Πτυχιακή Εργασία

Πανεπιστήμιο Πειραιώς

Τμήμα Πληροφορικής

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

2022-2023

Κατέβας Χρήστος

Επιβλέπων καθηγητής: Ψαράκης Μιχαήλ

Κίνητρο διατριβής και ευχαριστίες

Το κίνητρο αυτής της διπλωματικής εργασίας είναι να διερευνήσει και να αναλύσει και να αξιολογήσει τις δυνατότητες του Coral Edge TPU για υπολογιστές γενικής χρήσης και να δώσει λύσεις σε επιβαλλόμενες προκλήσεις. Η συνεισφορά μας σε αυτή τη διατριβή περιλαμβάνει, αρχικά, μια εκτενή αξιολόγηση για το TPU, ανάλυση της αρχιτεκτονικής του, συμπεριλαμβανομένων τόσο προεκπαιδευμένων όσο και προσαρμοσμένων νευρονικών δικτύων.

Θα ήθελα να εκφράσω την ευγνωμοσύνη μου στον επιβλέποντα καθηγητή Ψαράκη Μιχαήλ, ο οποίος με εμπιστεύτηκε και μου έδωσε την ευκαιρία να εκπονήσω τη διπλωματική μου εργασία στη νέα αυτή τεχνολογία του Coral Edge TPU. Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για την υποστηρίξη τους καθ΄ όλη την διάρκεια της ακαδημαϊκής μου πορείας.

Περίληψη

Η Τεχνητή Νοημοσύνη είναι ένας από τους ταχύτερα αναπτυσσόμενους τομείς στην πληροφορική. Τα σύγχρονα νευρωνικά δίκτυα (ΝΔ) απαιτούν μεγάλη υπολογιστική πολυπλοκότητα και, ως αποτέλεσμα, οι σύγχρονοι επεξεργαστές (CPU) γενικού σκοπού δυσκολεύονται να επιτύχουν επαρκή απόδοση. Η Google ανέπτυξε τις Tensor Processing Units (TPUs) για να επιταχύνει την εκτέλεση εφαρμογών ΤΝ τόσο σε κέντρα δεδομένων όσο και σε λοιπές εφερμογές, εκπληροτώντας αυτό τον σκοπό. Σε αυτή τη διπλωματική εργασία, θα ασχοληθούμε με το Edge TPU. Το Edge TPU είναι ένα μικρό ολοκληρωμένο κύκλωμα που επιτρέπει την ανάπτυξη εφαρμογών ΤΝ at the edge. Το Edge TPU είναι ικανό να εκτελεί 4 τρισεκατομμύρια πράξεις ανά δευτερόλεπτο, χρησιμοποιώντας 2 Watt ισχύος. Ωστόσο, η αρχιτεκτονική και το σύνολο εντολών τέτοιων επιταχυντών ΤΝ, εμφανίζουν διάφορες προκλήσεις και περιορισμούς.

Πραγματοποιήσαμε benchmarking στο TPU, με έτοιμα μοντέλα που παρέχει η Google, με στόχο να αξιολογήσουμε τις δυνατότητές του. Τα αποτελέσματα που προέκυψαν αποκαλύπτουν σημαντική επιτάχυνση για το Google Edge TPU σε σύγκριση με τους επεξεργαστές BCM2837(Raspberry Pi3A+), AMD Ryzen 5 3500u. Συνολικά, επιτυγχάνει σημαντική επιτάχυνση μεγάλου μεγέθους συνελικτικών νευρωνικών δικτύων (ΣΝΔ) και απλών τεχνητών νευρωνικών δικτύων. Το Edge TPU παρέχει έως και 10 φορές καλύτερη απόδοση από τον BCM2837 και 7 φορές μεγαλύτερη απόδοση από τον AMD Ryzen 5 3500u.

Το Tensor Processing Unit (TPU) είναι ένα ολοκληρωμένο κύκλωμα για συγκεκριμένες εφαρμογές, λειτουργεί σαν επιταχυντής εφαρμογών τεχνητής νοημοσύνης, μηχανική εκμάθηση νευρωνικών δικτύων. Αναπτύχθηκε από την Google, χρησιμοποιώντας ως βάση τη βιβλιοθήκη TensorFlow. Η Google άρχισε να χρησιμοποιεί εσωτερικά TPU το 2015 και το 2018 τα έκανε διαθέσιμα για χρήση τρίτων, τόσο ως μέρος της υποδομής cloud της όσο και προσφέροντας μια μικρότερη έκδοση του τσιπ προς πώληση.

Σε σύγκριση με μια μονάδα επεξεργασίας γραφικών(GPU), έχει σχεδιαστεί για μεγάλο όγκο υπολογισμών χαμηλής ακρίβειας (π.χ. ακρίβεια 8-bit) με περισσότερες λειτουργίες εισόδου/εξόδου ανά τζάουλ. Τα TPU ASIC είναι τοποθετημένα σε ένα συγκρότημα ψύκτρας, το οποίο μπορεί να χωρέσει σε μια υποδοχή σκληρού δίσκου μέσα σε ένα rack κέντρου δεδομένων. Διαφορετικοί τύποι επεξεργαστών είναι κατάλληλοι για διαφορετικούς τύπους μοντέλων μηχανικής εκμάθησης, οι TPU είναι κατάλληλες για CNN ενώ οι GPU έχουν πλεονεκτήματα για ορισμένα πλήρως συνδεδεμένα νευρωνικά δίκτυα και οι CPU μπορούν να έχουν πλεονεκτήματα για τα RNN.

Abstract

Artificial Intelligence is one of the fastest growing fields in IT. Modern neural networks (NNs) require high computational complexity and, as a result, modern general-purpose CPUs struggle to achieve sufficient performance. Google developed Tensor Processing Units (TPUs) to accelerate the execution of IT applications in both data centers and other applications, fulfilling this purpose. In this thesis, we will deal with the Edge TPU. The Edge TPU is a small integrated circuit that enables the development of IT applications at the edge. The Edge TPU is capable of performing 4 trillion operations per second, using 2 Watts of power. However, the architecture and instruction set of such AI accelerators present various challenges and limitations.

We benchmarked the TPU, with ready-made models provided by Google, aiming to evaluate its capabilities. The obtained results reveal a significant speedup for Google Edge TPU compared to BCM2837 (Raspberry Pi3A+), AMD Ryzen 5 3500u processors. Overall, it achieves significant speedup of large-scale convolutional neural networks (CNNs) and simple artificial neural networks. Edge TPU provides up to 10x better performance than BCM2837 and 7x better performance than AMD Ryzen 5 3500u.

The Tensor Processing Unit (TPU) is an application-specific integrated circuit, it acts as an accelerator for artificial intelligence applications, machine learning neural networks. It was developed by Google, using the TensorFlow library as a base. Google began using TPUs internally in 2015, and in 2018 made them available for third-party use, both as part of its cloud infrastructure and by offering a smaller version of the chip for sale.

Compared to a graphics processing unit (GPU), it is designed for a high volume of low-precision calculations (e.g. 8-bit precision) with more I/O operations per joule. The TPU ASICs are mounted on a heat sink assembly, which can fit into a hard drive bay inside a data center rack. Different types of processors are suitable for different types of machine learning models, TPUs are suitable for CNNs while GPUs have advantages for some fully connected neural networks and CPUs can have advantages for RNNs.

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας διαγραμμάτων

Κεφάλαιο 1

1.1 Εισαγωγή

Η τεχνητή νοημοσύνη είναι ένας ευρύς κλάδος της επιστήμης των υπολογιστών που ασχολείται με την κατασκευή έξυπνων μηχανών ικανών να εκτελούν εργασίες που απαιτούν ανθρώπινη νοημοσύνη. Η κύρια εφαρμογή της τεχνητής νοημοσύνης είναι η μηχανική μάθηση. Ο στόχος της μηχανικής μάθησης είναι να επιτρέψει στις μηχανές να μαθαίνουν από μέσα από μεγάλο όγκο δεδομένων. Η μηχανική μάθηση συνθέτει και ερμηνεύει πληροφορίες για την ανθρώπινη κατανόηση, σύμφωνα με προκαθορισμένες παραμέτρους, συμβάλλοντας στην εξοικονόμηση χρόνου, στη μείωση των σφαλμάτων, στη δημιουργία προληπτικών ενεργειών και στην αυ- τοματοποίηση διαδικασιών. Για την επίτευξη της αυτοματοποίησης τα νευρωνικά δίκτυα παίζουν τον κύριο ρόλο. Τα νευρωνικά δίκτυα αντικατοπτρίζουν τη συμπεριφορά του ανθρώπινου εγκεφάλου, επιτρέποντας στους υπολογιστές να αναγνωρίζουν μοτίβα και να επιλύουν κοινά προβλήματα στους τομείς της τεχνητής νοημοσύνη.

Τα νευρωνικά δίκτυα στοχεύουν σε λειτουργικότητα που μοιάζει με εκείνη του ανθρώπινου εγκεφάλου και βασίζονται σε έναν απλό τεχνητό νευρώνα: μια μη γραμμική συνάρτηση ενός αθροίσματος εισόδων. Αυτοί οι τεχνητοί νευρώνες οργανώνονται σε στρώματα, με τις εξόδους ενός στρώματος να γίνονται οι είσοδοι του επόμενου στην ακολουθία. Οι δύο φάσεις των νευρωνικών δικτύων ονομάζονται εκπαίδευση και πρόβλεψη και αναφέρονται στην ανάπτυξη έναντι της παραγωγής. Ο προγραμματιστής επιλέγει τον αριθμό των στρωμάτων και τον τύπο του ΝΔ και η εκπαίδευση καθορίζει τα βάρη. Τρία είδη νευρωνικών δικτύων είναι δημοφιλή στις μέρες μας:

1. Πολυεπίπεδοι αισθητήρες : κάθε νέο στρώμα-επίπεδο είναι ένα σύνολο μη γραμμικών λειτουργιών σταθμισμένου αθροίσματος όλων των πλήρως συνδεδεμένων εξόδων, από ένα προηγούμενο στρώμα.

2. Συνελικτικά νευρωνικά δίκτυα : Κάθε επόμενο στρώμα είναι ένα σύνολο μη γραμ- μικών συναρτήσεων σταθμισμένων αθροισμάτων χωρικά κοντινών υποσυνόλων εξόδων από το προηγούμενο στρώμα.

3. Περιοδικό νευρωνικό δίκτυο : Κάθε επόμενο στρώμα είναι μια συλλογή μη γραμμικών λειτουργιών σταθμισμένων αθροισμάτων εξόδων και της προηγούμενης κατάστασης.
Η ίδεα πίσω από το περιοδικό νευρωνικό δίκτυο έχει να κάνει με την απόφαση του τι να «ξεχάσει» και τι να “περάσει” ως κατάσταση στο επόμενο στρώμα.

