ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανίχνευση Πτώσεων μέσω TinyML σε Συσκευές IoT

Διπλωματική Εργασία

του

Παιδαράκη Σταύρου

Θεσσαλονίκη, Οκτώβριος 2023

ΑΝΙΧΝΕΥΣΗ ΠΤΩΣΕΩΝ ΜΕΣΩ TINYML ΣΕ ΣΥΣΚΕΥΕΣ IOT

Παιδαράκης Σταύρος

Πτυχίο Μαθηματικών, ΑΠΘ, 2021

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Παπαδημητρίου Παναγιώτης

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ηη/μμ/εεεε

Παπαδημητρίου Παναγιώτης	Μαμάτας Ελευθέριος	Ψάννης Κωνσταντίνος
.....	.....	.....

Παιδαράκης Σταύρος

.....

Περίληψη

Οι πτώσεις, που συχνά οδηγούν σε σοβαρούς τραυματισμούς και επιπλοκές, αποτελούν σημαντική απειλή για την ευημερία των ατόμων, ιδιαίτερα του ηλικιωμένου πληθυσμού. Αυτό έδωσε κίνητρο στην εξερεύνηση προηγμένων τεχνολογιών για τον μετριασμό αυτών των κινδύνων. Μια πολλά υποσχόμενη οδός είναι η ενσωμάτωση της μηχανικής μάθησης και του Διαδικτύου των Πραγμάτων, η οποία επιτρέπει την παρακολούθηση σε πραγματικό χρόνο και την έγκαιρη παρέμβαση. Αυτή η διπλωματική εμβαθύνει στον κλάδο της ανίχνευσης πτώσης χρησιμοποιώντας οπτικά στοιχεία, αξιοποιώντας το PoseNet, ένα ελαφρύ μοντέλο εκτίμησης σωματικής στάσης τελευταίας τεχνολογίας. Η καινοτομία έγκειται στη χρήση του TinyML, ενός αναπτυσσόμενου κλάδου που εστιάζει στην ανάπτυξη μοντέλων μηχανικής εκμάθησης σε συσκευές με περιορισμένους πόρους. Το προτεινόμενο σύστημα, εκπαιδευμένο και δοκιμασμένο σε ένα δημόσιο σύνολο δεδομένων ανίχνευσης πτώσεων, παρουσιάζει ακρίβεια έως και 90,5% μετά τον κβαντισμό με μέγεθος έως 708 KB. Επιπλέον, η μελέτη διερευνά τον αντίκτυπο της εκπαίδευσης των μοντέλων σε τρία διαφορετικά μεγέθη εικόνας, αποκαλύπτοντας πληροφορίες για την αντιστάθμιση μεταξύ ακρίβειας και υπολογιστικής απόδοσης για ανίχνευση πτώσης σε συσκευές με περιορισμένους πόρους.

Λέξεις Κλειδιά: Ανίχνευση πτώσεων, βαθιά μάθηση, TinyMl, Διαδίκτυο των Πραγμάτων, εκτίμηση σωματικής στάσης

Abstract

Falls, often resulting in severe injuries and complications, pose a significant threat to the well-being of individuals, particularly the elderly population. This has motivated the exploration of advanced technologies to mitigate these risks. One promising avenue is the integration of machine learning and Internet of Things, which enables real-time monitoring and early intervention. This thesis delves into the realm of fall detection using visual cues, leveraging PoseNet, a lightweight state-of-the-art pose estimation model. The innovation lies in the utilization of TinyML, a burgeoning field that focuses on deploying machine learning models on resource-constrained devices. The proposed system, trained and tested on a public fall detection dataset exhibits up to 90.5% accuracy after quantization with size as low as 708 KB. Moreover, the study explores the impact of training the models on three different image sizes, revealing insights into the trade-off between accuracy and computational efficiency for fall detection on resource constrained devices.

Keywords: Fall detection, deep learning, TinyML, Internet of Things, pose estimation

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

1 Εισαγωγή

1.1 Περιγραφή του Προβλήματος

Οι πτώσεις είναι ένα από τα πιο καίρια προβλήματα που αντιμετωπίζουν άτομα με ειδικές ανάγκες όπως ηλικιωμένοι, άτομα με αναπηρία, άτομα με Πάρκινσον, υπέρβαροι και παχύσαρκοι, και λοιπά. Αυτό το ζήτημα έχει επιδεινωθεί από τις δημογραφικές τάσεις, με πολλές χώρες να παρουσιάζουν γήρανση του πληθυσμού, οδηγώντας σε υψηλότερη συχνότητα πτώσεων μεταξύ των ηλικιωμένων. Την 1η Ιανουαρίου 2021 τα άτομα ηλικίας 65 ετών και άνω αντιπροσώπευαν το 20.8% του πληθυσμού της Ευρωπαϊκής Ένωσης. Έως το 2050 το ποσοστό αυτό αναμένεται να είναι περίπου 30%, 10% αύξηση περίπου σε σύγκριση με σήμερα [38]. Παγκοσμίως, το μερίδιο του πληθυσμού ηλικίας 60 ετών και άνω θα αυξηθεί από 1 δισεκατομμύριο το 2020 σε 1.4 δισεκατομμύρια έως το 2030. Μέχρι το 2050, ο παγκόσμιος πληθυσμός των ατόμων ηλικίας 60 ετών και άνω θα διπλασιαστεί (2.1 δισεκατομμύρια), ενώ ο αριθμός των ατόμων ηλικίας 80 ετών και άνω αναμένεται να τριπλασιαστεί μεταξύ 2020 και 2050 για να φτάσει τα 426 εκατομμύρια [40]. Η ανίχνευση πτώσης έχει γίνει συνεπώς ένα ενεργό και ουσιαστικό ερευνητικό θέμα στον τομέα της υγείας, ιδιαίτερα στην υγειονομική περίθαλψη για τους ηλικιωμένους.

Οι πτώσεις μεταξύ των ηλικιωμένων αποτελούν μείζονα ανησυχία για την υγεία παγκοσμίως λόγω και των σοβαρών συνεπειών που συνεπάγονται, όπως υψηλότερα ποσοστά νοσηρότητας και θνησιμότητας, χαμηλότερες λειτουργικές ικανότητες και το ενδεχόμενο μακροχρόνιας εισαγωγής σε μονάδες φροντίδας [29]. Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας, οι πτώσεις είναι η δεύτερη κυρία αιτία θανάτων που προκλήθηκαν από ακούσιο τραυματισμό, με 684 χιλιάδες περίπου θανάτους τον χρόνο, ενώ ο αριθμός των πτώσεων που είναι αρκετά σοβαρός ώστε να απαιτούν ιατρική φροντίδα ξεπερνάει τα 37 εκατομμύρια κάθε χρόνο [39]. Πέρα από τους τραυματισμούς που μπορεί να προκαλέσει μια πτώση στους ηλικιωμένους, οι συγγενείς επίσης επιβαρύνονται ψυχολογικά και πιέζονται οικονομικά [7]. Αντιμέτωποι με αυτήν την κατάσταση, είναι ιδιαίτερα κρίσιμη η γρήγορη και αποτελεσματική ανίχνευση των

πτώσεων των ηλικιωμένων για να δοθούν οι απαραίτητες ιατρικές βοήθειες.

Η σημασία της ανίχνευσης πτώσης είναι αναμφισβήτητη. Ωστόσο, οι παραδοσιακές μέθοδοι έχουν συναντήσει πολλά εμπόδια. Αυτές οι προσεγγίσεις βασίζονται συχνά σε φορητές συσκευές ή συστήματα παρακολούθησης βίντεο, το καθένα με τους περιορισμούς του. Οι φορητές συσκευές, αν και εύκολα προσιτές, ενδέχεται να υποφέρουν από ζητήματα άνεσης. Ακόμη, οι χρήστες, ιδιαίτερα οι ηλικιωμένοι, μπορεί να ξεχάσουν να τις φορέσουν, καθιστώντας τις συσκευές αναποτελεσματικές. Η βιντεοπαρακολούθηση, από την άλλη πλευρά, εγείρει ανησυχίες για την ιδιωτικότητα των χρηστών, ενώ η αποτελεσματικότητά τους δεν είναι εγγυημένη, ειδικά σε καταστάσεις όπου υπάρχει σκοτάδι ή όπου δεν είναι πλήρως ορατοί οι χρήστες.

Επιπρόσθετα, η ανίχνευση πτώσης με βάση την όραση παρουσιάζει μια πρόκληση για την επίτευξη ισορροπίας μεταξύ της μεγιστοποίησης του ρυθμού ανίχνευσης πτώσης και της ελαχιστοποίησης της υπολογιστικής πολυπλοκότητας. Πολλές τεχνικές που βασίζονται στην όραση απαιτούν συνήθως σημαντική επεξεργαστική ισχύ για ανάλυση βίντεο σε πραγματικό χρόνο, μια ανάγκη που μπορεί να μην είναι εφικτή για πρακτική αξιοποίηση σε πραγματικό χρόνο. Αναγνωρίζοντας τη χρονικά ευαίσθητη φύση των περιστατικών πτώσης, η επίτευξη ανίχνευσης σε πραγματικό χρόνο είναι ζωτικής σημασίας, καθώς οι καθυστερήσεις μπορεί να έχουν τρομερές συνέπειες, ειδικά όταν η άμεση ιατρική βοήθεια είναι επιτακτική. Οι παραδοσιακές μέθοδοι συχνά δυσκολεύονται με την ανίχνευση σε πραγματικό χρόνο λόγω της καθυστέρησης στην επεξεργασία και την επικοινωνία δεδομένων. Για να εξασφαλιστεί η έγκαιρη παρέμβαση, τα σύγχρονα συστήματα ανίχνευσης πτώσης πρέπει να λειτουργούν με ελάχιστη καθυστέρηση, αξιοποιώντας τεχνολογίες και αλγόριθμους αιχμής. Η γρήγορη ανίχνευση όχι μόνο ενισχύει τις προοπτικές ενός ευνοϊκού αποτελέσματος, αλλά παρέχει επίσης διαβεβαίωση τόσο σε εκείνους που είναι ευάλωτοι σε πτώσεις όσο και στους φροντιστές τους.

