Intel αρχιτεκτονικές αλλά και η αντίθετη διαδρομή. Η αύξηση του σφάλματος είναι ανάλογη της απόστασης των μετρήσεων που αναλύθηκαν παραπάνω. Μεγαλύτερη απόσταση στις μετρήσεις οδηγεί και σε μεγαλύτερο σφάλμα κατά την μεταφορά του προεκπαιδευμένου μοντέλου. Η αύξηση αυτή του σφάλματος βάση αποτελεσμάτων δεν καθιστά ικανοποιητική τη μεταφορά αυτή, οδηγώντας στην επιλογή της εκπαίδευσης ενός νέου μοντέλου.

6.4 Χρήση Transformer για την βελτίωση επίδοσης του Ithemal

6.4.1 Διαδικασία εκπαίδευσης Transformer

Μια νέα προσέγγιση στο κομμάτι των νευρωνικών δικτύων που εμφανίστηκε τα τελευταία χρόνια και χρησιμοποιείται σε μεγάλο βαθμό είναι αυτό των Transformers. Η διαδικασία εκπαίδευσης ενός Transformer από την αρχή είναι πολύ δύσκολη και χρονοβόρα και για το λόγο αυτό συχνά προτιμάται η χρήση ενός προεκπαιδευμένου. Η προσέγγιση αυτή έχει εξίσου καλά αποτελέσματα καθώς υπάρχει πληθώρα προεκπαιδευμένων Transformers για διαφορετικά είδη δεδομένων. Η ίδια τεχνική ακολουθήθηκε κι εδώ με χρήση ενός Transformer της οικογένειας BERT και συγκεκριμένα τον DistilBERT. Χρησιμοποιείται για δεδομένα κειμένου, είναι γρήγορος στην παραγωγή προβλέψεων και με σχετικά μικρό υπολογιστικό κόστος. Αρχικά έγινε η διαδικασία οργάνωσης και μορφοποίησης των δεδομένων στην παρακάτω μορφή.[40]

{"id": 4889de4889c24c89ff4889de4889c24c89ff , "text": "mov rsi,rbx; mov rdx,rax; mov rdi,r15; mov rsi,rbx; mov rdx,rax; mov rdi,r15; dec r10; jne 0x0 ", "uuid": "4889de4889c24c89ff4889de4889c24c89ff ", "score": 175}

Τα δεδομένα όπως σε όλες τις περιπτώσεις χωρίστηκαν σε train, validate και test με τα συνήθη ποσοστά ώστε να γίνει αντίστοιχα η εκπαίδευση, επαλήθευση και αξιολόγηση του μοντέλου-Transformer. Όπως και στις προηγούμενες προσεγγίσεις, έτσι κι εδώ χρησιμοποιήθηκαν οι μετρήσεις σε όλα τα συστήματα που αναφέρθηκαν παραπάνω

(Haswell, Broadwell, Skylake, Ice Lake, AMD) καθώς και τα δύο εργαλεία συλλογής μετρήσεων (timing-harness και nanoBench). Η διαδικασία εκπαίδευσης αρχικά απαιτούσε αρκετές ημέρες ώστε να ολοκληρωθεί και για το λόγο αυτό χρησιμοποιήθηκε μια A100 GPU, γεγονός που οδήγησε σε μεγάλη μείωση του χρόνου εκπαίδευσης.

6.4.2 Παρουσίαση αποτελεσμάτων και συμπεράσματα

Χρησιμοποιώντας τον ίδιο προεκπαιδευμένο Transformer έγιναν δύο εκπαιδεύσεις για κάθε σύνολο μετρήσεων, μια με 5 εποχές και μια με 100. Τα αποτελέσματα φαίνονται παρακάτω. Και σε αυτή την περίπτωση για παρουσίαση των αποτελεσμάτων εκτός από το μέσο σφάλμα χρησιμοποιήθηκαν οι μετρικές συσχέτισης (Kendall's Tau και Spearman Correlation) αλλά και η Μέση Απόλυτη Απόκλιση.

