Δ Ι Ε Θ Ν Ε Σ Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Τ Η Σ Ε Λ Λ Α Δ Ο Σ
Τ Μ Η Μ Α Μ Η Χ Α Ν Ι Κ Ω Ν Π Λ Η Ρ Ο Φ Ο Ρ Ι Κ Η Σ ,
Υ Π Ο Λ Ο Γ Ι Σ Τ Ω Ν Κ Α Ι Τ Η Λ Ε Π Ι Κ Ο Ι Ν Ω Ν Ι Ω Ν

Π ρ ό γ ρ α μ μ α Μ ε τ α π τ υ χ ι α κ ώ ν Σ π ο υ δ ώ ν σ τ η ν Ρ ο μ π ο τ ι κ ή

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΚΑΙ ΕΚΠΑΙΔΕΥΣΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΦΑΝΑΡΙΩΝ ΣΕ ΔΙΑΣΤΑΥΡΩΣΗ ΜΕ ΧΡΗΣΗ ΜΗΧΑΝΙΚΗΣ ΟΡΑΣΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΙΚΗΣ ΕΥΦΥΪΑΣ

Διπλωματική Εργασία του

Χριστόδουλου Στόικου (26)

Επιβλέπων: Στ. Βολογιαννίδης, Επίκουρος Καθηγητής

ΣΕΡΡΕΣ, ΜΑΡΤΙΟΣ 2022

Υπεύθυνη Δήλωση *: Βεβαιώνω ότι είμαι συγγραφέας αυτής της διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην διπλωματική εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος μεταπτυχιακών σπουδών της Ρομποτικής του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδας.*

1. Περίληψη

Η παρούσα εργασία έχει ως σκοπό την μελέτη πιθανής λύσης του προβλήματος της άσκοπης κίνησης που δημιουργείται στις διασταυρώσεις λόγω της αδυναμίας κατανόησης της επικρατούσας κατάστασης από το σύστημα διαχείρισης της κυκλοφορίας σε πραγματικό χρόνο. Η αδυναμία αυτή έχει ως αποτέλεσμα την κατασπατάληση πόρων και χρόνου των χρηστών του συστήματος, την περιβαλλοντική επιβάρυνση και την μειωμένη ασφάλεια. Η λύση που θα προταθεί θα αποτελείται από την κάμερα που θα μεταδίδει την ζωντανή εικόνα, το υλικό που θα εκτελούνται τα προγράμματα και τα λογισμικά που θα πραγματοποιούν την επεξεργασία της ζωντανής εικόνας και την εξαγωγή συμπερασμάτων.

Αρχικά θα παρουσιαστούν οι βιβλιοθήκες μηχανικής όρασης για την επεξεργασία της εικόνας, τα λογισμικά των νευρωνικών δικτύων που θα εξάγουν τις πληροφορίες από την εικόνα και οι πλατφόρμες που θα εκτελεστεί η επεξεργασία και θα αναφερθούν οι λόγοι επιλογής της βέλτιστης λύσης των Raspberry Pi και Intel Neural Compute Stick 2.

Στην συνέχεια, θα γίνει παρουσίαση των βημάτων εγκατάστασης και εκτέλεσης του OpenVINO σε Windows, της λήψης έτοιμων νευρωνικών δικτύων και της μετατροπής για το OpenVINO και θα δοκιμαστούν οι διάφορες επιλογές εκτέλεσης σε CPU, GPU και VPU.

Στο επόμενο βήμα θα γίνει η εγκατάσταση, η δοκιμή και η επίλυση των προβλημάτων εκτέλεσης του OpenVINO στο Raspberry Pi και η δοκιμή καλής λειτουργίας του Neural Compute Stick 2.

Στο τελευταίο κομμάτι θα δοκιμαστούν προτεινόμενα νευρωνικά δίκτυα, θα παρουσιαστούν τα αποτελέσματα και θα εξαχθούν τα συμπεράσματα για την επιλογή της βέλτιστης λύσης και τις βελτιώσεις που θα απαιτηθούν.

2. Summary

This thesis was created to tackle the traffic congestion problem that is caused by the inability of the current traffic management systems to process real time data of the traffic. This weakness of the system leads to great loss of recourses and time for the users of the traffic infrastructure. Moreover, it adds to environmental pollution and reduced safety. The proposed solution will be composed from a camera to feed the live image, the software to process the image and execute inferencing and the appropriate hardware to host the above.

At first the libraries for computer vision will be introduced, as well as the software dependencies for the neural networks that are responsible for the inferencing and final output and the hardware in which the all the data will be processed. Also, the reasons for selecting the combination of Raspberry Pi and Intel Neural Compute Stick 2 as the optimal choice, will be presented.

Next, the steps for installing in Windows will be presented, the testing and troubleshooting process for OpenVINO will be documented, as well as the download and conversion of neural networks and the various options of processing in CPU, GPU and VPU.

In the end, the suggested neural networks have been tested and their performance will be measured.

3. Περιεχόμενα

3.1 Πίνακας εικόνων

4. Εισαγωγή

Από την εποχή που δημιουργήθηκαν οι φωτεινοί σηματοδότες για την διαχείριση της κυκλοφορίας, δεν έχουν προχωρήσει ιδιαίτερα σε σχέση με τα άλματα της τεχνολογίας. Ο βασικός τρόπος ελέγχου είναι με προκαθορισμένους χρονοδιακόπτες, ενώ σε κάποιες διασταυρώσεις έχουν τοποθετηθεί αισθητήρες για την μείωση του χρόνου άσκοπης αναμονής, χωρίς να καλυτερεύουν την κατάσταση ιδιαίτερα.

Όλοι μας, κάποια στιγμή ή και καθημερινά, βρεθήκαμε σε μία αντίστοιχη κατάσταση, στην οποία σκεφτήκαμε ότι θα μπορούσε να υπάρξει μία πιο σύγχρονη και αποδοτικότερη λύση.

Η εκπόνηση της παρούσας διπλωματικής εργασίας έχει ως απώτερο σκοπό την παρουσίαση μίας λύσης η οποία θα συνδυάζει το χαμηλό κόστος κατασκευής, την χρήση υλικών και προγραμμάτων ευρείας παραγωγής και την βέλτιστη δυνατή απόδοση σε σχέση με το μέγεθος, την αξία και την κατανάλωση ενέργειας του συνόλου, ενώ θα είναι ανοιχτή σε επεκτασιμότητα και συνδεσιμότητα ενός ευρύτερου δικτύου με απόλυτο σκοπό την μείωση του χρόνου αναμονής των οχημάτων στις ρυθμιζόμενες διασταυρώσεις και κατά συνέπεια την μικρότερη σπατάλη χρόνου και επιβάρυνση του περιβάλλοντος από την άσκοπη λειτουργία των οχημάτων.

Η λύση αυτή θα χρησιμοποιεί την υπολογιστική όραση, δηλαδή θα λαμβάνει ζωντανή εικόνα του περιβάλλοντος χώρου μέσω κάμερας και με την χρήση νευρωνικών δικτύων θα αναλύει την κατάσταση που επικρατεί και θα την μετατρέπει σε δεδομένα.

Αναλυτικότερα, θα χρησιμοποιηθεί μία πλατφόρμα υλικού, πάνω στην οποία θα συνδέεται η κάμερα για την λήψη των ζωντανών εικόνων. Σε αυτήν, θα τρέχει μία βιβλιοθήκη που θα μπορεί να εισάγει τις εικόνες στο κυρίως πρόγραμμα. Στην συνέχεια, οι εικόνες θα αναλύονται είτε στην ίδια την πλατφόρμα, είτε σε εξωτερική συσκευή και με την χρήση νευρωνικών δικτύων θα εξάγονται αποτελέσματα σχετικά με τα οχήματα που βρίσκονται παρόντα κάθε χρονική στιγμή.

Θα πραγματοποιηθούν δοκιμές για την εύρεση των κατάλληλων υλικών και προγραμμάτων και θα αναφερθούν προοπτικές εξέλιξης του συστήματος.

5. State of the Art

Το πρόβλημα των απλών φωτεινών σηματοδοτών κυκλοφορίας είναι παγκόσμιο, οπότε υπάρχουν εταιρείες που προσπαθούν να το επιλύσουν σε διάφορες χώρες του κόσμου. Οι περισσότερες δεν αποτελούν τοπικές λύσεις για κάθε διασταύρωση, αλλά μέρος ενός πολύπλοκου και γενικού συστήματος που συλλέγει, αναλύει και αποφασίζει στο cloud. Γενικά συστήματα προσφέρονται από εταιρείες όπως η IBI, η Somos, η Alibaba, με την συλλογή των δεδομένων να γίνεται σε γενικό επίπεδο όπως από κάμερες κυκλοφορίας που δεν βρίσκονται μόνο σε διασταυρώσεις και από τα κινητά των χρηστών των κυκλοφοριακών υποδομών.

- IBIGROUP [1]
- SOMOS [2]
- Alibaba City Brain [3]
- Autoweek – Smart traffic lights [4]

Παρακάτω αναφέρονται κάποιες αξιόλογες προσπάθειες τοπικών συστημάτων.

5.1 Notraffic

Η εταιρεία NoTraffic ιδρύθηκε το 2017. Έχει έδρα στο Ισραήλ και γραφεία στην Καλιφόρνια. Το σύστημα αποτελείται από AI αισθητήρες, μηχανές βελτιστοποίησης και το κεντρικό σύστημα διαχείρισης. Σε κάθε διασταύρωση τοποθετούνται 4 αισθητήρες AI, ένας για κάθε κατεύθυνση, οι οποίοι αποτελούνται από κάμερες και ραντάρ, ενώ χρησιμοποιούν της πλατφόρμα NVIDIA Jetson σε συνδυασμό με GPU για την εκτέλεση των αλγορίθμων τεχνητής νοημοσύνης ανίχνευσης και κατηγοριοποίησης των χρηστών του δρόμου. Επιπλέον, έχουν εξοπλισμό συνδεσιμότητας με οχήματα, υποδομές και άλλες συσκευές που επικοινωνούν με τα πρωτόκολλα DSRC (Dedicated Short Range Communications) και CV2X (Cellular Vehicle To Everything).

Οι AI αισθητήρες συνδέονται τοπικά σε μία μηχανή βελτιστοποίησης (optimization engine) για την εκτέλεση των αλγορίθμων των διαδικασιών λειτουργίας των συνδεδεμένων σηματοδοτών και των δικλείδων ασφαλείας για την αποφυγή ατυχημάτων, καθώς και την βελτιστοποίηση της λειτουργίας της εκάστοτε διασταύρωσης. Επίσης, λειτουργεί και ως σταθμός επικοινωνίας με το κεντρικό σύστημα διαχείρισης για την αποστολή δεδομένων κυκλοφορίας και την βελτιστοποίηση της λειτουργίας σε μεγαλύτερο επίπεδο.

- NoTraffic [5]
- Nvidia about NoTraffic [6]
- Techcrunch – NoTraffic [7]

5.2 Surtrac

Το σύστημα Surtrac (Scalable Urban Traffic Control) είναι προϊόν της εταιρείας Rapid Flow Technologies. Αποτελείται από τον εξοπλισμό ανάλυσης βίντεο και ανίχνευσης οχημάτων, τον κεντρικό εγκέφαλο, τον ελεγκτή των σηματοδοτών και τον εξοπλισμό δικτύωσης.

Ο εξοπλισμός ανάλυσης βίντεο δέχεται ζωντανή εικόνα από απλές κάμερες κυκλοφορίας ή ασφαλείας, εκτελεί αλγόριθμους ανίχνευσης οχημάτων και μεταδίδει τα δεδομένα στον τοπικό εγκέφαλο. Στην συνέχεια, ο τοπικός εγκέφαλος λαμβάνει τα δεδομένα και από άλλους αισθητήρες που μπορεί να είναι εγκατεστημένοι στην διασταύρωση, όπως ραντάρ και επαγωγικούς βρόχους, και εκτελεί το λογισμικό που είναι υπεύθυνο για τον προγραμματισμό της διασταύρωσης. Ο ελεγκτής των σηματοδοτών εκτελεί τις αλλαγές στους σηματοδότες. Ο εξοπλισμός δικτύωσης διασυνδέει το εκάστοτε τοπικό σύστημα με τα υπόλοιπα.

- Surtrac [8]
- Wikipedia – Surtrac [9]

6. Επιλογή λύσεων

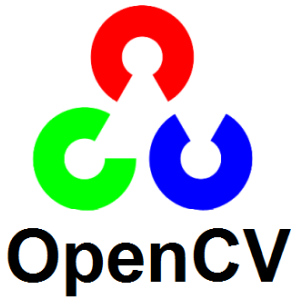
6.1 Επιλογή βιβλιοθήκης ανάπτυξης μηχανικής όρασης

Η βιβλιοθήκη ανάπτυξης μηχανικής όρασης είναι το σύνολο των εντολών, χάρη στο οποίο γίνεται η μετατροπή της εικόνας που λαμβάνουμε από την κάμερα, σε μορφή κατανοητή από το πρόγραμμα. Μέσα σε αυτήν περιλαμβάνονται συνήθεις πράξεις με τα δεδομένα εικόνας, απλές ή πολύπλοκες, και κύριο σκοπό έχει την παροχή μίας ενιαίας δομής και την απλούστευση της ενσωμάτωσής της σε εφαρμογές. Ανάλογα την βιβλιοθήκη, υπάρχει πιθανότητα οι δυνατότητές της να διευρύνονται καλύπτοντας και άλλα αντικείμενα.

Επιλαχούσες βιβλιοθήκες

1) OpenCV
2) BoofCV
3) SimpleCV
4) SOD

6.1.1 OpenCV (Open Computer Vision)



Η OpenCV είναι ανοιχτού κώδικα βιβλιοθήκη. Είναι η πιο παλιά και η πιο διαδεδομένη βιβλιοθήκη με κύριο στόχο την υπολογιστική όραση σε πραγματικό χρόνο. Υποστηρίζει διάφορες γλώσσες προγραμματισμού, όπως Python, C++, Java. Η OpenCV είναι cross-platform και υποστηρίζει Microsoft Windows, Linux και άλλα βασισμένα σε Unix, ενώ εκτελείται και σε φορητά λειτουργικά όπως Android, iOS, Maemo και Blackberry. Η διανομή και χρήσης της συνοδεύεται από άδεια BSD-3 Clause.

- Wikipedia – OpenCV [10]
- OpenCV [11]
- BSD license [12]

6.1.2 BoofCV

Η BoofCV είναι ανοιχτού κώδικα βιβλιοθήκη. Αναπτύχθηκε από τον Peter Abeles, με κύριο στόχο την υπολογιστική όραση σε πραγματικό χρόνο και τις σχετικές ρομποτικές εφαρμογές. Είναι γραμμένη σε Java, ενώ υπάρχει και wrapper για Python. Είναι οργανωμένη σε πακέτα, για την ευκολότερη χρήση της. Περιλαμβάνει λειτουργίες και χαμηλού και υψηλού επιπέδου, όπως επεξεργασία εικόνας, βαθμονόμηση κάμερας, εξαγωγή χαρακτηριστικών και αναγνώριση. Η διανομή και η χρήση της συνοδεύεται από άδεια Apache 2.0 για εμπορική και ακαδημαϊκή χρήση.

- BoofCV [13]
- Apache 2.0 [14]

6.1.3 SimpleCV

Η SimpleCV αποτελεί framework για την συγγραφή κώδικα υπολογιστικής όρασης σε Python. Χρησιμοποιεί βιβλιοθήκες όπως η OpenCV, απλοποιώντας την χρήση τους, καθώς δεν απαιτεί εξειδικευμένες γνώσεις επεξεργασίας εικόνας. Ως αποτέλεσμα, είναι ικανοποιητική για χρήστες με ελάχιστες γνώσεις πάνω στην επεξεργασία εικόνας. Δυστυχώς, η ιστοσελίδα, το GitHub και το blog δεν έχουν αναβαθμιστεί από το 2015, ενώ το forum δεν λειτουργεί. Πιθανότατα πρόκειται για νεκρό project.

- SimpleCV [15]
- Stackoverflow – SimpleCV [16]

6.1.4 SOD (Salient Object Detection)

Η SOD είναι ανοιχτού κώδικα βιβλιοθήκη και είναι ενσωματωμένη στις υπηρεσίες της εταιρείας pixlab, ενώ αναπτύχθηκε από την Symisc Systems. Είναι γραμμένη σε C++, αλλά υπάρχει και έκδοση για Python. Η SOD είναι cross-platform και δημιουργήθηκε για υπολογιστική όραση και μηχανική μάθηση, ενώ παρέχει και APIs για deep learning, προχωρημένη ανάλυση και επεξεργασία πολυμέσων σε πραγματικό χρόνο και αναγνώριση αντικειμένων διαφόρων κλάσεων. Κύρια ομάδα συσκευών χρήσης είναι όσες έχουν περιορισμένους υπολογιστικούς πόρους, αλλά και IoT συσκευές. Διανέμεται με άδεια GPLv3, ενώ υπάρχει και δυνατότητα απόκτησης άδειας μη αποκλειστικής εμπορικής χρήσης.

- Pixlab [17]

6.2 AI Frameworks

Τα frameworks στην τεχνητή νοημοσύνη (Deep Learning ή DL) είναι πακέτα λογισμικού, τα οποία προσφέρουν μία διεπαφή υψηλού επιπέδου (API) για τον σχεδιασμό, την εκπαίδευση, την επικύρωση σωστής λειτουργίας και την αποσφαλμάτωση των νευρωνικών δικτύων. Στην παρούσα εργασία θα αναφερθούν γνωστά frameworks, αλλά θα γίνει χρήση κυρίως του TensorFlow.

Frameworks

1) TensorFlow
2) Theano
3) Caffe
4) Keras
5) PyTorch

6.2.1 TensorFlow (TF)

Ο TensorFlow είναι το πιο γνωστό framework. Έχει ενσωματωμένη βιβλιοθήκη για προγραμματισμό ροής (dataflow) και διαφορετικότητας (differentiate). Λόγω της ευελιξίας του κατά την εκτέλεση, χρησιμοποιείται στην Τεχνητή Νοημοσύνη (AI) και σε όλα τα πεδία που καλύπτει αυτή. Υποστηρίζει πλήρως Python και C, ενώ λειτουργεί κανονικά και σε άλλες γλώσσες προγραμματισμού όπως C++, Java και Go. Με την χρήση πακέτων γραμμένων σε C, υποστηρίζει αρκετά περισσότερες γλώσσες. Είναι cross-platform και υποστηρίζει Microsoft Windows και Linux, ενώ εκτελείται και σε φορητά λειτουργικά όπως τα Android και iOS. Αναπτύχθηκε από την Google και διανέμεται με άδεια Apache 2.0. Σημαντικό προτέρημα της υποστήριξης από την Google είναι η άμεση χρήση του στο Google Colab.

- Wikipedia – TensorFlow [18]
- TensorFlow [19]
- Answerrocket – AI libraries [20]

6.2.2 Theano

Η Theano πήρε το όνομά της από την αρχαία Ελληνίδα φιλόσοφο Θεανώ, στην οποία αποδίδεται ο όρος “χρυσή τομή”. Δεν είναι framework, αλλά βιβλιοθήκη γραμμένη σε Python και CUDA με σκοπό την εκτέλεση μαθηματικών πράξεων και κυρίως πράξεις πινάκων, με αποδοτικότητα σε CPU και GPU. Αυτό επιτυγχάνεται χάρη στον ενσωματωμένο compiler για την μετατροπή του κώδικα σε πράξεις πινάκων και την χρήση της NumPy. Εκτελείται σε Microsoft Windows, Linux και MacOS. Διανέμεται με άδεια χρήσης BSD-3 Clause. Αναπτύχθηκε από τους Pascal Lamblin και Yoshua Bengio, καθηγητές στο πανεπιστήμιο του Montreal, στο MILA (Montreal Institute for Learning Algorithms). Δυστυχώς, ο ανταγωνισμός από τις πολυεθνικές που ασχολήθηκαν με την ανάπτυξη αντίστοιχου λογισμικού και οι ελλιπείς πόροι, οδήγησαν στον τερματισμό της υποστήριξης και της ανάπτυξης από την αρχική ομάδα. Όμως, από το 2021 δημιουργήθηκε διακλάδωση (fork) με την ονομασία Aesara.

- Wikipedia – Theano [21]
- MLM – Theano [22]
- GitHub – Aesara [23]

6.2.3 Caffe (Convolutional Architecture for Fast Feature Embedding)

Το framework Caffe απευθύνεται σε εφαρμογές deep learning. Σχεδιάστηκε και υλοποιήθηκε με γνώμονα την ταχύτητα εκτέλεσης, την κατάτμηση του κώδικα και την ευκολία έκφρασης. Η κύρια χρήση του ήταν στις εφαρμογές κατηγοριοποίησης και κατάτμησης εικόνων. Αναπτύχθηκε από το τμήμα έρευνας τεχνητής νοημοσύνης του πανεπιστημίου Berkeley (BAIR). Είναι γραμμένο σε γλώσσα C++, ενώ με χρήση διεπαφής λειτουργεί και σε Python. Διανέμεται με άδειας χρήσης BSD-2 Clause. Τον Απρίλιο του 2017 βγήκε η έκδοση Caffe2, ενώ τον Μάρτιο του 2018 ενσωματώθηκε στον PyTorch.

- BAIR Berkeley [24]
- Berkeleyvision – Caffe [25]
- GitHub – Caffe [26]

6.2.4 Keras

Ο Keras είναι πλαίσιο – βιβλιοθήκη ανοιχτού κώδικα για νευρωνικά δίκτυα. Είναι γραμμένος σε Python. Μέχρι την έκδοση 2.3, χρησιμοποιούνταν ως διεπαφή για διάφορα frameworks, όπως TensorFlow, Theano, CNTK. Από την έκδοση 2.4, υποστηρίζει μόνο τον TensorFlow. Σχεδιάστηκε με γνώμονα την φιλικότητα προς τον χρήστη, την κατάτμηση του κώδικα και την επεκτασιμότητα. Αυτό έχει ως αποτέλεσμα την δυνατότητα κατάτμησης των εργασιών και την εκτέλεση σε συμπλέγματα GPUs και TPUs.

- Keras [27]
- Wikipedia – Keras [28]

6.2.5 PyTorch

Το framework PyTorch είναι βασισμένο στην βιβλιοθήκη Torch. Είναι γραμμένο σε Python και χρησιμοποιείται για εφαρμογές υπολογιστικής όρασης (CV) και επεξεργασίας φυσικής γλώσσας (NLP – Natural Language Processing). Αν και είναι γραμμένο και βελτιστοποιημένο για Python, υποστηρίζει και C, C++ και CUDA. Αναπτύχθηκε από το εργαστήριο έρευνας τεχνητής νοημοσύνης του Facebook το 2016. Το 2018 ενσωματώθηκε στο framework Caffe2. Διανέμεται με τροποποιημένη άδεια BSD.

- Wikipedia – PyTorch [29]
- Quora – PyTorch [30]
- PyTorch [31]
- Wikipedia – BSD license [32]

6.3 AI accelerators / Tensor PUs / Vision PUs / Neural PUs

Για την εκτέλεση εφαρμογών τεχνητής νοημοσύνης, μπορούν να χρησιμοποιηθούν οι CPU, αλλά προσφέροντας την χαμηλότερη επιτάχυνση για την ενέργεια που καταναλώνουν.

Αντιθέτως, οι GPU προσφέρουν αρκετά ταχύτερη εκτέλεση εφαρμογών τεχνητής νοημοσύνης, καθώς τα γραφικά, για τα οποία είναι σχεδιασμένα, είναι σύνολο πινάκων. Έτσι, όπως και σε πολλά είδη τεχνητής νοημοσύνης, οι πράξεις των πινάκων γίνονται με μεγαλύτερη ταχύτητα για την ενέργεια που καταναλώνουν.

Επίσης, καθώς τα FPGAs μπορούν να προγραμματιστούν ως υλικό για συγκεκριμένη χρήση, χρησιμοποιούνται για την εκτέλεση εφαρμογών τεχνητής νοημοσύνης με αρκετά καλή ταχύτητα για την ενέργεια που καταναλώνουν.

Παρότι οι GPU και τα FPGA είναι ταχύτερα από τις CPU, δεν μπορούν να ξεπεράσουν υλικό το οποίο είναι σχεδιασμένο για ειδικές περιπτώσεις. Το υλικό αυτό είναι οι επιταχυντές τεχνητής νοημοσύνης το οποίο είναι υλικό ειδικά σχεδιασμένο για την ταχύτερη εκτέλεση εφαρμογών τεχνητής νοημοσύνης, μηχανικής μάθησης, νευρωνικών δικτύων και ρομποτικής όρασης. Η ταχύτητα εκτέλεσης για την ενέργεια που καταναλώνουν είναι μακράν καλύτερη από τα υπάρχοντα είδη υλικού.

Μέχρι στιγμής δεν υπάρχει κάποια σχεδίαση που να έχει κυριαρχήσει, όπως στις CPU, αλλά κάθε κατασκευαστής δίνει την δική του λύση, όπως και στην ορολογία που ακολουθείται.

Έτσι, το υλικό που σχετίζεται με την τεχνητή νοημοσύνη αναφέρεται με τους εξής τρόπους :

1) AI Accelerator
2) TPU – Tensor Processing Unit
3) VPU – Vision Processing Unit
4) NPU – Neural Processing Unit

Ο κυριότερος τρόπος κατάταξης είναι η επεξεργαστική ισχύς, η οποία μετριέται σε τρισεκατομμύρια πράξεις ανά δευτερόλεπτο (TOPS – Trillion Operations Per Second), ενώ η απόδοσή τους σε TOPS ανά Watt (TOPS/W).

Πλατφόρμες υλικού AI

1) Coral Edge USB accelerator
2) Intel Neural Compute Stick 2
3) NVIDIA Jetson Nano
4) Rock Pi N10

6.3.1 Coral Edge USB accelerator

Το Coral Edge USB accelerator είναι συσκευή USB, η οποία απαιτεί την σύνδεση σε μία κεντρική συσκευή για την επιτάχυνση εκτέλεσης εφαρμογών τεχνητής νοημοσύνης. Αναπτύχθηκε από την Google και αποτελεί μέρος του οικοσυστήματος Coral. Χρησιμοποιείται για την επιτάχυνση εκτέλεσης αλγορίθμων μηχανικής ευφυΐας σε συσκευές χαμηλής κατανάλωσης (on the edge). Υποστηρίζει Microsoft Windows 10, macOS, καθώς και όλες τις διανομές Linux που είναι βασισμένες στο Debian. Ως εκ τούτου, υποστηρίζει το Raspberry Pi. Υποστηρίζει μόνο τον TensorFlow Lite. Έχει δυνατότητα 4 τρισεκατομμυρίων πράξεων ανά δευτερόλεπτο, με απόδοση 2 TOPS/W, δηλαδή έχει μέγιστη κατανάλωση 2 Watts.

Figure 1 - Coral Edge USB Accelerator

- Coral AI [33]

6.3.2 Intel Neural Compute Stick 2

Το Intel Neural Compute Stick 2 κατασκευάστηκε από την Intel. Όπως το Coral Edge USB, και το INCS2 αποτελεί συσκευή USB, η οποία απαιτεί την σύνδεση σε μία κεντρική συσκευή για την επιτάχυνση εκτέλεσης αλγορίθμων τεχνητής νοημοσύνης. Η πρώτη γενιά αναπτύχθηκε από την Movidius, η οποία εξαγοράστηκε από την Intel το 2016. Υποστηρίζει ευρεία γκάμα λειτουργικών όπως Windows 10, macOS, Ubuntu, CentOS, Raspbian, ενώ μέσω της εργαλειοθήκης OpenVINO υποστηρίζει περισσότερα. Παρέχει υποστήριξη σε δημοφιλή frameworks όπως TensorFlow, Caffe2, Apache MXNet, ONNX, PyTorch. Όπως και το Coral Edge USB, έχει δυνατότητα 4 τρισεκατομμυρίων πράξεων ανά δευτερόλεπτο, με απόδοση 2 TOPS/W, δηλαδή έχει μέγιστη κατανάλωση 2 Watts.

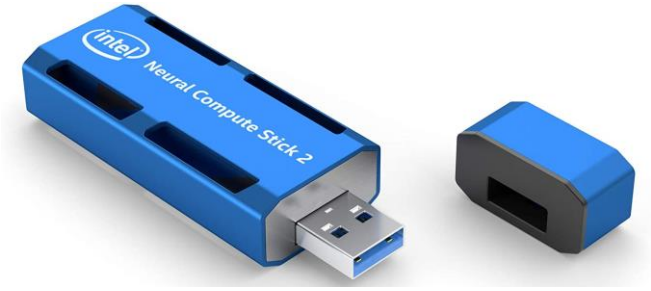


Figure 2 - Intel Neural Compute Stick 2

- Intel Neural Compute Stick 2 [34]
- Google coral vs Intel NCS2 [35]

6.3.3 NVIDIA Jetson Nano

Σε αντίθεση με τα USB AI accelerators, το Jetson Nano αποτελεί μία ολοκληρωμένη πλατφόρμα ανάπτυξης. Το λειτουργικό του σύστημα είναι το Linux4Tegra, το οποίο είναι βασισμένο σε Linux Ubuntu. Αποτελείται από μία 4πύρηνη CPU 64-bit ARM A57 και φέρει 4GB Ram, ενώ η βασική διαφορά του σε σχέση με άλλες πλατφόρμες ανάπτυξης αντίστοιχου μεγέθους, είναι ότι έχει ενσωματωμένη GPU 128 πυρήνων αρχιτεκτονικής Nvidia Maxwell, χάρη στην οποία μπορεί να εκτελέσει προγράμματα επεξεργασίας εικόνας, υπολογιστικής όρασης και τεχνητής νοημοσύνης. Η γλώσσα προγραμματισμού είναι η CUDA, η οποία προσφέρει την δυνατότητα προγραμματισμού, μέσω διεπαφής, σε άλλες γλώσσες όπως η C, C++ και η Fortran, ενώ μέσω wrappers μπορεί να προγραμματιστεί σε Python.

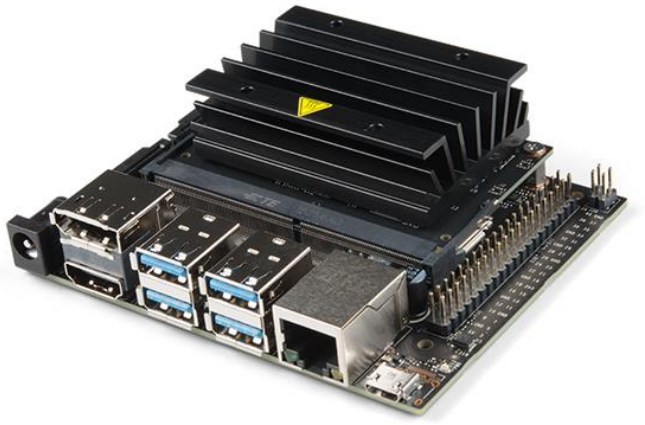


Figure 3 – Nvidia Jetson Nano

- The RPi of AI [36]
- Nvidia Jetson Nano and the other TPUs [37]
- Introduction to CUDA with Jetson Nano [38]

6.3.4 Rock Pi N10

Το Rock Pi N10 αναπτύχθηκε από την Radxa και είναι βασισμένο στο Raspberry Pi. Όπως και το Jetson Nano, αποτελεί και αυτό ολοκληρωμένη πλατφόρμα ανάπτυξης. Το λειτουργικό του σύστημα βασίζεται σε Debian Linux ή Android. Στα κύρια τεχνικά χαρακτηριστικά του περιλαμβάνονται μία διπύρηνη CPU Cortex-A72 και μία τετραπύρηνη CPU Cortex A53 και η Mali T860MP4 GPU, ενώ φέρει και ενσωματωμένη NPU με μέγιστη ικανότητα 3 TOPS. Αν και έχει αρκετά καλή και τεκμηριωμένη wiki, η κοινότητά του και η γενικότερη υποστήριξη είναι σχετικά μικρή.

Figure 4 – Radxa Rock Pi N10

- RockpiN10 specifications [39]
- Radxa Rock Pi N10 [40]

7. Επιλογή βέλτιστης λύσης

Η τελική επιλογή έγινε βάση των συγκρίσεων των παραπάνω υλικών και λογισμικών. Αφού καταγράφηκαν τα υπέρ και τα κατά του κάθε βήματος, όπως το κοστολόγιο, η ευκολία εκμάθησης, η μεγάλη κοινότητα, η μελλοντική ανάπτυξη και άλλα, βρήκα ότι βέλτιστη λύση αποτελεί ο συνδυασμός του Raspberry Pi και του Intel Neural Compute Stick 2. Ο συνδυασμός υπερέχει των άλλων λύσεων για τους παρακάτω λόγους :

1) Το RPi έχει την μεγαλύτερη κοινότητα στον κόσμο
2) Το OpenVINO εκτελείται και σε Raspbian (πλέον Raspberry Pi OS)
3) Το OpenVINO υποστηρίζει πληθώρα frameworks
4) Είναι η πιο πολλά υποσχόμενη συνδυαστική λύση
5) Μπορεί να προσφέρει γνώση στο ΠΜΣ Ρομποτικής

Αναλυτικότερα, το Raspberry Pi έχει την μεγαλύτερη κοινότητα στον κόσμο, καθώς είναι η πρώτη πλατφόρμα ανάπτυξης ευρείας αποδοχής, το οποίο προσφέρει μεγάλη υποστήριξη. Επίσης, το RPi έχει διάφορες εκδόσεις και άρα δεν περιορίζεται από 1 έκδοση υλικού με συγκεκριμένα τεχνικά χαρακτηριστικά

Το OpenVINO υποστηρίζει Raspberry Pi OS απευθείας από την ίδια την Intel, επειδή η ίδια έχει αναγνωρίσει την ευρεία εισροή του RPi σε κάθε είδους κοινότητα και αντικείμενο στον τομέα της ανάπτυξης πρωτοτύπων.

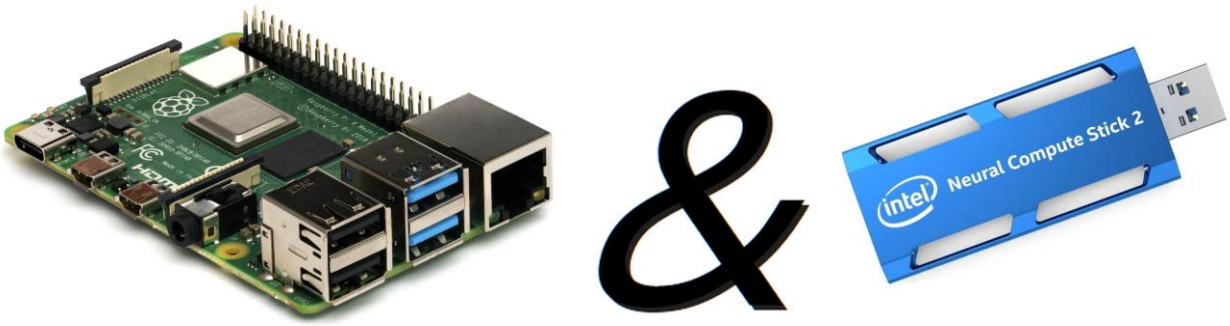


Figure 5 – Raspberry Pi 4 & Intel Neural Compute Stick 2

8. Τι είναι το OpenVINO

Το OpenVINO είναι ένα σύνολο εργαλείων της Intel για την ανάπτυξη και παραγωγή προσανατολισμένων στην μηχανική όραση λύσεων, οι οποίες θα βασίζουν την λειτουργία τους είτε στα προϊόντα της Intel, είτε στο cloud, είτε στην τελική συσκευή απευθείας (on the edge).

Χρησιμοποιείται ως ενδιάμεσος για την εξαγωγή των χαρακτηριστικών που ζητούνται από το εκάστοτε πρόγραμμα και ανάλογα την εφαρμογή, με την χρήση τεχνητής νοημοσύνης και νευρωνικών δικτύων, με σκοπό την ανάπτυξη ενιαία λύσης για όλες τις πλατφόρμες που υποστηρίζονται.

Επιλύει προβλήματα όπως η αναγνώριση αντικειμένων, η ταξινόμηση, η κατάτμηση με χρήση νευρωνικών δικτύων, η επεξεργασία εικόνας και η ρομποτική όραση.

Υποστηρίζει λογισμικά όπως τα Windows, τα Linux, αλλά και αρκετά hardware λογισμικά (κυρίως βασισμένα σε Linux). Οι συμβατές πλατφόρμες για την εκτέλεση είναι οι INTEL CPU, οι ενσωματωμένες GPU, οι επεξεργαστές όρασης (VPU) και τα FPGA. Μπορεί να γίνει εκτέλεση είτε σε κάποια πλατφόρμα ξεχωριστά, είτε σε συνδυασμό πολλών ενός ή και περισσότερων κατηγοριών.

Στο OpenVINO συμπεριλαμβάνονται διάφορα εργαλεία όπως το Deep Learning Deployment Toolkit (DLDT), τις βιβλιοθήκες για την εκτέλεση σε όλες τις πλατφόρμες που υποστηρίζει το OpenVINO, το Open Model Zoo που περιλαμβάνει εκπαιδευμένα νευρωνικά δίκτυα βελτιστοποιημένα για άμεση χρήση με το OpenVINO και εργαλεία για δοκιμή και αξιολόγηση των αποτελεσμάτων.

9. Υλοποίηση

9.1 Εγκατάσταση

Έχοντας συγκεντρώσει το απαραίτητο hardware, στην προκειμένη περίπτωση μία VPU INTEL, προχωρούμε στην λήψη του OpenVINO από την ιστοσελίδα της Intel επιλέγοντας το λειτουργικό μας σύστημα, την έκδοση και τον τρόπο εγκατάστασης.

- Intel OpenVINO overview [41]

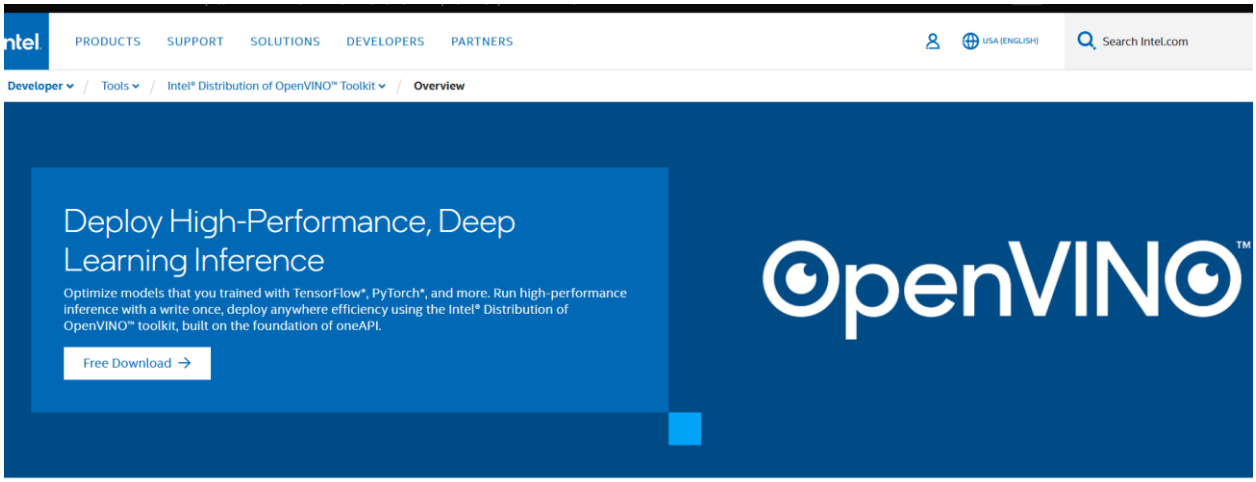


Figure 6 – OpenVINO κεντρική ιστοσελίδα

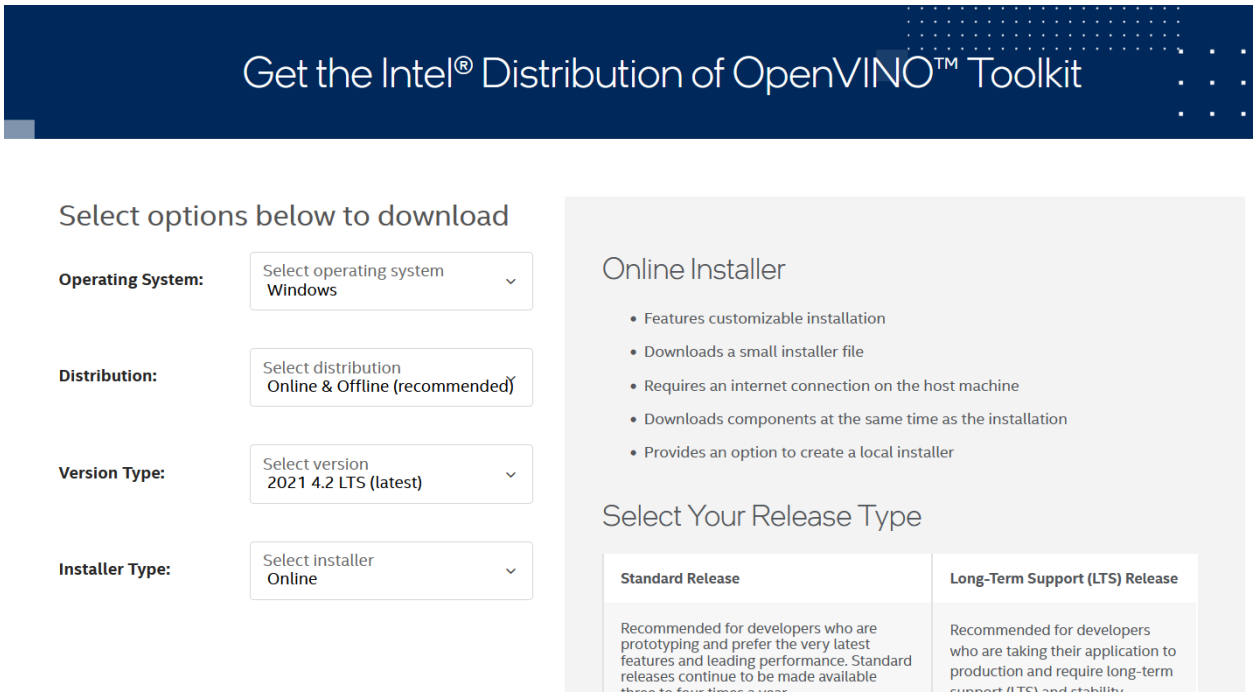


Figure 7 – OpenVINO ιστοσελίδα επιλογών λήψης

Στην συνέχεια, βάζουμε το email που θα χρησιμοποιήσουμε και την χώρα μας και το κατεβάζουμε.

Η βασική εγκατάσταση του OpenVINO περιλαμβάνει τα παρακάτω εργαλεία

Model Optimizer	Ο MO εισάγει, μετατρέπει και βελτιστοποιεί μοντέλα νευρωνικών δικτύων που έχουν προγραμματιστεί και εκπαιδευτεί σε δημοφιλή frameworks (TensorFlow, Caffe, MXNet), σε κατανοητή μορφή από τα υπόλοιπα εργαλεία του OpenVINO
Inference Engine	Το εργαλείο που εκτελεί την εξαγωγή συμπερασμάτων στις πλατφόρμες της Intel. Συμπεριλαμβάνει βιβλιοθήκες για απλή και εύκολη ενσωμάτωση στην ανάπτυξη εφαρμογών
OpenCV	Ανοιχτή βιβλιοθήκη υπολογιστικής όρασης βελτιστοποιημένη για το υλικό της Intel
Inference Engine samples	Παραδείγματα για την επίδειξη της χρήσης και λειτουργίας των διαφόρων λειτουργιών του OpenVINO (φόρτωση μοντέλων, εκτέλεση inferencing)
Demo applications	Ένα σύνολο εφαρμογών γραμμής εντολών που στόχο έχουν να χρησιμοποιηθούν ως πρότυπες για την ενσωμάτωση των εργαλείων του OpenVINO στην ανάπτυξη εφαρμογών
Additional tools	Ένα σύνολο εργαλείων για την χρήση των μοντέλων (μέτρηση ακρίβειας, βελτιστοποιητής έτοιμου μοντέλου, αυτόματο κατέβασμα μοντέλων)
Documentation για εκπαιδευμένα μοντέλα	Έγγραφη βοήθεια και επεξήγηση για τα προ-εκπαιδευμένα μοντέλα της Intel που βρίσκονται στο Model Zoo

- OpenVINO to Windows 10 [42]

Κατά την εγκατάσταση, αφήνουμε ότι έχει προεπιλεγμένο, για να γίνει πλήρης εγκατάσταση.

Στην συνέχεια, μας εμφανίζει διάφορες προειδοποιήσεις, οι οποίες δεν σταματούν την εγκατάσταση, αλλά ίσως δημιουργήσουν προβλήματα στην ομαλή λειτουργία του OpenVINO.

Στην συγκεκριμένη περίπτωση, οι προειδοποιήσεις για Intel CPU & GPU δεν μας απασχολούν διότι θα χρησιμοποιηθεί Intel VPU, ενώ η CMake θα γίνει εγκατάσταση αργότερα. Μετά από τα παραπάνω, ολοκληρώνεται η εγκατάσταση επιτυχώς.