1.2 Κίνητρα και Στόχος

Η ανακάλυψη του TinyML (Tiny Machine Learning) αντιπροσωπεύει μια κομβική πρόοδο στον τομέα της μηχανικής μάθησης και των edge συστημάτων. Επιτρέποντας

την ανάπτυξη μοντέλων μηχανικής μάθησης σε συσκευές με εξαιρετικά περιορισμένους πόρους, το TinyML έχει ανοίξει νέες δυνατότητες για λήψη αποφάσεων σε πραγματικό χρόνο. Αυτή η εξέλιξη έχει εκτεταμένα οφέλη σε διάφορους κλάδους, όπως την βιομηχανία και βεβαίως το healthcare. Η δυνατότητα επεξεργασίας δεδομένων απευθείας στο edge εξαλείφει την ανάγκη για συνεχή μετάδοση δεδομένων σε κεντρικούς servers, μειώνοντας την καθυστέρηση. Επιπλέον, το TinyML εξουσιοδοτεί τις συσκευές να λαμβάνουν context-aware αποφάσεις αυτόνομα, βελτιώνοντας την αποτελεσματικότητα και την ανταπόκριση. Αυτός ο μετασχηματισμός είναι έτοιμος να επαναπροσδιορίσει τον τρόπο με τον οποίο αλληλεπιδρούμε με την τεχνολογία, εγκαινιάζοντας μια νέα εποχή έξυπνων, ενσωματωμένων συστημάτων που μπορούν να εκτελέσουν πολύπλοκες εργασίες με ελάχιστους πόρους και χαμηλό κόστος.

Στόχος, λοιπόν αυτής της εργασίας είναι η αξιοποίηση του TinyML για την δημιουργία visual-based αλγορίθμων ανίχνευσης πτώσης που θα λειτουργούν σε πραγματικό χρόνο σε συσκευές περιορισμένων υπολογιστικών πόρων. Σκοπός είναι η εκπαίδευση ικανών μοντέλων και η μείωση των μεγεθών τους, ώστε να είναι έτοιμα για deployment σε ενσωματωμένες συσκευές.

1.3 Διάρθρωση της μελέτης

Τα κεφάλαια που ακολουθούν συνοψίζονται ως:

- Κεφάλαιο 2: Γίνεται μια ανασκόπηση ορισμένων εργασιών που ανήκουν στον τομέα της ανίχνευσης πτώσης.
- Κεφάλαιο 3: Αναλύει το θεωρητικό υπόβαθρο, έννοιες που είναι απαραίτητες για την κατανόηση της μεθόδου που προτείνει η συγκεκριμένη διπλωματική.
- Κεφάλαιο 4: Περιγράφει τα βήματα της μεθοδολογίας που υλοποιήθηκε.
- Κεφάλαιο 5: Αξιολογούνται τα μοντέλα και οι τεχνικές βελτιστοποίησης που εφαρμόστηκαν και παρατίθενται τα αποτελέσματα των πειραμάτων που διεξήχθησαν.
- Κεφάλαιο 6: Συνοψίζει τα συμπεράσματα που εξήχθησαν και αναφέρει πιθανές μελλοντικές επεκτάσεις.

2 Βιβλιογραφική Ανασκόπηση

Πληθώρα ερευνών έχει πραγματοποιηθεί τόσο στο παρελθόν [19, 23, 36], όσο και πιο πρόσφατα [1, 10, 33], στον τομέα της αυτόματης ανίχνευσης πτώσεων, που περιλαμβάνουν νέους τρόπους ανάπτυξης αλγορίθμων και νέα hardware. Σύμφωνα με τους Mubashir et al. (2013) [23], αυτά τα συστήματα μπορούν να κατηγοριοποιηθούν ως προς τον τρόπο που λαμβάνουν δεδομένα, σε συστήματα που μπορούν να φορεθούν (wearable), σε συστήματα που βασίζονται στο άμεσο περιβάλλον τους (ambient), και σε συστήματα που βασίζονται στην όραση.

Τα συστήματα που βασίζονται σε wearable devices είναι μια κοινή λύση και περιλαμβάνουν τη χρήση επιταχυνσιόμετρων, γυροσκοπίων, ή ακόμα και βαρόμετρων. Αυτοί οι αισθητήρες είναι συχνά συνημμένα στο σώμα του ατόμου, με πιο κοινές θέσεις τους καρπούς, γύρω από τη μέση, κάτω από τη μασχάλη ή πίσω από τον λοβό του αυτιού. Οι Hussein et al. (2019) [14] ανέπτυξαν αλγορίθμους μηχανικής μάθησης ανίχνευσης πτώσης και αναγνώρισης της δραστηριότητας που οδήγησε στην πτώση. Οι Jefiza et al. (2017) [15] συγκέντρωσαν δεδομένα επιταχυνσιόμετρου και γυροσκοπίου και τα προώθησαν σε ένα νευρωνικό δίκτυο. Σκοπός του ήταν η αναγνώριση διαφορετικών κινήσεων του ανθρώπου, όπως περπάτημα, ξάπλωμα, πτώση, σκύψιμο, και άλλα. Στην έρευνα των Kwolek και Kepski (2014) [20] συλλέχθηκαν δεδομένα από επιταχυνσιόμετρο και χάρτες βάθους που χρησιμοποιήθηκαν για ανίχνευση πτώσης. Η μετρημένη επιτάχυνση χρησιμοποιείται για να ανιχνεύσει εάν υπάρχει επιτάχυνση υψηλότερη από ένα κατώφλι, ενώ ο αλγόριθμος που διαθέτει εξάγει χαρακτηριστικά ατόμου για να κατηγοριοποιήσει την δράση ως πτώση ή όχι. Στην εργασία των Shazad et al. (2019) [32] σχεδιάστηκε ένα διεισδυτικό σύστημα σε smartphone, στο οποίο συλλέγονται σήματα επιταχυνσιόμετρου που είναι ενσωματωμένο σε smartphone και ο προτεινόμενος αλγόριθμος δύο βημάτων ανιχνεύει πτώσεις.

Τα πλεονεκτήματα της μεθόδου ανίχνευσης πτώσης που βασίζεται σε wearable αισθητήρες είναι: φορητότητα, εύκολη εφαρμογή, εκτέλεση σε πραγματικό χρόνο, λίγα ζητήματα απορρήτου και ικανοποιητική ακρίβεια. Ωστόσο, αυτή η μέθοδος έχει επίσης και αδυναμίες. Η πιο προφανής είναι ότι οι άνθρωποι πρέπει να φορούν την αντίστοιχη

συσκευή στο σώμα τους, κάτι που που μπορεί να προκαλέσει δυσφορία στον χρήστη. Επίσης αυτή η μέθοδος είναι ευάλωτη σε εξωτερικούς παράγοντες, όπως πίεση και κραδασμούς, και συνεπώς η σταθερότητα δεν είναι εγγυημένη.

Οι συσκευές περιβάλλοντος (ambient devices) χρησιμοποιούν εξωτερικούς αισθητήρες για τη μέτρηση και την εξέταση του περιβάλλοντος ενός ατόμου που επιβλέπεται. Τα συνήθη χαρακτηριστικά για την ανίχνευση πτώσης είναι η πίεση, οι κραδασμοί, ο ήχος, η υπέρυθρη ακτινοβολία. Οι Principi et al. (2015) [26] παρουσίασαν μια αρχιτεκτονική που ενσωματώνει σήματα ήχου σε περιπτώσεις έκτακτης ανάγκης. Το σύστημα ενημερώνει τους caregivers για πιθανά μη φυσιολογικά ακουστικά συμβάντα και παρέχει την δυνατότητα λειτουργίας του συστήματος με φωνητικές εντολές. Στην εργασία των Clemente et al. (2020) [5] χρησιμοποιείται η δόνηση που παράγεται από μια πτώση στο δάπεδο για την ανίχνευση πτώσης ηλικιωμένων. Το σύστημα, μάλιστα μπορεί να αναγνωρίσει την ταυτότητα του ατόμου που έπεσε από την δόνηση που δημιουργούν τα βήματα. Οι Jeon et al. (2017) [16] χρησιμοποίησαν μια διάταξη τριβοηλεκτρικής νανογεννήτριας (triboelectric nanogenerator - TENG) με την οποία προσμετράται πίεση για ανίχνευση πτώση με βάση το περιβάλλον. Το 2015 οι Khan et al. [18] πρότειναν ένα ακουστικό σύστημα ανίχνευσης πτώσης. Αρχικά αφαιρείται ο θόρυβος που προκαλείται στο βάθος από τις μετρήσεις ήχου που συλλέγονται. Στην συνέχεια εξάγονται χαρακτηριστικά από τα σήματα και τέλος, ένας ταξινομητής *k*-πλησιέστερων-γειτόνων (k-nearest-neighbors - KNN) συμπεραίνει αν το άτομο περπατάει ή πέφτει.