Τα τελευταία χρόνια, η τεχνητή νοημοσύνη έχει φέρει την αυτοματοποίηση σε πληθώρα εφαρμογών της καθημερινότητας, από την ιατροφαρμακευτική περίθαλψη, την αυτόνομη οδήγησηως και τις διαστημικές εφαρμογές. Τα νευρωνικά δίκτυα παρουσιάζουν υψηλή υπολογιστική πολυπλοκότητα, κάτι που δυσκολεύει τους επεξεργαστές γενικού σκοπού να ανταποκριθούν στις απαιτήσεις. Καθώς οι σύγχρονες εφαρμογές απαιτούν λήψη αποφάσεων σε πραγματικό χρόνο, η επεξεργασία και ανάλυση δεδομένων μετακινείται at the edge(εκτός των data center), δηλαδή όλο και πιο κοντά στους αισθητήρες συλλογής των δεδομένων. Ως εκ τούτου, δημιουργείται η ανάγκη για ανάπτυξη πιο εξειδικευμένων και καινοτόμων επεξεργαστών και επιταχυντών τεχνητής νοημοσύνης όπως είναι οι GPUs, οι VPUs και τα FPGAs. Προβλέποντας αυτή την τάση, η Google ανέπτυξε τις Tensor Processing Units (TPUs) για να επιταχύνει την εκτέλεση εφαρμογών μηχανικής μάθησης. Η TPU είναι ένας επιταχυντής που προσφέρει υψηλή απόδοση με μικρό φυσικό και ενεργειακό αποτύπωμα, επιτρέποντας την ανάπτυξη εφαρμογών ΤΝ τόσο σε κέντρα δεδομένων όσο και at the edge

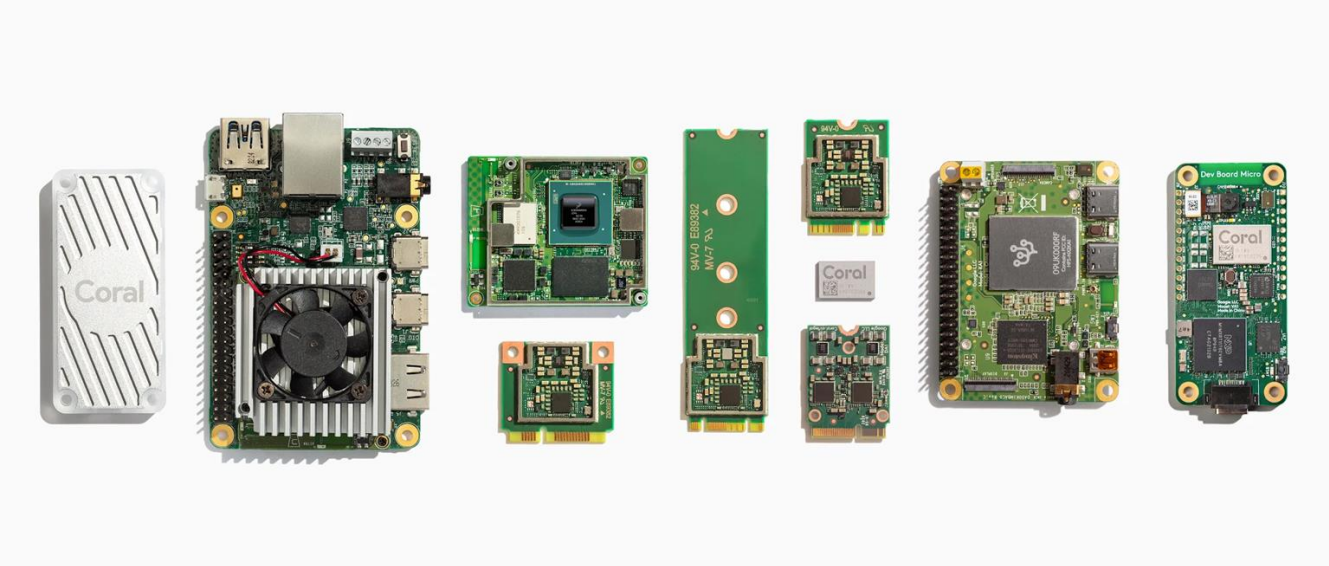
1.2 Μια ιστορική αναδρομή (4 γενιές)

1. Η πρώτης γενιάς TPU (2016) είναι μια μηχανή πολλαπλασιασμού μήτρας 8-bit, που οδηγείται με οδηγίες CISC από τον κεντρικό επεξεργαστή σε έναν δίαυλο PCIe 3.0. Κατασκευάζεται σε διαδικασία 28 nm με μέγεθος καλουπιού ≤ 33 mm2. Η ταχύτητα ρολογιού είναι 700MHz και έχει ισχύ θερμικής σχεδίασης 28–40W. Διαθέτει 28MiB μνήμης στο τσιπ και 4MiB συσσωρευτών 32-bit που λαμβάνουν τα αποτελέσματα μιας συστολικής συστοιχίας 256×256 8-bit. πολλαπλασιαστές. Στο πακέτο TPU υπάρχουν 8GiB DDR3SDRAM διπλού καναλιού 2133MHz που προσφέρει εύρος ζώνης 34GB/s. Οι εντολές μεταφέρουν δεδομένα προς ή από τον κεντρικό υπολογιστή, εκτελούν πολλαπλασιασμούς ή συνελίξεις πινάκων και εφαρμόζουν συναρτήσεις ενεργοποίησης.

2. Η δεύτερη γενιά TPU ανακοινώθηκε τον Μάιο του 2017. Ο σχεδιασμός TPU της πρώτης γενιάς περιοριζόταν από το εύρος ζώνης μνήμης και η χρήση 16 GB μνήμης υψηλού εύρους ζώνης στη σχεδίαση δεύτερης γενιάς αύξησε το εύρος ζώνης στα 600GB/s και την απόδοση στα 45teraFLOPS. Στη συνέχεια, οι TPU διατάσσονται σε μονάδες τεσσάρων τσιπ με απόδοση 180teraFLOPS. Στη συνέχεια, 64 από αυτές τις μονάδες συναρμολογούνται σε pods 256 chip με απόδοση 11.5petaFLOPS. Συγκεκριμένα, ενώ οι ***TPU πρώτης γενιάς περιορίζονταν σε ακέραιους αριθμούς, οι TPU δεύτερης γενιάς μπορούν επίσης να υπολογίζουν σε κινητή υποδιαστολ***. Αυτό καθιστά τα TPU δεύτερης γενιάς χρήσιμα τόσο για εκπαίδευση όσο και για εξαγωγή συμπερασμάτων μοντέλων μηχανικής εκμάθησης.

3. Η τρίτη γενιά επξεργαστών TPU (2018) είναι δύο φορές πιο ισχυροί από τους TPU δεύτερης γενιάς και θα αναπτυχθούν σε pods με τέσσερις φορές περισσότερα τσιπ από την προηγούμενη γενιά. Αυτό έχει ως αποτέλεσμα 8 φορές αύξηση στην απόδοση ανά pod (με έως και 1.024 μάρκες ανά pod) σε σύγκριση με την ανάπτυξη TPU δεύτερης γενιάς.

4. Η τέταρτη γενιά επεξεργαστών TPU ανακηνώθηκε τον Μάιο του 2021. Η 4η έκδοση των TPU βελτίωσε την απόδοση περισσότερο απο το διπλάσιο σε σχέση με τα τσιπ της 3ης γενιάς. Ένα μεμονωμένο v4 pod περιέχει 4.096 chips v4 και κάθε pod έχει δεκαπλάσια φορές το εύρος ζώνης διασύνδεσης ανά τσιπ σε κλίμακα, σε σύγκριση με οποιαδήποτε άλλη τεχνολογία δικτύωσης.



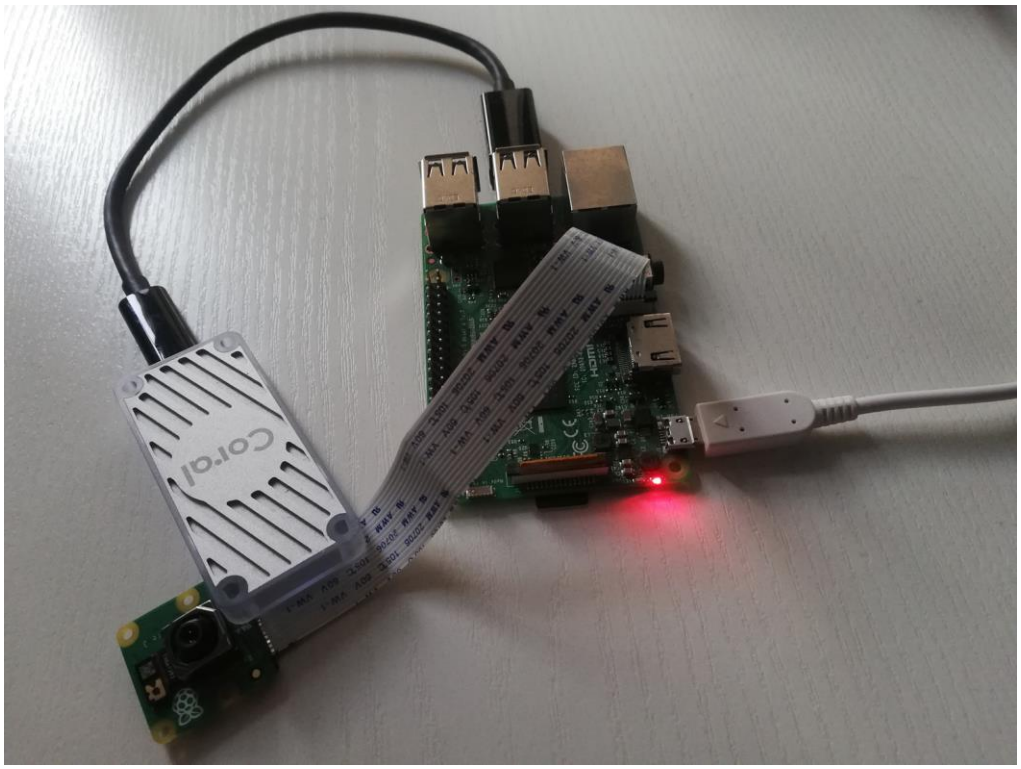
Εικόνα 1

Τον Ιούλιο του 2018, η Google ανακοίνωσε το Edge TPU. Το Edge TPU είναι το ειδικά κατασκευασμένο τσιπ ASIC της Google που έχει σχεδιαστεί για την εκτέλεση μοντέλων μηχανικής εκμάθησης (ML) για υπολογιστές αιχμής, που σημαίνει ότι είναι πολύ μικρότερο και καταναλώνει πολύ λιγότερη ενέργεια σε σύγκριση με τα TPU που φιλοξενούνται στα κέντρα δεδομένων της Google (γνωστά και ως Cloud TPU.). Τον Ιανουάριο του 2019, η Google έκανε το Edge TPU διαθέσιμο σε προγραμματιστές με μια σειρά προϊόντων με την επωνυμία Coral. Το Edge TPU είναι ικανό για 4 τρισεκατομμύρια λειτουργίες ανά δευτερόλεπτο με 2W ηλεκτρικής ισχύος. Ο χρόνος εκτέλεσης μηχανικής εκμάθησης που χρησιμοποιείται για την εκτέλεση μοντέλων στο Edge TPU βασίζεται στο TensorFlow Lite. Το Edge TPU είναι ικανό μόνο να επιταχύνει λειτουργίες διέλευσης, που σημαίνει ότι είναι κυρίως χρήσιμο για την εκτέλεση συμπερασμάτων. Το Edge TPU υποστηρίζει επίσης μόνο μαθηματικά 8-bit, πράγμα που σημαίνει ότι για να είναι συμβατό ένα δίκτυο με το Edge TPU, πρέπει είτε να εκπαιδευτεί χρησιμοποιώντας την τεχνική εκπαίδευσης με επίγνωση κβαντοποίησης TensorFlow ή η χρήση μετα- κβαντοποίηση εκπαίδευσης. Πλέον οι επεξεργαστές TPU ενσωματόνονται και σε κινητές συσκευές (για την επιτάχυνση εφαρμογών επεξεργασίας εικόνας και ήχου).