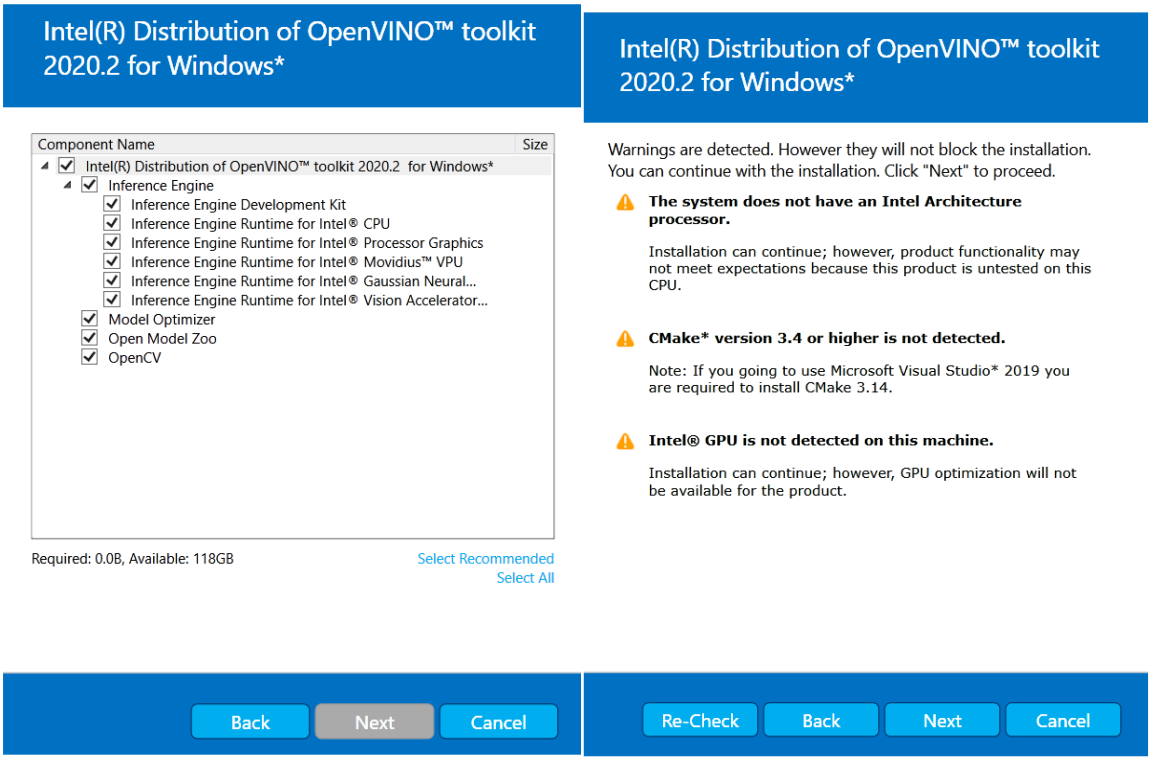


Figure 8 – OpenVINO προεπιλογές και προειδοποιήσεις εγκατάστασης

Κατά την χρονική περίοδο που εκπονήθηκε η παρούσα εργασία, το OpenVINO απαιτούσε έκδοση Python >= 3.6.5. Η χρήση της 3.7.7 κρίθηκε απαραίτητη λόγω της μη υποστήριξης νεότερων εκδόσεων από τον TensorFlow. Πλέον, ο TF απαιτεί εκδόσεις Python 3.7-3.9, και πιο συγκεκριμένα :

- TensorFlow 2.5 ή μεταγενέστερος απαιτεί Python 3.9
- TensorFlow 2.2 ή μεταγενέστερος απαιτεί Python 3.8

- TensorFlow requirements [43]

Επίσης, χρειάζεται το Visual Studio (η έκδοση community παρέχεται δωρεάν). Στην εγκατάστασή του, οι επιλογές που απαιτούνται για το OpenVINO, αφού επιλέξουμε εγκατάσταση Workloads – Desktop development with C++, είναι οι εξής:

1) Windows 10 SDK
2) Just-In-Time debugger

3) C++ profiling tools
4) C++ CMake tools for Windows
5) C++ ATL for latest v142 build tools
6) Test Adapter for Boost.Test
7) Test Adapter for Google Test
8) Live Share
9) C++ AddressSanitizer

Επιπροσθέτως, πρέπει να επιλεγούν μόνο τα ακόλουθα Individual components

1) C# and Visual Basic Roslyn compilers
2) C++ 2019 Redistributable Update
3) C++ CMake tools for Windows
4) MSBuild

Τελευταία απαραίτητη εγκατάσταση είναι το CMake. Για το VS2019 απαιτείται η έκδοση x64 CMake >= 3.14, την οποία μπορούμε να κατεβάσουμε από την ιστοσελίδα cmake.org.

Κατά την εγκατάσταση, προτείνεται η προσθήκη του CMake στο system Path, κάτι το οποίο θα μας γλιτώσει χρόνο από την χειροκίνητη προσθήκη αργότερα.

9.2 Αρχικές ρυθμίσεις

Μετά το πέρας των απαραίτητων εγκαταστάσεων, χρειάζεται να ρυθμιστούν οι παράμετροι. Προσωρινή ρύθμιση μπορεί να γίνει πηγαίνοντας στον φάκελο bin του OpenVINO (default C:\Program Files (x86)\IntelSWTools\openvino\bin\) και εκτελώντας από γραμμή εντολών το setupvars.bat.

Στην συνέχεια, πρέπει να ρυθμιστεί ο Βελτιστοποιητής Μοντέλων (Model Optimizer). Μέσω γραμμής εντολών, γίνεται η πλοήγηση στον φάκελο C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\model_optimizer\install_prerequisites και η εκτέλεση της εγκατάστασης των απαραίτητων αρχείων με την εντολή install_prerequisites.bat η οποία προχωράει σε εγκατάσταση για όλα τα frameworks. Στην παρούσα φάση θα χρησιμοποιηθεί μόνο TensorFlow , οπότε χρειάζεται η εκτέλεση της εντολή install_prerequisites_tf.bat.

Καθώς στο σύστημα είναι εγκατεστημένες δύο Python, η 3.8.1 και η 3.7.7, έγινε μετονομασία των αντίστοιχων εκτελέσιμων python.exe σε python381.exe και python377.exe. Αυτό σημαίνει ότι εμφανίζονται τα παρακάτω σφάλματα :

1) Μετονομασία του python.exe σε python377.exe

```
C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\model_optimizer\install_prerequisites>install_prerequisite
s_tf.bat
Python 3.7.7
ECHO is off.
Fatal error in launcher: Unable to create process using '"c:\python\python377\python377.exe"  "C:\Python\Python377\Scrip
ts\pip3.exe" install --user -r ..\requirements_tf.txt': The system cannot find the file specified.
```

2) Μετονομασία στο αρχικό python.exe

```
C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\model_optimizer\install_prerequisites>install_prerequisites_tf.bat
Error: Python is not installed. Please install Python 3.5 (64-bit) or higher from https://www.python.org/downloads/
```

Το πρόβλημα επιλύθηκε ανοίγοντας το αρχείο install_prerequisites.bat με Notepad++ και αλλάζοντας την εντολή python --version 2>NUL σε python377 --version 2>NUL.

```
15
16 :: Check if Python is installed
17 setlocal
18
19 python377 --version 2>NUL
20 if errorlevel 1 (
21    echo Error^: Python is not ins
22    goto error
```

Αφού ολοκληρωθεί η εγκατάσταση και των απαραίτητων προαπαιτούμενων, θα γίνει δοκιμή καλής λειτουργίας, εκτελώντας ένα από τα παραδείγματα του OpenVINO που προορίζονται για αυτό τον σκοπό.

9.3 1η δοκιμή

Πηγαίνοντας στον φάκελο demo (cd C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\demo\) εκτελούμε το αρχείο demo_squeezenet_download_convert_run.bat.

Στο σημείο αυτό υπάρχει πρόβλημα με τις αλλαγές των ονομασιών στα python.exe και εμφανίζονται πάλι errors (python377.exe δεν μπορεί να τρέξει τις εντολές, python.exe δεν βρίσκει εγκατεστημένη python). Έγινε διόρθωση όλων αφαιρώντας τελείως την 3.8.1 και κάνοντας repair την 3.7.7.

Εκτελώντας την παραπάνω εντολή για το demo, βγαίνουν τα εξής αποτελέσματα.

9.3.1 CPU only

Figure 9 – Εκτέλεση security barrier demo σε CPU

9.3.2 Intel NCS 2

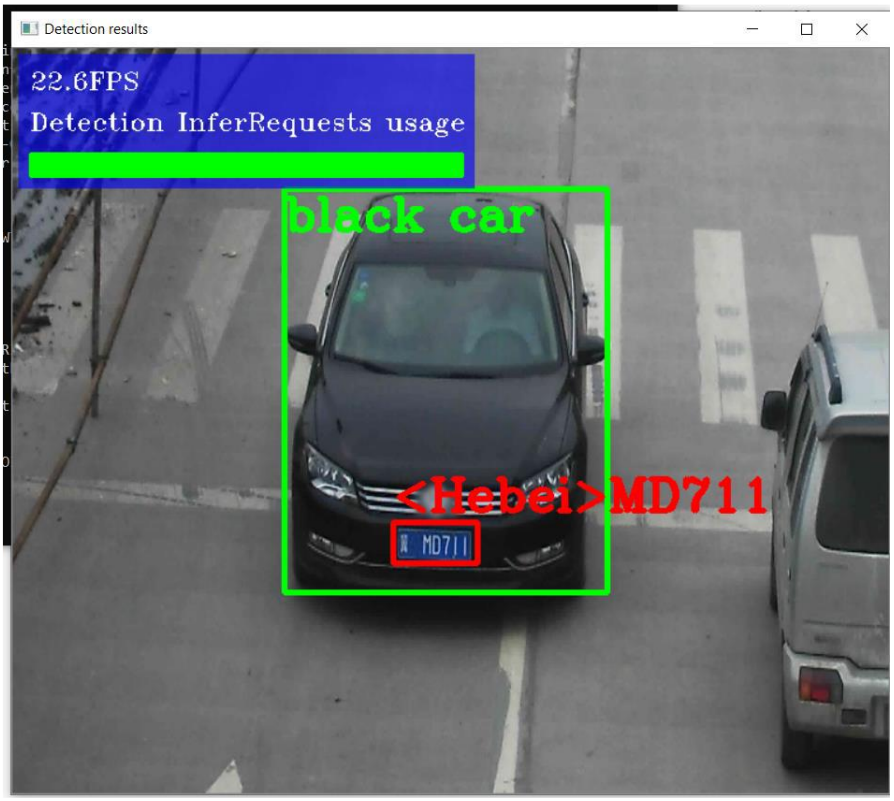


Figure 10 – Εκτέλεση security barrier demo στο NCS2 (MYRIAD)

Στο παραπάνω παράδειγμα, εκτελούνται τα εξής inferences:

1) Για την εύρεση του οχήματος
2) Για την αναγνώριση του είδους του οχήματος
3) Για την αναγνώριση του αριθμού εγγραφής της πινακίδας του οχήματος

Τα FPS υπολογίζονται με βάση τον χρόνο που χρειάστηκε για να ολοκληρωθούν τα 3 αιτήματα για Inferencing σε κάθε στιγμιότυπο. Η διαφορά δεν φαίνεται μεγάλη, εκτός αν λάβουμε υπόψιν μας ότι η CPU καταναλώνει περίπου 70 watts, ενώ το NCS2 λιγότερο από 2 watts.

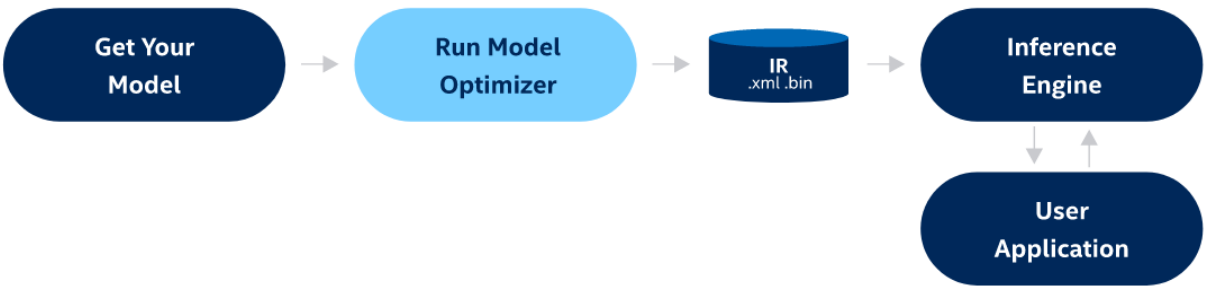


Figure 11 – Στάδια εκτέλεσης OpenVINO

9.4 Λήψη νευρωνικού δικτύου από το Open Model Zoo

Στο Open Model Zoo της Intel, παρέχονται νευρωνικά δίκτυα είτε της Intel, είτε άλλων δημιουργών για διάφορα από τα υποστηριζόμενα frameworks. Τα έτοιμα νευρωνικά δίκτυα της Intel παρέχονται στην μορφή IR, ενώ τα υπόλοιπα ενδέχεται να χρειάζονται μετατροπή, σύμφωνα με την διαδικασία που αναφέρθηκε προηγουμένως.

Αρχικά, πρέπει να πάμε μέσω του command line στον φάκελο που εγκαταστάθηκε το OpenVINO, στον model downloader

cd …\IntelSWTools\openvino\deployment_tools\model_downloader\

Σε εκείνο το σημείο βρίσκεται το αρχείο downloader.py που απαιτείται για την λήψη.

Με την εντολή

Python3 downloader.py -h

Εμφανίζονται οι δυνατές επιλογές εκτέλεσης

```
---> python3 downloader.py -h
usage: downloader.py [-h] [-c CONFIG] [--name NAME] [--print_all]
                     [-o OUTPUT_DIR]

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        path to YML configuration file
  --name NAME           name of topology for downloading
  --print_all           print all available topologies
  -o OUTPUT_DIR, --output_dir OUTPUT_DIR
                        path where to save topologies

list_topologies.yml - default configuration file
--->
```

-o/--output_dir	Επιλογή του φακέλου στον οποίο θα καταλήξουν τα αρχεία μετά το κατέβασμα. Αρχική ρύθμιση, κατέβασμα στον φάκελο του downloader με κάθε δίκτυο σε νέο υποφάκελο
--precisions	Τα δίκτυα προσφέρονται με διάφορες ακρίβειες (Int8, FP8, FP32 και άλλες). Αρχική ρύθμιση, κατέβασμα όλων των μορφών ακρίβειας
--num_attempts	Προσδιορισμός προσπαθειών κατεβάσματος για κάθε δίκτυο, σε περίπτωση που αποτύχει το κατέβασμα. Αρχική ρύθμιση, μία προσπάθεια για κάθε δίκτυο

--cache_dir	Δημιουργία φακέλου άμεσης πρόσβασης. Σε περίπτωση που ένα δίκτυο έχει κατέβει και δεν υπάρχει νεότερη έκδοση στον OMZ, το δίκτυο αντιγράφεται από αυτόν τον φάκελο.
-j/--jobs	Προσδιορισμός ταυτόχρονων λήψεων. Αρχική ρύθμιση μίας λήψης
--progress_format	Προσδιορισμός εμφάνισης λεπτομερειών λήψεων σε προγραμματιστική μορφή. Αρχική ρύθμιση, μορφή κατανοητή από τον χρήστη

Εκτελώντας την παραπάνω εντολή με επιλογή "--print_all", βλέπουμε όλα τα νευρωνικά δίκτυα που είναι διαθέσιμα για λήψη από τον Open Model Zoo.

Στην συνέχεια, εκτελούμε την εντολή Python3 downloader.py --name <model name>

Αν δεν προσδιοριστεί κάποια επιλογή, η εντολή θα τρέξει με την επιλογή --all, με την οποία προχωρεί σε λήψη όλων των νευρωνικών δικτύων που υπάρχουν στον Open Model Zoo, στον φάκελο του Model Downloader, σε δενδροειδή μορφή, δηλαδή δημιουργώντας έναν υποφάκελο για κάθε νευρωνικό δίκτυο.

9.5 Μετατροπή νευρωνικού δικτύου σε μορφή IR για εκτέλεση στο υλικό της Intel

Αρχικά θα πρέπει να γίνει η ρύθμιση του Model Optimizer, πηγαίνοντας στον φάκελο demo (cd C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\model_optimizer\install_prerequisites) και εκτελώντας το αρχείο install_prerequisites.bat.

Με αυτόν τον τρόπο, γίνεται η ρύθμιση για όλα τα frameworks. Ρύθμιση μόνο για συγκεκριμένο framework γίνεται εκτελώντας το αντίστοιχο αρχείο install_prerequisites_FRAMEWORK.bat

Παρακάτω θα παρουσιαστούν αναλυτικότερα οι επιλογές και η χρήση.

Όπως αναφέρθηκε, υπάρχουν διάφορα είδη νευρωνικών δικτύων, τα οποία προγραμματίζονται και εκπαιδεύονται σε πληθώρα frameworks. Παράλληλα, η Intel προσφέρει ευρύ φάσμα ειδών υλικού και ακόμα περισσότερες επιλογές ανά είδος, όπως για παράδειγμα διαφορετικές γενιές επεξεργαστών ή διαφορετικά μεγέθη FPGAs, τα οποία ενδέχεται να έχουν και διαφορετικά χαρακτηριστικά.

Σε περίπτωση που κάποιος θέλει να εκτελέσει ένα νευρωνικό δίκτυο, το οποίο είναι εκπαιδευμένο σε ένα συγκεκριμένο framework, πάνω σε έναν τύπο υλικού, θα χρειαστεί να προγραμματίσει ένα

είδος διεπαφής ή διασύνδεσης. Σε περίπτωση που πρέπει να αλλάξει κάτι από την παραπάνω σειρά, θα χρειαστεί να προγραμματίσει νέο είδος διεπαφής.

Σκοπός του Model Optimizer είναι η δημιουργία ενός κοινού API για την ευκολία στην επικοινωνία και εκτέλεση όλων των συμβατών frameworks τεχνητής νοημοσύνης με όλο το συμβατό υλικό της Intel.

- Βελτιστοποίηση με τον Model Optimizer [44]

Ο Model Optimizer συνοψίζεται σε 3 βασικά στάδια :

1) Μετατροπή του νευρωνικού δικτύου σε μορφή IR
2) Βελτιστοποίηση για εκτέλεση σε συγκεκριμένο υλικό
3) Μετατροπή βαρών σε συγκεκριμένη ακρίβεια

9.5.1 Μετατροπή του νευρωνικού δικτύου σε μορφή IR

Όπως προαναφέρθηκε, κάθε framework εξάγει τα νευρωνικά δίκτυα που προγραμματίζονται και εκπαιδεύονται σε αυτό, σε διαφορετική μορφή. Η Intel προχώρησε στην δημιουργία μίας κοινής μορφής αναπαράστασης, την λεγόμενη Ενδιάμεση Αναπαράσταση (IR – Intermediate Representation).

Σε αυτή την μορφή, κάθε νευρωνικό δίκτυο αποτελείται από 2 αρχεία :

- .bin – Το σύνολο των βαρών (weights) και των δυναμικών πόλωσης (biases) κάθε κόμβου
- .xml – Περιγραφή της τοπολογίας, των στρωμάτων, του τρόπου ενδιάμεσης διασύνδεσης και άλλες παραμέτρους του νευρωνικού δικτύου

9.5.2 Βελτιστοποίηση για εκτέλεση σε συγκεκριμένο υλικό

Σε κάθε εκτέλεση του Model Optimizer διευκρινίζεται με συγκεκριμένη επιλογή το υλικό στο οποίο πρόκειται να εκτελεστεί το Inferencing. Μεγάλο κομμάτι του αποτελεί η διαδικασία βελτιστοποίησης για το εκάστοτε υλικό, έτσι ώστε να επιταχυνθεί η εκτέλεση, να μειωθεί η χρήση πόρων του υλικού, μνήμης και κατανάλωσης. Οι τεχνικές που ακολουθούνται περιλαμβάνουν διάφορες τεχνικές, όπως την συνένωση στρωμάτων (layers fusion), την βελτιστοποίηση της περιγραφής (optimization description) και την ομαλοποίηση συνόλων (batch normalization).

Οι τεχνικές που χρησιμοποιούνται είναι γενικής χρήσης, δηλαδή δεν επηρεάζονται από το υλικό στο οποίο θα εκτελεστεί το νευρωνικό δίκτυο (hardware agnostic).

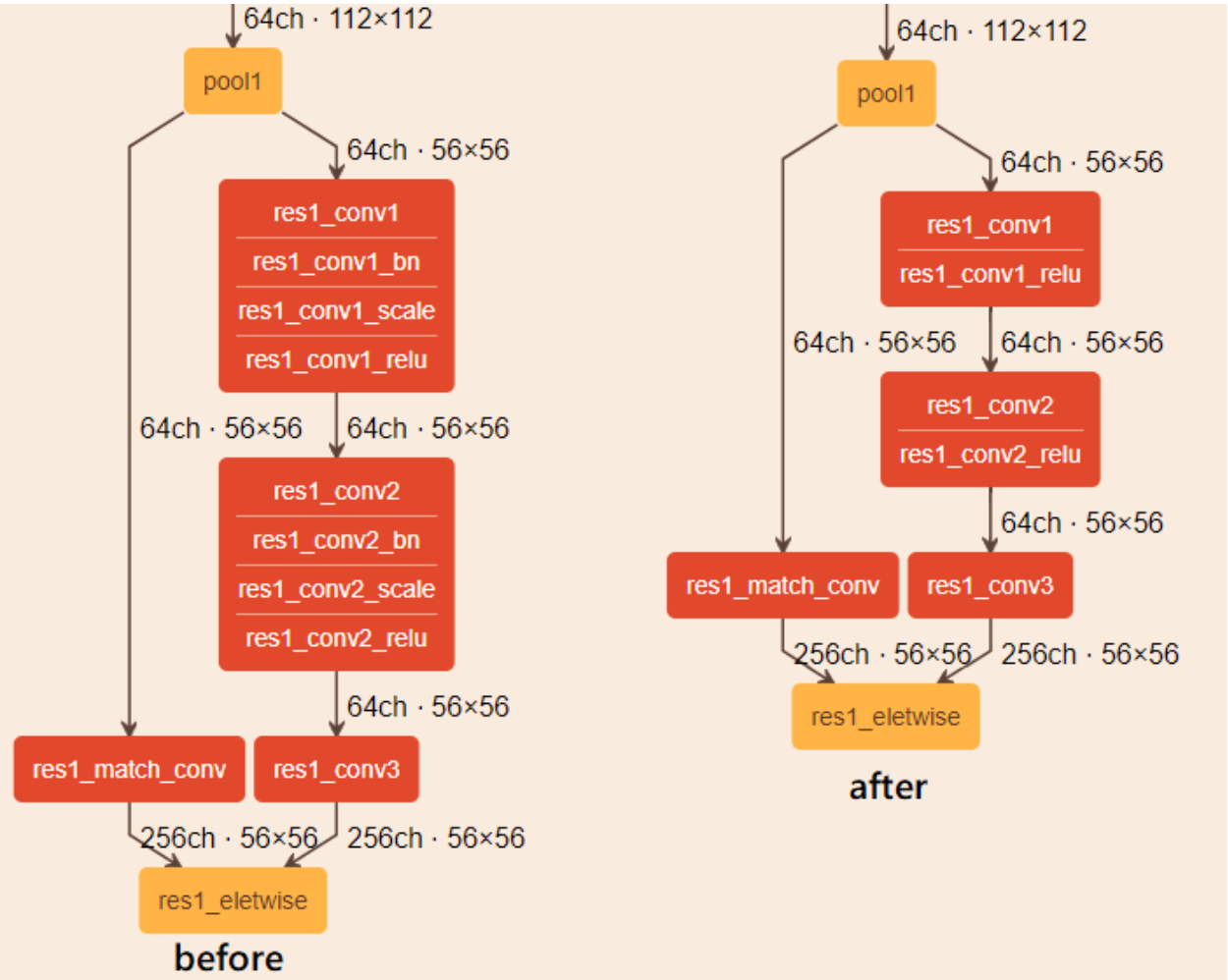


Figure 12 – Παράδειγμα βελτιστοποίησης νευρωνικού δικτύου με κατάργηση στρωμάτων

9.5.3 Μετατροπή βαρών σε συγκεκριμένη ακρίβεια

Τα νευρωνικά δίκτυα, συνήθως, προγραμματίζονται με ακρίβεια δεκαδικών 32 bit (Floating Point 32). Αυτή η ακρίβεια δεν είναι η βέλτιστη δυνατή για κάθε υλικό, λόγω των περιορισμένων δυνατοτήτων κάθε υλικού. Για παράδειγμα, οι ενσωματωμένες GPU εκτελούν βέλτιστα σε ακρίβεια δεκαδικών 16 bit (Floating Point 16), ενώ άλλο υλικό με ακρίβεια σε ακέραιο 8 bit (Integer 8).

Με την κατάλληλη επιλογή υλικού κατά την μετατροπή στον Model Optimizer, επιλέγεται και μετατρέπεται το νευρωνικό δίκτυο στην βέλτιστη μορφή ακρίβειας.

9.6 Εκτέλεση παραδείγματος στον Model Optimizer – Μετατροπή TF model για εκτέλεση στο NCS2

Σε περίπτωση που δεν χρησιμοποιηθεί έτοιμο δίκτυο από το Open Model Zoo (ή από αλλού σε μορφή για χρήση στο OpenVINO), πρέπει να ακολουθηθούν τα παρακάτω βήματα :

9.6.1 Ρύθμιση του Model Optimizer

Στο 1ο βήμα, εφόσον δεν έχει γίνει κατά την εγκατάσταση η ρύθμιση του Model Optimizer, εκτελείται το αρχείο install_prerequisites.bat που βρίσκεται στον φάκελο demo (cd C:\Program Files (x86)\IntelSWTools\openvino\deployment_tools\model_optimizer\install_prerequisites).

Με αυτόν τον τρόπο, γίνεται η ρύθμιση για όλα τα frameworks. Για ρύθμιση μόνο για συγκεκριμένο framework, εκτελείται το αντίστοιχο αρχείο install_prerequisites_FRAMEWORK.bat

9.6.2 Λήψη στιγμιότυπου από το μοντέλο του TensorFlow

Για να μετατραπεί σε εκτελέσιμο αρχείο από το NCS2 το μοντέλο, πρέπει να ληφθεί στιγμιότυπο. Αυτό γίνεται είτε κατεβάζοντας έτοιμο από το TF Zoo (https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md), είτε παγώνοντας το δικό μας μοντέλο από την Python με τον παρακάτω κώδικα (TF 1.0).

import TensorFlow as tf

from tensorflow.python.framework import graph_io

frozen = tf.graph_util.convert_variables_to_constants(sess, sess.graph_def, ["name_of_the_output_node"])

graph_io.write_graph(frozen, './', 'inference_graph.pb', as_text=False)

Το μοντέλο αποθηκεύεται σε μορφή .pb

Στην συγκεκριμένη περίπτωση, θα γίνει λήψη του SSD MobileNet V2 COCO

9.6.3 Μετατροπή σε IR μορφή με χρήση του Model Optimizer

Στην συνέχεια, από τον φάκελο του Model Optimizer (**\deployment_tools\model_optimizer), εκτελείται το mo_tf.py αρχείο (python3 mo_tf.py --saved_model_dir <SAVED_MODEL_DIRECTORY>).

Επειδή γίνεται μετατροπή του συγκεκριμένου μοντέλου, χρειάζεται η διευκρίνιση του input shape χρησιμοποιώντας το αντίστοιχο αρχείο ssd support.

Επίσης, πρέπει να διευκρινιστεί στον Model Optimizer το pipeline αρχείο που χρησιμοποιήθηκε για την δημιουργία του συγκεκριμένου μοντέλου.

Η πλήρης εντολή είναι

python mo_tf.py --input_model "C:\AI tests\ssd_mobilenet_v2_coco_2018_03_29\frozen_inference_graph.pb" --tensorflow_use_custom_operations_config ./extensions/front/tf/ssd_support.json --tensorflow_object_detection_api_pipeline_config "C:\AI tests\ssd_mobilenet_v2_coco_2018_03_29\pipeline.config"

Για την ανάγνωση και την αποθήκευση του μοντέλου, ίσως χρειαστεί να εκτελεστεί η command line με admin rights.

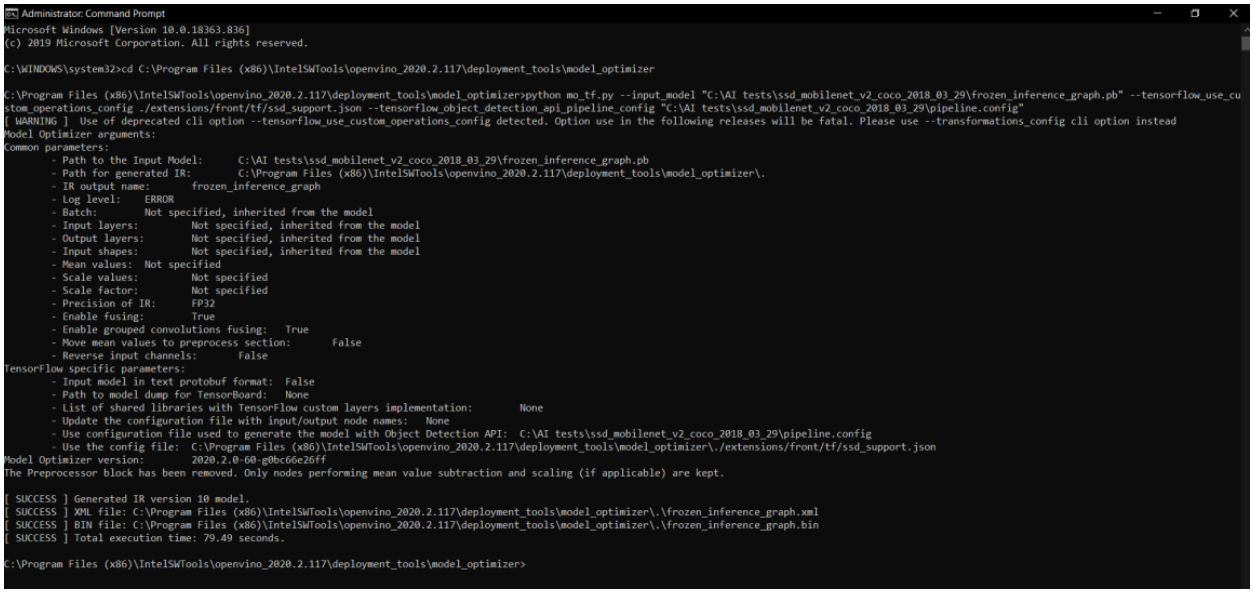
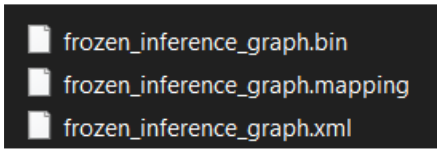


Figure 13 – Εκτέλεση μετατροπής νευρωνικού δικτύου σε μορφή IR



Με την επιτυχή ολοκλήρωση, δημιουργούνται τα παρακάτω αρχεία

XML : Περιγραφή της τοπολογίας του δικτύου

BIN : Περιέχει weights και biases σε δυαδική μορφή.

MAPPING : Χάρτης του δικτύου

Για την εκτέλεση του inferencing, απαιτούνται τα αρχεία .bin και .xml

Για την συγκεκριμένη εργασία θα χρησιμοποιηθούν νευρωνικά δίκτυα τα οποία είναι ήδη εκπαιδευμένα και σε έτοιμη μορφή IR (Intermediate Representation)

9.7 Επιλογή της κατάλληλης ακρίβειας των βαρών των νευρωνικών δικτύων

Όπως προαναφέρθηκε, κάθε υλικό εκτελεί βέλτιστα το Inferencing με δεδομένα τα οποία έχουν συγκεκριμένη ακρίβεια.

Στο παρακάτω σχήμα φαίνεται το μέγεθος μνήμης σε bits που καταλαμβάνει κάθε τύπος ακρίβειας (precision), αλλά και η κατανομή του μεγέθους της ακρίβειας (accuracy) που επιτυγχάνεται, κατά προσέγγιση.

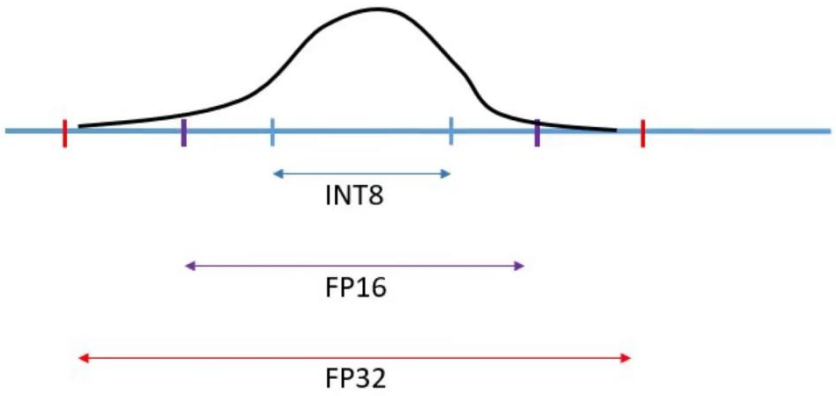


Figure 14 – Ποσοστιαία επίτευξη ακρίβειας(accuracy) για κάθε τύπο ακρίβειας(precision) βαρών νευρωνικών δικτύων κατά προσέγγιση

Όσο μεγαλύτερο το μέγεθος της μνήμης που χρειάζεται κάθε βάρος, από τόσο μεγαλύτερη ακρίβεια αποτελείται το νευρωνικό δίκτυο. Όμως, το κόστος αυτής της ακρίβειας είναι το υλικό που θα χρησιμοποιηθεί για την αποθήκευση, την μεταφορά, αλλά και την επεξεργασία των μεγαλύτερων αριθμών. Επίσης, επειδή δεν υποστηρίζονται όλες οι ακρίβειες από όλα τα υλικά, δεν σημαίνει απαραίτητα ότι η βέλτιστη λύση είναι πάντα η μέγιστη δυνατή αναπαράσταση.

Η επιλογή της κατάλληλης ακρίβειας γίνεται από διάφορους παράγοντες, όπως το υλικό στο οποίο πρόκειται να γίνει εκτέλεση, το μέγεθος των εικόνων για κάθε δευτερόλεπτο, τον χρόνο απόκρισης, την εφαρμογή που θα χρησιμοποιηθεί το δίκτυο.

Έτσι, θέτοντας εξαρχής τον στόχο, θα είναι εφικτό να γίνουν οι κατάλληλες επιλογές. Για παράδειγμα, δεν υπάρχει λόγος εκτέλεσης σε FP32, αν το υλικό δεν το υποστηρίζει ή αν η επιθυμητή βεβαιότητα αποτελέσματος είναι χαμηλότερη από αυτήν που δίνει το FP32 ή αν η συγκεκριμένη βεβαιότητα επιτυγχάνεται σε μεγαλύτερο από τον επιθυμητό χρόνο εκτέλεσης.

Για την διεξαγωγή δοκιμών και ελέγχου της απόδοσης του νευρωνικού δικτύου, η Intel παρέχει το OpenVINO Benchmark Tool. Το συγκεκριμένο εργαλείο παρέχεται γραμμένο και σε C++ και σε Python.

9.8 Εκτέλεση δοκιμών για benchmarking

Για την εκτέλεση του Inferencing υπάρχουν 2 τρόποι :

1) Synchronous mode
2) Asynchronous mode

- Benchmark tool documentation [45]

Με τον ίδιο τρόπο γίνεται η δοκιμή και στο benchmark tool. Για τους 2 παραπάνω τρόπους παρέχονται τα Synchronous API και Asynchronous API.

Στο Synchronous API η δοκιμή γίνεται με στόχο την βέλτιστη απόδοση σε χρόνο (latency oriented), ενώ σε Asynchronous API η δοκιμή γίνεται με στόχο την μέγιστη διακίνηση δεδομένων, δηλαδή εικόνων ανά δευτερόλεπτο – FPS (throughput oriented).

Πριν από την εκτέλεση του benchmark tool, πρέπει να γίνει εγκατάσταση όλων των προαπαιτήσεων.

Αρχικά μεταβαίνουμε στον φάκελο του benchmark tool και εκτελούμε την παρακάτω εντολή

Pip install -r requirements.txt

Αφού ολοκληρωθεί η εγκατάσταση όλων των προαπαιτούμενων, μπορεί να εκτελεστεί το benchmark tool.

Python3 benchmark_app.py -m <model> -i <input> -d <device>

Στις περισσότερες περιπτώσεις, το benchmark tool επιλέγει τις κατάλληλες ρυθμίσεις για την εκτέλεση απλώς και μόνο με την εισαγωγή του υλικού (-d CPU). Όμως, επειδή δεν είναι πάντα ακριβείς, δίνονται περισσότερες επιλογές, οι οποίες παρατίθενται παρακάτω.

```
benchmark_app.py [-h]    [-i PATH_TO_INPUT] -m PATH_TO_MODEL
                         [-d TARGET_DEVICE]
                         [-l PATH_TO_EXTENSION] [-c PATH_TO_CLDNN_CONFIG]
                         [-api {sync,async}] [-niter NUMBER_ITERATIONS]
                         [-b BATCH_SIZE]
                         [-stream_output [STREAM_OUTPUT]] [-t TIME]
                         [-progress [PROGRESS]] [-nstreams NUMBER_STREAMS]
                         [-nthreads NUMBER_THREADS] [-pin {YES,NO}]
                         [--exec_graph_path EXEC_GRAPH_PATH]
                         [-pc [PERF_COUNTS]]
```

-i <path to input>	Διαδρομή στην οποία βρίσκονται οι εικόνες οι οποίες θα χρησιμοποιηθούν από το benchmark tool
-m <path to model>	Διαδρομή στην οποία βρίσκεται το νευρωνικό δίκτυο που θα χρησιμοποιηθεί για τον έλεγχο από το benchmark tool

-d <device> -d MULTI:<d1>,<d2>,<dn> -d HETERO:<d1>,<d2>,<dn>	Ο τύπος του υλικού στο οποίο θα εκτελεστεί το benchmarking. Μόνο για συσκευές που είναι συνδεδεμένες και ενεργές στο σύστημα (CPU, GPU, VPU, FPGA) Υπάρχει η δυνατότητα εκτέλεσης σε πολλαπλές συσκευές (MULTI) Υπάρχει η δυνατότητα εκτέλεσης σε ετερογενείς συσκευές, εφόσον κάποιο τμήμα του νευρωνικού δικτύου δεν υποστηρίζεται στην συσκευή με την υψηλότερη προτεραιότητα (HETERO)
-api <api_type> -api sync -api async	Επιλογή του τύπου του API που θα χρησιμοποιηθεί. Synchronous API Asynchronous API Αρχική ρύθμιση εκτέλεσης σε Async
-b <batch size>	Το σύνολο των εικόνων που στέλνονται ταυτόχρονα για Inferencing
-nstreams <number of streams> -nstreams <d1>:<nst1>,<d1>:<nst2>	Το σύνολο των παράλληλων εκτελέσεων inferencing για εκτέλεση σε CPU ή/και GPU. Υπάρχει και η δυνατότητα επιλογής ανά συσκευή. Αρχική ρύθμιση η αυτόματη επιλογή ανά συσκευή.
-nthreads <number of threads>	Το σύνολο των πυρήνων που θα χρησιμοποιηθούν για inferencing σε CPU

Για την έκδοση του C++ benchmark tool οι επιλογές είναι διαφορετικές ως προς την γραφή τους, αλλά δίνουν σχεδόν ίδιες δυνατότητες.

- Benchmark tool C++ documentation [46]

9.9 Βήματα δοκιμών στο benchmark tool

1) Προετοιμασία προαπαιτούμενων
2) Εκτέλεση Inferencing σε CPU με batch = 1
3) Εκτέλεση Inferencing σε CPU με batch = 32
4) Εκτέλεση Inferencing σε GPU με batch = 1
5) Εκτέλεση Inferencing σε GPU με batch = 32
6) Εκτέλεση Inferencing σε CPU με batch = 1 σε 4 πυρήνες

9.9.1 Προετοιμασία προαπαιτούμενων

Αρχικά, πρέπει να επιλεγεί το νευρωνικό δίκτυο και να μετατραπεί σε μορφή IR. Στο συγκεκριμένο παράδειγμα επιλέχθηκε το squeeznet1.1 σε ακρίβεια FP32.

Επίσης, πρέπει να δημιουργηθεί φάκελος με τις εικόνες που θα χρησιμοποιηθούν ως δείγματα για το benchmark tool. Για το συγκεκριμένο παράδειγμα δημιουργήθηκε ο φάκελος TEST που περιέχει την εικόνα από το παράδειγμα security barrier, δηλαδή την εικόνα που χρησιμοποιήθηκε μετά την εγκατάσταση του OpenVINO για την δοκιμή καλής λειτουργίας του.

9.9.2 Εκτέλεση Inferencing σε CPU με batch = 1

Η πρώτη εκτέλεση θα γίνει με τις πιο βασικές ρυθμίσεις, δηλαδή μόνο με την εικόνα που θα χρησιμοποιηθεί, το υλικό που θα τρέξει και το νευρωνικό δίκτυο.

```
python3 benchmark_app.py
        -m …/squeezenet1.1_FP32.xml
        -i …/TEST/
        -d CPU
```

```
--->  ./benchmark_app
       -i    ~/TEST/
       -d    CPU
       -m    ~/squeezenet1.1_FP32.xml
```

Μετά την επιτυχή ολοκλήρωση, εμφανίζονται τα αποτελέσματα του χρόνου εκτέλεση ανά Inference και την μέγιστη δυνατότητα διακίνησης εικόνων, στην παρακάτω μορφή

```
[Step 7/8]
Start inference asynchronously (60000.00 ms duration, 2 inference requests in parallel)
Progress: [....................] 100.00% done

[Step 8/8] Dump statistics report
[ INFO ] Statistics collecting was not requested. No reports are dumped.
Progress: [....................] 100.00% done

Latency: 5.11 ms
Throughput: 380.44 FPS
--->
```

Figure 15 – Αποτέλεσμα συγκριτικής αξιολόγησης για παρτίδα 1 εισόδου στην CPU

9.9.3 Εκτέλεση Inferencing σε CPU με batch size = 32

Στην επόμενη εκτέλεση, διαφοροποιείται μόνο ο αριθμός των εικόνων που στέλνονται ως παρτίδες, με την κάθε παρτίδα να αποτελείται από 32 εικόνες. Στο συγκεκριμένο παράδειγμα, καθώς έχουμε λιγότερες από 32 εικόνες μέσα στον φάκελο TEST, χρησιμοποιούνται οι ίδιες εικόνες περισσότερες από 1 φορές.

```
--->  ./benchmark_app
       -i    ~/TEST/
       -d    CPU
       -m    ~/squeezenet1.1_FP32.xml
       -b    32
```

Μετά την επιτυχή ολοκλήρωση, εμφανίζονται τα αποτελέσματα. Ενώ η μέγιστη δυνατότητα διακίνησης εικόνων παραμένει υψηλή, ο χρόνος ανά Inferencing αυξήθηκε κατακόρυφα. Από αυτά, εξάγεται το συμπέρασμα ότι δεν υπάρχει λόγος για την εκτέλεση του συγκεκριμένου νευρωνικού δικτύου σε αυτό το υλικό με τέτοιο μέγεθος παρτίδας.

```
[Step 7/8]
Start inference asynchronously (60000.00 ms duration, 2 inference requests in parallel)
Progress: [....................] 100.00% done

[Step 8/8] Dump statistics report
[ INFO ] Statistics collecting was not requested. No reports are dumped.
Progress: [....................] 100.00% done

Latency: 174.10 ms
Throughput: 366.23 FPS
--->
```

Figure 16 – Αποτέλεσμα συγκριτικής αξιολόγησης για παρτίδα 32 εισόδων στην CPU

9.9.4 Εκτέλεση Inferencing σε GPU με batch = 1

Η επόμενη εκτέλεση θα είναι όμοια με την 1η, αλλά η συσκευή που θα χρησιμοποιηθεί θα είναι η GPU. Επίσης, θα χρησιμοποιηθεί η έκδοση με ακρίβεια 16 (FP16), καθώς στην GPU προτιμάται ο συγκεκριμένος τύπος.

```
--->  ./benchmark_app
       -i    ~/TEST/
       -d    GPU
       -m    ~/squeezenet1.1_FP16.xml
```

Τα αποτελέσματα είναι παρόμοια με την εκτέλεση στην CPU.

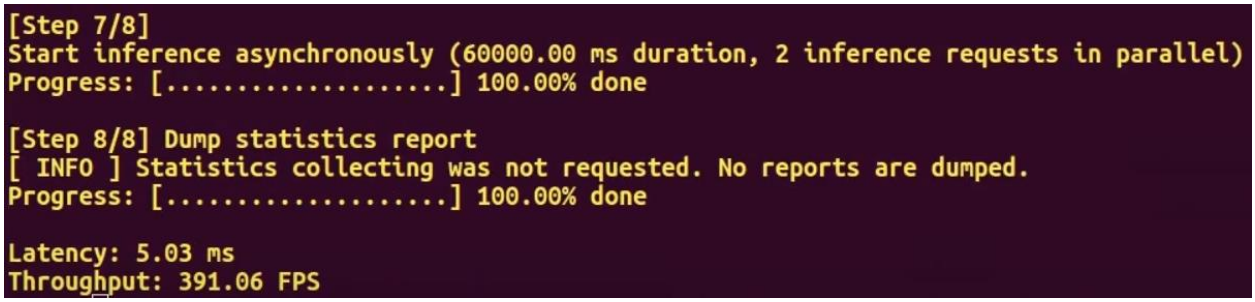
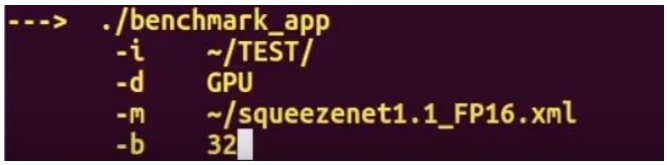


Figure 17 – Αποτέλεσμα συγκριτικής αξιολόγησης για παρτίδα 1 εισόδου στην GPU

9.9.5 Εκτέλεση Inferencing σε GPU με batch size = 32

Ομοίως με τις προηγούμενες εκτελέσεις, η αλλαγή που γίνεται είναι το μέγεθος της παρτίδας σε 32 εικόνες (batch size = 32).



Σε αντίθεση με την CPU, η εκτέλεση με μεγαλύτερο μέγεθος παρτίδας στην GPU, δίνει εμφανέστατα καλύτερη διακίνηση εικόνων, παρότι αυξάνεται ο χρόνος inferencing ανά εικόνα. Αυτό συμβαίνει γιατί η GPU είναι κατασκευασμένη για να εκτελεί παράλληλη επεξεργασία, σε αντίθεση με την CPU.