Το μεγαλύτερο πλεονέκτημα των συσκευών περιβάλλοντος σε σχέση με τις wearable συσκευές, είναι ότι δεν παρεμβαίνουν στην ζωή των ανθρώπων. Συμπληρωματικά, η ελάχιστη αλληλεπίδραση με τους ανθρώπους, σπανίως οδηγεί σε ζητήματα ασφάλειας και απορρήτου. Παρ'όλα αυτά, αυτές οι μέθοδοι έχουν χαμηλό εύρος ανίχνευσης και συνεχίζουν να επηρεάζονται από το εξωτερικό περιβάλλον παρά τις όποιες προφυλάξεις παρθούν.

Τα συστήματα που βασίζονται στην όραση (vision-based) επικεντρώνονται αποκλειστικά στα frames από τις ροές βίντεο για να εντοπίσουν πτώσεις. Αυτά τα συστήματα χρησιμοποιούν συνήθως κάμερες RGB ή βάθους (depth) μαζί με

αλγορίθμους επεξεργασίας εικόνας που εξάγουν χαρακτηριστικά όπως οριοθέτηση πλαισίων ή σιλουέτες για τη διευκόλυνση του εντοπισμού πτώσεων. Οι Sase at al. (2018) [31] εισήγαγαν μια μέθοδο για ανίχνευση πτώσεων χρησιμοποιώντας depth βίντεο. Η μέθοδός τους εντοπίζει πτώσεις συγκρίνοντας την ορισμένες περιοχές ενδιαφέροντος (regions of interest - ROI) με κάποια προκαθορισμένα όρια. Όμως εμφανίζει την αδυναμία του ότι αδυνατεί να εντοπίσει ανθρώπους όταν αυτοι είναι αδρανείς. Οι Keskes και Noumeir (2021) [17] πρότειναν έναν αλγόριθμο συνελικτικών δικτύων χωροχρονικού γράφου (Spatial Temporal Graph Convolutional Network - ST-GCN) για την αναγνώριση ενεργειών. Τα αποτελέσματα του αλγορίθμου εμφανίζουν αξιόπιστη απόδοση για την ανίχνευση πτώσης. Οι Miguel et al. (2017) [22] πρότειναν έναν ανιχνευτή πτώσης χαμηλού κόστους για έξυπνα σπίτια. Αυτός ο ανιχνευτής χρησιμοποιεί τους αλγόριθμους αφαίρεσης φόντου (background subtraction), φίλτρα Kalman, και οπτική ροή (optical flow) ως είσοδο σε έναν KNN αλγόριθμο, ο οποίος παράγει αποτελέσματα ανίχνευσης υψηλής ακρίβειας. Είναι μια συσκευή χαμηλού κόστους που δεν φοριέται αλλά λειτουργεί αποτελεσματικά μόνο την ημέρα. Στην εργασία των Foroughi et al. (2008) [9] εντοπίζονται πολυάριθμες συμβατικές εφαρμογές γηριατρικής παρακολούθησης και χρησιμοποιούνται χαρακτηριστικά της σιλουέτας του σώματος για οικιακή επιτήρηση. Η πειραματική τους επιτυχία υποδεικνύει μεγάλες δυνατότητες αναγνώρισης διαφόρων τύπων πτώσεων, συμπεριλαμβανομένων των πλάγιων, των μπροστινών και των προς τα πίσω πτώσεων. Οι Liu et al. (2010) [21] παρουσίασαν ένα σύστημα με ταξινομητή τον αλγόριθμο KNN. Οι παράμετροι ενός bounding box της ανθρώπινης σιλουέτας λειτουργούν ως είσοδοι για τον classifier. Ομοίως, στην έρευνα των Zerrouki και Houacine (2017) [35], τα χαρακτηριστικά της ανθρώπινης σιλουέτας που εντοπίζεται με διάφορες μεθόδους προωθούνται σε ένα hidden Markov μοντέλο για να κατηγοριοποιηθούν ακολουθίες βίντεο σε συμβάντα πτώσης ή μη πτώσης. Δυστυχώς, και αυτή η μέθοδος δεν λειτουργεί αποδοτικά στο σκοτάδι. Τέλος οι Ramirez et al. [28] πρότειναν μια προσέγγιση για αναγνώριση πτώσης και δραστηριοτήτων. Το σύστημα επιστρατεύει έναν αλγόριθμο εκτίμησης στάσης σώματος, του οποίου τα αποτελέσματα στην συνέχεια προωθούνται σε κλασικούς αλγορίθμους μηχανικής μάθησης για ταξινόμηση.

Προφανώς, και οι μέθοδοι ανίχνευσης πτώσης που βασίζονται στην όραση, παρουσιάζουν μειονεκτήματα. Ένα από αυτά είναι ο κίνδυνος να εμποδιστεί η ορατότητα του ατόμου που επιβλέπεται από αντικείμενα και έπιπλα. Επίσης, όπως αναφέραμε προηγουμένως, οι μεθοδοι ανίχνευσης πτώσης που βασίζονται σε RGB κάμερες, αδυνατούν να λειτουργήσουν στο σκοτάδι. Ωστόσο, το μεγαλύτερο ελάττωμα είναι το κόστος χρήσης των visual-based μεθόδων. Το μοντέλα που χρησιμοποιούνται για την ανίχνευση πτώσης είναι βαριά υπολογιστικά και απαιτούν αρκετούς πόρους. Μια διαδεδομένη λύση είναι η χρήση του cloud. Έτσι όμως, παρουσιάζονται νέες προκλήσεις. Η απουσία υψηλού bandwidth δικτύου σε συνδυασμό με τον χρόνο που χρειάζεται για την αποστολή των δεδομένων και την απόφαση στο cloud είναι σημαντικό πρόβλημα σε εργασίες ιατρικής παρακολούθησης, όπως η ανίχνευση πτώσης. Ακόμη, η χρήση του cloud θεσπίζει ερωτήματα για την ασφάλεια και το απόρρητο των δεδομένων.

Απάντηση σε αυτά τα ερωτήματα δίνει η ιδέα του TinyML. Το TinyML αναφέρεται σε τεχνικές και εργαλεία που διευκολύνουν την εκτέλεση των μοντέλων μηχανικής μάθησης σε πολύ μικρές και περιορισμένων πόρων συσκευές, όπως αισθητήρες, μικροελεγκτές και άλλες πλατφόρμες του διαδικτύου των πραγμάτων. Ο στόχος του TinyML είναι να φέρει τη δύναμη του machine learning σε αυτές τις συσκευές, επιτρέποντας τους αλγορίθμους να λειτουργούν τοπικά και σε πραγματικό χρόνο. Το TinyML δίνει λύσεις σε όλα τα ελαττώματα της χρήσης του cloud για visual fall detection που αναφέρθηκαν [6].

3 Θεωρητικό Υπόβαθρο

Η ενότητα αυτή περιλαμβάνει το βασικό θεωρητικό υπόβαθρο που χρησιμοποιήθηκε για την ανάπτυξη μοντέλων ανίχνευσης πτώσης. Αρχικά, γίνεται μια ανασκόπηση των τύπων της μηχανικής μάθησης. Στην συνέχεια αναλύονται τα γνωρίσματα των τεχνητών νευρικών δικτύων και των συνελικτικών νευρωνικών δικτύων, και τέλος περιγράφονται οι λειτουργίες και τα χαρακτηριστικά δύο σημαντικών αρχιτεκτονικών μοντέλων για την εργασία.

3.1 Μηχανική Μάθηση

Η μηχανική μάθηση (machine learning), μέρος της τεχνητής νοημοσύνης, είναι ένα σύνολο μεθόδων και αλγορίθμων πρόβλεψης που αποσκοπούν στην αυτόματη εκμάθηση από δεδομένα, στην αναγνώριση μοτίβων και στην επίλυση προβλημάτων με ελάχιστη ανθρώπινη συμμετοχή. Υπάρχουν τρεις κύριοι τύποι μάθησης [37]:

- Επιβλεπόμενη μάθηση (supervised learning)
- Μη επιβλεπόμενη μάθηση (unsupervised learning)
- Ενισχυτική μάθηση (reinforcement learning)

Στην επιβλεπόμενη (supervised) οι αλγόριθμοι παρατηρούν τις εισόδους (inputs) και τις εξόδους (outputs) που τους δίνονται και παράγουν μία συνάρτηση που αντιστοιχίζει τις εισόδους στις εξόδους. Μερικοί αλγόριθμοι επιβλεπόμενης μάθησης είναι τα δέντρα αποφάσεων (decision trees), οι μηχανές διανυσματικής υποστήριξης (support vector machines-SVM), τα τεχνητά νευρωνικά δίκτυα (artificial neural networks), η γραμμική παλινδρόμηση (linear regression) και η λογιστική παλινδρόμηση (logistic regression).

Εν αντιθέσει, στην μη επιβλεπόμενη μάθηση το μοντέλο δεν διαθέτει προκαθορισμένες ετικέτες (labels) ή σωστές απαντήσεις για τα δεδομένα που δέχεται. Αντί να προσπαθεί να προβλέψει μια συγκεκριμένη έξοδο για κάθε είσοδο, το μοντέλο στοχεύει να ανακαλύψει δομές, πρότυπα και σχέσεις μέσα στα δεδομένα χωρίς να του παρέχονται ετικέτες. Μερικοί ευρέως γνωστοί αλγόριθμοι μη επιβλεπόμενης μάθησης

είναι η συσταδοποίηση k-μέσων (k-means clustering), η ανάλυση σε κύριες συνιστώσες (Principal Component Analysis-PCA), οι αυτοκωδικοποιητές (autoencoders), και τα παραγωγικά ανταγωνιστικά δίκτυα (Generative Adversarial Networks-GAN).

