ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ & ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Συστηματική ανάλυση και σύγκριση μεθόδων επιτάχυνσης Μεγάλων Γλωσσικών Μοντέλων

Φοιτήτρια: Κοιλιά Νικολέττα, ΑΜ: 19387106

Επιβλέπων Καθηγητής: Χριστόφορος Κάχρης, Επίκουρος Καθηγητής

ΑΘΗΝΑ-ΑΙΓΑΛΕΩ, ΙΟΥΛΙΟΣ 2024

UNIVERSITY OF WEST ATTICA

FACULTY OF ENGINEERING

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Diploma Thesis

A comprehensive survey, taxonomy, and comparison of Accelerated Large Language Models

Student: Koilia Nikoletta, Registration Number: 19387106

Supervisor: Christoforos Kachris, Assistant Professor

ATHENS-EGALEO, July 2024

Η Διπλωματική Εργασία έγινε αποδεκτή και βαθμολογήθηκε από την εξής τριμελή επιτροπή:

Χριστόφορος Κάχρης, Επίκουρος Καθηγητής (επιβλέπων)	Μαρία Ραγκούση, Καθηγήτρια	Ευστάθιος Κυριάκης-Μπιτζάρος, Καθηγητής
ΧΡΙΣΤΟΦΟΡΟΣ ΚΑΧΡΗΣ Digitally signed by ΧΡΙΣΤΟΦΟΡΟΣ ΚΑΧΡΗΣ Date: 2024.08.28 16:16:45 +03'00' (Υπογραφή)	MARIA RAGKOUSI Digitally signed by MARIA RAGKOUSI Date: 2024.08.28 10:11:15 +03'00' (Υπογραφή)	Efstathios Kyriakis-Bitzaros 2024.08.01 22:39:48 +03'00' (Υπογραφή)

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η κάτωθι υπογεγραμμένη Κοιλιά Νικολέττα του Δημητρίου, με αριθμό μητρώου 19387106 φοιτήτρια του Πανεπιστημίου Δυτικής Αττικής της Σχολής ΜΗΧΑΝΙΚΩΝ του Τμήματος ΗΛΕΚΤΡΟΛΟΓΩΝ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ,

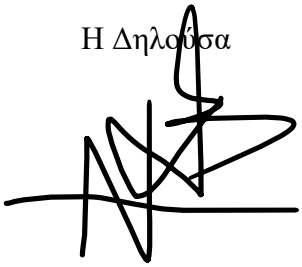
δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του διπλώματός μου.

Επιθυμώ την απαγόρευση πρόσβασης στο πλήρες κείμενο της εργασίας μου μέχρι το 2025 (6 μήνες) και έπειτα από αίτησή μου στη Βιβλιοθήκη και έγκριση του επιβλέποντος καθηγητή.»

Η Δηλούσα



Κοιλιά Νικολέττα

Αφιερωμένη στους γονείς μου Δημήτριο και Αικατερίνη Κοιλιά, καθώς και στην αδελφή μου Παρασκευή.

Ευχαριστίες

Θα ήθελα πρωτίστως να ευχαριστήσω τον επιβλέποντα καθηγητή μου κύριο Χριστόφορο Κάχρη, για την πολύτιμη βοήθεια και υποστήριξή του κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας, καθώς οι συμβουλές του και η άμεση ανταπόκρισή του σε ο,τιδήποτε του ζητήθηκε αποτέλεσαν ιδιαίτερο κίνητρο, με αποτέλεσμα σήμερα να έχει ολοκληρωθεί επιτυχώς η παρούσα διπλωματική εργασία.

Ακόμη, θα ήθελα να ευχαριστήσω τους γονείς μου και γενικότερα την όλη μου οικογένεια για την αμέριστη ηθική υποστήριξή τους τόσο κατά τη διάρκεια των σπουδών όσο και κατά διάρκεια ολοκλήρωσης της παρούσας διπλωματικής εργασίας, στοιχεία τα οποία σε κάθε νέο παιδί είναι απαραίτητα για να μπορεί να ολοκληρώσει επιτυχώς τις σπουδές του.

Εκφράζω, λοιπόν, τα θερμά μου ευχαριστήρια προς αυτές τις σπουδαίες προσωπικότητες για την ανεκτίμητη συμβολή τους στην επιτυχία μου.

Περίληψη

Μέχρι σήμερα δεν υπάρχει κάποια συστηματική και συγκριτική έρευνα για τους επιταχυντές υλικού (hardware accelerators) που να αναφέρονται στην υπολογιστική ισχύ των διάφορων γλωσσικών μοντέλων επιταχυντών. Σε αυτή την εργασία παρουσιάζεται κατ' αρχήν μια επισκόπηση σχετικά με τις έρευνες και τα μοντέλα επιταχυντών που έχουν παρουσιαστεί διαχρονικά για την επιτάχυνση των Μεγάλων Γλωσσικών Μοντέλων και της επεξεργασίας της Φυσικής Γλώσσας, χρησιμοποιώντας επιταχυντές υλικού.

Η επισκόπηση παρουσιάζει τα πλαίσια που έχουν προταθεί και στη συνέχεια πραγματοποιεί μια ποιοτική και ποσοτική σύγκριση όσο αφορά τη τεχνολογία και τον τύπο επεξεργαστή που χρησιμοποιούν (FPGA, ASIC, In-Memory, GPU), την ενεργειακή απόδοση, την επιτάχυνσης, καθώς και τους αντίστοιχους ρυθμούς (επιτάχυνσης και ενεργειακής απόδοσης).

Το μεγαλύτερο πρόβλημα είναι ότι οι υπάρχουσες σχετικές ερευνητικές προτάσεις συνήθως υλοποιούνται η καθεμία σε διαφορετική τεχνολογία (process technology). Αυτό έχει ως αποτέλεσμα να γίνεται δύσκολη η δίκαια σύγκριση των προτεινόμενων λύσεων. Σκοπός του εφαρμοσμένου, πειραματικού μέρους αυτής της διπλωματικής εργασίας ήταν να γίνει αναγωγή των αποτελεσμάτων σε μία κοινή τεχνολογία, κάνοντας έτσι δυνατή την δίκαια σύγκριση. Για την αναγωγή στην ίδια τεχνολογία (process technology) χρησιμοποιήθηκαν και αξιολογήθηκαν 2 προσεγγίσεις, μία θεωρητική αναγωγή (extrapolation) στην ίδια τεχνολογία και μία εργαστηριακή αναγωγή με βάση τα αποτελέσματα υλοποίησης ψηφιακών κυκλωμάτων σε διάφορες πλατφόρμες αναδιατασσόμενης λογικής (FPGA platforms). Στη συνέχεια παρουσιάζονται τα αποτελέσματα της υπολογιστικής επιτάχυνσης με αναγωγή στην ίδια τεχνολογία.

Ακόμη, παρατίθενται δύο παραδείγματα της Μηχανικής Μάθησης με χρήση των LLMs για τους κλασικούς επεξεργαστές. Τέλος η εργασία κλείνει με την εξαγωγή σημαντικών συμπερασμάτων με βάση τη θεωρητική μελέτη αλλά και το πειραματικό μέρος.

Λέξεις – κλειδιά

Τεχνική νοημοσύνη, Βαθιά Μάθηση, Μεγάλα Γλωσσικά Μοντέλα, Κωδικοποιητής, Αποκωδικοποιητής, FPGA, GPU, In-Memory, ASIC, Επιτάχυνση, Ενεργειακή Απόδοση, Ρυθμός Απόδοσης, Τεχνολογία των 7nm, Τεχνολογία των 16nm, VHDL, Ollama, Γραμμική Παλινδρόμηση, Δένδρο Αποφάσεων, Τυχαίο Δένδρο Αποφάσεων, Μηχανή Διανυσμάτων Απόφασης.

Abstract

Until now there is no comprehensive survey on the hardware accelerators to speed up the most computationally intensive tasks of Transformers. In this diploma thesis, we present a comprehensive survey on the several research efforts that have been published on the acceleration of transformer networks for Large Language Models and Natural Language Processing (NLP) using hardware accelerators.

The survey presents the frameworks that have been proposed and then performs a qualitative and quantitative comparison regarding the technology, the processing platform (FPGA, ASIC, In-Memory, GPU), the speedup, the energy efficiency, the performance, and the energy efficiency (GOPs/W) of each framework.

The main challenge a comparative study is faced with is that every proposed scheme is implemented on a different process technology, thus making the fair comparison a hard task. In the applied, experimental part of this diploma thesis, we extrapolate the results of the speedup and the performance of the hardware accelerators using 2 different approaches, a theoretical one and a more practical one. We implement part of the LLMs on several FPGA chips to extrapolate the results to the same process technology and then we make a fair comparison of the performance.

Additionally, two examples of Machine Learning using LLMs for classical processors are provided. The diploma thesis concludes with the extraction of significant results drawn from both the theoretical study (hardware accelerators) and the experimental study parts.

Keywords

Artificial Intelligence, Machine Learning, Deep Learning, Large Language Models, Encoder, Decoder, FPGA, GPU, In-Memory, ASIC, Speed-up, Energy efficiency, Performance, 7nm Technology, 16nm Technology, VHDL, Ollama, Linear Regression, Decision Trees, Random Forest, Support Vector Machines.

Περιεχόμενα

Κατάλογος Πινάκων

Κατάλογος Εικόνων

Κατάλογος Σχεδιαγραμμάτων

Αλφαβητικό Ευρετήριο

AI: Artificial intelligence

BERT: Bidirectional Representations from Transformers

DSC : Depth-wise Separable Convolution

GEMM : General Matrix Multiply

GPT: Generative Pre-trained Transformer

MHA : Multi-Head Attention

MHSA : Multi-Head Self-Attention

LLMs: Large Language Models

NLP : Natural Language Processing

OBS : Output Block Stationary

STA : Sparse Transformer Accelerator

SVM : Support Vector Machines

ViT : Visual Transformer

ΕΙΣΑΓΩΓΗ

Είναι γεγονός ότι, τα τελευταία χρόνια, με την εξέλιξη της Τεχνητής Νοημοσύνης στο πεδίο της Πληροφορικής εισάγεται για πρώτη φορά ο όρος «Μεγάλα Γλωσσικά Μοντέλα» (Large Language Models – LLMs). Τα Μεγάλα Γλωσσικά Μοντέλα αποτελούν ένα σημαντικό νέο κλάδο στο χώρο της Τεχνητής Νοημοσύνης (Artificial Intelligence – AI) και της Επεξεργασίας Φυσικής Γλώσσας (Natural Language Processing – NLP), καθώς έχουν τη δυνατότητα να παράγουν ανθρώπινο κείμενο (text) με ακρίβεια και αξιόλογη ποιότητα. Αυτό συμβαίνει γιατί ένα LLM αποτελεί ένα τύπο αλγορίθμου Τεχνητής Νοημοσύνης (AI) που χρησιμοποιεί διάφορες τεχνικές Βαθιάς Μάθησης (Deep Learning) και Μεγάλων Δεδομένων (Big Data), ώστε να μπορεί να παράγει / συνθέτει κείμενο που μοιάζει με την ανθρώπινη γλώσσα. Επίσης, όπως αναφέρθηκε, τα LLMs είναι βασισμένα στη Βαθιά Μάθηση, και συγκεκριμένα σε ένα τύπο Νευρωνικού Δικτύου το οποίο ονομάζεται Μοντέλο Επιτάχυνσης. Η δημιουργία ενός LLM απαιτεί πολύ αυξημένους υπολογιστικούς πόρους και εξειδικευμένες τεχνικές όπως η Μηχανική Μάθηση και ειδικότερα η Βαθιά Μάθηση.

Με απλά λόγια, ένα LLM είναι ένα πρόγραμμα υπολογιστή που έχει τροφοδοτηθεί με αρκετά παραδείγματα ώστε να μπορεί να αναγνωρίζει και να ερμηνεύει την ανθρώπινη γλώσσα ή άλλους τύπους πολύπλοκων δεδομένων και επιπλέον να συνθέτει απαντήσεις με τη μορφή κειμένου σε ερωτήσεις που δέχεται ως είσοδο και αναλύει. Πολλά από αυτά τα μοντέλα εκπαιδεύονται με δεδομένα που έχουν συγκεντρώσει από το Διαδίκτυο. Επιπλέον, τα LLMs χρησιμοποιούν ένα τύπο της Μηχανικής Μάθησης που ονομάζεται Βαθιά Μάθηση (Deep Learning), έτσι ώστε να κατανοήσουν και να διακρίνουν μέσα σε ένα κείμενο προτάσεις, λέξεις και χαρακτήρες. Η Βαθιά Μάθηση περιλαμβάνει την πιθανοθεωρητική ανάλυση μη δομημένων δεδομένων (unstructured data), η οποία τελικά επιτρέπει στο μοντέλο αυτό να αναγνωρίζει διακρίσεις μεταξύ του περιεχομένου χωρίς την ανθρώπινη παρέμβαση. Τα LLMs έχουν εφαρμογές σε πολλούς τομείς, οι οποίοι αναφέρονται σε επόμενη ενότητα. Ενδεικτικά, εδώ αναφέρονται κάποια από τα σήμερα διαθέσιμα μοντέλα, όπως το GPT, το BERT και το τελευταίο μοντέλο της OpenAI (GPT-4).

Αντικείμενο της διπλωματικής εργασίας

Στο πρώτο μέρος της διπλωματικής εργασίας, πραγματοποιείται μία βιβλιογραφική μελέτη πάνω στους επιταχυντές υλικού για LLMs. Η βιβλιογραφική αυτή μελέτη είναι αρκετά ενδιαφέρουσα, καθώς μέσα από μια ιστορική αναδρομή θα δούμε πως ξεκίνησαν τα Μεγάλα Γλωσσικά Μοντέλα, αλλά και την διαχρονική εξέλιξή τους εντός της Τεχνητής Νοημοσύνης, καθώς είναι ένα αντικείμενο που τα τελευταία χρόνια παρουσιάζει ιδιαίτερο ενδιαφέρον για όλη την Επιστημονική Κοινότητα. Άρα ένα πρώτο αντικείμενο της συγκεκριμένης διπλωματικής εργασίας

είναι η βιβλιογραφική μελέτη των μοντέλων επιτάχυνσης που υπάρχουν τα τελευταία χρόνια, καθώς και η σύγκρισή τους ως προς την επιτάχυνση (ρυθμό επιτάχυνσης), την ενεργειακή απόδοση, καθώς και η αναγωγή και η σύγκριση αυτών ως προς δύο συγκεκριμένες τεχνολογίες που έχουν επιλεγεί, (α) την τεχνολογία των 7nm και (β) την τεχνολογία των 16nm.

Στο πειραματικό μέρος της διπλωματικής εργασίας, συγκρίνουμε 2 διαφορετικούς τρόπους αναγωγής στην ίδια τεχνολογία για FPGAs. Στην πρώτη περίπτωση, η αναγωγή γίνεται σε θεωρητικό επίπεδο με βάση εξισώσεις που είναι διαθέσιμες στην βιβλιογραφία. Στη δεύτερη περίπτωση, η αναγωγή γίνεται με βάση τις πειραματικές μετρήσεις των πιο τυπικών συναρτήσεων των LLM σε διάφορες τεχνολογίες FPGAs. Στο ίδιο πλαίσιο, θα αξιοποιήσουμε την εφαρμογή Ollama των LLM, σε λογισμικό Linux, ώστε να εξετάσουμε και τους κλασικούς επεξεργαστές με χρήση των LLMs και να κάνουμε την αναγωγή σε ίδια τεχνολογία. Αυτές οι τρεις εφαρμογές αποτελούν το ερευνητικό μέρος της παρούσας διπλωματικής εργασίας.

Σκοπός και στόχοι

Σκοπός και στόχοι της παρούσας διπλωματικής επικεντρώνονται στα παρακάτω:

1. Η σύγκριση των ήδη υπαρχόντων LLMs σε σχέση με τον ρυθμό επιτάχυνσης και ενεργειακής απόδοσης αυτών.
2. Η εισαγωγή των προαναφερόμενων μοντέλων στη τεχνολογία των 7nm και η εκ νέου σύγκριση μεταξύ αυτών.
3. Η σύγκριση όλων των μοντέλων FPGA στη τεχνολογία των 16nm.
4. Η συσχέτιση μέσω κώδικα VHDL της συχνότητας με την καθυστέρηση και η σύγκριση με τη θεωρητική μελέτη για τα μοντέλα των FPGAs.
5. Η εξέταση μέσω του λογισμικού Linux του ρυθμού επιτάχυνσης μιας συγκεκριμένης πρότασης που υποβλήθηκε σε αυτό, σε κλασικούς επεξεργαστές.
6. Η εισαγωγή κώδικα Python για πρόβλεψη πιθανής επιτάχυνσης όταν χρησιμοποιούμε ανεξάρτητα χαρακτηριστικά τιμών υπολογιστών τα οποία δεν αναφέρονται.

Μεθοδολογία

Για την εκπόνηση της παρούσας διπλωματικής εργασίας χρησιμοποιήθηκε η εξής μεθοδολογία. Στην πρώτη φάση, πραγματοποιήθηκε βιβλιογραφική επισκόπηση για τη μελέτη των ήδη υπαρχόντων μοντέλων επιτάχυνσης (ψηφιακοί επιταχυντές). Στη συνέχεια, με βάση τα αποτελέσματα αυτής της επισκόπησης, έγινε η σύγκρισή τους ως προς την επιτάχυνση και την ενεργειακή απόδοση, με βάση τη τεχνολογία που αυτά χρησιμοποιούν (10nm, 14nm, 16nm, 20nm, 32nm, κ.ο.κ.). Κατόπιν, επιχειρήθηκε μια πιο δίκαιη σύγκριση στην τεχνολογία των 7 nm, αλλά και

των 16nm με τη χρήση θεωρητικών εξισώσεων και τέλος έγιναν εφαρμογές με χρήση των LLMs για κλασικούς επιταχυντές.

- Η πρώτη εφαρμογή αφορά την εύρεση της συχνότητας σε συγκεκριμένες τεχνολογίες για τα μοντέλα των FPGAs και τη σύγκρισή τους με τη θεωρητική μελέτη.
- Η δεύτερη εφαρμογή έγινε σε περιβάλλον Linux χωρίς τη χρήση GPU, προκειμένου να καταγράψουμε το ρυθμό επιτάχυνσης με βάση τον επεξεργαστή που χρησιμοποιείται από τον υπάρχοντα υπολογιστή.
- Η τρίτη και τελευταία εφαρμογή έγινε σε περιβάλλον Windows, με χρήση κώδικα σε γλώσσα προγραμματισμού Python.

Τα παραπάνω έγιναν προκειμένου να κάνουμε τη δική μας έρευνα πάνω στα LLMs, ώστε να δούμε ποιοί παράγοντες επηρεάζουν τους κλασικούς επεξεργαστές, πόσο σημαντικό ρόλο έχει η συχνότητα με την οποία μεταδίδουν το σήμα και να προβλέψουμε την πιθανή επιτάχυνση ενός υπολογιστή. Τέλος, στο τελευταίο κεφάλαιο της εργασίας, καταγράφονται τα συμπεράσματα από όλα τα στάδια και συγκρίσεις της παραπάνω μελέτης.

Καινοτομία

Η συγκεκριμένη εργασία αποτελεί καινοτομία για το πεδίο των Ηλεκτρονικών Υπολογιστών. Η πρωτοτυπία αυτής έγκειται στη σύγκριση των τεσσάρων διαφορετικών μοντέλων επιτάχυνσης σε δύο συγκεκριμένες τεχνολογίες που έχουμε επιλέξει, των 7nm αλλά και των 16nm για τα FPGAs, καθώς και οι εφαρμογές που έλαβαν χώρα σε διάφορους υπολογιστές χρησιμοποιώντας κλασικούς επεξεργαστές.

Δομή

Η παρούσα διπλωματική εργασία δομείται σε εννέα (9) κύρια μέρη. Το πρώτο μέρος καλύπτει το πώς ξεκίνησαν τα LLMs, καθώς και κάποιους βασικούς ορισμούς που θα χρειαστούν στη πορεία αυτής της μελέτης. Το δεύτερο μέρος αναφέρεται στις διαφορές που υπάρχουν ανάμεσα στα τέσσερα μοντέλα επιταχυντών (ASIC, FPGAs, In-Memory, GPU), αλλά και στις πράξεις πινάκων που χρησιμοποιούν τα μοντέλα αυτά. Ένα μεγάλο μέρος αυτής της εργασίας είναι η βιβλιογραφική μελέτη των μοντέλων αυτών και η σύγκρισή τους στην τεχνολογία των 7nm, όπως και η σύγκριση των FPGAs στη τεχνολογία των 16nm. Όλα τα προαναφερόμενα αποτελούν την πρώτη φάση (θεωρητική μελέτη) αυτής της διπλωματικής εργασίας.

Όσον αφορά στο εφαρμοσμένο, πειραματικό μέρος, αυτό αποτελείται από τρεις εφαρμογές, οι οποίες εκτελούνται σε πρόγραμμα Quartus, σε περιβάλλον Linux και τέλος σε περιβάλλον Windows. Η κάθε μια από αυτές έχει διαφορετικό σκοπό και στόχο.

Σε επίπεδο κεφαλαίων, η εργασία δομείται ως εξής:

Στο 1ο κεφάλαιο γίνεται ιστορική αναδρομή για τα Μεγάλα Γλωσσικά Μοντέλα, παρουσιάζονται κάποιοι ορισμοί που είναι απαραίτητοι για την κατανόηση των γλωσσικών μοντέλων, γίνεται αναφορά στην αρχιτεκτονική τους και τέλος παρουσιάζονται κάποια πλεονεκτήματα και περιορισμοί για την χρήση των μοντέλων αυτών.

Στο 2ο κεφάλαιο γίνεται αναφορά στις διάφορες κατηγορίες επιταχυντών που υπάρχουν (FPGA, GPU, In-Memory, ASIC) καθώς και στις μεταξύ τους διαφορές που παρουσιάζουν.

Τα κεφάλαια 3, 4, 5 και 6 αναφέρονται στους ψηφιακούς επεξεργαστές και ανάλογα με το μοντέλο επιτάχυνσης που διαθέτουν παρατίθενται αναλυτική βιβλιογραφική μελέτη, από τα πρώτα μοντέλα που δημοσιεύτηκαν μέχρι και τα πιο πρόσφατα αναλόγως σε ποια κατηγορία βρίσκονται. Επίσης, έχει σκοπό να παρουσιάσει τι μέθοδο χρησιμοποιούν οι ερευνητές (πολλαπλασιασμό πινάκων, πρόσθεση, αφαίρεση, συμπίεση) ώστε να φτιάξουν το μοντέλο επιτάχυνσης, καθώς και με τι επεξεργαστή (CPU-GPU) το συγκρίνουν για να πάρουν την αντίστοιχη επιτάχυνση και ενεργειακή απόδοση.

Στο 7ο κεφάλαιο γίνεται σύγκριση όλων των μοντέλων επιτάχυνσης που έχουν αναφερθεί παραπάνω. Η σύγκριση γίνεται με βάση την επιτάχυνση που λαμβάνουν, αλλά και με την ενεργειακή απόδοση αυτών. Για καλύτερα αποτελέσματα, γίνεται χρήση λογαριθμικής κλίμακας. Στο τέλος κάθε σύγκρισης λαμβάνονται παρατηρήσεις και συμπεράσματα. Σημαντικό ρόλο σε αυτή τη διπλωματική εργασία έχει η σύγκριση όλων των μοντέλων σε μια κοινή τεχνολογία επεξεργασίας που έχουμε λάβει, η οποία είναι η τεχνολογία των 7nm. Έτσι, γίνεται συσχετισμός της επιτάχυνσης και την ενεργειακής απόδοσης στα 7nm, και στο τέλος καταγράφονται κάποιες σημαντικές παρατηρήσεις σε σχέση με τα προαναφερόμενα. Στο τελευταίο μέρος αυτού του κεφαλαίου γίνεται η σύγκριση όλων των μοντέλων των FPGAs στην τεχνολογία των 16nm, καθώς και η σύγκριση της θεωρητικής μελέτης με τα πειραματικά αποτελέσματα του εφαρμοσμένου μέρους.

Στο 8ο κεφάλαιο γίνεται αναφορά στις εφαρμογές που ‘έτρεξαν’ σε διάφορους υπολογιστές, οι οποίες αφορούν τους κλασσικούς επεξεργαστές όπως τους επεξεργαστές της Intel.

Η πρώτη εφαρμογή αφορά τους επιταχυντές FPGAs, καθώς με χρήση κώδικα σε VHDL θέλουμε να βρούμε την συχνότητα που αντιστοιχεί σε διάφορες κατηγορίες-τεχνολογίες, να γίνει η

μετατροπή τους στη τεχνολογία των 16nm και να γίνει η σύγκρισή τους με τη θεωρητική μελέτη ως προς τον ρυθμό επεξεργασίας.

Η δεύτερη εφαρμογή έγινε σε λογισμικό Linux, όπου τρέξαμε ένα μοντέλο LLM για να εξετάσουμε τα LLMs και πώς μέσα από μια εφαρμογή μπορούμε να μετρήσουμε συγκεκριμένες παραμέτρους όπως η επιτάχυνση, εισάγοντας μια συγκεκριμένη ερώτηση. Την επιτάχυνση, λοιπόν, που εμφανίζει τη συγκρίνουμε με τα χαρακτηριστικά του εκάστοτε υπολογιστή, όπως μνήμη RAM, Cores, Threads, Συχνότητα και Processing Technology. Κάνοντας αυτές τις μετρήσεις μπορέσαμε να εξάγουμε κάποια σημαντικά αποτελέσματα.

Έπειτα, αναλύσαμε την τρίτη και τελευταία εφαρμογή με βάση τα χαρακτηριστικά των επεξεργαστών. Κύριο χαρακτηριστικό αυτής της εφαρμογής είναι πως με βάση τα χαρακτηριστικά των υπολογιστών που έλαβαν μέρος σε αυτή τη διπλωματική και εισάγοντας εμείς τα χαρακτηριστικά που θέλουμε να έχει ο δικός μας υπολογιστής, έχουμε ως αποτέλεσμα την πιθανή επιτάχυνση του υπολογιστή και πόσο αποκλίνει από τα υπάρχοντα χαρακτηριστικά (ανάλυση παραμέτρων).

Στο 9^{ο} κεφάλαιο αναφέρονται οι διαφορές μεταξύ της θεωρητικής και της πρακτικής προσέγγισης (η δεύτερη είναι η ουσιαστική συνεισφορά της δικής μας έρευνας), καθώς και τα συμπεράσματα που καταλήξαμε συγκρίνοντας θεωρητικά αποτελέσματα και πειραματικές μετρήσεις. Τέλος, παρατίθενται οι παρατηρήσεις και τα συμπεράσματα με βάση την έρευνα που έγινε σε αυτή την διπλωματική εργασία.

ΚΕΦΑΛΑΙΟ 1ο : Βασικά στοιχεία για τα Μεγάλα Γλωσσικά Μοντέλα

Στο πρώτο κεφάλαιο γίνεται ιστορική αναδρομή στην τεχνολογία των LLMs, καθώς και κάποιοι βασικοί ορισμοί που θα χρησιμεύσουν στην συνέχεια για καλύτερη κατανόηση της εργασίας.

1.1 Ιστορική αναδρομή

Η μοντελοποίηση της ανθρώπινης γλώσσας σε μεγάλη κλίμακα είναι μια υψηλά πολύπλοκη και απαιτητική μέθοδος που χρειάζεται αρκετούς πόρους για να αναπτυχθεί. Για αυτό τον λόγο η προσπάθεια αυτή έχει διαρκέσει αρκετές δεκαετίες. Ξεκίνησε το 1950 από τον ερευνητή Shanno, ο οποίος εφάρμοσε τη θεωρία της πληροφορίας στην ανθρώπινη γλώσσα και αξιολόγησε πόσο αποτελεσματικά μπορούν τα απλά μοντέλα της γλώσσας να προβλέπουν ή να συμπιέζουν κείμενα φυσικής γλώσσας. Από τότε μέχρι και σήμερα πραγματοποιούνται πάρα πολλές εργασίες κατανόησης και δημιουργίας της φυσικής γλώσσας, όπως η μετάφραση , η αναγνώριση της ομιλίας και άλλα [46].

Επίσης, το έναυσμα για να ολοκληρωθεί όλη αυτή η προσπάθεια της μοντελοποίησης της γλώσσας ξεκίνησε από τη Τεχνητή Νοημοσύνη (ΤΝ) που είναι ένα πεδίο της Επιστήμης των Υπολογιστών που επικεντρώνονται στη δημιουργία και στην παραγωγή λέξεων - κειμένου. Μέσα από την ΤΝ, η Μηχανική Μάθηση είναι ένα πεδίο που επιτρέπει στους υπολογιστές να μαθαίνουν από δεδομένα εισόδου χωρίς συγκεκριμένο προγραμματισμό. Τα μοντέλα της μηχανικής μάθησης χωρίζονται σε δύο κατηγορίες: τα επιβλέποντα μοντέλα και μη επιβλέποντα μοντέλα. Η χαρακτηριστική διαφορά τους είναι ότι τα επιβλέποντα μοντέλα μπορούν να κάνουν προβλέψεις. Τα δεδομένα αυτά μπορεί να είναι ένα όνομα, αριθμός, μία ετικέτα.

Στη συγκεκριμένη διπλωματική εργασία, θα επικεντρωθούμε στα επιβλέποντα μοντέλα. Στα μοντέλα αυτά οι τιμές των δεδομένων εισάγονται και το μοντέλο αυτό κάνει μια πρόβλεψη και τη συγκρίνει με τα δεδομένα που χρησιμοποιήθηκαν για την εκπαίδευση του. Εάν οι τιμές της δοκιμής και οι τιμές της πραγματικής εκπαίδευσης απέχουν, ορίζουμε την διαφορά αυτή ως σφάλμα. Το μοντέλο προσπαθεί να μειώσει το σφάλμα αυτό μέχρι οι τιμές αυτές να γίνουν παραπλήσιες. Αυτό είναι ένα κλασικό θέμα βελτιστοποίησης.

Με την πάροδο του χρόνο οι ερευνητές παρουσίασαν ότι τα μοντέλα της Βαθιάς Μάθησης μπορούν να διακριθούν επίσης σε δύο κατηγορίες Τεχνητής Νοημοσύνης : τα Παραγωγικά και Διακριτικά. Τα Παραγωγικά μοντέλα μπορούν να δημιουργήσουν νέα παραδείγματα δεδομένων, σε αντίθεση με τα Διακριτικά που διακρίνουν διαφορετικά είδη δεδομένων. Ένα Παραγωγικό Μοντέλο μπορεί να πάρει οτιδήποτε διαθέτει σε πληροφορία και δεδομένα και να δημιουργήσει ένα εντελώς καινούριο βασισμένο σε αυτές τις πληροφορίες. Επίσης, η Παραγωγική ΤΝ, η οποία είναι ένα

υποσύνολο της βαθιάς μάθησης, χρησιμοποιεί τεχνητά νευρωνικά δίκτυα και μπορεί να επεξεργαστεί τόσο επισημασμένα όσο και μη επισημασμένα δεδομένα, χρησιμοποιώντας μεθόδους επιβλεπόντων και μη επιβλεπόντων μάθησης. Χαρακτηριστικό παράδειγμα είναι τα Μεγάλα Γλωσσικά Μοντέλα τα οποία είναι ένα υποσύνολο της βαθιάς μάθησης.

Τα Μεγάλα Γλωσσικά Μοντέλα χρησιμεύουν στην κατανόηση των χαρακτήρων, λέξεων και κειμένων. Η Βαθιά Μάθηση περιέχει την ανάλυση πιθανότητας για τα δεδομένα τα οποία δεν είναι δομημένα, τα οποία επιτρέπουν στο μοντέλο της Βαθιάς Μάθησης να αναγνωρίζει τα κομμάτια του κειμένου που του δίνονται χωρίς την ανθρώπινη παρέμβαση.

Για αυτό τον λόγο, ένα καθοριστικό βήμα έκαναν οι ερευνητές για τη μοντελοποίηση της γλώσσας όταν εισήγαγαν τους μετασχηματιστές το 2017. Ο μετασχηματιστής (transformer) είναι ένα συγκεκριμένο είδος νευρωνικού δικτύου, ένα μοντέλο μηχανικής μάθησης και είναι η βασική εφεύρεση που βρίσκεται στο επίκεντρο της Τεχνητής Νοημοσύνης. Αυτό συνέβη γιατί πριν από τους μετασχηματιστές, οι αρχιτεκτονικές που υπήρχαν για την επεξεργασία κειμένου είχαν περιορισμούς στην απόδοση σε μεγάλους όγκους κειμένου.

Το πρώτο μοντέλο μετασχηματιστή δημιουργήθηκε το 2017 από την Google με σκοπό την μετάφραση ενός κειμένου από μία γλώσσα σε μία άλλη. Αυτού του είδους η αρχιτεκτονική επέτρεπε την αντιμετώπιση μακροχρόνιων εξαρτήσεων στο κείμενο με τη χρήση μηχανισμού προσοχής (attention mechanism) [47].

Υπάρχουν πολλά διαφορετικά είδη μοντέλων που μπορούμε να κατασκευάσουμε χρησιμοποιώντας τους μετασχηματιστές. Κάποια από αυτά παρέχουν ήχο, εικόνα, άλλα μπορούν να βοηθήσουν στην σύνθεση κειμένου και άλλα πολλά. Μετά το 2017, η αρχιτεκτονική των μετασχηματισμών έχει εξελιχθεί αρκετά, βελτιώνοντας τους προηγούμενους μηχανισμούς προσοχής, καθώς και τις πιο σύνθετες αρχιτεκτονικές. Για παράδειγμα, το ChatGPT, που είναι ένα από τα πιο σημαντικά μοντέλα των LLMs, έχει εκπαιδευτεί να παίρνει ένα κομμάτι κειμένου και να παράγει μια πρόβλεψη για το τι ακολουθεί στο κείμενο, καθώς καλύπτει ένα μεγάλο εύρος των tasks, όπως ερωτήσεις, περιλήψεις κειμένων, να παρέχει διάφορες πληροφορίες και άλλα πολλά. Αυτή η πρόβλεψη παίρνει την μορφή μιας κατανομής πιθανότητας πάνω σε πολλά διαφορετικά τμήματα κειμένου που θα μπορούσαν να ακολουθήσουν και μπορεί να δημιουργήσει εκθέσεις, ποιήματα, και άλλες διάφορες φόρμες κειμένου που θέλει ο χρήστης.

1.2 Μεγάλα Γλωσσικά Μοντέλα (Large Language Models)

Τα Μεγάλα Γλωσσικά Μοντέλα (LLMs) αναφέρονται σε μεγάλα, γενικής χρήσεως μοντέλα γλώσσας που μπορούν να προ-εκπαιδευθούν και στην συνέχεια να προσαρμοστούν για

συγκεκριμένους σκοπούς. Αυτά τα μοντέλα είναι εξειδικευμένα για γενικούς σκοπούς προκειμένου να λύσουν κοινά προβλήματα γλώσσας όπως η ταξινόμηση κειμένου, η απάντηση ερωτήσεων, η περίληψη παραγράφων και η δημιουργία κειμένου σε διάφορους τομείς. Μπορούν να λύσουν συγκεκριμένα προβλήματα σε διάφορους τομείς, όπως η χρηματοοικονομική, χρησιμοποιώντας μικρομεσαίου μεγέθους σύνολα δεδομένων ανά τομέα.

Ακόμη, ονομάζονται «γενικής χρήσεως» διότι είναι μοντέλα που μπορούν να επιλύσουν κοινά προβλήματα. Επίσης, λέγονται και «μεγάλα» λόγω του τεράστιου μεγέθους του συνόλου παραδειγμάτων εκπαίδευσης που απαιτούν, αλλά και του μεγάλου αριθμού των παραμέτρων τους. Τα LLMs περιέχουν πολλαπλά επίπεδα νευρωνικού δικτύου και διαθέτουν παραμέτρους που μπορούν να ρυθμιστούν κατά την εκπαίδευση του μοντέλου αυτού. Κατά την εκπαίδευση, αυτά τα μοντέλα μαθαίνουν να προβλέπουν την επόμενη λέξη σε μια πρόταση.

Ο όρος ‘Μεγάλα’ αναφέρεται στις ελεύθερες παραμέτρους που χρησιμοποιούνται στα Μεγάλα Γλωσσικά Μοντέλα, καθώς αυτός μπορεί να είναι τεράστιος. Αυτό οφείλεται στην πολυπλοκότητα της αρχιτεκτονικής του μοντέλου και της διάστασης που μπορεί να πάρει το κάθε μοντέλο. Επίσης, κάθε παράμετρος αντιστοιχεί σε ένα βάρος που προσαρμόζεται κατά την διάρκεια της εκπαίδευσης του μοντέλου αυτού και των πραγματικών τιμών που μπορεί να πάρει. Επομένως, ο αριθμός παραμέτρων είναι σημαντικός δείκτης της πολυπλοκότητας και της χωρητικότητας του μοντέλου και μπορεί να επηρεάσει την απόδοση σε διάφορα προβλήματα που μπορεί να προκύψουν.

Έτσι, λοιπόν, βάρος ονομάζουμε τις παραμέτρους που προσαρμόζονται κατά την διάρκεια της εκπαίδευσης των μεγάλων γλωσσικών μοντέλων για να προβλέψουν τις επόμενες λέξεις ή τις πιθανότητες ενός κειμένου. Κάθε βάρος αντιστοιχεί σε συγκεκριμένη σύνδεση μεταξύ νευρώνων, τα διάφορα επίπεδα του δικτύου των LLM [52].