timing-harness	Error(%)	Kendall's Tau	Spearman Correlation	MAD
HSW 5 Εποχές **100 Εποχές**	7.2 **3.1**	0.855 **0.916**	0.961 **0.981**	0.03 **0.008**
BDW 5 Εποχές **100 Εποχές**	7.4 **3.7**	0.852 **0.898**	0.96 **0.977**	0.031 **0.009**
SKL 5 Εποχές **100 Εποχές**	10 **6.7**	0.812 **0.867**	0.935 **0.957**	0.035 **0.016**
ICL 5 Εποχές **100 Εποχές**	8.2 **5.5**	0.874 **0.903**	0.973 **0.983**	0.038 **0.023**

Πίνακας 6.14: Πίνακας μετρικών για τους Transformers με timing-harness μετρήσεις.

nanoBench	Error(%)	Kendall's Tau	Spearman Correlation	MAD
BDW 5 Εποχές **100 Εποχές**	3.9 **1.3**	0.882 **0.90**	0.96 **0.966**	0.089 **0.009**
SKL 5 Εποχές **100 Εποχές**	4.3 **1.5**	0.876 **0.896**	0.956 **0.963**	0.094 **0.015**
ICL 5 Εποχές **100 Εποχές**	4.3 **1.4**	0.884 **0.906**	0.961 **0.968**	0.097 **0.017**
AMD 5 Εποχές **100 Εποχές**	5.4 **4.6**	0.858 **0.864**	0.945 **0.947**	0.108 **0.035**

Πίνακας 6.15: Πίνακας μετρικών για τους Transformers με nanoBench μετρήσεις.

Παρατηρώντας τα παραπάνω δεδομένα βλέπουμε αρχικά ότι και στις δύο εκδοχές αλλά και τα δύο εργαλεία μέτρησης τα αποτελέσματα είναι πολύ καλά και συγκεκριμένα καλύτερα και από το Ithemal. Ο Transformer ακόμα και με πολύ λίγες εποχές εκπαίδευσης κάνει πολύ καλές προβλέψεις. Αυξάνοντας τις εποχές φαίνεται ότι το σφάλμα μειώνεται ακόμα περισσότερο πετυχαίνοντας μάλιστα τα καλύτερα αποτελέσματα από όλες τις προσεγγίσεις που δοκιμάστηκαν. Τόσο η συσχέτιση όσο και η μέση απόκλιση επιβεβαιώνουν τα καλά αποτελέσματα. Αξίζει να σημειωθεί ότι το σφάλμα σε Skylake και Ice Lake μηχάνημα με τις μετρήσεις με το timing-harness αυξάνεται ελαφρώς λόγω της ποιότητας των μετρήσεων σε αυτή την περίπτωση. Και πάλι με την αύξηση των εποχών πέφτει αρκετά και συγκεκριμένα με τις μετρήσεις από το nanoBench το σφάλμα προσεγγίζει και πάλι το 2%. Τέλος, όπως και στις προηγούμενες περιπτώσεις, οι μετρήσεις σε AMD μηχάνημα οδηγούν και πάλι σε πολύ αξιόπιστα μοντέλα.

6.4.3 Αξιολόγηση επίδοσης ενός επεξεργαστή με προεκπαιδευμένο Transformer

Όπως και στην περίπτωση του Ithemal έτσι και στην προσέγγιση του Transformer έγινε προσπάθεια μεταφοράς κάθε μοντέλου στα υπόλοιπα μηχανήματα με σκοπό την αποφυγή της εκπαίδευσης. Και πάλι όπως και προηγουμένως εκτός του σφάλματος χρησιμοποιήθηκαν οι μετρικές συσχέτισης για την αξιολόγηση των νέων αυτών μοντέλων. Οι ομοιότητες των διαφόρων γενεών που αναφέρθηκαν και παραπάνω αναμένεται να οδηγήσουν και στην καλύτερη μεταφορά μεταξύ των εκπαιδευμένων Transformer στις γενιές αυτές (Haswell, Broadwell, Skylake).