Η ημιεπιβλεπόμενη μάθηση (semi-supervised learning) είναι μια προσέγγιση της μηχανικής μάθησης που συνδυάζει στοιχεία της επιβλεπόμενης και μη επιβλεπόμενης μάθησης Σε αυτόν τον τύπο μάθησης, ένα μοντέλο εκπαιδεύεται χρησιμοποιώντας ένα σύνολο δεδομένων που περιέχει δείγματα με ετικέτες και δείγματα χωρίς ετικέτες.

Τέλος, η ενισχυτική μάθηση είναι μια προσέγγιση της μηχανικής μάθησης, όπου ένας αλγόριθμος εκπαιδεύεται να λαμβάνει αποφάσεις μέσω αλληλεπιδράσεων με ένα περιβάλλον. Σε αυτόν τον τύπο μάθησης, το μοντέλο, ή όπως αποκαλείται, πράκτορας (agent), μαθαίνει να επιλέγει ενέργειες που το ανταμοίβουν και να αποφεύγει ενέργειες που το τιμωρούν, με στόχο να μεγιστοποιήσει ή να ελαχιστοποιήσει ένα συνολικό μέγεθος, π.χ. κέρδος, κόστος. Η ενισχυτική μάθηση εφαρμόζεται σε πολλούς τομείς, όπως την ρομποτική, την αυτοματοποίηση, τα παιχνίδια, τη διαχείριση πόρων και τις εξατομικευμένες προτάσεις προϊόντων.

3.2 Τεχνητά Νευρωνικά Δίκτυα

Τα τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks-ANN), εμπνευσμένα από τον τρόπο λειτουργίας βιολογικών νευρικών κυκλωμάτων, είναι αρχιτεκτονικές που διευκολύνουν τα υπολογιστικά συστήματα να μάθουν από δεδομένα που τους δίνονται. Τα νευρωνικά δίκτυα δομούνται από πολλαπλά υπολογιστικά επίπεδα (layers) με στόχο την παροχή μιας λογικής απόφασης ή ενός ορθού αποτελέσματος. Κάθε επίπεδο αποτελείται από πολλαπλούς τεχνητούς νευρώνες. Οι είσοδοι των νευρώνων μπορεί να είναι είτε καθαρά δεδομένα, είτε χαρακτηριστικά (features) ή μετρήσιμα στοιχεία που χαρακτηρίζουν τα δεδομένα.

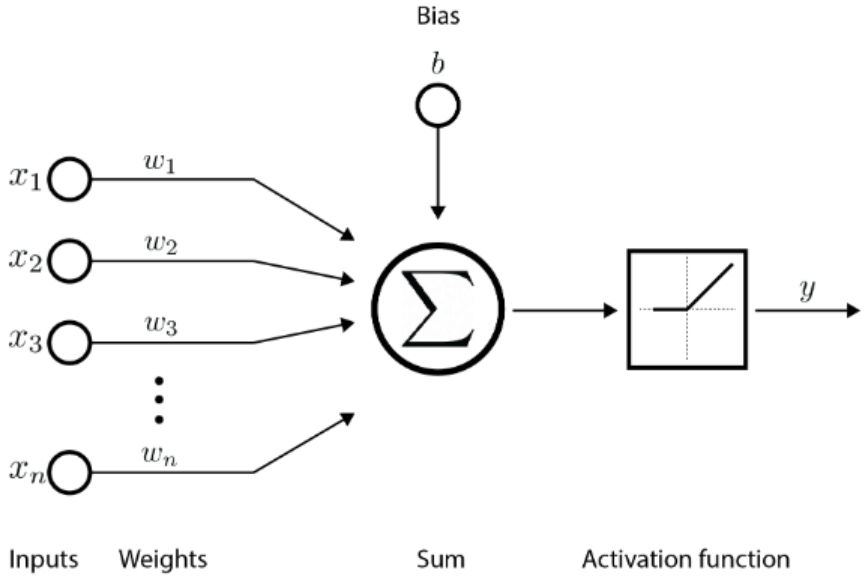
3.2.1 Νευρώνας

Το βασικό θεμέλιο και η υπολογιστική μονάδα των περισσότερων νευρωνικών δικτύων είναι ο νευρώνας, γνωστός και ως perceptron. Αντιστοιχίζει τις εισόδους x_1, x_2, ..., x_n, σε μια έξοδο y ύστερα από μια σειρά υπολογισμών. Κάθε x_n πολλαπλασιάζεται

με το βάρος (weight) w_n που του αναλογεί, πριν αθροιστούν όλα μαζί. Προεραιτικά μπορεί να προστεθεί στο σύνολο μια μεροληψία (bias) b. Τέλος το άθροισμα προωθείται στην συνάρτηση ενεργοποίησης (activation function), η οποία το μετατρέπει σε μια τιμή εξόδου. Η εικόνα 1 είναι μια σχηματική αναπαράσταση ενός νευρώνα. Αν $\mathbf{X}$, $\mathbf{W}$ οι πίνακες των εισόδων και βαρών αντίστοιχα, και σ μια συνάρτηση ενεργοποίησης, η έξοδος του νευρώνα μπορεί να δοθεί από την εξίσωση:

$$\mathbf{W} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$y = \sigma(\mathbf{X}^{\mathbf{T}}\mathbf{W} + b) \tag{1}$$



Σχήμα 1: Νευρώνας.

3.2.2 Συνάρτηση Ενεργοποίησης

Όπως υποδηλώνει η ονομασία, οι συναρτήσεις ενεργοποίησης καθορίζουν αν θα ενεργοποιηθεί ο νευρώνας ή όχι, ανάλογα με την τιμή του σταθμισμένου αθροίσματος

των εισόδων. Χωρίς την συνάρτηση ενεργοποίησης, η έξοδος του νευρώνα θα ήταν γραμμική ως προς τις εισόδους του, και συνεπώς νευρωνικά δίκτυα θα περιορίζονταν στον χώρο των γραμμικών προβλημάτων. Η μη γραμμικότητα που προσφέρουν οι συναρτήσεις ενεργοποίησης, επιτρέπουν στα ANN να σχηματίσουν αυθαίρετες συναρτήσεις και να λύσουν περίπλοκα προβλήματα. Ορισμένες από τις πιο γνωστές συναρτήσεις ενεργοποίησης είναι οι ακόλουθες:

- Συνάρτηση ReLU (Rectified Linear Unit function). Δημοφιλής για την ταχύτητα υπολογισμού της και την απλότητα της.

$$\sigma(x) = max(0, x) \tag{2}$$

  Ακόμη μια ιδιότητα που την κάνει επιθυμητή, είναι η πρώτη παράγωγος της που υπολογίζεται εύκολα και είναι είτε 0, είτε 1. Αυτό είναι ευνοϊκό για τον αλγόριθμο back propagation που θα δούμε αργότερα.

- Σιγμοειδής συνάρτηση (sigmoid function). Οι τιμές εξόδου της συνάρτησης ανήκουν στο διάστημα $(0, 1)$, γεγονός που την κάνει κατάλληλη για προβλήματα δυαδικής ταξινόμησης (binary classification).

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

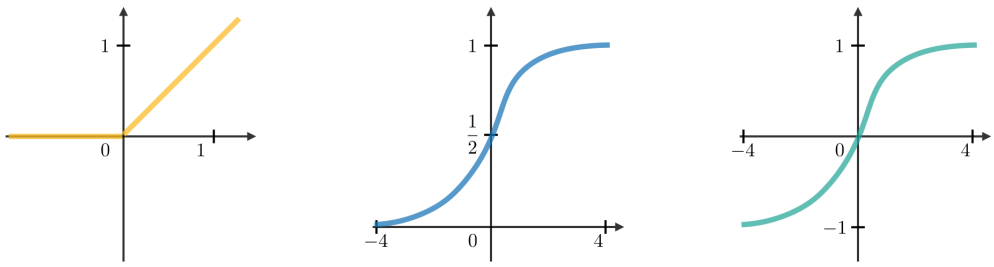
  Μια παραλλαγή της σιγμοειδής συνάρτησης για ταξινόμηση πολλαπλών κλάσεων είναι η συνάρτηση softmax. Χρησιμοποιείται κυρίως στα επίπεδα εξόδου των νευρωνικών δικτύων, καθώς μετατρέπει τις τιμές των νευρώνων σε κατανομές πιθανοτήτων.

$$\sigma(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{j=i}^{K} e^{x_j}} \quad \text{για } i = 1, \ldots, K \tag{4}$$

- Συνάρτηση υπερβολικής εφαπτομένης (Tanh function). Παρόμοια με την σιγμοειδή συνάρτηση, η tanh αντιστοιχίζει τις τιμές εισόδου στο διάστημα $(-1, 1)$.

$$\sigma(\mathbf{x}) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5}$$

Αξίζει να παρατηρήσουμε στο σχήμα 2 ότι οι συναρτήσεις ενεργοποίησης είναι αύξουσες και συνεπώς οι πρώτες παράγωγοι είναι μη αρνητικοί.



(α΄) ReLU **(β΄)** Sigmoid **(γ΄)** Tanh

Σχήμα 2: Συναρτήσεις ενεργοποίησης [48].