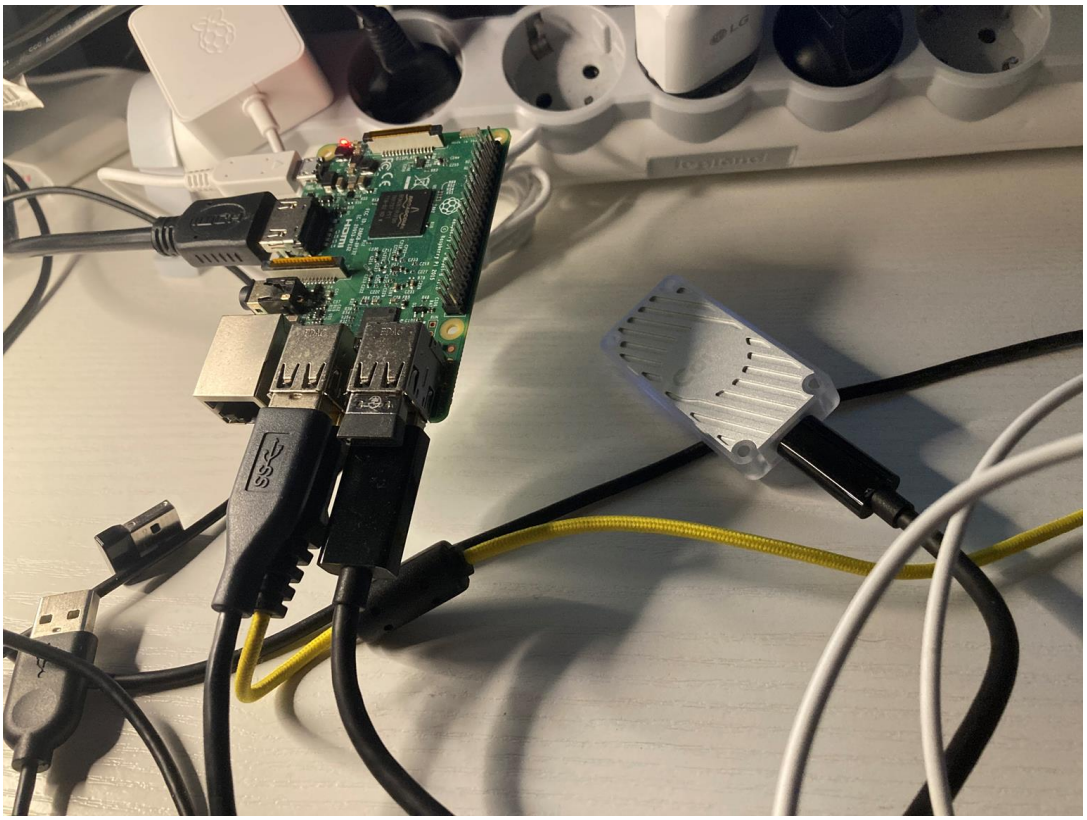
1.3 Συσκευές Coral

Το Coral TPU βγαίνει σε διάφορες εκδόσεις είτε για πρώτότυπα προϊόντα είτε για παραγωγής. Υπάρχει το raspberry pi (ARM CPU), USB Coral Accelerator, Dev Board mini.

Η έκδοση που θα μελετήσουμε είναι το USB Accelerator (με τη βοήθεια ενός raspberry pi 3A+ model και ενός Laptop με επεξεργαστή AMD Ryzen5 3500U CPU).



Εικόνα 2



Εικόνα 3

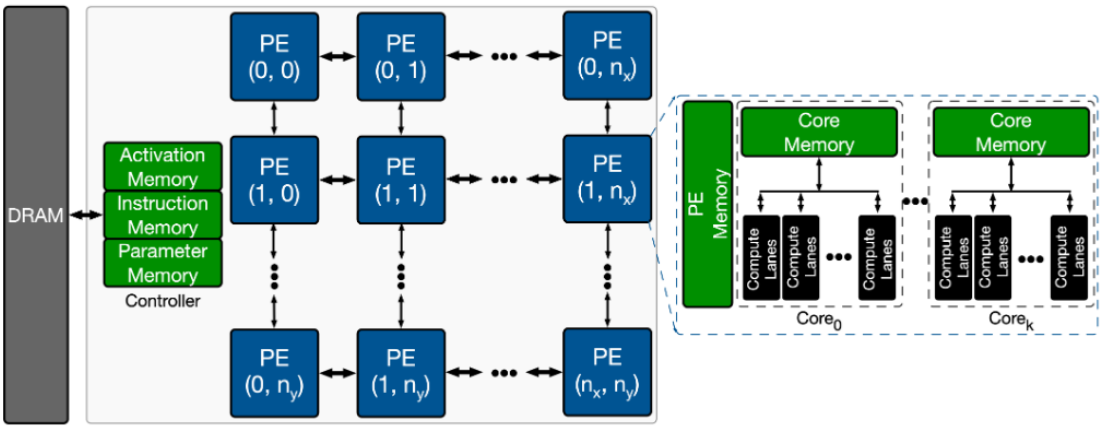
Κεφάλαιο 2

2.1 Η αρχιτεκτονική

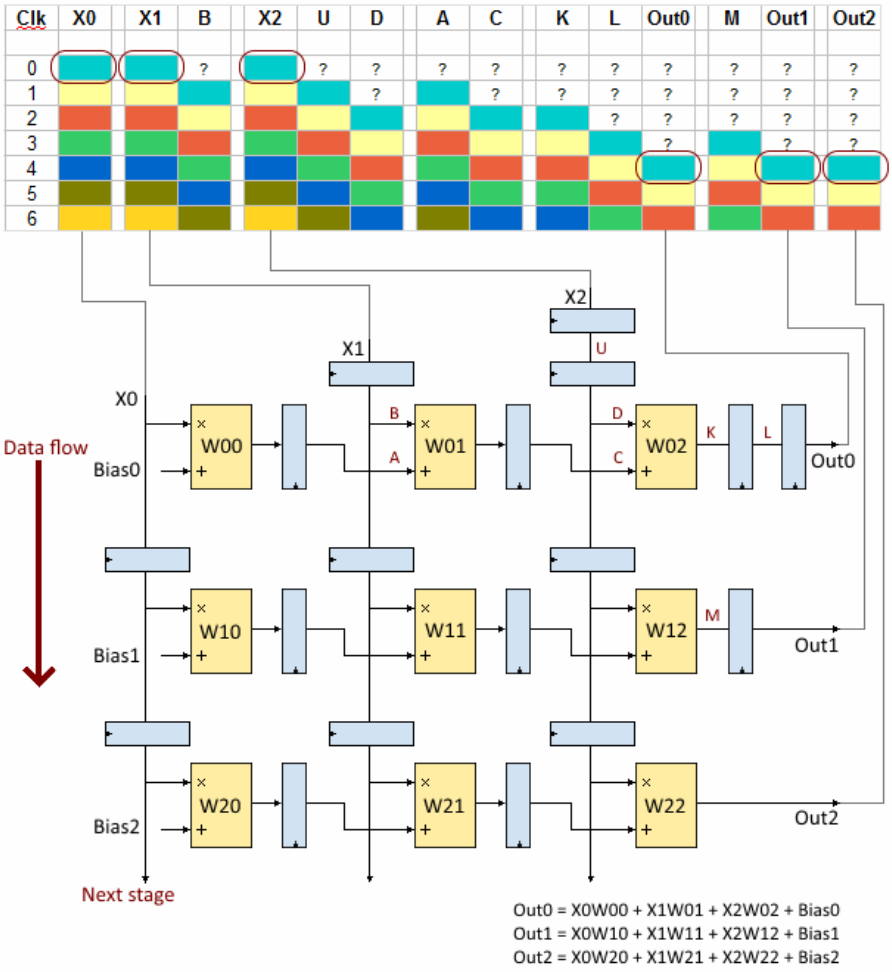
Η αρχιτεκτονική του Edge TPU επιταχύνει την εκτέλεση νευρωνικών δικτύων για σύγχρονες εφαρμογές TN. Το ASIC του Edge TPU βασίζεται σε μια συστολική συστοιχία (systolic array) πολλαπλασιαστών και συσσωρευτών, οι διαστάσεις των οποίων δεν έχουν αποκαλυφθεί ακόμα, αλλά σύμφωνα με την Q-Engineering, εκτιμάται ως μια συστοιχία μεγέθους 64 * 64 με συχνότητα ρολογιού τουλάχιστον clock = 480MHz. Μία μεμονωμένη TPU είναι ικανή να εκτελέσει 4 τρισεκατομμύρια πράξεις ανά δευτερόλεπτο (TOPS), χρησιμοποιώντας 0,5 Watt ισχύος για κάθε TOPS (2 TOPS / Watt). Τρέχοντας στα 480 MHz μπορεί θεωρητικά να εκτελέσει 64 * 64 * 480.000.000 = 2 τρισεκατομμύρια πολλαπλασιασμούς και προσθέσεις ανά δευτερόλεπτο. Η αν κοιτάξουμε σε μεμονωμένες πράξεις, είναι 4 τρισεκατομμύρια (4 TOPS). Η κατανάλωση ισχύος του chip φτανει τα 1,5 Watt (1.5A), ενώ συνολικα όλο το SoM καταναλώνει περί τα 4,5 Watt ισχύος.