Τέλος, τα LLMs χρησιμοποιούν την αρχιτεκτονική του μετασχηματιστή. Το μοντέλο του μετασχηματιστή αποτελείται από δύο βασικά μέρη, τον κωδικοποιητή (encoder) και τον αποκωδικοποιητή (decoder). Ο κωδικοποιητής αποτελείται από μια στοιβάδα των 6 επιπέδων και κάθε επίπεδο έχει δύο υπό-επίπεδα. Το πρώτο είναι το επίπεδο του Multi-Head Self-Attention, ενώ το δεύτερο είναι ένα πιο απλό επίπεδο το οποίο είναι πλήρως συνδεδεμένο δίκτυο για τη προώθηση θέσης. Από την άλλη πλευρά, ο αποκωδικοποιητής αποτελείται από μια στοιβάδα 6 πανομοιότυπων επιπέδων. Εκτός από τα υπό-επίπεδα που διαθέτει όπως και ο κωδικοποιητής, περιλαμβάνει ένα τρίτο υπό-επίπεδο που εκτελεί την προσοχή πολλαπλών κεφαλών πάνω στην έξοδο της στοίβας του κωδικοποιητή.

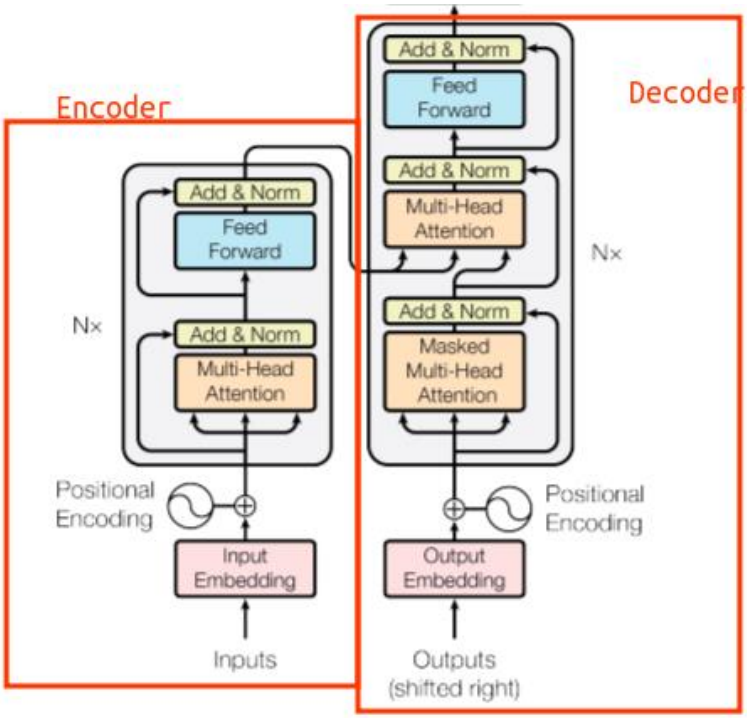
Η λειτουργία της προσοχής χαρτογραφεί ένα ερώτημα και ένα σύνολο ζευγών κλειδιού-τιμής σε μια έξοδο, όπου όλα είναι διανύσματα. Το αποτέλεσμα υπολογίζεται ως ένα σταθμισμένο

άθροισμα των τιμών, με κάθε τιμή να αποδίδεται βάρος που καθορίζεται από μια συνάρτηση συμβατότητας του ερωτήματος με το σχετικό κλειδί. Τέλος, η κωδικοποίηση θέσης χρησιμοποιείται για να συνδυάσει τις πληροφορίες σχετικά με τις θέσεις των χαρακτήρων που βρίσκονται στη σειρά.

1.3 Κωδικοποιητής-Αποκωδικοποιητής(Encoder-Only, Decoder-Only)

Η αρχιτεκτονική κωδικοποιητή-αποκωδικοποιητή αποτελεί τον πυρήνα των Μεγάλων Μοντέλων Γλώσσας και είναι μια αρχιτεκτονική που δέχεται και παράγει ακολουθίες. Αυτή η αρχιτεκτονική αποτελείται από δύο στάδια. Το πρώτο είναι το στάδιο του κωδικοποιητή. Σε αυτό το στάδιο, η είσοδος (για παράδειγμα σε μια φυσική γλώσσα) μετατρέπεται σε μια αναπαράσταση διανυσματικού χώρου (vector representation) μέσω του κωδικοποιητή. Αυτή η αναπαράσταση περιέχει πληροφορίες σχετικά με την σημασία της εισόδου.

Το δεύτερο στάδιο είναι το στάδιο του αποκωδικοποιητή. Σε αυτό το στάδιο, ο αποκωδικοποιητής λαμβάνει την αναπαράσταση που παράχθηκε από τον κωδικοποιητή και τη χρησιμοποιεί για να δημιουργήσει μια ακολουθία εξόδου, όπως μια μετάφραση σε μια άλλη γλώσσα.

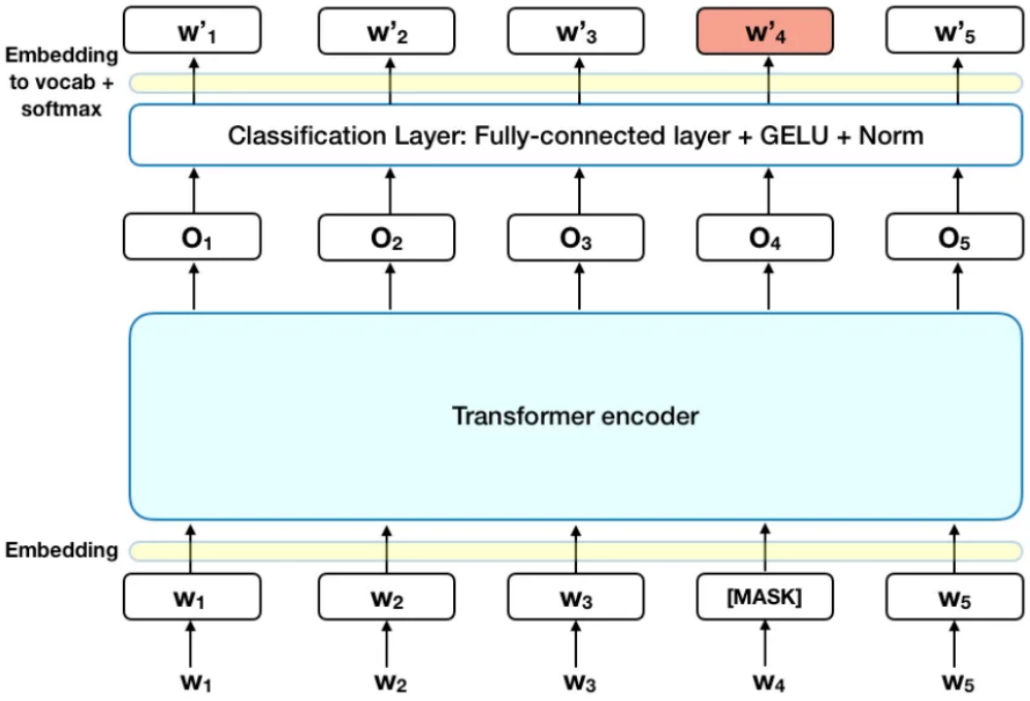


Εικόνα 2 : Γράφημα συνδεσμολογίας κωδικοποιητή - αποκωδικοποιητή

Πηγή: https://vaclavkosar.com/ml/Encoder-only-Decoder-only-vs-Encoder-Decoder-Transfomer

Δύο χαρακτηριστικά παραδείγματα που χρησιμοποιούν την αρχιτεκτονική του κωδικοποιητή και του αποκωδικοποιητή είναι το BERT και το GPT αντίστοιχα. Το μοντέλο BERT (Bidirectional Encoder Representations from Transformers) [56] είναι βασισμένο σε μετασχηματιστές για να

κατανοήσει και να παράγει την ανθρώπινη γλώσσα. Χρησιμοποιεί την αρχιτεκτονική του κωδικοποιητή, οποίο περιέχει Multi-Head και Self-Attention. Το μοντέλο αυτό είναι σχεδιασμένο για την κατανόηση κειμένου, το οποίο κάθε token που έχει αντιστοιχείται σε όλα τα tokens που βρίσκονται σε σειρά, παρέχοντας έτσι μια πιο βαθιά κατανόηση του πλαισίου. Με άλλα λόγια το μοντέλο αυτό υποδηλώνει έναν πρωταρχικό στόχο στη κατανόηση των εισερχομένων ακολουθιών παρά στην παραγωγή των εξερχομένων ακολουθιών.



Εικόνα 3 : Μοντέλο BERT

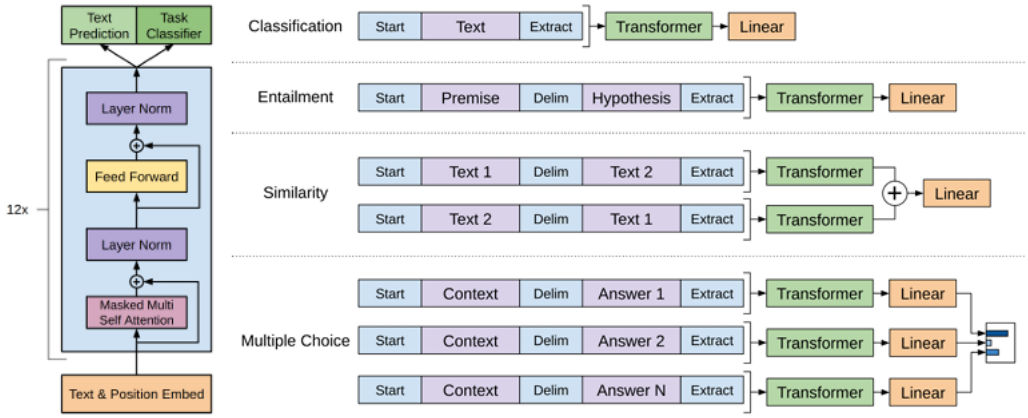
Πηγή : BERT Explained: State of the art language model for NLP | by Rani Horev | Towards Data Science

Το μοντέλο GPT , το οποίο έχει αναπτυχθεί από την OpenAI, ανήκει στον τομέα του νευρωνικού δικτύου που χρησιμοποιούν την αρχιτεκτονική των επιταχυντών και αποτελεί σημαντική πρόοδο στην τεχνική νοημοσύνη (AI), δίνοντας έμφαση σε εφαρμογές όπως το ChatGPT.

Το GPT χρησιμοποιεί μόνο τον αποκωδικοποιητή από το μοντέλο των Transformers. Χρησιμοποιεί συσσωρευμένα block του αποκωδικοποιητή. Κάθε block περιλαμβάνει το Self-Attention και είναι σχεδιασμένο με διάφορες λειτουργίες, προβλέποντας το επόμενο token σε μια πρόταση που έχει δοθεί από προηγούμενα tokens. Έτσι, αυτό το μοντέλο δίνει στις εφαρμογές την δυνατότητα να δημιουργήσουν ανθρώπινο κείμενο και περιεχόμενο, όπως εικόνες, μουσική και άλλα, καθώς και να απαντούν σε ερωτήσεις. Τέλος, είναι σημαντικό να αναφερθεί ότι πολλοί οργανισμοί

σε διάφορους κλάδους χρησιμοποιούν τα μοντέλα όπως το GPT για να δημιουργήσουν κείμενα και περιλήψεις.

Η τελευταία έκδοση του GPT είναι η GPT-4, η οποία είναι ένα πολύ ισχυρό μοντέλο των LLMs, καθώς τον Μάρτιο του 2024 έδωσαν οι ερευνητές στους χρήστες την δυνατότητα να μπορούν να βάζουν εικόνες και κείμενο ως είσοδο και το μοντέλο αυτό να παράγει ως έξοδο ένα κείμενο. Παρόλο που εξακολουθεί να μην είναι τόσο ικανό όπως ένας άνθρωπος , το μοντέλο αυτό πλησιάζει την ανθρώπινη απόδοση και μπορεί να βοηθήσει πάρα πολύ τόσο στον εκπαιδευτικό τομέα, όσο και στον επαγγελματικό.



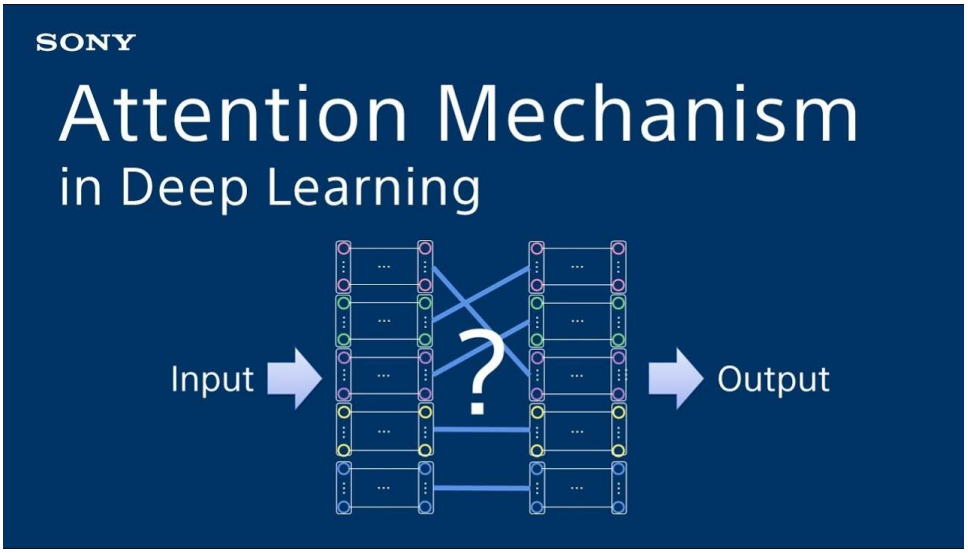
Εικόνα 4 : Μοντέλο GPT

Πηγή : language_understanding_paper.pdf

1.4 Μηχανισμός Προσοχής(Attention Mechanism)

Ακόμη, στην αρχιτεκτονική των επιταχυντών υπάρχει ο Μηχανικός Προσοχής. Ο μηχανισμός Προσοχής είναι ένα σημαντικό στοιχείο που χρησιμοποιείται αρκετά στις σύγχρονες αρχιτεκτονικές Μηχανικής Μάθησης, όπως τους μετασχηματιστές, έχοντας ως σκοπό τη βελτίωση της απόδοσης σε εφαρμογές που απαιτούν την επεξεργασία ακολουθιών, όπως η μετάφραση φυσικών γλωσσών και η παραγωγή κειμένου. Έτσι, ο μηχανισμός αυτός συνδέεται άμεσα τόσο με τον κωδικοποιητή όσο και με τον αποκωδικοποιητή. Αυτό συμβαίνει γιατί κατά την διάρκεια της κωδικοποίησης, ο μηχανισμός εφαρμόζεται πάνω στην είσοδο και κάθε τμήμα του έχει συσχετιστεί με ένα σύνολο βαρών, που αναπαριστούν την σημασιολογική σημασία του. Ο κωδικοποιητής χρησιμοποιεί αυτά τα βάρη προσοχής για να δημιουργήσει μια ενσωματωμένη αναπαράσταση της εισόδου, η οποία περιέχει πληροφορίες σχετικά με την είσοδο. Επιπλέον, συνδέεται με τον αποκωδικοποιητή, γιατί κατά την διάρκεια της αποκωδικοποίησης, ο μηχανισμός προσοχής χρησιμοποιείται για να δώσει βάρος στα

διαφορετικά τμήματα της ενσωματωμένης αναπαράστασης εισόδου. Σε κάθε βήμα αποκωδικοποίησης, επιλέγονται τμήματα της εισόδου που είναι σημαντικά για την τρέχουσα έξοδο που παράγεται.



Εικόνα 5 : Μηχανισμός Προσοχής

Πηγή:https://www.kaggle.com/code/jeongwonkim10516/attention-mechanism-for-nlp-beginners

Ο μηχανισμός προσοχής, επίσης, αποτελείται από δυο περαιτέρω μηχανισμούς : τον Multi-Head Attention και τον Self-Attention Mechanism. Ο πρώτος μηχανισμός εστιάζει τη προσοχή του σε διαφορετικά σημεία της εισόδου και ταυτόχρονα επιτρέπει στο μοντέλο να αναγνωρίζει πιο πολύπλοκα μοτίβα και συσχετίσεις μεταξύ των δεδομένων εισόδου. Ο δεύτερος, χρησιμοποιείται για τη καταγραφή εξαρτήσεων και σχέσεων που υπάρχουν μεταξύ των tokens ανεξαρτήτως της απόστασης που βρίσκονται. Αυτό γίνεται χρησιμοποιώντας τρεις πίνακες : το ερώτημα Q, το κλειδί K και την τιμή V. Χρησιμοποιώντας αυτούς τους πίνακες μπορούμε να καθορίσουμε πόση προσοχή πρέπει να δίνει κάθε token σε ένα άλλο, ώστε να βελτιώσει την ποιότητα της μετάφρασης ή οτιδήποτε άλλου που είναι βασισμένο στις ακολουθίες αυτές.

1.5 Πλεονεκτήματα και περιορισμοί των LLMs

Ένα κύριο χαρακτηριστικό των μοντέλων αυτών είναι η ικανότητα να ανταποκρίνονται σε απρόβλεπτες ερωτήσεις. Ένα απλό πρόγραμμα υπολογιστή λαμβάνει εντολές στην αποδεκτή σύνταξη του ή από ένα συγκεκριμένο σύνολο εισόδων από τον χρήστη. Αντίθετα, ένα LLM μπορεί να ανταποκριθεί σε φυσική γλώσσα και να χρησιμοποιήσει ανάλυση δεδομένων για να απαντήσει σε μια ερώτηση, δίνοντας μια λογική αιτιολόγηση στον χρήστη.

Όσο αφορά τις πληροφορίες που παρέχουν, τα LLMs μπορούν να είναι αξιόπιστα μόνο όσα μπορέσουν να απορροφήσουν δεδομένα. Αν έχουν ψευδή πληροφορίες, θα δώσουν λάθος ενημέρωση σε αυτό που ζητάει ο χρήστης. Επίσης, είναι σημαντικό να αναφερθεί πως τα μοντέλα αυτά κάποιες φορές δημιουργούν ψεύτικες πληροφορίες όταν δε μπορούν να παράγουν μια ακριβή απάντηση. Χαρακτηριστικό παράδειγμα είναι πως το 2022 το περιοδικό Fast Company ρώτησε το ChatGPT για την οικονομική κατάσταση της εταιρείας Tesla και έδωσε ως απάντηση πολλές πληροφορίες που τις δημιούργησε εκείνη τη στιγμή.

Επιπλέον, όσο αφορά την ασφάλεια, υπάρχουν εφαρμογές που βασίζονται σε LLMs και είναι ευάλωτες σε σφάλματα όπως κάθε άλλη εφαρμογή. Τα LLMs μπορούν επίσης να είναι υπόδειγμα από τις κακόβουλες εισόδους τους για να παρέχουν συγκεκριμένους τύπους απαντήσεων έναντι άλλων όπως για παράδειγμα ηθικά ζητήματα. Ακόμη , ένα από τα προβλήματα ασφάλειας που προκαλούν τα LLM είναι πως οι χρήστες έχουν τη δυνατότητα να ανεβάζουν ασφαλή και εμπιστευτικά δεδομένα προκειμένου να αυξήσουν τη παραγωγικότητα τους. Όμως, επειδή τα Μεγάλα Γλωσσικά Μοντέλα χρησιμοποιούν τις εισόδους που λαμβάνουν για να εκπαιδεύσουν τα μοντέλα τους, δεν έχουν σχεδιαστεί για να έχουν ασφαλή αποθήκευση και έτσι μπορεί να αποκαλύψουν εμπιστευτικά δεδομένα ως απάντηση σε ερώτηση των χρηστών. Ακόμη, είναι σχολαστικά και πιθανολογικά μοντέλα, για αυτό τον λόγο αν ο χρήστης στείλει την ίδια πρόταση πολλές φορές είναι πιθανό να λάβουμε διαφορετικές απαντήσεις. Τέλος , διαθέτουν παλιές πληροφορίες και από μόνα τους δεν μπορούν να έχουν πρόσβαση σε εξωτερικά δεδομένα, όπως είναι και πολύ «Μεγάλα», καθώς απαιτούνται πολλές χρήσεις της GPU για να εκπαιδευτεί αυτό το μοντέλο και να εξυπηρετήσει τον κάθε χρήστη.

1.6 Προκλήσεις και μελλοντικές κατευθύνσεις των LLMs

Όπως είναι γνωστό, τα Μεγάλα Γλωσσικά Μοντέλα έχουν πετύχει μια μεγάλη πρόοδο τα τελευταία χρόνια και έτσι έχουν δημιουργήσει πολλές προκλήσεις και μελλοντικούς στόχους για τη βελτίωση τους. Για αυτό τον λόγο, γίνεται τεράστια προσπάθεια για να βελτιωθούν και να αυξηθούν αυτά τα μοντέλα, παρόλο που υπάρχουν πολλές δυσκολίες. Για παράδειγμα, μια σημαντική πρόκληση που αντιμετωπίζουν οι ερευνητές είναι ότι τα Μεγάλα Γλωσσικά Μοντέλα να μπορέσουν να μετατραπούν σε Μικρά Γλωσσικά Μοντέλα (Small Language Models – SLM), η οποία θα αποτελέσει μια εναλλακτική των LLMs, ιδιαίτερα όταν χρησιμοποιούνται σε συγκριμένα καθήκοντα που μπορεί να μην απαιτούν μια πλήρη γενικότητα των Μεγάλων Μοντέλων. Επίσης, στο μέλλον οι ερευνητές φιλοδοξούν να έχουν ένα Multi-Modal Model, το οποίο να μπορεί να διαχειριστεί ένα ευρύ φάσμα δεδομένων όπως: διάφορους τύπους κειμένου, εικόνων, βίντεο, ήχου και άλλα. Αυτό έχει ως αποτέλεσμα να δημιουργηθούν διάφορες και πιο γενικές εφαρμογές σε τομείς όπου θα μπορούν να χρησιμοποιηθούν τα δεδομένα αυτά, όπως για παράδειγμα στη ρομποτική.

Τέλος, κάποιες σημαντικές προκλήσεις που αντιμετωπίζουν τα μοντέλα αυτά είναι το πόσο ασφαλής είναι και τα ηθικά ζητήματα που μπορεί να αντιμετωπίσει. Όσο αφορά τη διασφάλιση της ανθεκτικότητας και η ασφάλεια των LLMs από κακόβουλες επιθέσεις είναι μια πολύ σημαντική πρόκληση που πρέπει να αντιμετωπίσουν. Αυτό συμβαίνει γιατί τα LLMs χρησιμοποιούνται σε πραγματικές εφαρμογές και πρέπει να προστατεύονται από επιθέσεις για να αποφευχθεί η εκμετάλλευση τους, κυρίως όσο αφορά την παραπληροφόρηση. Επίσης, η αντιμετώπιση των ηθικών προβλημάτων είναι μια άλλη πρόκληση που αντιμετωπίζουν τα μοντέλα αυτά, καθώς γίνονται τεράστιες προσπάθειες για να είναι αντικειμενικά τα μοντέλα αυτά, πιο δίκαια και ικανά να μπορούν να χειρίζονται ένα ευαίσθητο υλικό με τον κατάλληλο τρόπο, καθώς ένα αρκετά μεγάλο ποσοστό ανθρώπων χρησιμοποιούν τα LLMs καθημερινά.

ΚΕΦΑΛΑΙΟ 2ο : Διαφορές επιταχυντών και πράξεις πινάκων

Στο κεφάλαιο αυτό θα επικεντρωθούμε στις τέσσερις διαφορετικές κατηγορίες των επιταχυντών: FPGA, ASIC, In-memory, GPU, στις διαφορές τους καθώς και σε μελέτες που έχουν γίνει με βάση μοντέλα που έχουν παρουσιαστεί τα τελευταία χρόνια. Επίσης, θα δώσουμε έμφαση σε τι μέθοδο χρησιμοποιεί το κάθε μοντέλο (σ.σ. πολλαπλασιασμό, αφαίρεση, διαίρεση, συμπίεση πινάκων).

2.1 Διαφορές μεταξύ FPGA, ASIC, In-Memory, GPU

Τα FPGA, ASIC, In-Memory και GPU είναι τέσσερα είδη επιταχυντών που χρησιμοποιούνται για την βελτίωση της απόδοσης των μοντέλων, βασίζονται σε μετασχηματιστές και κυρίως έχουν στόχο την επεξεργασία της φυσικής γλώσσας.

Τα FPGAs, είναι προγραμματισμένες συσκευές υλικού που μπορούν να τροποποιηθούν μετά την παραγωγή. Χρησιμοποιούν μια παράλληλη επεξεργασία για την ταυτόχρονη εκτέλεση διεργασιών, με αποτέλεσμα την υψηλή απόδοση και την χαμηλή κατανάλωση ενέργειας.

Οι μονάδες επεξεργασίας γραφικών (GPU) είναι γνωστές για τις ικανότητες τους στην παράλληλη επεξεργασία, και κυρίως για τη δυνατότητα τους να εκτελούν πράξεις πινάκων που απαιτούνται για τους υπολογισμούς νευρωνικών δικτύων. Γι’ αυτό τον λόγο, έχουν εξελιχθεί σε επιταχυντές γενικής χρήσεως, καθώς χρησιμοποιούνται συχνά σε εφαρμογές βαθιάς μάθησης.

Τα συστήματα υπολογισμού στη μνήμη (In-Memory) χρησιμοποιούν την ικανότητα τους να βρίσκονται κοντά στην μνήμη και τις μονάδες επεξεργασίας για να εκτελέσουν υπολογισμούς απευθείας στα αποθηκευμένα δεδομένα, μειώνοντας έτσι τη μεταφορά δεδομένων με χαμηλή καθυστέρηση. Αυτή η μέθοδος προσφέρει εξοικονόμηση ενέργειας μειώνοντας την ανάγκη μεταφοράς δεδομένων μεταξύ μνήμης και μονάδας επεξεργασίας.

Τα κυκλώματα ειδικών εφαρμογών (ASIC) είναι ολοκληρωμένα κυκλώματα που σχεδιάζονται ειδικά για συγκεκριμένους σκοπούς, προσφέροντας μέγιστη απόδοση και αποτελεσματικότητα ενέργειας. Αν και για να δημιουργηθούν αυτά τα κυκλώματα είναι αρκετά ακριβά, παρέχουν ασυναγώνιστη απόδοση για συγκεκριμένες εργασίες χάρη σε εξειδικευμένο υλικό για βασικές διεργασίες όπως μηχανισμοί προσοχής και τον πολλαπλασιασμό πινάκων.

Συνεπώς, η επιλογή του κατάλληλου επιταχυντή καθορίζεται από παράγοντες όπως η απόδοση, η ενεργειακή αποδοτικότητα και οι διαθέσιμοι πόροι που έχουμε. Τα FPGA προσφέρουν ευελιξία και οικονομία ενέργειας, ενώ τα GPU ξεχωρίζουν στον παράλληλο υπολογισμό. Η υπολογιστική στη μνήμη βελτιώνει τη ροή δεδομένων, αλλά τα ASIC προσφέρουν ανώτερη απόδοση σε μεγαλύτερο

κόστος. Κάθε είδος επιταχυντή προσφέρει συγκεκριμένα πλεονεκτήματα και σκέψεις, που επηρεάζουν τη χρησιμότητα τους για διάφορες εφαρμογές μέσα σε μοντέλα που βασίζονται σε μετασχηματιστές και σε άλλες υπολογιστικές δραστηριότητες.

2.2 Πράξεις πινάκων

Στα μεγάλα γλωσσικά μοντέλα χρησιμοποιούνται διάφορες πράξεις πινάκων, όπως ο πολλαπλασιασμός, η πρόσθεση, η αφαίρεση, η διαίρεση και η συμπίεση. Κάθε πράξη έχει διαφορετική εφαρμογή σε αυτά τα μοντέλα. Αναλόγως τις πράξεις πινάκων που εκτελούν τα διάφορα μοντέλα, έχουν ως στόχο να χάνουν ελάχιστα στην ακρίβεια, αλλά να κερδίζουν στη καθυστέρηση και στην ταχύτητα.

2.2.1 Πολλαπλασιασμός – ποσοτικοποίηση

Για αρχή, ο πολλαπλασιασμό πινάκων(GEMM), είναι η πιο σύνηθες και βασική πράξη που εκτελείται στο πλαίσιο του μηχανισμού μάθησης. Είναι μια τεχνική που βοηθάει στο να μειωθεί το μέγεθος των μεγάλων νευρικών δικτύων με τροποποίηση των βαρών συμπεριλαμβάνοντας και τα μεγάλα γλωσσικά μοντέλα. Τα μοντέλα αυτά, αναφέρονται στη μετατροπή των βαρών από τύπους υψηλότερης ακρίβειας δεδομένων σε τύπους χαμηλότερης ακρίβειας. Ουσιαστικά, επειδή τα νευρωτικά δίκτυα είναι μοντέλα που αναπαρίστανται στη μνήμη της GPU ή της μνήμης RAM και δημιουργούνται πολυδιάστατοι πίνακες αριθμών. Για την αποθήκευσή τους χρησιμοποιούνται διαφορετικοί τύποι δεδομένων όπως Float64, Float16 ή ακόμα και ακέραιοι αριθμοί, αλλά και ο τύπος δεδομένων που επιλέγουμε για να χρησιμοποιηθούν στη μνήμη. Επίσης, το μέγεθος που χρησιμοποιείται μπορεί να αναφερθεί στη βιβλιογραφία ως ακρίβεια, υποδεικνύοντας πόσα ψηφία χρησιμοποιούνται για να αναπαρασταθούν στη μνήμη. Ένα συνηθισμένο παράδειγμα είναι η μείωση των βαρών από FP16 σε INT4 (4-bit ακέραιο) που φαίνεται τα τελευταία χρόνια να είναι η βέλτιστη λύση μεταξύ απόδοσης και μεγέθους- ταχύτητας για αυτά τα μοντέλα, επιτρέποντας να λειτουργήσουν με φθηνότερο υλικό και μεγαλύτερες ταχύτητες.

2.2.2 Συμπίεση

Η συμπίεση είναι η δεύτερη πιο σύνηθες τεχνική που χρησιμοποιείται στα μεγάλα γλωσσικά μοντέλα. Αναφέρεται στη διαδικασία της μείωσης του μεγέθους του μοντέλου χωρίς να μειώνεται δραματικά η απόδοση του μοντέλου. Η συμπίεση μπορεί να εφαρμοστεί σε διάφορα στάδια, συμπεριλαμβανόμενων των βαρών του, του επιπέδου του και των αρχιτεκτονικών του. Υπάρχουν διάφορες τεχνικές συμπίεσης όπως η κβαντοποίηση των βαρών, η αποκοπή συνδέσεων, οι αραιές αναπαραστάσεις. Για παράδειγμα, η κβαντοποίηση των βαρών χρησιμεύει για την αποθήκευση των

βαρών ως πραγματικοί αριθμοί και μπορούν να κβαντοποιηθούν σε ακέραιους αριθμούς ή σε μικρότερα μεγέθη κινητής υποδιαστολής.

2.2.3 Πρόσθεση - αφαίρεση

Με την πρόσθεση των πινάκων μπορούν να συνδυαστούν πληροφορίες από διαφορετικές πηγές, όπως οι εξωτερικές γνώσεις, εκφράσεις από διαφορετικά κείμενα. Αυτό δίνει την δυνατότητα στον επιταχυντή να βελτιώσει την αναπαράσταση του μοντέλου και να προσθέσει πληροφορίες που μπορεί αρχικά να μην είχα περιληφθεί.

Η αφαίρεση πινάκων δίνει τη δυνατότητα να εξαχθούν συγκεκριμένες πληροφορίες ή χαρακτηριστικά από ένα μοντέλο. Αυτό μπορεί να χρησιμοποιηθεί για να μειωθεί η διάσταση του μοντέλου ή για να επιλεχθούν μόνο πληροφορίες που είναι σημαντικές για μια συγκεκριμένη εφαρμογή.

Γενικά, έχουν αρκετά πλεονεκτήματα. Για παράδειγμα, γίνεται εμπλουτισμός του μοντέλου με επιπλέον πληροφορίες, υπάρχει ευελιξία στην αναπαράσταση διαφορετικών πτυχών και βελτίωση της επίδοσης σε πολύπλοκες εφαρμογές.

2.2.4 Softmax

Το Softmax είναι μια συνάρτηση που έχει τη δυνατότητα να μετατρέψει τα αποτελέσματα ενός μοντέλου σε πιθανότητες. Συνήθως, χρησιμοποιείται στο τέλος ενός μοντέλου για να παράγει την τελική πιθανότητα. Η εξίσωση του Softmax φαίνεται παρακάτω:

$$\text{Softmax(xi)}= \frac{e^{xi/T}}{\Sigma_j e^{xj/T}} \ (1)$$

Πηγή : [2402.06196] Large Language Models: A Survey (arxiv.org)

Στην εξίσωση παρατηρούμε ότι υπάρχει η μεταβλητή Τ. Αυτή η μεταβλητή είναι η θερμοκρασία Τ που είναι μια παράμετρος που κυμαίνεται από το 0 έως 1 και επηρεάζει τις πιθανότητες που παράγονται από την συνάρτηση Softmax. Αν υπάρχει χαμηλή θερμοκρασία, μεταβάλλεται σημαντικά η κατανομή των πιθανοτήτων, ενώ η υψηλή θερμοκρασία δίνει προτεραιότητα στα tokens με υψηλή πιθανότητα.

ΚΕΦΑΛΑΙΟ 3ο : Βιβλιογραφική μελέτη των FPGAs

Τα τελευταία χρόνια έχουν αναπτυχθεί διάφορα ψηφιακά μοντέλα επεξεργαστών όπως FPGA. Από το 2020 έως και το 2024 έχουν εκτελεστεί πάνω από 15 διαφορετικά. Ξεκινώντας έτσι, από το 2020, και πηγαίνοντας στο 2024 θα εξετάσουμε την διαχρονική εξέλιξή τους, αναφέροντας τα πιο σημαντικά μοντέλα επιταχυντών.

3.1 FTRANS

Το FTRANS [1] παρουσιάστηκε από τον ερευνητή Binghigh Li το 2020 και είχε σκοπό την επιτάχυνση σε μεγάλη γλωσσική κλίμακα, η οποία είναι βασισμένη σε μετασχηματιστές και γίνεται αναφορά για την αρχιτεκτονική άλλα και για τον αλγόριθμο που χρησιμοποιεί. Πιο συγκεκριμένα επικεντρώνεται στη συμπίεση και στο σχεδιασμό επιτάχυνσης για την αντιμετώπιση υπολογιστών και αποθηκευτικών απαιτήσεων. Επίσης, το μοντέλο αυτό επιταχύνει έναν συντελεστή συμπίεσης έως 16 φορές με ελάχιστη επιδείνωση της ακρίβειας χρησιμοποιώντας ένα εξελιγμένο μοντέλο βάρους βασισμένο σε κυκλικό κώδικα μπλοκ (BCM) και για βελτιώσουν την καθυστέρηση τονίζονται μια στρατηγική βελτίωσης. Για αυτό τον λόγο, το μέγεθος του μοντέλου ξεπερνάει τις υλοποιήσεις των CPU και GPU όσο αφορά την επιτάχυνση και την ενεργειακή απόδοση. Σε σύγκριση με άλλες εφαρμογές το Ftrans, βελτιώνει την ταχύτητα και την ενεργειακή απόδοση κατά 81 και 9 φορές αντιστοίχως. Στη συγκεκριμένη μελέτη γίνεται σύγκριση με τον επεξεργαστή GPU RTX5000, χρησιμοποιώντας VCU118 (16nm). Τέλος, αναφέρεται ότι ο επιταχυντής αυτός έχει ρυθμό απόδοσης 170 GOPs και ρυθμό ενεργειακής απόδοσης 6.8 GOPs/W.

3.2 Multi-Head Attention

Το δεύτερο μοντέλο επιτάχυνσης που δημοσιεύτηκε το 2020 αφορά το Multi-Head Attention[3] και δημοσιεύτηκε από τον ερευνητή Siyuan Lu. Η μελέτη αυτή αναφέρεται σε έναν επιταχυντή υλικού για δύο κρίσιμα συστατικά μοντέλου του μετασχηματιστή. Αυτά είναι τα πολλαπλά κεφάλαια συστημάτων παροχής MHA ResBlock και το δίκτυο προώθησης θέσης FFN ResBlock, τα οποία αποτελούν και το κύριο μέρος του σχεδιασμού. Χρησιμοποιεί ως μέθοδο τη διαίρεση των πινάκων, με αποτέλεσμα να δώσει τη δυνατότητα στο ResBlock να μειώσουν τα στοιχεία που διαθέτουν. Επιπλέον, το μοντέλο αυτό αυξάνει την αποδοτικότητά του, η οποία χρησιμοποιείται σε εργασίες επεξεργασίας φυσικής γλώσσας (NLP). Για αυτό τον λόγο η υπολογιστική ροή είναι καλά σχεδιασμένη για να μεγιστοποιήσει την χρήση του πίνακα και οι μη γραμμικές λειτουργίες βοηθάνε στην ελαχιστοποίηση της πολυπλοκότητας και της καθυστέρησης. Τέλος, σε σύγκριση με CPU και GPU και το μοντέλο αυτό βελτιώνει τη ταχύτητα 16 φορές, χρησιμοποιώντας ως σύγκριση τον επεξεργαστή Nvidia V100 και τη τεχνολογία XCVU13p(14nm).

3.3 NPE

Το 2021, παρουσιάστηκε από τον ερευνητή Hamza Khan το μοντέλο επιταχυντή με ονομασία NPE[5], το οποίο είναι ένας επεξεργαστής επικάλυψης για τις δραστηριότητες της φυσικής γλώσσας (NLP) και είναι βασισμένο σε μετασχηματιστές όπως το BERT. Ο προγραμματισμός του επιταχυντή είναι αρκετά εύκολος και μπορεί να εκτελέσει αρκετά γρήγορα μια ποικιλία μοντέλων NLP χωρίς να γίνει κάποια συγκεκριμένη διαμόρφωση. Η μέθοδος που χρησιμοποιείται είναι ο πολλαπλασιασμός πινάκων για να μειώσει την υπολογιστική πολυπλοκότητα. Έχει δυνατότητες πραγματικού χρόνου με σημαντικά χαμηλή κατανάλωση ενέργειας σε σύγκριση με άλλους επεξεργαστές GPU και CPU. Ο επεξεργαστής χρησιμοποιεί μόνο μια τεχνική για να προσεγγίσει αρκετές μη γραμμικές συναρτήσεις, εξασφαλίζοντας έτσι την ευελιξία για μελλοντικά μοντέλα NLP. Σε σύγκριση με CPU και GPU, το NPE είναι 35 φορές πιο γρήγορα και 6 φορές πιο ενεργειακά αποδοτικός σε σχέση με άλλα μοντέλα, χρησιμοποιώντας ως επεξεργαστή σύγκρισης RTX5000 και την τεχνολογία VCU188(16nm).