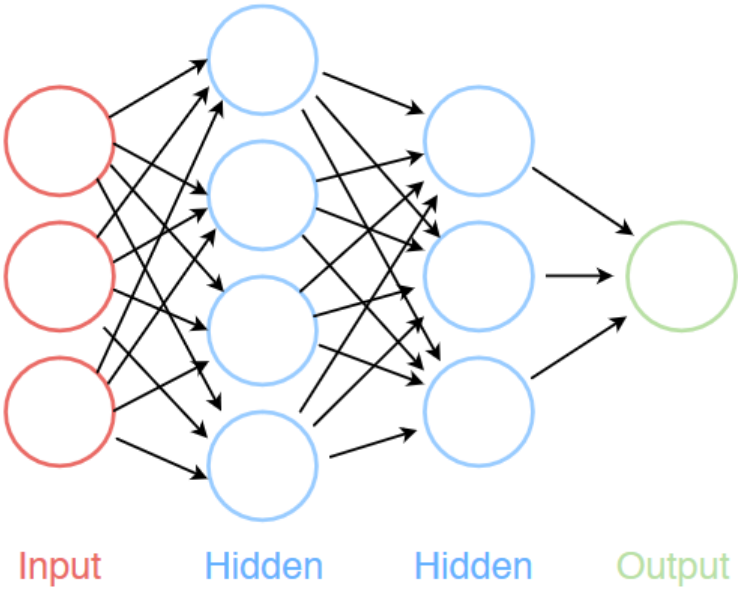
3.2.3 Πολυεπίπεδος Perceptron

Ένας πολυεπίπεδος perceptron (multilayer perceptron - MLP) είναι μια θεμελιώδης αρχιτεκτονική στο πεδίο των τεχνητών νευρωνικών δικτύων. Ένα MLP είναι ένα σύνολο διασυνδεδεμένων επιπέδων που αποτελούνται από νευρώνες. Ανήκει στο σύνολο αρχιτεκτονικών νευρωνικών δικτύων προώθησης (feedforward neural network - FNN) Έχουν σχεδιαστεί για να επεξεργάζονται δεδομένα που ρέουν από το input layer, στα κρυφά επίπεδα (hidden layers), και τέλος, στο output layer. Οι τιμές εξόδου $y^{(n)}$ για κάποιον n-οστό layer μπορούν να γραφούν ως:

$$\begin{bmatrix} y_1^{(n)} \\ y_2^{(n)} \\ \vdots \\ y_k^{(n)} \end{bmatrix} = \sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,l} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,l} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,l} \end{bmatrix} \begin{bmatrix} y_1^{(n-1)} \\ y_2^{(n-1)} \\ \vdots \\ y_l^{(n-1)} \end{bmatrix} + \begin{bmatrix} b_1^{(n)} \\ b_2^{(n)} \\ \vdots \\ b_k^{(n)} \end{bmatrix} \right) \tag{6}$$

$$\mathbf{y}^{(n)} = \sigma\left(\mathbf{W}\mathbf{a}^{n-1} + \mathbf{b}^n\right) \tag{7}$$

όπου k ο αριθμός των νευρώνων στον n-οστό layer και l ο αριθμός των νευρώνων στον $n-1$-οστό layer. Η 3 είναι μια σχηματική αναπαράσταση ενός MLP. Όταν το μοντέλο έχει παραπάνω από δύο hidden layers το αποκαλούμε βαθύ νευρωνικό δίκτυο (deep neural network - DNN).



Σχήμα 3: Πολυεπίπεδος Perceptron.

3.2.4 Συναρτήσεις Κόστους

Η συνάρτηση κόστους (cost function), επίσης γνωστή ως συνάρτηση απώλειας (loss function), είναι ένας τρόπος μέτρησης της συνολικής απόδοσης ενός μοντέλου μηχανικής μάθησης. Ποσοτικοποιεί το πόσο καλά οι προβλέψεις ενός μοντέλου αντιστοιχούν στις πραγματικές τιμές. Κύριος στόχος κατά την διάρκεια της εκπαίδευσης ενός μοντέλου είναι να ελαχιστοποιηθεί η τιμή της συνάρτησης κόστους και έτσι το μοντέλο να κάνει πιο ακριβείς προβλέψεις. Μια διαδεδομένη loss function είναι η cross-entropy. Αν (x_i, y_i), όπου x_i τα inputs και y_i τα πραγματικά outputs, και $\hat{y}_i$ τα outputs που προέβλεψε το μοντέλο, τότε η cross-entropy loss δίνεται από τον τύπο:

$$L(y_i, \hat{y}_i) = -\frac{1}{N}\sum_{i=1}^{N} [\hat{y}_i log(y_i) + (1-\hat{y}_i) log(1-y_i)] \tag{8}$$

Αυτός ο τύπος αφορά τα binary classification προβλήματα. Να σημειώσουμε ότι ο τύπος ορίζεται όταν $0 < \hat{y} < 1$. Εξαιτίας αυτού, μια συνάρτηση σαν τη σιγμοειδή, όπως ορίστηκε στην εξίσωση 3, είναι κατάλληλη για activation functionσε output layers.

3.2.5 Εκπαίδευση Νευρωνικών Δικτύων

Εκπαίδευση ενός νευρωνικού δικτύου είναι ουσιαστικά η εύρεση των παραμέτρων του μοντέλου που ελαχιστοποιούν την συνάρτηση κόστους. Αυτό γίνεται με τον αλγόριθμο κατάβασης κλίσης (gradient descent), ο οποίος προσαρμόζει τις παραμέτρους, βασιζόμενος στην παράγωγο ως προς τις παραμέτρους της συνάρτησης κόστους, δηλαδή:

$$w_{ij}^{t} \leftarrow w_{ij}^{t-1} - a\frac{\partial L}{\partial w_{ij}^{t-1}} \tag{9}$$

όπου a είναι ο ρυθμός μάθησης (learning rate) και t το χρονικό βήμα. Η παράγωγος του κόστους υπολογίζεται με τον αλγόριθμο οπισθοδιάδοσης (backpropagation). Αυτός, εκμεταλλευόμενος τον κανόνα της αλυσίδας, υπολογίζει τις παραγώγους layer προς layer, ξεκινώντας από τον output layer πηγαίνοντας πίσω προς τον input layer, εξού και το όνομα.

$$\frac{\partial L}{\partial w_{ij}^{k}} = \frac{\partial L}{\partial y_{j}^{k}}\frac{\partial y_{j}^{k}}{\partial w_{ij}^{k}} \tag{10}$$

Λόγω του κανόνα της αλυσίδας, η παράγωγος της απώλειας εξαρτάται και από την παράγωγο της συνάρτησης ενεργοποίησης $\sigma'(x)$, για αυτό η συνάρτηση ReLU είναι κατάλληλη, μιας και είναι οικονομικός ο υπολογισμός της παραγώγου της.

Η εφαρμογή του gradient descent μπορεί να γίνει χρονοβόρα όταν έχουμε μεγάλο σύνολο δεδομένων, καθώς ο υπολογισμός των παραμέτρων γίνεται χρησιμοποιώντας όλα τα δεδομένα. Για να βελτιωθεί ο χρόνος εκπαίδευσης, αναπτύχθηκαν διάφορες παραλλαγές, όπως η στοχαστική κατάβαση κλίσης (stochastic gradient descent - SGD), όπου αντί για όλα τα δεδομένα, ο αλγόριθμος επιλέγει τυχαίο ένα εκπαιδευτικό παράδειγμα για να υπολογίσει την παράγωγο σε κάθε επανάληψη της μάθησης, η κατάβαση κλίσης με μίνι φουρνιές (mini batch gradient descent), όπου το αρχικό σύνολο χωρίζεται σε μικρές φουρνιές και η παράγωγος υπολογίζεται για κάθε batch, η διάδοση μέσης τετραγωνικής ρίζας (Root Mean Square Propagation - RMSProp), η οποία προσαρμόζει τον ρυθμό μάθησης για κάθε παράμετρο, αναλόγως το μέγεθος των παραγώγων τους, και άλλες. Η πιο δημοφιλής παραλλαγή, είναι ο αλγόριθμος εκτίμησης προσαρμοστικής ροπής (Adaptive Moment Estimation - Adam). Ο Adam

προσαρμόζει την ρυθμό μάθησης για κάθε παράμετρο ανάλογα με το μέγεθος των παραγώγων. Οι παράμετροι με μεγαλύτερες παραγώγους λαμβάνουν μικρότερους ρυθμούς μάθησης, προλαμβάνοντας έτσι την απόκλιση κατά την μάθηση. Ακόμη διατηρεί μέσους όρους από προηγούμενες παραγώγους και τετραγωνισμένες παραγώγους που φθίνουν εκθετικά. Ο αλγόριθμος ενημερώνει τις παραμέτρους με τα ακόλουθα βήματα:

1. Υπολογισμός των πρώτων και δεύτερων ροπών των παραγώγων:
   - Πρώτη ροπή (μέση τιμή): $m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
   - Δεύτερη ροπή (μη επικεντρωμένη διακύμανση): $v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
2. Υπολογισμός των ροπών με διόρθωση της μεροληψίας:
   - $m_t^{corrected} = \frac{m_t}{1-\beta_1^t}$
   - $v_t^{corrected} = \frac{v_t}{1-\beta_2^t}$
3. Ενημέρωση των παραμέτρων: $w_{t+1} = w_t - \frac{a}{\sqrt{v_t^{corrected}} + \epsilon} \cdot m_t^{corrected}$

όπου g_t είναι η παράγωγος της συνάρτησης κόστους ως προς τις παραμέτρους του μοντέλου την επανάληψη t, β_1 και β_2 είναι δείκτες ελάττωσης και ϵ μια μικρή σταθερά που αποτρέπει την διαίρεση με το μηδέν. Η ιδιότητα του αλγορίθμου Adam να συγκλίνει γρηγορότερα απο άλλους αλγόριθμους και να απαιτεί ελάχιστες προσαρμογές τον κάνουν αρκετά ελκυστικό.