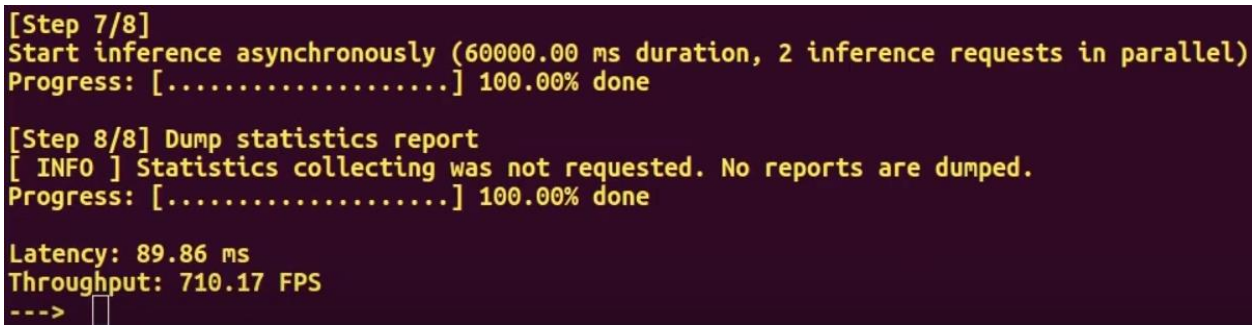


Figure 18 – Αποτέλεσμα συγκριτικής αξιολόγησης για παρτίδα 32 εισόδων στην GPU

9.9.6 Εκτέλεση Inferencing σε CPU με batch = 1 σε 4 πυρήνες

Στην τελευταία εκτέλεση, θα δοκιμαστεί η χρήση των 4 φυσικών πυρήνων για παράλληλη εκτέλεση inferencing. Στα προηγούμενα παραδείγματα, το benchmark tool επέλεγε αυτόματα την βέλτιστη

ρύθμιση, η οποία ήταν 2 αιτήματα παράλληλου inferencing. Αντιθέτως, στην συγκεκριμένη εκτέλεση, θα επιλεχθούν 4.

Ο κώδικας που θα χρησιμοποιηθεί για την python έκδοση του benchmark tool, είναι ο παρακάτω

```
python3 benchmark_app.py
    -m .../squeezenet1.1_FP32.xml
    -i .../TEST/
    -d CPU
    -nstreams 4
    -nthreads 4
```

```
--->  ./benchmark_app
      -i      ~/TEST/
      -d       CPU
      -m         ~/squeezenet1.1_FP32.xml
      -nireq     4
      -nthreads 4
```

Λόγω του της αύξησης των πυρήνων και των Inferencing αιτημάτων, υπάρχει μία μικρή βελτίωση.

Σε συστήματα με περισσότερους πόρους, η βελτίωση θα είναι αντίστοιχα μεγαλύτερη.

```
[Step 7/8]
Start inference asynchronously (60000.00 ms duration, 4 inference requests in parallel)
Progress: [....................] 100.00% done

[Step 8/8] Dump statistics report
[ INFO ] Statistics collecting was not requested. No reports are dumped.
Progress: [....................] 100.00% done

Latency: 9.85 ms
Throughput: 404.65 FPS
--->
```

Figure 19 – Αποτέλεσμα συγκριτικής αξιολόγησης για παράλληλη εκτέλεση σε 4 πυρήνες στην CPU

- Intel Benchmark tool tutorial [47]

9.10 Εκτέλεση Model Optimizer για μετατροπή σε συγκεκριμένη ακρίβεια

Αρχικά γίνεται η μετάβαση στον φάκελο του Model Optimizer (**\deployment_tools\model_optimizer) και η εκτέλεση του mo.py αρχείου με την παρακάτω εντολή

Python mo.py --input_model INPUT_MODEL --output_dir <OUTPUT_MODEL_DIR>

Όπως φαίνονται στο παράδειγμα που ακολουθεί, το οποίο είναι για caffe framework, για το νευρωνικό δίκτυο resnet-50, προσθέτοντας τις κατάλληλες επιλογές, μετατρέπεται το δίκτυο σε μορφή IR, στην αρχή με ακρίβεια FP32 και στην συνέχεια με ακρίβεια FP16.

9.10.1 Ακρίβεια FP32 (Floating Point 32 bit)

Python3 mo_caffe.py

--input_model <path to input model>

--output_dir <path to directory for output model>

--model_name <output model name>

--data_type <output data type & length (floating point, integer etc.)>

```
--->
---> cd ../model_optimizer/
--->
---> python3 mo_caffe.py
       --input_model $model/resnet-50.caffemodel
       --output_dir /home/
       --model_name resnet-50-FP32
       --data_type FP32
```

Figure 20 – Κώδικας μετατροπής νευρωνικού δικτύου σε μορφή IR ακρίβειας FP32

Μετά την επιτυχή εκτέλεση, θα δούμε ότι δημιουργήθηκαν τα 2 απαραίτητα αρχεία της μορφής IR (.xml & .bin)

```
[ SUCCESS ] Generated IR model.
[ SUCCESS ] XML file: /home/resnet-50-FP32.xml
[ SUCCESS ] BIN file: /home/resnet-50-FP32.bin
[ SUCCESS ] Total execution time: 7.23 seconds.
--->
```

9.10.2 Ακρίβεια FP16

Ομοίως, για την ακρίβεια FP16 (ή κάθε άλλη ζητούμενη), εκτελείται ο ίδιος κώδικας, αλλάζοντας μόνο το --data_type F16 (ή στην επιθυμητή ακρίβεια) και –model_name της μορφής IR.

```
---> python3 mo_caffe.py
       --input_model $model/resnet-50.caffemodel
       --output_dir /home/
       --model_name resnet-50-FP16
       --data_type FP16
```

```
[ SUCCESS ] Generated IR model.
[ SUCCESS ] XML file: /home/resnet-50-FP16.xml
[ SUCCESS ] BIN file: /home/resnet-50-FP16.bin
[ SUCCESS ] Total execution time: 7.55 seconds.
--->
```

Figure 21 – Κώδικας και αποτελέσματα μετατροπής νευρωνικού δικτύου σε μορφή IR ακρίβειας FP16

9.11 Προηγμένες ρυθμίσεις Model Optimizer

Πέρα από τις βασικές που προαναφέρθηκαν, ο Model Optimizer μπορεί να εκτελέσει και πιο προχωρημένες, όπως η προβολή πληροφοριών σχετικά με την τοπολογία του νευρωνικού δικτύου, την αλλαγή του μεγέθους της εισόδου του νευρωνικού δικτύου και την αλλαγή του μεγέθους της παρτίδας.

Καθώς είναι λειτουργίες που δεν θα χρησιμοποιηθούν εκτενέστερα στην παρούσα εργασία, θα γίνει επιγραμματική αναφορά σε αυτές.

Για την προβολή της τοπολογίας του νευρωνικού δικτύου, αφού γίνει η μετατροπή του σε μορφή IR, μπορούμε να δούμε το xml αρχείο που παράγεται.

Η αλλαγή του μεγέθους της εισόδου του νευρωνικού δικτύου γίνεται έτσι ώστε να μην χρειάζεται επανεκπαίδευση του δικτύου για εικόνες διαφορετικών διαστάσεων, αλλά ούτε και αλλαγή του συνόλου των εικόνων. Ο Model Optimizer αναλαμβάνει να προσαρμόσει όλα τα στρώματα ανάλογα με την αλλαγή που ζητήθηκε.

Επίσης, σε περίπτωση που χρειάζεται να εισέρχονται στο σύστημα πολλές εικόνες μαζί, αλλά να είναι ως ένα πακέτο στην είσοδο του νευρωνικού δικτύου, μπορεί να πραγματοποιηθεί αλλαγή της εισόδου του δικτύου. Για παράδειγμα, αντί της εισόδου 1 εικόνας 3 χρωμάτων και διαστάσεων 240x320 ([1,3,240,320]), μπορούμε να προσαρμόσουμε την είσοδο σε 4 εικόνες, δηλαδή [4,3,240,320]. Αυτή η αλλαγή, σε αντίθεση με την αλλαγή του μεγέθους της εισόδου του νευρωνικού δικτύου, δεν επηρεάζει τα υπόλοιπα στρώματα. Η αλλαγή αυτή είναι ίδια με την δήλωση batch size = 4. Ως σύνηθες αποτέλεσμα, θα αυξηθεί ο χρόνος απόκρισης και ταυτόχρονα θα αυξηθεί και η μέγιστη διακίνηση εικόνων.

Μία ακόμα δυνατότητα που παρέχει ο Model Optimizer, είναι η αποκοπή μέρους του νευρωνικού δικτύου. Για παράδειγμα, αν θέλουμε να τρέχει μόνο ένα συγκεκριμένο μέρος στην αρχή και να παραλείπεται από εκεί και κάτω έως ένα άλλο κομμάτι προς το τέλος, μπορούμε να το εξάγουμε.

Τέλος, είναι εφικτό να προεπεξεργαστούμε δεδομένα που θα εισέλθουν στο νευρωνικό δίκτυο. Συνήθης πρακτική είναι η ομαλοποίηση των δεδομένων. Από την στιγμή που τα βάρη των νευρώνων έχουν τιμές από 0 έως 1, μπορούν και τα pixels των εικόνων πάνω στις οποίες θα γίνει το Inferencing να μετατραπούν από την κλίμακα 0 έως 255 στην κλίμακα 0 έως 1. Αυτό επιτυγχάνεται από τον Model Optimizer, προσθέτοντας ένα ακόμα στρώμα στην αρχή του νευρωνικού δικτύου.

9.12 Inference Engine

Η μηχανή εξαγωγής συμπερασμάτων (Inference Engine) είναι το κομμάτι του OpenVINO που είναι υπεύθυνο για την εκτέλεση του εκάστοτε νευρωνικού δικτύου πάνω στο υλικό της Intel και την εξαγωγή συμπερασμάτων – αποτελεσμάτων πάνω στις εισόδους. Για παράδειγμα, αν σε μία εικόνα εκτελέσουμε νευρωνικό δίκτυο για την εξαγωγή των οχημάτων, ανάλογα τα ζητούμενα δεδομένα, το τελικό αποτέλεσμα θα είναι μία εικόνα που θα περιέχει πληροφορίες για τα οχήματα που βρέθηκαν, τον τύπο του κάθε οχήματος και άλλες σχετικές πληροφορίες, καθώς και την βεβαιότητα για το κάθε αποτέλεσμα.

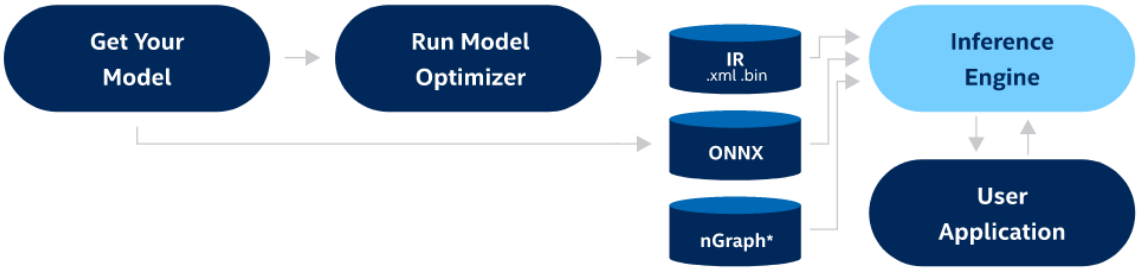


Figure 22 – Μορφές εισόδου στην Inference Engine

Η Inference Engine αποτελείται από ένα σύνολο βιβλιοθηκών γραμμένων σε C++. Όμως, παρέχονται διεπαφές και διασυνδέσεις για C και Python. Μέσω αυτής γίνεται η ανάγνωση της ενδιάμεσης αναπαράστασης (μορφή IR – Intermediate Representation) ή της ONNX μορφής του νευρωνικού δικτύου και η προσαρμογή στο εκάστοτε υλικό.

Η Inference Engine είναι δομημένη σε πρόσθετα (plugins). Μία γενική διεπαφή (API) αποτελεί το κύριο μέρος, ενώ ταυτόχρονα προσφέρει συνδεσιμότητα με συγκεκριμένο υλικό που επιλέγεται για την εκτέλεση μέσω εξειδικευμένων διεπαφών (hardware-specific APIs).

Στο παρακάτω γράφημα φαίνεται η χρήση του εκάστοτε plugin που χρησιμοποιείται για κάθε υλικό.

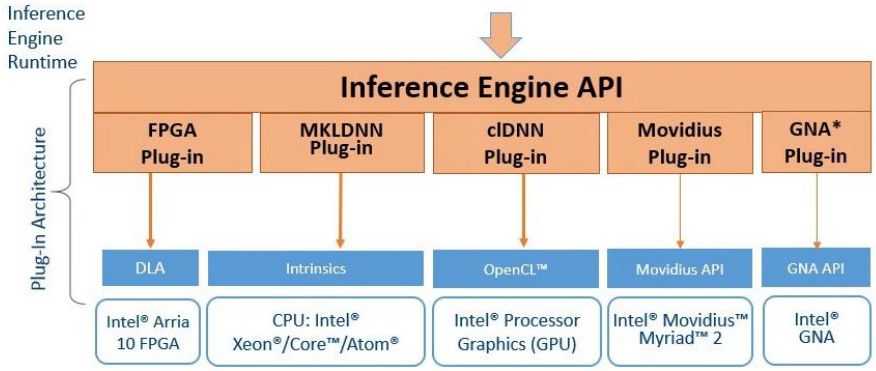


Figure 23 – Hardware plugins

Κατά την κλήση του εκάστοτε plugin, δεν εκτελείται απευθείας το Inferencing, αλλά εκτελούνται και άλλες βελτιστοποιήσεις ειδικές για κάθε υλικό.

9.13 Εξειδικευμένες βελτιστοποιήσεις υλικού Inference Engine

Οι βελτιστοποιήσεις αυτές μπορούν να χωριστούν σε 3 κατηγορίες επιπέδων :

1) Επικοινωνίας
2) Μνήμης
3) Πυρήνα

Βελτιστοποιήσεις σε επίπεδο επικοινωνίας (Network level optimizations)

Αφορούν την επικοινωνία των δεδομένων και για να επιτευχθούν, πραγματοποιούνται αλλαγές στα βήματα και στον τρόπο επικοινωνίας των δεδομένων μεταξύ τους, με αποτέλεσμα να αυξάνεται η απόδοση και να ελαχιστοποιείται ο χρόνος μετατροπής κατά την διάρκεια της εκτέλεσης του Inferencing.

Βελτιστοποιήσεις σε επίπεδο μνήμης (Memory level optimizations)

Αφορούν την αποτύπωση των δεδομένων στην μνήμη. Κυρίως αναφέρονται σε αναδιάταξη των δεδομένων στις θέσεις μνήμης, ανάλογα το είδος του υλικού, στο οποίο πρόκειται να εκτελεστεί το Inferencing.

Βελτιστοποιήσεις σε επίπεδο πυρήνα (Kernel level optimizations)

Οι βελτιστοποιήσεις σε επίπεδο kernel αφορούν το υλικό που πρόκειται να εκτελεστεί το Inferencing και έχουν σχέση με την αρχιτεκτονική που είναι σχεδιασμένο. Έτσι, η Inference Engine επιλέγει το κατάλληλο plugin για την βέλτιστη υλοποίηση.

9.14 Inference Engine workflow chart

Στο παρακάτω γράφημα φαίνονται τα βήματα της Inference Engine. Παράλληλα με την επεξήγηση θα παρουσιάζεται και ο απαραίτητος ελάχιστος κώδικας που δίνεται από την Intel για την υλοποίηση.

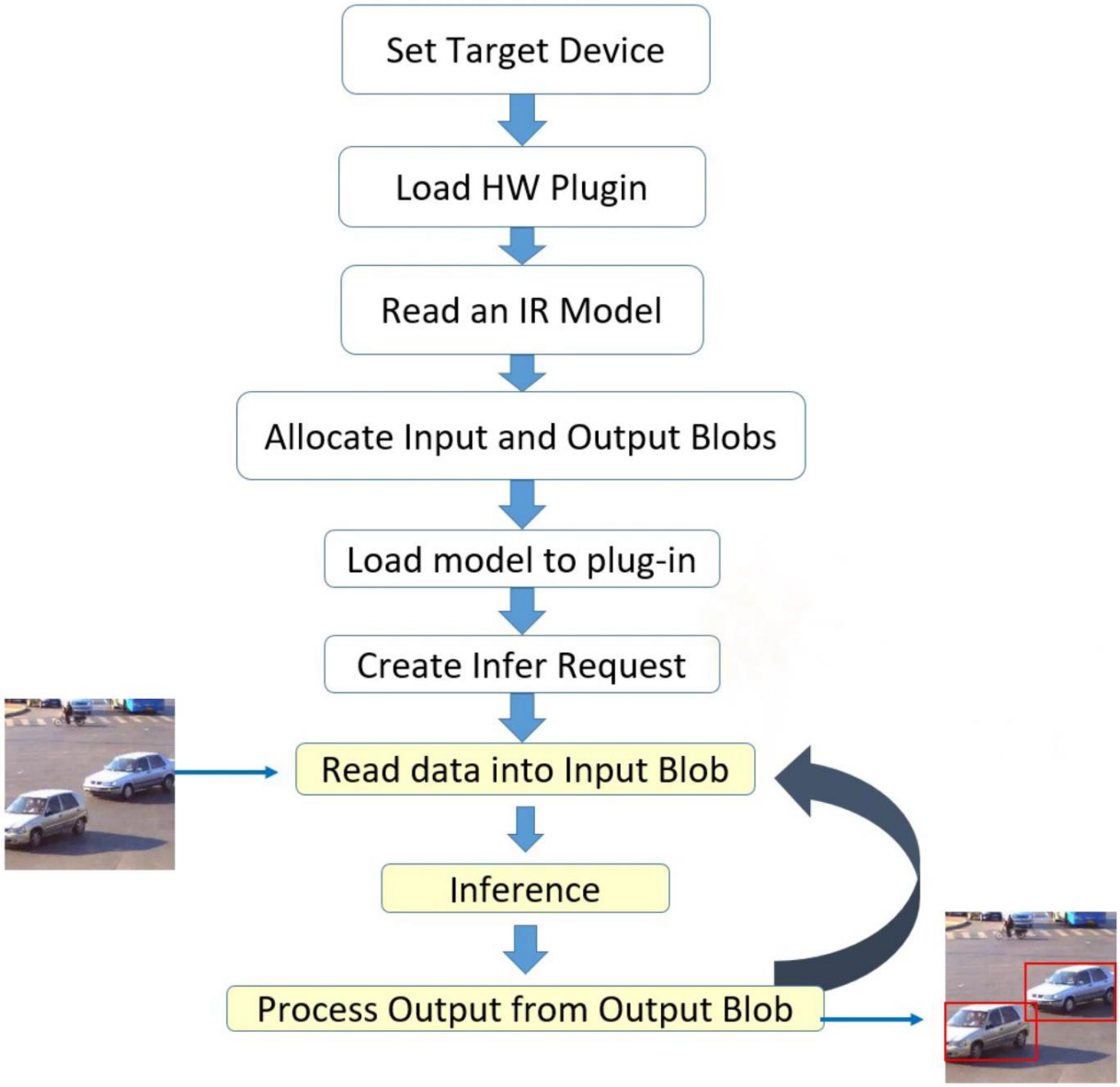


Figure 24 – Inference Engine workflow chart

Ο κώδικας αποτελεί μέρος του *object_detection_sample_ssd.py*

Στα βήματα 1 και 2 ορίζεται από τον χρήστη το υλικό που πρόκειται να εκτελεστεί το Inferencing και καλείται το κατάλληλο plugin.

```
from openvino.inference_engine import IENetwork, IEPlugin

plugin = IEPlugin(device = args.device, plugin_dirs = args.plugin_dir)
```

Στα βήμα 3 γίνεται η εισαγωγή του νευρωνικού δικτύου στην ροή της Inference Engine. Στο βήμα 4 γίνεται η αντιστοίχιση του νευρωνικού δικτύου με όλους τους νευρώνες και τα βάρη τους, στην μεταβλητή net.

```
model_xml = args.model
model_bin = os.path.splitext(model_xml)[0] + ".bin

net = IENetwork(model = model_xml, weights = model_bin)
```

Στο βήμα 5 φορτώνεται το νευρωνικό δίκτυο που εισήχθη στα προηγούμενα βήματα, στο κατάλληλο plugin που κλήθηκε κατά το βήμα 2.

```
Exec_net = plugin.load(network = net)
```

Στα επόμενα 3 βήματα εκτελείται το Inferencing. Εκτελείται συνεχόμενα όσο υπάρχουν δεδομένα, δηλαδή όσο υπάρχουν εικόνες ή ροή βίντεο.

Στο βήμα 6 εισάγεται η εικόνα σε μορφή πίνακα και μετατρέπεται σε διαστάσεις όσες και οι είσοδοι του νευρωνικού δικτύου.

```
# Read and preprocess input images
n, c, h, w = net.inputs[input_blob].shape
images = np.ndarray(shape = (n, c, h, w))
for i in range(n):
    image = cv2.imread(args.input[i])
    if image.shape[:-1] != (h, w):
        log.warning(f'Image {args.input} is resized from {image.shape[:-1]}
        to {(h, w)}')
        image = cv2.resize(image, (w, h))

    # Change data layout from HWC to CHW - OpenVINO expects CHW, original
image has HWC (Height, Width, Channels)
    image = image.transpose((2, 0, 1))
```

Στο βήμα 7 εκτελείται το Inferencing το οποίο αποθηκεύεται στην μεταβλητή res (result).

```
log.info('Starting inference in synchronous mode')
res = exec_net.infer(inputs={input_blob: image})
```

Στο βήμα 8 εξάγεται το αποτέλεσμα του Inferencing από την μεταβλητή res, η οποία είναι ένας πίνακας αριθμών. Η μορφή του πίνακα είναι το σύνολο των δεδομένων και εξαρτάται από την εφαρμογή που εκτελείται. Στο συγκεκριμένο παράδειγμα περιέχει

- συντεταγμένες των κορυφών των πλαισίων που είναι αποτέλεσμα της αναγνώρισης οχημάτων και ανθρώπων
- την λίστα με τις υψηλότερες βεβαιότητες για τα αντικείμενα που αναγνωρίστηκαν
- τις ετικέτες των αντικειμένων που αναγνωρίστηκαν

```
for i, probs in enumerate(res):
    probs = np.squeeze(probs)
    top_ind = np.argsort(probs)[-args.number_top:][::-1]
    print("Image {}\n".format(args.input[i]))
    for id in top_ind:
        label = int(labels[class_id]) if args.labels else int(class_id)
        log.info(f'Found: label = {label}, confidence = {confidence:.2f}')
```

9.15 Παράδειγμα : Εύρεση προσώπου, αναγνώριση χαρακτηριστικών και αναγνώριση ταυτότητας

Στο παρακάτω παράδειγμα θα εκτελεστεί το face recognition demo. Θα δοθεί ως είσοδο ένα βίντεο ενός προσώπου και 3 πρόσφατες φωτογραφίες του ιδίου προσώπου με διαφορετικά ονόματα η κάθε μία. Επίσης, θα εκτελεστούν 3 νευρωνικά δίκτυα, ένα για την εύρεση του προσώπου, ένα για την εύρεση των χαρακτηριστικών του προσώπου και ένα για την αναγνώριση της ταυτότητας του προσώπου.

Για την σωστή εκτέλεση, απαιτούνται τα παρακάτω :

- Η είσοδος, η οποία θα είναι το βίντεο
- Νευρωνικό δίκτυο εύρεσης του προσώπου, το οποίο θα είναι το face-detection-retail-0044 (-m_fd)
- Νευρωνικό δίκτυο εύρεσης χαρακτηριστικών, το οποίο θα είναι το landmarks-regression-retail-0009 (-m_lm)
- Νευρωνικό δίκτυο αναγνώρισης της ταυτότητας, το οποίο θα είναι το face-reidentification-retail-0095 (-m_reid)
- Οι φωτογραφίες αντιστοίχισης της ταυτότητας (-fg)
- Οι συσκευές εκτέλεσης του κάθε δικτύου (-d_fd, -d_lm, -d_reid)

Στην προκειμένη περίπτωση, όλα τα νευρωνικά δίκτυα θα εκτελεστούν στο Intel Neural Compute Stick 2.

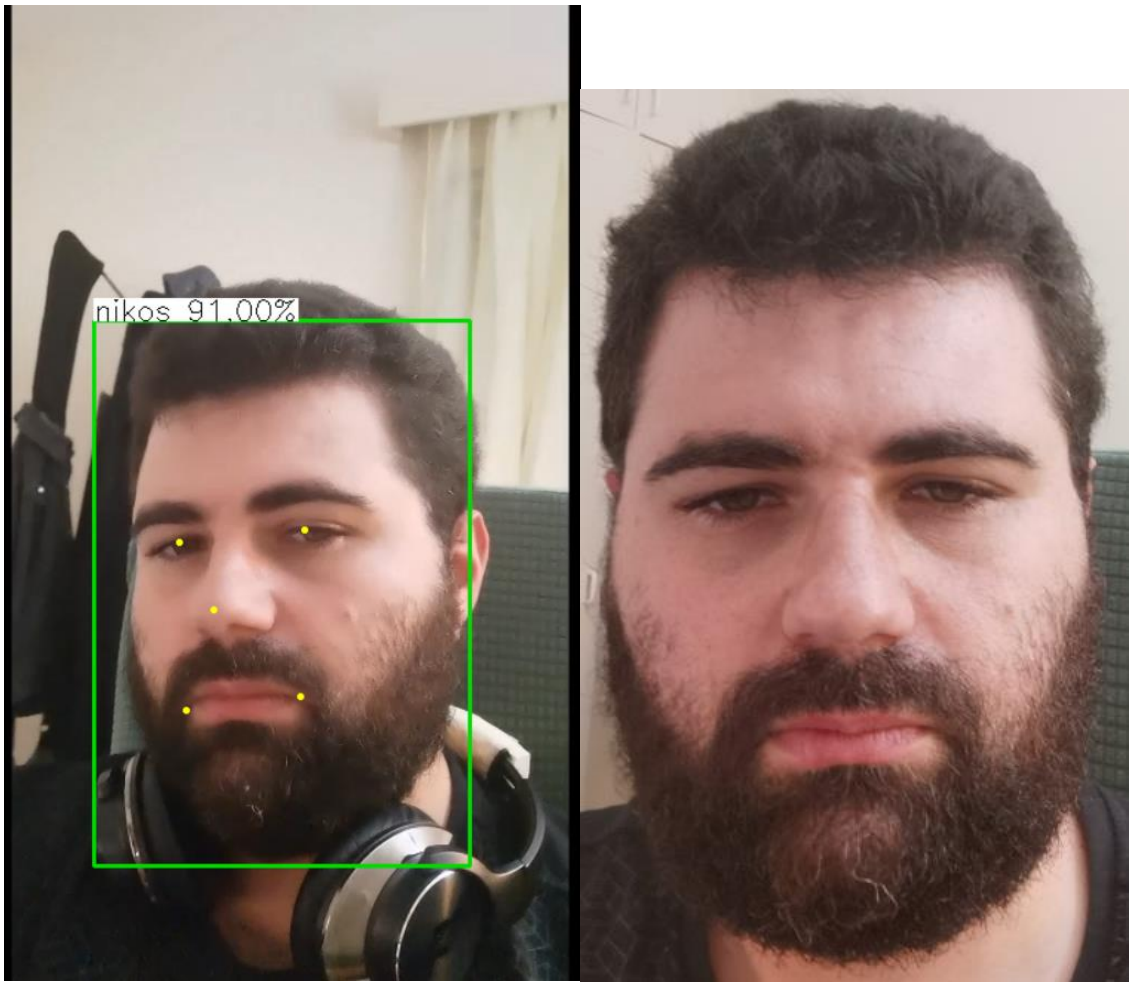
Στο σημείο αυτό θα πρέπει να σημειωθεί ότι το νευρωνικό δίκτυο που προτείνει η Intel στο συγκεκριμένο demo (το facial-landmarks-35-adas-0002) δεν είναι πλέον συμβατό. Επίσης, οι ελληνικοί χαρακτήρες στις διαδρομές ανάγνωσης δημιουργούν πρόβλημα και δεν εκτελείται.

Καθώς υπάρχουν 2 φωτογραφίες του προσώπου (με και χωρίς γυαλιά) με διαφορετικά ονόματα και στο βίντεο που θα χρησιμοποιηθεί τα γυαλιά αφαιρούνται και επανατοποθετούνται στο πρόσωπο, αναμένεται διαφορετική αναγνώριση ταυτότητας σε κάθε περίπτωση.

Η πλήρης εντολή για την εκτέλεση του παραδείγματος είναι η εξής :

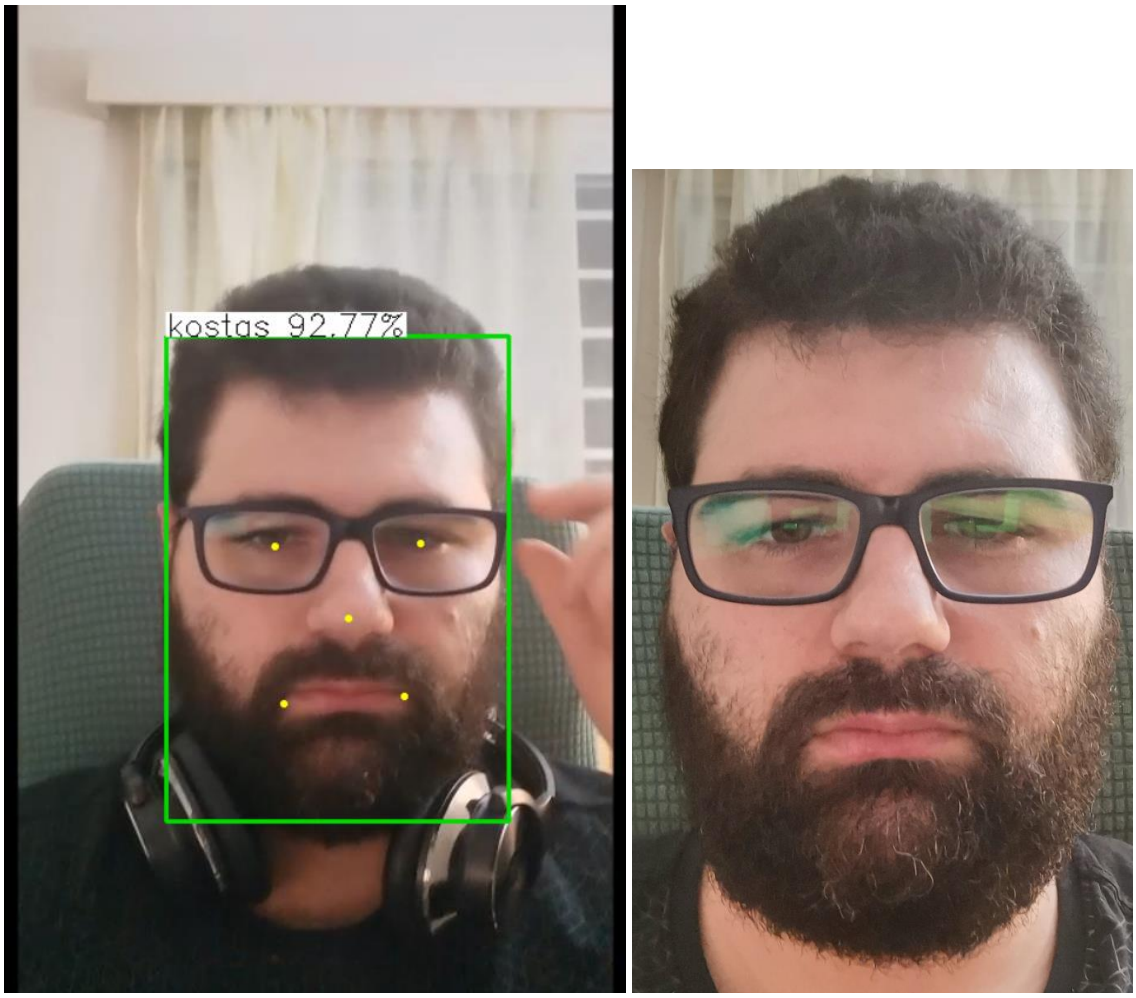
```
python "C:\Program Files
(x86)\Intel\openvino_2021.4.582\deployment_tools\inference_engine\demos\fac
e_recognition_demo\python\face_recognition_demo.py"
   -i "C:\FP16\test_face02H264.mp4"
   -m_fd "C:\Program Files
(x86)\Intel\openvino_2021.4.582\deployment_tools\tools\model_downloader\pub
lic\face-detection-retail-0044\face-detection-retail-0044-FP16.xml"
   -m_lm "C:\Program Files
(x86)\Intel\openvino_2021.4.582\deployment_tools\open_model_zoo\tools\downl
oader\intel\landmarks-regression-retail-0009\FP16\landmarks-regression-
retail-0009.xml"
   -m_reid "C:\FP16\face-reidentification-retail-0095.xml"
   --verbose
   -fg "C:\FP16\fg"
   -d_fd MYRIAD
   -d_lm MYRIAD
   -d_reid MYRIAD
```

Η 1η αναγνώριση χωρίς γυαλιά (αριστερά), αντιστοιχείται στην σωστή εικόνα nikos.jpg (δεξιά).



Καθ’ όλη την διάρκεια της εκτέλεσης αναγράφονται ο χρόνος απόκρισης (56,7 ms) και η διακίνηση εικόνων (17 FPS).

Στην συνέχεια του βίντεο, όταν έχουν φορεθεί τα γυαλιά, παρατηρούμε ότι και πάλι γίνεται επιτυχής ταυτοποίηση με την σωστή εικόνα (kostas.jpg).



Χρόνος απόκρισης (Latency) : 56,3 ms

Διακίνηση εικόνων (Throughput) : 17,1 FPS

Εκτέλεση με είσοδο βίντεο υψηλότερης ή χαμηλότερης ανάλυσης, δεν εμφανίζει σημαντική διαφορά στην απόκριση.

VPU 1080p	VPU 540p	CPU
Latency : 61,4 ms	Latency : 52,8 ms	Latency : 20,5 ms
Throughput : 15,3 FPS	Throughput : 16,4 FPS	Throughput : 35,0 FPS

Εκτέλεση και των 3 νευρωνικών δικτύων στην CPU (-d_fd CPU, -d_lm CPU, -d_reid CPU) δίνει καλύτερα αποτελέσματα με μείωση του χρόνου απόκρισης. Όμως, η διαφορά στην κατανάλωση ενέργειας σε σχέση με το INCS2 είναι μεγάλη (CPU 50W – NCS2 2W)

9.16 Παράδειγμα : Ανίχνευση πολλαπλών εισόδων (multi camera tracking)

Στο παρακάτω παράδειγμα θα εκτελεστεί το multi camera multi target tracking demo. Θα δοθούν ως είσοδοι 4 βίντεο και 2 νευρωνικά δίκτυα και θα εξαχθεί ως αποτέλεσμα η ανίχνευση και η αναγνώριση μοναδικών ατόμων, τα οποία θα διαχωρίζονται από τα υπόλοιπα με μοναδικούς χαρακτηρισμούς (ID 0, 1, 2).

Για την σωστή εκτέλεση, απαιτούνται τα παρακάτω :

- Οι είσοδοι, οι οποίες θα είναι τα βίντεο
- Νευρωνικό δίκτυο εύρεσης ανθρώπου, το οποίο θα είναι το person-detection-retail-0013 (-m_detector)
- Νευρωνικό δίκτυο επαναλαμβανόμενης αναγνώρισης ατόμου, το οποίο θα είναι το person-reidentification-retail-0277 (-m_reid)
- Η συσκευή εκτέλεσης των δικτύων (-d CPU)

Στην προκειμένη περίπτωση, όλα τα νευρωνικά δίκτυα θα εκτελεστούν στην CPU, καθώς δεν παρέχεται η δυνατότητα εκτέλεσης MULTI και το νευρωνικό δίκτυο reidentification δεν υποστηρίζεται από την VPU. Οι πληροφορίες συμβατότητας αντλήθηκαν από το αρχείο …\deployment_tools\open_model_zoo\intel_models\device_support.md

Model Name	CPU	GPU	MYRIA
person-detection-retail-0002	YES	YES	YES
person-detection-retail-0013	YES	YES	YES
person-reidentification-retail-0277	YES	YES	
person-reidentification-retail-0286	YES	YES	
person-reidentification-retail-0287	YES	YES	
person-reidentification-retail-0288	YES	YES	

Figure 25 –Λίστα υποστηριζόμενων συσκευών για συγκεκριμένα νευρωνικά δίκτυα

- Intel models device support [48]

Η πλήρης εντολή για την εκτέλεση του παραδείγματος είναι η εξής :

```
python "C:\Program Files
(x86)\Intel\openvino_2021.4.582\deployment_tools\open_model_zoo\demos\multi
_camera_multi_target_tracking_demo\python\multi_camera_multi_target_trackin
g_demo.py"
    --input "C:\FP16\test11.mp4" "C:\FP16\test22.mp4" "C:\FP16\test33.mp4"
"C:\FP16\test44.mp4"
    --m_detector "C:\FP16\person-detection-retail-0013\FP32\person-
detection-retail-0013.xml"
    --m_reid "C:\FP16\person-reidentification-retail-0277\FP32\person-
reidentification-retail-0277.xml"
    --config "C:\Program Files
(x86)\Intel\openvino_2021.4.582\deployment_tools\open_model_zoo\demos\multi
_camera_multi_target_tracking_demo\python\configs\person.py"
    -d CPU
```

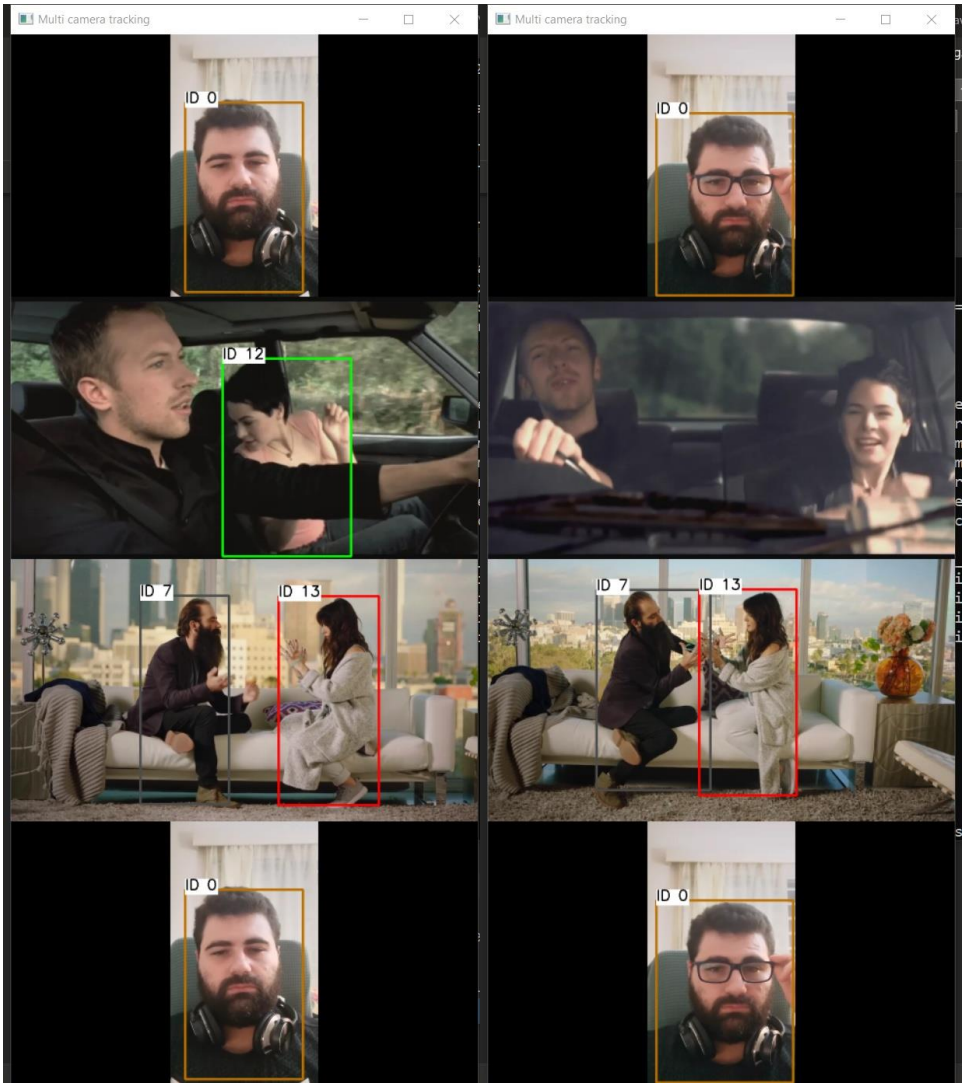


Figure 26 - Συνεχής ανίχνευσης πολλαπλών εισόδων

Το αποτέλεσμα είναι ικανοποιητικό, αφού αναγνωρίζεται το πρόσωπο κανονικά σε 2 εισόδους (1ο και 4ο βίντεο) ως ίδιο ID, ενώ στο 3ο βίντεο, παραμένει η αναγνώριση σωστή όσο δεν αλλάζει δραστικά το περιεχόμενο. Στο 2ο βίντεο δεν υπάρχει ικανοποιητική αναγνώριση λόγω των διαφορετικών γωνιών και των εμποδίων (ανεμοθώρακας, σκιές).

9.17 Παράδειγμα : Σύγκριση εκτέλεσης εκτίμησης ανθρώπινης πόζας σε CPU και VPU

Στο παρακάτω παράδειγμα θα εκτελεστεί το human estimation pose demo. Θα δοθεί ως είσοδος 1 βίντεο και 1 νευρωνικό δίκτυο και θα ληφθεί ως αποτέλεσμα η εκτίμηση της πόζας του κάθε ατόμου που αναγνωρίζεται στο κάθε στιγμιότυπο του βίντεο ως ένας εικονικός και απλοϊκός σκελετός

Για την σωστή εκτέλεση, απαιτούνται τα παρακάτω :

- Η είσοδος, η οποία θα είναι το βίντεο
- Νευρωνικό δίκτυο αναγνώρισης και εκτίμησης στάσης ανθρωπίνου σώματος, το οποίο θα είναι το human-pose-estimation-0007
- Η δήλωση της αρχιτεκτονικής του νευρωνικού δικτύου (--architecture_type ae)
- Η συσκευή εκτέλεσης των δικτύων (-d CPU / -d MYRIAD)

Πέρα των παραπάνω, δύναται η επιλογή και άλλων ρυθμίσεων, όπως :

- Τον αριθμό των παράλληλων αιτημάτων προς Inference
- Τον αριθμό των πυρήνων στους οποίους θα γίνεται το Inferencing (μόνο για την εκτέλεση σε CPU)

Το νευρωνικό δίκτυο που θα χρησιμοποιηθεί, είναι συμβατό για εκτέλεση και στην CPU και στην VPU

Model Name	CPU	GPU	MYRIAD
human-pose-estimation-0007	YES	YES	YES

Η πλήρης εντολή για την εκτέλεση του παραδείγματος είναι η εξής :

```
python "C:\Program Files (x86)\Intel\openvino_2021.4.582\deploy-
ment_tools\open_model_zoo\demos\human_pose_estimation_demo\python\hu-
man_pose_estimation_demo.py"
    --model "C:\FP16\human-pose-estimation-0007\FP32\human-pose-estimation-
0007.xml"
    --architecture_type ae
    --input "C:\FP16\shoppingmall.mp4"
    -d CPU
    -nireq 1
    -nthreads 1
```

Θα γίνουν δοκιμές με αλλαγή στα παράλληλα αιτήματα Inferencing, στους πυρήνες χρήσης και στην συσκευή εκτέλεσης. Σε κάθε εκτέλεση θα υπάρχει και στιγμιότυπο από το αποτέλεσμα. Στο τέλος θα παρουσιαστούν συγκεντρωμένοι οι χρόνοι και οι μέγιστες διακινήσεις εικόνων ανά περίπτωση.