3.4 Accommodating Transformer onto FPGA

Το επόμενο μοντέλο επιτάχυνσης που παρουσιάστηκε το 2021 είναι από τον ερευνητή Panje Qi με όνομα "Accommodating Transformer onto FPGA"[6], ο οποίος πρότεινε δύο τρόπους για να βελτιώσει τη ταχύτητα εκτέλεσης μοντέλων μετασχηματιστών σε FPGA. Ο πρώτος τρόπος μειώνει τον αριθμό των υπολογισμών που χρειάζεται το μοντέλο να πραγματοποιήσει. Ο δεύτερος τρόπος είναι να βελτιωθεί το υλικό που διαθέτουν και αφορά το σχεδιασμό ενός ειδικού τμήματος FPGA και δίνει την δυνατότητα να εκτελέσει το μοντέλο τη συμπίεση. Εκμεταλλευόμενος την συμπίεση αυτή, πετυχαίνετε καλύτερη απόδοση στη ταχύτητα. Έτσι, γίνεται σύγκριση με άλλα μοντέλα επιταχυντών και καταφέρνουν να πετύχουν 38 φορές καλύτερη ταχύτητα και 2 φορές καλύτερη ενεργειακή απόδοση, συγκρίνοντας το κυρίως με τον επεξεργαστή της Nvidia Guardo RTX 6000 και με τη τεχνολογία Alveo U200 (16nm).

3.5 Honqwu Peng

Ακόμη, το 2021, παρουσιάστηκε από τον Honqwu Peng[6] μια μέθοδος για τη βελτιστοποίηση των μοντέλων επιταχυντών σε FPGA χρησιμοποιώντας την αποκοπή στηλών με ισορροπία. Πιο συγκεκριμένα, γίνεται δημιουργία μιας προσέγγισης αποκοπής στηλών μπλοκ με ισορροπία, σχεδιασμό υλικού για την επιτάχυνση αραιών πολλαπλασιασμών πινάκων και στρατηγικές προγραμματισμού πόρων για υψηλό παραλληλισμό και ροή δεδομένων σε FPGAs. Έτσι, συνδυάζοντας αλγοριθμικές και αρχιτεκτονικές τεχνικές, το συγκεκριμένο μοντέλο transformer μπορεί να παρέχει βελτίωση της ταχύτητας και της αποδοτικότητας στις εργασίες επεξεργασίας φυσικής γλώσσας σε συστήματα FPGA. Ειδικότερα, χρησιμοποιώντας προσεγγίσεις αποκοπής

βαρών, οι ερευνητές στοχεύουν να ελαχιστοποιήσουν τον αριθμό των παραμέτρων βάρους. Δημιουργούν μια στρατηγική που επικεντρώνεται στην επίτευξη της ισορροπίας στηλών διαγράφοντας ίσο αριθμό στοιχείων μπλοκ στήλης από κάθε στήλη του πίνακα βάρους. Έτσι, το μέθοδος του μοντέλου μειώνεται δραματικά διατηρώντας την ακρίβεια πρόβλεψη. Με βάση διάφορα πειράματα που έλαβαν χώρα, ο σχεδιασμός του υλικού σε FPGA είναι αποτελεσματικός με επιτάχυνση 11 φορές καλύτερη σε σύγκριση με CPU και 2 φορές καλύτερος σε επιτάχυνση σε σύγκριση με GPU χρησιμοποιώντας ως βάση το RTX 2080Ti και με τεχνολογία Alveo U200(16nm).

3.6 VIA

Το 2022, παρουσίασε ο Teng Wang το ViA, [4] το οποίο παρουσιάζεται ως μια καινοτόμα αρχιτεκτονική επιταχυντή βασισμένη σε FPGA, που περιλαμβάνει μια μονάδα αναγνώρισης μνήμης, μονάδα εγγραφής μνήμης και δύο στοιχεία επεξεργασίας : το πρότυπο αυτό-προσοχής NSA και το MLP. Αναλύει πρώτα τη δομή των δεδομένων και τις υπολογιστικές απαιτήσεις του ViT και προτείνει στρατηγικές όπως η διαίρεση των δεδομένων για τη βελτίωση της αποδοτικότητας και τη μείωση της εξάρτησης από τη διαδρομή. Στη συνέχεια, περιγράφει την υλοποίηση του ViA σε FPGA και παρουσιάζει αποτελέσματα σύγκρισης με CPU, GPU καθώς και με προηγούμενους επιταχυντές FPGA. Ο ViA επιτυγχάνει σημαντική βελτίωση στην ενεργειακή απόδοση και την απόδοση σε σύγκριση με άλλες αρχιτεκτονικές, ενισχύοντας έτσι την αποτελεσματικότητα των επιταχυντών ViT σε FPGA. Έτσι, σε σύγκριση με άλλες CPU και GPU βελτιώνει την ταχύτητα 60 φορές και την ενεργειακή απόδοση 5 φορές. Στην μελέτη γίνεται σύγκριση της Nvidia Tesla V100 και της τεχνολογίας Alveo U50(16nm). Τέλος, επιτυγχάνει να έχει ρυθμό επιτάχυνσης 309.6 GOPs και ρυθμό ενεργειακής απόδοσης 7.9 GOPs/W

3.7 G. Tzanos

Επίσης, το 2022, ο Γ. Τζάνος [18] και οι υπόλοιποι ερευνητές παρουσίασαν μια μοναδική αρχιτεκτονική σχεδιασμένη για δίκτυα Transformer σε FPGAs που επιτυγχάνει σημαντικές επιταχύνσεις σε σχέση με τις υλοποιήσεις σε CPU. Τα δίκτυα αυτά είναι κρίσιμα στα υπάρχοντα μοντέλα NLP, όπως BERT, GPT3 καθώς αντιμετωπίζουν με επιτυχία εργασίες μετατροπής ακολουθιών. Η έρευνα αυτή υπογραμμίζει την έλλειψη τοπολογιών δικτύων Transformer που επιταχύνονται σε FPGA, ενώ ταυτόχρονα αναδεικνύει τη δυνατότητα των FPGA να βελτιώσουν την απόδοση ταχύτητας και την ενεργειακή αποδοτικότητα. Όσο αφορά την αρχιτεκτονική, η έρευνα αυτή επικεντρώνεται στην βελτιστοποίηση υπολογιστικών δραστηριοτήτων, όπως οι λειτουργίες GEMM, Gelu, Softmax, χρησιμοποιώντας μια υβριδική προσέγγιση υλοποίησης. Επίσης, οι βελτιστοποιήσεις αυτές γίνονται με πολλαπλασιασμό πινάκων και πετυχαίνει επιτάχυνση 2.3 φορές

σε σχέση με άλλες CPU. Επίσης, η σύγκριση γίνεται με χρήση της CPU Xeon (40 threads) και της τεχνολογίας Alveo U200(16nm) για FPGA.

3.8 DFX

Ο Seongmin Hong, το 2022, δημιούργησε έναν επιταχυντή FPGA με την αρχιτεκτονική DFX [8]για την βελτιστοποίηση των εργασιών παραγωγής κειμένου χρησιμοποιώντας το GPT-2. Η συγκεκριμένη αρχιτεκτονική αντιμετωπίζει την ανεπαρκή παράλληλη επεξεργασία κατά το στάδιο της παραγωγής της GPU. Στην συγκεκριμένη εργασία χρησιμοποιείται η μέθοδος της ποσοτοποίησης. Η μέθοδος αυτή είναι μια τεχνική με την οποία γίνεται επεξεργασία και βελτιστοποίηση όσο αφορά την απόδοση. Έτσι, αποτελεί μια τεχνική που μειώνει τον αριθμό των Bits που απαιτούνται για την αναπαράσταση των μοντέλων με μειωμένη ακρίβεια. Επίσης, βελτιώνει σημαντικά την απόδοση στην ταχύτητα και την ενεργειακή αποδοτικότητα συγκριτικά με άλλους επιταχυντές GPU, επιδεικνύοντας την αποτελεσματικότητα στη βελτίωση της απόδοσης που αφορούν την φυσική γλώσσα. Η μελέτη αυτή αναφέρεται, ακόμη, στην σπουδαιότητα της εξειδικευμένης αρχιτεκτονικής DFX για την επιτάχυνση της διαδικασίας end-to-end σε μοντέλα με βάση μετασχηματιστές, προσφέροντας τους έτσι μια πιο οικονομική και αποδοτική λύση. Σε σύγκριση με άλλους επιταχυντές το DFX είναι 3.8 φορές πιο γρήγορος και 4 φορές πιο ενεργειακά αποδοτικός. Τέλος, έχει ως επεξεργαστή σύγκρισης την Nvidia V100 και χρησιμοποιεί ως τεχνολογία Alveo U280(16nm).

3.9 STA

Ο Chao Fang, το 2022, παρουσίασε έναν αποδοτικό επιταχυντή (Sparse Transformer Accelerator – STA) [9] σε FPGA για να ανταποκριθεί στις υψηλές υπολογιστικές απαιτήσεις των μοντέλων που βασίζονται σε μετασχηματιστές. Οι συγγραφείς χρησιμοποιούν την δομή N:M σε μετασχηματιστές για να ελαχιστοποιήσουν τις λειτουργίες και το μέγεθος της μνήμης ενώ παράλληλα θέλουν να αυξήσουν την υπολογιστική απόδοση. Ο σχεδιασμός του STA αποτελείται από ένα ενιαίο μηχανισμό υπολογισμού για πολλαπλασιασμό πινάκων, μια ενότητα Softmax και ένα μηχανισμό πολλαπλασιασμό πινάκων (DMME). Το DMME παρέχει ένα μεγάλο φάσμα πολλαπλασιασμό πινάκων, ενώ το Softmax βοηθάει στην ελαχιστοποίηση της καθυστέρησης από τη μεταφορά δεδομένων μεταξύ των ενδιάμεσων δικτύων. Η μελέτη συγκρίνει την ακρίβεια του μοντέλου με διάφορους συνδυασμούς N:M και επιλέγει το 1:4 και το 1:8 για την μελλοντική δοκιμή της απόδοσης. Η χρήση του STA γίνεται σε μια συσκευή Intel Arria 10 SX660 και οδηγεί σε σημαντικά κέρδη στην ενεργειακή απόδοση σε σύγκριση με προηγούμενες μεθόδους FPGA. Παρέχει, επίσης, χαμηλότερη καθυστέρηση ενώ αυξάνει την ενεργειακή οικονομία.

3.9.1 STA-4, STA-8

Το μοντέλο STA χωρίζεται σε δύο υποκατηγορίες, STA-4 και STA-8 Για το STA-4 σε σύγκριση με άλλα μοντέλα επιτυγχάνει 6.7 φορές καλύτερη απόδοση και είναι καλύτερο ενεργειακά 10 φορές, έχοντας ως βάση σύγκριση την Nvidia RTX 2080Ti. Επίσης, έχει ρυθμό επιτάχυνσης 392.8 GOPs και ρυθμό ενεργειακής απόδοσης 33.6 GOPs/w. Σε αντίθεση, το STA-8 έχει μικρότερη απόδοση, δηλαδή σε σύγκριση με άλλους επιταχυντές έχει 4.4, παρόλο που είναι καλύτερο ενεργειακά, δηλαδή 12.3 φορές σε σύγκριση με άλλα μοντέλα και πετυχαίνει ρυθμό επιτάχυνσης 523.8 GOPs και ρυθμό ενεργειακής απόδοσης 41.2 GOPs/W.

3.10 HPTA

Το 2023 παρουσιάστηκε το HPTA [10] από τους Yuntao Han και Qiang Liu, οι οποίοι παρουσίασαν έναν επιταχυντή υψηλής απόδοσης, αξιοποιώντας έναν προσαρμοσμένο πίνακα πολλαπλασιασμού, δέντρο πρόσθετων και υποσύστημα μνήμης. Μπορεί να χειριστεί διάφορα είδη μετασχηματιστών που χρησιμοποιούνται στην Επεξεργασία Φυσικής Γλώσσας (NLP) και στην Όραση Υπολογιστή (CV).Γίνεται αξιολόγηση στην απόδοση του HPTA έναντι υλοποιήσεων CPU, GPU και άλλων FPGA. Τα αποτελέσματα έδειξαν ότι τόσο τα μοντέλα BERT όσο και Swin Transformer βελτιώθηκαν σημαντικά όσον αφορά την διάμετρο (ταχύτητα) και την ενεργειακή απόδοση. Σε σύγκριση με CPU και GPU, το HPTA επεξεργάστηκε το BERT έως και 44 φορές πιο γρήγορα και 175 φορές πιο ενεργειακά αποδοτικά. Ήταν επίσης 1,8 φορές ταχύτερο από τους προηγούμενους επιταχυντές FPGA.

3.11 FlexRun

Ακόμη, ο SUYEONC HUR παρουσίασε το 2023 το FlexRun[11], το οποίο προσφέρει μια πιθανή επιλογή για την επιτάχυνση σε μοντέλα NLP μέσω της αξιοποίησης των FPGA, τη βελτιστοποίηση του σχεδιασμού της αρχιτεκτονικής και την αυτοματοποίηση της διαδικασίας για να επιτευχθεί η υψηλή απόδοση σε διάφορες δραστηριότητες. Το μοντέλο αυτό αντιμετωπίζει ζητήματα που μπορεί να προκληθούν, όπως την ευρεία ποικιλία διαστάσεων, τις λειτουργίες του πίνακα και τις ομοιόμορφες λειτουργίες διανυσμάτων, με έναν ευέλικτο αρχιτεκτονικό πρότυπο σε FPGA και αλγόριθμους εξερεύνησης του χώρου σχεδίασης. Αυτό έχει ως στόχο την βελτιστοποίηση του σχεδιασμού μονάδων υπολογισμού και την αναδιαμόρφωση του επιταχυντή. Γίνονται, λοιπόν, συγκρίσεις με την αρχιτεκτονική Brainwave της Intel σε FPGA και GPU και δείχνουν σημαντικές επιταχύνσεις για διάφορα μοντέλα BERT και GPT2. Το συγκεκριμένο μοντέλο βελτιώνει την απόδοση του κατά 1.59 φορές από τις ελάχιστες τιμές FPGA ενώ υπερτερεί για τις υλοποιήσεις σε GPU κατά 2.79 φορές για το BERT και 2.59 φορές για το GPT2. Το FlexRun ξεπερνάει το GPU V100 της Nvidia κατά 2.69 φορές σε θέμα ταχύτητας με χρήση της τεχνολογίας Straxt 10GX(14nm).

3.12 Zhongyo Zhao

Το 2023, Zhongyo Zhao [13], παρουσίασε έναν επιταχυντή που χρησιμοποιεί έναν τρόπο χειρισμού δεδομένων με εξόδους σε σταθερό μπλοκ (OBS) για να εκτελεί αποτελεσματικά μοντέλα μετασχηματιστή για την αναγνώριση αντικειμένων. Ο τρόπος που προτείνεται είναι να διαιρεθούν οι είσοδοι και να γίνουν κατανομές των βαρών σε μικρούς πίνακες μπλοκ για να μειωθεί η προσπέραση της μνήμης στα δεδομένα εισόδου και βαρών. Επίσης, η ροή των δεδομένων του OBS διατηρεί το ποσοστό χρήσης με τη συλλογή μερικών αθροισμάτων, ενώ μειώνει κάπως σε σύγκριση με τη ροή δεδομένων εξόδου σε σταθερό μπλοκ. Αυτό έχει ως αποτέλεσμα να οδηγήσει σε βελτιωμένη συνολική ενεργειακή απόδοση. Ο συγκεκριμένος επιταχυντής υλοποιεί τη ροή δεδομένων και έχει ρυθμό επεξεργασίας 728.3 GOPs και ενεργειακή απόδοση 58.31 GOPs/W, υπερβαίνοντας προηγούμενους επιταχυντές βασισμένους σε CNN. Παρατηρείται ότι η κατανάλωση της ενέργειας μειώνεται κατά 33% σε σύγκριση με έναν κανονικό επιταχυντή. Η μελέτη αυτή χρησιμοποίησε επεξεργαστή Xilinx VC709 για να γίνει η σύγκριση και τη τεχνολογία Virtex™ 7VC707 (28nm).

3.13 SWIN

Το 2023, οι Zhiyang Liu, Zhenhua Ren και Pengyu Yin2 παρουσίασαν έναν επιταχυντή ειδικά για το μοντέλο Swin Transformer σε εργασίες όρασης υπολογιστών[12]. Η αρχιτεκτονική αυτή χρησιμοποιεί μια ιεραρχική δομή, και αντιμετωπίζει προβλήματα που αντιμετωπίζουν οι τυπικοί Transformers κατά την επεξεργασία των μεγάλων εικόνων, προβλήματα όπως η επιτάχυνση υλικού λόγω της μη γραμμικής τους φύσης. Το πλαίσιο αυτό επικεντρώνεται στην βελτίωση των υπολογισμών, όπως η κανονικοποίηση επιπέδου (Layer Norrmalization-Ln) , Softmax και το GELU τα οποία αποτελούν τα πιο δημοφιλή μοντέλα transformer. Γι’ αυτό τον λόγο, η μέθοδος που προτείνετε είναι ο πολλαπλασιασμός πινάκων καθώς είναι σχεδιασμένη για την υπολογιστική απόδοση συνελικτικών λειτουργιών. Ο συγκεκριμένος επιταχυντής προσφέρει σημαντική επιτάχυνση και οφέλη στην ενεργειακή απόδοση σε σχέση με τις βάσεις CPU και GPU.

Συγκεκριμένα, χωρίζεται σε τρεις διαφορετικές κατηγορίες, Swin-T, Swin-S, Swin-B. Για την πρώτη κατηγορία, Swin-T είναι 1.8 φορές πιο γρήγορος και 20.5 φορές πιο ενεργειακά αποδοτικός. Για την δεύτερη κατηγορία, Swin-S, είναι 1.7 φορές πιο γρήγορος και 18.6 φορές πιο ενεργειακά αποδοτικός, ενώ για την τρίτη κατηγορία, είναι 4.4 φορές πιο γρήγορος και 14.6 φορές πιο ενεργειακά αποδοτικός. Και τα τρία μοντέλα έχουν ως βάση σύγκρισης τον επεξεργαστή GeForce RTX 2080Ti της Nvidia και χρησιμοποιούν ως τεχνολογία την XCZU19EG (16nm). Τέλος, σημαντικό είναι να αναφερθεί ότι και τα τρία μοντέλα αποδίδουν ρυθμό επιτάχυνσης. Πιο συγκεκριμένα, οι ρυθμοί επιτάχυνσης και των τριών μοντέλων είναι παραπλήσιοι, 431.2, 403.5, 436.4 GOPs για Swin-T, Swin-B και Swin-S αντίστοιχα.

3.14 SSR

Το 2024, ο Jinming Zhuang παρουσίασε το SSR[2] ως μια μοναδική αρχιτεκτονική με έμφαση στην ισορροπία μεταξύ της καθυστέρησης και της απόδοσης στην επιτάχυνση του μετασχηματιστή. Χρησιμοποιεί διάφορα στοιχεία όπως FGPA και εξετάζει την δυνατότητα ανταλλαγής μεταξύ της καθυστέρησης και της απόδοσης για διάφορα μοντέλα, κάνοντας αυξήσεις στην απόδοση και στην ενεργειακή αποδοτικότητα. Η μέθοδος που χρησιμοποιείται είναι ο πολλαπλασιασμός πινάκων, καθώς ελέγχει τα δεδομένα επικοινωνίας μεταξύ των επιταχυντών και αναζητά τρόπους με τους οποίους μπορεί να βελτιωθεί η απόδοση. Το SSR, παρέχει εργαλεία ανοικτού κώδικα για την αναπαραγωγή αποτελεσμάτων καθώς και έχει την δυνατότητα να βελτιστοποιήσει την επικοινωνία μεταξύ των επιταχυντών , μειώνοντας το κόστος μετάδοσης δεδομένων. Σε σύγκριση με άλλες CPU και GPU, το SSR είναι περίπου 36 φορές ταχύτερο και 21 φορές πιο ενεργειακά αποδοτικό από τους προηγούμενους επιταχυντές. Στη συγκεκριμένη μελέτη γίνεται χρήση της Nvidia A10G GPU και της τεχνολογίας VCK190(7nm).

3.15 BETA

Το 2024, ο Yuhao Ji παρουσίασε έναν επιταχυντή δυαδικών μετασχηματιστών (BETA) [14] με αποτέλεσμα να έχει υψηλή απόδοση και ευελιξία. Αυτό επιτυγχάνεται με την μέθοδο της αφαίρεσης ροής υπολογισμού που έχει στόχο την βελτιστοποίηση της QMMs. Η μηχανή QMMs είναι μια προγραμματισμένη μηχανή που μπορεί να υποστηρίξει ένα μεγάλο φάσμα ακρίβειας , παρέχοντας παράλληλα υψηλό παραλληλισμό, ταχύτητα και ενεργειακή αποδοτικότητα. Μέσω διάφορων πειραμάτων γίνεται σύγκριση με προηγούμενους επιταχυντές σε FPGA και καταλήγουν στο συμπέρασμα ότι η ενεργειακή απόδοση βελτιώνεται συνεχώς. Σε σύγκριση με άλλες CPU και GPU δε γίνεται αφορά για την απόδοση της ταχύτητας αλλά για την ενεργειακή απόδοση καθώς είναι 22 φορές καλύτερη σε σύγκριση με άλλες. Στην μελέτη γίνεται χρήση της RTX3090 και της τεχνολογίας ZCU102(16nm), καθώς και το μοντέλο αυτό αποδίδει ρυθμό επιτάχυνσης και ρυθμό ενεργειακής απόδοσης 1436 GOPs και 174 GOPs/W αντίστοιχα.

3.16 Me-ViT

Το 2024 [15], οι Kyle Marino, Pengmiao Zhang και Viktor K. Prasanna παρουσίασαν το Me-ViT, ένα αποδοτικό στη μνήμη σχέδιο Vision Transformer που υπερτερεί των παραδοσιακών επιταχυντών ViT σε FPGA σε ταχύτητα και ενεργειακή αποδοτικότητα. Το Me-ViT συνδυάζει Self-Attention και μπλοκ Multi-Layer Perceptron, μειώνοντας τις μεταφορές δεδομένων και τις ενδιάμεσες εγγραφές φορτώνοντας τα βάρη μόνο μία φορά. Το Memory-Efficient Processing Element (ME-PE) ελαχιστοποιεί τη μετακίνηση δεδομένων και τις διακοπές υπολογισμών. Χρησιμοποιώντας συστολικούς πίνακες για πολλαπλασιασμό πινάκων, το Me-ViT βελτιστοποιεί την

πρόσβαση στη μνήμη, παρέχοντας κλιμακούμενες, υψηλής απόδοσης λύσεις για εργασίες όρασης σε FPGA. Σε σύγκριση με CPUs και GPUs, το Me-ViT είναι 5,1 φορές ταχύτερο και 4 φορές πιο ενεργειακά αποδοτικό, επιτυγχάνοντας ρυθμό επιτάχυνσης 2,682 GOPs. Η μελέτη χρησιμοποιεί την Nvidia TITAN RTX GPU και την τεχνολογία Alveo U200 (16nm) για σύγκριση..

3.17 TransAxx

Το 2024, οι Δημήτριος Δανόπουλος, Γεώργιος Ζερβάκης και Δημήτριος Σούντρης παρουσίασαν το TransAxx[16], ένα πλαίσιο που στοχεύει στη βελτίωση της αποδοτικότητας των μοντέλων Vision Transformer (ViT) μέσω προσεγγιστικού υπολογισμού. Περιλαμβάνει ένα σύστημα βασισμένο σε PyTorch που υποστηρίζει προσεγγιστικό υπολογισμό και αξιολογεί τις επιδράσεις του στα μοντέλα ViT. Η τεχνική περιλαμβάνει τη μελέτη της ευαισθησίας των transformers σε προσεγγιστικούς πολλαπλασιαστές, τη βελτίωση της ακρίβειας και τη χρήση του αλγορίθμου Monte Carlo Tree Search (MCTS) για τη δημιουργία προσεγγιστικών επιταχυντών. Βασικές τεχνικές για τη βελτίωση της ακρίβειας περιλαμβάνουν την ποσοτικοποίηση, την εκπαίδευση προ-βαθμονόμησης και την προσαρμοστική επανεκπαίδευση. Το πλαίσιο μειώνει την υπολογιστική πολυπλοκότητα και τις απαιτήσεις μνήμης, ενώ εξισορροπεί την ταχύτητα και την ενεργειακή αποδοτικότητα. Το TransAxx παρέχει μια ολοκληρωμένη προσέγγιση για τη βελτιστοποίηση των μοντέλων ViT, επιτρέποντας στους ερευνητές να βελτιώσουν την απόδοση με περιορισμένους πόρους μέσω μεθόδων όπως η ποσοτικοποίηση, η βαθμονόμηση και η επανεκπαίδευση.

3.18 Ikumi Okubo

Ο Ikumi Okubo το 2024 παρουσιάζει μια οικονομική αποδοτική υλοποίηση FPGA ενός Tiny Transformer μοντέλου βασισμένου σε τεχνική Neural ODE [17]. Η τεχνική αυτή χρησιμοποιεί λιγότερες παραμέτρους και απαιτεί λιγότερη μνήμη από τα μοντέλα ResNet, καθιστώντας την κατάλληλη για συσκευές με περιορισμένους πόρους. Το μοντέλο αποτελείται από ODEBlocks που επαναχρησιμοποιούν τις ίδιες παραμέτρους, μειώνοντας έτσι το πλήθος τους, και περιλαμβάνει μια μαθησιακή σχετική κωδικοποίηση θέσης για τη διατήρηση των πληροφοριών θέσης.

Η ποσοτικοποίηση γίνεται με μετατροπή δεδομένων πλήρους ακρίβειας σε n-bit ακέραιους χρησιμοποιώντας LLTs. Η αρχιτεκτονική συνδυάζει το ODENet με DSC και MHSA για μια υβριδική προσέγγιση, καθιστώντας το μοντέλο μνημονικά αποδοτικό. Η μελέτη δείχνει ότι η προτεινόμενη υλοποίηση FPGA του μοντέλου είναι 12.8 φορές πιο γρήγορη και 9.2 φορές πιο ενεργειακά αποδοτική σε σύγκριση με ARM Cortex-A53 CPU, χρησιμοποιώντας την τεχνολογία ZCU102 (16nm).

ΚΕΦΑΛΑΙΟ 4ο : Βιβλιογραφική μελέτη των ASIC

Στο κεφάλαιο αυτό γίνεται διαχρονική βιβλιογραφική μελέτη των επιταχυντών στη κατηγορία των ASIC.

4.1 A^3

Μία από τις πρώτες έρευνες για την επιτάχυνση των δικτύων Transformer προτάθηκε το 2020 από τον Hma και ονομάζεται A^3[26] Το άρθρο προτείνει έναν επιταχυντή υλικού για μηχανισμούς προσοχής σε νευρωνικά δίκτυα, που όχι μόνο επικεντρώθηκε στην αποδοτική υλοποίηση του μηχανισμού προσοχής σε υλικό, αλλά και στη μείωση του υπολογιστικού φόρτου μέσω αλγοριθμικής βελτιστοποίησης και προσέγγισης. Παρουσιάζει έναν μηχανισμό επιλογής για να μειώσει τον αριθμό των στόχων αναζήτησης και, κατά συνέπεια, τον υπολογιστικό φόρτο.

Το προτεινόμενο σχήμα δεν έχει υλοποιηθεί σε FPGA, αλλά έχει υλοποιηθεί σε ένα σχέδιο ακριβείας κύκλου με Verilog που στοχεύει σε ένα TSMC 40nm ASIC χρονισμένο στα 1GHz. Βάσει της αξιολόγησης απόδοσης, το προτεινόμενο σχήμα μπορεί να επιτύχει έως και 7 φορές ταχύτερη απόδοση σε σύγκριση με μια υλοποίηση σε Intel Gold 6128 CPU και έως και 11 φορές καλύτερη ενεργειακή απόδοση σε σύγκριση με μια υλοποίηση σε CPU.

4.2 Sanger

Το 2021,ο Liqiang Lu[27] και οι υπόλοιποι συνεργάτες του παρουσίασαν ένα πλαίσιο συν-σχεδιασμού που περιγράφει μια καινοτόμα προσέγγιση για την αντιμετώπιση του υπολογιστικού φόρτου που συνδέεται με τους μηχανισμούς προσοχής στα νευρωνικά δίκτυα, ειδικά σε εργασίες επεξεργασίας φυσικής γλώσσας και όρασης υπολογιστώ. Για να αντιμετωπιστεί αυτό, προτείνεται ένα πλαίσιο που αξιοποιεί την αραιότητα στις διεργασίες προσοχής μέσω συνεργασίας υλικού και λογισμικού.

Το Sanger εξειδικεύεται στην αραιή προσοχή, η οποία χρησιμοποιεί πολλαπλασιασμό πινάκων για να μειώσει τις απαιτήσεις υπολογισμού μνήμης. Επίσης, το μοντέλο αυτό, υπερτερεί σε απόδοση των επεξεργαστών V100 GPU, AMD Ryzen Threadripper 3970X CPU και άλλων επιταχυντών προσοχής 22.7 φορές χρησιμοποιώντας την τεχνολογία UMC(55nm) στα 500MHz. Ακόμη, μέσα από εφαρμογές το Sanger έχει ρυθμό επιτάχυνσης 529 GOPs και δεν καθιστάτε αναγκαίο να έχει ρυθμό ενεργειακής απόδοσης. Έτσι, το Sanger, μπορεί να περικόψει τα μοντέλα σε υψηλά επίπεδα αραιότητας διατηρώντας την ακρίβεια και επιδεικνύοντας την αποτελεσματικότητα του στην επεξεργασία των διεργασιών προσοχής.

4.3 SpAtten

Το 2021, ο Want και οι συνεργάτες του παρουσίασαν ένα πλαίσιο για την επιτάχυνση των μεγάλων γλωσσικών μοντέλων με την ονομασία Spatten[28]. Το SpAtten προτείνει ένα νέο σχήμα για την επιτάχυνση της Επεξεργασίας Φυσικής Γλώσσας (NLP) χρησιμοποιώντας τρεις αλγοριθμικές βελτιστοποιήσεις: σταδιακή περικοπή tokens, σταδιακή περικοπή κεφαλών και προοδευτική ποσοτικοποίηση για να μειώσει τον υπολογισμό και την πρόσβαση στη μνήμη.

Το προτεινόμενο σχήμα έχει υλοποιηθεί σε ένα σχέδιο ακριβείας κύκλου χρησιμοποιώντας SpinalHDL και έχει χαρτογραφηθεί σε ASIC χρησιμοποιώντας βιβλιοθήκη TSMC 40nm. Το SpAtten επιτυγχάνει επιτάχυνση 162x και 347x σε σχέση με μια GPU (TITAN Xp) και έναν επεξεργαστή Xeon CPU, αντίστοιχα. Σε όρους ενεργειακής απόδοσης, το SpAtten επιτυγχάνει εξοικονόμηση ενέργειας 1193x και 4059x σε σύγκριση με την GPU και την CPU, αντίστοιχα.

Μέσα από σειρές δοκιμών και αναλύσεων, το άρθρο δείχνει την αποτελεσματικότητα του SpAtten στη βελτίωση των διαδικασιών προσοχής, με αποτέλεσμα σημαντικές αυξήσεις στην απόδοση και στην ενεργειακή αποδοτικότητα. Έτσι, παρουσιάζεται ρυθμός επιτάχυνσης 360 GOPs και ρυθμός ενεργειακής απόδοσης 382 GOPs/W.

4.4 ELSA

Το 2021, ο Ham και οι συνεργάτες του παρουσίασαν μια προσέγγιση συν-σχεδιασμού υλικού-λογισμικού για την επιτάχυνση των δικτύων Transformer με την ονομασία ELSA[29]. Το ELSA μειώνει σημαντικά την υπολογιστική σπατάλη σε μια λειτουργία αυτό-προσοχής. Σε αντίθεση με το συμβατικό υλικό, όπως οι CPU ή οι GPU, που δεν μπορούν να επωφεληθούν από την προσέγγιση, το ELSA προτείνει εξειδικευμένο υλικό που μεταφράζει άμεσα αυτή τη μείωση για να βελτιώσει περαιτέρω την απόδοση και την ενεργειακή απόδοση.

Αξιολογούν διάφορα αντιπροσωπευτικά μοντέλα νευρωνικών δικτύων που εστιάζουν στην αυτό-προσοχή για να αποδείξουν την αποτελεσματικότητα του ELSA. Για την αξιολόγηση της απόδοσης, υλοποίησαν έναν προσαρμοσμένο εξομοιωτή για το ELSA που στοχεύει σε ένα 40nm ASIC χρονισμένο στα 1GHz. Το ELSA επιτυγχάνει έως και 157 φορές ταχύτερη απόδοση σε σύγκριση με τις GPU και βελτιώσεις δύο τάξεων μεγέθους στην ενεργειακή απόδοση σε σχέση με την GPU για τον υπολογισμό της αυτό-προσοχής.

4.5 SALO

Το 2022, ο Guan Shen [30] περιγράφει ένα χωρικό επιταχυντή που σχεδιάστηκε για να βελτιώσει την απόδοση του μετασχηματιστή επιτρέποντας υβριδικούς μηχανισμούς προσοχής για

μεγάλες ακολουθίες. Η μελέτη αντιμετωπίζει τα προβλήματα υπολογισμού και μνήμης που προκύπτουν από τις εκτεταμένες ακολουθίες εισόδου στους μετασχηματιστές. Το μοντέλο αυτό προσπαθεί να βελτιώσει την αποτελεσματικότητα του υπολογισμού της προσοχής χρησιμοποιώντας υβριδικά αραιά πρότυπα προσοχής και ένα χωρικό επιταχυντή. Η στρατηγική που χρησιμοποιεί είναι η ανάλυση δημοφιλών αραιών μεθόδων προσοχής για την εύρεση κοινών προτύπων υπολογισμού και την αξιολόγηση τους για επαναχρησιμοποίησης δεδομένων κατά τον υπολογισμό. Χρησιμοποιεί υβριδικούς μηχανισμούς προσοχής και έναν χωρικό επιταχυντή. Επιτυγχάνει σημαντικές επιταχύνσεις σε σύγκριση με GPU και CPU , όπως 25.5 φορές πιο γρήγορος και 336.1 φορές πιο ενεργειακά αποδοτικός, χρησιμοποιώντας ως βάση σύγκρισης Nvidia GTX 1080Ti με τεχνολογία FreePDK(45nm). Έτσι, παρουσιάζει καλύτερη επεξεργασία προσοχής μέσω αποτελεσματικής υλοποίησης υλικού και δεδομένων. Τέλος, εστιάζει στην ανάλυση αραιών μεθόδων προσοχής και την εφαρμογή στρατηγικών βελτίωσης υπολογισμού προσοχής.

4.6 AccelTran

Ο Shikhar Tuli το 2020 παρουσίασε μια νέα αρχιτεκτονική επιταχυντή , που ονομάζεται AccelTran[31], με στόχο την βελτίωση της αποδοτικότητας των μοντέλων μετασχηματιστών, ιδιαίτερα στον τομέα της επεξεργασίας φυσικής γλώσσας. Είναι ένα σύστημα πρόβλεψης με ελάχιστο κόστος. Η στρατηγική που χρησιμοποιείται είναι η συμπίεση πινάκων και γίνεται υιοθέτηση διάφορων ροών δεδομένων για την βελτίωση της επαναχρησιμοποίησης δεδομένων, βελτιώνοντας έτσι την ενεργειακή αποδοτικότητα.

Το AccelTran-Edge και το AccelTran-Server είναι δύο παραλλαγές της προτεινόμενης αρχιτεκτονικής που απευθύνονται σε διάφορες πλατφόρμες. Το πρώτο απευθύνεται σε φορητές συσκευές με περιορισμένο ενεργειακό προϋπολογισμό, επικεντρώνοντας στην εξοικονόμηση ενέργειας μέσω της χαμηλής κατανάλωσης ισχύος. Από την άλλη πλευρά, το δεύτερο, σχεδιάζεται ειδικά για εφαρμογές σε Cloud με έμφαση στην υψηλή ροή. Το AccelTran-Edge παρέχει πολύ μεγαλύτερη ροή με χαμηλή κατανάλωση ενέργειας σε σύγκριση με μια συσκευή Raspberry Pi, ενώ το AccelTran-server παρέχει 5.7 φορές μεγαλύτερη ροή και 3.7 φορές χαμηλότερη κατανάλωση ενέργειας σε σύγκριση με το μοντέλο Energon, χρησιμοποιώντας την τεχνολογία FinFet(14nm). Τέλος, γίνεται αναφορά στην συγκεκριμένη μελέτη για τον ρυθμό επιτάχυνσης που αποδίδουν. Για το AccelTran – server είναι 372.000 GOPs και για το AccelTran - edge είναι 7.520 GOPs, οι οποίοι είναι αρκετά μεγάλοι ρυθμοί σε σύγκριση με όλα τα προαναφερόμενα μοντέλα, καθώς η συγκεκριμένη μελέτη κάνει τη σύγκριση της με Nvidia A100 και Raspberry Pi αντίστοιχα.