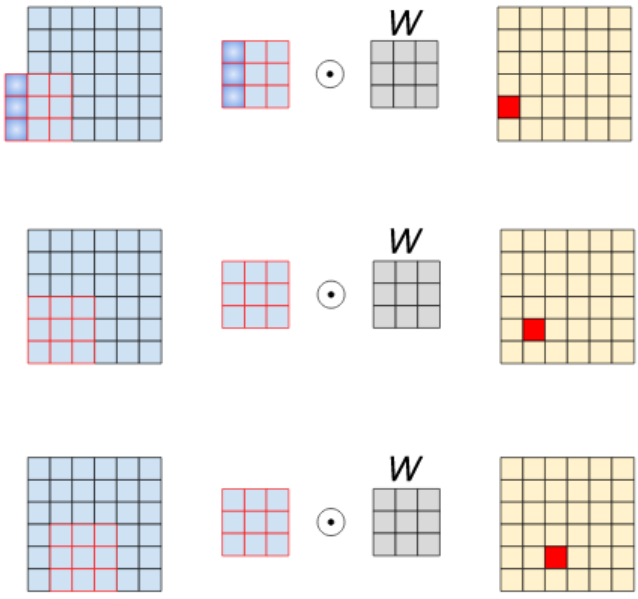
3.3 Συνελικτικά Νευρωνικά Δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (convolutional neural network) είναι τύποι FNN σχεδιασμένα για να επεξεργάζονται και να αναλύουν δεδομένα που μπορούν να αναπαρασταθούν ως πλέγματα, όπως εικόνες, βίντεο και φασματογραφήματα ήχου. Η ικανότητα τους να μαθαίνουν ιεραρχικά χαρακτηριστικά, από features χαμηλού επιπέδου (υφές, παρυφές) μέχρι υψηλού επιπέδου (περίπλοκα σχήματα, αντικείμενα), αλλά και εξαρτήσεις χρόνου που παρουσιάζονται στα inputs, τα κάνουν να διαπρέπουν σε προβλήματα όπως ταξινόμησης εικόνων (image classification), ανίχνευσης

αντικειμένων (object detection), και άλλων. Τα CNN σχηματίζονται με μία στοίβα από συνελικτικά επίπεδα (convolution layers), συγκεντρωτικά επίπεδα (pooling layers) και τέλος ένα πλήρως συνδεδεμένο επίπεδο (fully connected layer).

3.3.1 Συνελικτικό Επίπεδο

Στα convolution layers πραγματοποιείται το σημαντικότερο μέρος των εργασιών που εκτελούν αυτοί οι τύποι δικτύων, για αυτό και υιοθετήθηκε το όνομα για αυτά τα δίκτυα. Στην πιο απλή μορφή της, μια πράξη συνέλιξης μπορεί να περιγραφεί ως ένα φίλτρο (filter) ή πυρήνας (kernel) που προχωράει κατά μήκος ενός input, για παράδειγμα μιας εικόνας, και υπολογίζει το εσωτερικό γινόμενο του φίλτρου και της περιοχής της εικόνας στην οποία βρίσκεται το φίλτρο. Το αποτέλεσμα που παράγεται είναι μια νέα εικόνα, που ονομάζεται χάρτης χαρακτηριστικών (feature map). Ο ρυθμός με τον οποίο περνάει ένα φίλτρο πάνω από μια εικόνα, ονομάζεται βηματισμός (stride). Μια εικονική αναπαράσταση της πράξης της συνέλιξης δίνεται στο σχήμα 4.



Σχήμα 4: Διαδικασία Συνέλιξης [47].

Πιο αφηρημένα, ένας convolution layers δέχεται ένα **I** input με σχήμα $I_w \times I_h \times M$, όπου I_w το πλάτος, I_h το ύψος, M το βάθος ή κανάλι (channel, π.χ. οι εικόνες RGB έχουν τρία κανάλια) και εφαρμόζει ένα **K** φίλτρο μεγέθους $K_w \times K_h \times M \times N$ στο input,

με K_w, K_h πλάτος και ύψος και κανάλι $M \times N$. Το $\mathbf{O}$ output θα έχει σχήμα $O_w \times O_h \times N$, με O_w πλάτος, O_h ύψος, και κανάλι N. Το output feature map με stride 1 μπορεί να υπολογιστεί ως:

$$O_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot I_{k+i-1,l+j-1,m} \tag{11}$$

Έτσι το υπολογιστικό κόστος των convolutions είναι:

$$I_w \cdot I_h \cdot M \cdot N \cdot K_w \cdot K_h \cdot \tag{12}$$

Σε κάποιες περιπτώσεις, το φίλτρο δε μπορεί να περάσει από όλη την είσοδο. Λόγου χάρη, ένα φίλτρο 2×2 με βηματισμό 2 δε μπορεί να εφαρμοστεί πλήρως σε ένα feature map μεγέθους 5×5. Επομένως, είναι εύλογο να ενισχύουμε τις άκρες των feature maps με μηδενικά, για να καλύψουμε τα κομμάτια που δεν ¨χωράνε¨. Αυτή η τακτική ονομάζεται zero-padding.

Για να εισαχθεί το στοιχείο της μη γραμμικότητας στο μοντέλο, συνηθίζεται να εφαρμόζεται μια συνάρτηση ενεργοποίησης στο feature map και το αποτέλεσμα αποκαλείται χάρτης ενεργοποίησης (activation map). Η συνάρτηση που χρησιμοποιείται πιο συχνά είναι η ReLU, λόγω της ιδιότητας της να εξαλείφει όλες τις αρνητικές τιμές, αλλά και της συνεισφοράς της στον υπολογισμό της παραγώγου κόστους.

3.3.2 Επίπεδο Συγκέντρωσης

Ύστερα από έναν convolution layers, ένας pooling layer μειώνει το μέγεθος των feature/activation maps. Η τεχνική που εφαρμόζει αποκαλείται υποδειγματοληψία (down-sampling) και αυτό που κάνει, είναι να περνάει ένα φίλτρο με κάποιο stride πάνω από το feature mapκαι να υπολογίζει, ως επί το πλείστον, το μέγιστο ή τον μέσο όρο. Αυτά τα αποκαλούμε μέγιστη συγκέντρωση (max pooling) και μέση συγκέντρωση (average pooling), αντιστοίχως. Στόχος του pooling layer είναι μειώσει το υπολογιστικό κόστος να εξάγει αντιπροσωπευτικά χαρακτηριστικά και να αποτρέψει το overfitting, κατάσταση στην οποία το μοντέλο ανταπεξέρχεται ικανοποιητικά στα δεδομένα εκπαίδευσης αλλα αδυνατεί να κάνει γενίκευση σε ξένα δεδομένα.

3.3.3 Πλήρως Συνδεδεμένο Επίπεδο

Τυπικά, αν και όχι αναγκαία, το τελευταίο layer ενός CNN είναι το fully connected layer. Μετά από μια σειρά από convolution και pooling layers, τα τελικά features πλατύνονται (flatten) και προωθείται σε ένα κλασικό feed forward neural network, το οποίο δουλεύει όπως αναλύσαμε προηγουμένως.

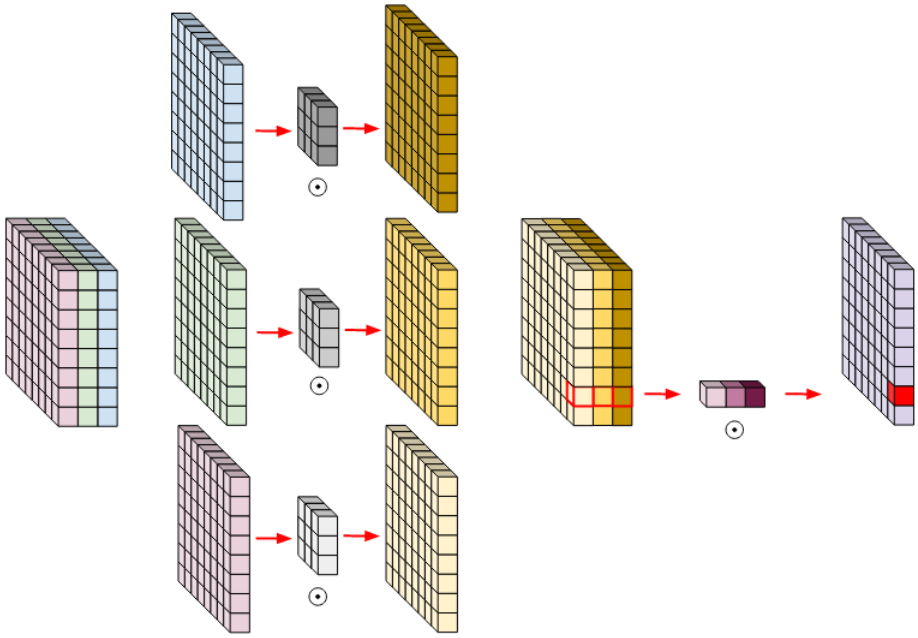
3.3.4 Διαχωρισμένες Κατά Βάθος Συνελίξεις

Η διαδικασία σχεδιασμού αποτελεσματικών CNN για περίπλοκες εργασίες είχε αρκετές προκλήσεις στο παρελθόν. Οι παραδοσιακές συνελίξεις, αν και ισχυρές, ήταν υπολογιστικά κοστοβόρες και κατανάλωναν αρκετή μνήμη λόγω των πολυάριθμων παραμέτρων τους. Αυτό περιόριζε την επεκτασιμότητά τους και την πρακτικότητα της ανάπτυξης μοντέλων deep learning για συσκευές με περιορισμένους πόρους, όπως κινητά και edge συσκευές. Για την αντιμετώπιση αυτών των ζητημάτων οι Russakovsky at al. (2015) [30] εισήγαγαν τις διαχωρισμένες κατά βάθος συνελίξεις (depth-wise separable convolutions). Αυτά τα ειδικά convolutions διασπούν την διαδικασία συνέλιξης σε δύο διακριτά βήματα: συνέλιξη σε βάθος (depth-wise convolution), η οποία συλλαμβάνει χωρικές πληροφορίες για κάθε κανάλι ανεξάρτητα, και σημειακή συνέλιξη (point-wise convolution), η οποία συνδυάζει αυτές τις πληροφορίες μεταξύ των καναλιών. Αυτή η καινοτόμος προσέγγιση αντιμετωπίζει τις προκλήσεις βελτιστοποιώντας τον τρόπο με τον οποίο τα νευρωνικά δίκτυα επεξεργάζονται πληροφορίες, καθιστώντας τα πιο αποτελεσματικά και πρακτικά για ένα ευρύτερο φάσμα εφαρμογών. Η εικόνα 5 είναι μια σχηματική αναπαράσταση των depth-wise separable convolutions.