9.17.1 nireq = 1, nthreads = 1, CPU

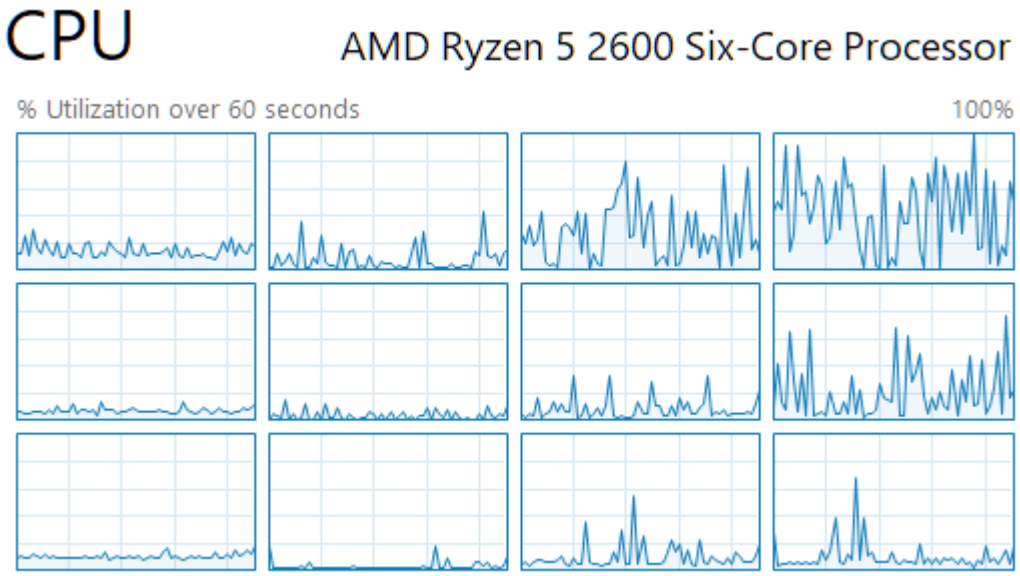


Figure 27 - Εκτίμηση πόζας & χρήση CPU για 1 αίτημα Inferencing σε 1 πυρήνα

9.17.2 nireq = 1, nthreads = 12, CPU

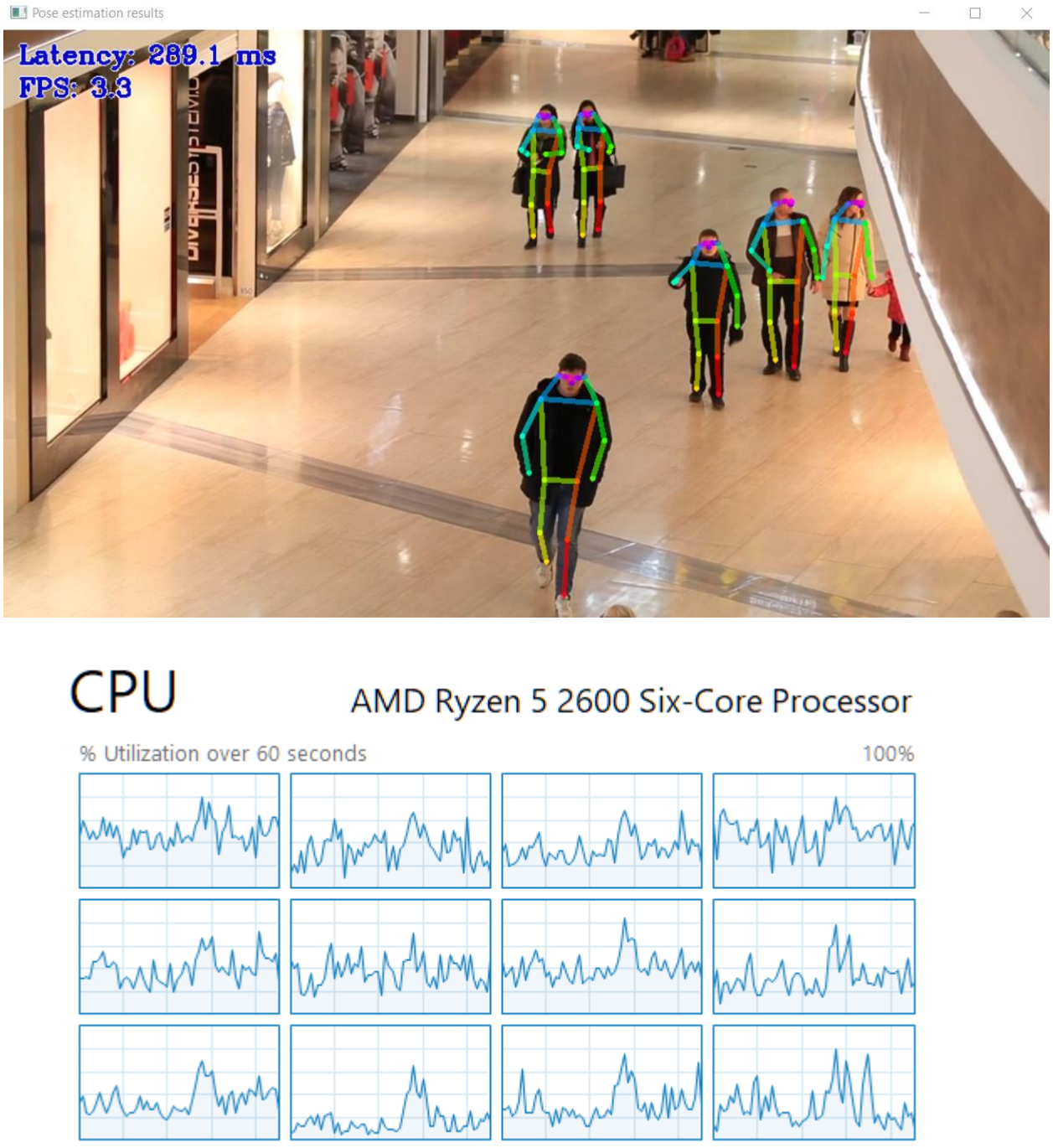


Figure 28 - Εκτίμηση πόζας & χρήση CPU για 1 αίτημα Inferencing σε 12 πυρήνες

9.17.3 nireq = 1, nthreads = 40, CPU

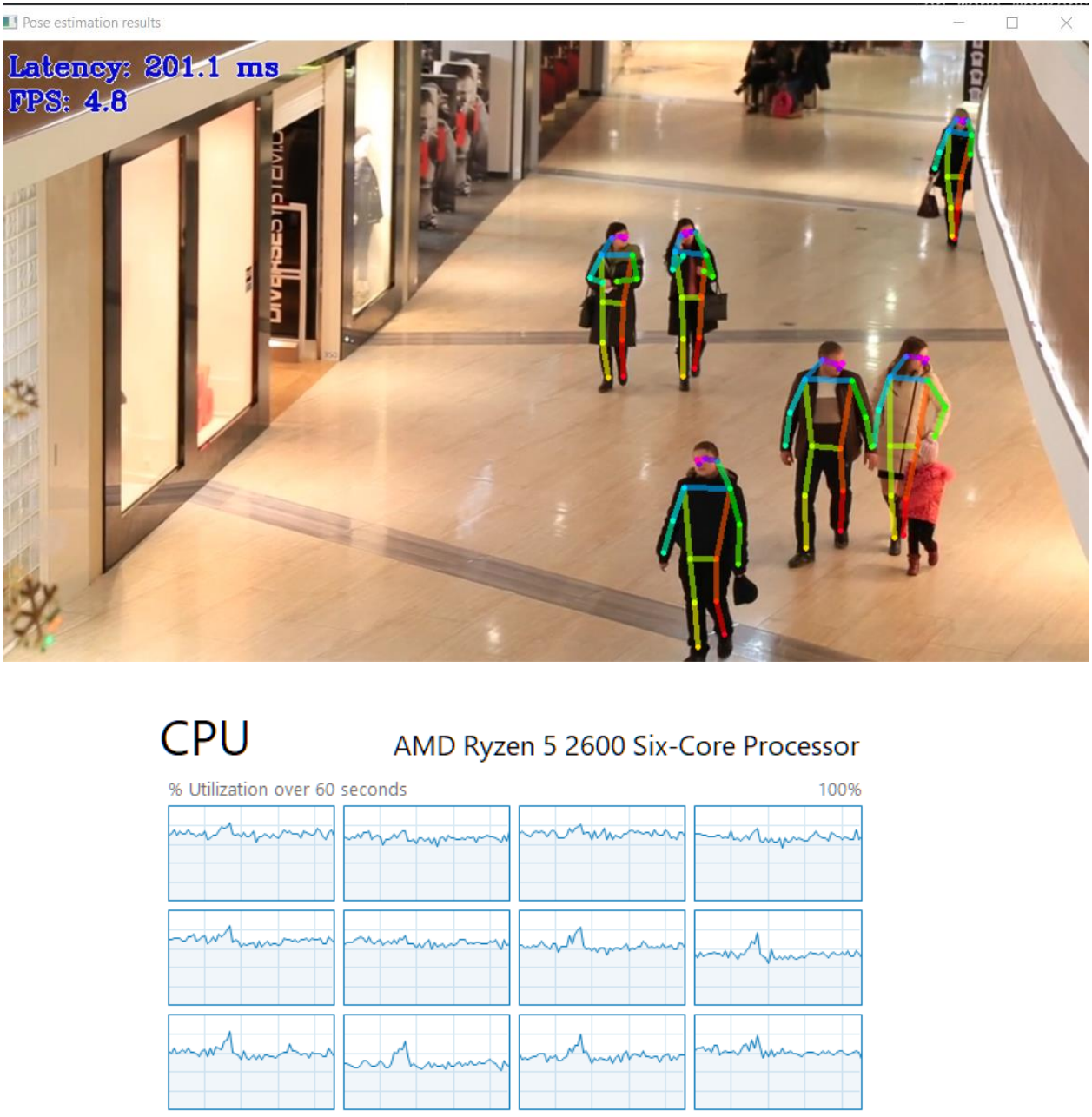


Figure 29 - Εκτίμηση πόζας & χρήση CPU για 1 αίτημα Inferencing σε 40 πυρήνες

9.17.4 nireq = 8, nthreads = 1, CPU

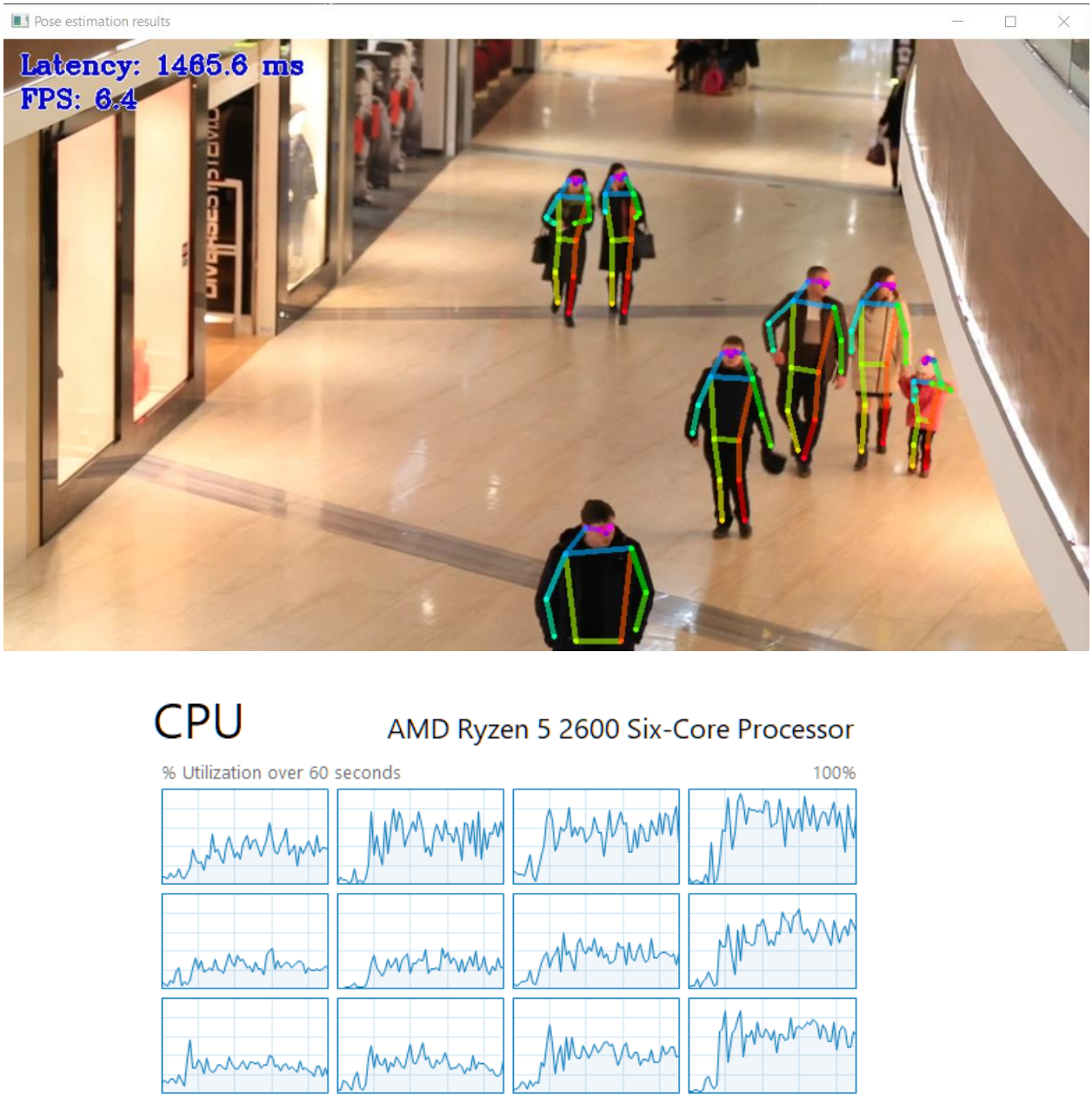


Figure 30 - Εκτίμηση πόζας & χρήση CPU για 8 αιτημάτων Inferencing σε 1 πυρήνα

9.17.5 nireq = 8, nthreads = 12, CPU

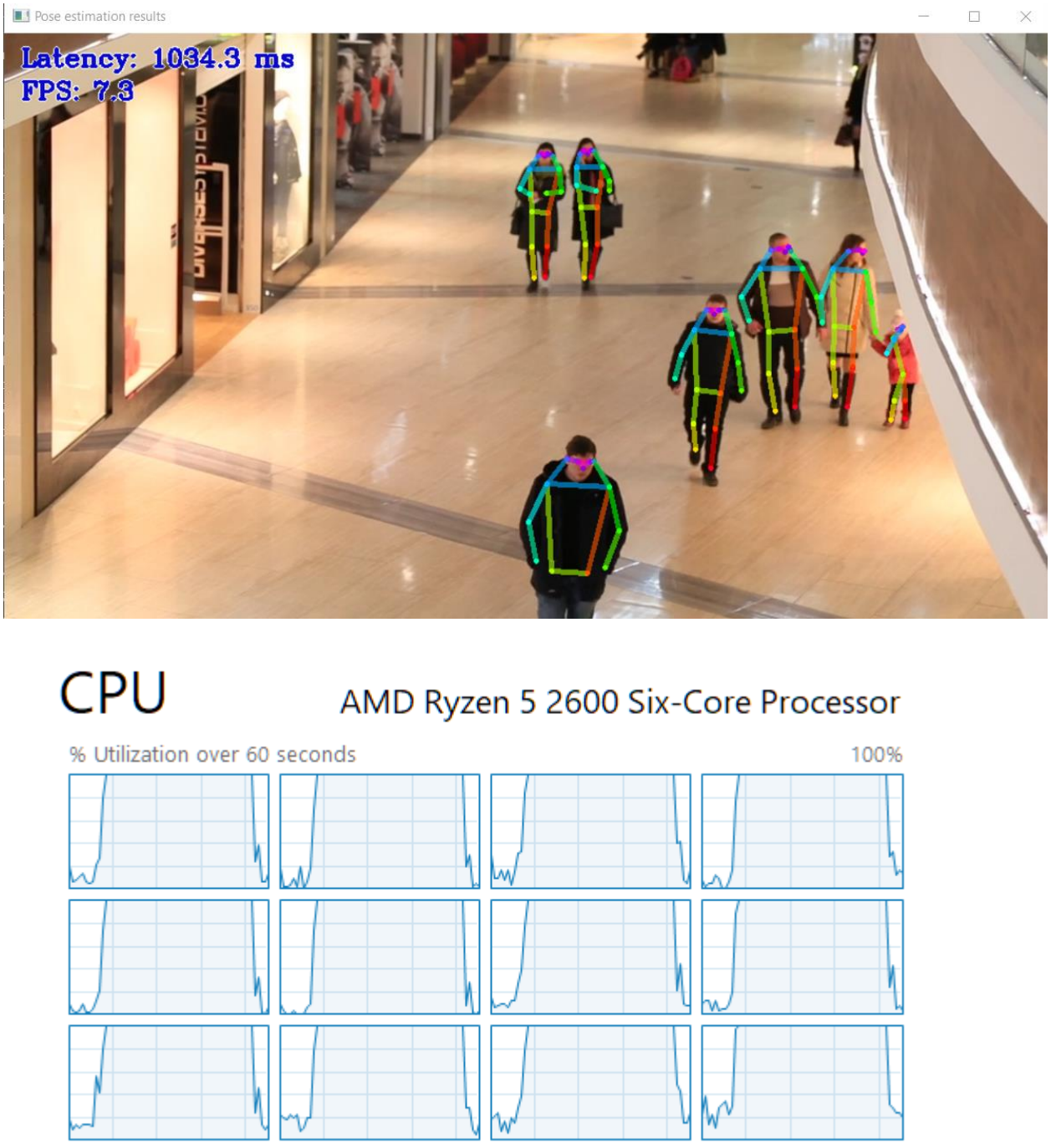


Figure 31 - Εκτίμηση πόζας & χρήση CPU για 8 αιτημάτων Inferencing σε 12 πυρήνες

9.17.6 nireq = 1, MYRIAD

Figure 32 - Εκτίμηση πόζας & χρήση CPU για 1 αίτημα Inferencing στην VPU

9.17.7 nireq = 8, MYRIAD

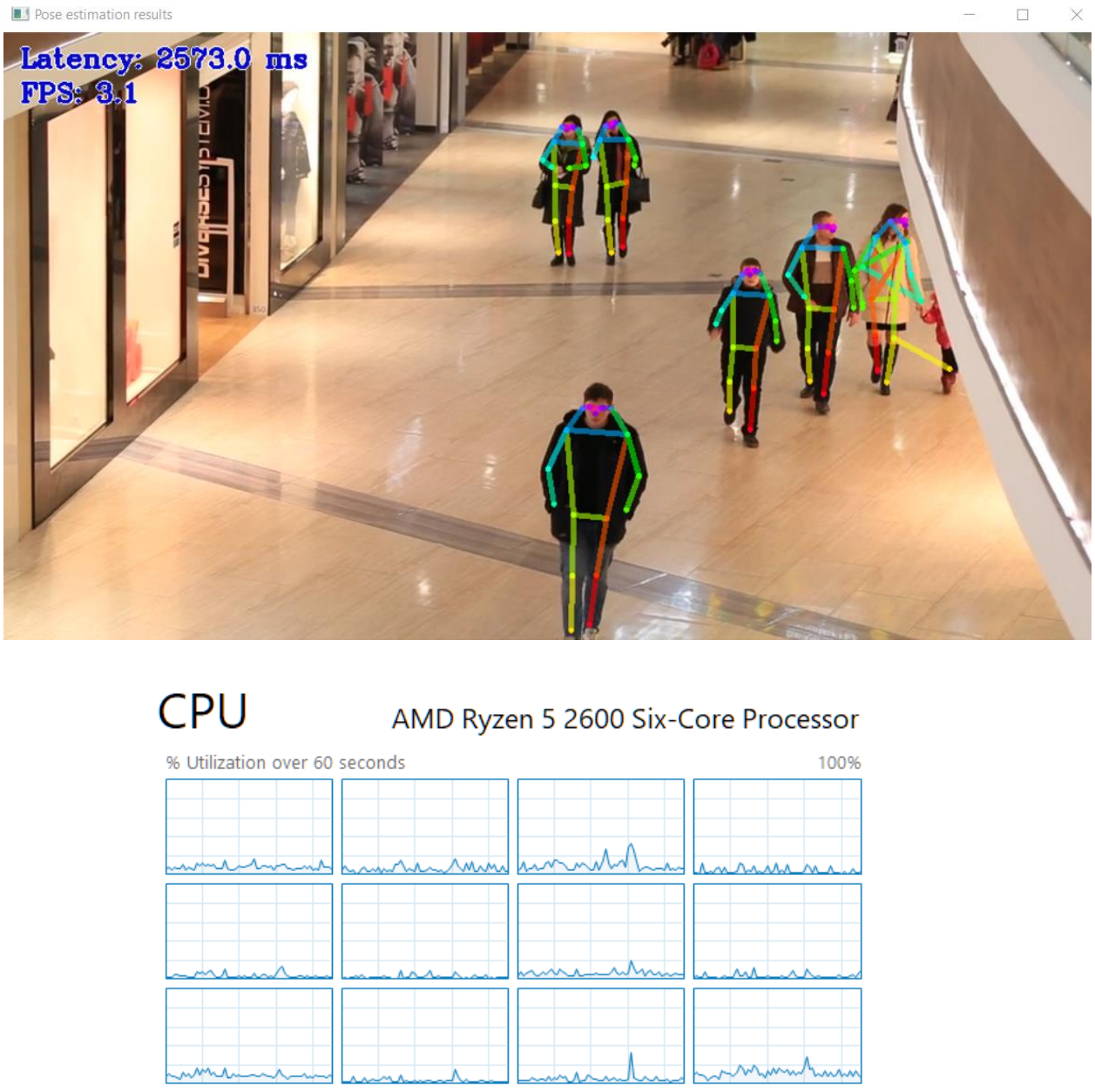


Figure 33 - Εκτίμηση πόζας & χρήση CPU για 8 αιτημάτων Inferencing στην VPU

9.17.8 nireq = 3, MYRIAD

Figure 34 - Εκτίμηση πόζας & χρήση CPU για 3 αιτημάτων Inferencing στην VPU

9.17.9 nireq = 8, nthreads = 12 Multi CPU & MYRIAD

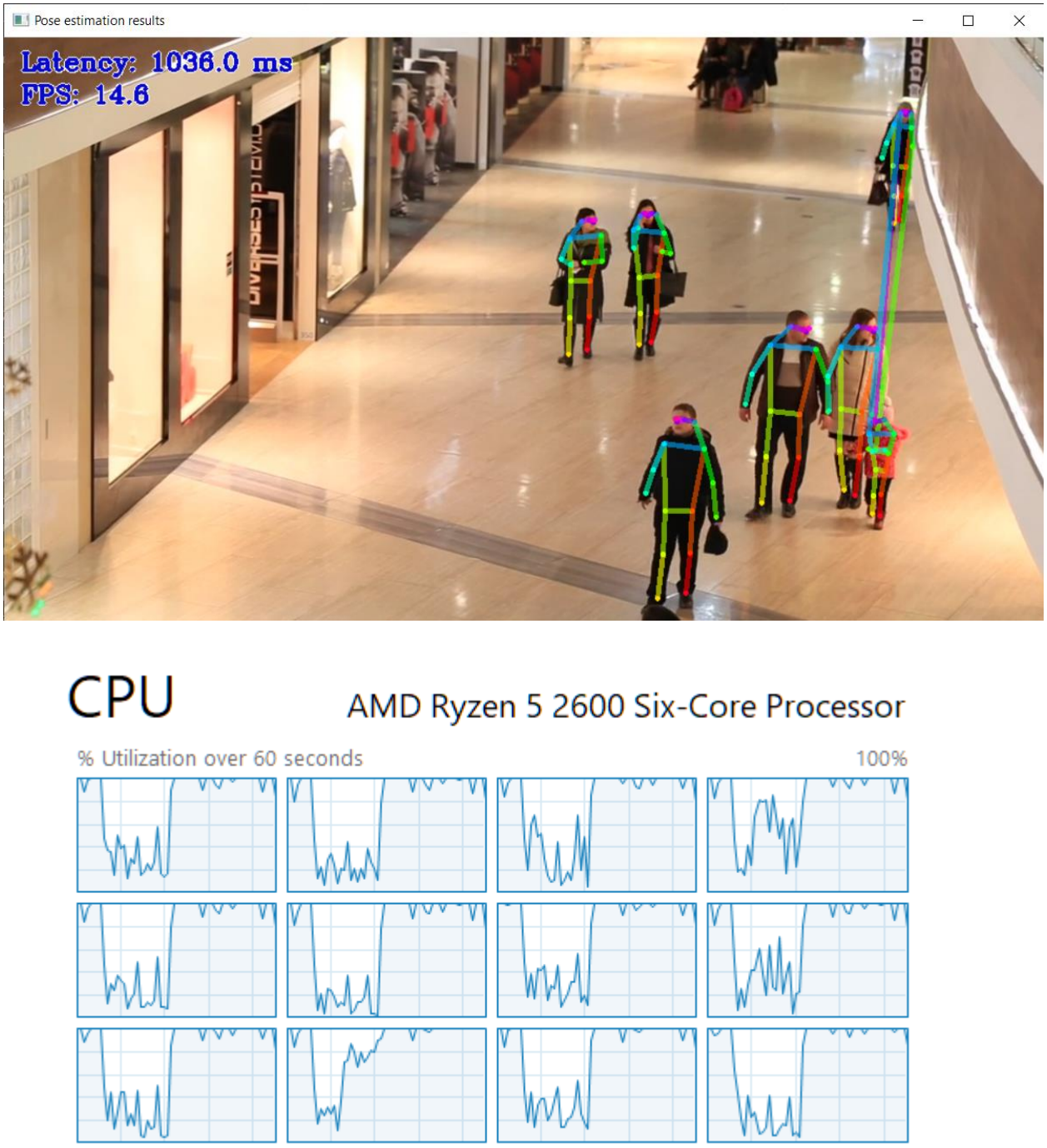


Figure 35 - Εκτίμηση πόζας & χρήση CPU για 8 αιτημάτων Inferencing σε 12 πυρήνες και VPU

Κατά την εκτέλεση σε 2 συσκευές, υπήρξε μεγαλύτερη διακύμανση στην απόκριση και την διακίνηση εικόνων, σε σχέση με την εκτέλεση σε 1 συσκευή. Το εύρος που παρατηρήθηκε, κατά προσέγγιση, είναι 700ms – 1200ms στον χρόνο απόκρισης και 6,5 – 15 FPS στην διακίνηση εικόνων.

9.17.10 Βέλτιστες τιμές απόκρισης και διακίνησης εικόνων

Ρυθμίσεις	Απόκριση (latency) σε ms	Διακίνηση (throughput) σε FPS
Inf. requests = 1 threads = 1 CPU	662	1,5
Inf. requests = 1 threads = 12 CPU	289	3,3
Inf. requests = 1 threads = 40 CPU	201	4,8
Inf. requests = 8 threads = 1 CPU	1465	6,4
Inf. requests = 8 threads = 12 CPU	1034	7,3
Inf. requests = 1 MYRIAD	685	1,4
Inf. requests = 3 MYRIAD	2573	3,1
Inf. requests = 8 MYRIAD	922	3,2
Inf. requests = 8 threads = 12 CPU & MYRIAD	1036	14,6

Figure 36 - Σύγκριση εκτελέσεων παραδείγματος εκτίμησης πόζας

9.18 Εγκατάσταση OpenVINO στο Raspberry Pi

Καθώς το Raspberry Pi πρόκειται για πλατφόρμα ανάπτυξης, η Intel παρέχει μόνο το Inference Engine, την τροποποιημένη έκδοση της OpenCV για υλικό Intel και κάποια παραδείγματα.

Ο Model Optimizer δεν περιλαμβάνεται, καθώς προορίζεται για χρήση από πλήρη συστήματα στα οποία γίνεται η ανάπτυξη του λογισμικού.

9.18.1 Λήψη OpenVINO

Αρχικά πρέπει να ληφθεί η πιο πρόσφατη έκδοση του OpenVINO για Raspbian και συγκεκριμένα το αρχείο **l_openvino_toolkit_runtime_raspbian_p_<version>.tgz**. Αυτό γίνεται είτε απευθείας από την ιστοσελίδα της Intel, είτε μέσω terminal.

Terminal :

```
wget
https://storage.openvinotoolkit.org/repositories/openvino/packages/2021.4/l
_openvino_toolkit_runtime_raspbian_p_2021.4.582.tgz
```

Σε περίπτωση που δεν είναι εγκατεστημένο το wget, χρειάζεται να γίνει εγκατάσταση με την εντολή sudo apt install wget

Το αρχείο θα αποθηκευτεί στον φάκελο που βρίσκεται το terminal. Για επιλογή διαφορετικού φακέλου, η σύνταξη της εντολής είναι :

```
wget -P <destination_folder> <url openvino>
```

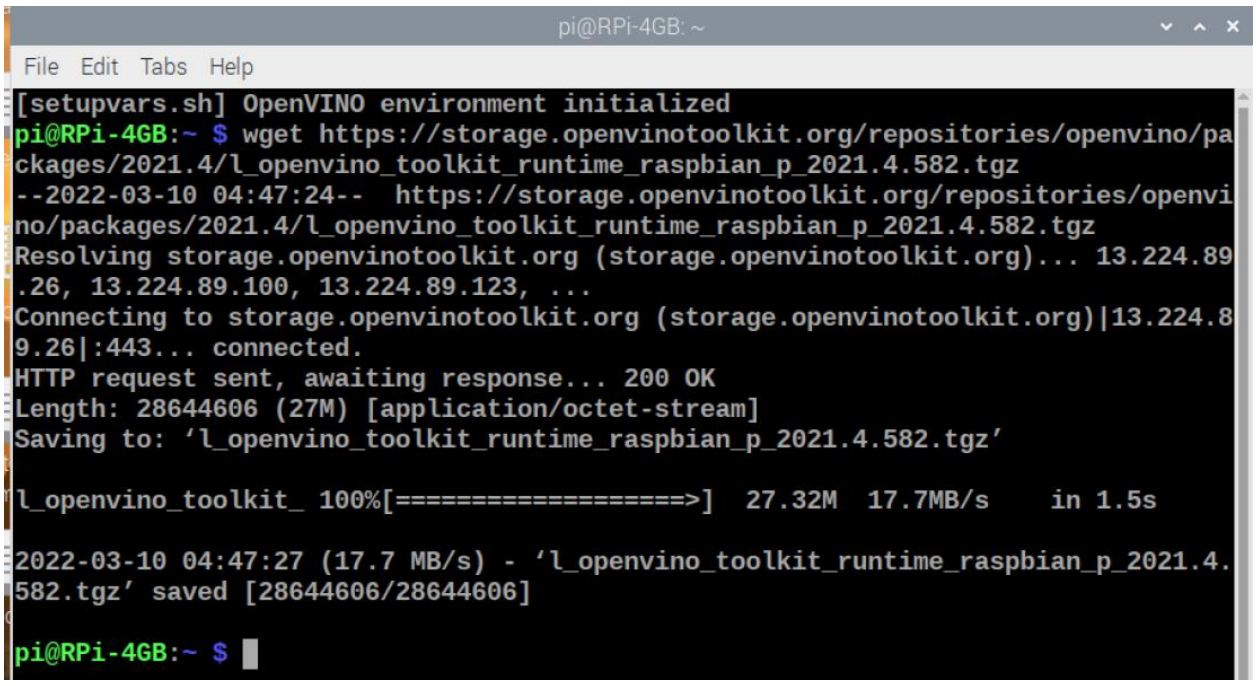


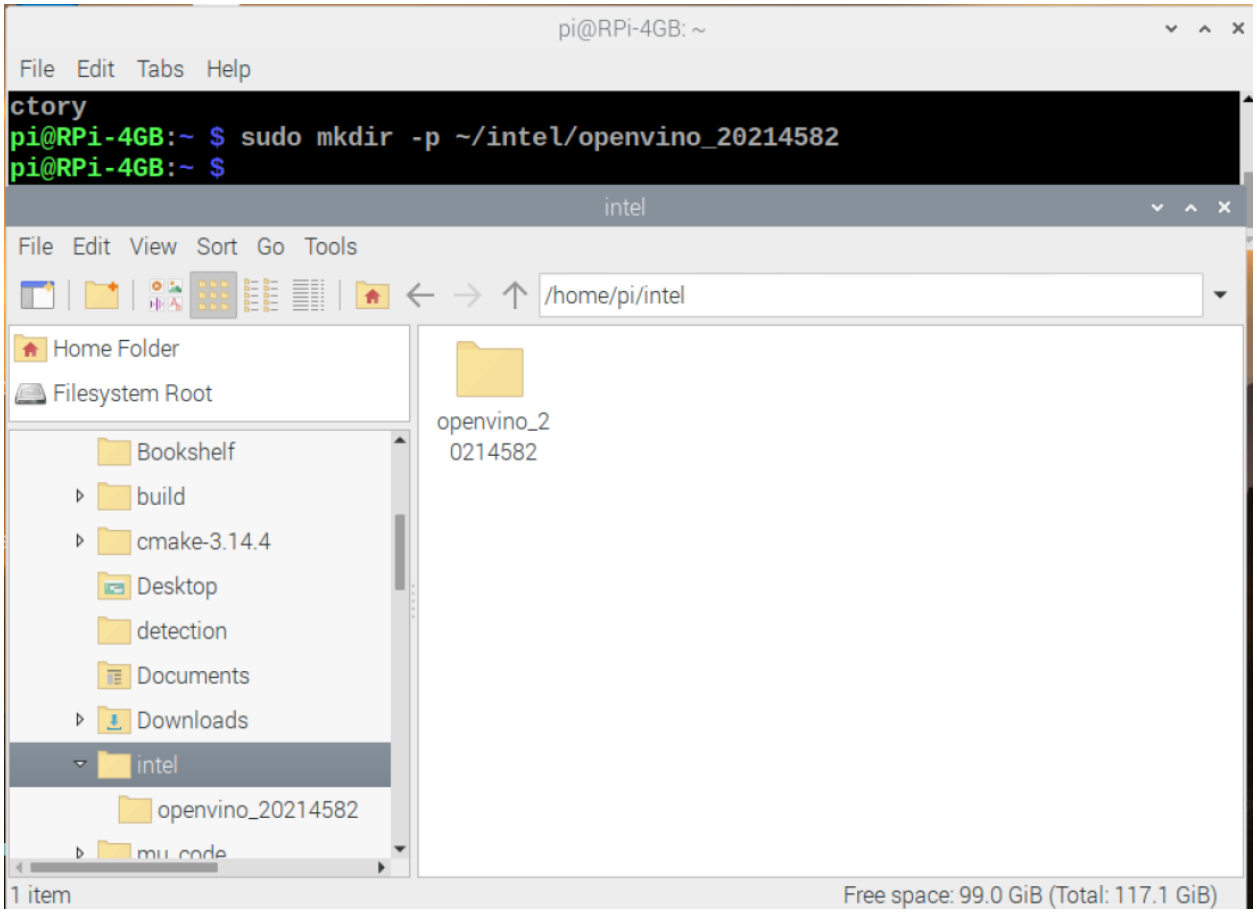
Figure 37 - Αποτέλεσμα εκτέλεσης εντολής λήψης OpenVINO σε RPi

9.18.2 Δημιουργία φακέλου OpenVINO

Στην συνέχεια, δημιουργείται ο φάκελος που θα γίνει η εγκατάσταση του OpenVINO.

```
sudo mkdir -p ~/intel/openvino_20214582
```

Εφόσον δεν παρουσιαστεί κανένα σφάλμα, θα έχει δημιουργηθεί ο φάκελος κανονικά.

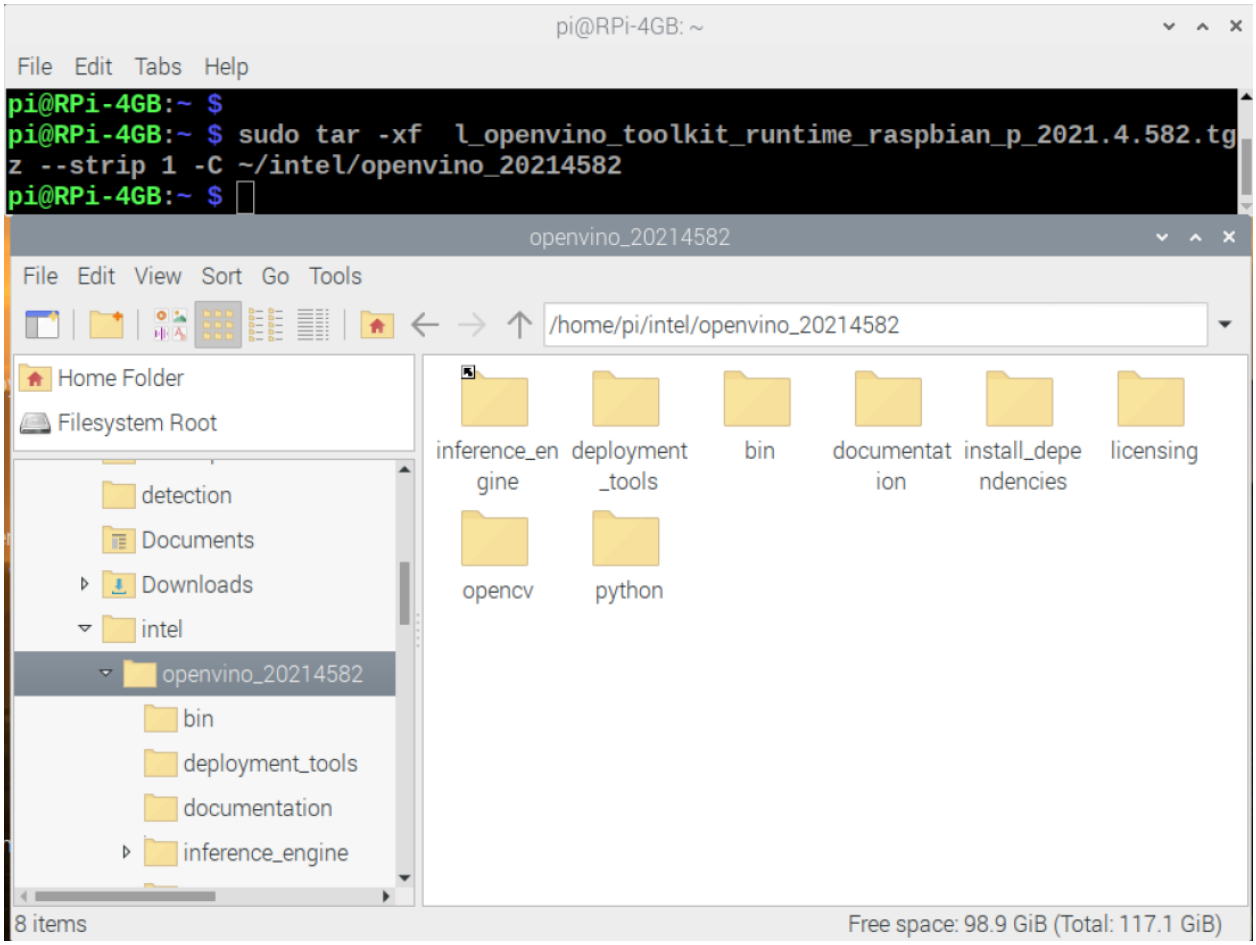


9.18.3 Αποσυμπίεση OpenVINO

Θα γίνει αποσυμπίεση του αρχείου που λήφθηκε, στον παραπάνω φάκελο.

```
sudo tar -xf  l_openvino_toolkit_runtime_raspbian_p_<version>.tgz --strip 1
-C ~/intel/openvino_20214582
```

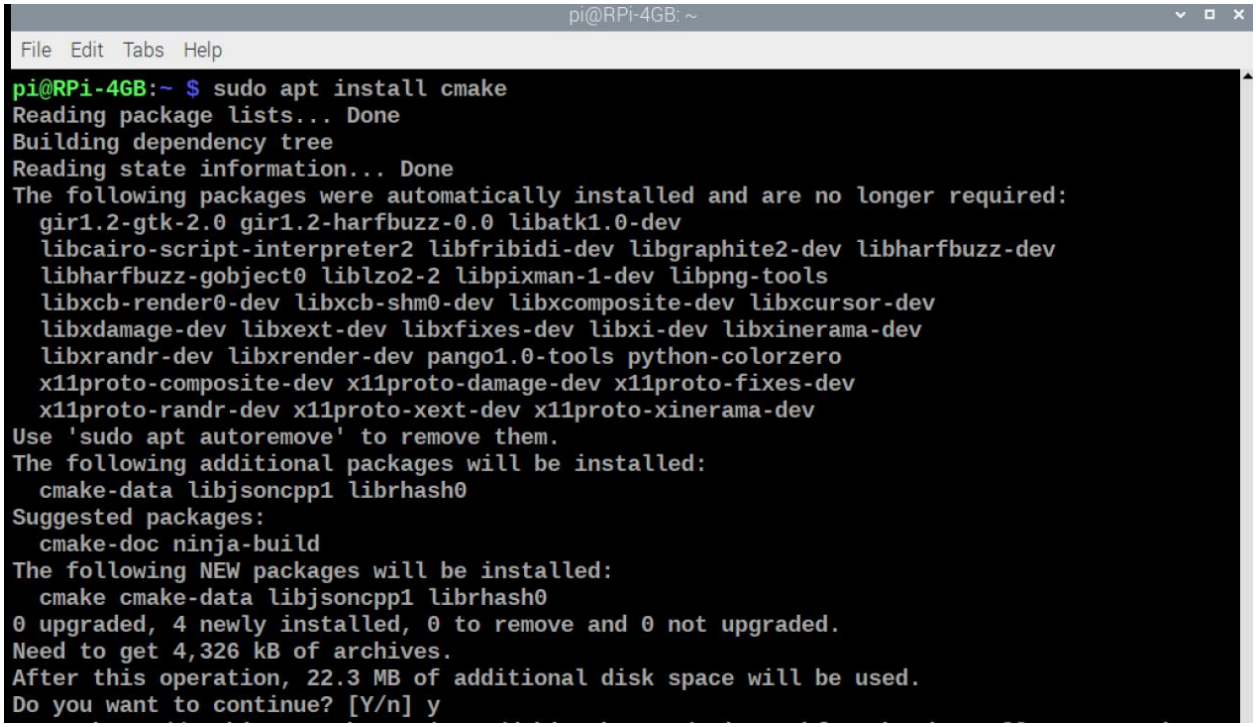
Εφόσον δεν παρουσιαστεί κανένα σφάλμα, θα έχουν εμφανιστεί τα αρχεία κανονικά.



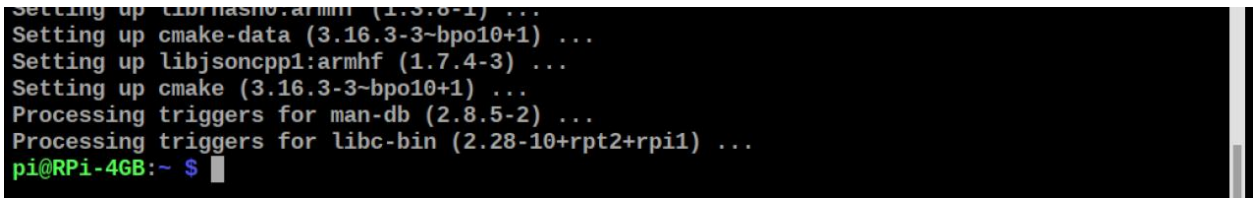
9.18.4 Εγκατάσταση CMAKE

Η εγκατάσταση του CMAKE γίνεται με την εντολή

```
sudo apt install cmake
```



Η εγκατάσταση είναι επιτυχής όταν φτάσει στο 100% η μπάρα εξέλιξης εγκατάστασης και δεν εμφανιστεί κάποιο σφάλμα.



9.18.5 Ρύθμιση μεταβλητών περιβάλλοντος

Όπως και στην εγκατάσταση των Windows, έτσι και στο Raspbian θα πρέπει να αρχικοποιούνται οι απαραίτητες μεταβλητές του OpenVINO κάθε φορά που ανοίγει ένα terminal.

Η εντολή με την οποία γίνεται η αρχικοποίηση είναι

```
source ~/intel/openvino_20214582/bin/setupvars.sh
```

Αν είναι επιτυχής, εμφανίζεται το παρακάτω μήνυμα.

```
pi@RPi-4GB:~ $ source ~/intel/openvino_20214582/bin/setupvars.sh
[setupvars.sh] OpenVINO environment initialized
pi@RPi-4GB:~ $
```

Επειδή η αρχικοποίηση του OpenVINO χάνεται κάθε φορά που κλείνει το παράθυρο terminal, μπορούμε να την μονιμοποιήσουμε με την παρακάτω εντολή

```
echo "source ~/intel/openvino_20214582/bin/setupvars.sh" >> ~/.bashrc
```

ΠΡΟΣΟΧΗ στο διπλό βέλος (>>) το οποίο είναι για την προσθήκη στο αρχείο bash, αντί του μονού που είναι για την αντικατάσταση ΟΛΩΝ των περιεχομένων του αρχείου

```
pi@RPi-4GB:~ $ echo "source ~/intel/openvino_20214582/bin/setupvars.sh" >> ~/.bashrc
pi@RPi-4GB:~ $
```

Αν η προσθήκη ήταν επιτυχής, ανοίγοντας ένα νέο terminal, θα εμφανίζεται το μήνυμα αρχικοποίησης του OpenVINO

```
pi@RPi-4GB: ~
File  Edit  Tabs  Help
[setupvars.sh] OpenVINO environment initialized
[setupvars.sh] OpenVINO environment initialized
[setupvars.sh] OpenVINO environment initialized
pi@RPi-4GB:~ $
```

Στην συγκεκριμένη περίπτωση έγινε προσθήκη 3 φορές. Για την αφαίρεση, αρκεί η διαγραφή των περιττών εγγραφών από το αρχείο bashrc. Θα πρέπει να ανοίξει με έναν επεξεργαστή κειμένου και να αφαιρεθούν οι εντολές, στο τέλος του αρχείου.

- How to bashrc [49]

```
pi@RPi-4GB:~ $ nano ~/.bashrc
```

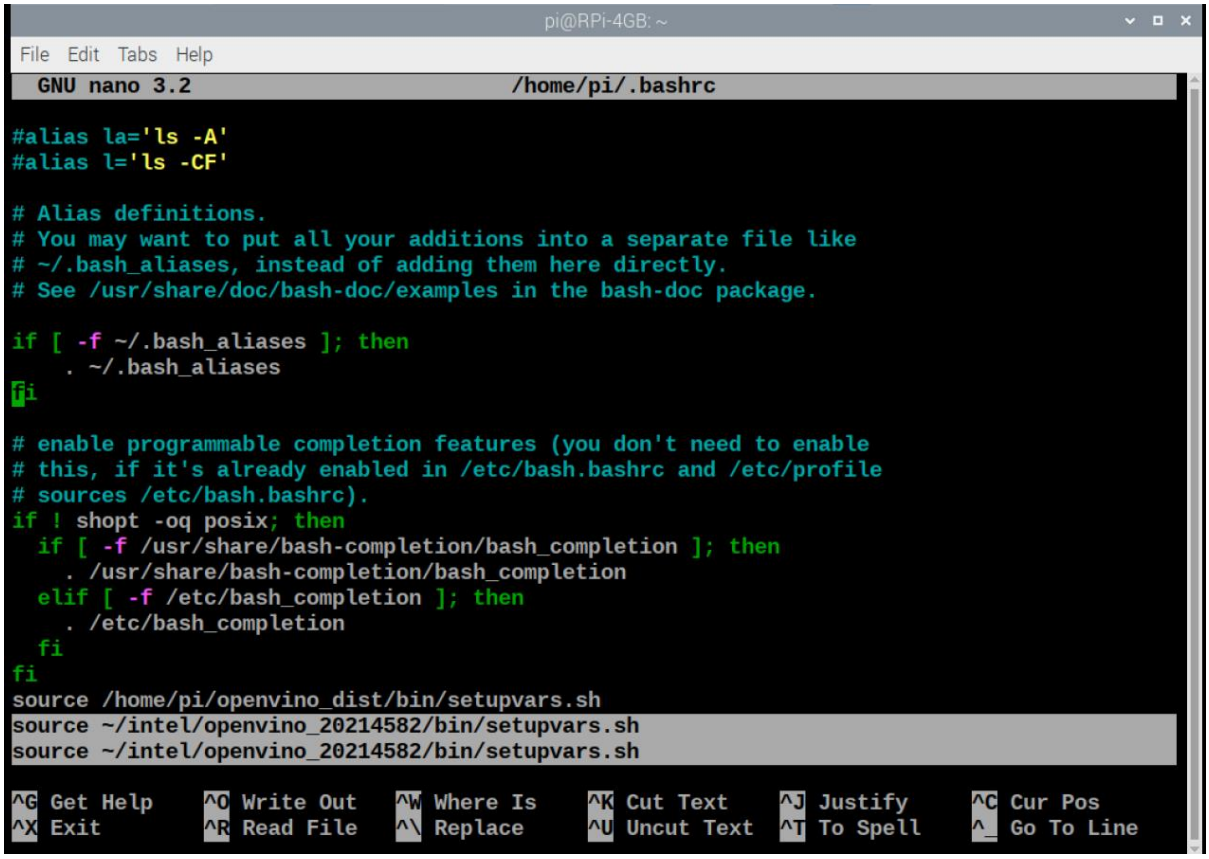


Figure 38 - Εγγραφή setupvars στο bash

ΠΡΟΣΟΧΗ να μην διαγραφεί κάτι άλλο από το αρχείο. Σε λανθασμένη διαγραφή του “f” που βρίσκεται στο τέλος, σε κάθε νέο terminal βγάζει το σφάλμα .bashrc: line 116: syntax error: unexpected end of file

9.18.6 Προσθήκη USB rules για το Neural Compute Stick 2

Αρχικά θα χρειαστεί να προστεθεί ο χρήστης στην ομάδα των χρηστών (users) με την εντολή

```
sudo usermod -a -G users "$(whoami)"
```

```
pi@RPi-4GB:~ $ sudo usermod -a -G users "$(whoami)"
```

Στην συνέχεια θα χρειαστεί αποσύνδεση και επανασύνδεση στο σύστημα.