4.7 DTQAtten

Το 2022, ο Tao Yang παρουσίασε το DTQAtten [32], μια τεχνική για την αντιμετώπιση των προκλήσεων στα μοντέλα επεξεργασίας φυσικής γλώσσας (NLP). Το DTQAtten συνδυάζει δυναμική κβαντοποίηση και εξειδικευμένη αρχιτεκτονική υλικού για να βελτιώσει την αποτελεσματικότητα και την απόδοση των μοντέλων NLP που βασίζονται στην προσοχή. Χρησιμοποιεί δυναμική κβαντοποίηση των δεδομένων ανάλογα με την ανοχή τους στο θόρυβο και συγχωνεύει την περικοπή και την κβαντοποίηση σε ένα συνεκτικό πλαίσιο βελτιστοποίησης.

Αυτή η προσέγγιση μειώνει το αποτύπωμα μνήμης, την καθυστέρηση εκτίμησης και την κατανάλωση ενέργειας σε μοντέλα NLP όπως τα BERT και GPT-2. Τα πειραματικά αποτελέσματα δείχνουν ότι το DTQAtten είναι 16.4 φορές πιο γρήγορο και 3.8 φορές πιο ενεργειακά αποδοτικό σε σύγκριση με την Nvidia Titan Xp του μοντέλου SpAtten, χρησιμοποιώντας τεχνολογία TMSC (40nm). Επιτυγχάνει ρυθμό απόδοσης 952 GOPs και ρυθμό ενεργειακής απόδοσης 1298,4 GOPs/W. Συνολικά, το DTQAtten βελτιώνει σημαντικά την ταχύτητα και την ενεργειακή απόδοση, ξεπερνώντας τους προηγμένους επιταχυντές όπως το A^3 και το SpAtten.

4.8 Energon

Το 2023, οι ερευνητές Zhe Zhou, Junlin Liu, Zhenyu Gu και Guangyu Sun παρουσίασαν το Energon [33]ως μια νέα αρχιτεκτονική που βελτιώνει σημαντικά την απόδοση σε εφαρμογές edge και cloud computing. Το σύστημα δοκιμάζεται σε δύο διαμορφώσεις: το Energon-edge και το Energon-server, και οι δύο είναι σχεδιασμένες για να καλύπτουν διαφορετικές ανάγκες υπολογισμού. Για να αντιμετωπίσει αποτελεσματικά τους μηχανισμούς προσοχής, ο σχεδιασμός περιλαμβάνει μονάδες φιλτραρίσματος, IPU, μονάδες προσοχής, πίνακες MAC, μονάδες Softmax και πολλαπλασιαστές. Το Energon-edge υπερέχει του επεξεργαστή ARM A72 CPU και του GPU TX2 . Επίσης, είναι 3057 φορές πιο γρήγορο και 2951 φορές πιο ενεργειακά αποδοτικό χρησιμοποιώντας την τεχνολογία FreePDK(45nm). Το Energon-server υπερέχει και των επεξεργαστών Xeon CPUs και των GPU V100 και είναι 168 φορές πιο γρήγορο και 10000 φορές πιο ενεργειακά αποδοτικό χρησιμοποιώντας την ίδια τεχνολογία με το Energon-edge.

4.9 SALO2

Το 2024, ο Jieru Zhao, παρουσιάζει το SALO2 [35] ως ένα βελτιωμένο πλαίσιο για την αποτελεσματική υπολογιστική των διαδικασιών προσοχής τόσο σε στατικά όσο και σε δυναμικά σενάρια αραιότητας σε σύγκριση με το προηγούμενο μοντέλο SALO. Το SALO2, συνδυάζει βελτιστοποιήσεις σε επίπεδο λογισμικού με επιταχυντές υλικού και μονάδες αντιστοίχισης προτύπων, παρέχει μια ευέλικτη λύση για μια ποικιλία εφαρμογών, εργασιών και ακολουθιών

εισόδου. Χρησιμοποιούνται αλγόριθμοι αραιής προσοχής, τεχνικές διαίρεσης και αναδιάταξης δεδομένων και στρατηγικές αντιστοίχισης προτύπων για τη βελτίωση της απόδοσης και της ευελιξίας των μηχανισμών προσοχής σε νευρωνικά δίκτυα. Τα ευρήματα της μελέτης υπογραμμίζουν τη χρησιμότητα του SALO2 στην επίτευξη μιας ισορροπίας μεταξύ απόδοσης και ευελιξίας, καθιστώντας το μια κατάλληλη επιλογή για την ενίσχυση των διαδικασιών προσοχής σε διάφορες εφαρμογές νευρωνικών δικτύων. Σε σύγκριση με άλλα μοντέλα είναι περίπου 25 φορές πιο γρήγορο και 70 φορές πιο ενεργειακά αποδοτικό, χρησιμοποιώντας ως βάση σύγκρισης την Nvidia RTX4090 με την τεχνολογία FreePDK(45nm).

4.10 H3D Transformer

Το 2024, προτείνεται μια νέα σχεδίαση επιταχυντή βασισμένη σε ετερογενή 3D για τα μοντέλα επιταχυντών[34]. Χρησιμοποιείται μια στρατηγική προσέγγισης που αξιοποιεί πολλές παραδοσιακές παραδειγματικές υπολογιστικές (CIM) και ψηφιακές μονάδες επεξεργασίας των τενσόρων (TPU). Η προτεινόμενη αρχιτεκτονική αντιμετωπίζει δύο βασικά ζητήματα που εμποδίζουν τα μοντέλα transformer από το να εφαρμοστούν στην άκρη: το κόστος περιοχής τσιπ και την κατανάλωση ενέργειας, συνδυάζοντας διάφορους τύπους υπολογιστών για να εκτελέσει διαφορετικά φορτία εργασίας MatMul με υψηλή αποδοτικότητα.

Επίσης, επιτρέπει την υψηλή πυκνότητα αποθήκευσης για μεγάλες παραμέτρους μοντέλων χρησιμοποιώντας προηγμένες τεχνολογίες δεσμευτικής σύνδεσης και υβριδικού δεσμευμένου. Για αυτό τον λόγο, προτεινόμενη αρχιτεκτονική στοιβάζει πολλαπλά D-CIM τσιπάκια 22 nm FeFET ψηφιακών CIM (D-CIM) για να αποκτήσει τεράστια αποθήκη μνήμης στο τσιπ κρατώντας παράλληλα το ελάχιστο μέγεθος του τσιπ. Διάφοροι τύποι συστημάτων υπολογιστών συνδυάζονται για να επεξεργαστούν τις διάφορες εργασίες MatMul με εξαιρετική ενεργειακή απόδοση. Έτσι, επιτυγχάνει απόδοση ενέργειας 10 TOPS/W(= 10000 GOPs/W) για τα μοντέλα BERT και GPT2, περίπου 2,6x ~ 3,1x μεγαλύτερη από τη βάση με 7nm TPU και FeFET μνήμης.

ΚΕΦΑΛΑΙΟ 5: Βιβλιογραφική μελέτη των In-Memory

Όμοια, τα τελευταία χρόνια έχουν αναπτυχθεί διάφορα μοντέλα σε In-Memory. Από το 2020 έως και το 2024 έχουν εκτελεστεί πάνω από 10 διαφορετικά. Ξεκινώντας έτσι, από το 2020, και πηγαίνοντας στο 2024 θα εξετάσουμε την διαχρονική εξέλιξή τους, αναφέροντας τα πιο σημαντικά μοντέλα επιταχυντών.

5.1 ATT

Το 2020, οι ερευνητές δημιούργησαν έναν επιταχυντή ReRAM ανθεκτικού σφάλματος για νευρωνικά δίκτυα με προσοχή[36]. Η μελέτη αντιμετωπίζει τα θέματα που προκύπτουν από δυσλειτουργικές συσκευές και δομές ReRAM, προσφέροντας έναν αλγόριθμο μη-ομοιόμορφης ομαδοποίησης Xbar (NuXG) ως τεχνική αντιγραφής για τη βελτίωση της ακρίβειας διατηρώντας την ενεργειακή οικονομία. Η έρευνα προτείνει ένα σχεδιασμό παραλληλισμού και εξετάζει επίσης την ανάλυση κινδύνου, δηλαδή τους κινδύνους δεδομένων στη διαδικασία σχεδιασμού.

Επίσης, περιγράφει μια τεχνική για τη μετατροπή λέξεων σε πίνακες, χρησιμοποιώντας ένα σύστημα ReRAM βασισμένο σε διασταύρωση για τον πολλαπλασιασμό πίνακα-διανύσματος, και υλοποιεί αλγορίθμους αντιγραφής για τυχόν αποτυχίες σε ReRAM. Ο σχεδιασμός του επιταχυντή περιλαμβάνει την ανάκτηση δεδομένων από την κύρια μνήμη, τη χρήση προσωρινών μνημών στο επεξεργαστή για γρήγορη πρόσβαση στα δεδομένα, καθώς και την ενσωμάτωση άλλων μονάδων για αναλογικού πεδίου υπολογισμού. Επίσης, υπογραμμίζεται η σημασία της ακρίβειας, της ενεργειακής οικονομίας και της βελτίωσης της απόδοσης. Τέλος, σε σύγκριση με την βάση της Nvidia GTX1080Ti, το ATT έχει 202 φορές καλύτερη επιτάχυνση καθώς και είναι 11 φορές πιο ενεργειακά αποδοτικό χρησιμοποιώντας την τεχνολογία Cacti 7.0(32nm).

5.2 ReTransformer

Το 2020, οι ερευνητές επικεντρώνονται σε δύο κύρια θέματα: την Multi-Head Attention και τις αρχιτεκτονικές επεξεργασίας στη μνήμη (PIM) βασισμένες σε ReRAM[37] . Η μεθοδολογία στοχεύει στη βελτιστοποίηση των υπολογισμών, τη μείωση της κατανάλωσης ενέργειας και την αύξηση της απόδοσης σε συστήματα PIM με νέες λύσεις ReRAM. Η μελέτη αναλύει τον μηχανισμό της MHA και την ενσωμάτωση της θέσης στο Self-Attention. Παρουσιάζει το ReRAM, μια μνήμη υψηλής πυκνότητας και χαμηλής κατανάλωσης ενέργειας, που εκτελεί λειτουργίες PIM όπως ο πολλαπλασιασμός διανυσμάτων-πινάκων στη μνήμη.

Η έρευνα αντιμετωπίζει τα προβλήματα συχνής επανεγγραφής σε ReRAM, οδηγώντας σε υψηλή κατανάλωση ενέργειας, και βελτιώνει τη λειτουργία Softmax για να μειώσει την ενεργειακή

κατανάλωση. Το μοντέλο ReTransformer, σε σύγκριση με άλλους επιταχυντές και την Nvidia Titan RTX, παρουσιάζει 23.2 φορές καλύτερη επιτάχυνση και 1086 φορές καλύτερη ενεργειακή απόδοση, χρησιμοποιώντας ReRAM (32nm). Το ReTransformer επιτυγχάνει ρυθμό επιτάχυνσης 81.9 GOPs και ρυθμό ενεργειακής απόδοσης 467.7 GOPs/W.

5.3 iMCAT

Το 2022, η Laguna και οι συνεργάτες του παρουσίασαν μια νέα αρχιτεκτονική στη μνήμη για την επιτάχυνση των δικτύων Transformer για μεγάλες προτάσεις, ονομάζοντάς την iMCAT[38]. Το προτεινόμενο πλαίσιο χρησιμοποιεί έναν συνδυασμό XBars και CAMs για να επιταχύνει τα δίκτυα Transformer. Η επιτάχυνση των δικτύων Transformer επιτυγχάνεται μέσω του συνδυασμού διαφόρων τεχνικών, όπως υπολογισμοί στη μνήμη για την ελαχιστοποίηση του κόστους μεταφοράς δεδομένων, προσωρινή αποθήκευση επαναχρησιμοποιήσιμων παραμέτρων για τη μείωση του αριθμού των λειτουργιών, εκμετάλλευση της διαθέσιμης παραλληλίας στον μηχανισμό προσοχής και, τέλος, χρήση κατακερματισμού ευαίσθητου στην τοπικότητα για το φιλτράρισμα των στοιχείων της ακολουθίας ανάλογα με τη σημασία τους.Η αξιολόγηση της απόδοσης δείχνει ότι αυτή η προσέγγιση επιτυγχάνει 200 φορές ταχύτερη απόδοση και 41 φορές καλύτερη ενεργειακή βελτίωση για ακολουθίες μήκους 4098.

5.4 iMTransformer

Το 2022, η Ann Franchesca Laguna παρουσιάζει την αρχιτεκτονική iMTransformer[40] για την διευκόλυνση του σχεδιασμού υλικού-λογισμικού σε συστήματα μνήμης. Το iMTransformer επικεντρώνεται στην αξιοποίηση της Προσοχής Πολλαπλών Κεφαλών (MHA) σε δίκτυα μετασχηματιστών, επιτρέποντας τον παράλληλο χειρισμό στοιχείων ακολουθίας. Επίσης, περιγράφει τα στρώματα κωδικοποιητή και αποκωδικοποιητή των δικτύων μετασχηματιστών, δίνοντας έμφαση στο ρόλο τους στη μετατροπή εισροών μεταβλητού μήκους σε διανύσματα χαρακτηριστικών σταθερού μήκους.

Η μελέτη εξετάζει τρεις τύπους προσοχής πολλαπλών κεφαλών: καλυμμένη, διπλή κατεύθυνση και κωδικοποιητή-αποκωδικοποιητή. Η έρευνα επίσης εξετάζει τεχνικές αραιής προσοχής, όπως η Αραιά Προσοχή Βασισμένη στην Τοποθεσία και η Αραιά Προσοχή Βασισμένη σε Περιεχόμενο, για να βελτιώσει την υπολογιστική πολυπλοκότητα. Ο παραλληλισμός της προσοχής περιορίζεται από τη συνολική χωρητικότητα μνήμης και την ισχύ θερμικού σχεδιασμού. Ακόμη, η προσοχή πολλαπλών κεφαλών διπλής κατεύθυνσης μπορεί να γίνει παράλληλη για να φτάσει σε ταχύτερες ταχύτητες και η κατανάλωση ενέργειας μπορεί να μειωθεί περαιτέρω αξιοποιώντας αραιότητα βάσει τοποθεσίας και αραιότητα βάσει περιεχομένου. Σε σύγκριση με άλλα μοντέλα επιταχυντών και την

Nvidia Titan RTX GPU είναι 11 φορές πιο γρήγορο και 12.6 φορές πιο ενεργειακά αποδοτικό χρησιμοποιώντας την τεχνολογία CAMS (12nm).

5.5 TransPIM

Το 2023, οι ερευνητές, παρουσίασαν το TransPIM[39] ως ένα καινοτόμο τρόπο βελτίωσης των λειτουργιών του Transformer με τη συνδυασμένη σχεδίαση λογισμικού και υλικού. Προσθέτει αρκετές μονάδες βοηθητικού υπολογισμού (ACUs) στη μνήμης για να βελτιώσει αποδοτικά τους αλγόριθμους μείωσης διανύσματος και Softmax. Το TransPIM βελτιώνει την ταχύτητα του και την ενεργειακή του απόδοση μελετώντας χαρακτηριστικά σχεδιασμού όπως ο παραλληλισμός. Η αύξηση του παραλληλισμού του δέντρου πρόσθεσης μπορεί να μειώσει την καθυστέρηση και την κατανάλωση ενέργειας έως και 10,8 και 5,7 φορές αντίστοιχα. Ο ιδανικός παραλληλισμός του δέντρου πρόσθεσης καθορίζεται από μια συμβιβαστική λύση μεταξύ απόδοσης και επιπλέον επιφάνειας. Επιπλέον, το μοντέλο αυτό χρησιμοποιεί διαίρεση δεδομένων με βάση το διακριτικό για να αυξήσει τον παραλληλισμό στο επίπεδο της μνήμης και να μειώσει τα κόστη μεταφοράς δεδομένων ανάμεσα στα επίπεδα. Η νέα αρχιτεκτονική του TransPIM, που συνδυάζει τις παραδοσιακές μορφές υπολογισμού στη μνήμη και κοντινής μνήμης, οδηγεί σε σημαντική αύξηση της ταχύτητας, της ενεργειακής οικονομίας και της συνολικής απόδοσης για τα μοντέλα του Transformer. Για αυτό τον λόγο το μοντέλο αυτό παρουσιάζει ρυθμό επιτάχυνσης 734 GOPs

5.6 TranCIM

Το 2023, οι ερευνητές παρουσίασαν το μοντέλο TranCIM[44], το οποίο παρέχει έναν καινοτόμο σχεδιασμό για τη βελτίωση της αποδοτικότητας των μοντέλων μετασχηματιστών, που χρησιμοποιείται συνήθως στην επεξεργασία φυσικής γλώσσας, στην όραση υπολογιστή και στη βιο-πληροφορική. Η κύρια δυσκολία που αντιμετωπίζεται είναι η σημαντική μεταφορά δεδομένων και η επεξεργασία που απαιτούνται στα μοντέλα μετασχηματιστή λόγω του τεράστιου αριθμού πολλαπλασιασμών πινάκων. Για αυτό τον λόγο, οι συγγραφείς προτείνουν το TranCIM, έναν επιταχυντή μετασχηματιστή πλήρους ψηφιακού CIM με διαμορφώσιμες αλληλεπικαλυπτόμενες και παράλληλες λειτουργίες που βελτιώνουν την πρόσβαση στη μνήμη και τους υπολογισμούς για στρώματα προσοχής και πλήρως συνδεδεμένα στρώματα.

Το TranCIM χρησιμοποιεί μια λειτουργία pipeline με μια αρχιτεκτονική μακροεντολής CIM μεταφοράς bitline για τον αποτελεσματικό υπολογισμό πινάκων προσοχής. Η αρχιτεκτονική CIM πλήρους ψηφιακής μορφής ενεργοποιεί όλες τις γραμμές ταυτόχρονα, βελτιστοποιώντας τη χρήση της διάταξης ενώ διατηρεί την ακρίβεια. Σε σύγκριση με άλλα μοντέλα επιταχυντών και την Nvidia Jetson Nanoείναι 16.9 φορές πιο γρήγορο και 1.6 φορές πιο ενεργειακά αποδοτικό χρησιμοποιώντας την τεχνολογία CMOS(28nm).

5.7 H3Datten

Το 2023, οι ερευνητές, παρουσίασαν μια μοναδική αρχιτεκτονική που βελτιώνει την αποδοτικότητα υπολογισμού των μετασχηματιστών όρασης[43]. Η H3DAtten συνδυάζει αναλογικά (ACIM) και ψηφιακά (DCIM) στοιχεία υπολογισμού εντός μνήμης για την ενίσχυση της απόδοσης των οπτικών μετασχηματιστών. Η αρχιτεκτονική χρησιμοποιεί το μοντέλο μετασχηματιστή Swin, γνωστό για την ακρίβειά του σε αναγνώριση εικόνας και ανίχνευση αντικειμένων.

Η H3DAtten διαχωρίζει τα χαρακτηριστικά εισόδου σε παράθυρα για τοπική επεξεργασία προσοχής, με μεταβαλλόμενο πλάτος παραθύρου ανά στάδιο για εξαγωγή χαρακτηριστικών σε διαφορετικές κλίμακες. Το ACIM χρησιμοποιεί μετατροπείς τάσης σε ψηφιακά σήματα (ADCs) για τη μετατροπή αναλογικών δεδομένων σε ψηφιακά bits, ενώ το DCIM χρησιμοποιεί SRAM για πράξεις MatMul, μειώνοντας την ενέργεια και την καθυστέρηση εγγραφής.

Η αρχιτεκτονική H3DAtten συγκρίνεται με υπάρχοντες επιταχυντές υλικού για μοντέλα μετασχηματιστών, επιδεικνύοντας τα οφέλη της 3D ενσωματωμένης υβριδικής προσέγγισης. Ο ρυθμός επιτάχυνσης της H3DAtten είναι 1600 GOPs και ο ρυθμός ενεργειακής απόδοσης είναι 7.100 GOPs/W.

5.8 X-Former

Το τελευταίο μοντέλο για το έτος 2023, είναι το X-Former[41] και περιγράφει μια μοναδική αρχιτεκτονική, η οποία αποσκοπεί στην επιτάχυνση της διαδικασίας συμπερασμάτων Μετασχηματιστών. Η αρχιτεκτονική αυτή χρησιμοποιεί κελιά ReRAM ως μονάδες υπολογισμού εντός μνήμης για την αποθήκευση βαρών μοντέλου, ενσωματώνοντας τα βάρη των ενσωματωμένων συστημάτων σε ReRAM ως κανονική μνήμη και κελιά SRAM για ενδιάμεσες ενεργοποιήσεις και μερικά αθροίσματα. Το X-Former βελτιώνει τη μηχανή προσοχής, ώστε να κλιμακώνεται σε οποιοδήποτε αριθμό στρωμάτων αποθηκεύοντας μόνο ένα στρώμα στους πίνακες SRAM. Σε σύγκριση με τις κανονικές GPUs και τους κορυφαίους επιταχυντές εντός μνήμης, ο σχεδιασμός παρέχει πολύ ταχύτερη απόδοση και χαμηλότερη κατανάλωση ενέργειας και αυτό φαίνεται καθώς σε σύγκριση με την Nvidia GeForce GTX 1060 πετυχαίνει 85.0 ταχύτερη επιτάχυνση και 7.5 φορές καλύτερη ενεργειακή απόδοση. Επίσης, παρουσιάζει σημαντικό ρυθμό ενεργειακής κατανάλωσης, ο οποίος είναι 13.440,0 GOPs/W. Τέλος, η πειραματική τεχνική ενσωματώνει μοντελοποίηση σε επίπεδο συστήματος με έναν προσομοιωτή αρχιτεκτονικής για την αξιολόγηση διαφόρων δραστηριοτήτων στο δίκτυο μετασχηματιστών.

5.9 HARDSEA

Το 2024, οι ερευνητές παρουσίασαν το HARDSEA[45] ως μια νέα αρχιτεκτονική επιταχυντή για μοντέλα μετασχηματιστών, εστιάζοντας στον μηχανισμό αυτό-προσοχής, ο οποίος έχει τετραγωνική πολυπλοκότητα σε σχέση με τον αριθμό των εισαγωγικών συμβόλων. Ο επιταχυντής HARDSEA παρουσιάζει μια υβριδική αναλογική-ψηφιακή λύση για μοντέλα μετασχηματιστή που εστιάζει στον μηχανισμό αυτό-προσοχής. Η προσέγγιση αραιώσεως αυτό-προσοχής με ποσοτικοποιημένο προϊόν και η ψηφιακή SRAM-CIM για υπολογισμό εντός μνήμης βοηθούν στη μείωση του κόστους υπολογισμού και μνήμης αξιοποιώντας την αραιότητα στον υπολογισμό συνάφειας, Softmax και περίληψη πληροφοριών. Η τεχνική αξιολόγησης της εργασίας δείχνει ότι η προτεινόμενη προσέγγιση λειτουργεί καλά με δύο διακριτικά μοντέλα μετασχηματιστή. Τέλος, σε σύγκριση με άλλα μοντέλα πετυχαίνει 28.5 φορές καλύτερη επιτάχυνση και είναι πολύ καλύτερο ενεργειακά καθώς έχει 1,894.3 φορές καλύτερη ενεργειακή απόδοση σε σύγκριση με Nvidia RTX 3090. Το μοντέλο παρουσιάζει ρυθμό επιτάχυνσης και ρυθμό ενεργειακής απόδοσης 921.6 GOPs και 943.7 GOPs/W αντίστοιχα.

5.10 PRIMATE

Το 2024, παρουσίασαν την μελέτη Primate[46] ως ένα νέο πλαίσιο συν-σχεδιασμού υλικού-λογισμικού για την επιτάχυνση των μοντέλων μετασχηματιστών μέσω δυναμικού κλαδέματος συμβόλων. Το PRIMATE επικεντρώνεται σε τεχνολογίες PIM, οι οποίες μπορούν να υποστηρίξουν μεγαλύτερη χωρητικότητα από την SRAM ενώ παρέχουν μεγαλύτερο εύρος ζώνης, χαμηλότερη καθυστέρηση και μεγαλύτερη σταθερότητα των υπολογισμών από τα συστήματα μνήμης μη-μεταβλητής κατάστασης. Επίσης, παρουσιάζει μια στρατηγική διάρθρωσης και απεικόνισης pipeline για να μειώσει την υπό-αξιοποίηση και να μεγιστοποιήσει τον παραλληλισμό κατά την επεξεργασία μιας ροής ακολουθιών εισόδου. Για το λογισμικό του μοντέλου αυτού, προσφέρεται μια τεχνική απεικόνισης pipeline καθώς και ένα πλαίσιο βελτιστοποίησης για να μεγιστοποιήσει την απόδοση και την αποδοτικότητα. Η μελέτη δείχνει ότι το PRIMATE βελτιώνει την απόδοση κατά 30,6 φορές, την αποδοτικότητα χώρου κατά 29,5 φορές και την αποδοτικότητα ενέργειας κατά 4,3 φορές σε σύγκριση με τον υπάρχον κορυφαίο επιταχυντή PIM για μετασχηματιστές TransPIM (έχει ως βάση την Nvidia RTX 2080Ti).

ΚΕΦΑΛΑΙΟ 6: Βιβλιογραφική μελέτη των GPU

Στο κεφάλαιο αυτό γίνεται διαχρονική βιβλιογραφική μελέτη των επιταχυντών στη κατηγορία των GPU.

6.1. TurboTransformer

Το 2021, οι Jiarui Fang και Yang Yu παρουσίασαν τον επιταχυντή TurboTransformers[19], μια τεχνική για την αποδοτική εξυπηρέτηση μοντέλων Μετασχηματιστή σε GPU για εισόδους με μεταβλητό μήκος. Αντιμετώπισαν τις δυσκολίες της προσθήκης μηδενικών σε μικρότερες ακολουθίες για να ταιριάζουν με το μήκος της μεγαλύτερης ακολουθίας σε ένα πακέτο. Χρησιμοποίησαν δυναμικό προγραμματισμό για την επίλυση του ζητήματος της βελτιστοποίησης, αυξάνοντας τον ρυθμό απόκρισης κατά 35% σε σύγκριση με τη μη χρήση πακέτων.

Για τη μείωση του μεγέθους της μνήμης, το TurboTransformers εισάγει έναν διανεμητή με μεταβλητό μήκος που χρησιμοποιεί τεχνική διαχείρισης μνήμης βασισμένη σε τμήματα και μηχανισμό επαναχρησιμοποίησης χώρου στο γράφημα υπολογισμού, μειώνοντας τη χρήση μνήμης κατά 50% σε σύγκριση με έναν διανεμητή αναφοράς. Δοκιμάζοντας το σύστημα με διάφορα μοντέλα Μετασχηματιστή, συμπεριλαμβανομένων των BERT και Albert, οι συγγραφείς διαπίστωσαν ότι το TurboTransformers ξεπερνά το PyTorch και το ONNXRuntime σε καθυστέρηση και απόδοση για εισόδους με μεταβλητό μήκος, όντας 2.8 φορές πιο γρήγορο.

6.2 Jaewan Choi

Την ίδια χρονιά, το 2022, ο ερευνητής Jaewan Choi, παρουσίασε τη μελέτη με τίτλο "Επιτάχυνση των Δικτύων Transformer μέσω της Επανασύνδεσης των Επιπέδων Μαλακών Περιθωρίων" και παρέχει έναν τρόπο επιτάχυνσης του επιπέδου Softmax στα δίκτυα μετασχηματιστών[20]. Η έρευνα παρουσιάζει μια τεχνική επανασύνδεσης για την επιτάχυνση του επιπέδου Softmax στα δίκτυα μετασχηματιστών, η οποία έχει γίνει όλο και πιο σημαντική καθώς τα μοντέλα μετασχηματιστών επεξεργάζονται μακρύτερες ακολουθίες για να βελτιώσουν τα ποσοστά ακρίβειας. Η προτεινόμενη τεχνική διαιρεί το επίπεδο Softmax σε αρκετά υπό-επίπεδα, αλλάζει το πρότυπο πρόσβασης δεδομένων και στη συνέχεια συνδυάζει τα αποσυναρμολογημένα υπό-επίπεδα Softmax με τις διαδικασίες που ακολουθούν και προηγούμενες. Αυτή η μέθοδος επιταχύνει την εκτίμηση του BERT, GPT-Neo, BigBird και Longformer σε ένα τρέχον GPU κατά έως και 1.25x, 1.12x, 1.57x και 1.65x αντίστοιχα, μειώνοντας σημαντικά την κίνηση μνήμης εκτός επεξεργαστή.

6.3 LightSeq2

Το 2022, οι ερευνητές Xiaohui Wang , Yang Wei και οι υπόλοιποι συνεργάτες του, παρουσίασαν το μοντέλο "LightSeq2"[21] που επικεντρώνονται στη βελτιστοποίηση της εκπαίδευσης του μοντέλου BERT παρέχοντας μοναδικές στρατηγικές που βελτιώνουν την αποδοτικότητα διατηρώντας την ακρίβεια. Το LightSeq2 περιγράφει μια πλήρη στρατηγική για τη βελτίωση της εκπαίδευσης του μοντέλου BERT που χρησιμοποιεί καινοτόμες προσεγγίσεις όπως ο υπολογισμός μεικτής ακρίβειας, αποτελεσματικές διαδικασίες μείωσης και στρατηγικές παράλληλης επεξεργασίας. Το LightSeq2 βελτιώνει σημαντικά την απόδοση του μοντέλου Transformer. Η μελέτη βελτιστοποιεί τα υπολογιστικά γραφήματα για τους Μετασχηματιστές, με έμφαση στη συγχώνευση μη-γραμμικών πυρήνων και στην ομαδοποίηση πράξεων στοιχείου-με-στοιχείο και μείωσης για παράλληλη εκτέλεση. Αντιμετωπίζει επίσης την προσοχή Softmax στους μετασχηματιστές, εξασφαλίζοντας αριθμητική σταθερότητα και αποτελεσματικές διεργασίες. Επιπλέον, το άρθρο αξιολογεί την απόδοση των μοντέλων BERT, επικεντρώνοντας στο επίπεδο κωδικοποιητή Transformer. Τέλος, συγκρίνει το LightSeq2 με το Hugging Face BERT και το DeepSpeed BERT, επιδεικνύοντας επιταχύνσεις 1.44x για το BERT-Base και 1.28x για το BERT-Large.

6.4 LLMA

Το 2023, ο Nan Yang παρουσίασε το LLMA[22], έναν επιταχυντή για μεγάλα γλωσσικά μοντέλα (LLMs) που αξιοποιεί την αλληλεπίδραση μεταξύ της εξόδου του LLM και της αναφοράς, χρήσιμο σε πολλές πραγματικές καταστάσεις. Το LLMA προσφέρει έναν μηχανισμό αποκωδικοποίησης εκτίμησης με αναφορά, ο οποίος επιταχύνει την εκτίμηση του LLM μέσω της αλληλεπίδρασης με την αναφορά, χωρίς την ανάγκη νέων μοντέλων.

Αυτός ο μηχανισμός επιλέγει κείμενο, μεταφέρει τα διακριτικά στον αποκωδικοποιητή του LLM και καθορίζει την αποδοχή τους βάσει των πιθανοτήτων των εξόδων. Η τεχνική επιτρέπει αποτελεσματική παράλληλη εκτέλεση σε επιταχυντές διανυσμάτων όπως οι GPU, ενώ διατηρεί τα ίδια αποτελέσματα με τη γρήγορη αποκωδικοποίηση. Το LLMA είναι απλό στην υλοποίηση και ανάπτυξη, καθιστώντας το μια πρακτική επιλογή για την επιτάχυνση της εκτίμησης του LLM. Οι μελέτες δείχνουν ότι το LLMA μπορεί να προσφέρει πάνω από 2 φορές ταχύτερη εκτέλεση σε διαφορετικά μεγέθη μοντέλων, χρησιμοποιώντας ως επεξεργαστή την Nvidia 32G V100 GPU.

6.5 FlexGen

Το 2023, οι ερευνητές παρουσίασαν το μοντέλο FlexGen [23]ως ένα σύστημα υψηλής ροής για την παραγωγή LLMs, σχεδιασμένο για την επεξεργασίας χρόνου αναμονής σε περιβάλλοντα με περιορισμένους πόρους. Το FlexGen δημιουργεί 32 διακριτικά για κάθε προτροπή, δοκιμάζοντας μήκη 512

και 1024, και αξιολογεί τη ροή παραγωγής ως το πλήθος των δημιουργημένων διακριτικών διαιρεμένο με τον χρόνο προσαρμογής και αποκωδικοποίησης. Σε σύγκριση με τα DeepSpeed Zero-Interface και Hugging Face Accelerate, το FlexGen παρέχει 40 φορές μεγαλύτερη ροή με τον ίδιο χρόνο αναμονής, χρησιμοποιώντας μια Nvidia T4 (16GB) GPU. Χτισμένο πάνω σε PyTorch, το μοντέλο αυτό αξιοποιεί πολλαπλά CUDA streams και CPU threads για τον συνδυασμό I/O, αυξάνοντας σημαντικά την απόδοση μέσω της υπολογιστικής ισχύος της CPU και της επικάλυψης αποτελεσμάτων.

6.6 vLLMs

Το 2023, οι ερευνητές παρουσίασαν το μοντέλο vLLMs[24] ως μια λύση για την αποτελεσματική διαχείριση μνήμης κατά την εξυπηρέτηση LLM. Οι συγγραφείς προτείνουν τη χρήση μιας στρατηγικής διαχείρισης μνήμης γνωστής ως PagedAttention, η οποία διαχωρίζει την προσοχή κλειδιού-τιμής (KV) σε μπλοκ σταθερού μεγέθους και χρησιμοποιεί τη σελιδοποίηση για τη διατήρησή τους. Αυτή η προσέγγιση βελτιώνει την αποδοτικότητα της μνήμης. Επίσης, το μοντέλο αυτό υποστηρίζει μεικτές προσεγγίσεις αποκωδικοποίησης με διάφορα πρότυπα κοινής χρήσης και πρόσβασης στη μνήμη, χρησιμοποιώντας ένα επίπεδο αντιστοίχισης για τη μετατροπή των λογικών μπλοκ σε φυσικά μπλοκ, βελτιστοποιώντας περαιτέρω τη χρήση της μνήμης και μειώνοντας το συνολικό αποτύπωμα μνήμης των LLMs.

6.7 ALISA

Το 2024, οι ερευνητές παρουσίασαν το μοντέλο ALISA[25], μια μέθοδο για την επιτάχυνση μεγάλων γλωσσικών μοντέλων (LLMs) χρησιμοποιώντας αραιή προσοχή παραθύρων και δυναμικό χρονοδρομολόγησης. Η έρευνα υποστηρίζει ότι οι υπάρχουσες αλγοριθμικές βελτιστοποιήσεις δεν προσφέρουν ανταγωνιστική ακρίβεια στα LLMs. Ως λύση, προτείνουν την προσέγγιση SWA, η οποία δημιουργεί αραιά μοτίβα που είναι τοπικά στατικά και παγκοσμίως δυναμικά.

Η δυναμική χρονοδρομολόγηση βελτιώνει την απόδοση, ισορροπώντας την πρόσβαση στη μνήμη και την επεξεργασία σε επίπεδο tokens. Το σύστημα ALISA συνδυάζει την SWA, τη δυναμική χρονοδρομολόγηση και τη συμπίεση KV, μειώνοντας σημαντικά το συνολικό αποτύπωμα μνήμης των KV κατανομέων. Η έρευνα δείχνει ότι η προτεινόμενη τεχνική υπερνικά προηγούμενες μεθόδους ως προς την ακρίβεια και την απόδοση, συγκρίνοντάς την με τρεις οικογένειες μοντέλων LLM ανοικτού κώδικα.

ΚΕΦΑΛΑΙΟ 7: Σύγκριση των μοντέλων

Στο συγκεκριμένο κεφάλαιο θα γίνει σύγκριση όλων των μοντέλων που έχουν αναφερθεί παραπάνω με βάση τον ρυθμό επιτάχυνσης τους και την ενεργειακή τους απόδοση, καθώς και θα αναπαρασταθούν διαγράμματα στην τεχνολογία των 7nm και των 16nm.