Μια depth-wise convolution που εφαρμόζει ένα φίλτρο σε κάθε κανάλι input, μπορεί να γραφεί ως:

$$\hat{O}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} \cdot I_{k+i-1,l+j-1,m} \tag{13}$$

όπου $\hat{\mathbf{K}}$ ένα depth-wise convolutionφίλτρο μεγέθους $K_w \times K_h \times M$ και το m-οστό φίλτρο του $\hat{\mathbf{K}}$ εφαρμόζεται στο m-οστό κανάλι του $\mathbf{I}$ για να παράξει το m-οστό feature του $\mathbf{O}$.



Σχήμα 5: Διαχωρισμένες Κατά Βάθος Συνελίξεις [47].

Έτσι, το υπολογιστικό κόστος μιας depth-wise convolution είναι:

$$I_w \cdot I_h \cdot M \cdot K_w \cdot K_h \tag{14}$$

Μια point-wise convolution που εφαρμόζει ένα φίλτρο $\hat{\mathbf{K}}$ μεγέθους 1×1 και στην συνέχεια υπολογίζει τον γραμμικό συνδυασμό των εξόδων των depth-wise convolutions γράφεται ως:

$$O_{k,l,n} = \sum_m \hat{K}_{1,1,m,n} \cdot \hat{O}_{k,l,m} \tag{15}$$

με κόστος:

$$I_w \cdot I_h \cdot M \cdot 1 \cdot 1 \cdot N \tag{16}$$

Προσθέτοντας τα κόστη 14 και 16, το κόστος ενός depth-wise convolution είναι:

$$I_w \cdot I_h \cdot M \cdot K_w \cdot K_h + I_w \cdot I_h \cdot M \cdot N \tag{17}$$

Αν το διαιρέσουμε με την 12, και συμπεριλάβουμε ότι $K_w = K_h = k$, παρατηρούμε πόσο

πιο αποδοτική είναι οι depth-wise convolutions σε σχέση με τις απλές convolutions:

$$\frac{I_w \cdot I_h \cdot M \cdot K_w \cdot K_h + I_w \cdot I_h \cdot M \cdot N}{I_w \cdot I_h \cdot M \cdot K_w \cdot K_h} = \frac{1}{N} + \frac{1}{k^2} \tag{18}$$

3.3.5 MobileNets

Μια οικογένεια αρχιτεκτονικών νευρωνικών δικτύων που έχουν σχεδιαστεί για να χειρίζονται αποτελεσματικά τις προκλήσεις της ανάπτυξης μοντέλων βαθιάς μάθησης σε συσκευές με περιορισμένους πόρους, όπως κινητά τηλέφωνα και ενσωματωμένα συστήματα είναι τα MobileNets [13]. Ανεπτυγμένα από ερευνητές της Google, τα MobileNets χρησιμοποιούν depth-wise convolutions ως βασικό δομικό στοιχείο για να επιτύχουν μια αξιοσημείωτη ισορροπία μεταξύ της απόδοσης του μοντέλου και της υπολογιστικής αποδοτικότητας. Χάρη στα depth-wise convolutions, μειώνουν σημαντικά τον αριθμό των υπολογισμών και των παραμέτρων, καθιστώντας τα κατάλληλα για εφαρμογές σε πραγματικό χρόνο σε συσκευές με περιορισμένη υπολογιστική ισχύ και μνήμη. Αυτή η αποτελεσματικότητα δεν θέτει σε κίνδυνο την ικανότητα του δικτύου να καταγράφει σημαντικές λειτουργίες από δεδομένα, καθώς τα MobileNets επιτυγχάνουν σταθερά ανταγωνιστική απόδοση σε διάφορες εργασίες όρασης υπολογιστή (computer vision), συμπεριλαμβανομένης της ταξινόμησης εικόνων, της ανίχνευσης αντικειμένων και της σημασιολογικής τμηματοποίησης (semantic segmentation). Η αρχιτεκτονική του MobileNetV1 παρουσιάζεται τον πίνακα 1.

Για να βελτιωθεί περαιτέρω η αντιστάθμιση μεταξύ μεγέθους και ακρίβειας μοντέλου, το MobileNetV1 εισάγει παραμέτρους που αποκαλούνται πολλαπλασιαστής πλάτους (width multiplier) και πολλαπλασιαστής ανάλυσης (resolution multiplier). Ο πολλαπλασιαστής πλάτους κλιμακώνει τον αριθμό των φίλτρων σε κάθε επίπεδο, ελέγχοντας αποτελεσματικά το πλάτος του μοντέλου. Ένας μικρότερος πολλαπλασιαστής πλάτους μειώνει τον αριθμό των παραμέτρων και των υπολογισμών, καθιστώντας το μοντέλο πιο ελαφρύ, αλλά δυνητικά θυσιάζει μερική απόδοση. Από την άλλη πλευρά, ο πολλαπλασιαστής ανάλυσης προσαρμόζει το μέγεθος της εικόνας εισόδου. Η μείωση της ανάλυσης συμβάλλει στη μείωση των υπολογιστικών απαιτήσεων, καθιστώντας τα MobileNets ακόμη πιο κατάλληλα για περιβάλλοντα χαμηλών πόρων.

Πίνακας 1: Αρχιτεκτονική του MobileNetV1 [13].

Input size $224 \times 224 \times 3$			
Block	Layer	Filter, Stride	Output Size
block 1	conv	3×3, s2	$112 \times 112 \times 32$
block 2	conv dw	3×3, s1	$112 \times 112 \times 32$
	conv	1×1, s1	$112 \times 112 \times 64$
block 3	conv dw	3×3, s2	$56 \times 56 \times 64$
	conv	1×1, s1	$56 \times 56 \times 128$
block 4	conv dw	3×3, s1	$56 \times 56 \times 128$
	conv	1×1, s1	$56 \times 56 \times 128$
block 5	conv dw	3×3, s2	$28 \times 28 \times 128$
	conv	1×1, s1	$28 \times 28 \times 256$
block 6	conv dw	3×3, s1	$28 \times 28 \times 256$
	conv	1×1, s1	$28 \times 28 \times 256$
block 7	conv dw	3×3, s2	$14 \times 14 \times 256$
	conv	1×1, s1	$14 \times 14 \times 512$
block 8-12	conv dw	3×3, s1	$14 \times 14 \times 512$
	conv	1×1, s1	$14 \times 14 \times 512$
block 13	conv dw	3×3, s2	$7 \times 7 \times 512$
	conv	1×1, s1	$7 \times 7 \times 1024$
block 14	conv dw	3×3, s2	$7 \times 7 \times 1024$
	conv	1×1, s1	$7 \times 7 \times 1024$
classification block	avg pool	7×7, s1	$1 \times 1 \times 1024$
	FC	1024×1000, s1	$1 \times 1 \times 1000$
	Softmax	Classifier, s1	Output

Αυτοί οι πολλαπλασιαστές προσφέρουν έναν λεπτομερή έλεγχο της αντιστάθμισης μεταξύ της απόδοσης και της ακρίβειας του μοντέλου (όπως φαίνεται από τους πίνακες 2, 3), επιτρέποντας την προσαρμογή για να ταιριάζει με τις συγκεκριμένες απαιτήσεις των διαφορετικών εφαρμογών.

Πίνακας 2: MobileNet Width Multiplier [13].

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.50 MobileNet-224	63.7%	159	1.3
0.25 MobileNet-224	50.4%	41	0.5

Πίνακας 3: MobileNet Resolution [13].

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	190	4.2
1.0 MobileNet-128	64.4%	186	4.2

Συνεχίζοντας την κληρονομιά του MobileNetV1, το MobileNetV2 εισάγει ανεστραμμένα υπολείμματα (inverted residuals), γραμμική συμφόρηση (linear bottleneck) και πιο αποτελεσματικές αρχιτεκτονικές. Το MobileNetV3 εισάγει την αναζήτηση αρχιτεκτονικής, βελτιώνοντας την απόδοση και την ακρίβεια. Τόσο το V2 όσο και το V3 στοχεύουν στη βελτίωση της αποδοτικότητας του MobileNetV1 για συσκευές με περιορισμένους πόρους.