(timing-harness)	Haswell	Broadwell	Skylake	Icelake
Haswell **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	**0** 0 **0**	**3.9** 9.9 **3.8**	**7.8** 12.7 **8.2**	**12.4** 16.9 **12.8**
Broadwell **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	**4** 10.6 **3.5**	**0** 0 **0**	**8.9** 14.2 **7.8**	**13.2** 17.4 **12.9**
Skylake **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	**7.3** 11.6 **7.6**	**8.2** 12.2 **7.9**	**0** 0 **0**	**13.2** 15.7 **12.2**
Icelake **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	**9.2** 14.1 **10.3**	**10.1** 15.1 **10.5**	**10.1** 14.9 **10.8**	**0** 0 **0**

Πίνακας 6.16: Πίνακας σφάλματος για μεταφορά Transformers σε άλλους επεξεργαστές με timing-harness μετρήσεις.

(nanoBench)	Broadwell	Skylake	Icelake	AMD
Broadwell **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	 **0** 0 **0**	 **2.4** 7.9 **2.3**	 **8.6** 13.8 **7**	 **25.2** 18.8 **15**
Skylake **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	 **2.1** 8.3 **2.8**	 **0** 0 **0**	 **7.5** 12.7 **6.1**	 **25.2** 17.7 **13.4**
Icelake **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	 **5.8** 12.3 **5.6**	 **4.3** 11.7 **8.4**	 **0** 0 **0**	 **23.1** 19.3 **14.5**
AMD **Απόκλιση Μετρήσεων** Error Μεταφοράς Ithemal **Error Μεταφοράς Transformer**	 **18.3** 14.5 **10.6**	 **16.6** 13.7 **9.5**	 **20.1** 16.1 **11.6**	 **0** 0 **0**

Πίνακας 6.17: Πίνακας σφάλματος για μεταφορά Transformers σε άλλους επεξεργαστές με nanoBench μετρήσεις.

Spearman Correlation (timing-harness)	Haswell	Broadwell	Skylake	Icelake
Haswell	1	0.97	0.95	0.94
Broadwell	0.976	1	0.952	0.941
Skylake	0.958	0.958	1	0.94
Icelake	0.93	0.93	0.91	1

Πίνακας 6.18: Πίνακας Spearman Correlation για μεταφορά Transformers σε άλλους επεξεργαστές με timing-harness μετρήσεις.

Spearman Correlation (nanoBench)	Broadwell	Skylake	Icelake	AMD
Broadwell	1	0.96	0.94	0.92
Skylake	0.964	1	0.953	0.934
Icelake	0.95	0.945	1	0.92
AMD	0.94	0.93	0.93	1

Πίνακας 6.19: Πίνακας Spearman Correlation για μεταφορά Transformers σε άλλους επεξεργαστές με nanoBench μετρήσεις.

Όπως και στην περίπτωση του Ithemal έτσι και σε αυτή την προσέγγιση η προσπάθεια μεταφοράς του εκπαιδευμένου μοντέλου οδηγεί σε επιβάρυνση και αύξηση του σφάλματος. Η διαφοροποίηση εδώ είναι το γεγονός ότι μεταξύ κοντινών γενεών (Haswell, Broadwell, Skylake) η αύξηση του σφάλματος είναι τέτοια που θα έκανε εφικτή τη μεταφορά. Αντίθετα η μεταφορά των παραπάνω σε Ice Lake ή AMD αυξάνει και πάλι παραπάνω το σφάλμα μειώνοντας την ποιότητα και την αξιοπιστία του μοντέλου. Τα αποτελέσματα αυτά επιβεβαιώνονται και από τη διακύμανση των τιμών των μετρικών συσχέτισης μεταξύ πραγματικών τιμών και προβλέψεων των μοντέλων.