7.1 Σύγκριση ρυθμού επεξεργασίας - τεχνολογίας

Ο *πίνακας 1* δείχνει όλα τα μοντέλα που έχουν αναφερθεί παραπάνω, καθώς και την τεχνολογία που χρησιμοποιούν, το ρυθμό επεξεργασίας και την ενεργειακή απόδοση:

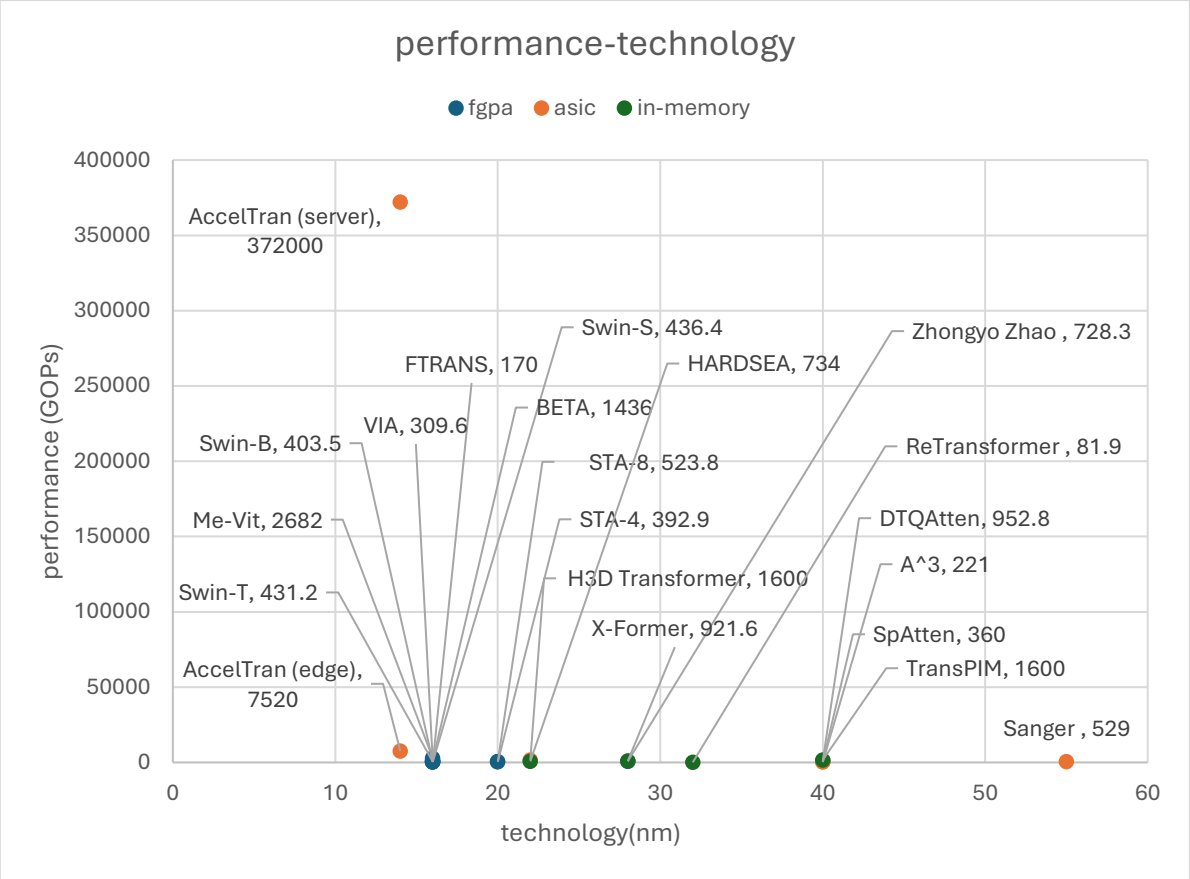
Πίνακας 1

Όνομα μοντέλου	Τύπος επιταχυντή	Τεχνολογία	Ρυθμός Επιτάχυνσης (GOPS)	Ενεργειακή απόδοση (GOPS/W)
FTRANS	FPGA	VCU118 (16nm)	170.0	6.8
VIA	FPGA	Alveo U50 (16nm)	309.6	7.9
BETA	FPGA	ZCU102(16nm)	1,436.0	174.0
NPE	FPGA	Zynq Z-7100(28nm)	-	-
Me-Vit	FPGA	Alveo U200(16nm)	2,682.0	-
Swin-T	FPGA	XCZU19EG (16nm)	431.2	-
Swin-S	FPGA	XCZU19EG (16nm)	436.4	-
Swin-B	FPGA	XCZU19EG (16nm)	403.5	-
STA-4	FPGA	Arria 10SX660(20nm)	392.9	33.6
STA-8	FPGA	Arria 10SX660(20nm)	523.8	41.2
Zhongyo Zhao	FPGA	Virtex™ 7VC707 (28nm)	728.3	58.3
Sanger	ASIC	UMC (55nm)	529.0	-
SpAtten	ASIC	TSMC (40nm)	360.0	382.0
A^3	ASIC	TSMC (40nm)	221.0	269.0

AccelTran (edge)	ASIC	FinFET(14nm)	7,520.0	-
AccelTran (server)	ASIC	FinFET(14nm)	372,000.0	-
H3D Transformer	ASIC	FeFET (22nm)	1,600.0	10,000.0
DTQAtten	ASIC	TSMC(40nm)	952.8	1,298.4
ReTransformer	In-Memory	ReRam(32nm)	81.9	467.7
X-Former	In-Memory	CMOS(32nm)	-	13,440.0
HARDSEA	In-Memory	COMS Verilog RTL (28nm)	921.6	943.7
TransPIM	In-Memory	Cacti-3DD(22nm)	734.0	-
H3DAtten	In-Memory	TSMC(40nm)	1,600.0	7,100.0
TranCim	In-Memory	CMOS(28nm)	-	20,500.0
Primate	In-Memory	HBM2E(10nm)	-	-
iMTransformer	In-Memory	CAMS (12nm)	-	-
NeuPIM	GPU	Cacti 7.0(22nm)	-	-
ALISA	GPU	H100 (4nm)	-	-
Jaewan Choi	GPU	A100(7nm)	-	-
Nan Yang	GPU	V100(12nm)	-	-
LightSeq2	GPU	A100(7nm)	-	-
FlexGen	GPU	12nm	-	-
vLLMs	GPU	A100(7nm)	-	-
TurboTransformer	GPU	RTX 2060(16nm)	-	-

Σημείωση: Στη συγκεκριμένη διπλωματική εργασία έγινε σύγκριση μεταξύ απόδοσης–τεχνολογίας και ενεργειακής απόδοσης-τεχνολογίας και όχι με βάση την επιτάχυνση (speedup) που πετυχαίνει κάθε μοντέλο. Αυτό έγινε καθώς εάν χρησιμοποιούσαμε ως μέτρο σύγκρισης την επιτάχυνση δεν θα μπορούσαμε να κάνουμε σωστές συγκρίσεις και αντιστοιχίσεις.

Με βάση τον πίνακα παραπάνω παρατίθενται κάποια διάγραμμα. Αυτά τα διαγράμματα αναφέρονται στην επίδοση της ταχύτητας σε σύγκριση με την τεχνολογία που βρίσκεται κάθε μοντέλο.



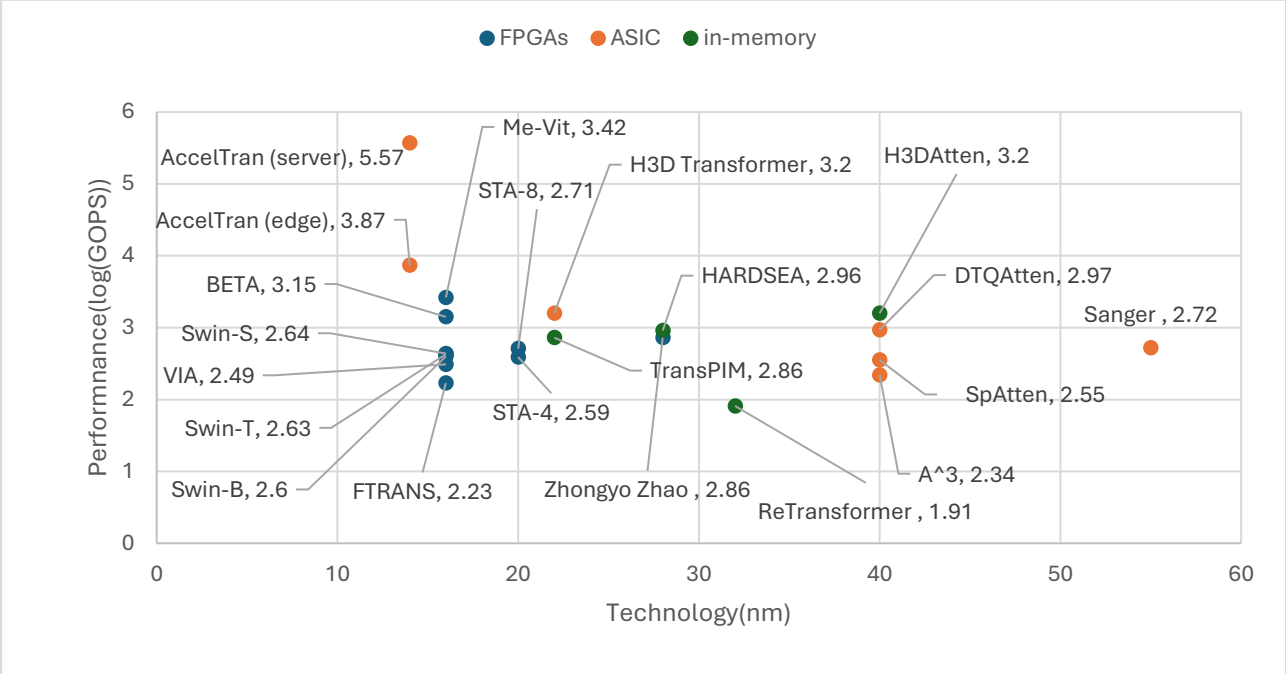
Σχεδιάγραμμα 1 : Απόδοση- Τεχνολογίας

Παρατηρούμε ότι με βάση την τεχνολογία όλα τα μοντέλα έχουν διαφορετική απόδοση. Την πιο υψηλή απόδοση την έχει το μοντέλου AccelTran (server) με 372,000 και την πιο χαμηλή το μοντέλο ReTransformer. Ακόμη, παρατηρείται ότι τα μοντέλα που βρίσκονται στην ίδια τεχνολογία όπως, το ViA, Me-ViT, Ftrans και άλλα, δεν έχουν παρόμοιο ρυθμό επιτάχυνσης και αυτό οφείλεται στο γεγονός ότι δε χρησιμοποιούν τον ίδιο επεξεργαστή ως μέτρο σύγκρισης. Επίσης, φαίνεται πως το μοντέλο Swin-S έχει καλύτερη απόδοση, όμως η σύγκριση δε γίνεται με τον ίδιο επεξεργαστή, άρα δεν μπορεί να είναι δίκαιη. Παρόλα αυτά, δε μπορούμε να πούμε ότι είναι δίκαιη η σύγκριση που κάνουμε, καθώς τα μοντέλα δε χρησιμοποιούν την ίδια τεχνολογία ή τον ίδιο επεξεργαστή.

Όμως, για να έχουμε καλύτερα αποτελέσματα, πήραμε τον λογάριθμο της απόδοσης σε σύγκριση με την τεχνολογία, οι οποίες τιμές φαίνονται στον παρακάτω πίνακα :

Πίνακας 2

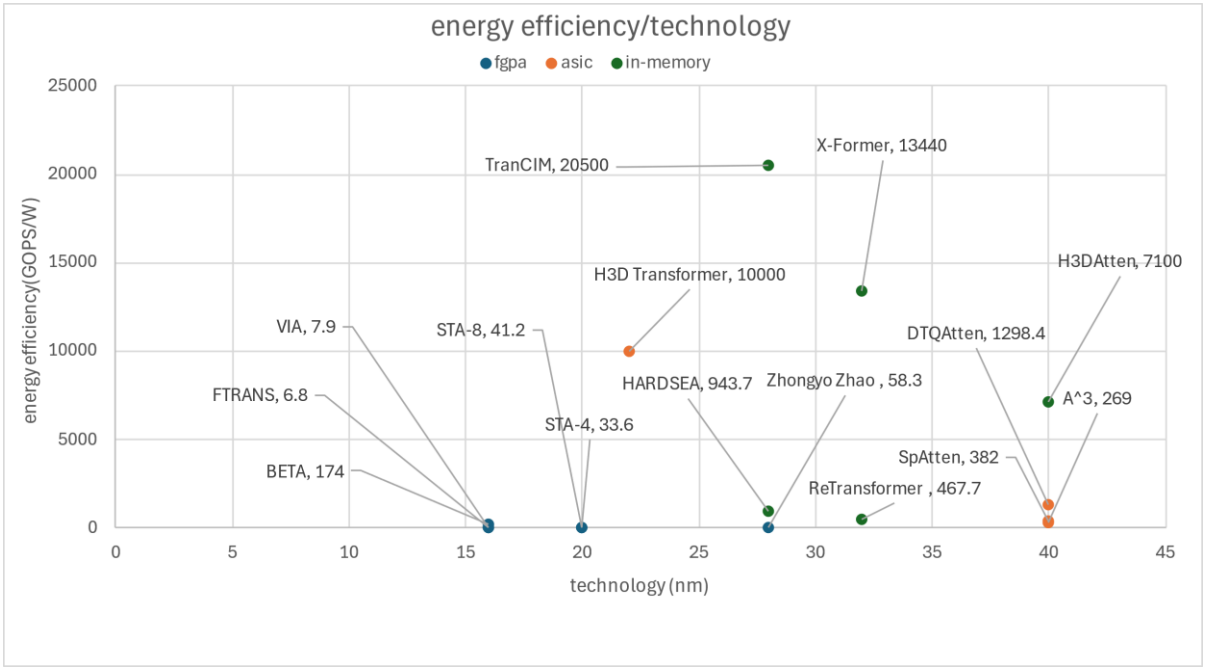
Name	Performance	logPerformance
Retransformer	81.9	1.91
STA-4	392.9	2.59
STA-8	523.8	2.71
Ftrans	170	2.23
ViA	309.6	2.49
Beta	1436	3.15
AccelTran (server)	372000	5.57
AccelTran (edge)	7520	3.87
Me-ViT	2682	3.42
Swin-T	431.2	2.63
Swin-S	436.4	2.64
Swin-B	403.5	2.6
H3D Transformer	1600	4
Sanger	529	2.72
SpAtten	360	2.55
Zhongyo Zhao	728.3	58.3
DTQAtten	952.8	2.97
Hardsea	921.6	2.96
TransPIM	734	2.86
A3	221	2.34



Σχεδιάγραμμα 2: Απόδοση- Τεχνολογία σε λογαριθμική κλίμακα

7.2 Σύγκριση ενεργειακής απόδοσης - τεχνολογίας

Ακόμη, συγκρίνουμε την ενεργειακή απόδοση με την αντίστοιχη τεχνολογία στην οποία βρίσκεται το κάθε μοντέλο με βάση τον πίνακα.



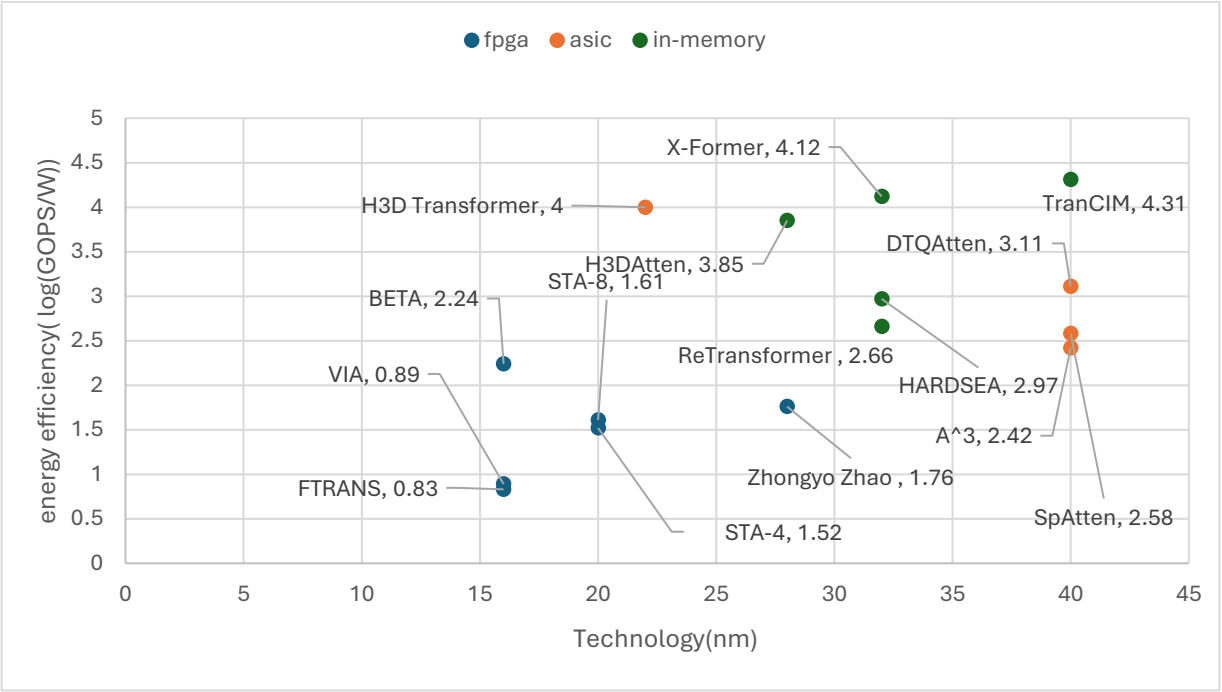
Σχεδιάγραμμα 3: Ενεργειακή απόδοση - Τεχνολογία

Παρατηρούμε ότι όταν τα μοντέλα που χρησιμοποιούν την μνήμη κυρίως (In-Memory Accelerators), έχουν υψηλότερη ενεργειακή απόδοση. Αυτό συμβαίνει γιατί έχουν μειωμένη κίνηση

δεδομένων καθώς η συγκεκριμένη αρχιτεκτονική επιτρέπει στα δεδομένα να επεξεργάζονται απευθείας στην μνήμη, χωρίς να μεταφέρονται από την μνήμη στην κεντρική μονάδα επεξεργασίας (CPU). Για να έχουμε καλύτερη απεικόνιση υπολογίσαμε τον λογάριθμο της ενεργειακής απόδοσης σε σύγκριση με την τεχνολογία.

Πίνακας 3

Name	Energy efficiency	Log(e.e)
FTRANS	6.8	0.83
VIA	7.9	0.89
BETA	174	2.24
STA-4	33.6	1.52
STA-8	41.2	1.61
Zhongyo Zhao	58.3	1.76
SpAtten	382	2.58
A^3	269	2.42
H3D Transformer	10000	4
DTQAtten	1298.4	3.11
ReTransformer	467.7	2.66
X-Former	13440	4.12
HARDSEA	943.7	2.97
H3DAtten	7100	3.85
TranCIM	20500	4.31



Σχεδιάγραμμα 4: Ενεργειακή απόδοση - Τεχνολογία σε λογαριθμική κλίμακα

Προφανώς, έχουμε πιο κατανοητά νούμερα, με μικρότερη απόκλιση και έτσι είναι πιο εύκολο να παρατηρήσουμε ότι ισχυριστήκαμε στην προηγούμενη παρατήρηση.

7.3 Σύγκριση απόδοσης στη τεχνολογία 7nm

Είναι σημαντικό να αναφερθεί ότι δεν υπάρχει ένας εύκολος τρόπος σύγκρισης της επιτάχυνσης και της ενεργειακής απόδοσης, όταν τα μοντέλα που υπάρχουν έχουν το καθένα διαφορετικά χαρακτηριστικά. Γι' αυτό τον λόγο, θα γίνει η μετατροπή τους σε μία συγκεκριμένη τεχνολογία, ώστε να μπορέσουμε να τα συγκρίνουμε καλύτερα.

Με βάση το άρθρο των Aaron Stillmaker και B.Baas με τίτλο : «Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7nm» μετατρέψαμε τις διάφορες τιμές του ρυθμού επιτάχυνσης, αλλά και της ενεργειακής απόδοσης, σε μία συγκεκριμένη τιμή της τεχνολογίας των 7nm, ώστε να κάνουμε μια πιο δίκαια σύγκριση της επιτάχυνσης και της ενεργειακής απόδοσης και να ληφθούν καλύτερα συμπεράσματα.

Με βάση τον πίνακα 2 του άρθρου, ο οποίος φαίνεται παρακάτω, καθώς και τον πίνακα 5, αλλά και τις εξισώσεις που θα γραφτούν παρακάτω, κάναμε την μετατροπή και τα αντίστοιχα διαγράμματα. Όσο αφορά τον πίνακα 2, εμείς χρησιμοποιήσαμε την τάση που αντιστοιχεί σε κάθε τεχνολογία που έχουμε. Στην δική μας περίπτωση χρησιμοποιήσαμε την τάση των 32nm, 20nm, 14nm και 7nm, καθώς για τις άλλες τεχνολογίες δεν υπάρχουν τα αντίστοιχα νούμερα.

Table 2
Characteristics of different technology nodes [23]. The modeled measurements are for a single inverter in an FO4 chain. The energy value is the average energy required for a single inverter transition from low to high, or high to low.

				SimulatedPerformance of Inverter		
Production Year	Technology Node (nm)	Technology Type	V_{DD} (V)	Delay (ps)	Energy (fJ)	Power (μW)
1999	180	Bulk	1.8	77.2	27.5	105
2001	130	Bulk	1.2	34.7	5.20	26.1
2004	90	Bulk	1.1	26.5	2.62	13.0
2007	65	Bulk	1.1	19.8	1.72	8.58
2008	45	High-k	1.1	10.9	1.05	5.19
2010	32	High-k	0.97	9.8	0.51	2.47
2012	20	Multi-Gate	0.9	9.66	0.198	1.51
2013	16[a]	Multi-Gate	0.86	6.12	0.179	1.28
2013	14[a]	Multi-Gate	0.86	4.02	0.144	0.995
2015	10	Multi-Gate	0.83	3.24	0.122	0.866
2017	7	Multi-Gate	0.8	2.47	0.111	0.789

Εικόνα 6: πίνακας χαρακτηριστικών με βάση έτος – τεχνολογία

Πηγή : VLSI-Scaling-Stillmaker.pdf (ucdavis.edu)

Για τον πίνακα 5, χρησιμοποιήσαμε για τις τεχνολογίες 32, 20, 16, 14, 7 (nm), τους αντίστοιχους αριθμούς που σχετίζονται με τους συντελεστές που χρειάζονται για την εξίσωση 5 και 8 του συγκεκριμένου άρθρου.

Table 5
The polynomial coefficient values to be used with Eqs. (5)–(7) to attain the factors to be used to generate the scaling factor between two technology nodes and voltages.

			Delay Coefficients (Eq. (5))				Energy Coefficients (Eq. (6))			Power Coefficients (Eq. (7))		
Type		Node	a_{d3}	a_{d2}	a_{d1}	a_{d0}	a_{e2}	a_{e1}	a_{e0}	a_{p2}	a_{p1}	a_{p0}
Bulk		180 nm	–	97.09	−356.7	406.5	–	24.64	−17.98	–	101000	−79720
		130 nm	−76.65	334.9	−493.4	275.8	7.171	−6.709	2.904	27020	−15450	5630
		90 nm	−60.34	262.5	−384.2	210.9	4.762	−4.781	2.092	17320	−11230	4328
		65 nm	−53.3	230.4	−333.9	178.6	3.755	−4.398	1.975	12890	−10510	4362
High-k	HP	45 nm	−501.6	1567	−1619	566.1	1.018	−0.3107	0.1539	5462	−1760	522.4
		32 nm	−1047	2982	−2797	873.5	0.8367	−0.4341	0.1701	4001	−1733	533.6
	LP	45 nm	−285.7	1239	−1795	898.8	1.103	−0.362	0.2767	6297	−3009	1124
		32 nm	−325.9	1374	−1922	913.2	0.9559	−0.7823	0.471	4557	−3037	1323
Multi-Gate	HP	20 nm	–	34.63	−66.37	41.15	0.373	−0.1582	0.04104	2922	−1286	299.9
		16 nm	–	24.8	−47.52	28.87	0.2958	−0.1241	0.03024	2133	−882.6	197.7
		14 nm	−40.66	109.2	−100.6	35.92	0.2363	−0.09675	0.02239	1675	−711	159
		10 nm	−34.95	93.65	−85.99	30.4	0.2068	−0.09311	0.02375	1456	−621.6	143.8
		7 nm	−28.58	76.6	−70.26	24.69	0.1776	−0.09097	0.02447	1179	−515.7	123.4
	LSTP	20 nm	−160.5	514.1	−558.6	217.5	0.2632	−0.14	0.06841	2096	−962.4	287.1
		16 nm	−114.6	366.7	−397.4	153.6	0.2139	−0.1187	0.05639	1609	−715.5	205.7
		14 nm	−85.37	271.6	−292.2	111.4	0.1556	−0.06472	0.03066	1259	−554.1	152.3
		10 nm	−71.76	228.6	−246.3	93.91	0.1261	−0.0518	0.02769	1046	−422.7	118.9
		7 nm	−61.79	196.1	−210.3	79.55	0.09365	−0.03409	0.02043	815.2	−307.3	87.54

Εικόνα 7 : Συντελεστές καθυστέρησης και ενέργειας με βάση την τεχνολογία

Πηγή : VLSI-Scaling-Stillmaker.pdf (ucdavis.edu)

Οι εξισώσεις που χρειάστηκαν είναι οι εξής:

Για να βρούμε την νέα επιτάχυνση :

$$Dx = \frac{DelayFactorx}{DelayFactory} Dy \ (2)$$

Αφού ορίσουμε πρώτα τον παράγοντα καθυστέρησης, ο οποίο ορίζεται ως :

$$DelayFactor = a_{d3}V^3 + a_{d2}V^2 + a_{d1}V + a_{d0} \ (3)$$

Στη συνέχεια, θα βρούμε τις τιμές που αντιστοιχούν στην ανάλογη τεχνολογία. Για την κάθε τεχνολογία αντιστοιχούνται κάποιοι συντελεστές καθυστέρησης στην ανάλογη τάση που χρησιμοποιούν. Έτσι, με βάση τον πίνακα του άρθρου έχουμε τα εξής :

Πίνακας 4

Technology	V_{dd}	A_{d3}	A_{d2}	A_{d1}	A_{d0}
7	0.8	-28.58	76.6	-70.26	24.69
14	0.86	-40.66	109.2	-100.6	35.92
16	0.86	-	24.8	-47.52	28.87
20	0.9	-	34.63	-66.37	41.15
32	0.97	1047	2982	2797	873.5

Σημείωση: χρησιμοποιήσαμε το HP (High Permormance), γιατί όλα τα μοντέλα που έχουμε επιλέξει είναι υψηλής απόδοσης.

Έτσι, ξεκινώντας από την τεχνολογία των 7nm που θέλουμε να συγκρίνουμε όλα τα υπάρχοντα μοντέλα και καταλήγοντας στην τεχνολογία των 32nm γίνονται οι παρακάτω πράξεις:

$DelayFactor_x^{(7nm)} = -28.58x0.8^3 + 76.6x0.8^2 - 70.26x0.8 + 24.69 \Rightarrow$ **$DelayFactor_{(7nm)} = 4.409$**

$DelayFactor_y^{(14nm)} = -40.66x0.86^3 + 109.2x0.86^2 - 100.6x0.86 + 35.92 \Rightarrow$ **$DelayFactor_{(14nm)} = 4.31$**

$DelayFactor_y^{(16nm)} = 24.8x0.86^2 - 47.52x0.86 + 28.87 \Rightarrow$ **$DelayFactor_{(16nm)} = 6.34$**

$DelayFactor_y^{(20nm)} = 34.63x0.9^2 - 66.37x0.9 + 41.15 \Rightarrow$ **$DelayFactor_{(20nm)} = 9.467$**

$DelayFactor_y^{(32nm)} = -1047x0.97^3 + 2982x0.97^2 - 2797x0.97 + 873.5 \Rightarrow$ **$DelayFactor_{(32nm)} = 11.46$**

Χρησιμοποιώντας την εξίσωση (2) $Dx = \frac{DelayFactorx}{DelayFactory} x\ Dy$ και αντικαθιστώντας για Dy την ήδη υπάρχουσα απόδοση, για $DelayFactor_x$ την τεχνολογία στην οποία θέλουμε να φτάσουμε, δηλαδή

στα 7nm και για $DelayFactor_y$ τη τεχνολογία στην οποία βρίσκεται ήδη το μοντέλο που θα συγκρίνουμε, καταλήξαμε στα παρακάτω αποτελέσματα:

Τα μοντέλα που χρησιμοποιούν την τεχνολογία των 32nm :

1. **Retransformer** : Dx= $\frac{4.409}{11.46}$x 81.9 = **31.50**

Τα μοντέλα που χρησιμοποιούν την τεχνολογία των 20nm :

1. **STA-4 : Dx =**$\frac{4.409}{9.467}$x 392.9 = **182.98**
2. **STA-8 : Dx =** $\frac{4.409}{9.467}$x 523.8 = **243.94**

Τα μοντέλα που χρησιμοποιούν την τεχνολογία των 16nm :

1. **Ftrans : Dx =** $\frac{4.409}{6.34}$x 170 = **118.2**
2. **ViA : Dx =** $\frac{4.409}{6.34}$x 309.6 =**215.172**
3. **Beta : Dx =** $\frac{4.409}{6.34}$x 1436 =**898.02**
4. **Me-Vit : Dx =** $\frac{4.409}{6.34}$x 2682= **1863.99**
5. **Swin-T: Dx =** $\frac{4.409}{6.34}$x 431.2 = **299.684**
6. **Swin-S: Dx =** $\frac{4.409}{6.34}$x 436.4=**303.298**
7. **Swin-B: Dx =** $\frac{4.409}{6.34}$x 403.5= **280.432**

Τα μοντέλα που χρησιμοποιούν την τεχνολογία των 14nm :

1. **AccelTran(server) : Dx** = $\frac{4.409}{4.31}$x 372000 = **380544.78**
2. **AccelTran(edge) : Dx =** $\frac{4.409}{4.31}$x 7343.04 = **7493.98**

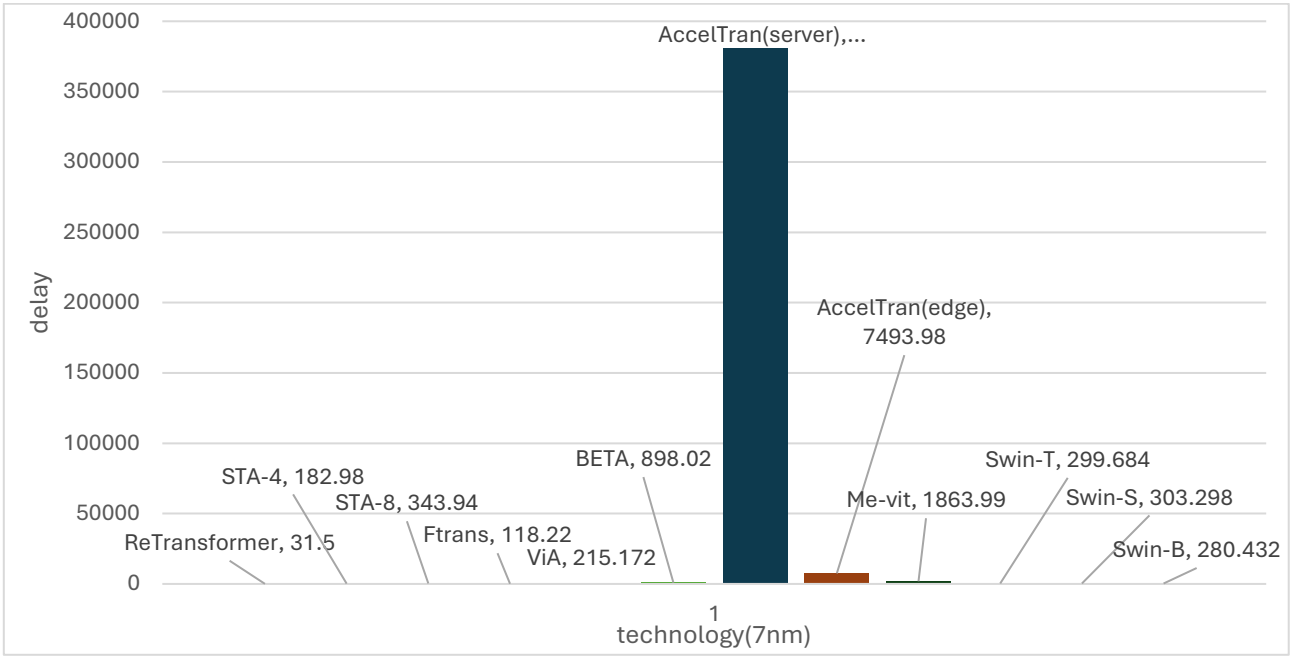
<u>Σημείωση:</u> η απόδοση είναι το αντίστροφο της καθυστέρησης και όσο μικραίνει η τεχνολογία είναι λογικό να αυξάνεται η απόδοση.

Συνοπτικά, τα αποτελέσματα που πήραμε τα περάσαμε στο παρακάτω πίνακα για να πάρουμε το αντίστοιχα διαγράμματα στη τεχνολογία των 7nm.

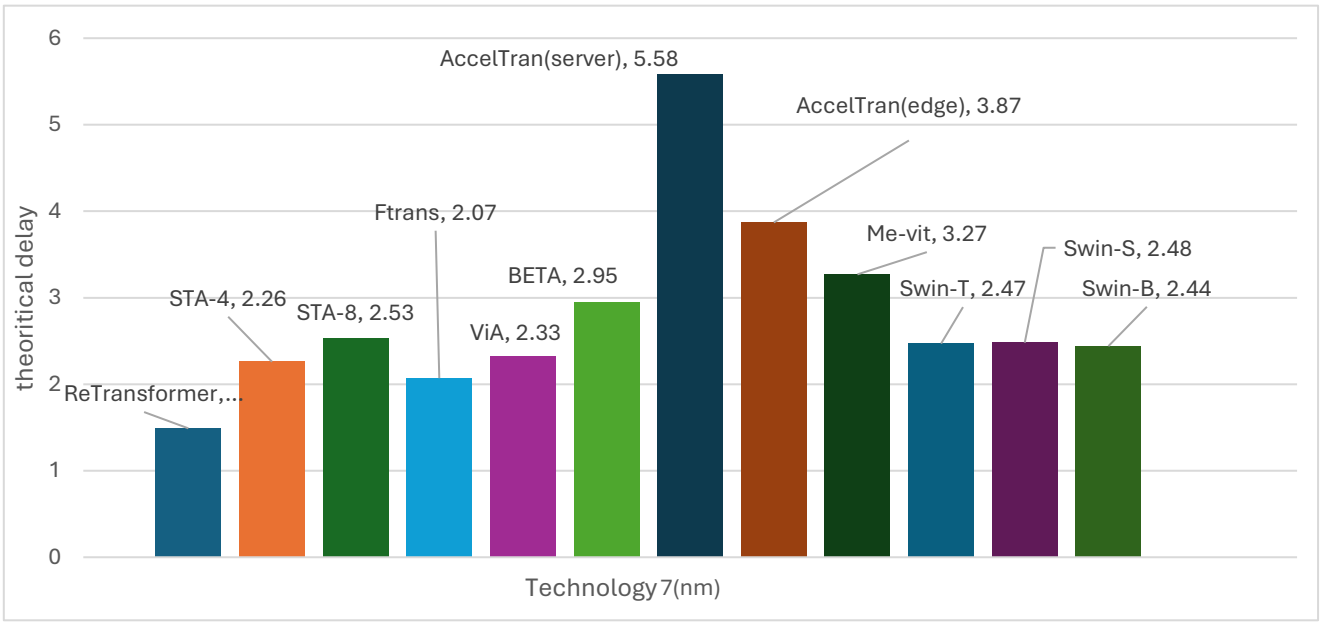
Πίνακας 5

Technology (nm)	New technology (nm)	Name	Dx_{new}	Log(Dx)	Performance (GOPs)	logPerf
32	7	ReTranformer	31.50	1.49	212.87	2.32
20	7	STA-4	182.98	2.26	843.63	2.92
20	7	STA-8	243.94	2.53	1124.70	3.05
16	7	Ftrans	118.22	2.07	244.45	2.38
16	7	ViA	215.17	2.33	445.19	2.64
16	7	Beta	898.02	2.95	2064.92	3.31
16	7	Me-ViT	1863.99	3.27	3856.6	3.58
16	7	Swin-T	299.684	2.47	620.05	2.79
16	7	Swin-B	303.298	2.48	627.52	2.79
16	7	Swin-S	280.432	2.44	580.22	2.76
14	7	AccelTran (server)	380544.78	5.58	363647.08	5.56
14	7	AccelTran (edge)	7493.98	3.87	7178.15	3.85

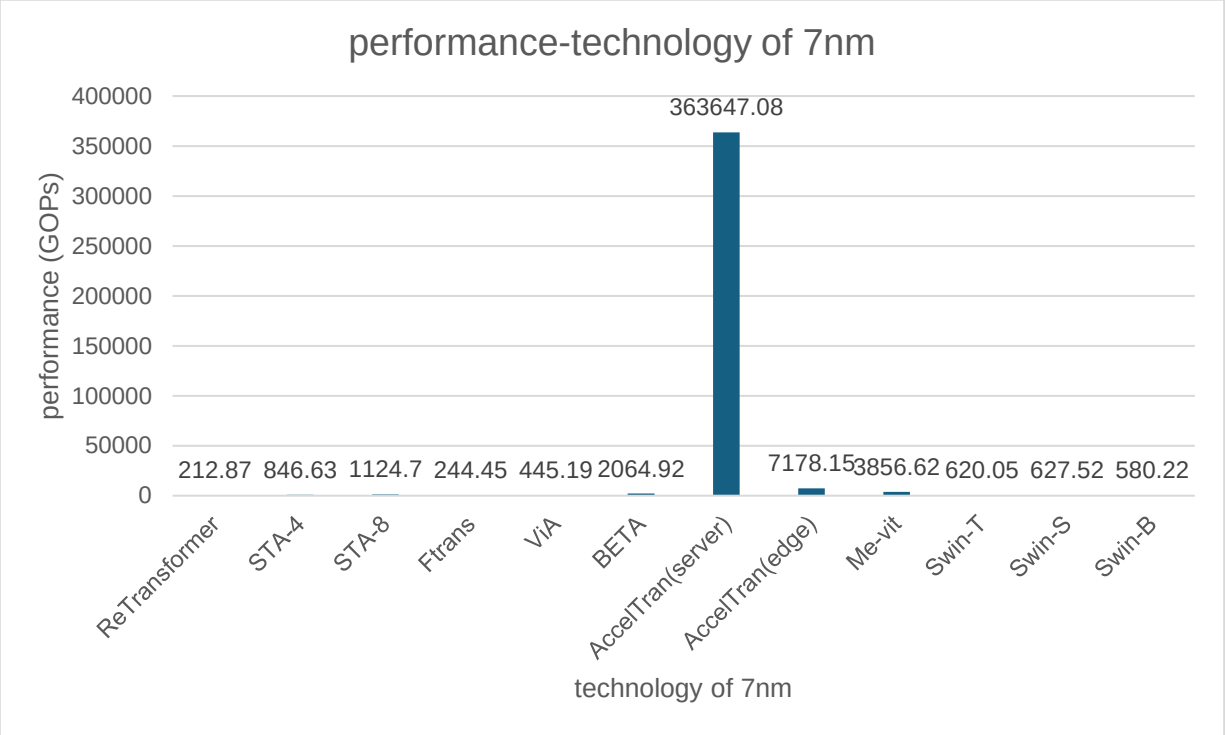
Δίνεται το διάγραμμα της επίδοσης ταχύτητας στην τεχνολογία των 7nm.