Η μνήμη, το Edge TPU ενσωματώνει μεγάλη μνήμη εντός του chip για να διατηρεί τα ενδιάμεσα αποτελέσματα που επαναχρησιμοποιούνται αργότερα. Αυτό επιτρέπει μεγαλύτερη ταχύτητα εκτέλεσης σε σύγκριση με τη φόρτωση των δεδομένων των παραμέτρων από εξωτερική μνήμη. Η SRAM του TPU χρησιμοποιείται ως «cache», παρόλο που είναι μια scratchpad μνήμη που έχει εκχωρηθεί από τον μεταγλωτιστή. Σε αντίθεση με τους συμβατικούς επεξεργαστές, το Edge TPU χρησιμοποιεί ένα σετ εντολών τύπου CISC και οποιαδήποτε λειτουργία συντελείται μέσω του κεντρικού υπολογιστή. Επίσης, οι μονάδες TPU υποστηρίζουν μόνο λειτουργίες σε περιορισμένο επίπεδο ακρίβειας, όσο επαρκεί για να ικανοποιήσει τις απαιτήσεις των σύγχρονων εφαρμογών TN, μειώνοντας σημαντικά τόσο το κόστος τους όσο και τις ενεργειακές απαιτήσεις.

(Λεπτομέρειες από την Google για την αρχιτεκτονική του TPU δεν έχουν δοθεί)



Διάγραμμα 1



Διάγραμμα 2

2.2 TensorFlow και εργαλεία ανάπτυξης

Το TensorFlow είναι μαθηματική βιβλιοθήκη την οποία ανέπτυξε η Google(ομάδα τεχνητής νοημοσύνης Deep Mind). Σκοπός της δημιουργίας της ήταν η διευκόλυνση διαδικασιών όπως ο προγραμματισμός ροής δεδομένων, ενώ βρήκε χρήση και στην μηχανική μάθηση έχοντας ως παράδειγμα τα νευρωνικά δίκτυα. Η δημόσια διανομή του TensorFlow έγινε τον Νοέμβριο του 2015 κάτω από την άδεια ανοιχτού λογισμικού της Apache (Apache 2.0 Open Source License).

Η διαδικασία που ακολουθείται για την εφαρμογή ενός μοντέλου μηχανικής μάθησης στο TPU είναι απλή και περιλαμβάνει τα ακόλουθα βήματα. Ανάπτυξη του κώδικα σε TensorFlow (δομή και εκμάθηση του νευρωνικού δικτύου). Στη συνέχεια, το μοντέλο μετατρέπεται σε format «TensorFlow Lite», μεταγλωτίζεται από τον Edge TPU Compiler και τέλος εκτελείται στη TPU με χρήση του TF Lite API. Το TensorFlow Lite είναι ένα framework που επιτρέπει την ταχύτερη ανάπτυξη και εκτέλεση εφαρμογών μηχανικής μάθησης σε ενσωματωμένες συσκευές. Ένα μοντέλο TensorFlow Lite αντιπροσωπεύεται σε μια ειδική φορητή μορφή γνωστή ως Flat-Buffers. Αυτό παρέχει πολλά πλεονεκτήματα σε σχέση με τη μορφή buffer πρωτοκόλλου του TensorFlow, όπως μειωμένο μέγεθος (μικρό αποτύπωμα κώδικα) και ταχύτερη εξαγωγή συμπερασμάτων (τα δεδομένα έχουν άμεση πρόσβαση χωρίς περαιτέρω ανάλυση) που επιτρέπει στο TensorFlow Lite να εκτελείται αποτελεσματικά σε συσκευές με περιορισμένους υπολογιστικούς πόρους και περιορισμένη μνήμη. Προκειμένου να μπορεί να εκτελεστεί και να επιταχυνθεί ένα μοντέλο από το TPU πρέπει να ικανοποιούνται κάποιες προϋποθέσεις. Το TPU υποστηρίζει πράξεις μόνο για 8-bit ακέραιους αριθμούς, κάτι που σημαίνει πως το μοντέλο πρέπει να κβαντιστεί κατά την μετατροπή σε «.tflite» μέσω του TF Lite Converter. Η κβάντιση είναι μια μορφή βελτιστοποίησης, για τη μείωση του μεγέθους του μοντέλου και του χρόνου εκτέλεσης με ελάχιστη απώλεια ακρίβειας.

Με τη χρήση του converter τα μοντέλα αποθηκεύονται σε μορφή «SavedModel» και δημιουργούνται είτε χρησιμοποιώντας API υψηλού επιπέδου τύπου «Keras» είτε χαμηλού επιπέδου (από τα οποία δημιουργούνται concrete functions). Για να πραγματοποιηθεί η κβάντιση πρέπει να οριστεί ενα representative dataset, δηλαδή ένα σύνολο δεδομένων ενδεικτικών των αναμενόμενων εισόδων και εξόδων του εκάστοτε μοντέλου. Αυτό γίνεται ώστε κατά την κβάντιση και αποκβάντιση οποιουδήποτε μοντέλου, το TPU να λάβει την σωστή είσοδο αλλά και να μας επιστρέψει τη σωστή έξοδο.

Στο δεύτερο στάδιο επεξεργασίας ενός μοντέλου, ώστε να είναι συμβατό με το TPU, επιστρατεύεται ο Edge TPU Compiler. Ο Compiler αντιστοιχίζει τα operations ενός δικτύου σε εκείνα που υποστηρίζει το TPU, αν αυτό είναι εφικτό, καθώς υποστηρίζονται μόνο ορισμένα operations για TN, όπως είναι η συνέλιξη, το max pooling, ο πολλαπλασιασμός πινάκων. Οι μη υποστηριζόμενες λειτουργίες εκτελούνται από τον ARM επεξεργαστή, κάτι που συνεπάγεται επιπρόσθετο χρόνο στην εκτέλεση ενός δικτύου. Επίσης, ο compiler είναι αυτός που προ-αποφασίζει πόση από την διαθέσιμη SRAM του TPU θα αναθέσει σε κάθε μοντέλο που μεταγλωττίζεται για αυτό, ενώ στην περίπτωση που η χωρητικότητα της δεν μπορεί να εξυπηρετήσει ένα δίκτυο, τότε γίνεται χρήση και της εξωτερικής μνήμης.
Όλα τα αποελέσματα βασίζονται στο Edge TPU βασίζονται στα API TensorFlow Lite (το TensorFlow υποτυρίζει Python και C++).

Ο μετατροπέας TensorFlow Lite παίρνει ένα μοντέλο TensorFlow και δημιουργεί ένα TensorFlow Μοντέλο “Lite” (μια βελτιστοποιημένη μορφή FlatBuffer τα αρχεία με κατάληξη .tflite). Κατά τη διάρκεια της μετατροπής, μπορούν να εφαρμοστούν βελτιστοποιήσεις όπως η κβαντοποίηση για τη μείωση του μεγέθους του μοντέλου και λανθάνουσα κατάσταση με ελάχιστη ή καμία απώλεια στην ακρίβεια. Στο TensorFlow 2, τα μοντέλα αποθηκεύονται χρησιμοποιώντας το μορφή SavedModel και δημιουργούνται είτε χρησιμοποιώντας τα υψηλού επιπέδου tf.keras.* API (Keras μοντέλο) ή τα χαμηλού επιπέδου tf.* API (από τα οποία δημιουργείτε συγκεκριμένες συναρτήσεις).

2.3 Λογισμικά και εγκατάσταση

Η διαδικασία εγκατάστασης είναι εξαρτάται από την έκδοση του λειτουργικού στα Linux. Το Edge TPU υποστηρίζει εκδόσεις της python μεταξύ των 3.6 και 3.9. Οτιδήποτε νεότερο ή παλιότερο δεν θα αφήσει τα μοντέλα να τρέξουν στην πλακέτα.

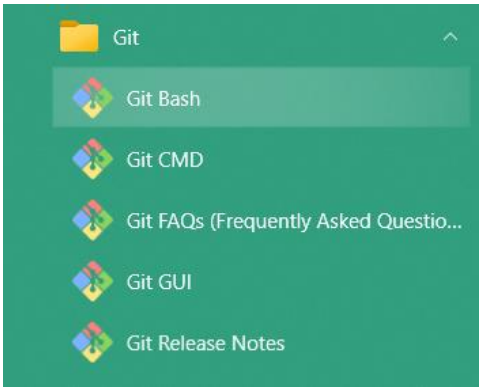
Μπορεί να χρησιμοποιηθεί εικονικό περιβάλλον για στηθεί μία από τις συμβατές εκδόσεις στην περίπτωση που το λειτουργικό έχει προγκατεστημένη νεότερη έκδοση της python (μη δοκιμάσετε να απεγκαταστήσετε την Python, πλέον είναι απαραίτητη για τα γραφικά του λειτουργικού). Η διαδικασία είναι περίπλοκη και μακροσκελής, αλλά υπάρχει εκτενής οδηγός: pyenv.

2.3.1 Windows 10 (Επίσημα δεν υποστηρίζονται τα Windows 11 αλλά η εγκατάσταση είναι δυνατή ακολουθώντας ακριβώς τα ίδια βήματα)

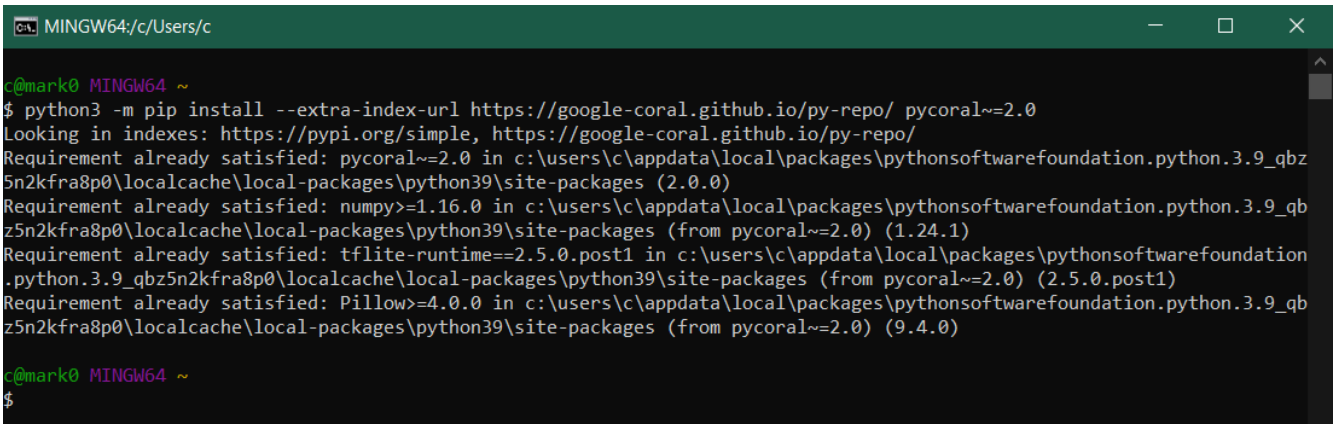
- Εγκατάσταση Python

Εκδόσεις που υποστηρίζονται 3.6 – 3.9 (Συνήσταται η έκοδση 3.8.6 ως η πιο σταθερή)