Η εγκατάσταση των USB rules θα γίνει με την εντολή

```
sh ~/intel/openvino_20214582/install_dependencies/install_NCS_udev_rules.sh
```

Μετά την επιτυχή εκτέλεση, μπορούμε να συνδέσουμε το NCS2 στο Raspberry Pi.

```
pi@RPi-4GB:~ $ sh ~/intel/openvino_20214582/install_dependencies/install_NCS_udev_rules.sh
Updating udev rules...
Udev rules have been successfully installed.
pi@RPi-4GB:~ $
```

9.18.7 Ετοιμασία και εκτέλεση παραδείγματος ανίχνευσης αντικειμένου

9.18.7.1 Φάκελος build

Αρχικά θα χρειαστεί να δημιουργηθεί ένας φάκελος για την ετοιμασία των αρχείων.

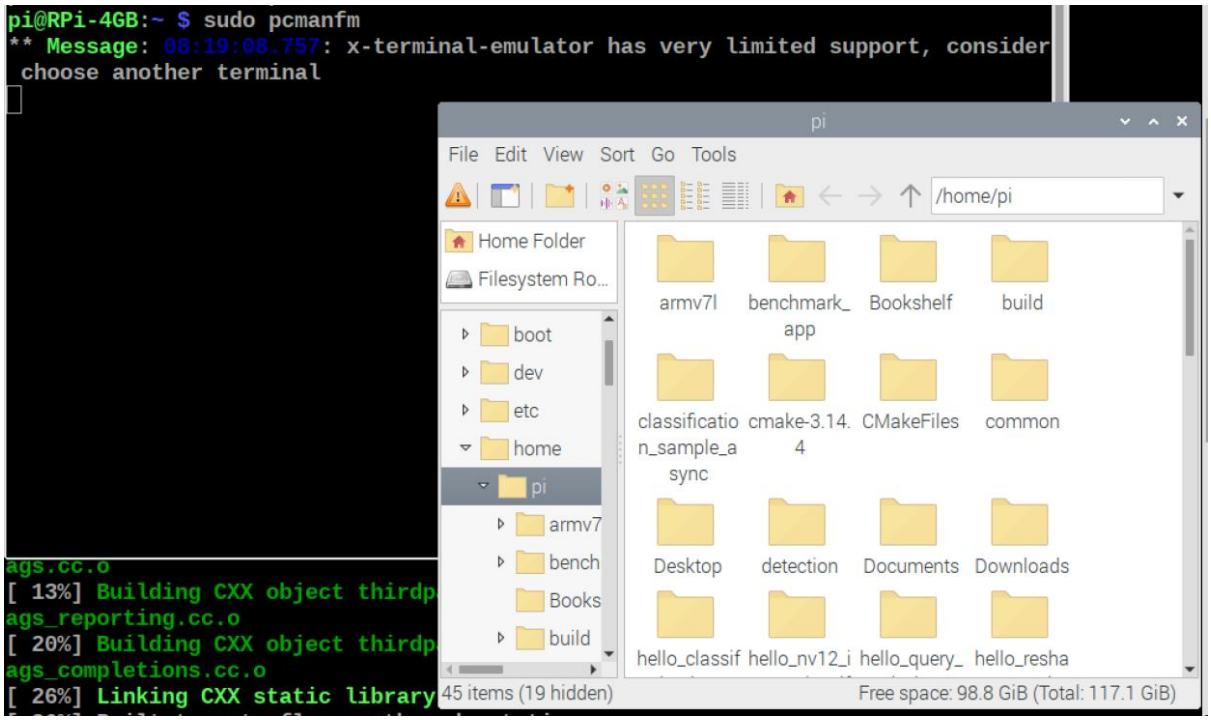
Δημιουργείται ο φάκελος build, πηγαίνοντας στον φάκελο ~/intel.

```
pi@RPi-4GB:~/intel/openvino_20214582/inference_engine/samples $ cd ~/intel/openvino_2021458
2/inference_engine/samples/
pi@RPi-4GB:~/intel/openvino_20214582/inference_engine/samples $ sudo mkdir build
pi@RPi-4GB:~/intel/openvino_20214582/inference_engine/samples $ cd build/
pi@RPi-4GB:~/intel/openvino_20214582/inference_engine/samples/build $
```

Ο φάκελος θα πρέπει να δημιουργηθεί σε χώρο με δικαιώματα εγγραφής και διαγραφής από τον χρήστη. Ο πιο εύκολος τρόπος για να επιτευχθεί αυτό είναι να ανοίξει ο File Manager με δικαιώματα root user.

```
sudo pcmanfm
```

Δεξί κλικ στον φάκελο που απαιτείται -> Properties -> Permissions -> Change content -> Anyone



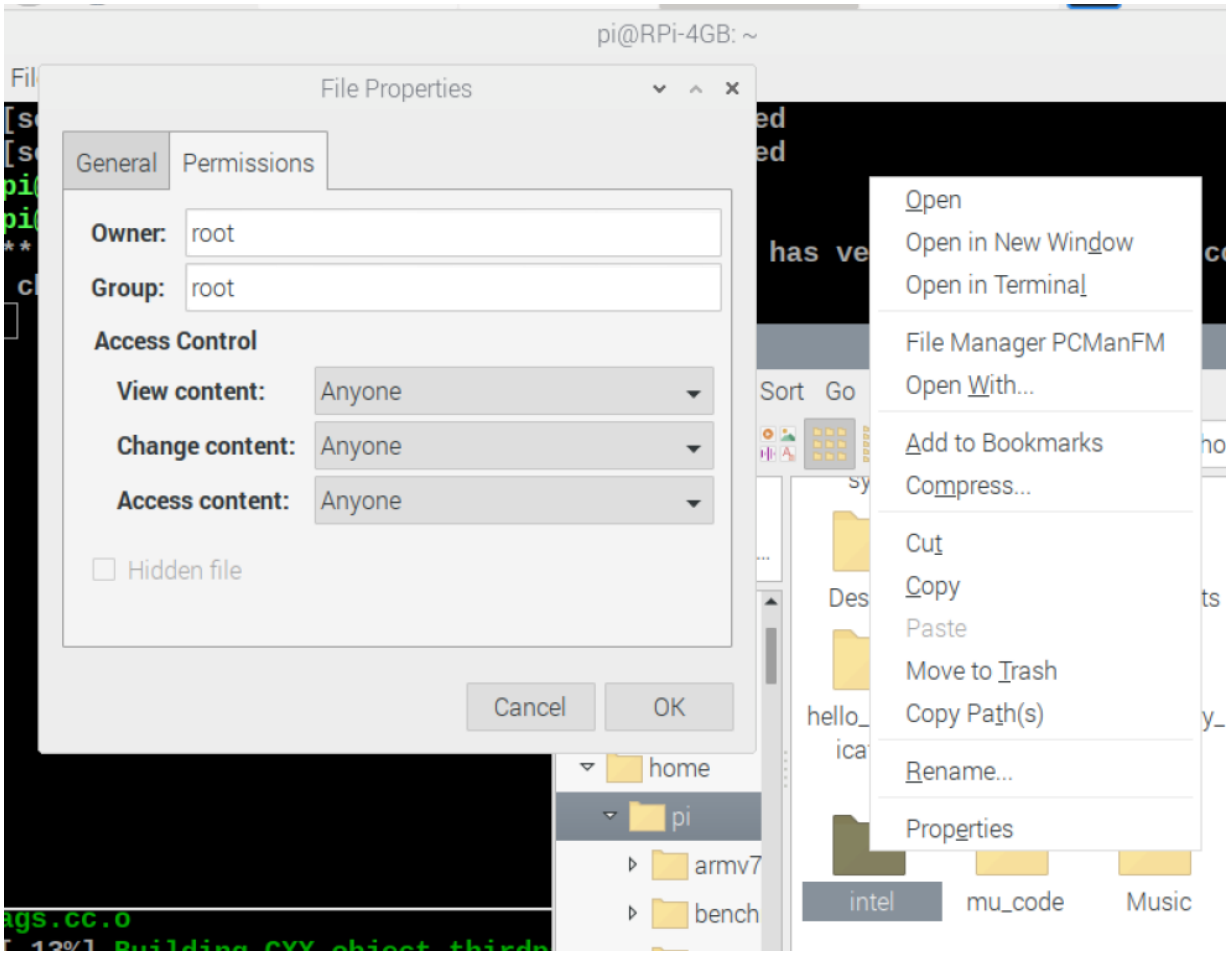
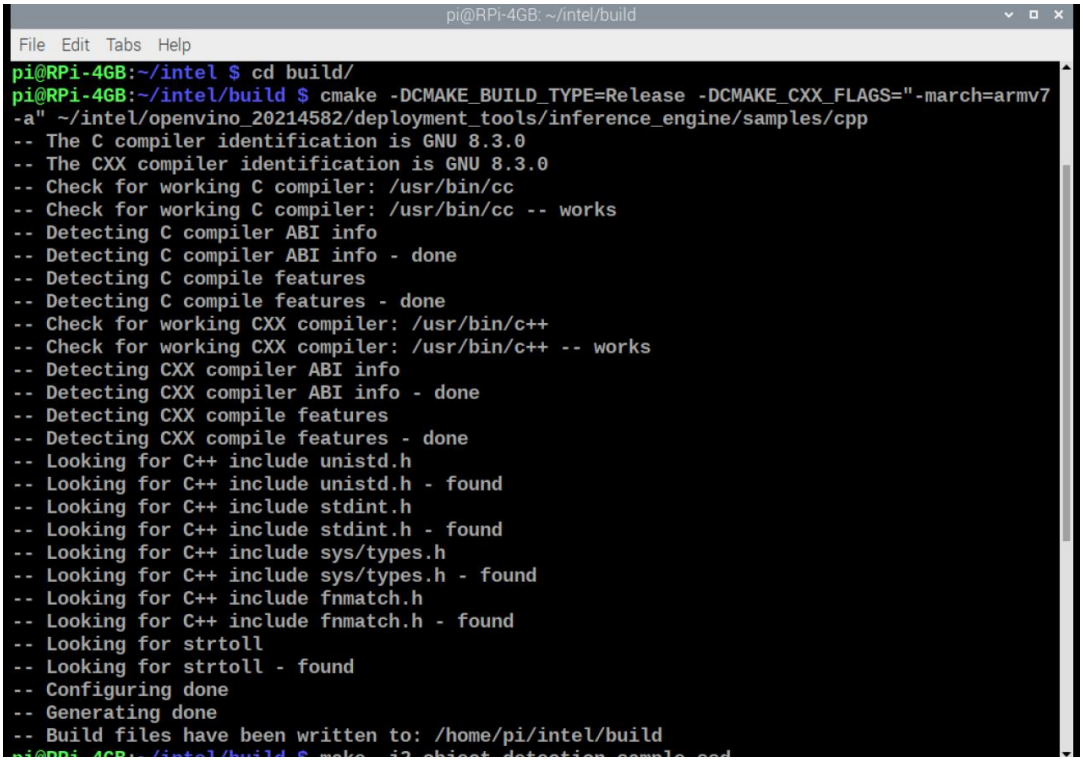


Figure 39 - Παραχώρηση δικαιωμάτων εγγραφής σε φάκελο

9.18.7.2 Cmake τα παραδείγματα

Μέσα στον φάκελο build εκτελείται η παρακάτω εντολή για την δημιουργία των κατάλληλων αρχείων του παραδείγματος

```
cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS="-march=armv7-a"
~/intel/openvino_20214582/deployment_tools/inference_engine/samples/cpp
```



9.18.7.3 Περίπτωση σφάλματος cannot find InferenceEngine.cmake

Σε περίπτωση που εμφανίσει σφάλμα

```
CMake Error at common/utils/CMakeLists.txt:15 (find_package): Find
InferenceEngine.cmake
```

Αναφέροντας ότι δεν μπορεί να βρεθεί στο CMAKE PATH το InferenceEngine

```
$ cmake -DCMAKE_BUILD_TYPE=Release -DCMAKE_CXX_FLAGS="-march=armv7-a" /opt/intel
/openvino/deployment_tools/inference_engine/samples

CMake Error at CMakeLists.txt:203 (find_package):
By not providing "FindInferenceEngine.cmake" in CMAKE_MODULE_PATH this
project has asked CMake to find a package configuration file provided by
"InferenceEngine", but CMake did not find one.

Could not find a package configuration file provided by "InferenceEngine"
(requested version 2.0) with any of the following names:

InferenceEngineConfig.cmake
inferenceengine-config.cmake

Add the installation prefix of "InferenceEngine" to CMAKE_PREFIX_PATH or
set "InferenceEngine_DIR" to a directory containing one of the above files.
If "InferenceEngine" provides a separate development package or SDK, be
sure it has been installed.


-- Configuring incomplete, errors occurred!
```

1) Πρέπει να τρέξει το setupvars.sh για να αρχικοποιηθεί το OpenVINO

Ή

2) Πρέπει να οριστούν οι σωστές διαδρομές για τους φακέλους ngraph και InferenceEngine με τις εντολές

```
export ngraph_DIR=/home/pi/openvino/build/ngraph
export InferenceEngine_DIR=/home/pi/openvino/build
```

- Package configuration error [50]

9.18.7.4 Make / build το παράδειγμα ανίχνευσης αντικειμένου

Στην συνέχεια, ακολουθεί η δημιουργία του εκτελέσιμου αρχείου του παραδείγματος, με την εντολή

```
make -j2 object_detection_sample_ssd
```

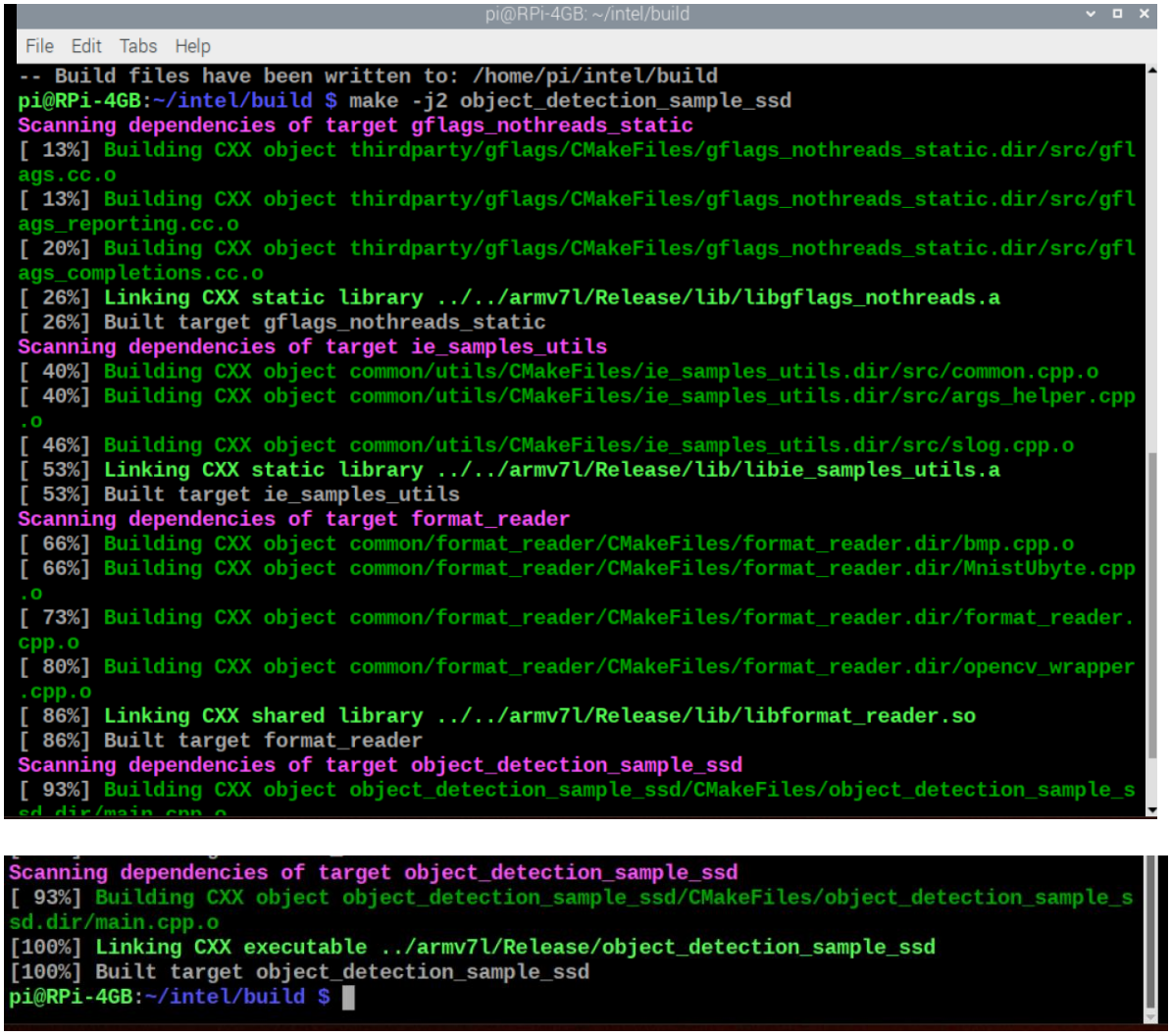
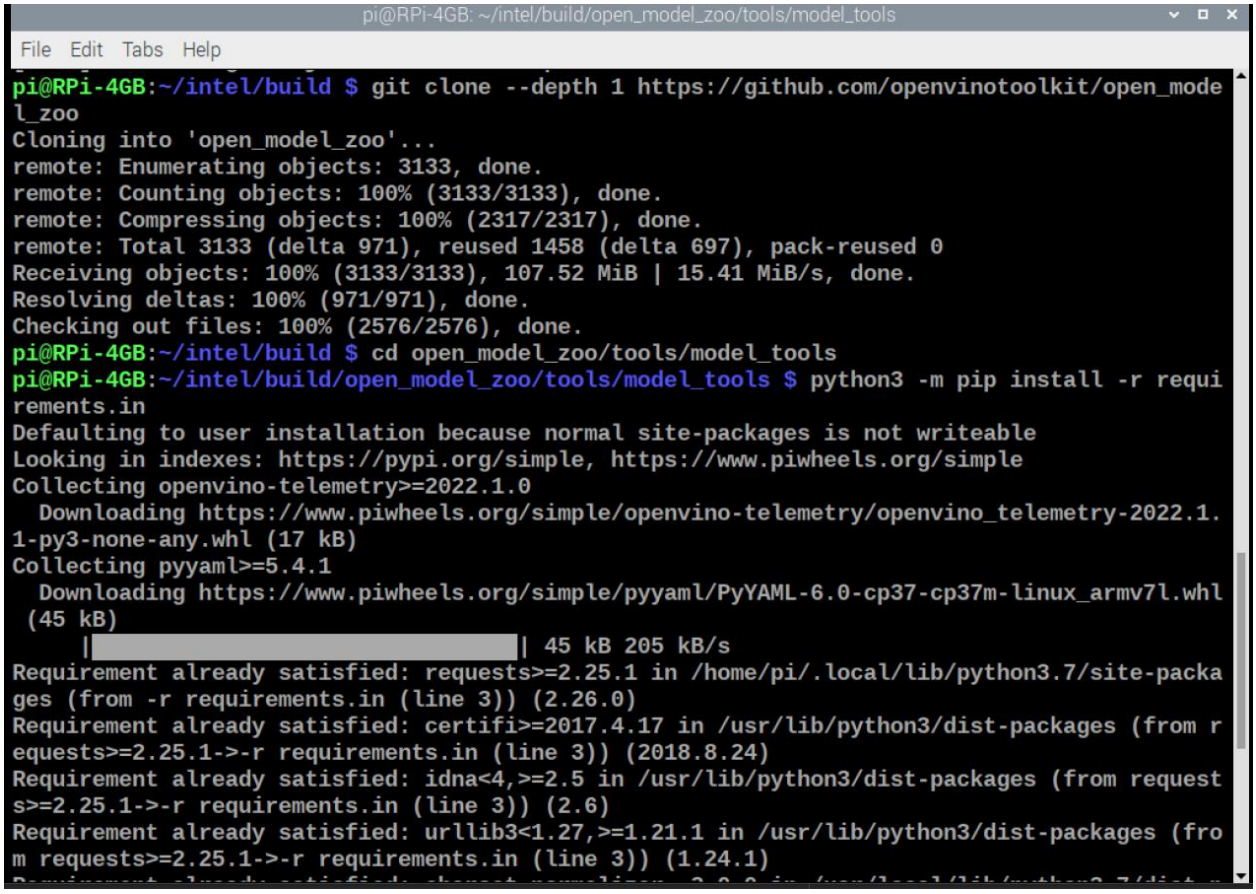


Figure 40 - Δημιουργία εκτελέσιμου αρχείου

9.18.7.5 Προετοιμασία και λήψη νευρωνικού δικτύου

Σε αυτό το σημείο έχουν ετοιμαστεί τα εκτελέσιμα αρχεία και χρειάζεται να ετοιμαστούν τα αρχεία του νευρωνικού δικτύου. Αρχικά θα πρέπει να αντιγραφεί από το GitHub ο Open Model Zoo. Στην συνέχεια πρέπει να γίνουν εγκατάσταση τα προαπαιτούμενα. Τέλος, με την χρήση του downloader, θα αποθηκευτούν τα 2 αρχεία του νευρωνικού δικτύου στον τοπικό φάκελο.

```
git clone --depth 1 https://github.com/openvinotoolkit/open_model_zoo
cd open_model_zoo/tools/model_tools
python3 -m pip install -r requirements.in
python3 downloader.py --name face-detection-adas-0001
```



Αν εμφανίσει μήνυμα ασυμβατότητας πακέτων, πρέπει να γίνουν οι κατάλληλες αναβαθμίσεις και αλλαγές στις εκδόσεις. Υπάρχει περίπτωση να μην εξαλειφθούν τελείως λόγω συγκρούσεων με άλλα πακέτα.

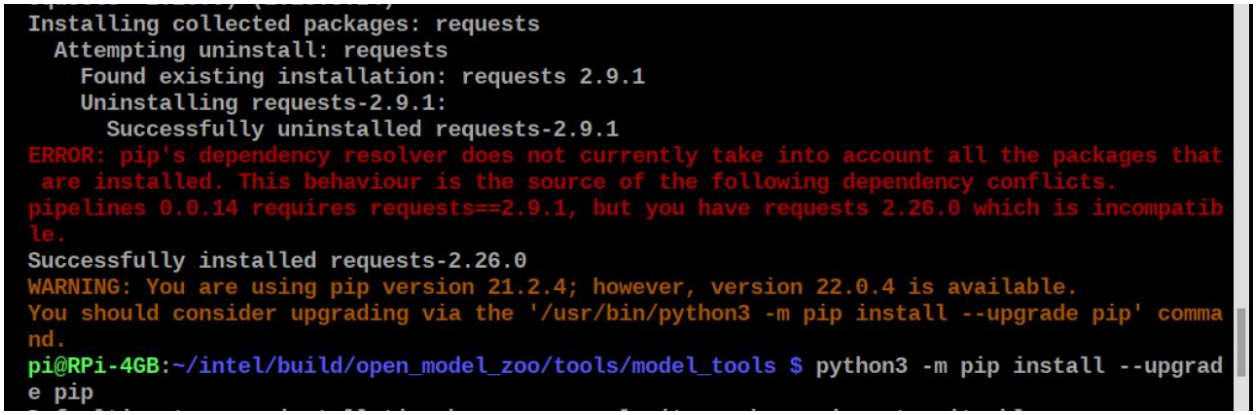


Figure 41 - Μήνυμα ενημέρωσης ασυμβατότητας πακέτων Python

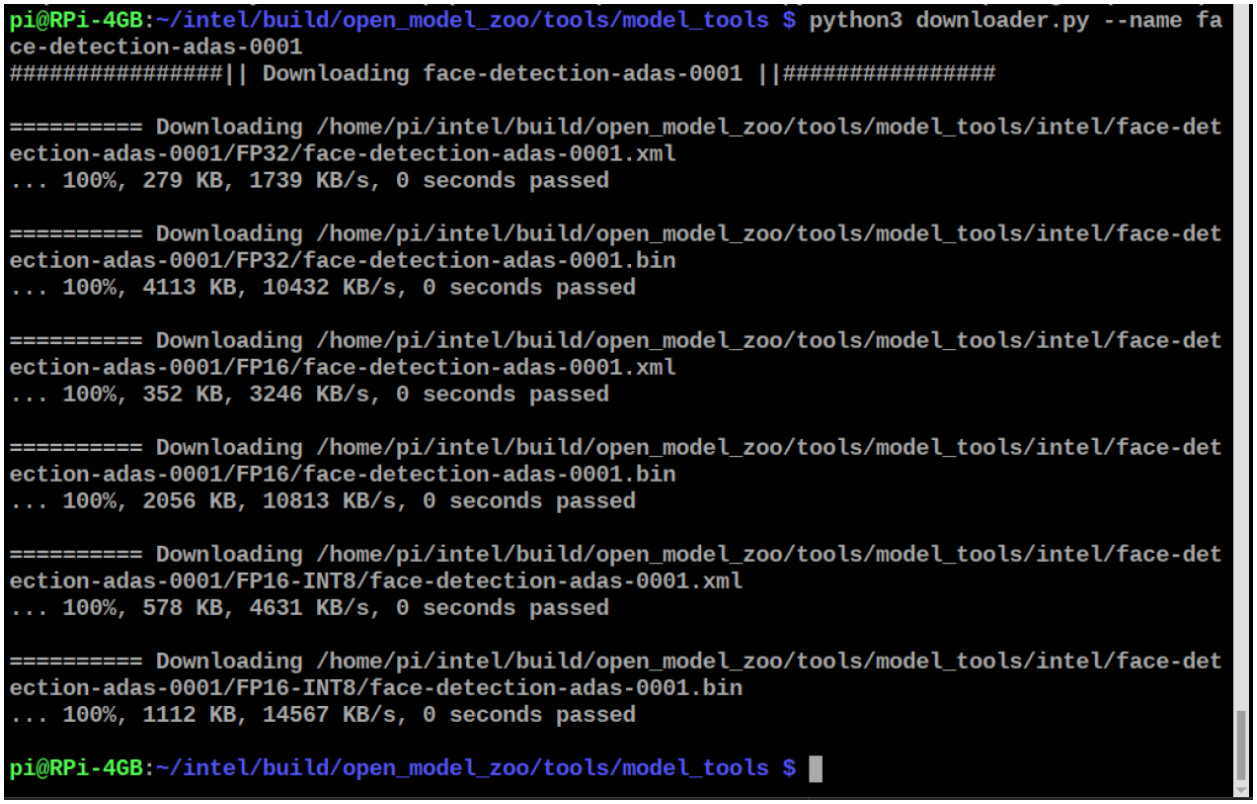


Figure 42 - Αποτέλεσμα λήψης όλων των ακριβειών νευρωνικού δικτύου

9.18.7.6 Εκτέλεση παραδείγματος

Το πρόγραμμα εμφανίζει σφάλμα κατά την εκτέλεσή του, χωρίς διευκρίνιση. Το πρόβλημα έχει διορθωθεί σε νεότερη έκδοση πακέτων του Raspbian.

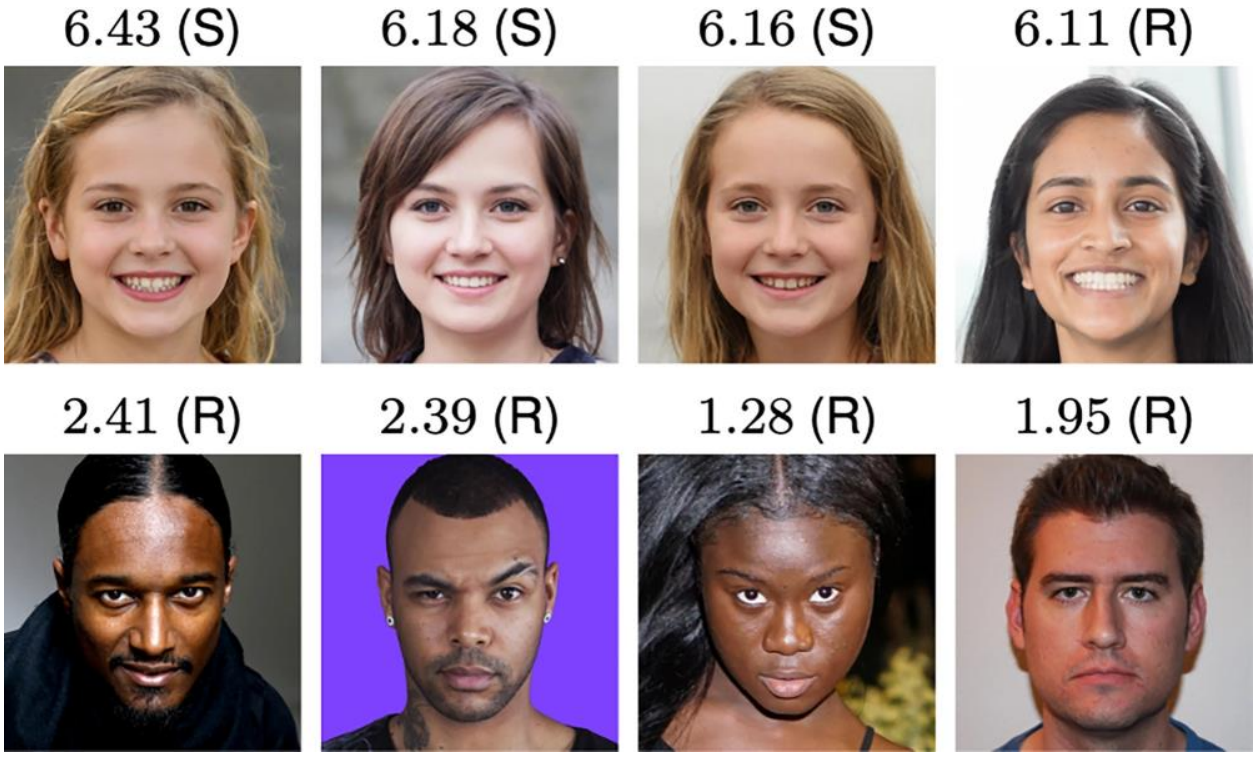
Η λύση είναι η πλήρης ενημέρωση των πακέτων με την εντολή

sudo apt update -y && sudo apt upgrade -y

- RPi update and upgrade [51]

Αφού ολοκληρωθεί επιτυχώς η ενημέρωση, επανεκτελείται το πρόγραμμα ανίχνευσης αντικειμένου. Στην προκειμένη περίπτωση θα χρησιμοποιηθεί νευρωνικό δίκτυο που θα βρίσκει πρόσωπα.

Η εικόνα που επιλέχθηκε ως είσοδος είναι η παρακάτω



- AI-synthesized faces [52]

Η εκτέλεση γίνεται με την εντολή

```
./armv7l/Release/object_detection_sample_ssd
    -m /home/pi/Downloads/face-detection-adas-0001.xml
    -d MYRIAD
    -i /home/pi/Downloads/faces2.jpg
```

Στο terminal που εκτελέστηκε η εντολή, εμφανίζονται οι σχετικές πληροφορίες, όπως η έκδοση της Inference Engine, του OpenVINO, το νευρωνικό δίκτυο που χρησιμοποιείται, τις αρχικές διαστάσεις και τις προσαρμοσμένες στην είσοδο του νευρωνικού δικτύου διαστάσεις της εικόνας.

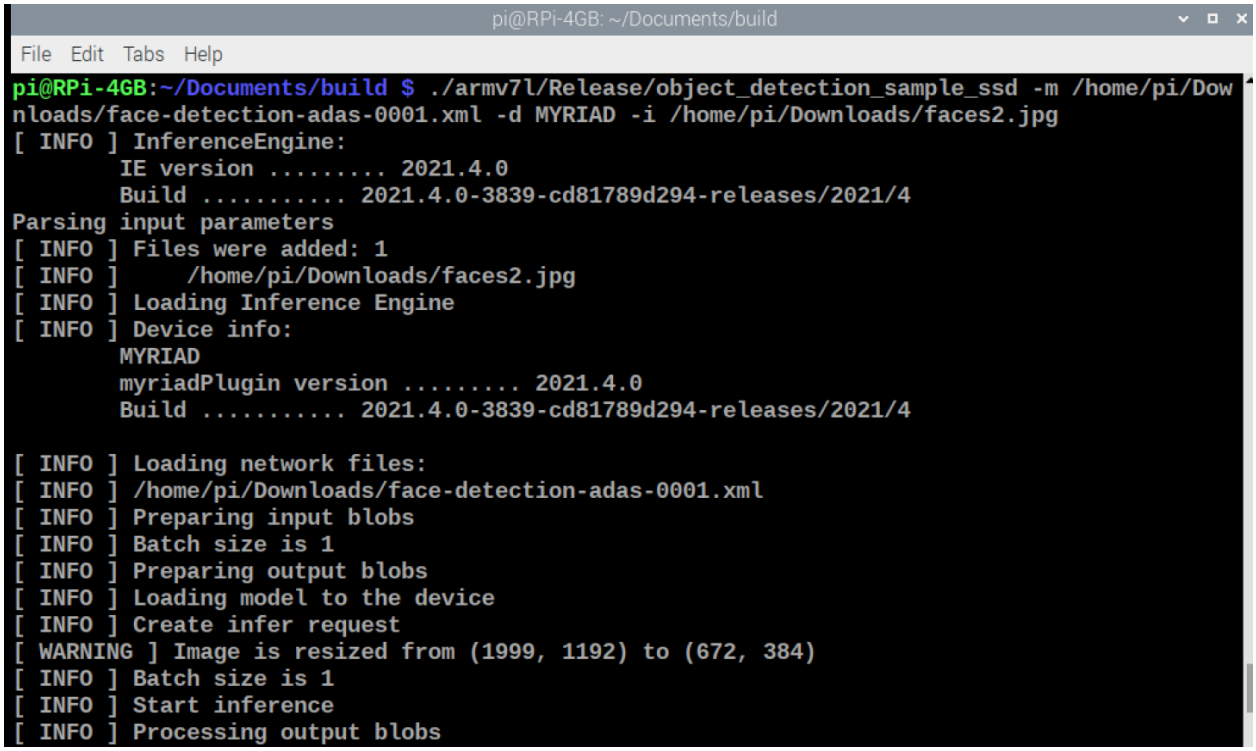


Figure 43 - Πληροφορίες και βήματα εκτέλεσης ανίχνευσης

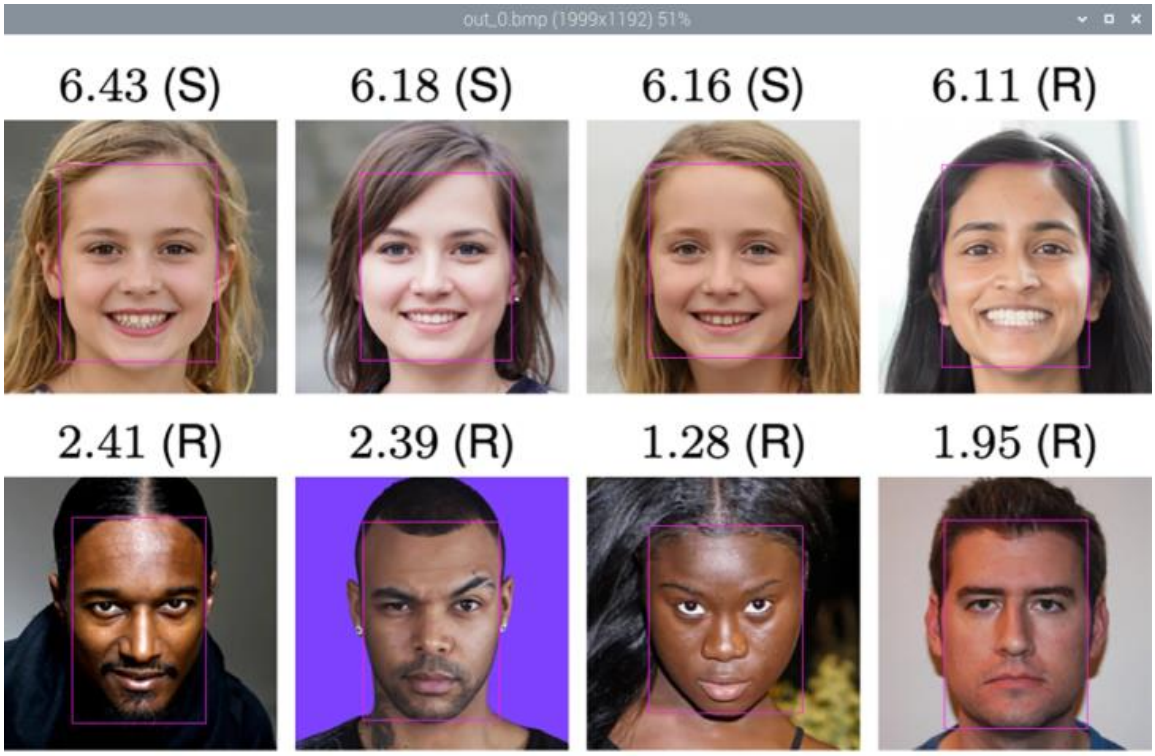
Αφού ολοκληρωθεί το Inferencing, στο ίδιο παράθυρο εμφανίζει τα στοιχεία που ανιχνεύθηκαν, ένα ανά γραμμή, με φθίνουσα σειρά, από την μεγαλύτερη βεβαιότητα προς την μικρότερη βεβαιότητα. Εάν οι βεβαιότητες είναι ίσες, τότε εμφανίζονται με την σειρά που ανιχνεύθηκαν μέσα στην εικόνα με προτεραιότητα από πάνω προς τα κάτω και από αριστερά προς τα δεξιά.

[Α/Α στοιχείου, Α/Α εικόνας] στοιχείο, βεβαιότητα ανίχνευσης = Εύρος (0 – 1) (Συντεταγμένες αριστερής πάνω γωνίας παραλληλόγραμμου ανίχνευσης) – (Συντεταγμένες δεξιάς κάτω γωνίας παραλληλόγραμμου ανίχνευσης) Α/Α παρτίδας εικόνων που εισήχθησαν προς Inferencing : XX – μήνυμα ενημέρωσης ότι το συγκεκριμένο στοιχείο θα εκτυπωθεί πάνω στην εικόνα της εξόδου

Η συγκεκριμένη εικόνα περιέχει 8 πρόσωπα, όπως επιβεβαιώνεται και από την εκτέλεση, αλλά και από την εικόνα με τα εκτυπωμένα πλαίσια.

```
[0,1] element, prob = 1    (97,171)-(371,515) batch id : 0 WILL BE PRINTED!
[1,1] element, prob = 1    (620,186)-(884,513) batch id : 0 WILL BE PRINTED!
[2,1] element, prob = 1    (1123,170)-(1388,508) batch id : 0 WILL BE PRINTED!
[3,1] element, prob = 1    (1632,172)-(1890,524) batch id : 0 WILL BE PRINTED!
[4,1] element, prob = 1    (118,785)-(352,1144) batch id : 0 WILL BE PRINTED!
[5,1] element, prob = 1    (625,793)-(863,1139) batch id : 0 WILL BE PRINTED!
[6,1] element, prob = 1    (1122,800)-(1391,1127) batch id : 0 WILL BE PRINTED!
[7,1] element, prob = 1    (1637,790)-(1887,1154) batch id : 0 WILL BE PRINTED!
[8,1] element, prob = 0.39209    (1371,1061)-(1476,1184) batch id : 0
[9,1] element, prob = 0.0366211    (1313,982)-(1464,1225) batch id : 0
[10,1] element, prob = 0.0341797    (1320,1092)-(1414,1212) batch id : 0
[11,1] element, prob = 0.0332031    (1507,997)-(1657,1240) batch id : 0
```

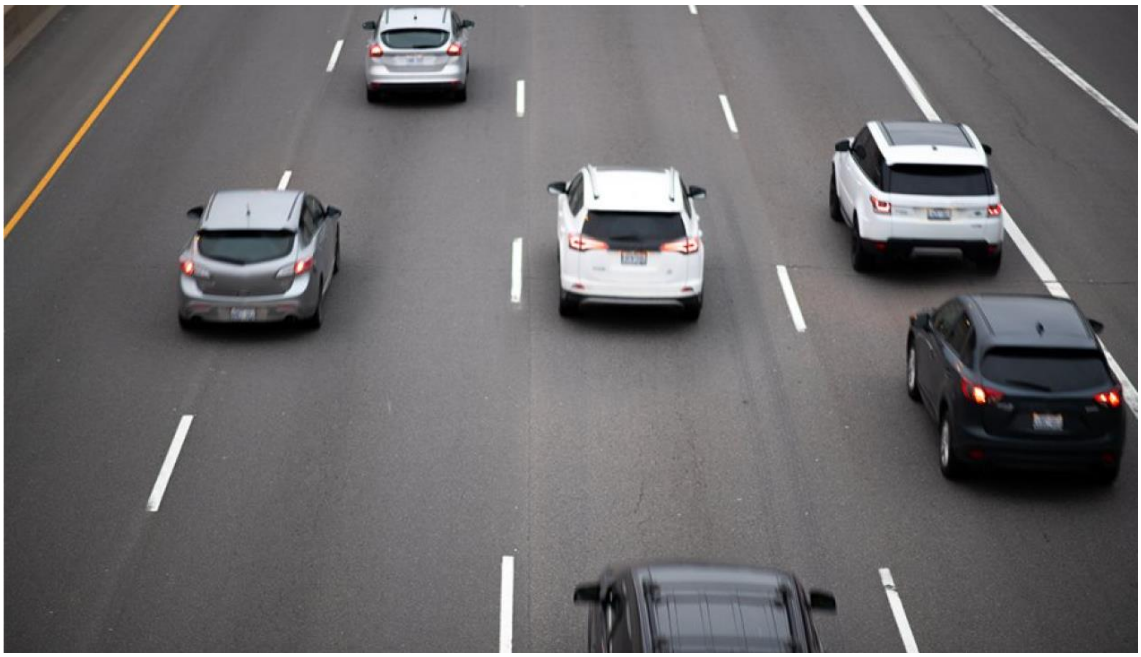
Figure 44 - Βεβαιότητες και συντεταγμένες ανιχνευθέντων προσώπων



9.18.7.7 Εκτέλεση σε εικόνα χωρίς πρόσωπο

Ως δοκιμή, εκτελείται ξανά το παραπάνω παράδειγμα, αλλά με είσοδο εικόνα που δεν περιέχει κανένα πρόσωπο.

Ως αποτέλεσμα πρέπει να υπάρξει μεν ανίχνευση στοιχείων, αλλά με πολύ μικρή βεβαιότητα και χωρίς να έχει εκτυπωθεί κανένα πλαίσιο ανίχνευσης στην εικόνα που δημιουργείται κατά την έξοδο.



Μέγιστη βεβαιότητα ανίχνευσης προσώπου 6,15%.

Άρα κανένα πλαίσιο δεν εκτυπώνεται στην αρχική εικόνα, οπότε η εικόνα εξόδου είναι ίδια με την αρχική.

```
[ INFO ] Create infer request
[ WARNING ] Image is resized from (900, 518) to (672, 384)
[ INFO ] Batch size is 1
[ INFO ] Start inference
[ INFO ] Processing output blobs
[0,1] element, prob = 0.0615234    (275,4)-(857,503) batch id : 0
[1,1] element, prob = 0.0561523    (10,6)-(529,487) batch id : 0
[2,1] element, prob = 0.0385742    (185,13)-(456,375) batch id : 0
[3,1] element, prob = 0.0356445    (418,12)-(471,95) batch id : 0
[4,1] element, prob = 0.0332031    (310,-12)-(688,308) batch id : 0
[5,1] element, prob = 0.0322266    (374,23)-(573,243) batch id : 0
[6,1] element, prob = 0.0292969    (468,451)-(496,496) batch id : 0
[7,1] element, prob = 0.0292969    (497,-30)-(567,92) batch id : 0
[8,1] element, prob = 0.0292969    (-50,135)-(324,449) batch id : 0
```

Figure 45 - Ορθή αδυναμία εύρεσης προσώπων σε αυτοκίνητα

9.18.7.8 Εκτέλεση σε πρόσωπο υπό γωνία

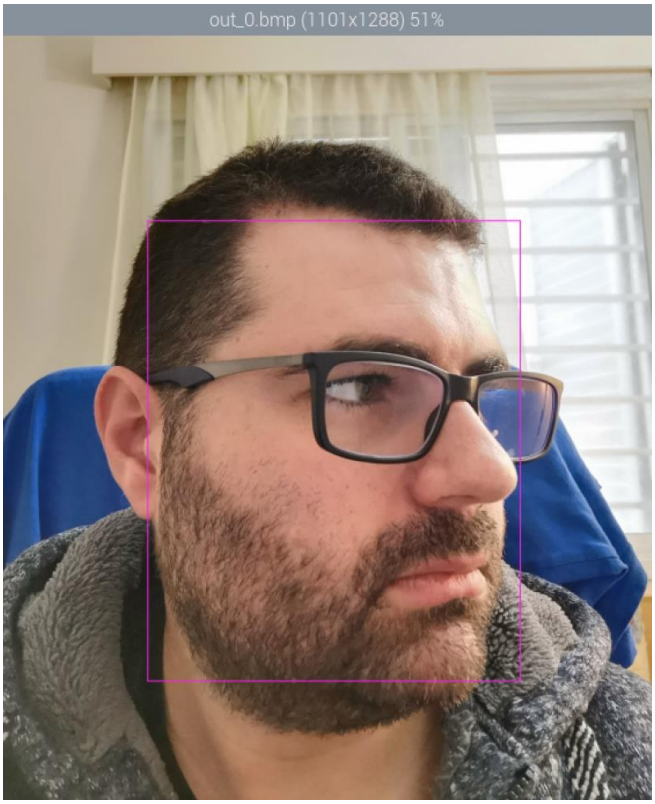
Για δοκιμή, εκτελείται ξανά το παραπάνω παράδειγμα άλλες 2 φορές με φωτογραφία προσώπου υπό γωνία.