Κεφάλαιο 7 - Επίλογος και Μελλοντικές επεκτάσεις

Η πρόβλεψη των κύκλων του ρολογιού που απαιτούνται από έναν επεξεργαστή για να ένα μέρος κώδικα από εντολές assembly είναι ένα βασικό θέμα για σχεδιαστές compilers και για μηχανικούς και ερευνητές. Έχοντας μια καλή εκτίμηση για αυτό το μέγεθος είναι δυνατή η πρόβλεψη και η εκτίμηση της απόδοσης του συστήματος (runtime performance). Χαρακτηριστικά παραδείγματα της χρησιμότητας της μέτρησης είναι η κατανομή καταχωρητών (register allocation) και η δρομολόγηση εντολών (instruction scheduling). Η εναλλακτική της εκτίμησης, η μέτρηση δηλαδή εκτελώντας τον κώδικα κάθε φορά, αποτελεί μια πολύ υπολογιστικά κοστοβόρα λύση και δεν προτιμάται από τους χρήστες. Ενώ αρχικά η Intel και άλλοι οργανισμοί ανέπτυξαν μοντέλα με προφανώς καλύτερα αποτελέσματα από την αναλυτική μέθοδο εδώ αναλύθηκε ένα καινοτόμο μοντέλο-εργαλείο, το Ithemal.
Ο τομέας των νευρωνικών δικτύων τα τελευταία χρόνια αναπτύσσεται με πολύ γρήγορους ρυθμούς λόγω της μεγάλης έξαρσης της ποσότητας των δεδομένων αλλά και των υπολογιστικών πόρων. Οπότε όπως αναμενόταν βρήκαν εφαρμογή και συνδυάστηκαν και σε αυτό τον τομέα.
Το Ithemal που αναπτύχθηκε είναι πολύ πιο ακριβές από τα πιο σύγχρονα μοντέλα (state-of-the-art models) και μπορεί να χρησιμοποιηθεί σε πλήθος μικροεπεξεργαστών. Πρόκειται για το πρώτο εργαλείο που κάνει την παραπάνω πρόβλεψη με χρήση τεχνικών μηχανικής μάθησης. Συγκεκριμένα μαθαίνει το πως να προβλέπει την απόδοση ενός συνόλου εντολών χρησιμοποιώντας μια προσέγγιση βασισμένη σε ένα LSTM-δίκτυο.

7.1 Συμπεράσματα

Παρατηρώντας όλα τα παραπάνω αποτελέσματα αρχικά φαίνεται η δυνατότητα για γρήγορη και αξιόπιστη πρόβλεψη της απόδοσης ενός τμήματος κώδικα ανεξάρτητα από τις ιδιότητες του εκάστοτε επεξεργαστή. Ξεκινώντας από την προσέγγιση του Ithemal, αυτό εκπαιδεύεται με καλά αποτελέσματα με το σύνολο δεδομένων BHive και παρουσιάζει πολύ καλά αποτελέσματα σε όλες τις γενιές και οικογένειες επεξεργαστών που δοκιμάστηκε. Τα καλά νούμερα ακρίβειας προφανώς προκύπτουν από τη σύγκριση με τις ποιοτικές μετρήσεις που έδωσαν σε όλες τις περιπτώσεις τόσο το timing-harness όσο και το nanoBench εργαλείο.
Σχετικά με τη μεταφορά ενός προεκπαιδευμένου μοντέλου σε μηχανή με άλλο επεξεργαστή αυτό είναι εφικτό με μια αύξηση στο παραγόμενο σφάλμα, ειδικά. Όσο περισσότερες διαφορές έχουν οι επεξεργαστές, τόσο αυξάνεται και το σφάλμα γεγονός που έγκειται στις διαφορές τεχνολογίας και μικροαρχιτεκτονικής μεταξύ των επεξεργαστών. Δεδομένου, όμως, του χρόνου που χρειάζεται για την εκπαίδευση εξ αρχής ενός μοντέλου (περίπου 8 ώρες) σε περίπτωση απαίτησης μεγαλύτερης ακρίβειας από τους προγραμματιστές θα επιλέγονταν η εκπαίδευση ενός νέου μοντέλου.
Έπειτα, στην προσπάθεια για βελτίωση της απόδοσης των προβλέψεων με την προσέγγιση του Transformer που δοκιμάστηκε τελευταία, αυτή έχει εξίσου καλά αποτελέσματα και με εκπαίδευση σε 100 εποχές πετυχαίνει την εκπαίδευση των βέλτιστων μοντέλων. Ο χρόνος που απαιτείται για την εκπαίδευση αυτή είναι περίπου 12 ώρες ενώ για 5 εποχές μόλις 30 λεπτά γεγονός που δίνει τη δυνατότητα στους προγραμματιστές και για σύντομη και για πιο χρονοβόρα εκπαίδευση. Πρόκειται για μια επιλογή με καλά αποτελέσματα και εύκολο στην υλοποίηση και τη διαδικασία εκπαίδευσης.