- Η εγκατάστση ενός Bash κάνει την διαδικασία της εκτέλεσης των μοντέλων και την εγκατάσταση διάφορων βιβλιοθηκών αρκετά εύκολη. Χρησιμοποιήσα το git για Windows(64bit έκδοση).
- Απαιτείται η βιβλιοθήκη της C++. Από github κατεβάστε και κάντε extract τον φάκελο. Εκτελέστε το αρχείο install.bat
- PyCoralΑνοίξτε το Git Bash και εκτελέστε την εντολή



python3 -m pip install –extra-index-url https://google-coral.github.io/py-repo/ pycoral~=2.0



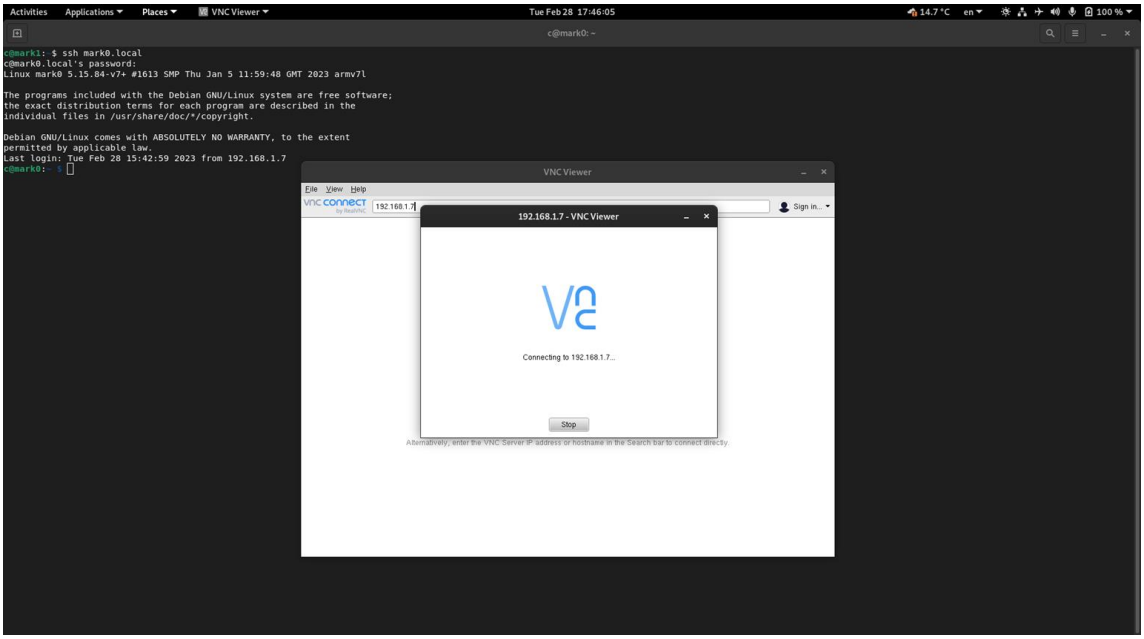
Εικόνα 4

Μετά την ολοκήρωση της διαδικασίας το usb Accelerator μπορεί να συνδεθεί στον υπολογιστή (Για τη μέγιστη απόδοση να γίνει χρήση USB3 θύρα)

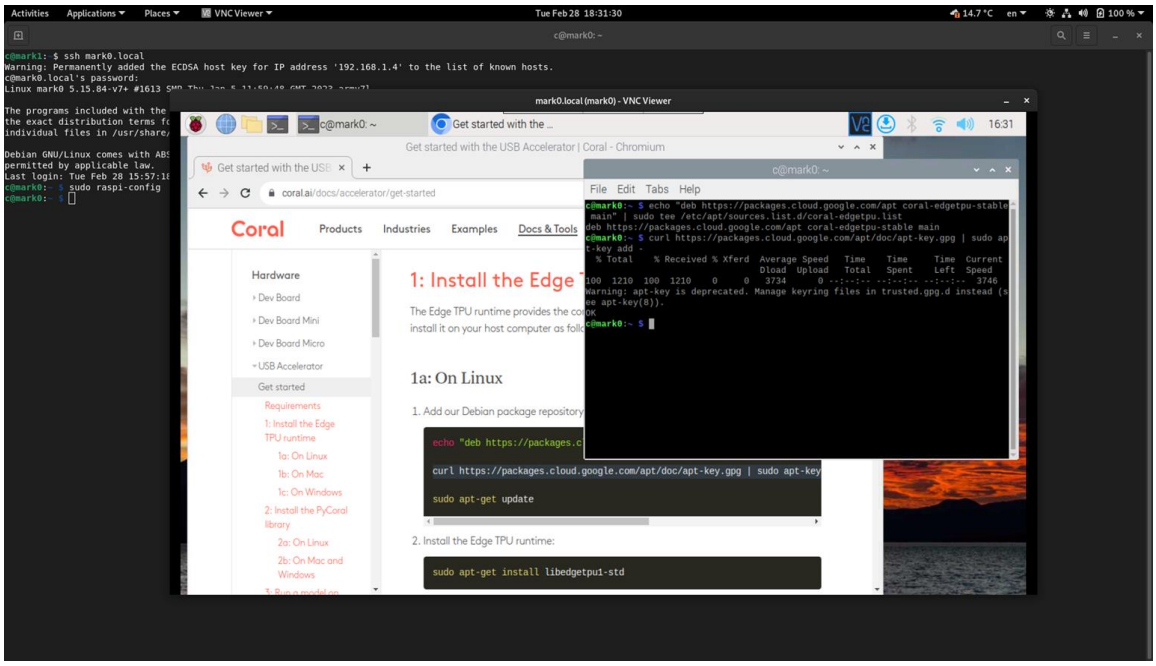
2.3.2 Linux (Raspberry Pi OS / Ubuntu / Debian με χρήση pyenv και έκδοση python 3.8.6)

Η διαδικασία της εγκατάστασης των προαπαιτούμενων βιβλιοθηκών σε όλες τις Debian εκδόσεις είναι πανομοιότυπη (ενδέχεται να χρείαζεται η εγκατάσταση επιπλέον πακέτων όπως το matplotlib). (Από την οικογένεια των Ubuntu τα Ubuntu 18.04 παρατηρήθηκε ότι είναι τα πιο σταθερά. Οι εκδόσεις των Debian δεν παρουσίασαν κανένα πρόβλημα). Αντίστοιχα στις εκδόσεις του Raspberry Pi OS δεν παρουσίασαν σφάλματα.

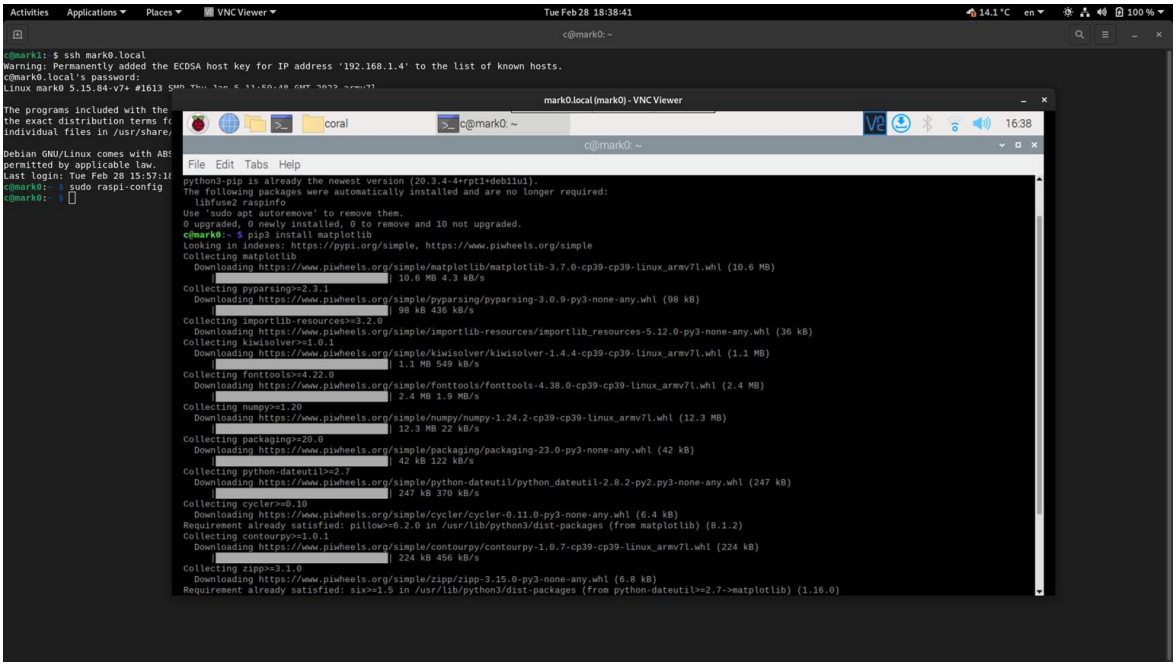
Στο Raspberry Pi δύνεται η δυνατότητα να λειτουργήσει χωρίς οθόνη, χρησιμοποιώντας SSH κατά την εγκατάσταση (σε τοπικό δίκτυο με προκαθορισμένη IP, για να είναι δυνατή η σύνδεση από εξωτερικό υπολογιστή). Για την υποστήριξη γραφικών GUI μέσω SSH έγινε χρήση του VNC-Viewer (είναι ήδη προεγκατεστημένο στο Raspberry Pi OS).



Εικόνα 5



Εικόνα 6



Εικόνα 7

Κατά την εγκατάσταση δίνεται η δυνατότητα ο TPU να τρέχει με τη μέγιστη συχνότητα στο ρολόι του (sudo apt-get install libedgetpu1-max).

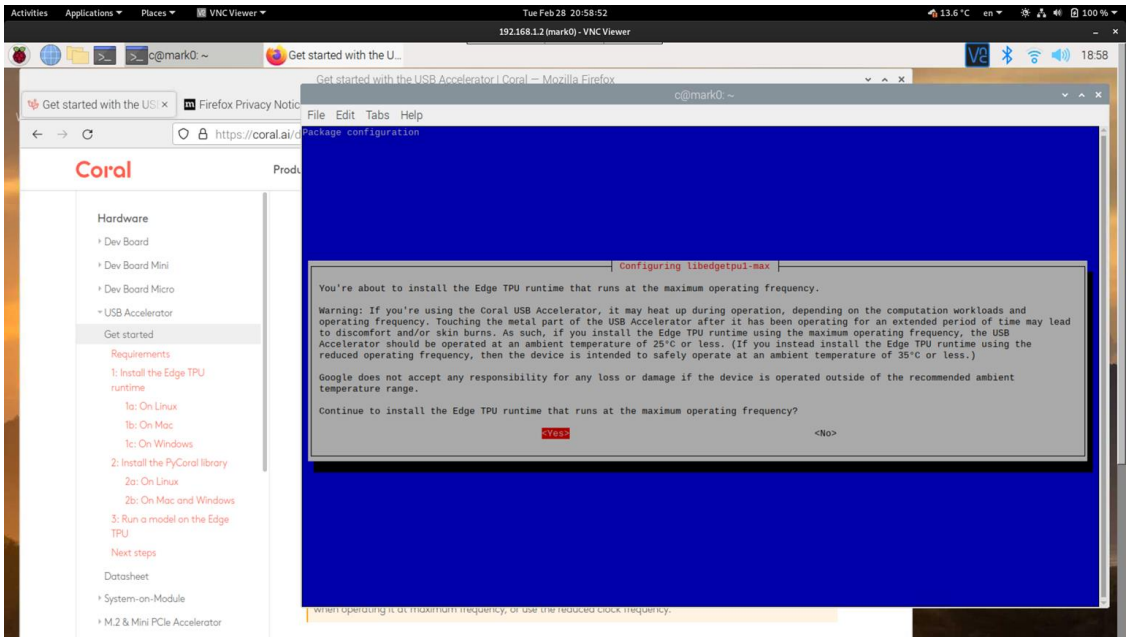
ΠΡΟΣΟΧΗ

Ανάλλογα την έκταση και των φόρτο των εργασιών, αυξάνεται και η θερμοκρασία του μεταλλικού πλαισίου!