Ακολουθούν οι βεβαιότητες της κάθε εκτέλεσης και οι φωτογραφίες εξόδου.

Εκτέλεση 1η

```
[ INFO ] Files were added: 1
[ INFO ]     /home/pi/Downloads/myface2.jpg
[ INFO ] Loading Inference Engine
[ INFO ] Device info:
         MYRIAD
         myriadPlugin version ......... 2021.4.0
         Build ........... 2021.4.0-3839-cd81789d294-releases/2021/4

[ INFO ] Loading network files:
[ INFO ] /home/pi/Downloads/face-detection-adas-0001.xml
[ INFO ] Preparing input blobs
[ INFO ] Batch size is 1
[ INFO ] Preparing output blobs
[ INFO ] Loading model to the device
[ INFO ] Create infer request
[ WARNING ] Image is resized from (1101, 1288) to (672, 384)
[ INFO ] Batch size is 1
[ INFO ] Start inference
[ INFO ] Processing output blobs
[0,1] element, prob = 0.999512    (248,308)-(871,1077) batch id : 0 WILL BE PRINTED!
[1,1] element, prob = 0.0249023    (974,1162)-(1036,1293) batch id : 0
[2,1] element, prob = 0.0170898    (719,738)-(840,821) batch id : 0
[3,1] element, prob = 0.0170898    (957,874)-(1247,1428) batch id : 0
```

Figure 46 - Ανίχνευση προσώπου υπό γωνία 1



Εκτέλεση 2η

```
[ INFO ] Files were added: 1
[ INFO ]     /home/pi/Downloads/myface3.jpg
[ INFO ] Loading Inference Engine
[ INFO ] Device info:
         MYRIAD
         myriadPlugin version ......... 2021.4.0
         Build ........... 2021.4.0-3839-cd81789d294-releases/2021/4

[ INFO ] Loading network files:
[ INFO ] /home/pi/Downloads/face-detection-adas-0001.xml
[ INFO ] Preparing input blobs
[ INFO ] Batch size is 1
[ INFO ] Preparing output blobs
[ INFO ] Loading model to the device
[ INFO ] Create infer request
[ WARNING ] Image is resized from (1224, 1260) to (672, 384)
[ INFO ] Batch size is 1
[ INFO ] Start inference
[ INFO ] Processing output blobs
[0,1] element, prob = 0.756348    (164,360)-(921,1162) batch id : 0 WILL BE PRINTED!
[1,1] element, prob = 0.486328    (274,364)-(881,787) batch id : 0
[2,1] element, prob = 0.0258789    (59,49)-(273,636) batch id : 0
[3,1] element, prob = 0.0239258    (-12,-155)-(205,681) batch id : 0
```

Figure 47 - Ανίχνευση προσώπου υπό γωνία 2

Με τα παραπάνω, επιβεβαιώνεται η καλή λειτουργία της διαδικασίας του Inferencing στο Raspberry Pi. Στο επόμενο στάδιο θα πραγματοποιηθούν τα βήματα για την επίτευξη του αρχικού στόχου.

9.18.8 Εκτέλεση παραδείγματος ανίχνευσης σε Python

Καθώς όλα τα προγράμματα θα υλοποιηθούν σε Python, θα πραγματοποιηθεί και δοκιμή καλής λειτουργίας.

Η εκτέλεση θα γίνει με την εντολή

```
python3
/home/pi/Downloads/object_detection_demo/python/python/object_detection_dem
o.py
   -m /home/pi/Downloads/person-vehicle-bike-detection-crossroad-
0078/FP16/person-vehicle-bike-detection-crossroad-0078.xml
   -at SSD\
   -i /home/pi/Downloads/walk.jpg
   -d MYRIAD
```

9.18.8.1 Σφάλμα models

Εμφανίζεται το σφάλμα

```
Traceback (most recent call last):
  File "/home/pi/Downloads/object_detection_demo/python/object_detection_demo.py
", line 33, in <module>
    import models
ModuleNotFoundError: No module named 'models'
```

Με αναζήτηση των εγκατεστημένων πακέτων Python διαπιστώνεται ότι το models έχει μετονομαστεί σε doqu. Αφού γίνει η εγκατάσταση του doqu (*python3 pip install doqu*), πρέπει να αλλαχθεί η γραμμή 33 του object detection

Από “import models”

Σε “import doqu as models”

9.18.8.2 Σφάλμα document_base

Εκτελείται ξανά η εντολή για το object detection

Εμφανίζεται το σφάλμα

```
In file doqu _init_
ModuleNotFoundError : No module named 'document_base'
```

Ενώ το αρχείο document_base υπάρχει κανονικά.

Η λύση είναι να δηλωθεί ότι το αρχείο βρίσκεται στον ίδιο φάκελο, το οποίο επιτυγχάνεται με την προσθήκη 1 τελείας πριν το αρχείο. Προσθήκη της στην γραμμή 21 του _init_

Από “from document_base import Document, Many”

Σε “from .document_base import Document, Many”

9.18.8.3 Σφάλμα validators

Στην συνέχεια, αφού εκτελεστεί η εντολή για το object detection, εμφανίζεται σφάλμα ότι στο document_base δεν βρίσκει το validators.

Η λύση είναι ίδια με την προηγούμενη, δηλαδή να δηλωθεί ότι το αρχείο βρίσκεται στον ίδιο φάκελο. Στην γραμμή 14 δηλώνεται η αλλαγή

Από “import validators”

Σε “from .validators import validators”

9.18.8.4 Σφάλμα validators.py (ur error)

Στο validators.py, στην γραμμή 330 και στην γραμμή 331 εμφανίζει σφάλμα άγνωστου στοιχείου “ur”

Καθώς τα παραπάνω modules έχουν γραφτεί για python 2, το “ur” δεν είναι συμβατό με την python 3 που χρησιμοποιείται. Η λύση είναι να αλλάξει

Από “ur”

Σε “r”

9.18.8.5 Σφάλμα συντακτικού document_base

Στο document_base line 53

Print self, save() – Invalid syntax

Όπως και προηγουμένως, το σφάλμα εμφανίζεται λόγω της αλλαγής εκδόσεων Python. Διορθώνεται η συγκεκριμένη γραμμή

Από “print self, save()”

Σε “print (self), save()”

Η συγκεκριμένη αλλαγή γίνεται σε όλα τα “print” που έχει το document_base

9.18.8.6 Σφάλμα monitors – Εκτέλεση object detection σε Windows

Με την επανεκτέλεση, εμφανίζεται σφάλμα μη ύπαρξης του module monitors. Μετά από την αποτυχημένη προσπάθεια εγκατάστασης του module monitors μέσω της pip, καθώς δεν υφίσταται τέτοιο module, εκτελείται το παράδειγμα object detection μέσω της πλήρους εγκατάστασης του OpenVINO στα Windows.

Η διαδικασία ολοκληρώνεται επιτυχώς, οπότε με αναζήτηση του κώδικα διαπιστώνεται ότι όλα τα παραπάνω σφάλματα εμφανίστηκαν λόγω της απουσίας των αρχείων που βρίσκονται στον φάκελο common του OpenVINO και παρέχονται από την Intel.

Το επόμενο βήμα είναι η αντιγραφή των αρχείων στην κατάλληλη θέση στο Raspberry Pi. Έτσι, παύουν να εμφανίζονται τα σφάλματα και εκτελείται κανονικά το object_detection_demo.py στο Raspberry Pi.

9.18.8.7 Αποτελέσματα εκτέλεσης object detection

Όπως φαίνεται και από τις παρακάτω εικόνες, το Inferencing πραγματοποιήθηκε επιτυχώς και ανιχνεύθηκαν 2 αντικείμενα με class ID 1 (person) και βεβαιότητα 99,8% και 99,9% αντίστοιχα.

```
[ INFO ] Initializing Inference Engine...
[ INFO ] Loading network...
[ INFO ] Reading network from IR...
[ INFO ] Use SingleOutputParser
[ INFO ] Loading network to MYRIAD plugin...
[ INFO ] Starting inference...
To close the application, press 'CTRL+C' here or switch to the output window and
 press ESC key
Latency: 694.1 ms
FPS: 7.0
```

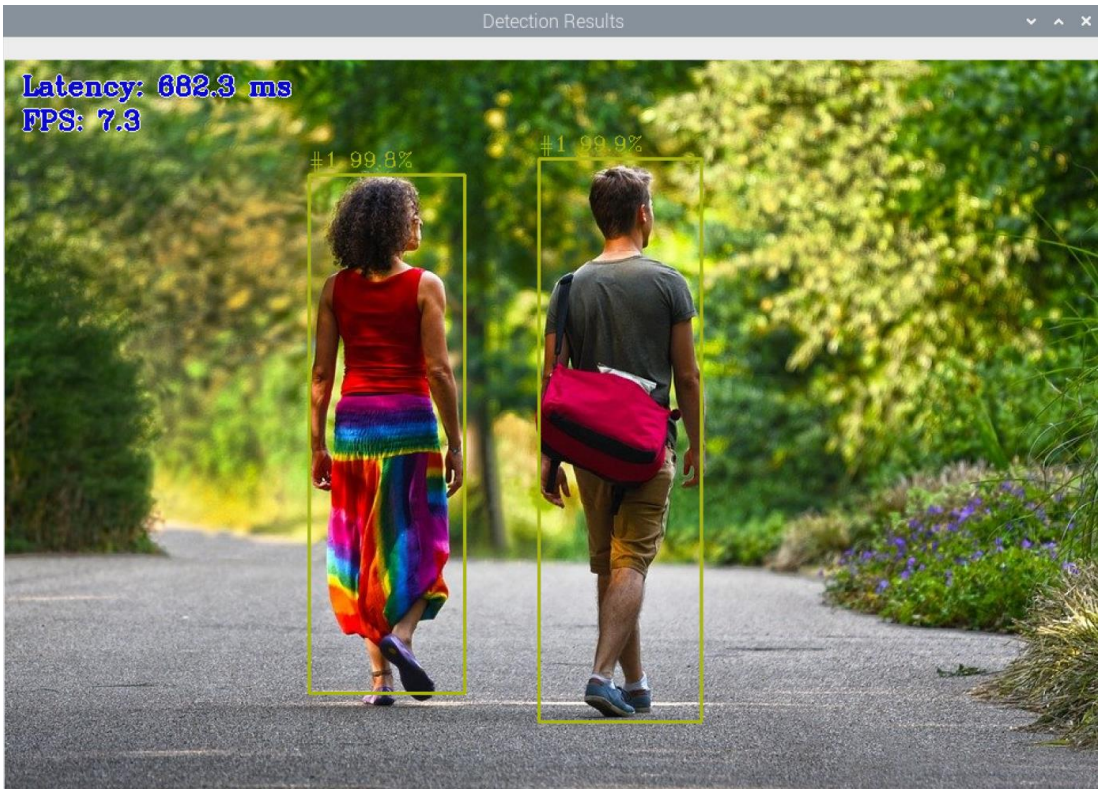


Figure 48 - Ανίχνευση ανθρώπων (ID 1)

10. Εκτέλεση πραγματικών συνθηκών

Αρχικά πρέπει να συλλεχθούν βίντεο από δρόμους με κίνηση οχημάτων για να εκτελεστεί το Inferencing. Αυτά μπορούν να βρεθούν στο διαδίκτυο. Όμως, η βέλτιστη λύση είναι να συλλεχθούν από σημεία στα οποία επικεντρώνεται η συγκεκριμένη εργασία. Έτσι, θα ληφθούν από διάφορες γωνίες, ώστε να υπάρξει σύγκριση για την εύρεση της βέλτιστης δυνατής λύσης.

Τα βίντεο θα πρέπει να είναι από την κάμερα του Raspberry Pi, καθώς αυτή είναι και η κάμερα που θα χρησιμοποιηθεί για την συγκεκριμένη εργασία. Το σημείο που θα γίνει η λήψη είναι διασταύρωση κεντρικών δρόμων. Έτσι, επειδή δεν είναι εφικτή η ζωντανή σύνδεση του RPi με υπολογιστή, αλλά ούτε και με οθόνη για την προβολή της εικόνας της κάμερας σε πραγματικό χρόνο και δεν υπάρχει σύνδεση του RPi με το διαδίκτυο, θα πρέπει να υλοποιηθεί η λύση της λήψης βίντεο κατά βούληση μέσω φυσικού κουμπιού.

10.1 Εγγραφή βίντεο κατά βούληση με την χρήση κουμπιού

10.1.1 Κύκλωμα button

Στους ακροδέκτες του Raspberry Pi θα συνδεθεί φυσικό button για την ενεργοποίηση της λήψης βίντεο. Καθώς υπάρχει η πιθανότητα της δημιουργίας εσφαλμένων εντολών λόγω της φυσικής αναπήδησης του κουμπιού, θα πρέπει να προστεθεί λειτουργία debouncing. Αναλυτικότερα, το button bounce προέρχεται από την φυσική λειτουργία του κουμπιού. Όταν ο χρήστης πατάει ή αφήνει το κουμπί, αυτό δεν αλλάζει κατάσταση ακαριαία, αλλά ενδέχεται για δέκατα του χιλιοστού του δευτερολέπτου να αναπηδήσει και να αλλάξει αρκετές φορές την κατάστασή του ανάμεσα στις 2 καταστάσεις του. Ως μη επιθυμητό αποτέλεσμα εμφανίζεται η καταχώρηση εξτρά αλλαγών της κατάστασης. Στην παρακάτω εικόνα φαίνεται η κυμματομορφή ενός κουμπιού κατά το πάτημα.

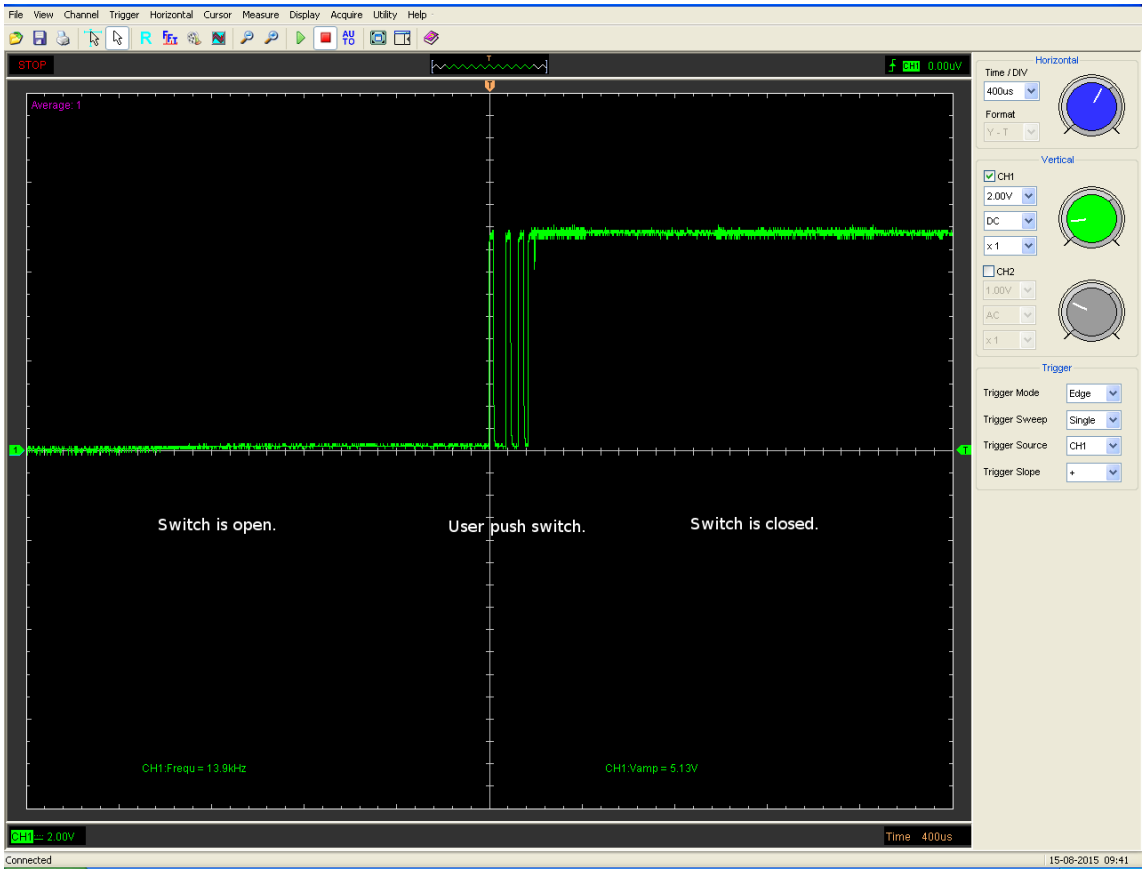


Figure 49 - Πάτημα μπουτόν και φαινόμενο αναπήδησης, αναπαράσταση σε παλμογράφο

Όπως προαναφέρθηκε, πρέπει να προστεθεί λειτουργία debouncing για να αποφευχθεί η δημιουργία εσφαλμένων εντολών. Αυτό επιτυγχάνεται είτε με χρήση λογισμικού είτε με χρήση εξτρά υλικού. Για την λύση με χρήση λογισμικού έχουν αναφερθεί περιπτώσεις δυσλειτουργίας στο Raspberry Pi, οπότε θα επιλεγεί η λύση με την χρήση εξτρά υλικού.

- Allaboutcircuits – Switch bounce [53]
- Texas Instruments – Debounce a switch [54]

Ο πιο απλός τρόπος είναι με την χρήση πυκνωτή, αλλά χωρίς να λύνει εντελώς το πρόβλημα, καθώς στην μετάβαση από κατάσταση ON σε κατάσταση OFF ενδέχεται να εμφανιστεί η περίπτωση του bounce. Η ενδεδειγμένη λύση είναι με την χρήση του ολοκληρωμένου CD40106B CMOS Hex Schmitt -Trigger Inverter, το οποίο αλλάζει κατάσταση μόνο όταν η είσοδος ξεπεράσει κάποια προκαθορισμένα όρια και δεν μπαίνει σε λειτουργία άγνωστης κατάστασης. Αναλυτικότερα, όταν η είσοδος είναι κάτω από 0,8V, το Schmitt έχει έξοδο λογικό 1. Μέχρι η είσοδος να ανέβει πάνω από το όριο των 2V, το Schmitt παραμένει λογικό 1 και αλλάζει σε λογικό 0 μετά τα 2V.

Ευτυχώς, το Raspberry Pi έχει ενεργοποιημένη υστέρηση για να αποφεύγει το φαινόμενο. Επίσης, στο συγκεκριμένο πρόγραμμα αγνοούνται τα πολλαπλά πατήματα, οπότε δεν επηρεάζουν την λειτουργία.

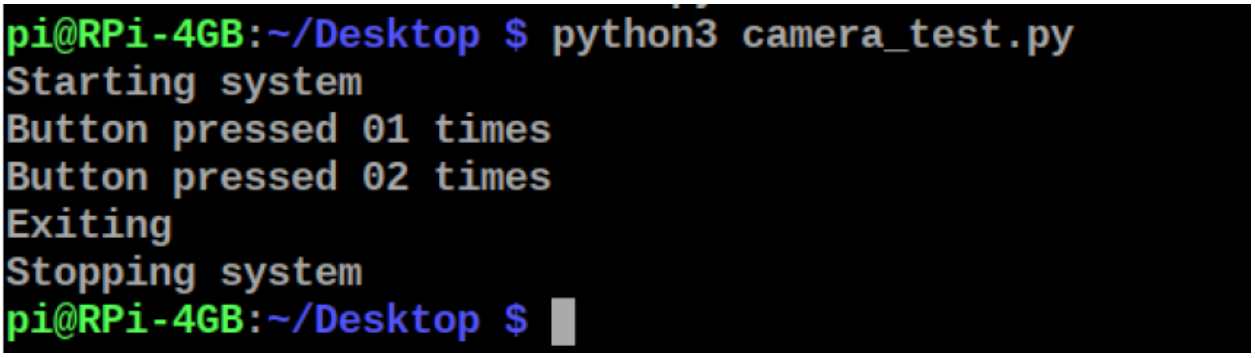
10.1.2 Πρόγραμμα Python εγγραφής video

Το πρόγραμμα δομήθηκε με σκοπό την απλή και κατανοητή μορφή του και την λειτουργικότητά του.

Τα βίντεο ονοματίζονται με την μορφή **video01_DD-MM-YYYY_HH-MM-ss.h264**, για παράδειγμα video01_13-03-2021_13-26-11.h264

```
from picamera import PiCamera
from time import sleep
from gpiozero import LED, Button
from cv2 import waitKey
import datetime

cam = PiCamera()
button = Button (17) #Button εγγραφής
led = LED (27) #LED ένδειξης εγγραφής
button2 = Button (22) #Button τερματισμού προγράμματος

cam.resolution = (640, 480) #Ανάλυση καταγραφής κάμερας
cam.framerate = 30 #Ρυθμός εικόνων ανά δευτερόλεπτο
vid_count = 1 #Αύξων αριθμός για την αρίθμηση των βίντεο που αποθηκεύονται


cam.start_preview() #Έναρξη παραθύρου εικόνας από κάμερα - Για λόγους
Troubleshooting
sleep(1)
print ('Starting system') #Μήνυμα έναρξης προγράμματος - Troubleshooting
while True:
   if button.is_pressed: #Όταν πατηθεί το button 1, προχωράει στα βήματα
της εγγραφής
      print ('Button pressed %02d times' %vid_count) #Μήνυμα ένδειξης
αριθμού εγγραφών
      now = datetime.datetime.now() #Αποθήκευση ημερομηνίας και ώρας αυτή
την στιγμή
      cam.start_recording('/home/pi/test_videos/video%02d_%s.h264' #Έναρξη
εγγραφής
         % (vid_count, now.strftime("%d-%m-%Y_%H-%M-%S"))) #Προσθήκη του
Α/Α του βίντεο και της ημερομηνίας και ώρας στην ονομασία του βίντεο
      vid_count += 1 #Αύξηση αριθμού για το επόμενο βίντεο
      led.on() #Άναμμα LED για την οπτική ένδειξη ότι εγγράφεται βίντεο
      sleep(30) #Αναμονή 30" για να ολοκληρωθεί η εγγραφή
      cam.stop_recording() #Τερματισμός εγγραφής
      led.off() #Σβήσιμο LED

   if button2.is_pressed: #Όταν πατηθεί το button 2
      print ('Exiting') #εμφανίζεται μήνυμα τερματισμού - Troubleshooting
      break #και διακόπτεται το πρόγραμμα


sleep(2)
print ('Stopping system') #Μήνυμα τερματισμού - Troubleshooting
cam.stop_preview #Τερματισμός ζωντανής εικόνας
```

Κατά την εκτέλεση, εμφανίζονται τα σχετικά μηνύματα στο terminal



Μετά την επιτυχή δοκιμή της εγγραφής, γίνεται η συλλογή των βίντεο.

Στα επόμενα στάδια θα γίνει χρήση τους για τις δοκιμές.

10.2 Επιλογή νευρωνικών δικτύων για την ανίχνευση οχημάτων

Για την υλοποίηση της ανίχνευσης, θα χρειαστεί να γίνει διαλογή των συμβατών νευρωνικών δικτύων που θα χρησιμοποιηθούν, έτσι ώστε να γίνουν οι δοκιμές και να βρεθεί η βέλτιστη δυνατή λύση.

Μετά την επιλογή των δικτύων, θα πρέπει να ακολουθηθούν τα βήματα συλλογής των νευρωνικών δικτύων και μετατροπής σε μορφή IR. Αυτά θα γίνουν σε υπολογιστή που έχει Windows, γιατί, όπως έχει προαναφερθεί, η έκδοση του OpenVINO για Raspberry OS (Raspbian) δεν περιλαμβάνει τον Model Optimizer.

Στην συνέχεια, τα αρχεία θα μεταφερθούν στο Raspberry Pi και αλλάζοντας τις κατάλληλες ρυθμίσεις, θα γίνουν οι δοκιμές για κάθε νευρωνικό δίκτυο.

Τα νευρωνικά δίκτυα που θα χρησιμοποιηθούν, επιλέχθηκαν με βάση την δυνατότητα εκτέλεσής τους στο Neural Compute Stick 2, την ακρίβεια των αποτελεσμάτων που δίνονται από τον κατασκευαστή και την ευκολία στην εκτέλεση σε υλικά χαμηλών δυνατοτήτων. Στην βιβλιογραφία υπάρχουν οι σχετικοί υπερσύνδεσμοι για κάθε νευρωνικό δίκτυο.

10.2.1 Πίνακας νευρωνικών δικτύων ανίχνευσης αντικειμένων

Μοντέλο	Πληροφορίες	Framework
pedestrian-and-vehicle-detector-adas-0001	Average Precision for pedestrians 88% AP for vehicles 90% Target pedestrian size 60x120 pixels Target vehicle size 40x30 pixels GFLOPS 3.974 MParams 1.650	Caffe
person-vehicle-bike-detection-2004	AP @ [IoU=0.50:0.95] 0.274 (internal test set) GFlops 1.811 MParams 2.327	PyTorch
person-vehicle-bike-detection-crossroad-1016	Mean Average Precision (mAP) 62.55% AP people 73.63% AP vehicles 77.84% AP bikes 36.18% Max objects to detect 200 GFlops 3.560	PyTorch
person-vehicle-bike-detection-crossroad-yolov3-1020	Mean Average Precision (mAP) 48.89% AP people 58.94% AP vehicles 62.05% AP bikes/motorcycles 25.66% GFlops 65.98 MParams 61.92	Keras
vehicle-detection-0202	AP @ [IoU=0.50:0.95] 0.363 (internal test set) GFlops 3.143 MParams 1.817	PyTorch
vehicle-detection-adas-0002	Average Precision (AP) 90.6% Target vehicle size 40 x 30 pixels on Full HD image Max objects to detect 200 GFlops 2.798 MParams 1.079	Caffe
yolo-v2-tiny-vehicle-detection-0001	mAP 88.64% coco_precision 94.97%	Keras

	GFLOPs 5.424 MParams 11.229	
yolo-v4-tf	mAP 71.23% COCO mAP (0.5) 77.40% COCO mAP (0.5:0.05:0.95) 50.26% GFLOPs 129.5567 MParams 64.33	Keras

Figure 50 - Πίνακας πληροφοριών νευρωνικών δικτύων ανίχνευσης

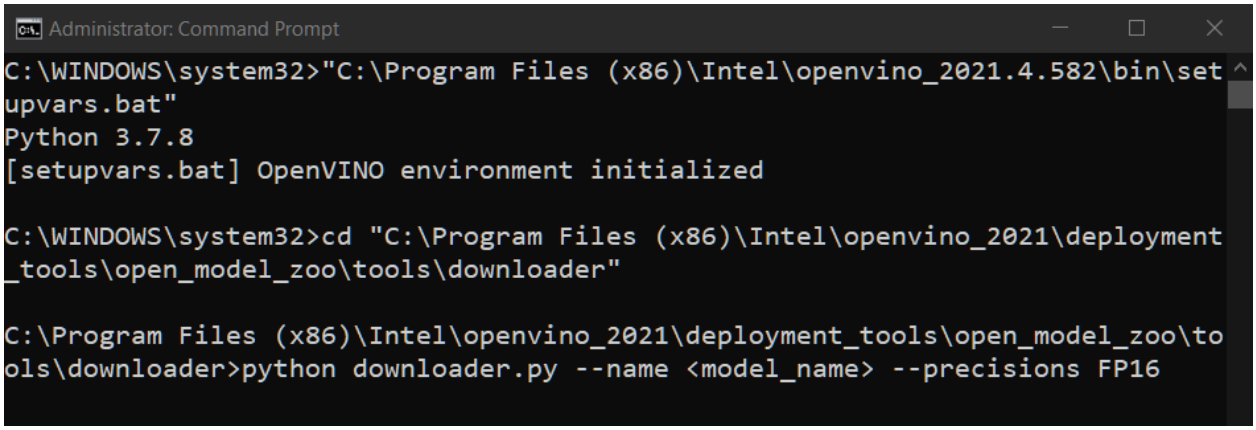
Mean Average Precision [0,5:0,05:0,95] σημαίνει πως η μέση ακρίβεια αποτελεί τον μέσο όρο των μέσων ακριβειών για όριο ανίχνευσης από 0,5 έως 0,95 με βήμα αύξησης 0,05 (0,5 – 0,55 – 0,6 … 0,90 – 0,95)

- pedestrian-and-vehicle-detector-adas-0001 [55]
- person-vehicle-bike-detection-2004 [56]
- person-vehicle-bike-detection-crossroad-1016 [57]
- person-vehicle-bike-detection-crossroad-yolov3-1020 [58]
- vehicle-detection-0202 [59]
- vehicle-detection-adas-0002 [60]
- yolo-v2-tiny-vehicle-detection-0001 [61]
- yolo-v4-tf [62]

10.2.2 Συλλογή και μετατροπή των νευρωνικών δικτύων

Η διαδικασία έχει αναλυθεί σε προηγούμενο στάδιο. Τα βήματα είναι ίδια για όλα τα νευρωνικά δίκτυα.

Αρχικά πρέπει να εκτελεστεί το downloader.py και να ληφθούν όλα τα παραπάνω νευρωνικά δίκτυα.



Για διευκόλυνση, μπορούν να ληφθούν όλα τα αρχεία με μία εντολή.

Αυτό γίνεται με την δημιουργία ενός νέου αρχείου txt μέσα στο οποίο αναγράφονται τα ονόματα των νευρωνικών δικτύων που θα χρησιμοποιηθούν. Στην συνέχεια αποθηκεύεται το αρχείο σε μορφή .LST

```
pedestrian-and-vehicle-detector-adas-0001
person-vehicle-bike-detection-2004
person-vehicle-bike-detection-crossroad-1016
person-vehicle-bike-detection-crossroad-yolov3-1020
vehicle-detection-0202
vehicle-detection-adas-0002
yolo-v2-tiny-vehicle-detection-0001
yolo-v4-tf
```

Με την παρακάτω εντολή λαμβάνονται όλα τα σχετικά δίκτυα. Επειδή το NCS2 υποστηρίζει μόνο νευρωνικά δίκτυα ακρίβειας FP16, γίνεται επιλογή της λήψης μόνο των αρχείων αυτής της ακρίβειας.

```
C:\Program Files (x86)\Intel\openvino_2021\deployment_tools\open_model_zoo\tools\
downloader>python downloader.py --list "C:\FP16\od.lst" --precisions FP16
```

Εμφανίζονται πληροφορίες για το κάθε αρχείο που κατεβαίνει, ξεχωριστά, όπως η ταχύτητα λήψης, το ποσοστό ολοκλήρωσης, τον χρόνο που παρήλθε.

```
################|| Downloading pedestrian-and-vehicle-detector-adas-0001 ||######
##########

========== Downloading C:\Program Files (x86)\Intel\openvino_2021\deployment_tool
s\open_model_zoo\tools\downloader\intel\pedestrian-and-vehicle-detector-adas-0001
\FP16\pedestrian-and-vehicle-detector-adas-0001.xml
... 100%, 239 KB, 1698 KB/s, 0 seconds passed

========== Downloading C:\Program Files (x86)\Intel\openvino_2021\deployment_tool
s\open_model_zoo\tools\downloader\intel\pedestrian-and-vehicle-detector-adas-0001
\FP16\pedestrian-and-vehicle-detector-adas-0001.bin
... 100%, 3222 KB, 10293 KB/s, 0 seconds passed

################|| Downloading person-vehicle-bike-detection-2004 ||############
###

========== Downloading C:\Program Files (x86)\Intel\openvino_2021\deployment_tool
s\open_model_zoo\tools\downloader\intel\person-vehicle-bike-detection-2004\FP16\p
erson-vehicle-bike-detection-2004.xml
... 100%, 1266 KB, 8117 KB/s, 0 seconds passed
```

Όλα τα νευρωνικά δίκτυα είναι σε έτοιμη μορφή Intermediate Representation (*.xml & *.bin), εκτός από το yolo-v4-tf. Το συγκεκριμένο δίκτυο έρχεται σε διαφορετική μορφή. Τα αρχεία του περιλαμβάνουν την τοπολογία, τα στρώματα, τους νευρώνες και τα βάρη τους και χρειάζεται να μετατραπεί σε μορφή IR, με την εντολή

```
python converter.py --name yolo-v4-tf --precisions FP16
```

```
C:\Program Files (x86)\Intel\openvino_2021\deployment_tools\open_model_zoo\
tools\downloader>python converter.py --name yolo-v4-tf --precisions FP16
========== Running pre-convert script for yolo-v4-tf
Pre-convert command: C:\Python37\python.exe -- "C:\Program Files (x86)\Inte
l\openvino_2021.4.582\deployment_tools\open_model_zoo\models\public\yolo-v4
-tf\pre-convert.py" -- "C:\Program Files (x86)\Intel\openvino_2021\deployme
nt_tools\open_model_zoo\tools\downloader\public\yolo-v4-tf" "C:\Program Fil
es (x86)\Intel\openvino_2021\deployment_tools\open_model_zoo\tools\download
er\public\yolo-v4-tf"

2022-03-15 06:19:39.167321: W tensorflow/stream_executor/platform/default/d
so_loader.cc:60] Could not load dynamic library 'cudart64_110.dll'; dlerror
: cudart64_110.dll not found
2022-03-15 06:19:39.167615: I tensorflow/stream_executor/cuda/cudart_stub.c
c:29] Ignore above cudart dlerror if you do not have a GPU set up on your m
achine.
Loading weights.
Weights Header:  0 2 5 [32032000]
Parsing Darknet config.
Creating Keras model.
Parsing section net_0
Parsing section convolutional_0
conv2d bn mish (3, 3, 3, 32)
2022-03-15 06:19:45.112243: I tensorflow/compiler/jit/xla_cpu_device.cc:41]
```

Figure 51 - Εκτέλεση μετατροπής yolov4 σε μορφή IR

Αν υπάρξουν σφάλματα, διακόπτεται η μετατροπή και εμφανίζεται το κατάλληλο μήνυμα λάθους. Διαφορετικά, εμφανίζεται μήνυμα επιτυχούς μετατροπής, ο χρόνος ολοκλήρωσης και οι διαδρομές που αποθηκεύτηκαν τα αρχεία xml και bin.

```
mize
  function_optimizer: Graph size after: 2182 nodes (1636), 4456 edges (
3908), time = 84.129ms.
  function_optimizer: function_optimizer did nothing. time = 2.017ms.

[ SUCCESS ] Generated IR version 10 model.
[ SUCCESS ] XML file: C:\Program Files (x86)\Intel\openvino_2021\deploy
ment_tools\open_model_zoo\tools\downloader\public\yolo-v4-tf\FP16\yolo-
v4-tf.xml
[ SUCCESS ] BIN file: C:\Program Files (x86)\Intel\openvino_2021\deploy
ment_tools\open_model_zoo\tools\downloader\public\yolo-v4-tf\FP16\yolo-
v4-tf.bin
[ SUCCESS ] Total execution time: 79.05 seconds.
It's been a while, check for a new version of Intel(R) Distribution of
OpenVINO(TM) toolkit here https://software.intel.com/content/www/us/en/
develop/tools/openvino-toolkit/download.html?cid=other&source=prod&camp
id=ww_2021_bu_IOTG_OpenVINO-2021-4-LTS&content=upg_all&medium=organic o
r on the GitHub*
```

Figure 52 - Επιτυχής ολοκλήρωση μετατροπής yolov4 σε μορφή IR

Στην συνέχεια, αντιγράφονται τα αρχεία στο Raspberry Pi.

10.3 Δοκιμές προσομοίωσης ανίχνευσης πραγματικών συνθηκών

Σε αυτό το σημείο, αφού έχουν συγκεντρωθεί τα νευρωνικά δίκτυα και υπάρχουν τα απαραίτητα βίντεο που θα χρησιμοποιηθούν για τις δοκιμές προσομοίωσης χρήσης πραγματικών συνθηκών, οργανώνονται στον παρακάτω πίνακα. Πέρα από τα βίντεο που εγγράφηκαν, έχουν ληφθεί από το διαδίκτυο και 2 βίντεο από δρόμους που περιλαμβάνουν πληθώρα οχημάτων και ανθρώπων για την δοκιμή και την σύγκριση των διαφορετικών γωνιών και συνθηκών.

Στην συνέχεια, θα παρατεθεί ο κώδικας που εκτελείται για την διαδικασία του Inferencing. Περιλαμβάνει την ανάγνωση των βίντεο, δηλαδή των εικόνων που τα συνθέτουν, την είσοδο των νευρωνικών δικτύων, την εξαγωγή συμπερασμάτων και την προβολή των αποτελεσμάτων.

Κατά τις δοκιμές, θα οριστεί το -nireq (παράλληλα αιτήματα Inferencing) στο 2, καθώς μετά από δοκιμές είναι η βέλτιστη επιλογή για το μεγαλύτερο ποσοστό των περιπτώσεων στον συνδυασμό Raspberry Pi και Intel Neural Compute Stick 2

10.3.1 Πίνακας δοκιμών ανίχνευσης

Νευρωνικό Δίκτυο	Βίντεο	Αποτελέσματα
pedestrian-and-vehicle-detector-adas-0001 MobileNet v1.0 + SSD	01-video02_traffic_side.h264	Latency : 153,3ms
		FPS : 11,5
	02-video08_traffic_front.h264	Latency : 158,1ms
		FPS : 11,0
	03-highway_footage_short.mp4	Latency : 150,9ms
		FPS : 11,5
	04-cars_side_view_mini.mp4	Latency : 185,8ms
		FPS : 9,5
person-vehicle-bike-detection-2004 MobileNetV2	01-video02_traffic_side.h264	Latency : 253,5ms
		FPS : 7,0
	02-video08_traffic_front.h264	Latency : 250,4ms
		FPS : 7,2
	03-highway_footage_short.mp4	Latency : 259,2ms
		FPS : 6,9
	04-cars_side_view_mini.mp4	Latency : 244,4ms
		FPS : 7,5

person-vehicle-bike-detection-crossroad-1016 MobileNetV2 + SSD	01-video02_traffic_side.h264	Latency : 182,2ms
		FPS : 9,4
	02-video08_traffic_front.h264	Latency : 180,4ms
		FPS : 9,9
	03-highway_footage_short.mp4	Latency : 181,9ms
		FPS : 9,6
	04-cars_side_view_mini.mp4	Latency : 195,1ms
		FPS : 9,1
person-vehicle-bike-detection-crossroad-yolov3-1020 Yolo V3	01-video02_traffic_side.h264	Latency : 4927,4ms
		FPS : 0,3
	02-video08_traffic_front.h264	Latency : 4900,6ms
		FPS : 0,3
	03-highway_footage_short.mp4	Latency : 6464,0ms
		FPS : 0,3
	04-cars_side_view_mini.mp4	Latency : 5084,9ms
		FPS : 0,3
vehicle-detection-0202 MobileNetV2	01-video02_traffic_side.h264	Latency : 168,6ms
		FPS : 10
	02-video08_traffic_front.h264	Latency : 167,5ms
		FPS : 10,4
	03-highway_footage_short.mp4	Latency : 161,7ms
		FPS : 10,8
	04-cars_side_view_mini.mp4	Latency : 189,7ms
		FPS : 9,4
vehicle-detection-adas-0002 MobileNet v1	01-video02_traffic_side.h264	Latency : 153,6ms
		FPS : 11,1
	02-video08_traffic_front.h264	Latency : 154,9ms
		FPS : 11,2
	03-highway_footage_short.mp4	Latency : 139,9ms
		FPS : 11,7
	04-cars_side_view_mini.mp4	Latency : 188,1ms
		FPS : 9,5

yolo-v2-tiny-vehicle-detection-0001 Yolo V2	01-video02_traffic_side.h264	Latency : 60,1ms
		FPS : 26,1
	02-video08_traffic_front.h264	Latency : 60,1ms
		FPS : 25,9
	03-highway_footage_short.mp4	Latency : 61,7ms
		FPS : 25,3
	04-cars_side_view_mini.mp4	Latency : 61,6ms
		FPS : 24,9
yolo-v4-tf Yolo V4	01-video02_traffic_side.h264	Latency : 4406,5ms
		FPS : 0,4
	02-video08_traffic_front.h264	Latency : 4481,9ms
		FPS : 0,4
	03-highway_footage_short.mp4	Latency : 4525,5ms
		FPS : 0,4
	04-cars_side_view_mini.mp4	Latency : 4521,4ms
		FPS : 0,4

Figure 53 - Πίνακας δοκιμών και αποτελεσμάτων νευρωνικών δικτύων ανίχνευσης

10.3.2 Πρόγραμμα python ανίχνευσης αντικειμένων

Το πρόγραμμα που θα εκτελεστεί αποτελείται από το κυρίως μέρος και επιμέρους πακέτα που εκτελούν τις διάφορες λειτουργίες.