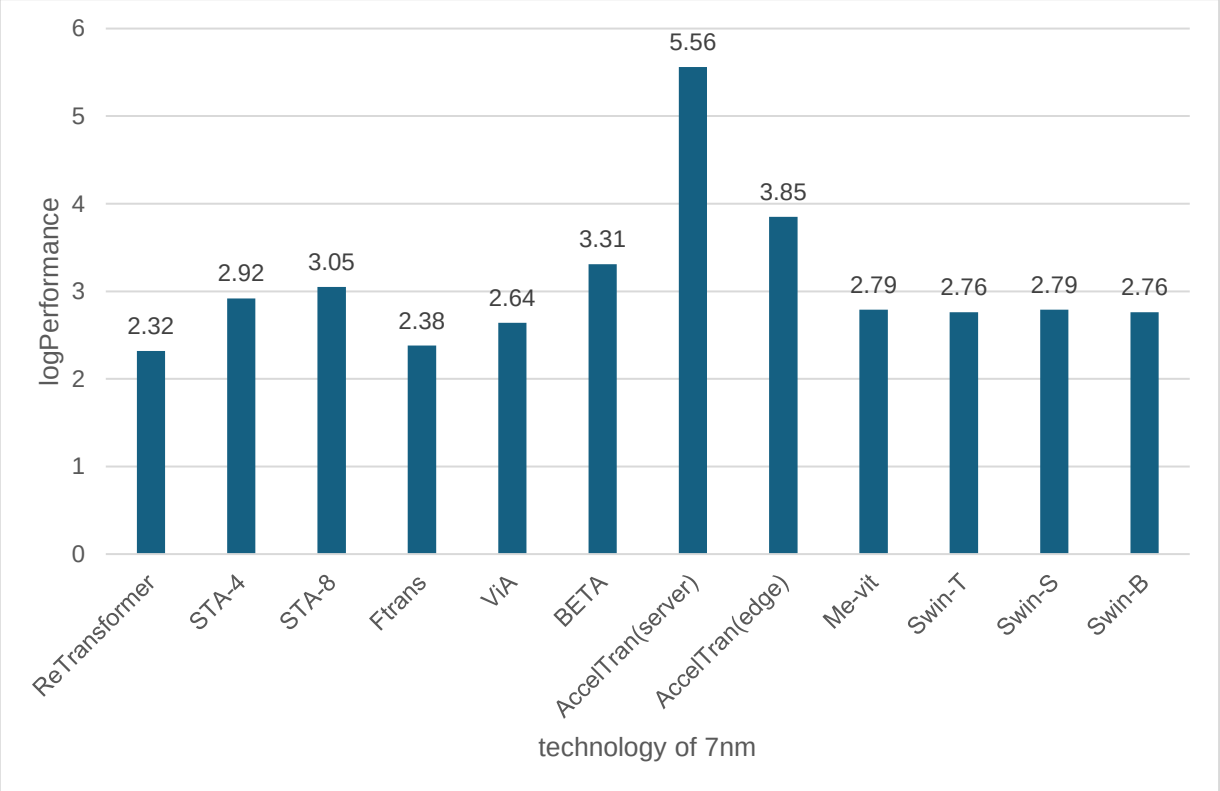
Σχεδιάγραμμα 5 : καθυστέρησης- Τεχνολογίας με αναγωγή στα 7nm



Σχεδιάγραμμα 6 : Θεωρητική καθυστέρηση (λογαριθμική κλίμακα) – Τεχνολογία στα 7nm



Σχεδιάγραμμα 7 : Απόδοσης- Τεχνολογίας με αναγωγή στα 7nm



Σχεδιάγραμμα 8 : λογάριθμος απόδοσης- Τεχνολογίας με αναγωγή στα 7nm

7.4 Σύγκριση ενεργειακής απόδοσης στη τεχνολογία 7nm

Στη συνέχεια, συγκρίθηκε η ενεργειακή απόδοση των μοντέλων που έχουν αναφερθεί παραπάνω στη τεχνολογία των 7nm.Για να γίνει αυτό, χρειάστηκαν οι συντελεστές της ενεργείας που είναι απαραίτητοι για τη μετατροπή τους στη τεχνολογία που έχουμε επιλέξει. Έτσι, ο παρακάτω πίνακας δείχνει συνοπτικά αναλόγως με τι τεχνολογία χρησιμοποιείται το μοντέλο, τι τάση χρειάζεται και τους αντίστοιχους συντελεστές.

Πίνακας 6

Τεχνολογία (nm)	Τάση VD	ae2	ae1	ae0
32	0.97	0.8367	-0.4341	0.1701
20	0.90	0.373	-0.1582	0.04104
16	0.86	0.2958	-0.1241	0.03024
7	0.80	0.1776	-0.09097	0.02447

Με βάση το επιστημονικό άρθρο των Aaron Stillmaker και B.Baas με τίτλο: «Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7nm» χρησιμοποιήθηκαν οι παρακάτω τύποι εξίσωσης για να προσδιορίσουμε την νέα ενεργειακή απόδοση.

$$Ex = \frac{EnergyyFactorx}{EnergyFactory} Ey \ (4)$$

$$EnergyFactor = a_{e2}V^2 + a_{e1}V + a_{e0} \ (5)$$

Πρώτα βρήκαμε όλους τους EnergyFactor που χρειάζονται για την μετατροπή. Αφού, θέλουμε όλα τα μοντέλα να τα μετατρέψουμε στη τεχνολογία των 7nm, ξεκινήσαμε βρίσκοντας πρώτα τον $EnergyFactor_x$(7nm) και στη συνέχεια για την τεχνολογία 16, 20, 32 (nm), και παίρνοντας τις αντίστοιχες τιμές από τον πίνακα 6 έχουμε :

$EnergyFactor_x$ (7nm) = 0.1776 x 0.8^2 -0.09097 x 0.8 + 0.2447 ⇨ **$EnergyFactor_x$(7nm)=0.065 GOPs/W**

$EnergyFactor_y$ (16nm) = 0.2958 x 0.86^2 -0.1241 x 0.86 + 0.03024 ⇨ **$EnergyFactor_x$(16nm)= 0.142 GOPs/W**

$EnergyFactor_y$ (20nm) = 0.373 x 0.90^2 -0.1582 x 0.90 + 0.04104 ⇨ **$EnergyFactor_x$(20nm)= 0.200 GOPs/W**

$EnergyFactor_y$ (32nm) = 0.8367 x 0.97^2 -0.4341 x 0.96 + 0.1701 ⇨ **$EnergyFactor_x$(7nm)= 0.536 GOPs/W**

Σημείωση : *για την εξίσωση (4) χρησιμοποιήθηκε ως $EnergyFactor_x$ η ενεργειακή απόδοση των 7nm (δηλαδή η τεχνολογία που έχει επιλεχτεί ώστε να μετατραπούν όλα τα μοντέλα) και ως $EnergyFactor_y$ η ενεργειακή απόδοση των μοντέλων ως προς την τεχνολογία που χρησιμοποιούν.*

Παρατηρούμε ότι με βάση τον πίνακα 1 τα μοντέλα που έχουν ενεργειακή απόδοση για την μτατροπή στη τεχνολογία των 7nm είναι : Ftrans , ViA, Beta, Sta-4, Sta-8 , ReTransformer,X-Former.

Τα μοντέλα που χρησιμοποιούν την τεχνολογία των 16nm :

1. **FTRANS** : $EnergyDelay_x^{(7nm)}$ new = 0.065/0.142 x 6.8 = **3.11 GOPs/W**
2. **ViA** : $EnergyDelay_x^{(7nm)}$ new = 0.065/0.142 x 7.9 = **3.61 GOPs/W**
3. **BETA** : $EnergyDelay_x^{(7nm)}$ new = 0.065/0.142 x 174 = **76.64 GOPs/W**

Τα μοντέλα που χρησιμοποιούν τη τεχνολογία των 20nm :

1. **STA-4**: $EnergyDelay_x^{(7nm)}$ new = 0.065/0.200 x 33.6 = **10.92 GOPs/W**
2. **STA-8** : $EnergyDelay_x^{(7nm)}$ new = 0.065/0.200 x 41.2 = **13.39 GOPs/W**

Τα μοντέλα που χρησιμοποιούν τη τεχνολογία των 32nm :

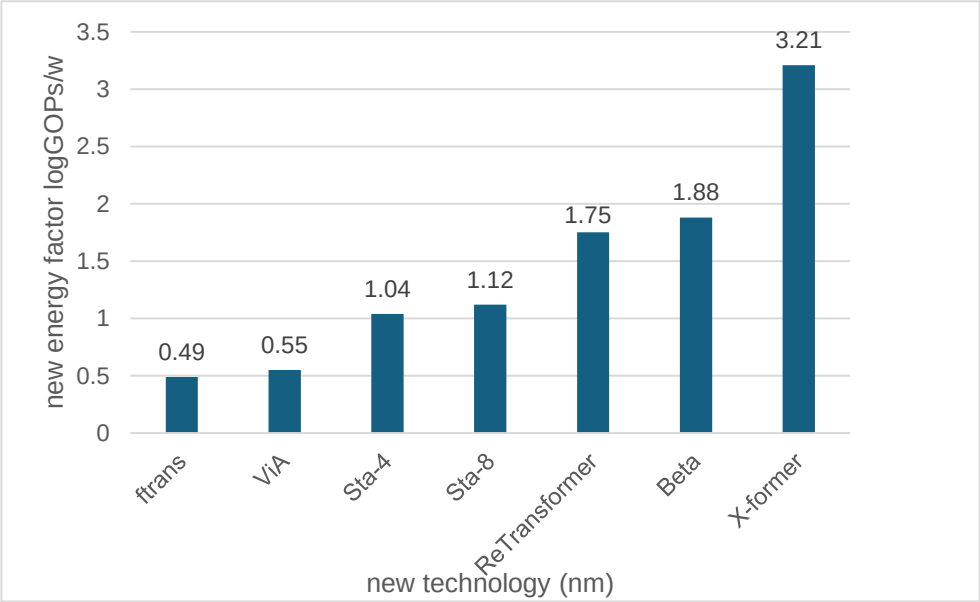
1. **ReTransformer** : $EnergyDelay_x^{(7nm)}$ new = 0.065/0.536 x 467.7 = **56.71 GOPs/W**
2. **X-Former** : $EnergyDelay_x^{(7nm)}$ new = 0.065/0.536 x 13440= **1629.85 GOPs/W**

Συνοπτικά, τα αποτελέσματα που πήραμε τα περάσαμε στο παρακάτω πίνακα για να πάρουμε το αντίστοιχο διάγραμμα στη τεχνολογία των 7nm

Πίνακας 7

Name	Technology (nm)	New Technology (nm)	EnergyFactorx **(**GOPs/W)	Log (energyfactor)
FTRANS	16	7	3.11	0.49
ViA	16	7	3.61	0.55
BETA	16	7	76.64	1.88
STA-4	20	7	10.92	1.04
STA-8	20	7	13.39	1.12
ReTransformer	32	7	56.71	1.75
X-Former	32	7	1629.85	3.21

Με βάση τον πίνακα 7 παίρνουμε το παρακάτω διάγραμμα:



Σχεδιάγραμμα 9 : new energyfactor - technology (7nm)

Παρατηρούμε ότι όλα τα μοντέλα έχουν μικρότερη ενεργειακή απόδοση στη τεχνολογία των 7nm. Αυτό συμβαίνει γιατί όσο βελτιώνεται η τεχνολογία, βελτιώνονται και τα επίπεδα κατανάλωσης ενέργειας.

Σημείωση : για τα μοντέλα που δεν αναφέρονται δεν υπάρχουν οι αντίστοιχες τεχνολογίες στο επιστημονικό άρθρο του Aaron Stillmaker.

7.5 Σύγκριση ρυθμού επεξεργασίας στη τεχνολογία 16nm για τα μοντέλα FPGAs

Επιλέξαμε να κάνουμε τη μετατροπή στα 16nm, γιατί παρατηρήσαμε ότι τα περισσότερα μοντέλα των FGPAs είναι σε 16nm. Αυτή η μετατροπή θα μας χρησιμεύσει στην πορεία της

συγκεκριμένης Διπλωματικής Εργασίας. Έτσι, με βάση το επιστημονικό άρθρο των Aaron Stillmaker και B.Baas με τίτλο : «Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7nm», και τις εξισώσεις που αναφέρονται παραπάνω θα γίνει η μετατροπή στη τεχνολογία των 16nm. Συγκεκριμένα, θα χρησιμοποιήσουμε τον πίνακα 4 και τις εξισώσεις 2-3 για την απόδοση. Έτσι, :

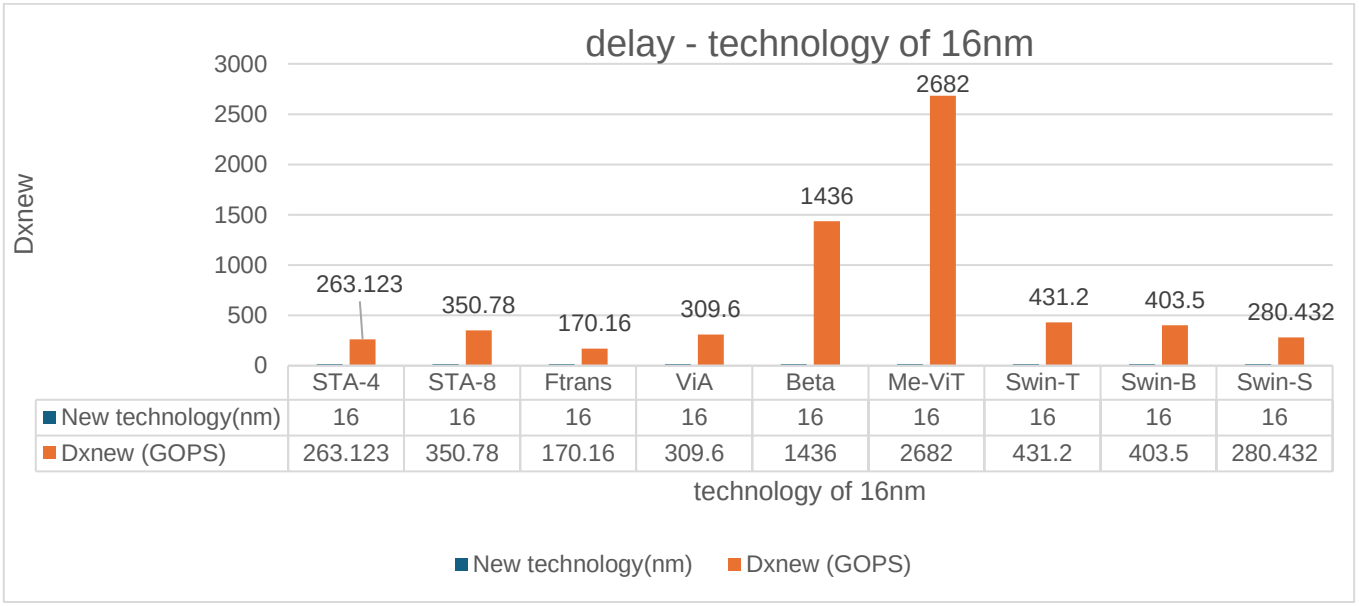
Επειδή τα περισσότερα μοντέλα είναι ήδη στη τεχνολογία των 16nm, θα μετατρέψουμε μόνο το μοντέλο του STA που χωρίζεται σε δύο διαφορετικές κατηγορίες : το STA-4 και το STA-8. Συγκεκριμένα :

- Για το STA-4 : Dx = $\frac{6.34}{9.467} x\ 392.9 =$**263.123**
- Για το STA-8 : Dx = $\frac{6.34}{9.467} x\ 523.8 =$ $\mathbf{350.78}$

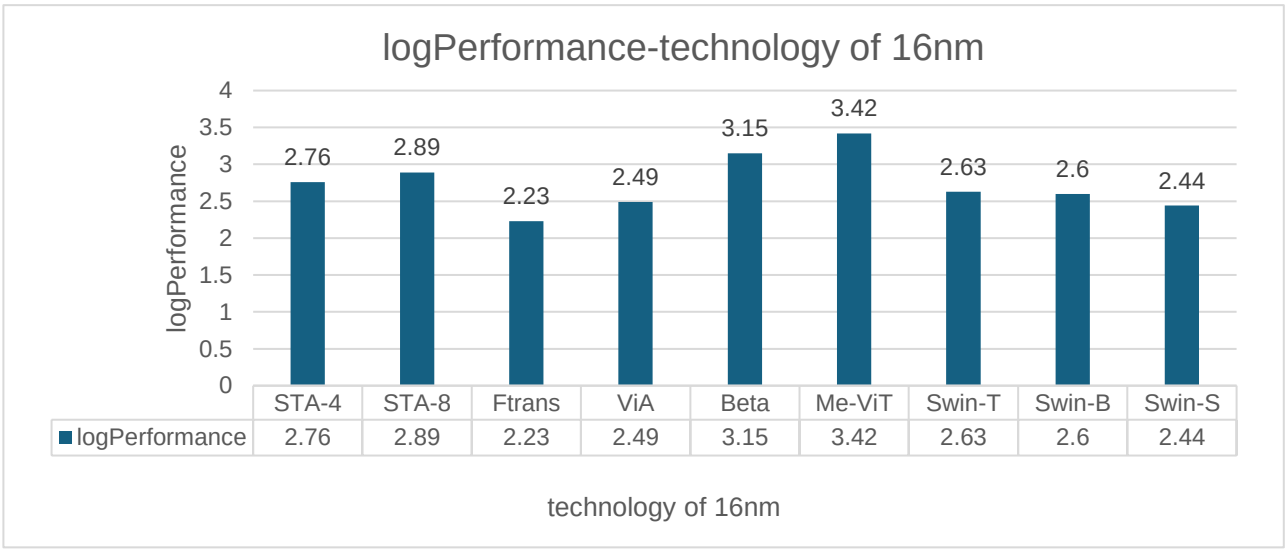
Και με βάση αυτή τη μετατροπή παίρνουμε τον παρακάτω πίνακα και το αντίστοιχο σχεδιάγραμμα:

Πίνακας 8

Technology (nm)	New technology(nm)	Name	Dx_{new}	Log(Dx)	New performance (GOPS)	logPerformance
20	16	STA-4	263.123	2.42	586.68	2.76
20	16	STA-8	350.78	2.54	782.147	2.89
16	16	Ftrans	170.16	2.23	170	2.23
16	16	ViA	309.6	2.49	309.6	2.49
16	16	Beta	1436	3.16	1436	3.15
16	16	Me-ViT	2682	3.42	2682	3.42
16	16	Swin-T	431.2	2.63	431.2	2.63
16	16	Swin-B	403.5	2.60	403.5	2.60
16	16	Swin-S	280.432	2.44	280.432	2.44



Σχεδιάγραμμα 10: Καθυστέρησης - Τεχνολογία των 16nm



Σχεδιάγραμμα 11 : λογάριθμος ρυθμού επεξεργασίας- τεχνολογία των 16nm

ΚΕΦΑΛΑΙΟ 8 : Εφαρμογές & Σύγκριση των μοντέλων

Μέχρι τώρα, η βιβλιογραφική έρευνα και η μελέτη που έγινε στο προηγούμενο κεφάλαιο (κεφάλαιο 7) αφορούσε ψηφιακούς επεξεργαστές όπως τα FGPA, ASIC. Γι' αυτό τον λόγο, σε αυτό το κεφάλαιο, γίνονται εφαρμογές που αφορούν τους κλασσικούς επεξεργαστές όπως για παράδειγμα τους επεξεργαστές της εταιρίας Intel για να εξετάσουμε ποιοι παράγοντες επηρεάζουν, τελικά, τους επιταχυντές, αλλά και να μπορέσουμε τα θεωρητικά αποτελέσματα με τα πρακτικά έχοντας εστιάσει σε μια συγκεκριμένη τεχνολογία. Πρώτα, εκτελέστηκε κώδικας σε γλώσσα προγραμματισμού VHDL με χρήση του λογισμικού Quartus για να μπορέσουμε να δούμε αν η θεωρητική αναγωγή στη τεχνολογία των 16nm για τα μοντέλα των FPGAs είναι ορθή, αλλά και κατά πόσο διαφέρουν. Για αυτό τον λόγο, θα τρέξουμε τον κώδικα σε διάφορες τεχνολογίες που θα επιλέξουμε, οι οποίες είναι των 20 , 28, 40, 55, 65, 180 nm. Όμως, έγινε μόνο για τα μοντέλα των FPGAs, καθώς δεν διαθέτουμε τα απαραίτητα εργαλεία για να τα άλλα μοντέλα επιταχυντών, καθώς είναι πιο πολύπλοκα.

Επίσης, εκτελέσαμε μια εφαρμογή των LLM, το Ollama. Χρησιμοποιήσαμε την εφαρμογή αυτή σε διάφορους υπολογιστές κάνοντας του μια συγκεκριμένη ερώτηση, και αναλόγως τον επεξεργαστή που χρησιμοποιούν, χωρίς να γίνεται χρήση της GPU, μας έδωσαν την αντίστοιχη απάντηση, καθώς και πόσο χρόνο χρειάστηκε για να επεξεργαστεί τα δεδομένα και να απαντήσει στη συγκεκριμένη ερώτηση, μετρώντας έτσι τον ρυθμό επιτάχυνσης του εκάστοτε υπολογιστή.

Στη συνέχεια, εφαρμόσαμε με κώδικα Python σε λογισμικό Windows για να μετρήσουμε την πιθανή επιτάχυνση και το πιθανό σφάλμα της μέτρησης αυτής, εισάγοντας χαρακτηριστικά υπολογιστή που θέλουμε να έχει, χρησιμοποιώντας τέσσερις διαφορετικές μεθόδους. Στο τέλος κάθε εφαρμογής λαμβάνονται κάποια σημαντικά συμπεράσματα.

8.1 Κώδικας VHDL και σύγκριση

Είναι προφανές ότι τα περισσότερα μοντέλα των FPGAs, χρησιμοποιούν τη μέθοδο του πολλαπλασιασμού πινάκων. Γι' αυτό τον λόγο, θα εκτελέσουμε κώδικα σε γλώσσα προγραμματισμού VHDL με σκοπό να αναζητήσουμε την συχνότητα που φέρουν οι διάφορες τεχνολογίες που έχουμε επιλέξει, χρησιμοποιώντας ως πράξη πινάκων τον πολλαπλασιασμό. Έπειτα, με βάση τη συχνότητα που θα βρούμε, θα μπορέσουμε να βρούμε την καθυστέρηση (Delay), καθώς η καθυστέρηση είναι αντιστρόφως ανάλογη της συχνότητας (f= 1/T) και να κάνουμε σύγκριση της θεωρητικής αναγωγής με την εργαστηριακή αναγωγή.

Για τον κώδικα VHDL χρησιμοποιήθηκε το λογισμό Quartus της Intel και οι τεχνολογίες των 20, 28, 40, 55, 65 και 180 nm με βάση τα παρακάτω πινακάκια.

Device Family	Intel® Arria 10	Arria V GZ	Arria V GX, GT, SX, ST	Arria II GZ	Arria II GX	Arria GX
Year of introduction	2013	2012	2011	2010	2009	2007
Process technology	20 nm	28 nm	28 nm	40 nm	40 nm	90 nm
Recommended for New Designs	Yes	Yes	Yes	No	No	Discontinued

Εικόνα 8 : Intel All Arria Generations

Πηγή : https://www.intel.com/content/www/us/en/products/details/fpga/arria.html

All Stratix® Generations

Device Family	Intel® Stratix® 10	Stratix® V	Stratix® IV	Stratix® III	Stratix® II GX	Stratix® II	Stratix® GX	Stratix®
Year of Introduction	2013	2010	2008	2006	2005	2004	2003	2002
Process Technology	14 nm Tri-Gate	28 nm	40 nm	65 nm	90 nm	90 nm	130 nm	130 nm
Recommended for New Designs	Yes	No	No	No	Discontinued	Discontinued	Discontinued	Discontinued

Εικόνα 9 : Intel All Stratix Generations

Πηγή : https://www.intel.com/content/www/us/en/products/details/fpga/stratix.html

All Cyclone® FPGA Generations

Device Family	Cyclone® 10 GX FPGA	Cyclone® 10 LP FPGA	Cyclone® V FPGA	Cyclone® IV FPGA	Cyclone® III FPGA	Cyclone® II FPGA	Cyclone® FPGA
Process Technology	20 nm	65 nm	28 nm	65 nm	65 nm	90 nm	130 nm
Recommended for new designs	Yes	Yes	Yes	Yes	No	No	Discontinued

Εικόνα 10 : Intel All Cyclone FPGA Generations

Πηγή : https://www.intel.com/content/www/us/en/products/details/fpga/cyclone.html

All MAX® Generations and Key Features

Device Family	Intel® MAX 10 FPGA	MAX V CPLD	MAX II Z CPLD	MAX II CPLD
Process Technology	55 nm	180 nm	180 nm	180 nm
Key Features	Yes	Yes	No	No

Εικόνα 11 : Intel All Max Generations

Πηγή : https://www.intel.com/content/www/us/en/products/details/fpga/max.html

Ο κώδικας που χρησιμοποιήσαμε για τη συγκεκριμένη διπλωματική γράφτηκε από τον Vaggeli Vassalo και βρίσκεται σε αυτή την ιστοσελίδα: MatMult_VHDL/vhdl/matmult/matmult_core.vhd at master · vangvassalos/MatMult_VHDL (github.com)

Χρησιμοποιήσαμε τον συγκεκριμένο κώδικα γιατί είναι ένας αρκετά χρήσιμος κώδικας για να καταλάβουμε πως γίνεται ο πολλαπλασιασμός πινάκων, καθώς και η πρόσθεση στοιχείων, και με βάση τις βιβλιοθήκες που έχουμε επιλέξει να βρούμε τη συχνότητα του σήματος που αντιστοιχεί στην αντίστοιχη τεχνολογία. Έτσι, μέσω του κώδικα και της συχνότητας που θα βρούμε θα μπορέσουμε να ορίσουμε την καθυστέρηση που αναλογεί σε κάθε τεχνολογία και να κάνουμε ένα πιο τίμιο συσχετισμό με την θεωρητική μελέτη.

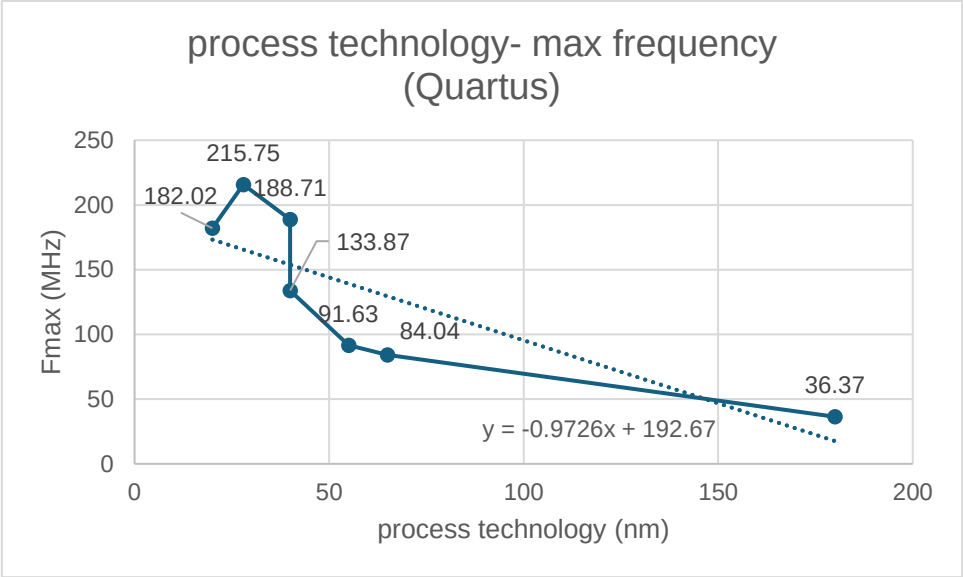
Είναι σημαντικό να αναφερθεί πως όσο το Process Technology είναι μικρό, τόσο είναι και ο χρόνος καθυστέρησης που πρέπει να διανύσει το σήμα και έτσι υπάρχει πιο μικρή καθυστέρηση. Συνεπώς, επιτυγχάνεται μεγαλύτερη συχνότητα καθώς η συχνότητα με τον χρόνο που διανύει ένα σήμα είναι αντιστρόφως ανάλογα πόσα και αυτό φαίνεται από την παρακάτω σχέση :

$$Period = \frac{1}{Frequency} \; or \; T = \frac{1}{f} \; (6)$$

Οι βιβλιοθήκες που χρησιμοποιήσαμε, καθώς και τα αποτελέσματα που λάβαμε από τον κώδικα αναγράφονται στον παρακάτω πίνακα :

Πίνακας 9

Όνομα βιβλιοθήκης	Process Technology	Fmax (MHz)
arria 10	20	182.02
starix V	28	215.75
starix IV	40	188.71
Arria II GX	40	133.87
max 10	55	91.63
cyclone IV	65	84.04
max V	180	36.37



Σχεδιάγραμμα 12 : Process Technology(nm) – Max Frequency Quartus

Με βάση τη συγκεκριμένη γραφική παράσταση, αλλά και την εξίσωση που αναγράφεται στην γραφική παράσταση, παρατηρούμε οι τιμές αλλάζουν γραμμικά. Επίσης, η κλίση της εξίσωσης είναι αρνητική , καθώς η τεχνολογία κυμαίνεται από την πιο πρόσφατη (20nm) στην πιο παλιά (180nm), και τέλος παρατηρείται ότι όσο μεγαλώνει το Process Technology (nm) τόσο μικραίνει και η Max Frequency.

Επίσης, η γραμμική συνάρτηση που αναγράφεται σε αυτή τη γραφική παράσταση είναι η

y= -0.9726x + 192.67 (7)

Στη συνέχεια, με βάση το επιστημονικό άρθρο του Aaron Stillmaker "VLSI-Scaling-Stillmaker.pdf (ucdavis.edu) " και τον πίνακα 2 που περιέχεται μέσα σε αυτό γίνεται η αναγωγή της καθυστέρησης (Delay) στην ίδια τεχνολογία .

Πίνακας 10

Technology Node (nm)	Delay(ps)
180	77.2
65	19.8
45	10.9
32	9.8
20	9.66
16	6.12

Εκτελώντας ακριβώς την ίδια διαδικασία παίρνουμε τη θεωρητική εξίσωση, η οποία είναι :

Y= -0.716x+130.049 (8)

Παρατηρείται ότι η κλίση μεταξύ των δύο γραφικών παραστάσεων είναι σχετικά κοντά, χωρίς να υπάρχει μεγάλη απόκλιση και αυτό δείχνει ότι υπάρχει τάση που όσο μικραίνει το τρανζίστορ τόσο ανεβαίνει η συχνότητα.

Για να κάνουμε τη σύγκριση που θέλουμε, θα χρησιμοποιήσουμε την εξίσωση $Y= ax + b$. Θα χρησιμοποιήσουμε αυτή την εξίσωση, γιατί παρατηρούμε ότι οι εξισώσεις που έχουμε λάβει και από τα δύο διαγράμματα είναι γραμμικές.

Θα χρησιμοποιήσουμε τις συχνότητες τις οποίες χρησιμοποιούν οι ερευνητές για κάθε μοντέλο FPGA. Σημαντικό είναι να αναφερθεί ότι επιλέξαμε την μέγιστη συχνότητα από κάθε τεχνολογία, καθώς η δική μας έρευνα γίνεται πάνω σε υψηλής απόδοσης μοντέλα. Στη συνέχεια, η μετατροπή θα γίνει στη τεχνολογία των 16nm, καθώς η τεχνολογία αυτή χρησιμοποιείται περισσότερο στα μοντέλα των FGPAs.

- Για την τεχνολογία των 28nm : 200MHz

  200= -0.716 x 28 +b => b = 220.048

  y' = -0.716 x 16 + 220.048 => y' = 208.60 MHz

- Για την τεχνολογία των 20nm : 200MHz

  200= -0.716 x 20 +b => b = 214.32

  y' = -0.716 x 16 + 214.32 => y' = 202.86 MHz

- Για την τεχνολογία των 14 nm : 275 MHz

  275 = -0.716 x 14 +b => b = 285.024

  y’ = -0.716 x 14 + 285.024 => y’ = 273.57 MHz

- Για την τεχνολογία των 7 nm : 1000MHz

  1000= -0.716 x 7 +b => b = 1005.012

  y’ = -0.716 x 16 + 1005.012 => y’ = 993.56 MHz

Στον παρακάτω πίνακα, αναγράφονται τα περισσότερα μοντέλα σε FPGAs, η συχνότητα με την οποία λειτουργούσαν και την αντίστοιχη τεχνολογία, αλλά και η καινούρια συχνότητα και τεχνολογία στην οποία βρίσκονται.

Πίνακας 11

FPGAs	Frequency (MHz)	Technology (nm)	New Frequency (MHz)	New Technology (nm)
NPE	200	28	208.60	16
Zhongyo Zhao	200	28	208.60	16
STA-8	200	20	202.86	16
STA-4	200	20	202.86	16
Georgios Tzanos	411	16	411	16
ViA	300	16	300	16
Me-Vit	300	16	300	16
DFX	200	16	200	16
HPTA	200	16	200	16
Swin-T	200	16	200	16
Swin-B	200	16	200	16
Swin-S	200	16	200	16
BETA	190	16	190	16
FlexRun	275	14	273.57	16
SSR	1000	7	993.56	16

Με βάση τον παραπάνω πίνακα μπορούμε να βρούμε κατά πόσο αυξήθηκε η επιτάχυνση όταν έγινε η μετατροπή της συχνότητας στα 16nm. Για να γίνει αυτό, θα αναφέρουμε μόνο τα επιστημονικά άρθρα στα οποία αναφέρεται η επιτάχυνση, τα οποία αναφέρονται στον πίνακα 8.

Σημείωση: για το μοντέλο του επιστημονικού άρθρου Zhongyo Zhao,όπως και το NPE, δεν μπορούμε να κάνουμε τη μετατροπή, καθώς δεν υπάρχει η τεχνολογία των 28nm στο επιστημονικό άρθρο του Aaron Stillmaker " VLSI-Scaling-Stillmaker.pdf (ucdavis.edu) ". Επίσης, δεν μπορούμε να κάνουμε τη μετατροπή και για τα μοντέλα των FlexRun και SSR, καθώς δεν αναφέρεται η απόδοση τους.

Για τα μοντέλα που χρησιμοποιούν τη τεχνολογία των 20nm και με βάση τον πίνακα 11:

202.86/200= **1.0143x αυξήθηκε.**

Τα μοντέλα που χρησιμοποιούν τις τεχνολογίες αυτή είναι :

- STA-4 : 1.0143 x GOPs (20nm) = 1.0143 x 392. 9 = 398.51 GOPs.
- STA-8 : 1.0143 x GOPs (20nm) = 1.0143 x 523.8 = 531.29 GOPs

Άρα, αν οι δύο υποκατηγορίες του μοντέλου STA λειτουργούσαν στη τεχνολογία των 16nm, θα είχαν καλύτερη απόδοση σε σύγκριση με τη τεχνολογία των 20nm.

8.2 Εφαρμογή Ollama

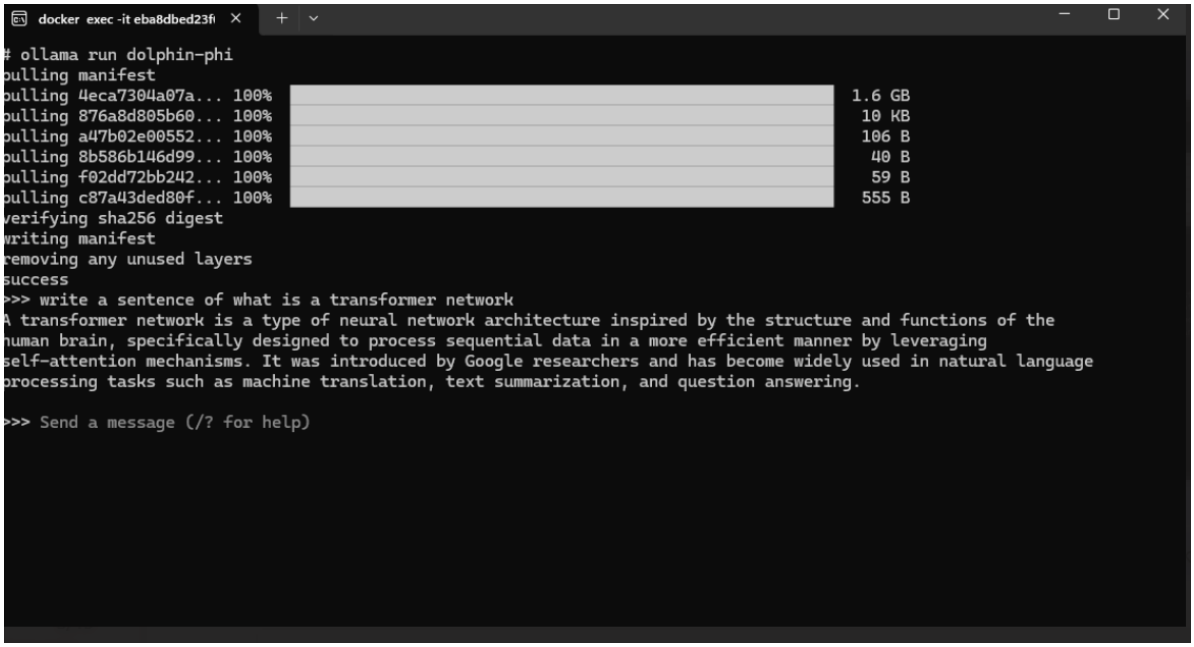
Η δεύτερη εφαρμογή, το Ollama, αποτελεί και αυτή ερευνητικό μέρος αυτής της διπλωματικής, και αφορά τους κλασικούς επεξεργαστές με χρήση των LLMs.

Για αρχή, βεβαιωθήκαμε ότι οι υπολογιστές χρησιμοποιούν το λογισμικό των Linux και στη συνέχεια κατεβάσαμε το μοντέλο Ollama με χρήση της εντολής : *curl http://ollama.com/install.sh | sh* χρησιμοποιώντας το command line των Linux.