Παρατηρήθηκαν «κολλήματα» στον TPU ειδικά σε υψηλές θερμοκρασίες, ειδικά όταν ήταν σε λειτουργία μέγιστης συχνίοτητας.

Για την επαναφορά στις εργοστασιακές ρυθμίσεις του TPU εκτελούμε την παρακάτω εντολή :

sudo apt-get install libedgetpu1-std



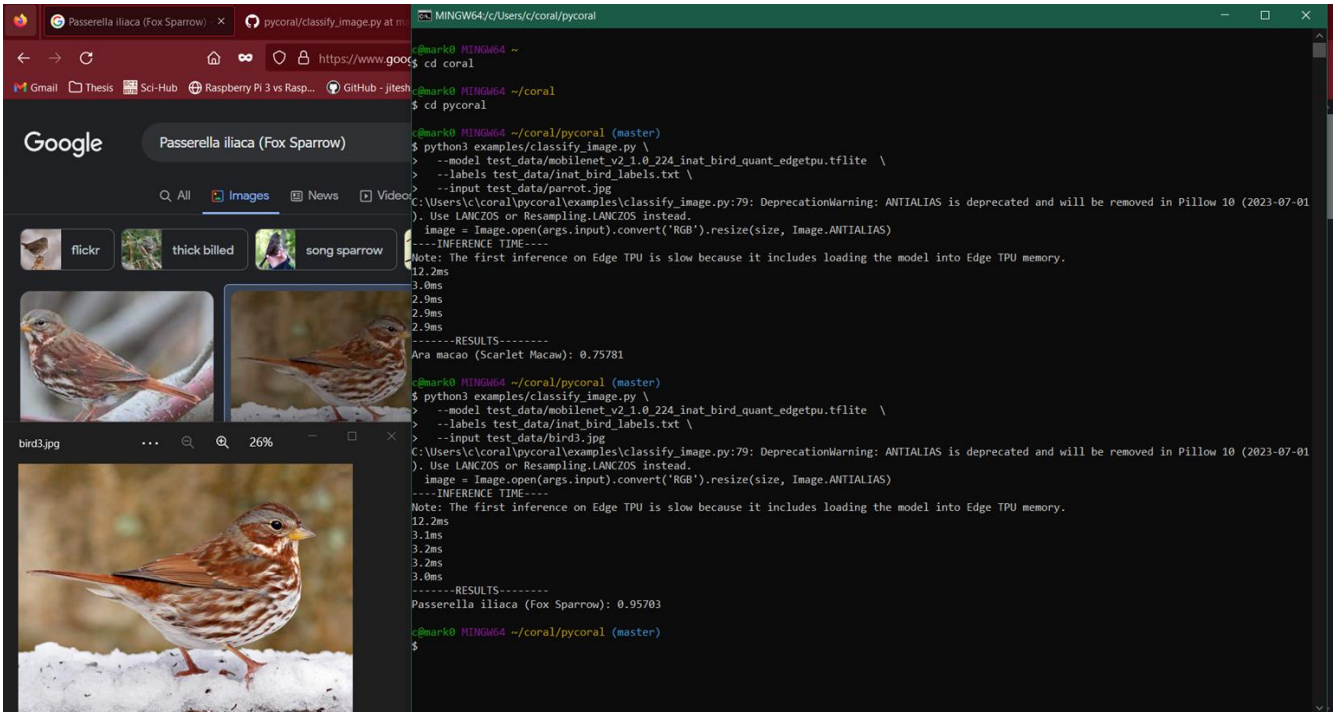
Εικόνα 8

Κεφάλαιο 3

3.1 Παράδειγματα και Bench markings

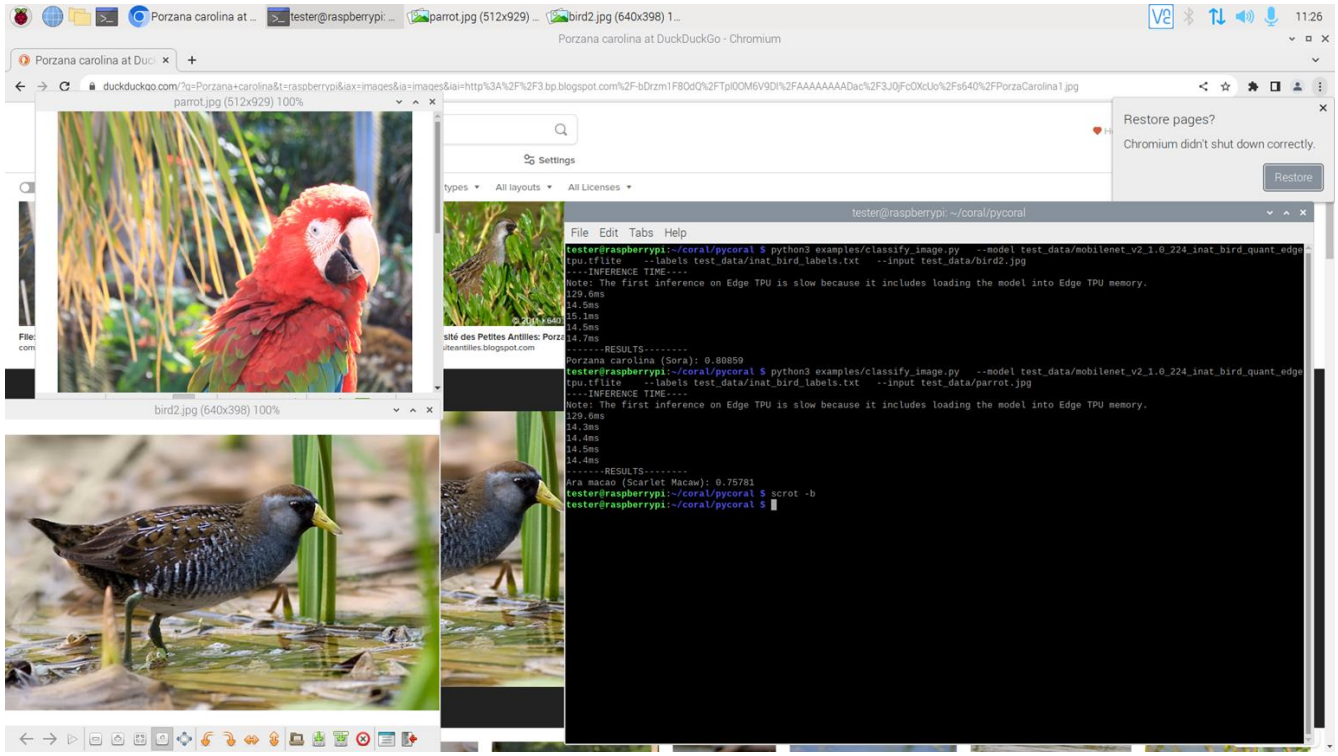
Windows10 (USB3 port AMD Ryzen 5 3500U 64bit, 8GB RAM**)**

Το παράδειγμα που ακολουθεί είναι ένα «έτοιμο» πρόγραμμα με προεκπαιδευμένο μοντέλο από την Google για τον εντοπισμό και την ταξινόμιση πτηνών.



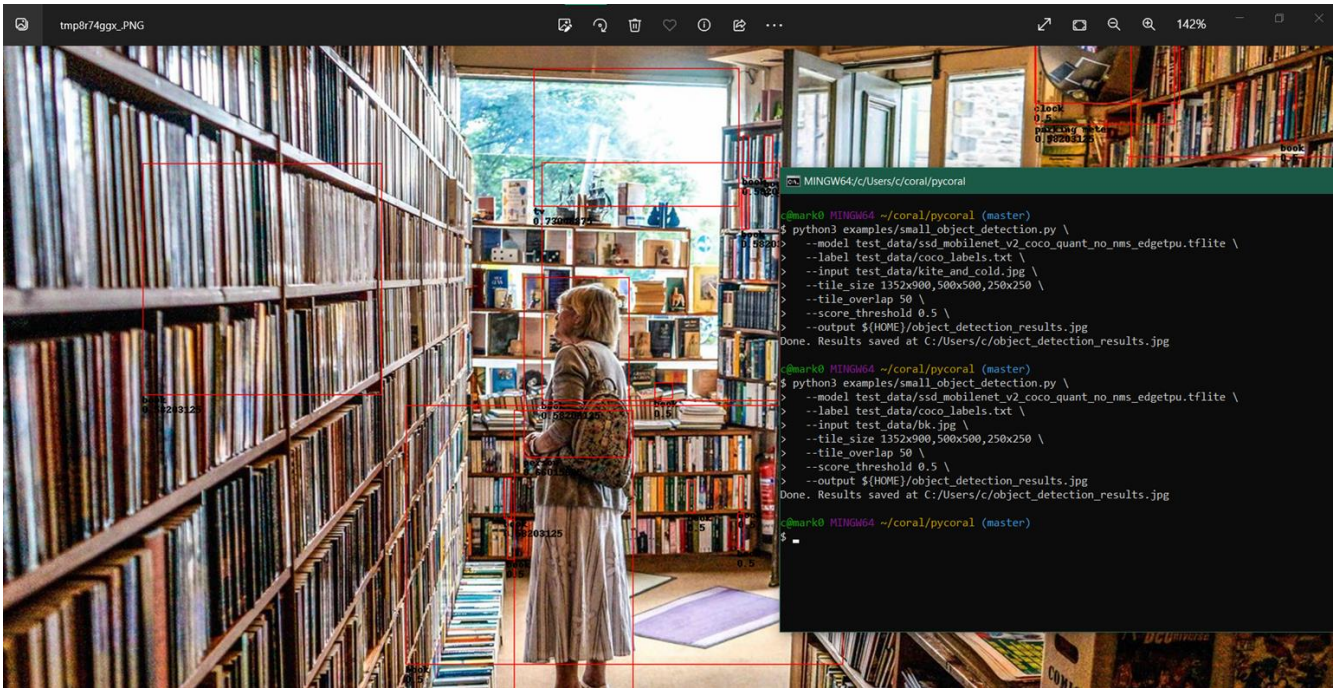
Εικόνα 9

Αποτελέσματα του προγράμματος classify_image.py με δεδομένα διάφορα πτηνά.
Παρατηρούμε ότι η πρώτη σάρωση είναι και η πιο αργή (σε αυτήν «φορτώνονται» τα μοντέλα και οι εικόνες στον TPU). Οι λοιπές σαρώσεις για την «εξακρίβωση» και ταυτοποίηση του πτηνού είναι «σχεδόν» ακαριαίες. Στο τελικό αποτέλσμα αναγράφεται και το ποσοστό επιτυχίας του μοντέλου ως προς την εικόνα-είσοδο (0.95 = 95% ακρίβεια στο μοντέλο).