Στην αξιολόγηση και πάλι της δυνατότητας χρήσης ενός προεκπαιδευμένου μοντέλου σε άλλους επεξεργαστές με τη χρήση του Transformer η επιβάρυνση-σφάλμα μειώνεται και μάλιστα προσεγγίζει την διαφορά των μετρήσεων μεταξύ των διαφόρων επεξεργαστών.

7.2 Μελλοντικές επεκτάσεις

Με βάση όλα τα παραπάνω αποτελέσματα και τις παραδοχές που έγιναν στο πλαίσιο της εργασίας φανερώνουν τη δυνατότητα περαιτέρω μελέτης σχετικά με την υλοποίηση ενός μοντέλου πρόβλεψης της απόδοσης ενός τμήματος κώδικα.

7.2.1 Βελτιώσεις σε επίπεδο διαχείρισης κώδικα

Στην προσπάθεια μελλοντικής επέκτασης της συγκεκριμένης δουλειάς μια πρώτη σκέψη θα ήταν η δυνατότητα πρόβλεψης ενός πλήρους κώδικα από το μοντέλο. Πιο συγκεκριμένα να μην περιορίζεται η πρόβλεψη μόνο σε τμήματα κώδικα αλλά σε πλήρη κώδικα ώστε να προσφέρει ακόμα μεγαλύτερη βοήθεια στους προγραμματιστές και τους σχεδιαστές και να έχουν ακόμα πιο πλήρη εικόνα του κώδικά τους. Επίσης θα μπορούσαν να γίνουν αλλαγές ώστε το εργαλείο να προβλέπει το αποτέλεσμα μιας διακλάδωσης και να μπορεί να διαχειριστεί και τέτοιες εντολές.

7.2.2 Βελτιώσεις σε επίπεδο υλικού

Το μοντέλο του Ithemal δοκιμάστηκε σε αρκετές από τις αρχιτεκτονικές της Intel καθώς και σε AMD. Θα μπορούσαν να γίνουν περισσότερες δοκιμές και σε πιο παλιές αρχιτεκτονικές ώστε να φανούν πιο έντονα οι διαφορές τόσο σε χρόνο και απόδοση όσο και σε αποτελέσματα καθώς βελτιώνονται οι αρχιτεκτονικές. Επιπλέον, μπορεί να γίνει προσπάθεια εφαρμογής του μοντέλου σε αρχιτεκτονικές με διαφορετικό ISA, όπως σε RISC-V και ARM.