Το Ollama είναι ένα εργαλείο το οποίο επιτρέπει στον χρήστη να τρέχει Μεγάλα Γλωσσικά Μοντέλα (LLM) ή μοντέλα στατικής γλώσσας (SLM) στον υπολογιστή μας. Είναι μια υπηρεσία REST API που λειτουργεί στον υπολογιστή μας. Αυτό είναι ένα από τα πιο σύνηθες μοντέλα που υπάρχουν γιατί περιέχει πολλές ενσωματώσεις. Επίσης, είναι απλό, ισχυρό και κατανοητό. Ακόμη, διαθέτει διάφορες υποκατηγορίες οι οποίες μπορούν να τρέξουν σε Linux, MacOS και Windows. Έχει βιβλιοθήκες για Nodejs και Python, οι οποίες είναι πολύ χρήσιμες. Επιπλέον, μπορεί να τρέξει και σε Docker και σε διάφορα Cluster. Τέλος, έχει βιβλιοθήκες μοντέλων που μπορούμε να κάνουμε pull και push για να πάρουμε διάφορες απαντήσεις, όπως κάναμε και εμείς. Με άλλα λόγια, έχει ένα αρκετά μεγάλο κατάλογο με ολοκληρώσεις από την κοινότητα της πληροφορικής.[59]

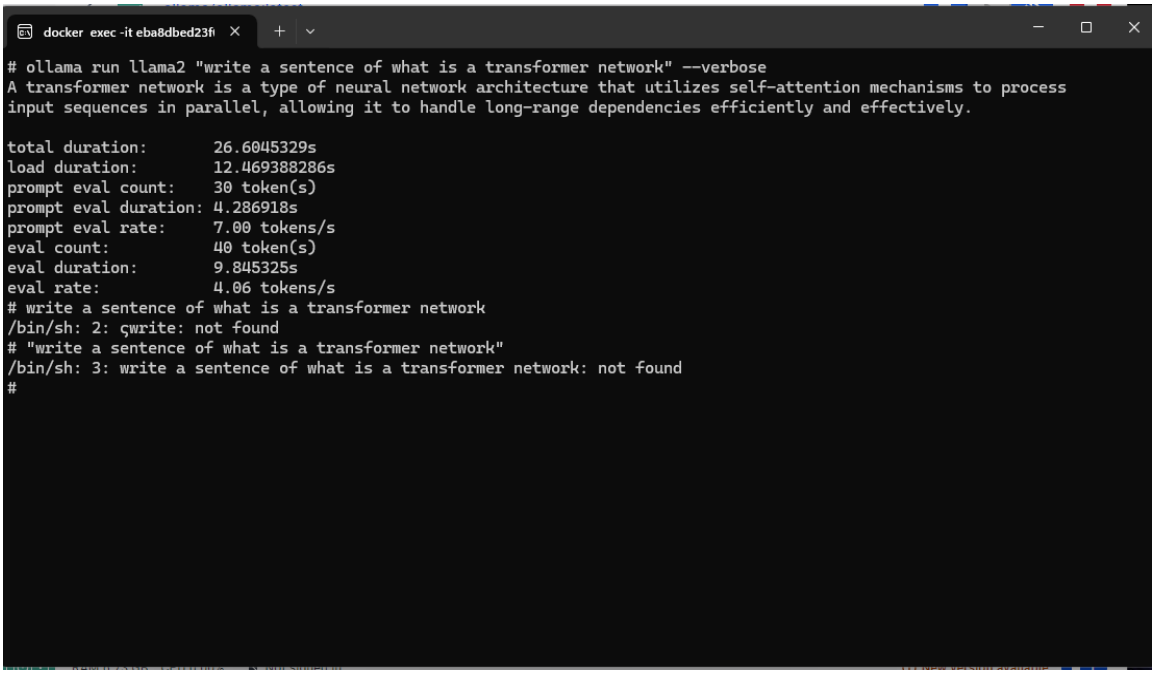
Αφού, λοιπόν, κατεβάσαμε το μοντέλο Ollama, τρέξαμε ένα από τα μοντέλα αρχιτεκτονικής που έχει ενσωματωμένα το "*dolphin-phi*". Όσο αφορά το μοντέλο "dolphin-phi" είναι ένα μοντέλο το οποίο περιέχει δύο βασικά στοιχεία, τα οποία είναι ο κωδικοποιητής και ο αποκωδικοποιητής. Όπως έχει αναφερθεί και παραπάνω, στον κωδικοποιητή εισάγεται μια ακολουθία και την παράγει σε ακολουθίες σταθερού μήκους, ενώ ο αποκωδικοποιητής δημιουργεί την ακολουθία εξόδου που είναι βασισμένη σε αναπαραστάσεις σταθερού μήκους. Για να εκπαιδευτεί το μοντέλο αυτό, χρησιμοποιείται το πλαίσιο του Ollama, το οποίο περιέχει προ εκπαιδευμένα μοντέλα. Έτσι, λοιπόν, μέσω του μοντέλου "dolphin-phi" τρέξαμε τον κώδικα σε εξωτερικό περιβάλλον ώστε να επιβεβαιωθούμε ότι τρέχει σωστά και του δώσαμε την παρακάτω εντολή:

"Ollama run dolphin-phi write a sentence (short) of what is a transformer network"



Εικόνα 12 : Κώδικας σε Linux με τη βιβλιοθήκη dolphin-phi

Γράψαμε την παρακάτω εντολή για να πάρουμε την αντίστοιχη απόδοση του υπολογιστή:



Εικόνα 13 : Εντολή verbose σε Linux

Η χρήση του docker προκαλεί καθυστέρηση στις μετρήσεις, καθώς είναι μια μέθοδος container. Για αυτό τον λόγο θα το τρέξουμε και χωρίς το πρόγραμμα docker ώστε να έχουμε καλύτερα αποτελέσματα. Επίσης, παρακάτω θα δούμε τι εκφράζουν οι εντολές prompt evaluate count, prompt evaluate duration, prompt evaluate rate, evaluate count, evaluate duration και evaluate rate, ώστε να γίνει πιο κατανοητό τι βλέπουμε στην οθόνη μας. Έτσι, :

1. **Prompt evaluate count**. Αναφέρεται στους αριθμούς που χρειάζεται μια εντολή για να αξιολογηθεί σε ένα συγκεκριμένο πλαίσιο ή περιβάλλον.

2. **Prompt evaluate duration**. Αναφέρεται στην αναπαράσταση της εντολής στον χρόνο που απαιτείται για την επεξεργασία μιας εντολής, συνήθως μετρώντας από την στιγμή που εισάγεται η εντολή μέχρι την ολοκλήρωση της εκτέλεσής της. Με άλλα λόγια αποτελεί τη διάρκεια από την εισαγωγή μιας εντολής εώς την έξοδο της.

3. **Prompt evaluate rate**. Δείχνει πόσο γρήγορα εκτελούνται οι εντολές μεταξύ τους και πόσο γρήγορα δημιουργούνται οι απαντήσεις.

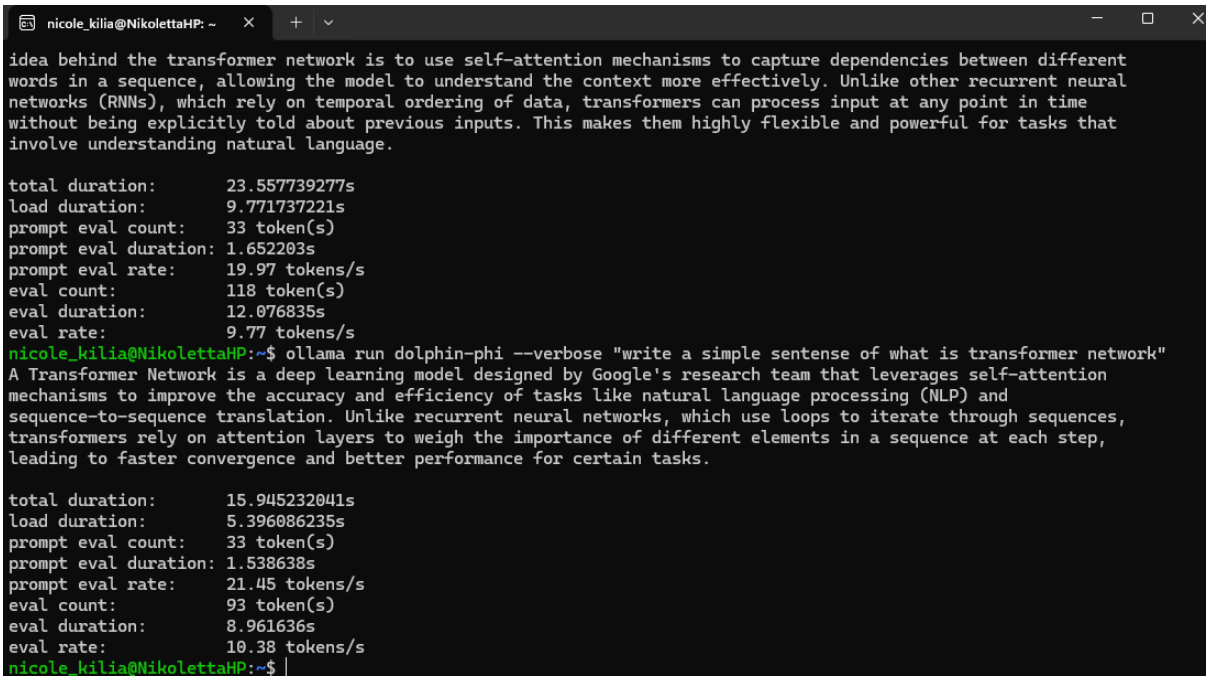
4. **Evaluate rate**. Αναφέρεται στο συνολικό αριθμό εκτελέσεων μιας διεργασίας, ενός script, μέσα σε ένα περιβάλλον.

5. **Evaluate duration**. Αναφέρεται στο χρονικό διάστημα που απαιτείται για να αξιολογηθεί μια εντολή.

6. **Evaluate rate**. Είναι ο ρυθμός με τον οποίο πραγματοποιούνται οι εντολές, δηλαδή το πόσο γρήγορα τρέχουν και μετριούνται σε μονάδες χρόνου. Γενικά, παρέχει την αποτελεσματικότητα και την απόδοση των εντολών μέσα σε ένα σύστημα.

Στη συγκεκριμένη διπλωματική εργασία ασχοληθήκαμε με το *evaluate rate*, καθώς θέλουμε να μετρήσουμε και να συγκρίνουμε την επιτάχυνση του υπολογιστή δίνοντας του μια συγκεκριμένη πρόταση.

Επαναλαμβάνουμε, λοιπόν, την διαδικασία χωρίς το πρόγραμμα docker. Ανοίγουμε το λογισμικό WSL που είναι ένα Virtual Machine για Linux και αφού κατεβάσουμε μια πολύ δημοφιλής έκδοση των Linux το Ubuntu και γράφουμε την εντολή: *$ curl -fsSL https://ollama.com/install.sh | sh*. Αυτή η εντολή μας κατεβάζει το Ollama στο Ubuntu. Αφού κατέβει επιλέγουμε ένα μοντέλο από την βιβλιοθήκη που έχει μικρό μέθεδος, ώστε να κατέβει γρήγορα. Αυτό το μοντέλο είναι το Dolphin-phi που έχουμε αναφέρει παραπάνω. Για να το κατέβασουμε, γράφουμε στο command line : *$ ollama run dolphin-phi.* Αφού το κατεβάσουμε και γράφουμε επίσης στο command line *: $ollama run dolphin-phi –verbose "write a short sentence of what is a transformer network".* Γράφοντας , λοιπόν, με αυτή την πρόταση τρέχουμε το μοντέλο Dolphin-phi μέσω του Ollama, το ρωτάμε μια συγκεκριμένη ερώτηση, η οποία είναι «γράψε μου μια σύντομη πρόταση για το τι ορίζουμε δίκτυο επιτάχυνσης». Η εντολή *verbose* μας βοηθάει να μετρήσουμε την επιτάχυνση του υπολογιστή για να γράψει την πρόταση που του δώσαμε. Οπότε τρέχοντας όλα τα παραπάνω περιμένουμε να δούμε στην οθόνη μας το παρακάτω μήνυμα :



Εικόνα 14 : Μέτρηση ρυθμού επεξεργασίας του υπολογιστή σε Linux

Το τρέχουμε στον υπολογιστή μας 6 φορές καθώς, η πρώτη μέτρηση δε μετράει. Αυτό συμβαίνει γιατί τη πρώτη φορά που τρέχουμε το μοντέλο, ο υπολογιστής φορτώνει με δεδομένα και μετράει άλλες φορές περισσότερη επιτάχυνση και άλλες φορές λιγότερη (cold start). Έτσι, αφού το τρέξουμε 6 φορές παίρνουμε τις αντίστοιχες τιμές του evaluate rate, γιατί αυτή η τιμή είναι η αντίστοιχη της επιτάχυνσης στα Linux, και βρίσκουμε την μέση επιτάχυνση. Έτσι, οι μετρήσεις που πήραμε από διάφορους υπολογιστές φαίνονται στον παρακάτω πίνακα :

Πίνακας 12

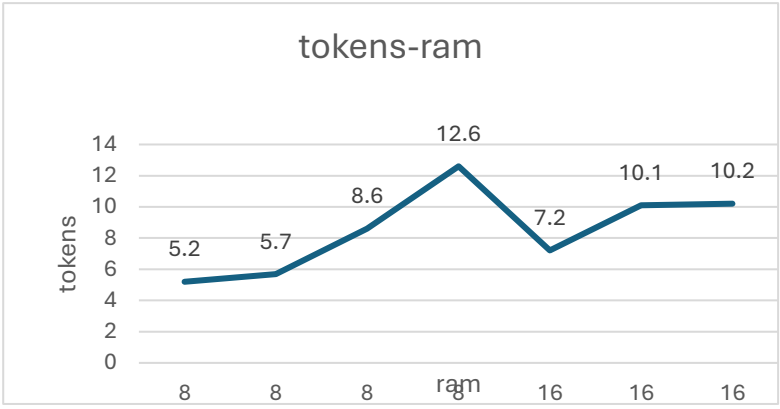
Name	RAM	Cores	Threads	Process Tech	Frequency (GHz)	1st	2nd	3rd	4th	5th	6th	Avg
ARM Ryzen 5 3500U	8.0	4	8	12	2.1	5.0	5.9	5.9	6.1	5.6	-	5.7
i5-10505	8.0	4	12	14	3.2	8.1	7.9	8.6	8.8	8.9	8.8	8.6
Intel i7-1065G7	8.0	4	8	10	1.3	13.1	13.6	11.1	12.9	12.4	13.2	12.6
Intel i5 - 1035G1	8.0	4	8	10	1.0	5.2	5.1	5.2	5.1	5.2	5.3	5.2
13th Gen Intel(R) i7-1355U	16.0	10	12	10	1.7	10.4	10.4	9.4	10.6	9.4	11.0	10.2
11th gen intel i5-1135G7	16.0	4	8	10	2.4	11.0	10.7	10.0	9.6	10.4	9.9	10.1
Intel i7-8550U	16.0	4	8	14	1.8	7.7	6.9	7.5	6.8	7.1	-	7.2

Αφού τρέξαμε σε διάφορους υπολογιστές το μοντέλο μας, σχηματίσαμε τα παρακάτω διαγράμματα τα οποία σχετίζονται με τις πληροφορίες που λάβαμε. Σταθερός παράγοντας είναι το *evaluate rate*, δηλαδή η επιτάχυνση του υπολογιστή και αλλάζουμε τους παράγοντες, μνήμη RAM, Cores, Threads, Συχνότητα, Process Technology

- Το πρώτο διάγραμμα **tokens – RAM**

Πίνακας 13

RAM	Tokens avg
8	5.2
8	5.7
8	8.6
8	12.6
16	7.2
16	10.1
16	10.2

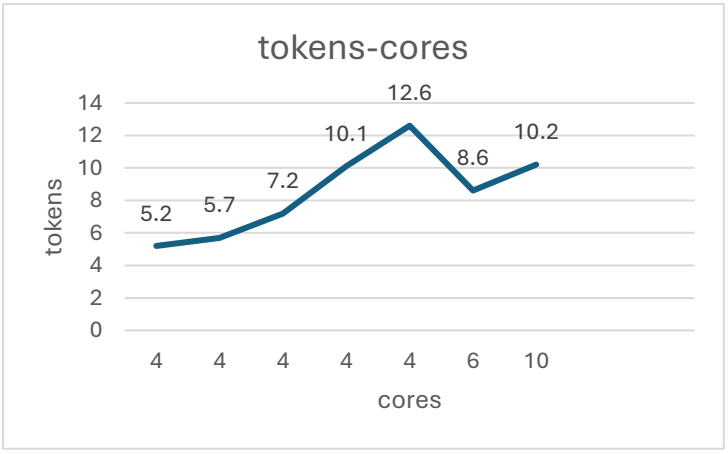


Σχεδιάγραμμα 13 : RAM - tokens

- Το δεύτερο διάγραμμα **tokens – cores**

Πίνακας 14

Cores	Tokens avg
4	5.2
4	5.7
4	7.2
4	10.1
4	12.6
6	8.6
10	10.2

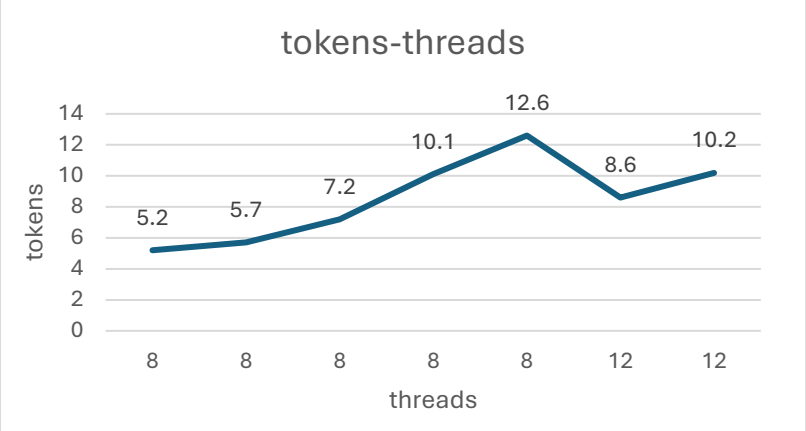


Σχεδιάγραμμα 14 : Cores – tokens

- Το τρίτο διάγραμμα **tokens- Threads**

Πίνακας 15

Threads	Tokens avg
8	5.2
8	5.7
8	7.2
8	10.1
8	12.6
12	8.6
12	10.2

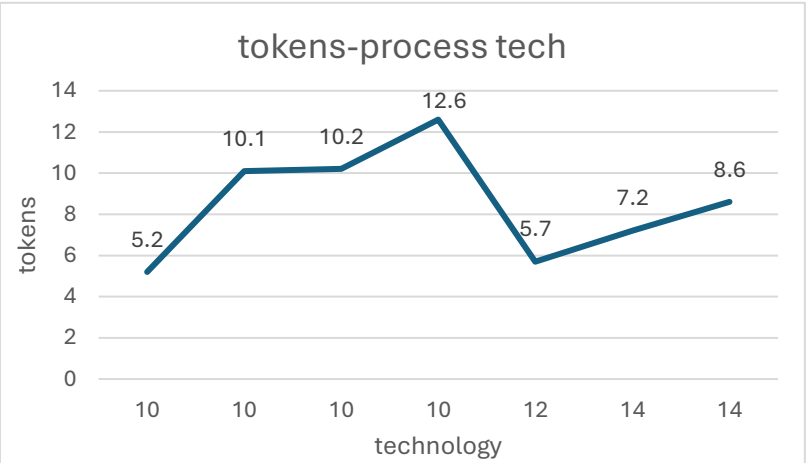


Σχεδιάγραμμα 15 : Threads - tokens

- Το τέταρτο διάγραμμα **tokens- process technology**

Πίνακας 16

Process Tech	Tokens avg
10	5.2
10	10.1
10	10.2
10	12.6
12	5.7
14	7.2
14	8.6

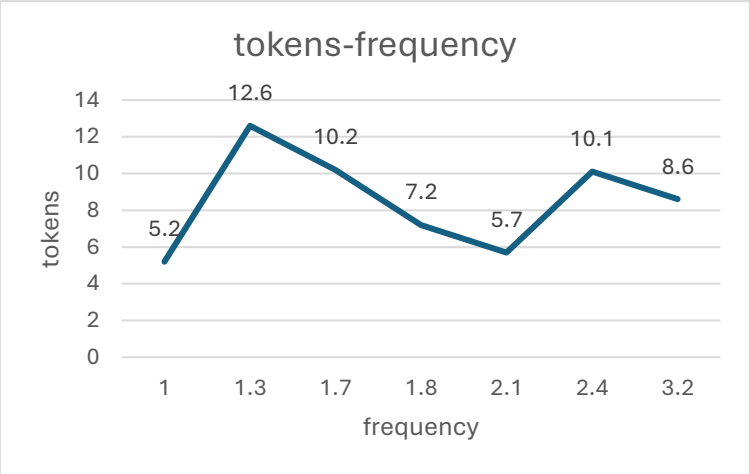


Σχεδιάγραμμα 16 : process technology – tokes

- Το τελευταίο διάγραμμα αφορά **tokens – frequency**

Πίνακας 17

frequency	Tokens avg
1	5.2
1.3	12.6
1.7	10.2
1.8	7.2
2.1	5.7
2.4	10.1
3.2	8.6



Σχεδιάγραμμα 17 : Frequency – tokens

Παρατηρούμε ότι και όλες τις περιπτώσεις τα διαγράμματα που σχηματίστηκαν δεν είναι γραμμικές, άρα δεν μπορούμε να κάνουμε σύγκριση και συσχέτιση στις τεχνολογίες που έχουμε επιλέξει. Παρόλα αυτά, παρατηρούμε επίσης, ότι η απόδοση του υπολογιστή επηρεάζεται πιο πολύ αναλόγως τα cores και τα threads που έχει κάποιος υπολογιστής, παρά από την συχνότητα, την τεχνολογία στην οποία βρίσκεται και στην μνήμη που χρησιμοποιεί.

8.3 Ανάλυση παραμέτρων βελτιστοποίησης

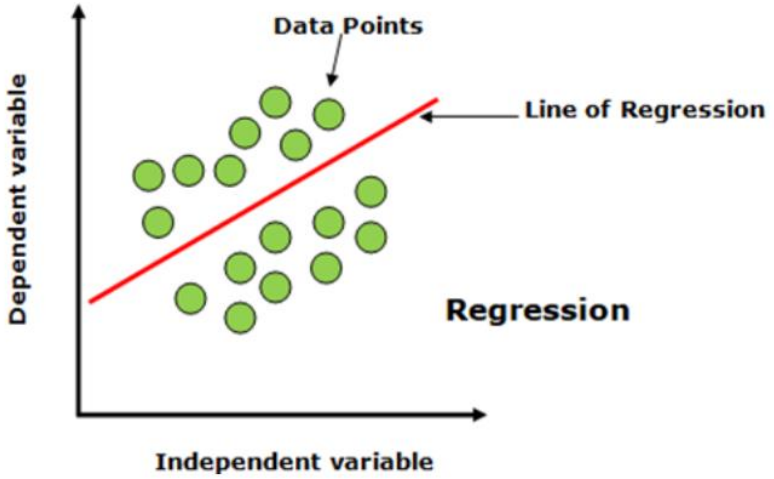
Στο σημείο αυτό αναλύουμε τις παραμέτρους που έχουν πιο καθοριστικό ρόλο στην απόδοση των εφαρμογών LLM όταν εκτελούνται σε τυπικούς επεξεργαστές (CPU-GPU). Για την ανάλυση των παραμέτρων αυτών χρησιμοποιήσαμε τις εξής τέσσερις μεθόδους οι οποίες αφορούν την παλινδρόμηση (Regression).

Επιλέξαμε να χρησιμοποιήσουμε την παλινδρόμηση και τους διάφορους τύπους της, γιατί η παλινδρόμηση είναι μια μέθοδος που μπορούμε να προβλέψουμε τιμές και να βελτιστοποιήσουμε τιμές.

Έτσι, η ανάλυση έγινε με βάση τις τέσσερις μεθόδους της, πρώτα με χρήση της γραμμικής παλινδρόμησης, έπειτα με χρήση του δένδρου αποφάσεων, μετά με την μέθοδο των τυχαίων δένδρων αποφάσεων και τέλος με SVM.

8.3.1 Ανάλυση με χρήση γραμμικής παλινδρόμησης (Linear Regression)

Δοκιμάσαμε την μέθοδο από το Machine Learning που ονομάζεται παλινδρόμηση (Regression) και συγκεκριμένα τη γραμμική παλινδρόμηση (Linear Regression). Αυτή η μέθοδος αναφέρεται σε ένα τύπο αλγορίθμου επίβλεψης όπου ο στόχος είναι να προβλέψει μια συνεχή τιμή βασισμένη σε χαρακτηριστικά εισόδου. Δηλαδή, χρησιμοποιείται όταν η μεταβλητή είναι συνεχής και έχει ένα εύρος πιθανών τιμών. Στόχος της εφαρμογής αυτής, είναι γενικά να βρεθούν ποιοι παράγοντες επηρεάζουν την επιτάχυνση μέσα από τα δεδομένα που θα του δώσουμε.



Εικόνα 15 : Γραμμική Παλινδρόμηση

Η βασική ιδέα είναι να βρεθεί η σχέση μεταξύ των ανεξάρτητων μεταβλητών, οι οποίες είναι αυτές που εισάγουμε εμείς, τα χαρακτηριστικά τους, την εξαρτώμενη μεταβλητή και τον στόχο που ορίζουμε εμείς. Στη συνέχεια, να χρησιμοποιηθεί αυτή η σχέση που θα βρεθεί για να προβλέψει δεδομένα. Στη δική μας περίπτωση, χρησιμοποιήσαμε τη γραμμική παλινδρόμηση, η οποία είναι ένας αλγόριθμος που δείχνει τη γραμμική σχέση μιας εξαρτημένης μεταβλητής με μια ανεξάρτητη μεταβλητή για να προβλέψει τα αποτελέσματα μελλοντικών γεγονότων.

Συνοπτικά, λοιπόν, η παλινδρόμηση στη Μηχανική Μάθηση είναι ένα αρκετά ισχυρό εργαλείο για την πρόβλεψη συνεχών αποτελεσμάτων με βάση τα χαρακτηριστικά εισόδου και βρίσκει εφαρμογές σε διάφορους τομείς, όπως οι οικονομικές επιστήμες, στο κλάδο της υγείας και σε πολλά άλλα. Πως χτίσαμε τον κώδικα και για να φτιάξουμε το δικό μας μοντέλο ακολουθήσαμε τα παρακάτω βήματα :

1. **Συλλογή δεδομένων**. Συγκετρώσαμε τα δεδομένα τα οποία θέλουμε να αναλύσουμε, τα οποία φαίνονται αναλυτικά στον πίνακα 12. Βεβαιωθήκαμε ότι έχουμε τις τιμές που θέλουμε και επιλέξαμε ποιες θα είναι οι εξαρτημένες και ποιες οι ανεξάρτητες.

2. **Χωρισμός δεδομένων**. Διαιρέσαμε το σύνολο των δεδομένων σε δύο υποσύνολα. Το ένα αφορά το σύνολο εκπαίδευσης και το άλλο το σύνολο της δοκιμής. Συνήθως, χρησιμοποιείται ένα μεγάλο μέρος των δεδομένων για εκπαίδευση, για παράδειγμα το 80%, και το υπόλοιπο για δοκιμή. Έτσι, το εκτελέσαμε και εμείς.

3. **Επιλογή χαρακτηριστικών.** Έαν κριθεί απαραίτητο, μπορούμε να επεξεργαστούμε τα δεδομένα ώστε να δημιουργήσουμε νέα δεδομένα που μπορεί να βοηθήσει στην απόδοση του μοντέλου μας. Για παράδειγμα, μπορεί να περιλαμβάνει τεχνικές οι οποίες θα βοηθήσουν στην κλιμάκωση, κανονικοποίηση ή την κωδικοποιήση των μεταβλητών.

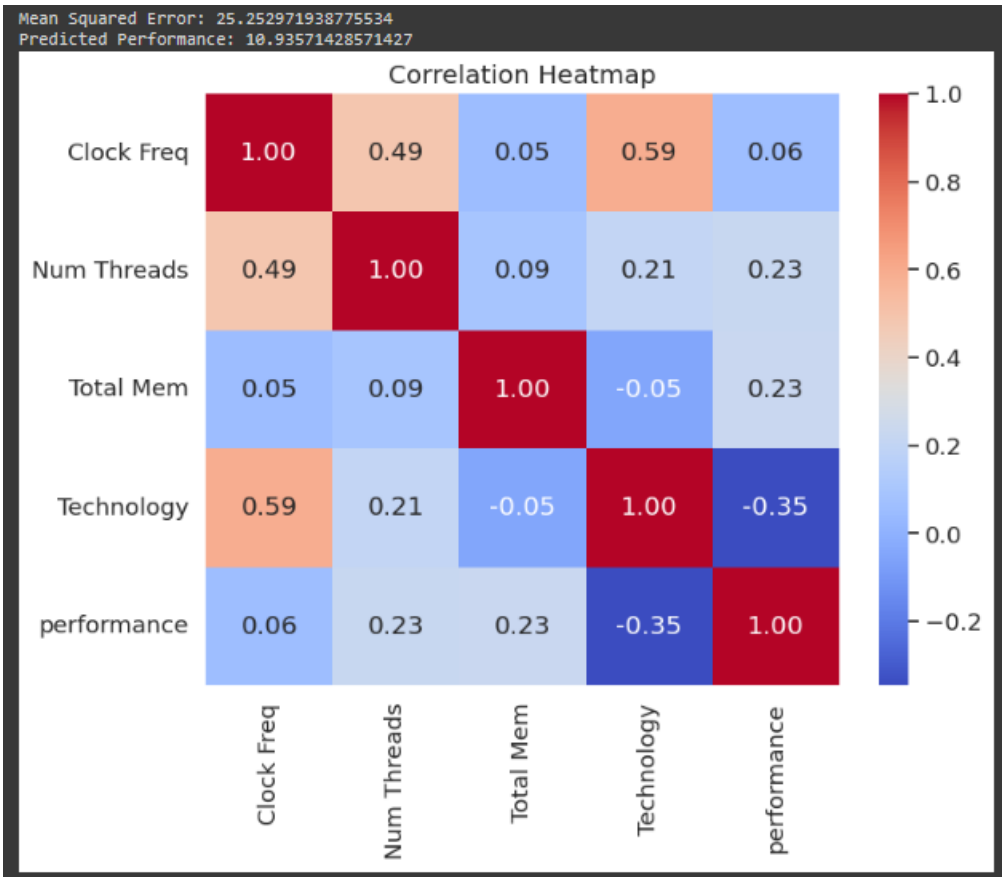
4. **Επιλογή μοντέλου**. Επιλέγουμε τι μοντέλο θέλουμε να χρησιμοποιήσουμε. Εμείς, όπως έχει αναφερθεί παραπάνω, επιλέξαμε το μοντέλο της γραμμικής παλινδρόμησης.

5. **Εκπαίδευση μοντέλου.** Χρησιμποιούμε τα δεδομένα εκπαίδευσης για να προσαρμόσουμε το μοντέλο που έχουμε επιλέξει (γραμμική παλινδρόμηση). Αυτό περιλαμβάνει την εκτίμηση των βαρών της γραμμική εξίσωσης που ταιριάζει καλύτερα στη σχέση μεταξύ των μεταβλητών.

6. **Αξιολόγηση μοντέλου**. Αφού έχει εκπαιδευτεί το μοντέλο και έχουμε αφήσει περίπου το 20% των δεδομένων για δοκιμή, αξιολογούμε την απόδοση του μοντέλου με τα δεδομένα αυτά. Συνήθως, οι αξιολογήσεις περιλαμβάνουν το μέσο τετραγωνικό σφάλμα (MSE), το ριζωμένο μέσο τετραγωνικό σφάλμα (RMSE) και το μέσο απόλυτο σφάλμα (MAE). Μέσα από αυτό θα βρούμε πόσο αποκλείνουν τα μοντέλα μεταξύ τους.

Για ευκολία χρησιμοποιήσαμε κώδικα σε γλώσσα προγραμματισμού Python. Επιλέξαμε να το εκτελέσουμε σε λογισμικό των Windows με χρήση του προγράμματος Colab. Το Colab είναι ένα χρήσιμο εργαλείο της Google που παρέχει Jupyter Notebook και δεν απαιτεί καμία ρύθμιση για να χρησιμοποιηθεί. Είναι κατάλληλο για Μηχανική Μάθηση. Έτσι, τρέξαμε στο Colab τον κώδικα , σε γλώσσα Python , με βάση τα χαρακτηριστικά των υπολογιστών που είχαμε χρησιμοποιήσει και στην προηγούμενη εφαρμογή (πίνακας 12) . Έτσι έχουμε :

Με βάση τον κώδικα παίρνουμε το παρακάτω σχεδιάγραμμα :



Σχεδιάγραμμα 18 : Linear Regression

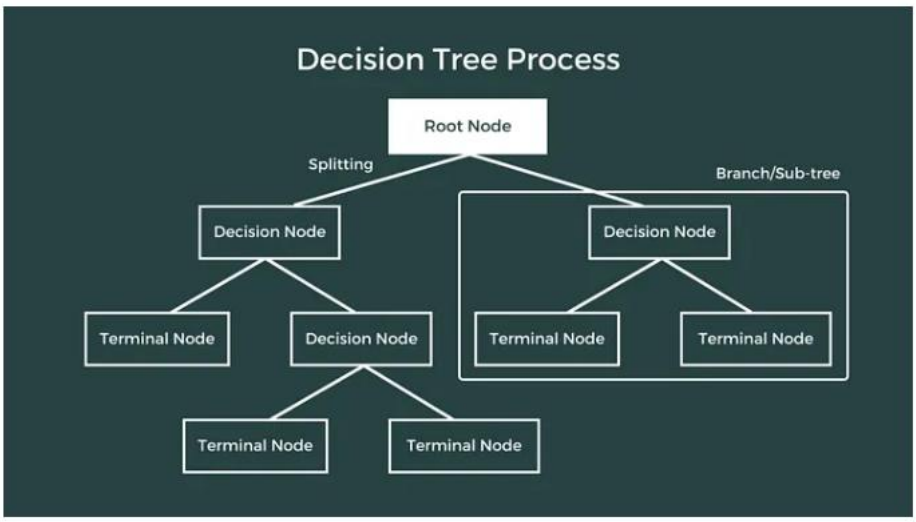
Για να δούμε ποιοι παράγοντες επηρεάζουν την απόδοση,τελικά, πρέπει να παρατηρήσουμε τη τελευταία στήλη που είναι κάθετα και διαβάζοντας τις τιμές του Heatmap παρατηρείται ότι σημαντικό ρόλο έχουν τόσο τα Threads όσο και η μνήμη RAM που έχει ο κάθε υπολογιστής. Όσο αφορά την συχνότητα και την τεχνολογία φαίνεται χαρακτηριστικά στο σχεδιάγραμμα (Heatmap) ότι η απόδοση επηρεάζεται ελάχιστα από τη συχνότητα και καθόλου από την τεχνολογία που έχει ο κάθε επεξεργαστής. Επίσης, το σφάλμα για την πρόβλεψη που κάναμε είναι σχετικά μικρό, αυτό σημαίνει ότι όσο πιο μικρό σφάλμα τόσο πιο ακριβείς είναι οι προβλέψεις που κάνει το συγκεκριμένο μοντέλο.

8.3.2 Ανάλυση με χρήση Δένδρου Απόφασης (Decision Tree Regressor)

Η χρήση του δένδρου απόφασης είναι ένα από τους πιο συχνούς αλγορίθμους που χρησιμοποιούνται στην μηχανική μάθηση για την επίλυση προβλημάτων παλινδρόμησης και ταξινόμησης. Ο αλγόριθμος χρησιμοποιεί ένα μοντέλο αποφάσεων που μοιάζει με ένα δένδρο για να προβλέψει είτε την τιμή του στόχου (παλινδρόμηση) είτε την κλάση στόχου (ταξινόμηση).

Η διαδικασία ξεκινάει με χρήση διαίρεσης από το ριζικό κόμβο και ακολουθείται από δένδρο που τελικά οδηγείται σε ένα φύλλο κόμβου (τερματικό κόμβος) που περιέχει την πρόβλεψη ή το τελικό αποτέλεσμα του αλγορίθμου. Η κατασκευή των δένδρων γίνεται ως εξής. Πρώτα επιλέγονται

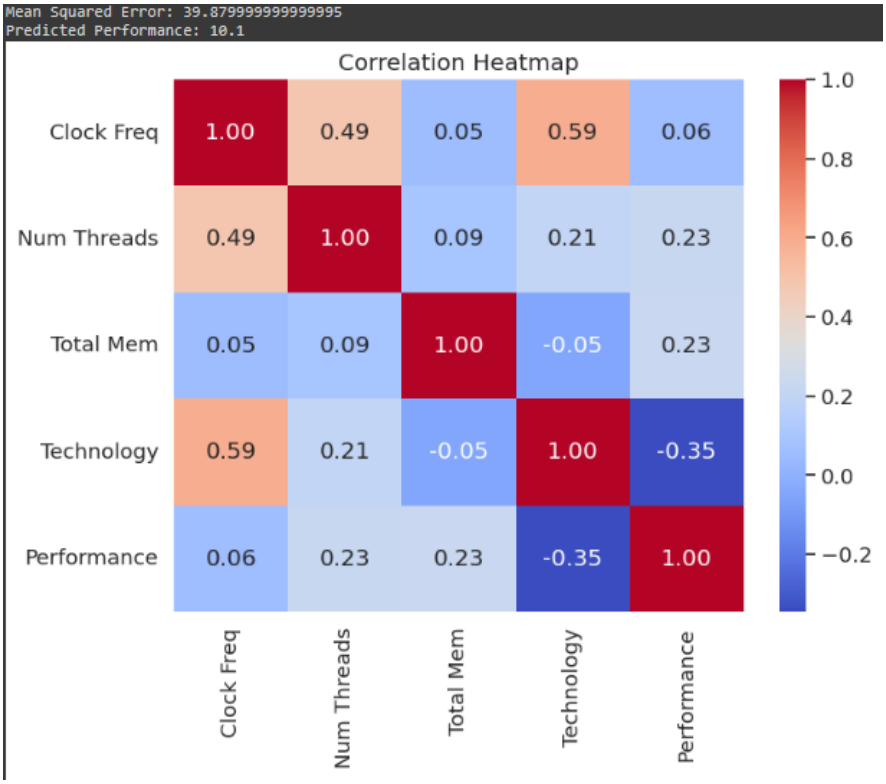
σε κάθε βήμα μια μεταβλητή που διαιρεί καλύτερα το σύνολο των στοιχείων. Κάθε υπό-δένδρο που δημιουργείται μπορεί να αναπαρασταθεί ως δυαδικό δένδρο όπου ένας κόμβος απόφασης διαιρείται σε δύο κόμβους με βάση τις συνθήκες.