10.3.2.1 Object_detection.py

10.3.2.1.1 Module import

```
import colorsys
import logging
import random
import sys
from argparse import ArgumentParser, SUPPRESS
from pathlib import Path
from time import perf_counter

import cv2
import numpy as np
from openvino.inference_engine import IECore

sys.path.append(str(Path(__file__).resolve().parents[2] / 'common/python'))


import models
import monitors
from pipelines import get_user_config, AsyncPipeline
from images_capture import open_images_capture
from performance_metrics import PerformanceMetrics
from helpers import resolution
```

Στο πρώτο κομμάτι του προγράμματος γίνονται όλες οι εισαγωγές των πακέτων.

colorsys	Πακέτο το οποίο χρησιμοποιείται για την μετατροπή των εικόνων μεταξύ των διαφορετικών χρωματικών συστημάτων (color systems)
logging	Πακέτο το οποίο χρησιμοποιείται για την παρακολούθηση των γεγονότων σε ένα πρόγραμμα και την αποστολή διαβαθμισμένων μηνυμάτων για σφάλματα, ενημερώσεις, προειδοποιήσεις
random	Πακέτο που χρησιμοποιείται για την παραγωγή τυχαίων αριθμών σε διάφορα εύρη, όπως μη προσημασμένους ακεραίους 8bit (0-255), ακέραιους 32bit (2^31 έως 2^31-1)
sys	Το βασικό πακέτο της Python, το οποίο χρησιμοποιείται για την εκτέλεση διαφόρων λειτουργιών του λειτουργικού περιβάλλοντος στην Python
argsparse	Πακέτο το οποίο χρησιμοποιείται για την εύκολη επικοινωνία του προγράμματος με τον χρήστη και για την εισαγωγή επιλογών στο σύστημα

pathlib	Πακέτο που χρησιμοποιείται για την αναπαράσταση διαδρομών στο σύστημα αρχείων για διάφορα λειτουργικά περιβάλλοντα
time	Πακέτο το οποίο προσφέρει λειτουργίες σχετικές με τον χρόνο, της ώρα, την ημερομηνία
cv2	Το πακέτο της OpenCV για την Python. Προσφέρει λειτουργίες σχετικά με την επεξεργασία εικόνας.
numpy	Πακέτο το οποίο προσφέρει διάφορες λειτουργίες σχετικές με πράξεις πινάκων. Ιδιαίτερα χρήσιμο, καθώς η ψηφιακή αναπαράσταση των εικόνων αποτελείται από ένα σύνολο πινάκων
openvino inference_engine	Το κύριο πακέτο του OpenVINO, το οποίο παρέχεται από την Intel για την Inference Engine. Αποτελεί την διεπαφή για την εκτέλεση του Inferencing, κυρίως για την επικοινωνία με το εκάστοτε plugin του υλικού
models	Πακέτο της Intel, το οποίο αποτελείται από διάφορες επιμέρους ενότητες – πακέτα και είναι υπεύθυνο για τις διάφορες λειτουργίες των νευρωνικών δικτύων, όπως η εισαγωγή τους, η μετατροπή του μεγέθους των εικόνων για να ταιριάζουν με τις εισόδους των νευρωνικών δικτύων, την σχεδίαση των πλαισίων που προκύπτουν μετά το Inferencin
monitors	Πακέτο της Intel, το οποίο αποτελεί μέσο επικοινωνίας με το αντίστοιχο πακέτο monitors που είναι γραμμένο σε C++. Σκοπός του είναι να εμφανίζει τα αποτελέσματα του Inferencing στις αντίστοιχες εικόνες
pipelines	Πακέτο της Intel, το οποίο δημιουργεί την δομή των δεδομένων που στέλνονται στο υλικό για να εκτελεστεί το Inferencing
images_capture	Πακέτο της Intel το οποίο χρησιμοποιείται για την εξαγωγή των δεδομένων των εικόνων ή βίντεο που εισέρχονται στο σύστημα για Inferencing
performance metrics	Πακέτο της Intel που χρησιμοποιείται για τις μετρήσεις των επιδόσεων του συστήματος, όπως η διακίνηση δεδομένων και η χρονική απόκριση του υλικού
helpers	Πακέτο της Intel το οποίο χρησιμοποιείται για την εισαγωγή τονισμένου κειμένου στις εικόνες που εξάγονται από το Inferencing μαζί με τα αποτελέσματά του, καθώς και την και την μετατροπή τους στις ζητούμενες διαστάσεις (argument : output_resolution)

Figure 54 - Πίνακας python modules του προγράμματος ανίχνευσης αντικειμένων

- colorsys [63]
- logging [64]
- random [65]

- sys [66]
- argsparse [67]
- pathlib [68]
- time [69]
- OpenCV – CV2 [70]
- Numpy [71]
- OpenVINO inference engine core [72]

10.3.2.1.2 Arguments parser

Στο επόμενο κομμάτι, δημιουργείται ο Arguments Parser. Σκοπός του είναι να δηλώσει τις δυνατές επιλογές που μπορεί να εισάγει ο χρήστης στο σύστημα κατά την εκτέλεσή του. Αποτελείται από τα εξής 4 στάδια

1) Add arguments – Οι προαπαιτούμενοι ορισμοί που πρέπει να γίνουν από τον χρήστη. Χωρίς αυτούς δεν δύναται να εκτελεστεί το πρόγραμμα. Ζητούνται

- --model : η διαδρομή προς το νευρωνικό δίκτυο
- --architecture_type : η αρχιτεκτονική του νευρωνικού δικτύου
- --input : η είσοδος για την επεξεργασία και το Inferencing
- --device : η συσκευή στην οποία θα εκτελεστεί το Inferencing

2) Common model arguments – Ορισμοί που είναι κοινοί για όλες τις εκτελέσεις

- --labels : η διαδρομή προς το αρχείο που θα χρησιμοποιηθεί για τις ετικέτες αναγώρισης
- --prob_threshold : το όριο της βεβαιότητας που χρησιμοποιείται για τον διαχωρισμό των ανιχνεύσεων έγκυρες και άκυρες
- --keep_aspect_ratio : σε περίπτωση αλλαγής διαστάσεων της εικόνας, να διατηρείται η αναλογία ύψους και πλάτους
- --input_size : δήλωση των διαστάσεων της πρώτης εικόνας που θα χρησιμοποιηθεί για την ανάλυση κειμένου

```
def build_argparser():
    parser = ArgumentParser(add_help=False)
    args = parser.add_argument_group('Options')
    args.add_argument('-h', '--help', action='help', default=SUPPRESS,
help='Show this help message and exit.')
    args.add_argument('-m', '--model', help='Required. Path to an .xml file
with a trained model.',
                      required=True, type=Path)
    args.add_argument('-at', '--architecture_type', help='Required. Specify
model\' architecture type.',
                      type=str, required=True, choices=('ssd', 'yolo',
'yolov4', 'faceboxes', 'centernet', 'ctpn',
                                                        'retinaface',
'ultra_lightweight_face_detection',
                                                        'retinaface-
pytorch'))
    args.add_argument('-i', '--input', required=True,
                      help='Required. An input to process. The input must
be a single image, '
                           'a folder of images, video file or camera id.')
    args.add_argument('-d', '--device', default='CPU', type=str,
                      help='Optional. Specify the target device to infer
on; CPU, GPU, HDDL or MYRIAD is '
                           'acceptable. The demo will look for a suitable
plugin for device specified. '
                           'Default value is CPU.')

    common_model_args = parser.add_argument_group('Common model options')
    common_model_args.add_argument('--labels', help='Optional. Labels
mapping file.', default=None, type=str)
    common_model_args.add_argument('-t', '--prob_threshold', default=0.5,
type=float,
                                   help='Optional. Probability threshold
for detections filtering.')
    common_model_args.add_argument('--keep_aspect_ratio',
action='store_true', default=False,
                                   help='Optional. Keeps aspect ratio on
resize.')
    common_model_args.add_argument('--input_size', default=(600, 600),
type=int, nargs=2,
                                   help='Optional. The first image size
used for CTPN model reshaping. '
                                        'Default: 600 600. Note that
submitted images should have the same resolution, '
                                        'otherwise predictions might be
incorrect.')
```

3) Inference arguments – Ορισμοί σχετικά με την εκτέλεση του Inferencing

- ---num_infer_requests : ο μέγιστος αριθμός ενεργών παράλληλων αιτημάτων Inferencing
- --num_streams : ο μέγιστος αριθμός παράλληλων καναλιών που θα στέλνουν δεδομένα
- --num_threads : ο μέγιστος αριθμός των λογικών πυρήνων που θα χρησιμοποιηθούν για Inferencing. Μόνο για εκτέλεση σε CPU

4) IO arguments – Ορισμοί για το I/O (input-output) της εκτέλεσης

- --loop : εκτέλεση ως κλειστός βρόχος χωρίς τερματισμό του Inferencing

- --output : διαδρομή και ονομασία του αρχείου που θα αποθηκευτούν οι εικόνες που εξάγονται από το Inferencing
- --output_limit : όριο εικόνων που θα εγγράφονται στο αρχείο της εξόδου
- --no_show : εκτέλεση χωρίς προβολή των αποτελεσμάτων του Inferencing
- --output_resolution : διαστάσεις των εικόνων που εγγράφονται στο αρχείο εξαγωγής

```
infer_args = parser.add_argument_group('Inference options')
infer_args.add_argument('-nireq', '--num_infer_requests', help='Optional.
Number of infer requests',
                        default=0, type=int)
infer_args.add_argument('-nstreams', '--num_streams',
                        help='Optional. Number of streams to use for
inference on the CPU or/and GPU in throughput '
                             'mode (for HETERO and MULTI device cases use
format '
                             '<device1>:<nstreams1>,<device2>:<nstreams2>
or just <nstreams>).',
                        default='', type=str)
infer_args.add_argument('-nthreads', '--num_threads', default=None,
type=int,
                        help='Optional. Number of threads to use for
inference on CPU (including HETERO cases).')

io_args = parser.add_argument_group('Input/output options')
io_args.add_argument('--loop', default=False, action='store_true',
                     help='Optional. Enable reading the input in a loop.')
io_args.add_argument('-o', '--output', required=False,
                     help='Optional. Name of the output file(s) to save.')
io_args.add_argument('-limit', '--output_limit', required=False,
default=1000, type=int,
                     help='Optional. Number of frames to store in output. '
                          'If 0 is set, all frames are stored.')
io_args.add_argument('--no_show', help="Optional. Don't show output.",
action='store_true')
io_args.add_argument('--output_resolution', default=None, type=resolution,
                     help='Optional. Specify the maximum output window
resolution '
                          'in (width x height) format. Example: 1280x720. '
                          'Input frame size used by default.')
io_args.add_argument('-u', '--utilization_monitors', default='', type=str,
                     help='Optional. List of monitors to show initially.')
```

5) Input transform arguments – Ορισμοί σχετικά με τα χαρακτηριστικά των εικόνων εισόδου

- --reverse_input_channels : εναλλαγή των χρωματικών καναλιών των εικόνων (RGB σε BGR)
- -- mean_values : ομαλοποίηση της εισόδου πραγματοποιώντας αφαίρεση των μέσων τιμών από το κάθε κανάλι της εισόδου
- --scale_values : διαίρεση με τις τιμές του εκάστοτε καναλιού για μετατροπή κλίμακας

6) Debug arguments – Χρήσιμοι ορισμοί για την αποσφαλμάτωση

- --raw_output_message : ενεργοποίηση των ακατέργαστων δεδομένων που εξάγονται από το Inferencing

```
input_transform_args = parser.add_argument_group('Input transform options')
input_transform_args.add_argument('--reverse_input_channels',
default=False, action='store_true',
                                  help='Optional. Switch the input channels
order from '
                                       'BGR to RGB.')
input_transform_args.add_argument('--mean_values', default=None,
type=float, nargs=3,
                                  help='Optional. Normalize input by
subtracting the mean '
                                       'values per channel. Example: 255
255 255')
input_transform_args.add_argument('--scale_values', default=None,
type=float, nargs=3,
                                  help='Optional. Divide input by scale
values per channel. '
                                       'Division is applied after mean
values subtraction. '
                                       'Example: 255 255 255')

debug_args = parser.add_argument_group('Debug options')
debug_args.add_argument('-r', '--raw_output_message', help='Optional.
Output inference results raw values showing.',
                        default=False, action='store_true')
return parser
```

10.3.2.1.3 ColorPalette

Η συγκεκριμένη κλάση χρησιμοποιείται για τον χρωματισμό των bounding boxes και οποιασδήποτε άλλης εγγραφής πάνω στην εικόνα εξόδου για τις διαφορετικές κατηγορίες των αντικειμένων.

```
class ColorPalette:
    def __init__(self, n, rng=None):
        assert n > 0

        if rng is None:
            rng = random.Random(0xACE)

        candidates_num = 100
        hsv_colors = [(1.0, 1.0, 1.0)]
        for _ in range(1, n):
            colors_candidates = [(rng.random(), rng.uniform(0.8, 1.0),
rng.uniform(0.5, 1.0))
                                 for _ in range(candidates_num)]
            min_distances = [self.min_distance(hsv_colors, c) for c in
colors_candidates]
            arg_max = np.argmax(min_distances)
            hsv_colors.append(colors_candidates[arg_max])

        self.palette = [self.hsv2rgb(*hsv) for hsv in hsv_colors]

    @staticmethod
    def dist(c1, c2):
        dh = min(abs(c1[0] - c2[0]), 1 - abs(c1[0] - c2[0])) * 2
        ds = abs(c1[1] - c2[1])
        dv = abs(c1[2] - c2[2])
        return dh * dh + ds * ds + dv * dv

    @classmethod
    def min_distance(cls, colors_set, color_candidate):
        distances = [cls.dist(o, color_candidate) for o in colors_set]
        return np.min(distances)

    @staticmethod
    def hsv2rgb(h, s, v):
        return tuple(round(c * 255) for c in colorsys.hsv_to_rgb(h, s, v))

    def __getitem__(self, n):
        return self.palette[n % len(self.palette)]

    def __len__(self):
        return len(self.palette)
```

10.3.2.1.4 Get model

Η συνάρτηση get_model χρησιμοποιείται για την εισαγωγή του νευρωνικού δικτύου και την εκτέλεση των κατάλληλων μετατροπών που απαιτούνται, με βάση την ορισμένη αρχιτεκτονική

```
def get_model(ie, args):
    input_transform = models.InputTransform(args.reverse_input_channels,
args.mean_values, args.scale_values)
    common_args = (ie, args.model, input_transform)
    if args.architecture_type in ('ctpn', 'yolo', 'yolov4', 'retinaface',
                                  'retinaface-pytorch') and not
input_transform.is_trivial:
        raise ValueError("{} model doesn't support input
transforms.".format(args.architecture_type))

    if args.architecture_type == 'ssd':
        return models.SSD(*common_args, labels=args.labels,
keep_aspect_ratio_resize=args.keep_aspect_ratio)
    elif args.architecture_type == 'ctpn':
        return models.CTPN(ie, args.model, input_size=args.input_size,
threshold=args.prob_threshold)
    elif args.architecture_type == 'yolo':
        return models.YOLO(ie, args.model, labels=args.labels,
                           threshold=args.prob_threshold,
keep_aspect_ratio=args.keep_aspect_ratio)
    elif args.architecture_type == 'yolov4':
        return models.YoloV4(ie, args.model, labels=args.labels,
                             threshold=args.prob_threshold,
keep_aspect_ratio=args.keep_aspect_ratio)
    elif args.architecture_type == 'faceboxes':
        return models.FaceBoxes(*common_args,
threshold=args.prob_threshold)
    elif args.architecture_type == 'centernet':
        return models.CenterNet(*common_args, labels=args.labels,
threshold=args.prob_threshold)
    elif args.architecture_type == 'retinaface':
        return models.RetinaFace(ie, args.model,
threshold=args.prob_threshold)
    elif args.architecture_type == 'ultra_lightweight_face_detection':
        return models.UltraLightweightFaceDetection(*common_args,
threshold=args.prob_threshold)
    elif args.architecture_type == 'retinaface-pytorch':
        return models.RetinaFacePyTorch(ie, args.model,
threshold=args.prob_threshold)
    else:
        raise RuntimeError('No model type or invalid model type (-at)
provided: {}'.format(args.architecture_type))
```

10.3.2.1.5 Draw detections

Η συνάρτηση draw detections χρησιμοποιείται για τον σχεδιασμό και την εγγραφή των αποτελεσμάτων του Inferencing επάνω στην εικόνα εξόδου.

```
def draw_detections(frame, detections, palette, labels, threshold,
output_transform):
    size = frame.shape[:2]
    frame = output_transform.resize(frame)
    for detection in detections:
        if detection.score > threshold:
            class_id = int(detection.id)
            color = palette[class_id]
            det_label = labels[class_id] if labels and len(labels) >=
class_id else '#{}'.format(class_id)
            xmin = max(int(detection.xmin), 0)
            ymin = max(int(detection.ymin), 0)
            xmax = min(int(detection.xmax), size[1])
            ymax = min(int(detection.ymax), size[0])
            xmin, ymin, xmax, ymax = output_transform.scale([xmin, ymin,
xmax, ymax])
            cv2.rectangle(frame, (xmin, ymin), (xmax, ymax), color, 2)
            cv2.putText(frame, '{} {:.1%}'.format(det_label,
detection.score),
                        (xmin, ymin - 7), cv2.FONT_HERSHEY_COMPLEX, 0.6,
color, 1)
            if isinstance(detection, models.DetectionWithLandmarks):
                for landmark in detection.landmarks:
                    landmark = output_transform.scale(landmark)
                    cv2.circle(frame, (int(landmark[0]), int(landmark[1])),
2, (0, 255, 255), 2)
    return frame
```

10.3.2.1.6 Print raw results

Η συνάρτηση print raw results χρησιμοποιείται για την εκτύπωση των δεδομένων που εξήχθησαν από το Inferencing, στην γραμμή εντολών που εκτελείται το πρόγραμμα.

```
def print_raw_results(size, detections, labels, threshold):
    log.info(' Class ID | Confidence | XMIN | YMIN | XMAX | YMAX ')
    for detection in detections:
        if detection.score > threshold:
            xmin = max(int(detection.xmin), 0)
            ymin = max(int(detection.ymin), 0)
            xmax = min(int(detection.xmax), size[1])
            ymax = min(int(detection.ymax), size[0])
            class_id = int(detection.id)
            det_label = labels[class_id] if labels and len(labels) >=
class_id else '#{}'.format(class_id)
            log.info('{:^9} | {:10f} | {:4} | {:4} | {:4} | {:4} '
                     .format(det_label, detection.score, xmin, ymin, xmax,
ymax))
```

10.3.2.1.7 Main()

Το κύριο κομμάτι του προγράμματος, το οποίο εκτελείται για να πραγματοποιήσει την σύνδεση των επιμέρους τμημάτων και την επίτευξη του τελικού σκοπού, βρίσκεται στην κύρια συνάρτηση.

Αρχικά δηλώνονται οι ορισμοί του χρήστη σε μία μεταβλητή και αντιστοιχείται ο πυρήνας της Inference Engine σε μία άλλη.

Στην συνέχεια, εμφανίζεται μήνυμα εκκίνησης στην γραμμή εντολών.

Επίσης, αποθηκεύονται σε μεταβλητές το νευρωνικό δίκτυο, η δομή της ροής των δεδομένων, οι μετρήσεις επιδόσεων και η χρωματική παλέτα και αρχικοποιείται η σύλληψη των εικόνων και το πακέτο εγγραφής του βίντεο της εξόδου.

```
def main():
    args = build_argparser().parse_args()

    log.info('Initializing Inference Engine...')
    ie = IECore()

    plugin_config = get_user_config(args.device, args.num_streams,
args.num_threads)

    log.info('Loading network...')

    model = get_model(ie, args)

    detector_pipeline = AsyncPipeline(ie, model, plugin_config,
                                      device=args.device,
max_num_requests=args.num_infer_requests)

    cap = open_images_capture(args.input, args.loop)

    next_frame_id = 0
    next_frame_id_to_show = 0

    log.info('Starting inference...')
    print("To close the application, press 'CTRL+C' here or switch to the
output window and press ESC key")

    palette = ColorPalette(len(model.labels) if model.labels else 100)
    metrics = PerformanceMetrics()
    presenter = None
    output_transform = None
    video_writer = cv2.VideoWriter()
```

10.3.2.1.8 while loop κύριου προγράμματος

Ο βρόχος While εκτελείται συνεχώς. Ο μόνος τρόπος να σταματήσει η εκτέλεσή του, είναι με τα σημεία break που υπάρχουν σε συγκεκριμένα σημεία μέσα του.

Αρχικά γίνεται έλεγχος ότι το pipeline δεν εμφανίζει κάποιο σφάλμα. Στην συνέχεια, ξεκινάει η επεξεργασία των ολοκληρωμένων αιτημάτων Inferencing που έρχονται από το Neural Compute Stick μέσω του αγωγού δεδομένων (pipeline).

Εφόσον υπάρχουν αποτελέσματα, αποθηκεύονται σε μεταβλητές. Επίσης, αν έχει ζητηθεί από τον χρήστη, εμφανίζονται στην γραμμή εντολών τα ακατέργαστα δεδομένα που εξήχθησαν.

Με την χρήση της κλάσης Presenter από το πακέτο Monitors, και την OpenCV, εκτυπώνονται τα αποτελέσματα του Inferencing πάνω στην εκάστοτε εικόνα. Επίσης, καταχωρούνται πληροφορίες για τις μετρήσεις απόδοσης και διακόπτεται η εκτέλεση του προγράμματος, εφόσον ζητηθεί από τον χρήστη με το κουμπί Escape.

```
while True:
    if detector_pipeline.callback_exceptions:
        raise detector_pipeline.callback_exceptions[0]
    # Process all completed requests
    results = detector_pipeline.get_result(next_frame_id_to_show)
    if results:
        objects, frame_meta = results
        frame = frame_meta['frame']
        start_time = frame_meta['start_time']

        if len(objects) and args.raw_output_message:
            print_raw_results(frame.shape[:2], objects, model.labels,
args.prob_threshold)

        presenter.drawGraphs(frame)
        frame = draw_detections(frame, objects, palette, model.labels,
args.prob_threshold, output_transform)
        metrics.update(start_time, frame)

        if video_writer.isOpened() and (args.output_limit <= 0 or
next_frame_id_to_show <= args.output_limit-1):
            video_writer.write(frame)
        next_frame_id_to_show += 1

        if not args.no_show:
            cv2.imshow('Detection Results', frame)
            key = cv2.waitKey(1)

            ESC_KEY = 27
            # Quit.
            if key in {ord('q'), ord('Q'), ESC_KEY}:
                break
            presenter.handleKey(key)
        continue
```

Αφού έχει ολοκληρωθεί η εξαγωγή και η εγγραφή των αποτελεσμάτων του Inferencing, ετοιμάζεται η επόμενη εικόνα. Εισάγεται από το βίντεο και μορφοποιείται για την είσοδο στο νευρωνικό δίκτυο. Τέλος, μεταφέρεται μέσω του pipeline στο NCS2 για Inferencing. Σε περίπτωση που δεν είναι διαθέσιμο το pipeline, περιμένει για να αδειάσει.

```
#2nd part
        if detector_pipeline.is_ready():
            # Get new image/frame
            start_time = perf_counter()
            frame = cap.read()
            if frame is None:
                if next_frame_id == 0:
                    raise ValueError("Can't read an image from the input")
                break
            if next_frame_id == 0:
                output_transform = models.OutputTransform(frame.shape[:2],
args.output_resolution)
                if args.output_resolution:
                    output_resolution = output_transform.new_resolution
                else:
                    output_resolution = (frame.shape[1], frame.shape[0])
                presenter = monitors.Presenter(args.utilization_monitors,
55, (round(output_resolution[0] / 4), round(output_resolution[1] / 8)))
                if args.output and not video_writer.open(args.output,
cv2.VideoWriter_fourcc(*'MJPG'), cap.fps(), output_resolution):
                    raise RuntimeError("Can't open video writer")
            # Submit for inference
            detector_pipeline.submit_data(frame, next_frame_id, {'frame':
frame, 'start_time': start_time})
            next_frame_id += 1

        else:
            # Wait for empty request
            detector_pipeline.await_any()

detector_pipeline.await_all()
```

Σε αυτό το τμήμα, επαναλαμβάνονται εντολές που έτρεξαν στην αρχή και απαιτούνται για την επεξεργασία των αποτελεσμάτων του Inferencing. Εξάγονται οι πληροφορίες από το pipeline που επιστρέφει δεδομένα από το Inferencing που έτρεξε στο NCS2, αποτυπώνονται στην εξαγόμενη εικόνα, αποθηκεύονται δεδομένα για τις μετρήσεις απόδοσης και εξάγονται οι τελικές εικόνες για εμφάνιση στον χρήστη ή για την αποθήκευσή τους σε αρχείο.

```
#3rd part
    # Process completed requests
    for next_frame_id_to_show in range(next_frame_id_to_show,
next_frame_id):
        results = detector_pipeline.get_result(next_frame_id_to_show)
        while results is None:
            results = detector_pipeline.get_result(next_frame_id_to_show)
        objects, frame_meta = results
        frame = frame_meta['frame']
        start_time = frame_meta['start_time']

        if len(objects) and args.raw_output_message:
            print_raw_results(frame.shape[:2], objects, model.labels,
args.prob_threshold)

        presenter.drawGraphs(frame)
        frame = draw_detections(frame, objects, palette, model.labels,
args.prob_threshold, output_transform)
        metrics.update(start_time, frame)

        if video_writer.isOpened() and (args.output_limit <= 0 or
next_frame_id_to_show <= args.output_limit-1):
            video_writer.write(frame)

        if not args.no_show:
            cv2.imshow('Detection Results', frame)
            key = cv2.waitKey(1)

            ESC_KEY = 27
            # Quit.
            if key in {ord('q'), ord('Q'), ESC_KEY}:
                break
            presenter.handleKey(key)
```

10.3.2.1.9 Print metrics & exit

Στο τελευταίο τμήμα του προγράμματος, γίνεται η εμφάνιση των στατιστικών απόδοσης και η έξοδος από το πρόγραμμα.

```
    metrics.print_total()
    print(presenter.reportMeans())


if __name__ == '__main__':
    sys.exit(main() or 0)
```

10.4 Εκτέλεση δοκιμών

Με βάση τον πίνακα δοκιμών, θα γίνει η εκτέλεση για όλα τα νευρωνικά δίκτυα στο εκάστοτε βίντεο.

Πρώτα αναγράφεται η εντολή για την εκτέλεση του εκάστοτε παραδείγματος και στην συνέχεια εμφανίζονται στιγμιότυπα με αποτυπωμένα αποτελέσματα από τις εκτελέσεις.

10.4.1 pedestrian-and-vehicle-detector-adas-0001

10.4.1.1 pedestrian-and-vehicle-detector-adas-0001 – 01-video02_traffic_side.h264

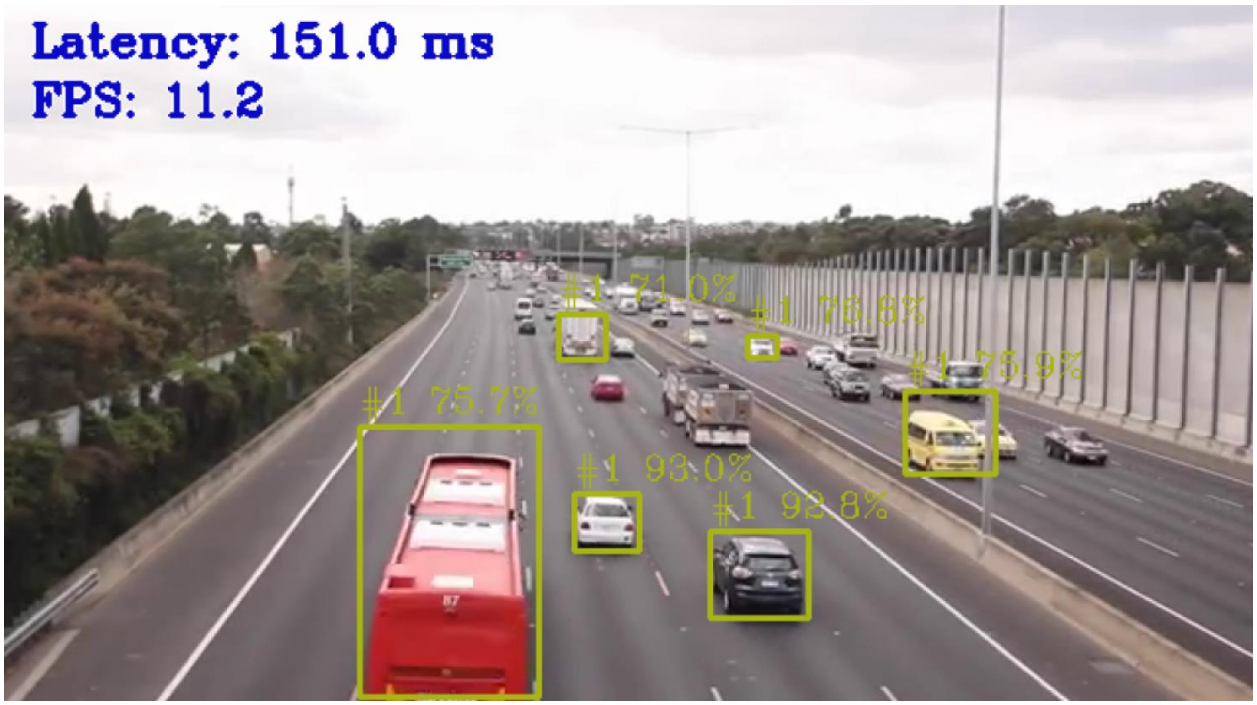
```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/pedestrian-and-vehicle-detector-adas-
0001/FP16/pedestrian-and-vehicle-detector-adas-0001.xml
    --architecture_type ssd
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```

10.4.1.2 pedestrian-and-vehicle-detector-adas-0001 – 02-video08_traffic_front.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/pedestrian-and-vehicle-detector-adas-
0001/FP16/pedestrian-and-vehicle-detector-adas-0001.xml
    --architecture_type ssd
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```

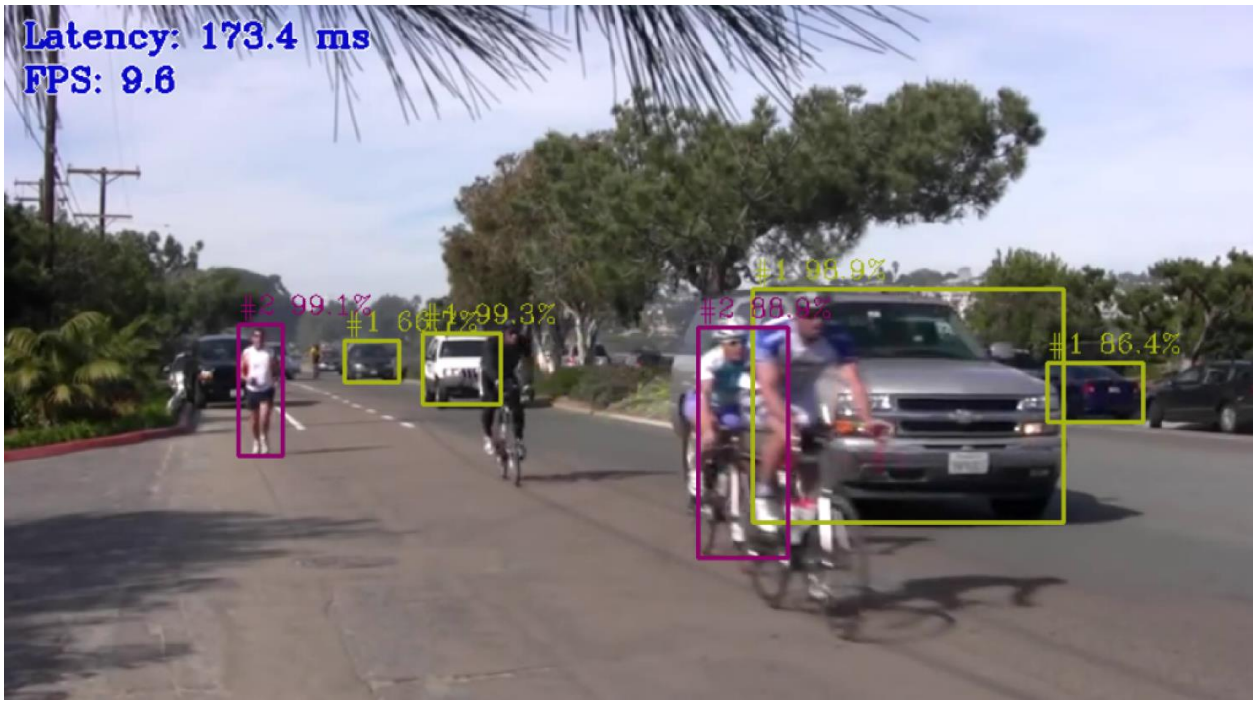
10.4.1.3 pedestrian-and-vehicle-detector-adas-0001 – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/pedestrian-and-vehicle-detector-adas-
0001/FP16/pedestrian-and-vehicle-detector-adas-0001.xml
    --architecture_type ssd
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```



10.4.1.4 pedestrian-and-vehicle-detector-adas-0001 – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/pedestrian-and-vehicle-detector-adas-
0001/FP16/pedestrian-and-vehicle-detector-adas-0001.xml
    --architecture_type ssd
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```



10.4.2 person-vehicle-bike-detection-2004

10.4.2.1 person-vehicle-bike-detection-2004 – 01-video02_traffic_side.h264

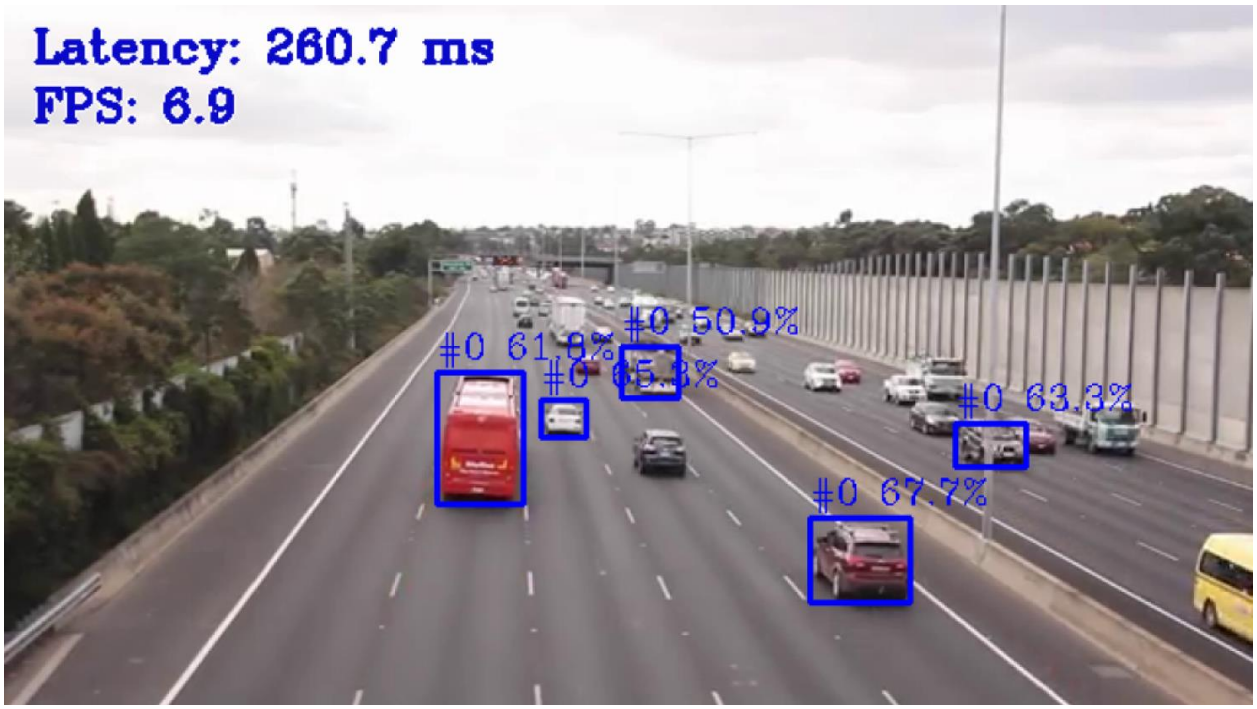
```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
2004/FP16/person-vehicle-bike-detection-2004.xml
    --architecture_type ssd
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```

10.4.2.2 person-vehicle-bike-detection-2004 – 02-video08_traffic_front.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
2004/FP16/person-vehicle-bike-detection-2004.xml
    --architecture_type ssd
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```

10.4.2.3 person-vehicle-bike-detection-2004 – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
2004/FP16/person-vehicle-bike-detection-2004.xml
    --architecture_type ssd
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```



10.4.2.4 person-vehicle-bike-detection-2004 – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
2004/FP16/person-vehicle-bike-detection-2004.xml
    --architecture_type ssd
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```

10.4.3 person-vehicle-bike-detection-crossroad-1016

10.4.3.1 person-vehicle-bike-detection-crossroad-1016 – 01-video02_traffic_side.h264

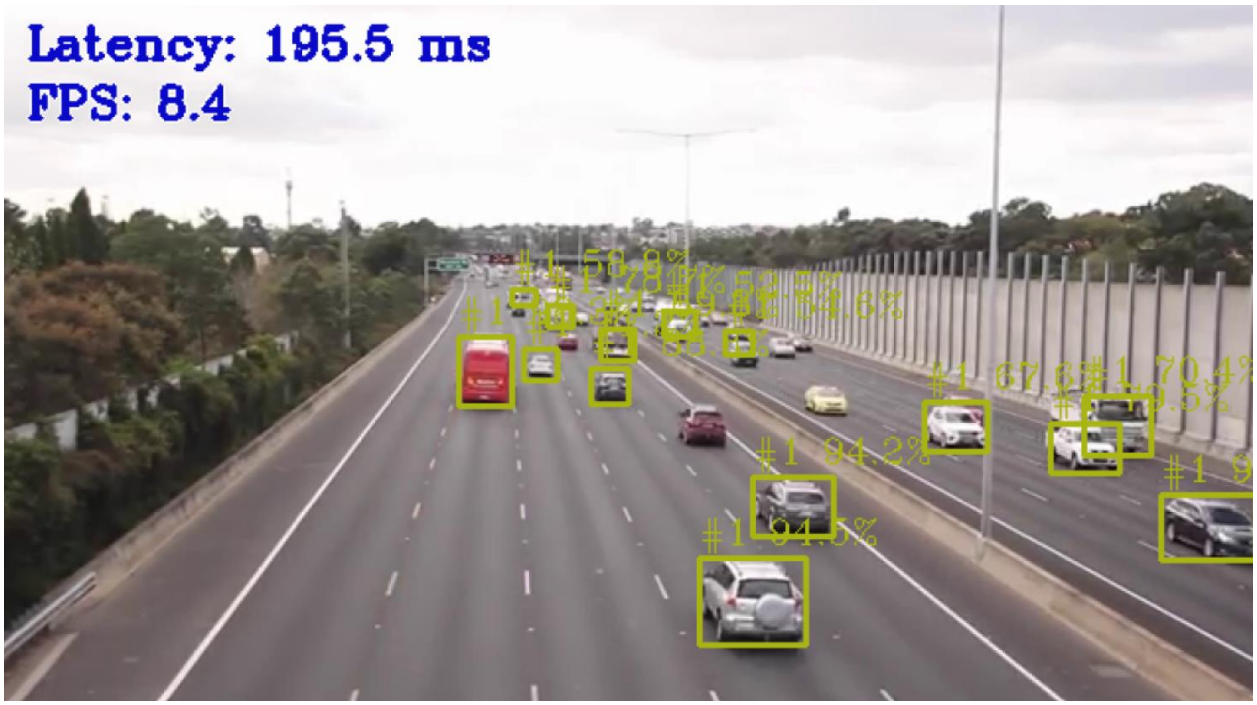
```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-1016/FP16/person-vehicle-bike-detection-crossroad-1016.xml
    --architecture_type ssd
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```

10.4.3.2 person-vehicle-bike-detection-crossroad-1016 – 02-video08_traffic_front.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-1016/FP16/person-vehicle-bike-detection-crossroad-1016.xml
    --architecture_type ssd
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```

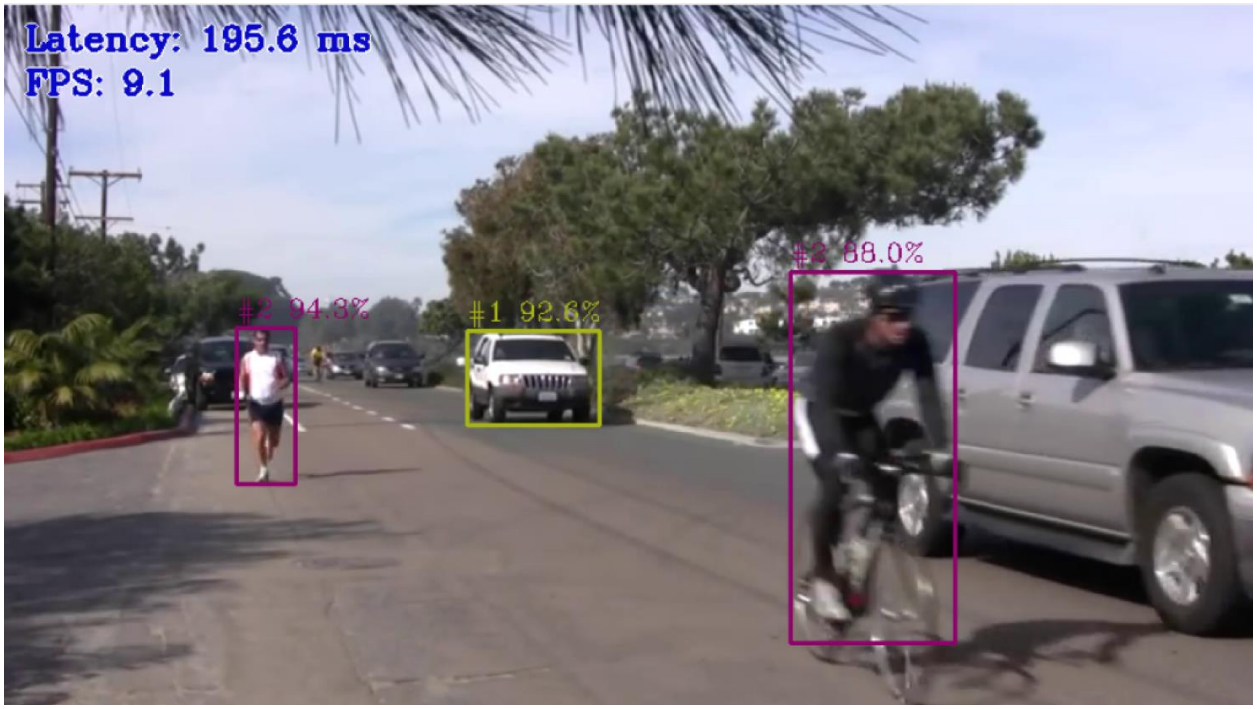
10.4.3.3 person-vehicle-bike-detection-crossroad-1016 – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-1016/FP16/person-vehicle-bike-detection-crossroad-1016.xml
    --architecture_type ssd
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```



10.4.3.4 person-vehicle-bike-detection-crossroad-1016 – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-1016/FP16/person-vehicle-bike-detection-crossroad-1016.xml
    --architecture_type ssd
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```



10.4.4 person-vehicle-bike-detection-crossroad-yolov3-1020

10.4.4.1 person-vehicle-bike-detection-crossroad-yolov3-1020 – 01-video02_traffic_side.h264

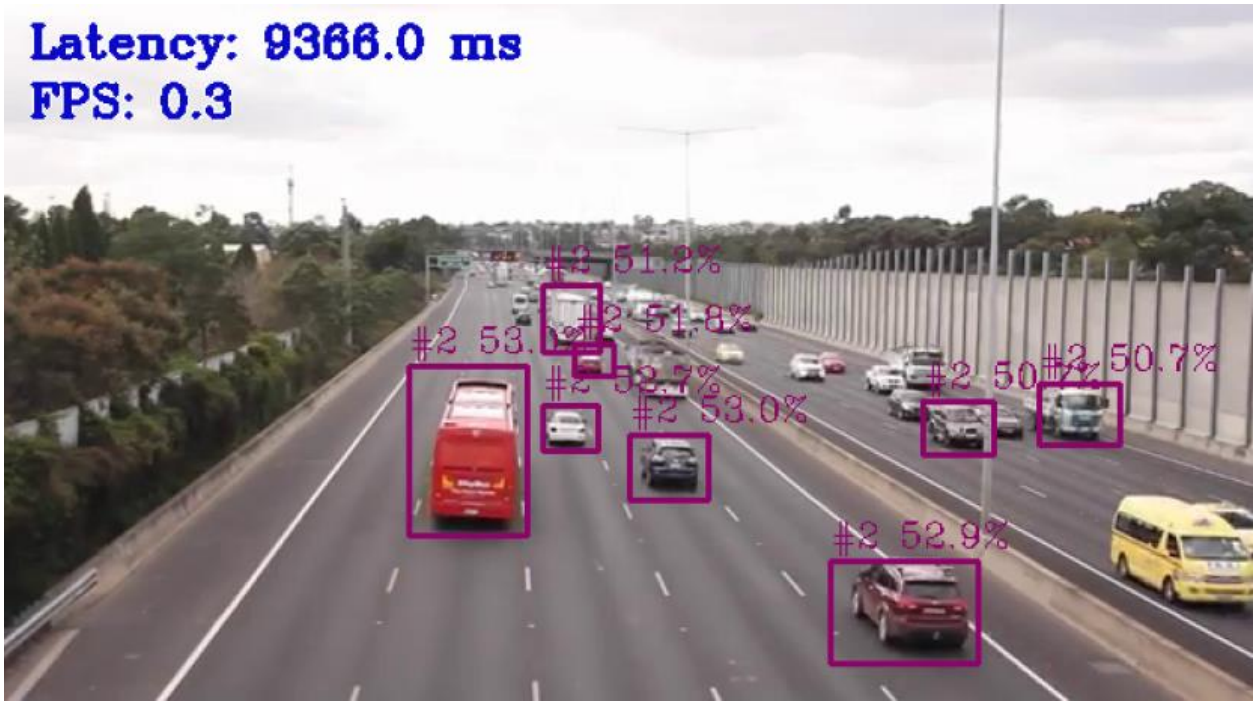
```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-yolov3-1020/FP16/person-vehicle-bike-detection-crossroad-yolov3-
1020.xml
    --architecture_type yolo
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```

10.4.4.2 person-vehicle-bike-detection-crossroad-yolov3-1020 – 02-video08_traffic_front.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-yolov3-1020/FP16/person-vehicle-bike-detection-crossroad-yolov3-
1020.xml
    --architecture_type yolo
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```

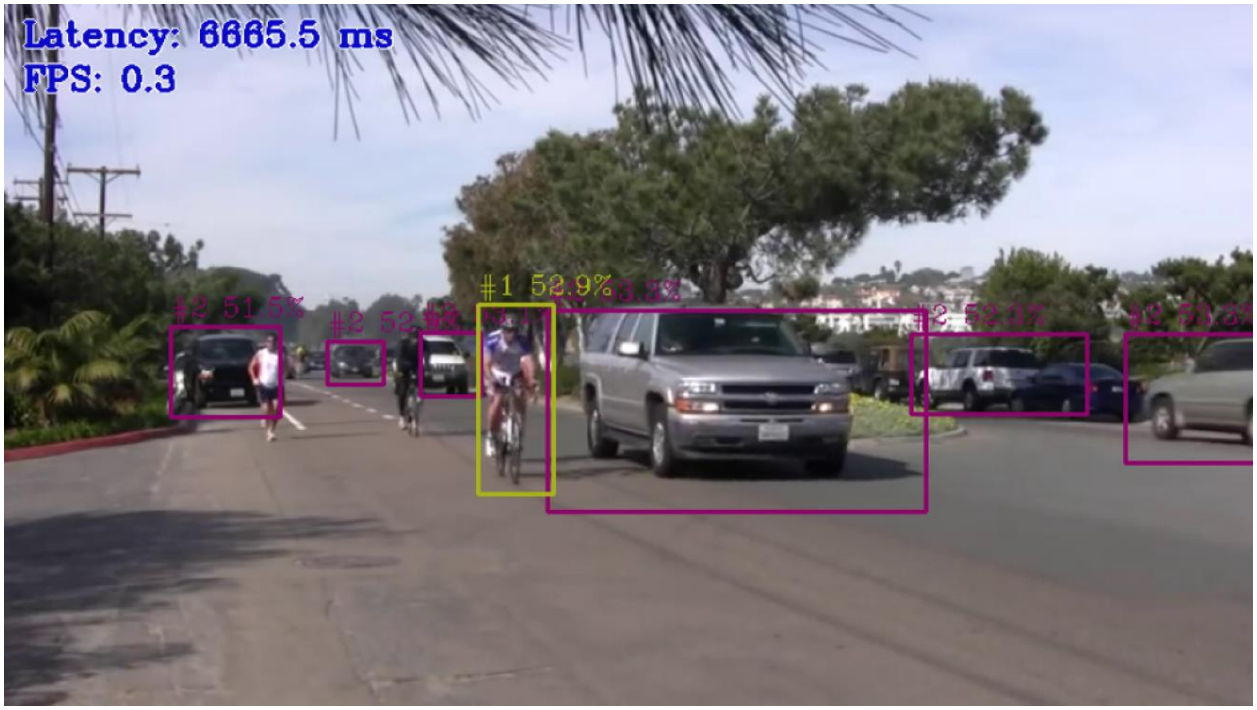
10.4.4.3 person-vehicle-bike-detection-crossroad-yolov3-1020 – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-yolov3-1020/FP16/person-vehicle-bike-detection-crossroad-yolov3-
1020.xml
    --architecture_type yolo
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```



10.4.4.4 person-vehicle-bike-detection-crossroad-yolov3-1020 – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/person-vehicle-bike-detection-
crossroad-yolov3-1020/FP16/person-vehicle-bike-detection-crossroad-yolov3-
1020.xml
    --architecture_type yolo
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```