7.2.3 Βελτιώσεις σε επίπεδο αρχιτεκτονικής

Ενδιαφέρον ακόμα θα είχε να γίνουν παρεμβάσεις σε επίπεδο αρχιτεκτονικής του μοντέλου. Αυτές οι αλλαγές αρχικά θα μπορούσαν να αφορούν το κομμάτι των νευρωνικών, είτε παραμείνει LSTM δίκτυο και γίνουν κάποιες βελτιώσεις είτε ενσωματωθεί κάποιος Transformer. Εκτός από τον τομέα των νευρωνικών του μοντέλου αλλαγές θα μπορούσαν να γίνουν και στα υπόλοιπα layer, με χαρακτηριστικό παράδειγμα τον τρόπο συλλογής και επεξεργασίας των token για την κάθε εντολή. Σημαντικό κομμάτι ικανό να βελτιωθεί είναι και αυτό των assumptions όπου μπορεί να γίνουν βελτιώσεις στη θεώρηση ότι θα έχουμε πάντα cache hits και να προστεθεί ένα κομμάτι σχετικό με τη μνήμη και την πιθανότητα να βρίσκονται τα δεδομένα σε αυτήν.

Ο κώδικας βρίσκεται σο github στη παρακάτω διεύθυνση με λεπτομέρειες για την εγκατάσταση και τη λειτουργία κάθε αρχείου.
https://github.com/teoaivalis/Diploma-Thesis-Ithemal/tree/main

Βιβλιογραφία

1. Intel® Architecture Code Analyzer URL:https://www.intel.com/content/www/us/en/developer/articles/tool/architecture-code-analyzer.html
2. The LLVM Compiler Infrastructure URL:https://llvm.org
3. LLVM-Wikipedia URL:https://en.wikipedia.org/wiki/LLVM
4. Clement Courbet . ≪The LLVM Compiler Infrastructure≫ URL:https://github.com/llvm/llvm-project
5. LLVM Compiler Infrastructure URL:https://llvm.org/docs/index.html
6. Souradip Ghosh ≪LLVM Machine Code Analyzer (llvm-mca)≫ URL:https://souradipghosh.com/files/mca.pdf
7. Chris Lattner, Vikram Adve LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation URL:https://llvm.org/pubs/2004-01-30-CGO-LLVM.pdf
8. Compiler Explorer URL:https://godbolt.org
9. Charith Mendis, Alex Renda, Saman Amarasinghe, Michael Carbin Github-Ithemal URL:https://github.com/ithemal/Ithemal
10. Charith Mendis, Alex Renda, Saman Amarasinghe, Michael Carbin ≪Ithemal: Accurate, Portable and Fast Basic Block Throughput Estimation using Deep Neural Networks≫ URL:https://arxiv.org/pdf/1808.07412.pdf
11. Andreas Abel and Jan Reineke ≪uiCA: Accurate Throughput Prediction of Basic Blocks on Recent Intel Microarchitectures≫ URL:https://dl.acm.org/doi/pdf/10.1145/3524059.3532396
12. Andreas Abel and Jan Reineke, Github-uiCA (uops.info Code Analyzer) URL:https://github.com/andreas-abel/uiCA
13. Alex Renda, Yishen Chen, Charith Mendis, Michael Carbin, ≪DiffTune: Optimizing CPU Simulator Parameters with Learned Differentiable Surrogates≫ URL:https://arxiv.org/pdf/2010.04017.pdf
14. Charith Mendis, Github-DiffTune: Optimizing CPU Simulator Parameters with Learned Differentiable Surrogates URL:https://github.com/ithemal/DiffTune
15. Lingda Li, Santosh Pandey, Thomas Flynn, Hang Liu, Noel Wheeler, Adolfy Hoisie, ≪SimNet: Accurate and High-Performance Computer Architecture Simulation using Deep Learning≫ URL:https://arxiv.org/pdf/2105.05821.pdf
16. Lingda Li, Github-SimNet URL:https://github.com/lingda-li/simnet
17. Μηχανική μάθηση-Wikipedia URL:https://el.wikipedia.org/wiki/Μηχανική_μάθηση
18. Stuart Russel, Peter Norvig, ≪Τεχνητή Νοημοσύνη, Μια σύγχρονη προσέγγιση, 4η αμερικανική έκδοση≫
19. Simon Haykin, ≪Νευρωνικά Δίκτυα και Μηχανική Μάθηση, 3η έκδοση≫
20. Ιωάννης Γ. Τσούλος, ≪Τεχνητά Νευρωνικά Δίκτυα≫ URL:https://www.dit.uoi.gr/e-class/modules/document/file.php/249/ΔΙΑΛΕΞΕΙΣ/lecture1.pdf
21. Αθανάσιος Χασιακός, Παναγιώτης Τσίκας, ≪Εξελικτικοί Αλγόριθμοι Βελτιστοποίησης≫ URL:https://eclass.upatras.gr/modules/document/file.php/CIV1756/8-Artificial%20Neural%20Networks%20%28ANN%29.pdf