Εικόνα 16 : Decision Tree

Πηγή: https://medium.com/@theclickreader/decision-tree-regression-explained-with-implementation-in-python-1e6e48aa7a47

Αλλάζοντας έτσι, τον κώδικα που αναπτύχθηκε στο κεφάλαιο 4.2.1, και την σειρά "*model = LinearRegression()* σε "*model = DecisionTreeRegressor(random_state=100)*" και παίρνουμε το παρακάτω σχεδιάγραμμα :

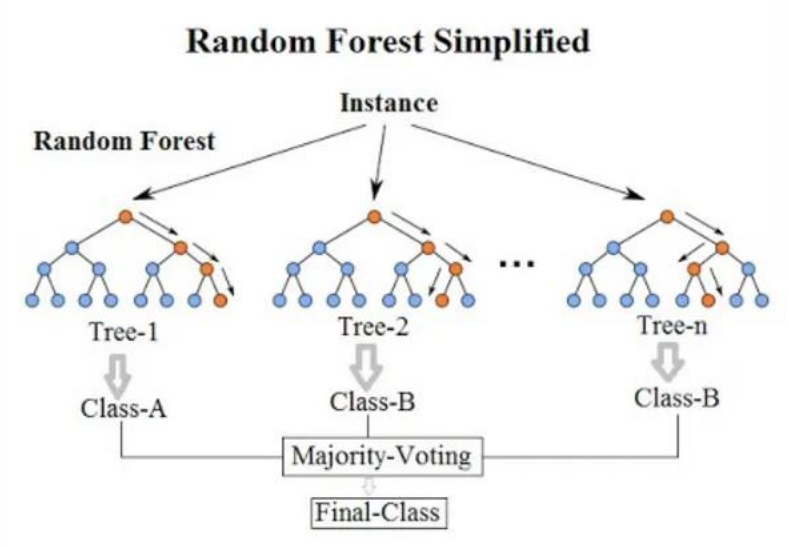


Σχεδιάγραμμα 19 : το decision tree

Παρατηρούμε ότι σε αυτή τη περίπτωση έχουμε μεγαλύτερο σφάλμα σε σύγκριση με την προηγούμενη μελέτη, καθώς και ελάχιστη διαφορά στην επιτάχυνση με την πρώτη περίπτωση να είναι καλύτερη. Όσο αφορά το σχεδιάγραμμα, για να είναι πιο γρήγορος ο υπολογιστής που θέλουμε σημαντικό ρόλο έχουν τα threads και η μνήμη που χρησιμοποιεί ο υπολογιστής.

8.3.3 Ανάλυση με χρήση Τυχαίου Δάσους (Random Forest)

Τα τυχαία δάση ή τυχαία δένδρα αποφάσεως είναι μια ομάδα συνεργασίας από τα δένδρα απόφασης που λειτουργούν μαζί για να παρέχουν μόνο μια έξοδο. Ο αλγόριθμος αυτός είναι μια ισχυρή μάθηση δένδρων στην μηχανική μάθηση. Έχει ως λειτουργία έναν αριθμό δένδρων κατά την διάρκεια της εκπαίδευσης. Κάθε δένδρο κατασκευάζεται χρησιμοποιώντας ένα τυχαίο υποσύνολο του συνόλου δεδομένων για να μετρήσει ένα τυχαίο υποσύνολο χαρακτηριστικών. Ο αλγόριθμος συγκεντρώνει τα αποτελέσματα όλων των δένδρων, είτε για εργασίες ταξινόμησης είτε με το μέσο όρο της παλινδρόμησης. Αυτή η διαδικασία, παρέχει σταθερά και ακριβή αποτελέσματα.

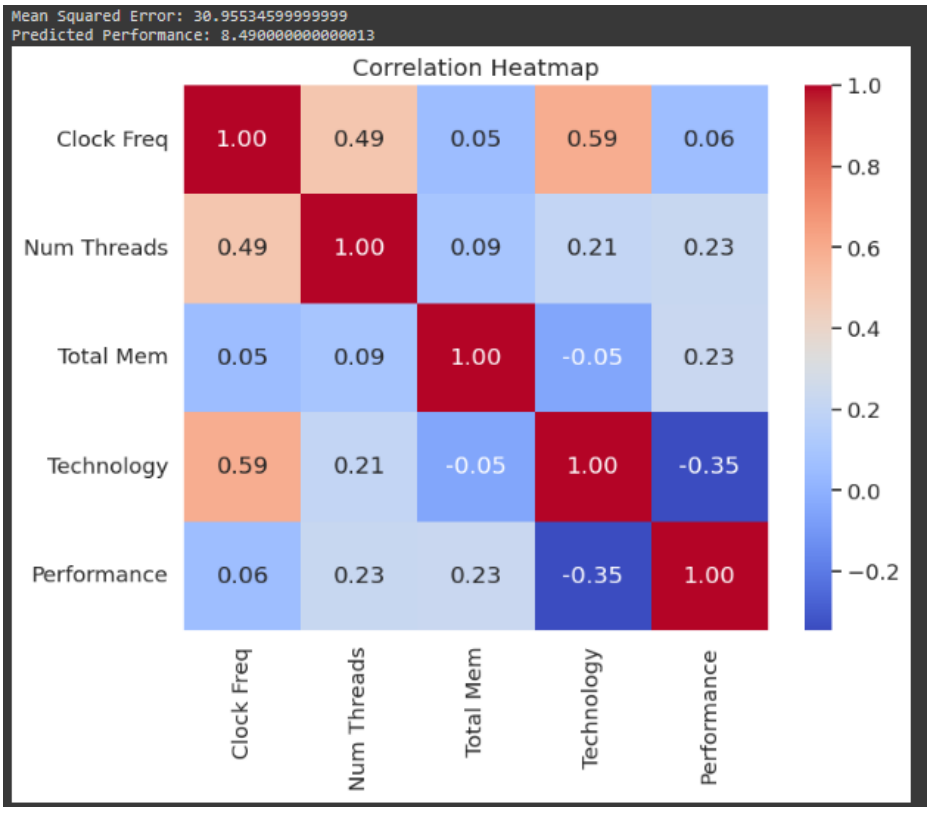


Εικόνα 17 : τυχαία δένδρα αποφάσεως

Πηγή : https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

Αλλάξαμε τον κώδικα που είχαμε στην ενότητα 4.2.1 στη σειρά " *model = LinearRegression()* " και το αντικαταστήσαμε με το " model = RandomForestRegressor(random_state=100)" .

Το διάγραμμα που παίρνουμε από τον κώδικα είναι :

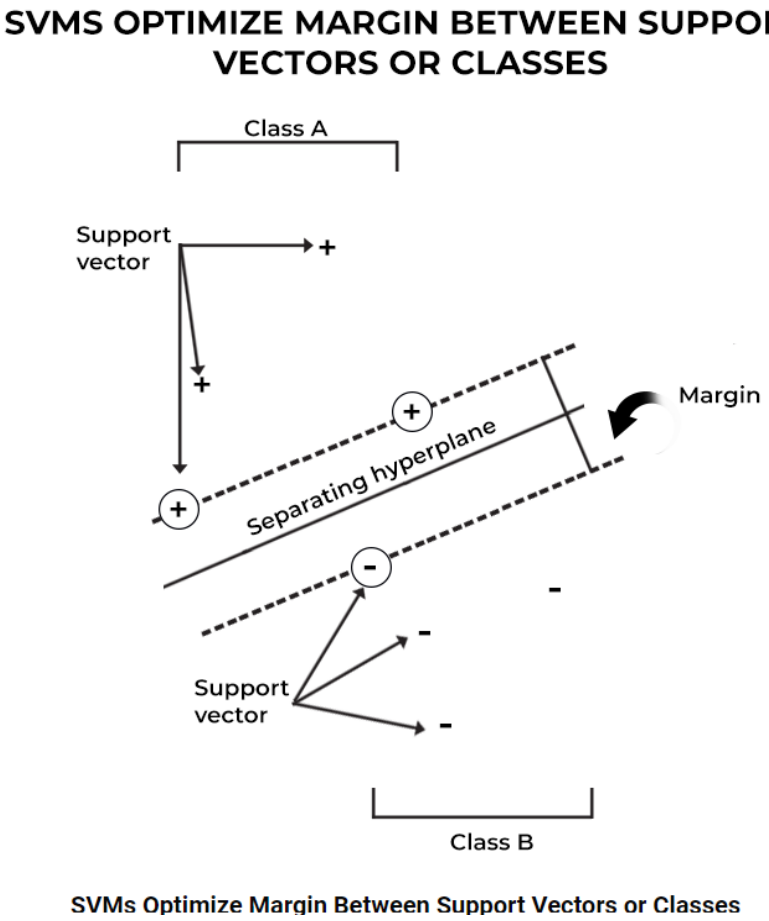


Σχεδιάγραμμα 20: Random Forest

Παρατηρούμε ότι έχουμε μικρότερο σφάλμα σε σύγκριση με την διαδικασία του δένδρου αποφάσεως αλλά μεγαλύτερο σφάλμα σε σύγκριση με την γραμμική παλινδρόμηση. Επίσης, παρατηρείται ότι δεν υπάρχει καλύτερη επιτάχυνση σε σύγκριση με την γραμμική παλινδρόμηση και το μέσο τετραγωνικό σφάλμα να είναι αρκετά μεγαλύτερο σε σύγκριση με την μέθοδο του δένδρου αποφάσεων και τη γραμμική παλινδρόμηση.

8.3.4 Ανάλυση με χρήση Support Vector Machines

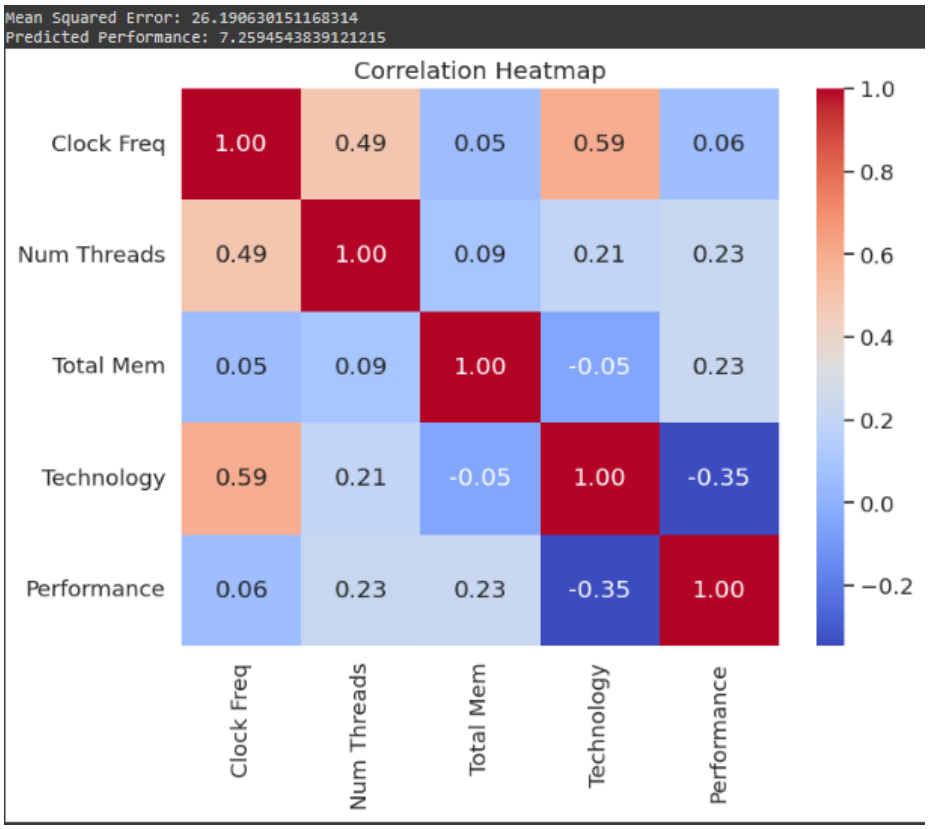
Το SVM είναι ένας αλγόριθμος της μηχανικής μάθησης που χρησιμοποιεί τα επιβλέποντα μοντέλα για την επίλυση πολύπλοκων προβλημάτων κατηγοριοποίησης, παλινδρόμησης και ανίχνευσης ακραίων τιμών. Γι' αυτό τον λόγο, αποτελεί και μια εναλλακτική εφαρμογή του SoftMax στην κατηγοριοποίηση. Για να αναπτυχθεί ο αλγόριθμος αυτός χρειάστηκαν κρυφές αναπαραστάσεις σε ένα νευρωνικό δίκτυο που εκπαιδεύεται με στόχους. Οι μεταβλητές αυτές χρησιμοποιούνται ως είσοδοι στο SVM. Η μέθοδος αυτή έχει σκοπό να βελτιώνει συχνά τον ρυθμό απόδοσης.



Εικόνα 18 : SVM

Πηγή : https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/

Έτσι, αλλάξαμε τον κώδικα που είχαμε στην ενότητα 4.2.1. και τη γραμμή *" model = LinearRegression()"* σε *" model = SVR(kernel='linear') "* και πήραμε το παρακάτω σχεδιάγραμμα:



Σχεδιάγραμμα 21 : SVM

Παρατηρούμε το σχεδιάγραμμα, και μπορούμε να καταλήξουμε στο συμπέρασμα ότι και πάλι σημαντικό ρόλο στην επιτάχυνση έχει το Number of Threads και η μνήμη RAM. Ακόμη, σημαντικό είναι να αναφερθεί ότι η πιθανή επιτάχυνση είναι μικρότερη σε σύγκριση με τα προηγούμενα μοντέλα (linear regression, decision tree και Random Forest), καθώς και έχει παραπλήσιο τετραγωνικό σφάλμα με την πρώτη περίπτωση.

ΚΕΦΑΛΑΙΟ 9ο: ΣΥΜΠΕΡΑΣΜΑΤΑ

Όσο αφορά τη βιβλιογραφική μελέτη, παρατηρούμε ότι τα περισσότερα μοντέλα χρησιμοποιούν τη μέθοδο του πολλαπλασιασμού ως πράξη πινάκων. Όπως αναφέρθηκε παραπάνω, είναι η πιο σύνηθες πράξη των πινάκων, που μπορεί να προσφέρει στο κάθε μοντέλο μικρότερη καθυστέρηση και μεγαλύτερη ακρίβεια.

Τα FPGAs, μπορούν να έχουν χαμηλότερη καθυστέρηση σε εφαρμογές όπως αυτές που χρησιμοποιούν βίντεο, live streaming κλπ. σε σύγκριση με αυτά που χρησιμοποιούν CPU-GPU. Τα μοντέλα των FPGA είναι από τα μοντέλα που έχουν υψηλή απόδοση και διαπρέπουν σε εφαρμογές που απαιτούν παράλληλη επεξεργασία δεδομένων σε πραγματικό χρόνο, αλλά δεν μπορούν να ξεπεράσουν την απόδοση των ASIC. Όσο αφορά την ενεργειακή απόδοση τους είναι καλύτερη από τους γενικούς επεξεργαστές CPU που έχουν ορισμένα καθήκοντα.

Παρατηρούμε, επίσης, ότι οι επιταχυντές που είναι βασισμένοι σε Asics έχουν μικρότερη κατανάλωση ενέργειας σε σύγκριση με τα άλλα τρία μοντέλα επιταχυντών. Αυτό συμβαίνει γιατί η αρχιτεκτονική που χρησιμοποιούν μπορεί να ρυθμίσει κατάλληλα το hardware, ώστε να μειώσει την ενέργεια αυτή. Επίσης, χρησιμοποιούν πολλαπλές λειτουργίες και έτσι προσφέρουν περισσότερη ενεργειακή απόδοση από ότι ένα τσιπ.

Ακόμη, παρατηρείται ότι οι επιταχυντές που χρησιμοποιούν την μνήμη (In- Memory) έχουν καλύτερη και μεγαλύτερη ενεργειακή απόδοση σε σύγκριση με τα άλλες τρεις κατηγορίες επιταχυντών, γιατί μειώνεται η μεταφορά των δεδομένων μεταξύ της μνήμης και των μονάδων επεξεργασίας και έτσι παράγεται περισσότερη ενέργεια. Επίσης, σημαντικό ρόλο έχει και η συχνότητα, καθώς τα μοντέλα αυτά, In-Memory, λειτουργούν σε χαμηλές συχνότητες αποθηκεύοντας περισσότερη ενέργεια όταν υπάρχει δυναμική κατανάλωση ενέργειας.

Για τη σύγκριση μεταξύ των μοντέλων επιτάχυνσης που υπάρχουν τα τελευταία χρόνια και τις τέσσερις διαφορετικές κατηγορίες, δε μπορούμε να κάνουμε κάποια σημαντική σύγκριση και παρατήρηση. Αυτό συμβαίνει γιατί τα μοντέλα δεν χρησιμοποιούν ούτε την ίδια τεχνολογία αλλά ούτε και τον ίδιο επεξεργαστή. Για αυτό τον λόγο γίνεται η σύγκριση της βιβλιογραφικής μελέτης (των τεσσάρων κατηγοριών) στη τεχνολογία των 7nm, ώστε να μπορέσουμε να λάβουμε καλύτερα συμπεράσματα. Με βάση τη θεωρητική μελέτη και τη σύγκριση των 7nm παρατηρούμε όλα τα μοντέλα έχουν καλύτερη απόδοση σε αυτή την τεχνολογία. Αντίθετα, όσο αφορά την ενεργειακή τους απόδοση, έχουν χαμηλότερη κατανάλωση ενέργειας σε σύγκριση με την τεχνολογία στην οποία βρισκόντουσαν, καθώς όσο βελτιώνεται και εξελίσσεται η τεχνολογία, βελτιώνεται και η ενεργειακή απόδοση.

Για το δεύτερο μέρος της διπλωματικής αυτής το οποίο χωρίζεται σε τρία διαφορετικά μέρη τα οποία είναι : η χρήση κώδικα σε VHDL και η σύγκριση της συχνότητας με τη καθυστέρηση για τα μοντέλα που χρησιμοποιούν FGPAs (εργαστηριακή αναγωγή), την χρήση του Ollama και την πιθανή απόδοση ενός υπολογιστή με δεδομένα που εισάγει ο χρήστης (ανάλυση παραμέτρων βελτιστοποίησης).

Για τη πρώτη εφαρμογή, η οποία εκτελείται με κώδικα VHDL για τα μοντέλα των FPGAs έχοντας που έχουν ως κύρια πράξη πινάκων τον πολλαπλασιασμό, παρατηρούμε ότι οι τιμές αλλάζουν γραμμικά, και αυτό φαίνεται από την εξίσωση που αναγράφεται δίπλα στη γραφική παράσταση. Επίσης, είναι σημαντικό να τονιστεί ότι όσο μεγαλώνει το process technology τόσο μικραίνει η συχνότητα. Είναι λογικό το αποτέλεσμα αυτό, καθώς με βάση την σχέση της συχνότητας με τη καθυστέρηση, όσο αυξάνεται για παράδειγμα η συχνότητα τόσο μικραίνει η καθυστέρηση και όσο μικραίνει η καθυστέρηση αυξάνεται η απόδοση αφού είναι αντιστρόφως ανάλογα ποσά. Επιπλέον, παρατηρείται το ίδιο και στη θεωρητική μελέτη που έλαβε χώρα σε αυτή τη διπλωματική.

Μετατρέποντας όλα τα μοντέλα που έχουν ρυθμό επιτάχυνσης στη τεχνολογία των 16nm παρατηρείται ότι όλα τα μοντέλα έχουν καλύτερη απόδοση σε σύγκριση με την απόδοση στη τεχνολογία που βρισκόντουσαν. Χαρακτηριστικό παράδειγμα είναι το μοντέλο του STA, με τις δύο υποκατηγορίες STA-4 και STA-8, που στην τεχνολογία των 16nm η επιτάχυνση τους έχει αυξηθεί κατά 1.0143 φορές. Αυτό συμβαίνει γιατί, όσο μικραίνει το Process Technology επιτρέπεται η κατασκευή πιο ισχυρών και αποδοτικών FPGAs που μπορούν να εκτελέσουν υπολογιστικές λειτουργίες, όπως ο πολλαπλασιασμός πινάκων, έχοντας πολύ καλύτερη απόδοση.

Όσο αφορά τη δεύτερη εφαρμογή δίνεται μια πιθανή απάντηση στην ερώτηση που έγινε και το χρονικό πλαίσιο που χρειάστηκε για να βρεθεί και να γραφτεί η απάντηση. Συγκεκριμένα, αυτή η ιδιότητα είναι ένα χαρακτηριστικό παράδειγμα της λειτουργίας των LLM. Έτσι, λαμβάνοντας υπόψιν τις μετρήσεις (σ.σ. τιμές απόδοσης), άλλα και τα χαρακτηριστικά που είχε κάθε υπολογιστής, υπολογιστές που είχαμε στην διάθεση μας, παρατηρούμε ότι σημαντικό ρόλο στον ρυθμό επιτάχυνσης έχει η μνήμη RAM, αλλά και τα Threads που χρησιμοποιεί κάθε υπολογιστής. Αυτό φαίνεται και στα διαγράμματα που έχουμε παραθέσει παραπάνω. Παρόλα αυτά, τα διαγράμματα δεν είναι γραμμικά, δε μπορούμε να κάνουμε κάποια σημαντική σύγκριση, όπως η μετατροπή τους στη τεχνολογία των 7nm.

Για αυτό τον λόγο κάναμε την ανάλυση παραμέτρων, η οποία είναι μια εφαρμογή των LLMs σε λογισμικό Windows, ώστε να κάνουν πρόβλεψη του ρυθμού επιτάχυνσης, αναλύοντας τις παραμέτρους από την προηγούμενη εφαρμογή, εισάγοντας τα υπάρχοντα χαρακτηριστικά των υπολογιστών καθώς και χαρακτηριστικά υπολογιστών που θα θέλαμε να έχει ο δικό μας υπολογιστής.

Με βάση τον κώδικα και τα διαγράμματα που φαίνονται παραπάνω, είναι καλύτερο της γραμμικής παλινδρόμησης και μετά ο αλγόριθμος του SVM, γιατί όσο αφορά το δεύτερο υπάρχει μικρότερο σφάλμα σε σύγκριση με όλες τις μεθόδους.

Παρατηρείται ότι η περίπτωση του αλγορίθμου SVM είναι αρκετά κοντά στην πρώτη περίπτωση της γραμμικής παλινδρόμησης. Αυτό συμβαίνει γιατί και τα δύο μοντέλα περιλαμβάνουν την προσαρμογή ενός γραμμικού μοντέλου στα δεδομένα. Η μόνη διαφορά τους είναι ότι μπορεί να χρησιμοποιηθεί για προβλήματα ταξινόμησης όσο και για παλινδρόμηση, ενώ η γραμμική παλινδρόμηση σχεδιάστηκε ειδικά για προβλήματα που αφορούν την παλινδρόμηση.

Επιπλέον, το SVM έχει ώς στόχο να μεγιστοποιήσει το περιθώριο μεταξύ κλάσεων ή δεδομένων, ενώ η γραμμική παλινδρόμηση ελαχιστοποιεί το τετραγωνικό σφάλμα μεταξύ πραγματικών και προβλεπόμενων τιμών. Έτσι, είναι λογικό να έχει μικρότερο το τετραγωνικό σφάλμα στην πρώτη περίπτωση σε σύγκριση με όλες τις άλλες περιπτώσεις.

Συμπερασματικά, το linear regression χρησιμοποιείται όταν υπάρχει γραμμική σχέση μεταξύ των τιμών εισόδου και των τιμών του στόχου. Είναι πιο γρήγορο και πιο απλή μέθοδος αλλά υπάρχουν περιορισμοί. Σε αντίθεση, η μέθοδος SVM είναι μια πιο ικανή μέθοδος που μπορεί να χειριστεί μη γραμμικές τιμές, παρόλο που είναι υπολογιστικά ευαίσθητη, προσφέρει μεγαλύτερη ευελιξία σε μοντέλα που έχουν πολύπλοκα δεδομένα.

Τέλος, μπορούμε να καταλήξουμε στο συμπέρασμα ότι σημαντικό ρόλο για τους κλασικούς επιταχυντές έχουν τα Threads και η μνήμη RAM, ενώ για τους ψηφιακούς επιταχυντές έχει η συχνότητα.

Βιβλιογραφία – Αναφορές - Διαδικτυακές Πηγές

1. *Bingbing Li, Santosh Pandey and Haowen Fang, "FTRANS: energy-efficient acceleration of transformers using FPGA", 2020*
2. *Jinming Zhuang, Zhuoping Yang, "SSR: Spatial Sequential Hybrid Architecture for Latency Throughput Tradeoff in Transformer Acceleration", 2024*
3. *Siyuan Lu, "Hardware Accelerator for Multi-Head Attention and Position-Wise Feed-Forward in the Transformer", 2020*
4. *Teng Wang, Lei Gong and Chao Wang, "ViA: A Novel Vision-Transformer Accelerator Based on FPGA", 2022*
5. *Hamza Khan, Asma Khan, Zainab Khan "NPE: An FPGA-based Overlay Processor for Natural Language Processing", 2021*
6. *Panjie Qi, Yuhong Song, Hongwu Peng, "Acoomodating Transformer onto FPGA : Coupling the Balanced Model Compression and FPGA- Implementation Optimization", 2021*
7. *Hongwu Peng "Accelerating Transformer-based Deep Learning Models on FPGA using Column Balanced Block Pruning", 2021*
8. *Seongmin Hong, Seungjae Moon, " DFX: A Low-latecy Multi- FPGA Appliance for Accelerating Transformer-based Text Generation", 2022*
9. *Chao Fang, Shouliang Guo, "An Efficient Hardware Accelerator for Sparse Transformer Neural Networks", 2022*
10. *Yuntao Han, Qiang Liu, "HPTA: A High Performance Transformer Accelerator Based on FPGA", 2023*
11. *Suyeon Hur, Seongmin Na, "A Fast and Flexible FPGA-based Accelerator for Neural Language Processing Neural Network", 2023*
12. *Zhiyang Liu, Pengyu Yin, Zhenhua Ren, "An Efficient FPGA-Based Accelerator for Swin Transformer", 2023*
13. *Zhongyo Zhao, "An FPGA-Based Transformer Accelerator Using Output Block Stationary Dataflow for Object Recognition Applications",2023*

14. *Yuhao Ji, Chao Fang, Zhongfeng Wang, "Beta : Binarized Energy-Efficient Transformer Accelerator at the Edge", 2024*

15. *Kyle Marino, Pengmiao Zhang, Viktor Prasanna "ME-ViT: A Single-Load Memory-Efficient FPGA Accelerator for Vision Transformers", 2024*

16. *Dimitrios Danopoulos, Georgios Zervakis, Dimitrios Soudris, Jörg Henkel, "TransAxx: Efficient Transformers with Approximate Computing", 2024*

17. *Ikumi Okubo, "A Cost-Efficient FPGA Implementation of Tiny Transformer Model using Neural ODE", 2024*

18. *Georgios Tzanos, "Hardware Acceleration of Transformer Network using FPGAs",2021*

19. *Jiarui Fang, Yang Yu, Chengduo Zhao,"TurboTransformers: An Efficient GPU Serving System For Transformer Models", 2021*

20. *Jaewan Choi, "Accelerating Transformer Networks through Recomposing Softmax Layers",2022*

21. *Xiaohui Wang, Yang Wei, Ying Xiong, Guyue Huang, Xian Qian, Yufei Ding, Mingxuan Wang, Lei Li,"LightSeq2: Accelerated Training for Transformer-based Models on GPUs", 2022*

22. *Nan Yang, "Inference with Reference: Lossless Acceleration of Large Language Models", 2023*

23. *Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Re, Ion Stoica, Ce Zhang "FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GP", 2023*

24. *Woosuk Kwon, Zhuohan Li, "Efficient Memory Management for Large Language Model Serving with PagedAttention", 2023*

25. *Youpeng Zhao, Di Wu, Jun Wang "ALISA: Accelerating Large Language Model Inference via Sparsity-Aware KV Caching", 2024*

26. *Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H. Oh "a3: Accelerating Attention Mechanisms in Neural Networks with Approximation", 2020*

27. *Liqiang Lu , Yicheng Jin ,"Sanger: A Co-Design Framework for Enabling Sparse Attention using Reconfigurable Architecture",2021*

28. *Hanrui Wang "SpAtten: Efficient Sparse Attention Architecture with Cascade Token and Head Pruning",2021*

29. *Tae Jun Ham, Yejin Lee, Seong Hoon Seo "ELSA: Hardware-Software Co-design for Efficient, Lightweight Self-Attention Mechanism in Neural Network", 2021*

30. *Guan Shen, Jieru Zhao , Quan Chen, "SALO: An Efficient Spatial Accelerator Enabling Hybrid Sparse Sparse Attention Mechanisms for Long Sequences", 2022*

31. *Shikhar Tuli, Niraj K. Jha, "AccelTran: A Sparsity-Aware Accelerator for Dynamic Inference With Transformers",2022*

32. *Tao Yang; Dongyue Li; Zhuoran Song; Yilong Zhao; Fangxin Liu; Zongwu Wang; Zhezhi He; Li Jiang "DTQAtten: Leveraging Dynamic Token-based Quantization for Efficient Attention Architecture",2022*

33. *Zhe Zhou, Junlin Liu, Zhenyu Gu, Guangyu Sun ,"Energon: Towards Efficient Acceleration of Transformers Using Dynamic Sparse Attention",2023*

34. *Yandong Luo, Shimeng Yu, "H3D-Transformer: A Heterogeneous 3D (H3D) Computing Platform for Transformer Model Acceleration on Edge Devices",2024*

35. *Jieru Zhao; Pai Zeng; Guan Shen; Quan Chen; Minyi Guo "Hardware-Software Co-Design Enabling Static and Dynamic Sparse Attention Mechanisms",2024*

36. *Haoqiang Guo , Lu Peng Jian Zhang "ATT: A Fault-Tolerant ReRAM Accelerator for Attention-based Neural Networks",2020*

37. *Xiaoxuan Yang , Bonan Yan , Hai Li "ReTransformer: ReRAM-based Processing-in-Memory Architecture for Transformer Acceleration", 2020*

38. Ann Franchesca Laguna; Arman Kazemi; Michael Niemier; X. Sharon Hu *"In-Memory Computing based Accelerator for Transformer Networks for Long Sequences",2022*

39. *Minxuan Zhou; Weihong Xu; Jaeyoung Kang; Tajana Rosing "TransPIM: A Memory-based Acceleration via Software-Hardware Co-Design for Transformer", 2022*

40. *Ann Franchesca Laguna, Mohammed Mehdi Sharifi, "Hardware-Software Co-Design of an In-Memory Transformer Network Accelerator", 2022*

41. *Shrihari Sridharan, Jacob R. Stevens, Kaushik Roy, Anand Raghunathan "X-Former: In-Memory Acceleration of Transformers", 2023*

42. *Yan Ding; Chubo Liu; Mingxing Duan; Wanli Chang; Keqin Li; Kenli Li "HAIMA: A Hybrid SRAM and DRAM Accelerator-in-Memory Architecture for Transformer",2023*

43. *Wantong Li, Madison Manley, James Read, Ankit Kaul,"H3DAtten: Heterogeneous 3-D Integrated Hybrid Analog and Digital Compute-in-Memory Accelerator for Vision Transformer Self-Attention",2023*

44. *Fengbin Tu; Zihan Wu; Yiqi Wang; Ling Liang; Liu Liu "TranCIM: Full-Digital Bitline-Transpose CIM-based Sparse Transformer Accelerator With Pipeline/Parallel Reconfigurable Modes",2023*

45. *Shiwei Liu; Chen Mu; Hao Jiang; Yunzhengmao Wang; Jinshan Zhang; "HARDSEA: Hybrid Analog-ReRAM Clustering and Digital-SRAM In-Memory Computing Accelerator for Dynamic Sparse Self-Attention in Transformer",2024*

46. *Yue Pan; Minxuan Zhou; Chonghan Lee; Zheyu Li; Rishika Kushwah; Vijaykrishnan Narayanan; Tajana Rosing "PRIMATE: Processing in Memory Acceleration for Dynamic Token-pruning Transformers",2024*

47. *https://www.coursera.org/learn/introduction-to-large-language-models/supplement/NLVj2/introduction-to-large-language-models-reading*

48. *https://developers.google.com/machine-learning/resources/intro-llms*

49. *https://medium.com/@theclickreader/decision-tree-regression-explained-with-implementation-in-python-1e6e48aa7a47*

50. *https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/*

51. *https://www.spiceworks.com/tech/big-data/articles/what-is-support-vector-machine/*

52. *https://www.coursera.org/learn/introduction-to-generative-ai/lecture/TJ28r/introduction-to-generative-ai*

53. *Yichuan Tang, "Deep Learning using Support Vector Machines"* *https://www.cs.toronto.edu/~tang/papers/dlsvm.pdf*

54. *https://www.intel.com/content/www/us/en/artificial-intelligence/programmable/fpga-gpu.html*

55. *https://www.geeksforgeeks.org/differentiate-between-support-vector-machine-and-logistic-regression/*

56. BERT Explained: State of the art language model for NLP | by Rani Horev | Towards Data Science

57. *Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu*
Richard Socher, Xavier Amatriain, Jianfeng Gao, « Large Language Models : A Survey »
[2402.06196] Large Language Models: A Survey (arxiv.org)

58. *Aaron Stillmaker and B.Baas titled "Scaling equations for the accurate prediction of CMOS device performance from 180 nm to 7nm,"vcl.ece.ucdavis.edu/pubs/2017.02.VLSIintegration.TechScale/VLSI-Scaling-Stillmaker.pdf*

59. *https://ollama.com/library/llama3*

Παράρτημα Α

Εδώ παρατίθενται οι κώδικες που χρησιμοποιήθηκαν για το ερευνητικό κομμάτι :

Ο κώδικας 1 σε γλώσσα προγραμματισμού VHDL :

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.all;
use work.matmult_pkg.all;

entity matmult_top_new is
   port (
      clk      : in std_logic;
      rst      : in std_logic;
      a        : in t_in_mat;
      b        : in t_in_mat;
      c        : out t_out_mat
      );
end matmult_top_new;

architecture behave of matmult_top_new is

  signal r_RESULT : t_out_mat := (others => (others => (others => '0')));
   signal r_A     : t_in_mat  := (others => (others => (others => '0')));
   signal r_B     : t_in_mat  := (others => (others => (others => '0')));



begin

  g_ROW_MUL : for i in 0 to t_in_mat'length-1 generate
      g_COL_MUL: for j in 0 to t_in_mat'length-1 generate
         ROW_COL_MULT : matmult_core
            port map (
                     i_CLK => clk,
                     i_RST => rst,
                     i_A  => r_A(i),
                     i_B  => r_B(j),
```

```
                    o_C  => r_RESULT(i)(j)
                  );
      end generate g_COL_MUL;
   end generate g_ROW_MUL;

  r_A <= a;
   r_B <= b;
   c <= r_RESULT;
end behave;
```

Κώδικας 2 : Γλώσσα προγραμματισμού Python.

```
 import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt


from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error


# Example data (replace with your actual data)

clock_freq = np.array([ 1.0, 1.3, 1.7, 1.8, 2.1, 2.4, 3.2])  # GHz

num_threads = np.array([8, 8, 12, 8, 8, 8, 12])

total_memory = np.array([8, 8, 16, 16, 8, 16, 8])  # GB

performance = np.array([5.2, 12.6, 10.2, 7.2, 5.7, 10.1, 8.6])  # Performance metric

technology = np.array ([10, 10, 10, 14, 12 ,10, 14]) #nm
```

```
# Combine features into a feature matrix

X = np.column_stack((clock_freq, num_threads, total_memory, technology))

X1 = np.column_stack((clock_freq, num_threads, total_memory, technology, performance))

y = performance


# Create a correlation matrix

corr_matrix = np.corrcoef(X.T)

corr_matrix1 = np.corrcoef (X1.T )


# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=100)


# Train the linear regression model

model = LinearRegression()

model.fit(X_train, y_train)


# Make predictions on the test set

y_pred = model.predict(X_test)


# Evaluate the model

mse = mean_squared_error(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
```

```
# Example prediction using new input values

new_input = np.array([[3.0, 8, 16, 10]])  # New data point (clock_freq=3.0GHz, threads=8, memory=16GB, technology= 10nm)

predicted_performance = model.predict(new_input)

print(f"Predicted Performance: {predicted_performance[0]}")


# Plot a heatmap of the correlation matrix

sns.set(font_scale=1.2)

plt.figure(figsize=(8, 6))

sns.heatmap(corr_matrix1, annot=True, fmt=".2f", cmap="coolwarm",


        xticklabels=["Clock Freq", "Num Threads", "Total Mem", "Technology", "performance"],

        yticklabels=["Clock Freq", "Num Threads", "Total Mem", "Technology", "performance"])

plt.title("Correlation Heatmap")

plt.show()
```