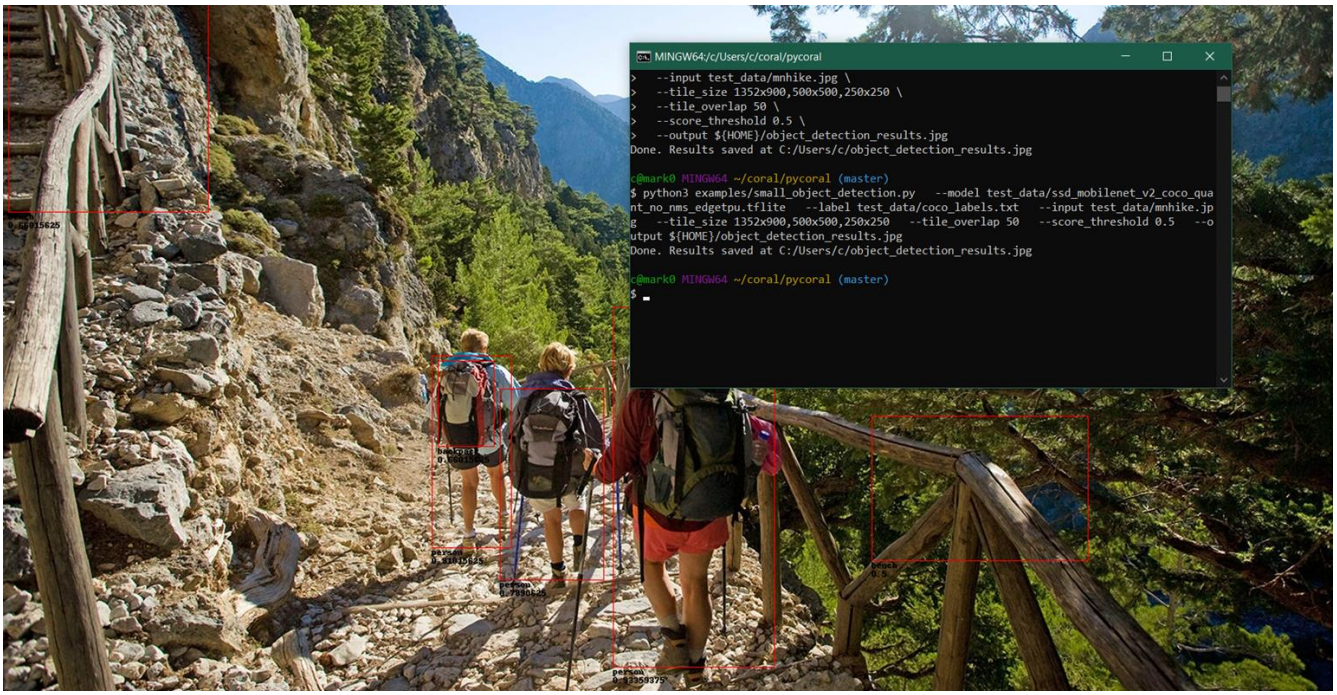


Εικόνα 10

Αντίστοιχα το ίδιο πρόγραμμα σε Raspberry pi 3A+ (USB2 port Cortex-A53 (ARMv8) 64bit 512MB RAM). Παρατηρούμε «δραματική» αύξηση στους χρόνους ανάγνωσης του μοντέλου αλά και της εκτέλεσης του μοντέλου (10 – 14 φορές μεγαλύτερες). Αυτό οφείλεται στη θύρα USB2 του Raspberry Pi3 (σε σύγκριση με την USB3 θύρα του υπολογιστή).



Εικόνα 11



Εικόνα 12

Αποτέλέσματα από τα «έτοιμα» μοντέλα και προγράμματα που δίνει η Google. Το πρόγρμμα ονομάζεται small_object_detection.py. (Για την διαφορετική είσοδο εικόνων πρέπει να απο-

θηκευσουμε την/τις εικόνες στον φάκελο με τα tests και να αλλάξουμε αντίστοιχα και το path κατά την εκτέλεση των εντολών (το όνομα της εκάστοτε εικόνας), όπως φαίνεται και στις παραπάνω εικόνες με τους ορειβάτες και το βιβλιοπωλείο.

3.2 Raspberry PI

Το παρακάτω πείραμα αφορά τη μέτρηση της απόδοσης 4 μοντέλων (Pi 4 4GB – 8GB , Pi 3B και Pi 3A+) του Raspberry PI. Η απόδοση μετριέται με και χωρίς επιταχυντή Coral USB. Το ίδιο σύνολο σεναρίων Python χρησιμοποιείται για την εκτέλεση ταξινόμησης εικόνων χρησιμοποιώντας ένα μοντέλο μηχανικής εκμάθησης (MobileNet V1) σε όλα τα μοντέλα. Αυτό επιτυγχάνεται με την εναλλαγή της ίδιας κάρτας micro SD μεταξύ των διαφορετικών παραλλαγών. Η ρύθμιση φαίνεται στην παρακάτω εικόνα.

- 24elegate.py χρησιμοποιεί το mobilenet_v1_1.0_224_quant.tflite μοντέλο και δεν απαιτεί το coral usb

- 24elegate_coral.py χρησιμοποιεί το mobilenet_v1_1.0_224_quant_edgetpu.tflite μοντέλο και κάνει χρήση του usb accelerator

Η διαδικασία που εκτυελούν και τα 2 προγράμματα είναι πανομοιότυπη

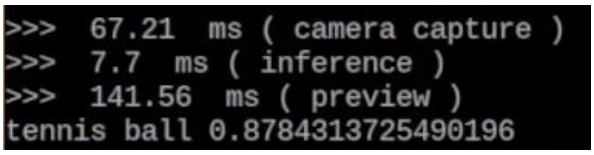
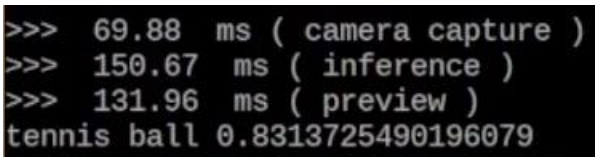
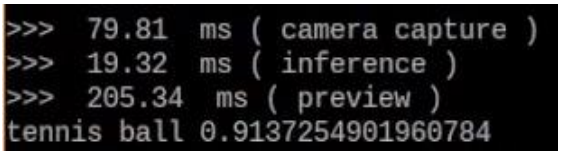
1. Εισαγάγετε το load_delgate από το tflite_runtime.interpreter

2. Αλλαγή της διαδρομή του αρχείου μοντέλου και κατεύθυνση προς το να δείχνει στο αρχείο μοντέλου ‘edgetpu’.

3. Δημιουργία διερμηνέα με συνάρτηση ‘load_delegate’.

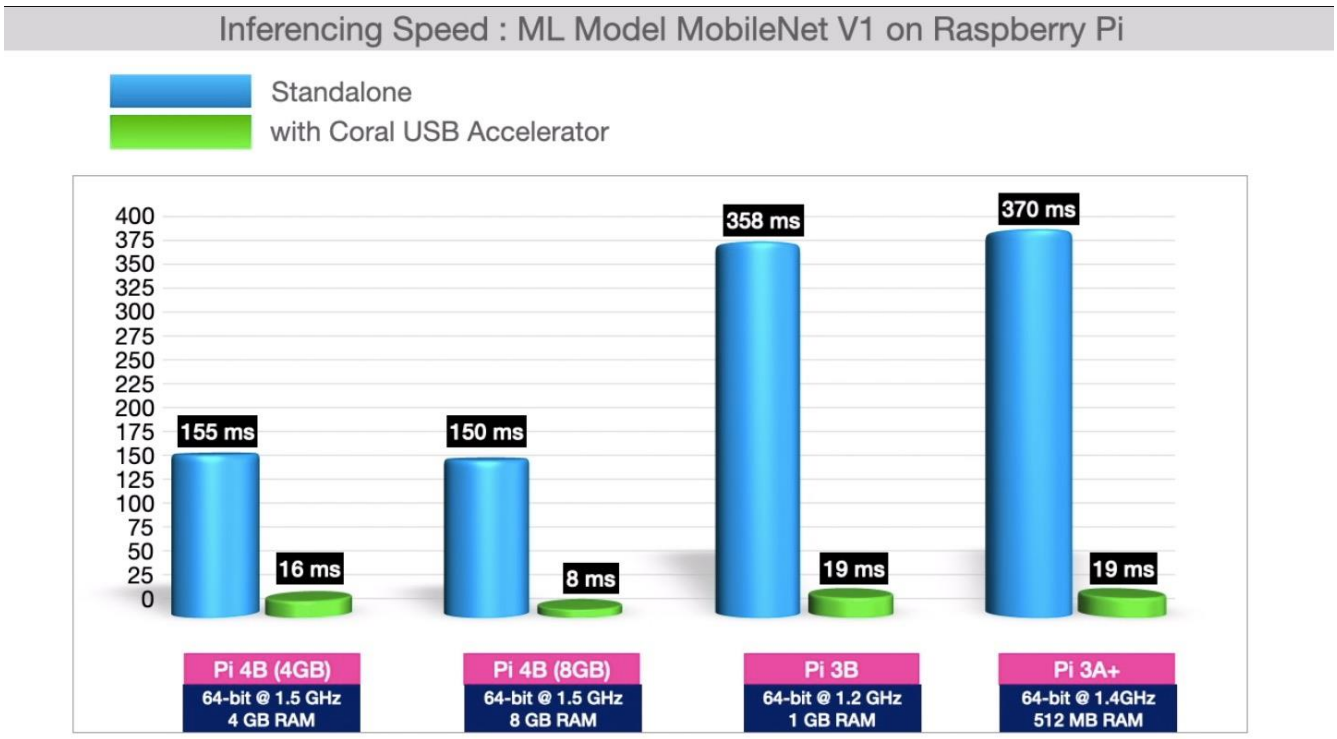
Η «Λήψη κάμερας» και η «Προεπισκόπηση» περιλαμβάνουν τη λήψη μιας εικόνας (frame) από την κάμερα και την εμφάνισή της σε ένα παράθυρο εξόδου. Υπάρχουν πολλές μέθοδοι. Μια τέτοια μέθοδος είναι να εκτελέσετε την εργασία που σχετίζεται με την κάμερα μέσω της βιβλιοθήκης OpenCV. Το συμπέρασμα περιλαμβάνει τη λήψη προβλέψεων από το αρχείο μοντέλου με βάση την εικόνα εισόδου. Ο χρόνος που απαιτείται σε αυτό το βήμα εξαρτάται από το αρχείο μοντέλου που χρησιμοποιείται. Ο χρόνος συμπερασμάτων μπορεί να διαφέρει από μοντέλο σε μοντέλο ανάλογα με το πόσες κλάσεις έχει. Χωρίς εξωτερική επιτάχυνση υλικού, αυτή η εργασία εκτελείται από την CPU και καταναλώνει σχεδόν όλους τους πόρους του επεξεργαστή. Προκειμένου να δημιουργηθούν εφαρμογές που χρησιμοποιούν ένα μοντέλο μηχανικής εκμάθησης για μια περίπτωση χρήσης σε πραγματικό χρόνο, είναι επιτακτική ανάγκη ο χρόνος εξαγωγής συμπερασμάτων να είναι όσο το δυνατόν χαμηλότερος για να επιτευχθεί μέγιστο FPS. Το πρόγραμμα Python classify_coral.py “τοποθετεί” το τμήμα συμπερασμάτων στον επιταχυντή USB Coral και μειώνει δραστικά τον χρόνο επεξεργασίας.