10.4.5 vehicle-detection-0202

10.4.5.1 vehicle-detection-0202 – 01-video02_traffic_side.h264

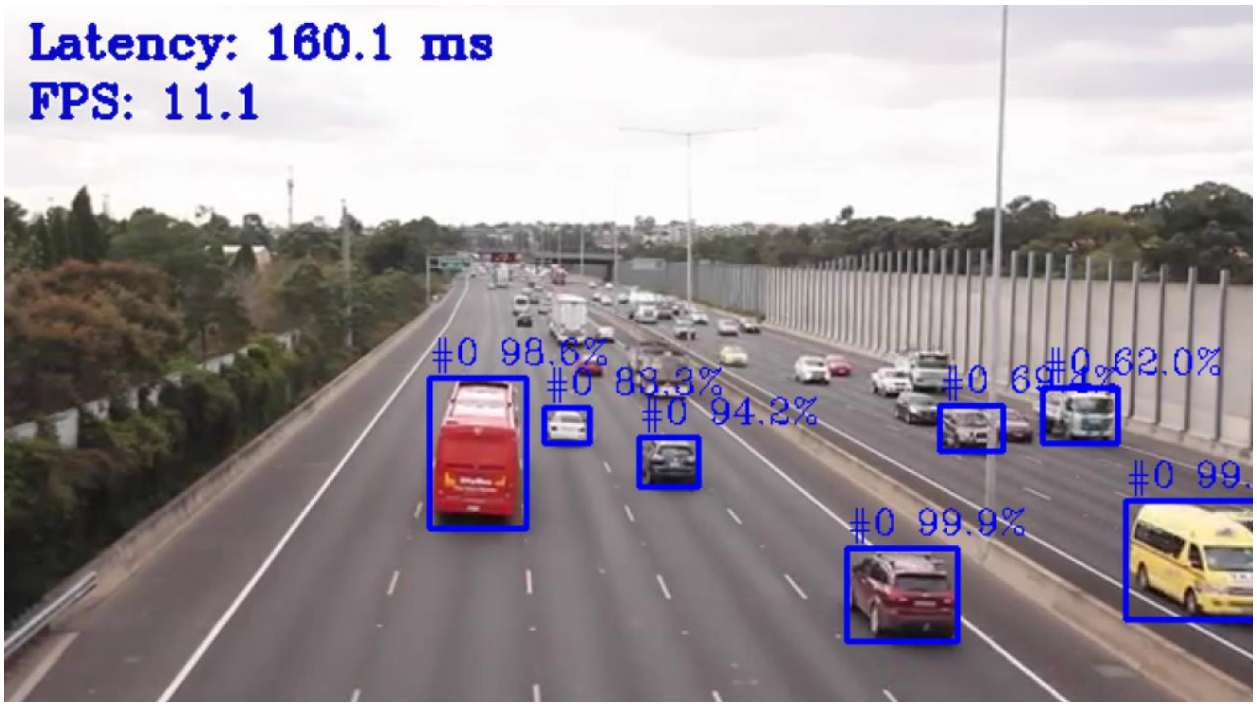
```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-0202/FP16/vehicle-
detection-0202.xml
    --architecture_type ssd
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```

10.4.5.2 vehicle-detection-0202 – 02-video08_traffic_front.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-0202/FP16/vehicle-
detection-0202.xml
    --architecture_type ssd
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```

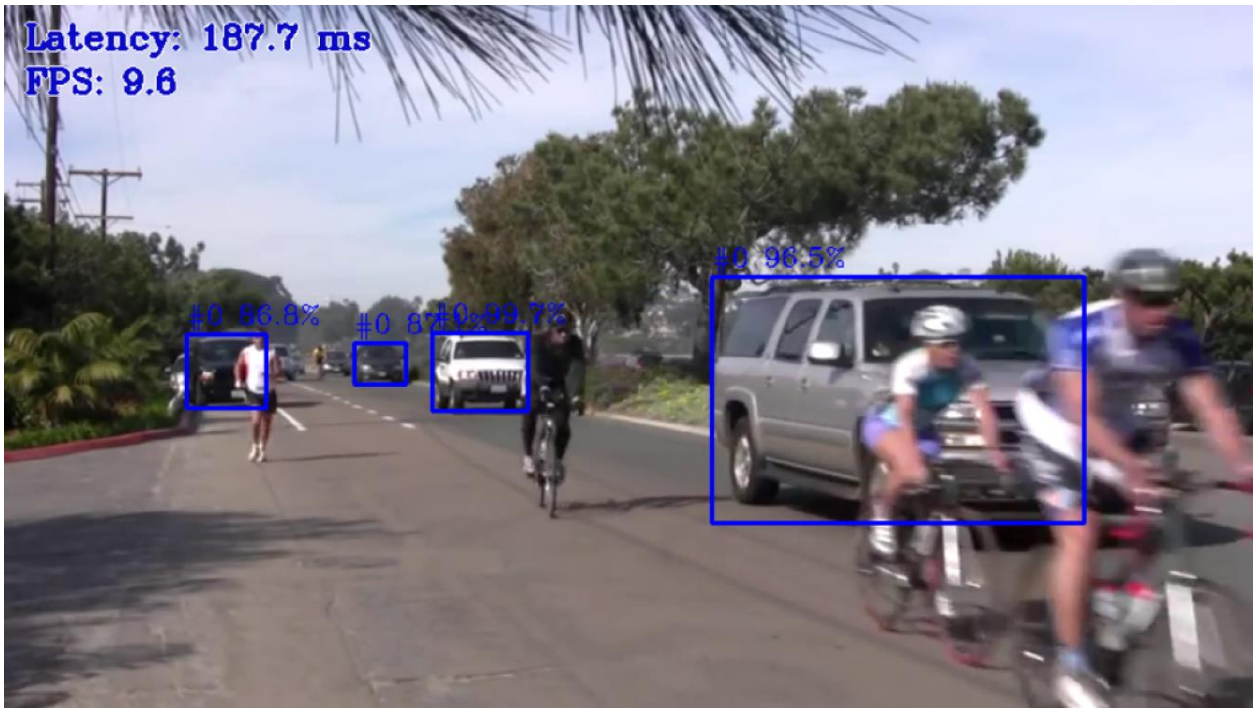
10.4.5.3 vehicle-detection-0202 – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-0202/FP16/vehicle-
detection-0202.xml
    --architecture_type ssd
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```



10.4.5.4 vehicle-detection-0202 – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-0202/FP16/vehicle-
detection-0202.xml
    --architecture_type ssd
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```



10.4.6 vehicle-detection-adas-0002

10.4.6.1 vehicle-detection-adas-0002 – 01-video02_traffic_side.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-adas-
0002/FP16/vehicle-detection-adas-0002.xml
    --architecture_type ssd
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```

10.4.6.2 vehicle-detection-adas-0002 – 02-video08_traffic_front.h264

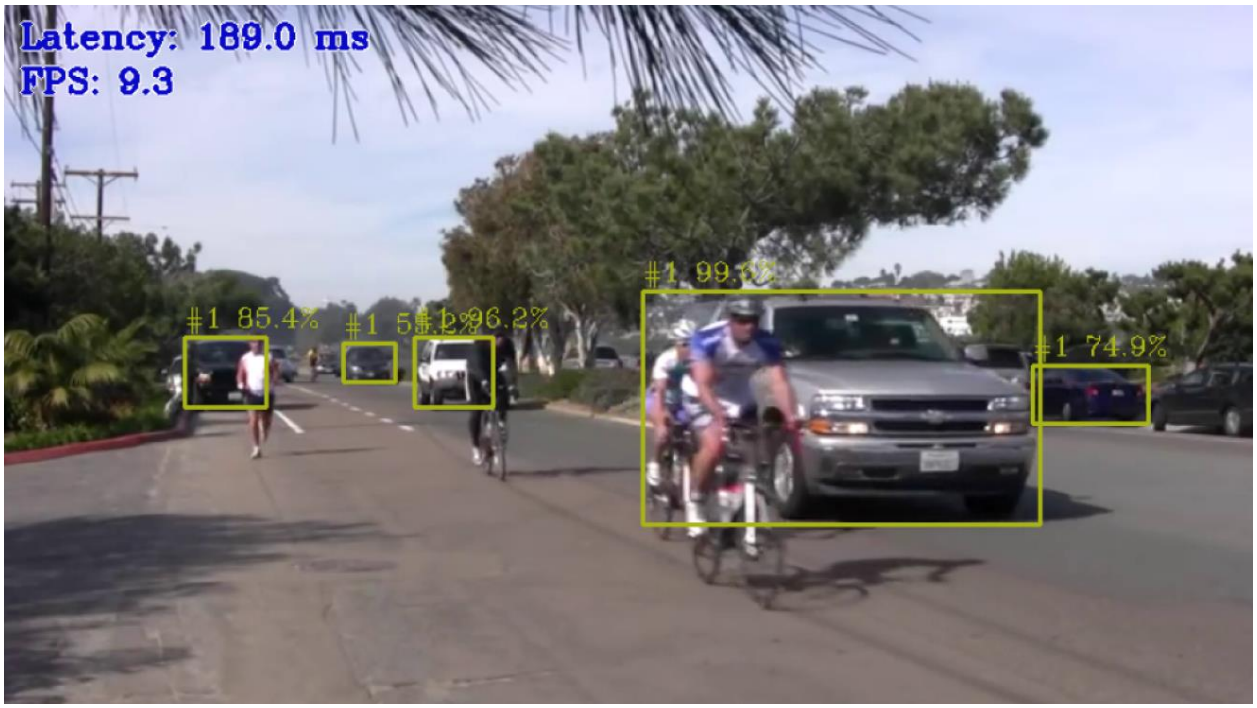
```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-adas-
0002/FP16/vehicle-detection-adas-0002.xml
    --architecture_type ssd
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```

10.4.6.3 vehicle-detection-adas-0002 – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-adas-
0002/FP16/vehicle-detection-adas-0002.xml
    --architecture_type ssd
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```

10.4.6.4 vehicle-detection-adas-0002 – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/vehicle-detection-adas-
0002/FP16/vehicle-detection-adas-0002.xml
    --architecture_type ssd
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```

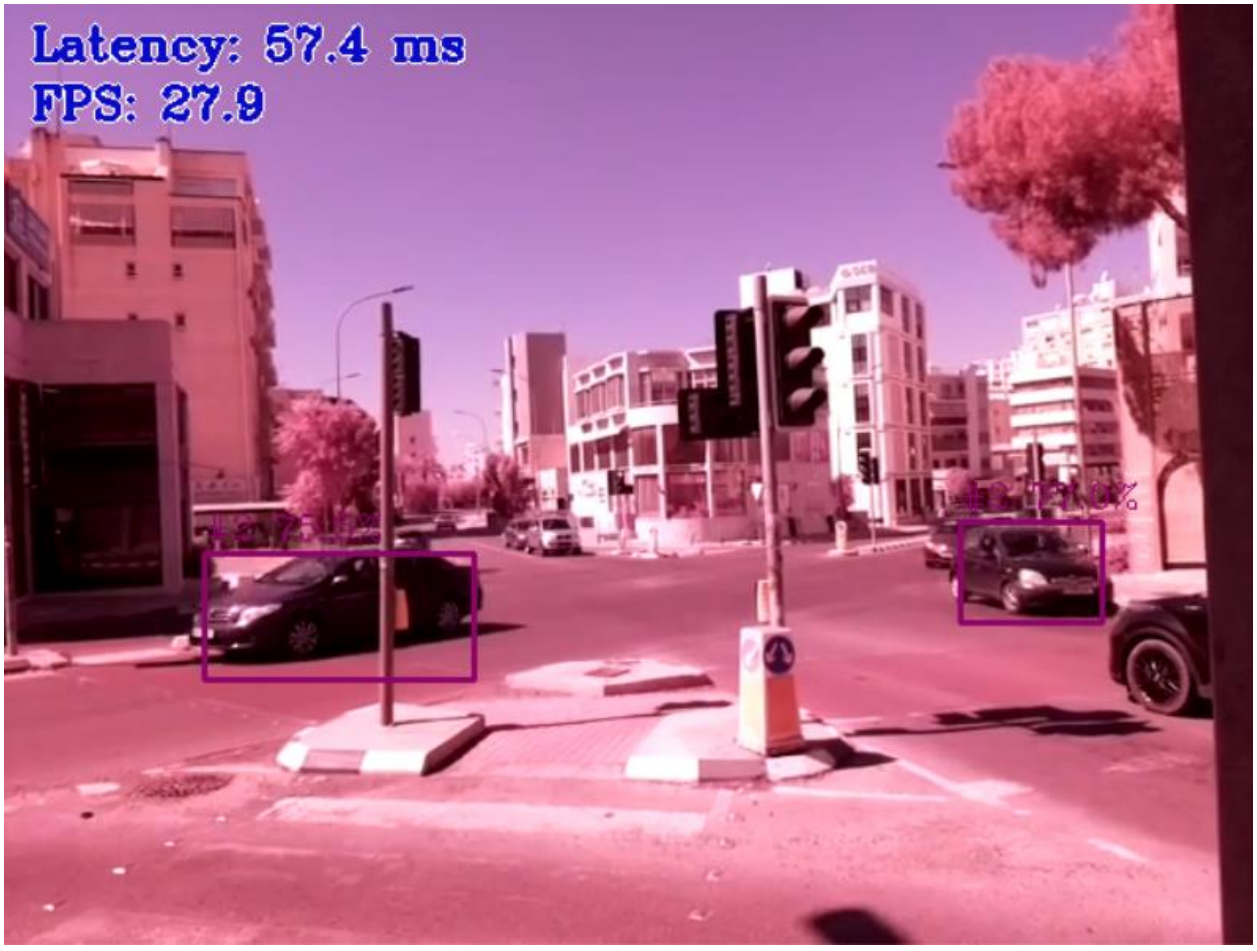


10.4.7 yolo-v2-tiny-vehicle-detection-0001

Λόγω προβληματικού yolo parser στην έκδοση του OpenVINO για το Raspberry Pi, οι δοκιμές για το yolo v2 έγιναν σε υπολογιστή με την εκτέλεση του Inferencing στο NCS2. Οι μόνες αλλαγές που απαιτούνται, είναι οι διαδρομές προς το πρόγραμμα, το νευρωνικό δίκτυο και το βίντεο.

10.4.7.1 yolo-v2-tiny-vehicle-detection-0001 – 01-video02_traffic_side.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v2-tiny-vehicle-detection-
0001/FP16/yolo-v2-tiny-vehicle-detection-0001.xml
    --architecture_type yolo
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```



10.4.7.2 yolo-v2-tiny-vehicle-detection-0001 – 02-video08_traffic_front.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v2-tiny-vehicle-detection-
0001/FP16/yolo-v2-tiny-vehicle-detection-0001.xml
    --architecture_type yolo
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```

10.4.7.3 yolo-v2-tiny-vehicle-detection-0001 – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v2-tiny-vehicle-detection-
0001/FP16/yolo-v2-tiny-vehicle-detection-0001.xml
    --architecture_type yolo
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```

10.4.7.4 yolo-v2-tiny-vehicle-detection-0001 – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v2-tiny-vehicle-detection-
0001/FP16/yolo-v2-tiny-vehicle-detection-0001.xml
    --architecture_type yolo
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```

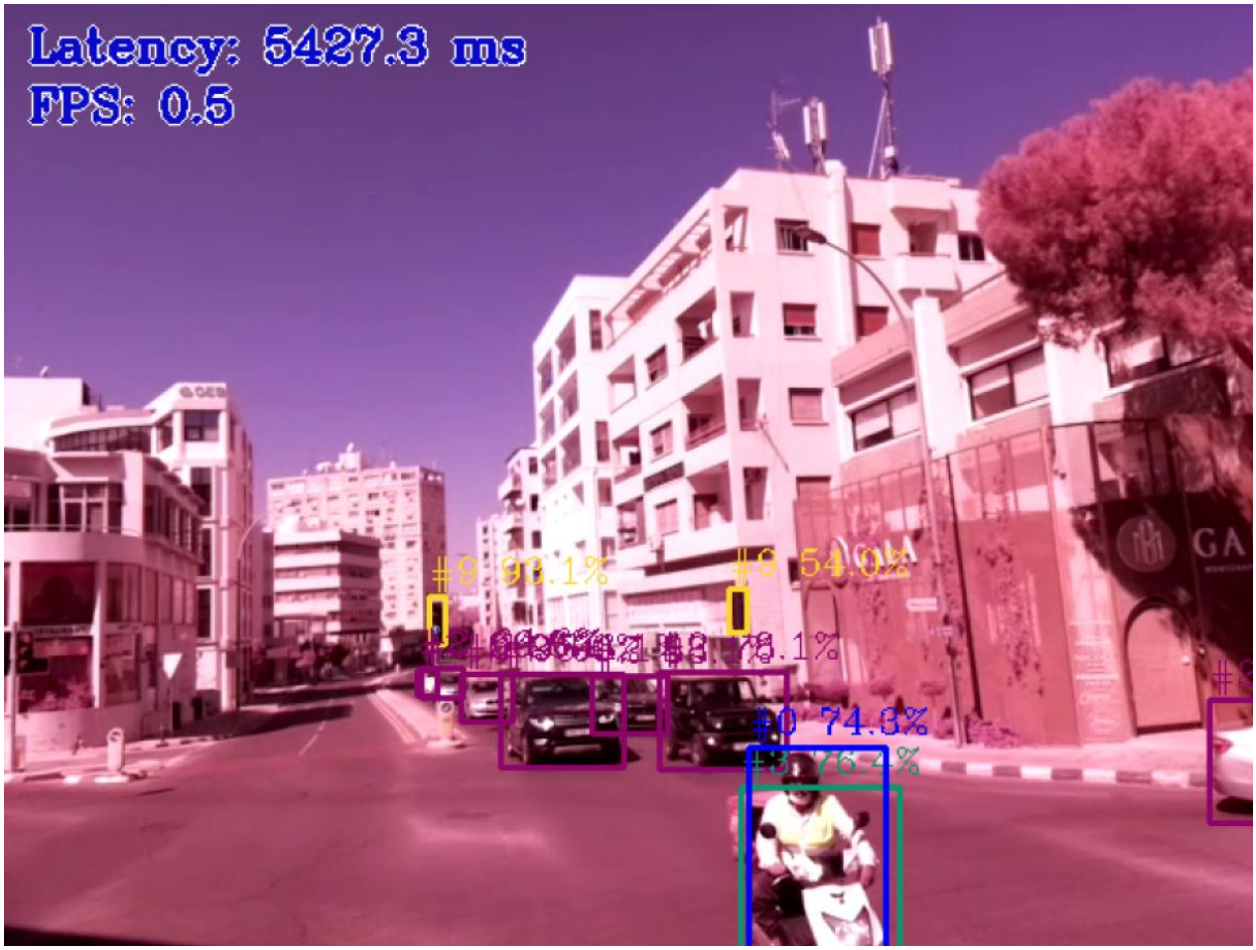
10.4.8 yolo-v4-tf

10.4.8.1 yolo-v4-tf – 01-video02_traffic_side.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v4-tf/FP16/yolo-v4-tf.xml
    --architecture_type yolov4
    --input /home/pi/Videos/01-video02_traffic_side.h264
    --device MYRIAD
    -nireq 2
```

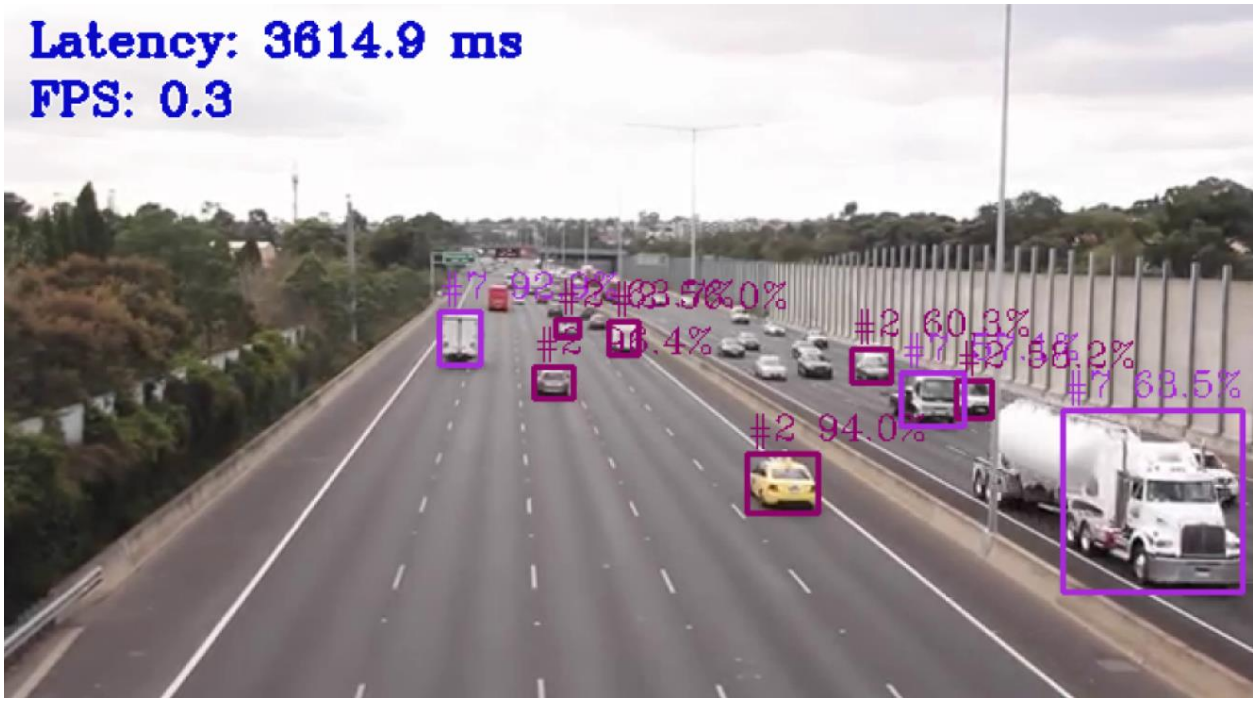
10.4.8.2 yolo-v4-tf – 02-video08_traffic_front.h264

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v4-tf/FP16/yolo-v4-tf.xml
    --architecture_type yolov4
    --input /home/pi/Videos/02-video08_traffic_front.h264
    --device MYRIAD
    -nireq 2
```



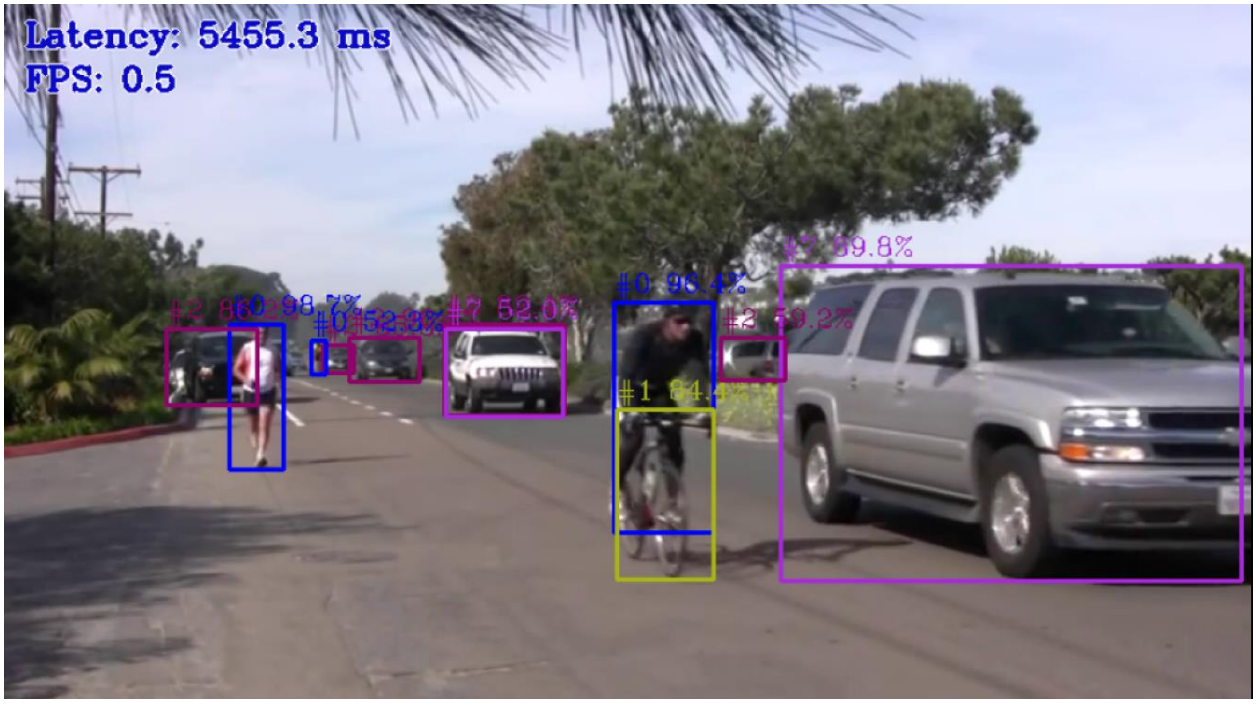
10.4.8.3 yolo-v4-tf – 03-highway_footage_short.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v4-tf/FP16/yolo-v4-tf.xml
    --architecture_type yolov4
    --input /home/pi/Videos/03-highway_footage_short.mp4
    --device MYRIAD
    -nireq 2
```



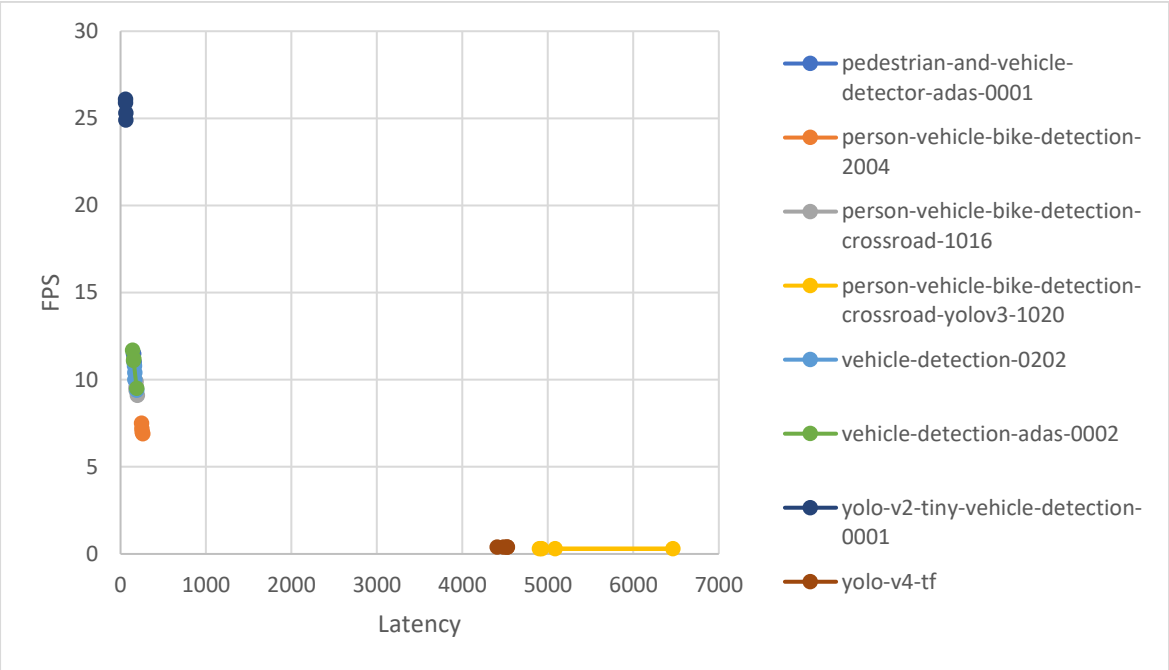
10.4.8.4 yolo-v4-tf – 04-cars_side_view_mini.mp4

```
python3 /home/pi/Documents/python/object_detection.py
    --model /home/pi/Downloads/models/yolo-v4-tf/FP16/yolo-v4-tf.xml
    --architecture_type yolov4
    --input /home/pi/Videos/04-cars_side_view_mini.mp4
    --device MYRIAD
    -nireq 2
```



11. Συμπεράσματα

Τοποθετώντας τα αποτελέσματα σε γράφημα, γίνονται κατανοητές οι διαφορές μεταξύ των νευρωνικών δικτύων στους χρόνους απόκρισης και εξαγωγής δεδομένων για την εκτέλεση στο συγκεκριμένο υλικό.



Η ακρίβεια των εξαγόμενων δεδομένων, δηλαδή το κατά πόσο το κάθε νευρωνικό δίκτυο πετυχαίνει τον εντοπισμό και την αναγνώριση των αντικειμένων σε ικανοποιητικό βαθμό, αναφέρεται στην ιστοσελίδα του εκάστοτε νευρωνικού δικτύου, στις οποίες έχουν γίνει παραπομπές προηγουμένως.

Για την επιλογή του βέλτιστου νευρωνικού δικτύου, θα πρέπει να ληφθούν υπόψιν ο χρόνος απόκρισης δεδομένων των πόρων του συστήματος, η ακρίβεια στην αναγνώριση, το αν επιλύει το πρόβλημα της παρούσας εργασίας, στην περίπτωση που δεν το επιλύει το αν επιτυγχάνει την ακρίβεια για την οποία εκπαιδεύτηκε και την δυνατότητα χρήσης του σε μελλοντική εξέλιξη.

Λαμβάνοντας υπόψιν τα αποτελέσματα με το throughput και την απόκριση των νευρωνικών δικτύων των παραπάνω δοκιμών για το σύστημα Raspberry Pi & Intel NCS2, καταλήγουμε στο εξής συμπέρασμα.

Η βέλτιστη επιλογή νευρωνικού δικτύου αποτελεί το yolo-v4-tf. Όμως, οι πόροι που απαιτούνται ξεφεύγουν από την παρούσα εργασία. Σε μελλοντική εξέλιξής της, θα μπορούσε να ληφθεί υπόψιν και να πραγματοποιηθεί με βάση τα παραπάνω.

Για τον συνδυασμό Raspberry Pi & Intel Neural Compute Stick 2, το yolo-v2-tiny-vehicle-detection-0001 προσφέρει την καλύτερη λύση από τα παραπάνω νευρωνικά δίκτυα σε βαθμό απόδοσης, αλλά αδυνατεί να εξάγει ικανοποιητικά αποτελέσματα για την περίπτωση των έξυπνων φαναριών, όπου απαιτείται μεγαλύτερη ακρίβεια για την διασφάλιση του υψηλού επιπέδου ασφάλειας.

Ρόλο στην ικανοποιητική ακρίβεια έχει και η γωνία υπό την οποία λαμβάνεται η εικόνα. Η βέλτιστη γωνία είναι κοιτώντας τα οχήματα από ευθεία μπροστά ή πίσω και από ύψος μεγαλύτερο των 4 μέτρων και χαμηλότερο των 7 μέτρων, για να καλύπτεται μία ικανοποιητικά μεγάλη περιοχή.

Τα νευρωνικά δίκτυα vehicle-detection-0202 και vehicle-detection-adas-0002 απορρίπτονται καθώς ανιχνεύουν μόνο οχήματα και όχι κατ’ ελάχιστο και ανθρώπους.

Συνεπώς, η βέλτιστη επιλογή για την παρούσα εργασία είναι τα νευρωνικά δίκτυα pedestrian-and-vehicle-detector-adas-0001 και person-vehicle-bike-detection-crossroad-1016, που όμως δεν αποτελούν ασφαλή λύση πέρα από τα πλαίσια της παρούσας εργασίας.

Για εκτέλεση σε πραγματικό χρόνο με ικανοποιητική ακρίβεια για την πραγματοποίηση των έξυπνων φαναριών κυκλοφορίας, απαιτούνται νευρωνικά δίκτυα ικανά να επιτυγχάνουν συνεχή βεβαιότητα άνω του 60% συνεχόμενα. Όπως προαναφέρθηκε, πιθανή εξέλιξη της διπλωματικής εργασίας θα ήταν η ανάπτυξη σε σύστημα με περισσότερους πόρους.

Η Intel πλέον παρέχει το OpenVINO Deep Learning Workbench, το οποίο αποτελεί ένα γραφικό περιβάλλον για την εισαγωγή ενός νευρωνικού δικτύου, την δοκιμή της απόδοσης και την ακρίβειας, την βελτιστοποίησή του και την προετοιμασία του για χρήση στην παραγωγή εφαρμογών. Τα περισσότερα βήματα που αναλύθηκαν κατά την εκπόνηση της εργασίας, με την χρήση του OpenVINO DL Workbench εκτελούνται ταχύτατα και απροβλημάτιστα, το οποίο βοηθάει στην μελλοντική ανάπτυξη της ιδέας σε σύστημα με περισσότερους πόρους.

12. Ορολογία

Inference = Συμπέρασμα

Inferencing = Η διαδικασία εξαγωγής συμπερασμάτων, εν προκειμένω η εξαγωγή χαρακτηριστικών από εικόνα, ανάλογα το αποτέλεσμα που θέλουμε να επιτύχουμε

Cloud = Σύννεφο, δηλαδή η υπολογιστική και άλλη σχετική ισχύς, η οποία βρίσκεται σε κάποιο απομακρυσμένο κέντρο δεδομένων (datacenter), στην οποία εκτελούμε δύσκολες και πολύπλοκες διαδικασίες

Datacenter = Συνήθως, μία μεγάλη ομάδα διασυνδεδεμένων υπολογιστών που παρέχουν μεγαλύτερη επεξεργαστική ισχύ, αποθηκευτικό χώρο και ταχύτητα

CPU = Central Processing Unit – Κεντρική μονάδα επεξεργασίας, το υλικό που μπορεί να εκτελέσει πράξεις γενικού σκοπού

GPU = Graphics Processing Unit – Μονάδα επεξεργασίας γραφικών, το υλικό που εξειδικεύεται στην εκτέλεση και αποτύπωση γραφικών, τα οποία αποτελούνται από πίνακες

FPGA = Field Programmable Gate Array – Μονάδα συστοιχιών λογικών πυλών με δυνατότητα προγραμματισμού του υλικού μετά την παραγωγή του

Framework = Πλαίσιο – Η βασική δομή πάνω στην οποία αναπτύσσονται οι εφαρμογές

On the edge (ote) = Ο τελικός χρήστης / συσκευή και ότι βρίσκεται φυσικά συνδεδεμένο σε αυτόν και χρησιμοποιείται για να επιτελέσει απαραίτητες διαδικασίες, απαραίτητες για την λειτουργία του συστήματος

OpenVINO = Open Visual Inferencing and Neural Network Optimization – Ελεύθερης χρήσης βελτιστοποίηση εξαγωγής συμπερασμάτων οπτικού περιεχομένου και νευρωνικών δικτύων – Το πρόγραμμα της INTEL για την εκτέλεση του inferencing πάνω στις συμβατές πλατφόρμες

Detection = Η αναγνώριση αφορά την διαδικασία της εύρεσης ενός (ή και περισσότερων) ζητούμενου αντικειμένου από μία εικόνα, σύμφωνα με τα χαρακτηριστικά που εξάγονται από αυτήν

Classification = Η ταξινόμηση αφορά την διαδικασία του διαχωρισμού ενός αντικειμένου σε κάποια ομάδα, σύμφωνα με τις ζητούμενες από την εκάστοτε εφαρμογή

Segmentation = Η κατάτμηση μίας εικόνας με σκοπό την εξαγωγή χαρακτηριστικών σύμφωνα με τα ζητούμενα της εκάστοτε εφαρμογής

Cross-platform = Η άνευ περιορισμού χρήση σε διάφορες πλατφόρμες είτε υλικού, είτε λογισμικού

Machine Learning = Μηχανική μάθηση, δηλαδή η διαδικασία της δημιουργίας αλγορίθμων, οι οποίοι θα βελτιώνονται αυτόματα χάρη στην εμπειρία των συνεχών εκτελέσεων και ευρέσεων των σφαλμάτων, αλλά και χάρη στην συλλογή δεδομένων.

API (Application Programming Interface) = Η Διεπαφή Προγραμματισμού Εφαρμογών είναι τα κομμάτια ενός προγράμματος ή μίας εφαρμογής που επιτρέπουν την ένωσή της με άλλες εφαρμογές ή προγράμματα, μέσω συγκεκριμένων εντολών ή διαδικασιών.

Button bounce = Αναπήδηση κουμπιού, δηλαδή η δημιουργία εσφαλμένης εντολής λόγω της ύπαρξης ρεύματος ανάμεσα στους 2 ακροδέκτες του κουμπιού, είτε χωρίς την ζητούμενη ενέργεια του πατήματος, είτε μετά από πάτημα και λόγω της δημιουργίας τόξων λόγω των μικρών αποστάσεων, η οποία καταχωρείται εσφαλμένα ως εντολή. Η διαδικασία αποτροπής του συγκεκριμένου προβλήματος λέγεται debouncing.

(G)Flops = (Giga) FLoating point Operations Per Second – Πράξεις κινητής υποδιαστολής ανά δευτερόλεπτο, μονάδα μέτρησης απόδοσης για εφαρμογές που απαιτούν τέτοιες πράξεις. Ως μέτρο σύγκρισης, ο υπερυπολογιστής Fugaku έχει μέγιστη απόδοση 1,07 ExaFlops (Exa = 10^18), μία μέση οικιακή CPU 225 GFlops (Giga = 10^9) σε FP32 (4,75 εκατομμύρια φορές γρηγορότερος) [73]

params = Οι εγγενείς παράμετροι ενός νευρωνικού δικτύου, οι οποίες αλλάζουν αυτόματα από το ίδιο το δίκτυο, όπως τα βάρη (weights) και τα δυναμικά πολώσεως (biases) των νευρώνων

IoU = Intersection over Union – Η διατομή της ένωσης είναι ο όρος που χρησιμοποιείται στην ανίχνευση αντικειμένων για να δηλώσει την αναλογία ανάμεσα στον χώρο που καλύπτεται και στον κοινό χώρο των πλαισίων οριοθέτησης σε μία εικόνα. Σε γενικές γραμμές, οποιαδήποτε αναλογία μεγαλύτερη του 0,5 θεωρείται καλή. [74]

COCO = Common Objects in COntext – Αποτελεί μία μεγάλη βάση δεδομένων που περιλαμβάνει εικόνες στις οποίες υπάρχουν επεξηγήσεις, τίτλοι, κατηγορίες, λεζάντες και βασικά σημεία (ανάλογα τον τύπο του περιεχομένου) για χρήση στην εκπαίδευση νευρωνικών δικτύων ανίχνευσης, κατάτμησης και αναγνώρισης αντικειμένων.

Colour system = Σύστημα χρωμάτων, ή χρωματικό μοντέλο, είναι η περιγραφή των χρωμάτων με μαθηματική μορφή. Υπάρχουν διάφορα συστήματα, όπως το RGB (Red Green Blue), το HSV (Hue Saturation Value) και το HSL (Hue Saturation Luminosity). Χρησιμοποιούνται για την αναπαράσταση των φυσικών χρωμάτων σε ψηφιακή μορφή και η αναπαράσταση αποτελείται από ένα εύρος τιμών του κάθε στοιχείου σε κάθε σύστημα. Για παράδειγμα στο RGB[0-255, 0-255, 0-255], στο HSV[0-360, 0-100, 0-100] και στο HSL[0-360, 0-100, 0-100].

13. Βιβλιογραφία

[1] IBIGROUP. [Ηλεκτρονικό]. Available: https://www.ibigroup.com/services/planning-urban-design/.

[2] SOMOS. [Ηλεκτρονικό]. Available: https://www.somos.srl/traffic-lights.

[3] Alibaba. [Ηλεκτρονικό]. Available: https://damo.alibaba.com/product/CVIE.

[4] J. Motavalli, Autoweek, [Ηλεκτρονικό]. Available: https://www.autoweek.com/news/a34498280/its-time-for-smart-traffic-lights/.

[5] NoTraffic. [Ηλεκτρονικό]. Available: https://notraffic.tech/how-it-works/.

[6] NVIDIA. [Ηλεκτρονικό]. Available: https://blogs.nvidia.com/blog/2019/10/31/notraffic-ai-startup-intersections/.

[7] Techcrunch. [Ηλεκτρονικό]. Available: https://techcrunch.com/2021/07/13/traffic-management-platform-notraffic-raises-17-5m-in-series-a-to-double-team-size/.

[8] Surtrac. [Ηλεκτρονικό]. Available: https://www.rapidflowtech.com/surtrac/what-is-surtrac.

[9] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Scalable_Urban_Traffic_Control.

[10] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/OpenCV .

[11] OpenCV. [Ηλεκτρονικό]. Available: https://opencv.org/ .

[12] opensource. [Ηλεκτρονικό]. Available: https://opensource.org/licenses/BSD-3-Clause.

[13] BoofCV. [Ηλεκτρονικό]. Available: https://boofcv.org/index.php?title=Main_Page .

[14] Apache. [Ηλεκτρονικό]. Available: https://www.apache.org/licenses/LICENSE-2.0.

[15] SimpleCV. [Ηλεκτρονικό]. Available: http://simplecv.org/.

[16] stackoverflow. [Ηλεκτρονικό]. Available: https://stackoverflow.com/questions/21215896/the-difference-between-simplecv-and-opencv.

[17] pixlab. [Ηλεκτρονικό]. Available: https://sod.pixlab.io/.

[18] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/TensorFlow.

[19] Google-TensorFlow. [Ηλεκτρονικό]. Available: https://www.tensorflow.org/.

[20] Answerrocket. [Ηλεκτρονικό]. Available: https://www.answerrocket.com/ai-libraries/.

[21] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Theano_(software).

[22] MLM. [Ηλεκτρονικό]. Available: https://machinelearningmastery.com/introduction-python-deep-learning-library-theano/.

[23] Github. [Ηλεκτρονικό]. Available: https://github.com/aesara-devs/aesara.

[24] Berkeley. [Ηλεκτρονικό]. Available: https://bair.berkeley.edu/.

[25] berkeleyvision. [Ηλεκτρονικό]. Available: https://caffe.berkeleyvision.org/.

[26] Github. [Ηλεκτρονικό]. Available: https://github.com/BVLC/caffe/blob/master/LICENSE.

[27] Keras. [Ηλεκτρονικό]. Available: https://keras.io/.

[28] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Keras.

[29] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/PyTorch.

[30] Quora. [Ηλεκτρονικό]. Available: https://www.quora.com/What-is-PyTorch-used-for.

[31] pytorch. [Ηλεκτρονικό]. Available: https://pytorch.org/.

[32] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/BSD_licenses#3-clause.

[33] coralAI. [Ηλεκτρονικό]. Available: ttps://coral.ai/products/accelerator/.

[34] Intel. [Ηλεκτρονικό]. Available: https://software.intel.com/en-us/neural-compute-stick.

[35] arrow. [Ηλεκτρονικό]. Available: https://www.arrow.com/en/research-and-events/articles/google-coral-edge-tpu-accelerator-vs-intel-neural-compute-stick-2.

[36] tomshardware. [Ηλεκτρονικό]. Available: https://www.tomshardware.com/news/jetson-nano-features-price,38856.html.

[37] marketscreener. [Ηλεκτρονικό]. Available: https://www.marketscreener.com/NVIDIA-CORPORATION-57355629/news/NVIDIA-What-s-the-Difference-Between-Jetson-Nano-Raspberry-Pi-Neural-Compute-Stick-and-Edge-TPU-28804697/.

[38] makerpro. [Ηλεκτρονικό]. Available: https://maker.pro/nvidia-jetson/tutorial/introduction-to-cuda-programming-with-jetson-nano.

[39] seeedstudio. [Ηλεκτρονικό]. Available: https://www.seeedstudio.com/ROCK-PI-N10-Model-A-RK3399Pro-4GB-LPDDR3-16GB-eMMC-p-4379.html.

[40] radxa. [Ηλεκτρονικό]. Available: https://wiki.radxa.com/RockpiN10.

[41] intel. [Ηλεκτρονικό]. Available: https://www.intel.com/content/www/us/en/developer/tools/openvino-toolkit/overview.html.

[42] intel. [Ηλεκτρονικό]. Available: https://docs.openvino.ai/latest/openvino_docs_install_guides_installing_openvino_windows.html.

[43] tensorflow. [Ηλεκτρονικό]. Available: https://www.tensorflow.org/install/pip#system-requirements.

[44] intel. [Ηλεκτρονικό]. Available: https://docs.openvino.ai/latest/openvino_docs_MO_DG_prepare_model_Model_Optimization_Techniques.html.

[45] intel. [Ηλεκτρονικό]. Available: https://docs.openvino.ai/latest/openvino_inference_engine_tools_benchmark_tool_README.html.

[46] intel. [Ηλεκτρονικό]. Available: https://docs.openvino.ai/latest/openvino_inference_engine_samples_benchmark_app_README.html#doxid-openvino-inference-engine-samples-benchmark-app-r-e-a-d-m-e .

[47] intel. [Ηλεκτρονικό]. Available: https://docs.openvino.ai/latest/openvino_inference_engine_samples_benchmark_app_README.html.

[48] Github. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/blob/master/models/intel/device_support.md.

[49] dexterindustries. [Ηλεκτρονικό]. Available: https://www.dexterindustries.com/howto/run-a-program-on-your-raspberry-pi-at-startup/.

[50] intel. [Ηλεκτρονικό]. Available: https://www.intel.com/content/www/us/en/support/articles/000055345/boards-and-kits.html.

[51] makeuseof. [Ηλεκτρονικό]. Available: https://www.makeuseof.com/tag/raspberry-pi-update-raspbian-os/.

[52] S. J. N. &. H. Farid. [Ηλεκτρονικό]. Available: https://www.pnas.org/doi/10.1073/pnas.2120481119.

[53] J. Christoffersen. [Ηλεκτρονικό]. Available: https://www.allaboutcircuits.com/technical-articles/switch-bounce-how-to-deal-with-it/.

[54] T. Instruments. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=e1-kc04jSE4.

[55] intel/pedestrian-and-vehicle-detector-adas-0001. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/blob/9f0cb05b4f5338d5defa3e6ce95ea75fb1baed2b/models/intel/pedestrian-and-vehicle-detector-adas-0001.

[56] intel/person-vehicle-bike-detection-2004. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/aaa38a4fde7a1eff4120fff7e77ac1af92e1771d/models/intel/person-vehicle-bike-detection-2004.

[57] intel/person-vehicle-bike-detection-crossroad-1016. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/aaa38a4fde7a1eff4120fff7e77ac1af92e1771d/models/intel/person-vehicle-bike-detection-crossroad-1016.

[58] intel/person-vehicle-bike-detection-crossroad-yolov3-1020. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/aaa38a4fde7a1eff4120fff7e77ac1af92e1771d/models/intel/person-vehicle-bike-detection-crossroad-yolov3-1020.

[59] intel/vehicle-detection-0202. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/intel/vehicle-detection-0202.

[60] intel/vehicle-detection-adas-0002. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/intel/vehicle-detection-adas-0002.

[61] intel/yolo-v2-tiny-vehicle-detection-0001. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/aaa38a4fde7a1eff4120fff7e77ac1af92e1771d/models/intel/yolo-v2-tiny-vehicle-detection-0001.

[62] GOOGLEpublic/yolo-v4-tf. [Ηλεκτρονικό]. Available: https://github.com/openvinotoolkit/open_model_zoo/tree/master/models/public/yolo-v4-tf.

[63] python. [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/colorsys.html.

[64] python. [Ηλεκτρονικό]. Available: https://docs.python.org/3/howto/logging.html.

[65] python. [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/random.html.

[66] python. [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/sys.html.

[67] python. [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/argparse.html.

[68] python. [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/pathlib.html.

[69] python. [Ηλεκτρονικό]. Available: https://docs.python.org/3/library/time.html.

[70] pypi. [Ηλεκτρονικό]. Available: https://pypi.org/project/opencv-python/.

[71] numpy. [Ηλεκτρονικό]. Available: https://numpy.org/doc/stable/user/absolute_beginners.html.

[72] intelPythonIE. [Ηλεκτρονικό]. Available: https://docs.openvino.ai/latest/api/ie_python_api/api.html.

[73] Wikipedia. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/TOP500#TOP_500.

[74] pyimagesearch. [Ηλεκτρονικό]. Available: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.