22. Kamil Krzyk, Cost Function of Linear Regression
URL:https://builtin.com/machine-learning/cost-function
23. Nagesh Singh Chauhan, Optimization Algorithms in Neural Networks
URL:https://www.kdnuggets.com/2020/12/optimization-algorithms-neural-networks.html
24. Sumit Saha, ≪A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way≫
URL:https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
25. Mehreen Saeed, An Introduction to Recurrent Neural Networks and the Math That Powers Them
URL:https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/
26. Christopher Olah, Understanding LSTM Networks
URL:https://colah.github.io/posts/2015-08-Understanding-LSTMs/
27. Sepp Hochreiter, Jürgen Schmidhuber, ≪LONG SHORT-TERM MEMORY≫
URL:https://www.bioinf.jku.at/publications/older/2604.pdf
28. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, ≪Attention Is All You Need≫
URL:https://arxiv.org/pdf/1706.03762.pdf
29. La Javaness R&D, Regression with Text Input Using BERT and Transformers
URL:https://lajavaness.medium.com/regression-with-text-input-using-bert-and-transformers-71c155034b13
30. Yishen Chen, Ajay Brahmakshatriya, Charith Mendis, Alex Renda, Eric Atkinson, Ondˇrej Sykora, Saman Amarasinghe, Michael Carbin, ≪BHive: A Benchmark Suite and Measurement Framework for Validating x86-64 Basic Block Performance Models≫,
URL:https://groups.csail.mit.edu/commit/papers/19/ithemal-measurement.pdf
31. Yishen Chen, Charith Mendis, Alex Renda, Github-ithemal/bhive, URL: https://github.com/ithemal/bhive
32. 20 Popular Machine Learning Metrics. Part 1: Classification & Regression Evaluation Metrics, URL: https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce
33. How to Calculate the Median Absolute Deviation in Python,
URL: https://datagy.io/mean-absolute-deviation-python/
34. What is R Squared? R2 Value Meaning and Definition,
URL:
https://www.freecodecamp.org/news/what-is-r-squared-r2-value-meaning-and-definition/
35. How to Find Outliers in Python, URL:
https://absentdata.com/python/how-to-find-outliers-in-python/
36. M. G. Kendall, ≪A New Measure of rank Correlation≫,
URL: https://www.jstor.org/stable/2332226?origin=crossref
37. Python | Kendall Rank Correlation Coefficient,
URL: https://www.geeksforgeeks.org/python-kendall-rank-correlation-coefficient/
38. Spearman's Rank Correlation, URL:
https://www.geeksforgeeks.org/spearmans-rank-correlation/

39. Rank of all elements in an array, URL: https://www.geeksforgeeks.org/rank-elements-array/
40. DistilBERT, URL: https://huggingface.co/docs/transformers/main/model_doc/distilbert
41. Haswell (microarchitecture), URL:https://en.wikipedia.org/wiki/Haswell_(microarchitecture)
42. Skylake (microarchitecture), URL:https://en.wikipedia.org/wiki/Skylake_(microarchitecture)
43. Broadwell (microarchitecture), URL:https://en.wikipedia.org/wiki/Broadwell_(microarchitecture)
44. Ice Lake (microprocessor), URL:https://en.wikipedia.org/wiki/Ice_Lake_(microprocessor)
45. AMD EPYC™ 7402, URL: https://www.amd.com/en/products/cpu/amd-epyc-7402
46. Xeon URL: https://en.wikipedia.org/wiki/Xeon