3.3 Συγρίσεις χρόνων

Χωρίς TPU	Με TPU
Raspberry Pi 4B (4GB)	
	
Raspberry Pi 4B (8GB)	
	
Raspberry Pi 3B	
	
Raspberry Pi 3A+	
	

Πίνακας 1

Σε όλες τις παραπάνω περιπτώσεις, παρατηρούμε τη δραστική μείωση του χρόνου συμπερασμάτων κατά την χρήση του Coral. Ωστόσο, η «λήψη κάμερας» και η «προεπισκόπηση» εξακολουθούν να απαιτούν τον ίδιο χρόνο, επειδή μόνο το τμήμα εξαγωγής συμπερασμάτων επεξεργάζεται μέσα στο υλικό Coral. Μια επισκόπηση των αποτελεσμάτων παρέχεται από αυτό το γράφημα.



Διάγραμμα 3

Ο κώδικας υλοποιεί ένα παράδειγμα που εκτελεί ανίχνευση αντικειμένων στα πλαίσια κάμερας χρησιμοποιώντας τη βιβλιοθήκη OpenCV.

Πριν την εκτέλεση του κώδικα, χρειάζεται να δηλωθούν τα εξής:

- TEST_DATA: η διαδρομή του φακέλου που περιέχει τα μοντέλα ανίχνευσης αντικειμένων και τις ετικέτες.

- Περιεχόμενα TEST_DATA: το μοντέλο ανίχνευσης προσώπου "mobilenet_ssd_v2_face_quant_postprocess_edgetpu.tflite" και το μοντέλο ανίχνευσης αντικειμένων "mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite", καθώς και το αρχείο ετικετών "coco_labels.txt" που περιέχει τις ετικέτες για το μοντέλο αντικειμένων.

Ο κώδικας χρησιμοποιεί τον πίνακα argparse για την είσοδο παραμέτρων κατά το ξεκίνημα του προγράμματος, έτσι ώστε να μπορεί να διαλέξει το μοντέλο ανίχνευσης που θα χρησιμοποιηθεί και να καθορίσει το όριο σκορ ανίχνευσης που χρησιμοποιείται για την απόφαση αντικειμένου.

Στη συνέχεια, ο κώδικας ανοίγει την κάμερα, διαβάζει συνεχώς τα καρέ της κάμερας, επεξεργάζεται κάθε καρέ με το μοντέλο ανίχνευσης και εμφανίζει τα αποτελέσματα στην οθόνη. Η ανίχνευση αντικειμένων γίνεται σε πραγματικό χρόνο.

Αυτή η συνάρτηση παίρνει μια εικόνα σε μορφή OpenCV (cv2_im), μέγεθος εισόδου (inference_size), αντικείμενα (objs) και ετικέτες (labels) ως είσοδο και προσθέτει πλαίσια γύρω από τα αντικείμενα και ετικέτες πάνω τους. Αντί για την αρχική ανάλυση της εικόνας, οι ανιχνευτές αντικειμένων συνήθως λειτουργούν σε μια μειωμένη ανάλυση (inference_size) για να βελτιώσουν την ταχύτητα της ανίχνευσης. Επομένως, η συνάρτηση αυτή υπολογίζει το κλιμάκωση x και y των αντικειμένων στην αρχική ανάλυση της εικόνας και στη συνέχεια τα αντιστοιχεί στο αρχικό μέγεθος της εικόνας. Στη συνέχεια, για κάθε αντικείμενο στη λίστα των αντικειμένων (objs), υπολογίζει τις συντεταγμένες του πλαισίου (bbox), προσθέτει το πλαίσιο στην εικόνα (cv2.rectangle), και προσθέτει την ετικέτα του αντικειμένου πάνω στο πλαίσιο (cv2.putText). Τέλος, επιστρέφει την εικόνα με τα πλαίσια και τις ετικέτες.

Κεφάλαιο 4

4.1 Πρίληψη Εφαρμογής

Η παρακάτω εφαρμογή κάνει χρήση του Edge TPU για να τρέξει μια παραθυριακή εφαρμογή αναγνώρισης αντικειμένων από ένα ήδη προεκπαιδευμένο μοντέλο της Google. Μπορούν να αναγνωριστούν άνθρωποι και αντικείμενα όπως ποδήλατα, κινητά τηλέφωνα, ψυγεία, πιρούνια, βιβλία, μολύβια. Ο TPU επεξεργάζεται κατά μέσο όρο 70-80 frames ανά δευτερόλεπτο. Παρόμοια εφαρμογή αποδίδει 10-30 frames ανά δευτερόλεπτο αντίστοιχα στον επεξεργαστή του Raspberry pi και του λάπτοπ

4.2 Κώδικας Εφαρμογής

```python
import argparse
import cv2
import os

from pycoral.adapters.common import input_size
from pycoral.adapters.detect import get_objects
from pycoral.utils.dataset import read_label_file
from pycoral.utils.edgetpu import make_interpreter
from pycoral.utils.edgetpu import run_inference


def main():
    #define os path
    default_model_dir = '../all_models'
    #trained models
    default_model = 'mobilenet_ssd_v2_coco_quant_postprocess_edgetpu.tflite'
    #labels
    default_labels = 'coco_labels.txt'

    parser = argparse.ArgumentParser()
    parser.add_argument('--model', help='.tflite model path',
                        default=os.path.join(default_model_dir,default_model))
    parser.add_argument('--labels', help='label file path',
                        default=os.path.join(default_model_dir, default_labels))
    parser.add_argument('--top_k', type=int, default=3,
                        help='number of categories with highest score to display')
    parser.add_argument('--camera_idx', type=int, help='Index of which video source to use. ', default = 0)
    parser.add_argument('--threshold', type=float, default=0.1,
                        help='classifier score threshold')
    args = parser.parse_args()

    print('Loading {} with {} labels.'.format(args.model, args.labels))
    interpreter = make_interpreter(args.model)
    interpreter.allocate_tensors()
    labels = read_label_file(args.labels)
    inference_size = input_size(interpreter)
    #enable camera driver
    cap = cv2.VideoCapture(args.camera_idx)

```

```
40
41      #Check if camera is enabled, then create windows with live output
42      while cap.isOpened():
43          ret, frame = cap.read()
44          if not ret:
45              break
46          cv2_im = frame
47
48          cv2_im_rgb = cv2.cvtColor(cv2_im, cv2.COLOR_BGR2RGB)
49          cv2_im_rgb = cv2.resize(cv2_im_rgb, inference_size)
50          run_inference(interpreter, cv2_im_rgb.tobytes())
51          objs = get_objects(interpreter, args.threshold)[:args.top_k]
52          cv2_im = append_objs_to_img(cv2_im, inference_size, objs, labels)
53
54          cv2.imshow('frame', cv2_im)
55          if cv2.waitKey(1) & 0xFF == ord('q'):
56              break
57
58      cap.release()
59      cv2.destroyAllWindows()
60
61  def append_objs_to_img(cv2_im, inference_size, objs, labels):
62      height, width, channels = cv2_im.shape
63      scale_x, scale_y = width / inference_size[0], height / inference_size[1]
64      for obj in objs:
65          bbox = obj.bbox.scale(scale_x, scale_y)
66
67          #define statistics
68          x0, y0 = int(bbox.xmin), int(bbox.ymin)
69          x1, y1 = int(bbox.xmax), int(bbox.ymax)
70          #print recognition score
71          percent = int(100 * obj.score)
72          #import object from txt
73          label = '{}% {}'.format(percent, labels.get(obj.id, obj.id))
74
75          #Define object recognition shape and colors
76          cv2_im = cv2.rectangle(cv2_im, (x0, y0), (x1, y1), (0, 0, 0), 2)
77          #Text from labels.txt output on cv2 image
78          cv2_im = cv2.putText(cv2_im, label, (x0, y0+30),
79                               cv2.FONT_HERSHEY_SIMPLEX, 1.0, (255, 0, 0), 2)
80      return cv2_im
81
82  if __name__ == '__main__':
83      main()
```

Βιβλιογραφία

1. https://ai.googleblog.com/2019/11/introducing-next-generation-on-device.html
2. https://en.wikipedia.org/wiki/Tensor_Processing_Unit
3. https://en.wikipedia.org/wiki/Application-specific_integrated_circuit
4. https://en.wikipedia.org/wiki/AI_accelerator
5. https://www.techradar.com/news/computing-components/processors/google-s-tensor-processing-unit-explained-this-is-what-the-future-of-computing-looks-like-1326915
6. https://cloud.google.com/tpu/docs/system-architecture-tpu-vm?hl=en
7. https://coral.ai/docs/accelerator/datasheet/
8. https://dspace.lib.ntua.gr/xmlui/handle/123456789/55937
9. https://coral.ai/docs/m2/datasheet/
10. https://el.wikipedia.org/wiki/Tensor_Flow
11. https://google.github.io/flatbuffers/
12. https://arxiv.org/pdf/2102.10423.pdf
13. https://towardsdatascience.com/a-basic-introduction-to-tensorflow-lite-59e480c57292
14. https://github.com/jiteshsaini/coral_USB_ml_accelerator
15. https://helloworld.co.in/article/image-classification-tensorflow-lite
16. https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi
17. https://proceedings.mlsys.org/paper/2022/file/31fefc0e570cb3860f2a6d4b38c6490d-Paper.pdf