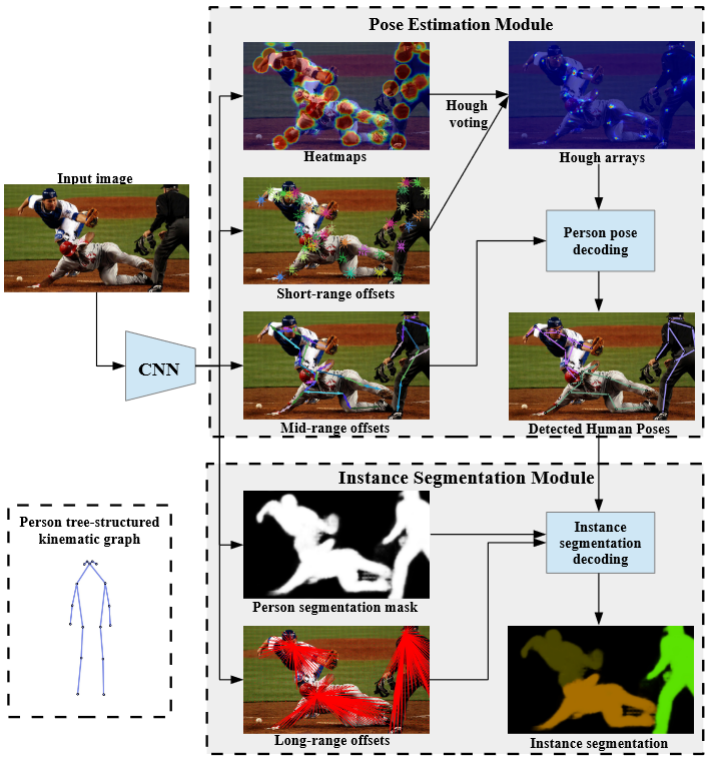
Ουσιαστικά, τα MobileNets αποτελούν παράδειγμα της ιδέας της επίτευξης ισορροπίας μεταξύ της υπολογιστικής απόδοσης και της ακρίβειας του μοντέλου. Προσφέρουν ένα φάσμα αρχιτεκτονικών, επιτρέποντας στους προγραμματιστές να προσαρμόσουν το νευρωνικό δίκτυο στους συγκεκριμένους περιορισμούς της συσκευής-στόχου τους, ενώ παράλληλα επιτυγχάνουν μια λογική ακρίβεια. Αυτή η ευελιξία έχει κάνει τα MobileNets μια δημοφιλή επιλογή για εφαρμογές machine learning σε ένα ευρύ φάσμα συσκευών.

3.4 Εκτίμηση Σωματικής Στάσης - PersonLab

Η εκτίμηση σωματικής στάσης (pose estimation) στην υπολογιστική όραση αφορά τον προσδιορισμό των θέσεων των βασικών αρθρώσεων του σώματος σε μια εικόνα, ώστε να κατανοηθεί η στάση του σώματος. Επιτρέπει στις μηχανές να ερμηνεύουν την ανθρώπινη κίνηση και στάση, συμβάλλοντας σε εφαρμογές όπως η αναγνώριση δραστηριότητας, ο έλεγχος χειρονομιών και η αλληλεπίδραση ανθρώπου-υπολογιστή. Το pose estimation μπορεί να γίνει με δύο προσεγγίσεις, την πάνω-προς-κάτω (top-down), όπου πρωτίστως μέσω object detection εντοπίζεται ολόκληρος ο άνθρωπος με ένα πλαίσιο οριοθέτησης (bounding box) και στην συνέχεια εντοπίζονται οι αρθρώσεις μέσα bounding box , και η κάτω-προς-πάνω (bottom-up), όπου εκτιμούνται απευθείας οι αρθρώσεις από την εικόνα και στην συνέχεια οργανώνονται για να κατασκευάσουν την στάση. Με το HRNet μπορεί να εκτιμηθεί η στάση ανθρώπου στον δισδιάστατο αλλά και στον τρισδιάστατο χώρο. Το OpenPose [4], το Mask R-CNN [11], το HRNet [34] και το AlphaPose [8] είναι ορισμένα pose estimation μοντέλα.

Οι ερευνητές Papandreou et al. [25] παρουσίασαν το 2018 το PersonLab, μια νέα προσέγγιση για την ταυτόχρονη ανίχνευση πολλαπλών ανθρώπων, την εκτίμηση της στάσης τους και την τμηματοποίηση παρουσίας (instance segmentation) σε εικόνες. Αυτή η μέθοδος ακολουθεί την bottom-up προσέγγιση, εστιάζοντας στον εντοπισμό 17 μεμονωμένων σημείων κλειδιών (keypoints) σώματος και προσώπου για κάθε άτομο και στη συνέχεια στην ομαδοποίηση τους. Το σύστημα PersonLab, όπως παρουσιάζεται στο σχήμα 6 έχει την ακόλουθη λογική.

Στόχος είναι να ανιχνευθούν ορατά keypoints που ανήκουν σε άτομα στην εικόνα. Το μοντέλο, χρησιμοποιώντας ένα CNN, προβλέπει χάρτες θερμότητας (heatmaps), όπου κάθε channel αντιστοιχεί σε έναν συγκεκριμένο τύπο keypoint , π.χ. αριστερός αγκώνας, υποδεικνύοντας την πιθανότητα ύπαρξης ενός keypoint σε κάποια περιοχή της εικόνας. Τα διανύσματα μετατόπισης μικρής εμβέλειας (short-range offset vectors) προβλέπονται επίσης για να βελτιώσουν την ακρίβεια εντοπισμού των keypoints. Για να συσχετίσει τα keypoints του κάθε ανθρώπου που εμφανίζεται στην εικόνα και να σχηματίσει την κάθε σωματική στάση, εισάγονται τα διανύσματα μετατόπισης μεσαίας εμβέλειας (mid-range offset vectors) που συνδέουν ζεύγη γειτονικών keypoints. Τέλος προτείνει έναν



Σχήμα 6: Αρχιτεκτονική PersonLab [25].

γρήγορο greedy αλγόριθμο αποκωδικοποίησης για την ομαδοποίηση των keypoints σε σωματικές στάσεις. Το PersonLab πέρα από μοντέλο για pose estimation, κάνει και instance segmentation, όμως η ανάλυση αυτής της δυνατότητας ξεπερνά τους στόχους της συγκεκριμένης διπλωματικής.

Η προτεινόμενη προσέγγιση επιδεικνύει ισχυρή απόδοση σε εργασίες ανίχνευσης ανθρώπων, και εκτίμησης σωματικής στάσης. Αξιοποιώντας τα heatmaps και τα offset vectors εντοπίζει με ακρίβεια τις αρθρώσεις, θέτοντας ένα νέο πρότυπο για αυτές τις εργασίες.

3.5 Κβαντισμός - Quantization

Ο κβαντισμός είναι μια τεχνική που βελτιστοποιεί τα μοντέλα μηχανικής μάθησης μειώνοντας το μέγεθος μνήμης τους και τις υπολογιστικές τους απαιτήσεις. Μετατρέποντας συνεχείς τιμές σε μικρότερες αναπαραστάσεις ακεραίων, όπως ακέραιοι 8-bit αριθμοί, το quantization βελτιώνει την ταχύτητα απόκρισης και την

αποτελεσματικότητα. Αυτό είναι ιδιαίτερα πολύτιμο για εφαρμογές όπου οι αποφάσεις σε πραγματικό χρόνο είναι ζωτικής σημασίας, όπως η ανίχνευση πτώσης. Ωστόσο, το quantization κρύβει και ορισμένους κινδύνους. Τα χαμηλότερα όρια ακρίβειας για τις τιμές των παραμέτρων μπορεί να οδηγήσουν σε ελαφρά μείωση της ακρίβειας του μοντέλου. Οι εξελίξεις, βέβαια, σε τεχνικές όπως η εκπαίδευση με επίγνωση κβαντισμού (quantization-aware training) μπορούν να περιορίσουν τις απώλειες. Συνολικά, ο κβαντισμός έχει γίνει ένα εργαλείο για την βελτιστοποίηση των μοντέλων μηχανικής μάθησης, και ζωτικό μάλιστα, στο πλαίσιο των IoT συσκευών.

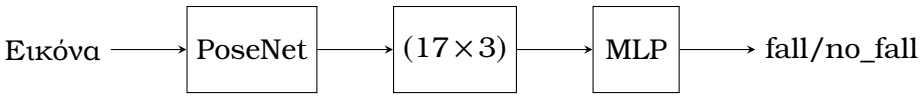
4 Μεθοδολογία

Στην παρούσα ενότητα ορίζεται το πρόβλημα της πτώσης και αναλύονται οι μέθοδοι που ακολουθήθηκαν για την επίλυση του. Περιγράφεται η θεμελιώδης αρχιτεκτονική των μοντέλων ανίχνευσης πτώσης που εκπαιδεύτηκαν, ορίζονται οι μετρικές που χρησιμοποιήθηκαν για την αξιολόγηση των μοντέλων και παρουσιάζονται το σύνολο δεδομένων και τα εργαλεία που αξιοποιήθηκαν.

4.1 Ορισμός Πτώσης

Τα γνωρίσματα μιας πτώσης μπορεί να ποικίλουν δραστικά ανάλογα με διάφορους παράγοντες. Για παράδειγμα, οι πτώσεις ενός ανθρώπου που περπατάει, στέκεται, κοιμάται ή κάθεται σε μια καρέκλα, έχουν σημαντικές διαφορές μεταξύ τους, αλλά και εμφανίζουν ορισμένα κοινά χαρακτηριστικά. Στην έρευνα των Bendary et al. [3] οι τύποι των πτώσεων ομαδοποιήθηκαν σε τρεις βασικές κατηγορίες, την εμπρόσθια πτώση, την πλάγια πτώση και την πτώση προς τα πίσω, ενώ στην έρευνα των Putra et al. [27] γίνεται μια ευρύτερη κατηγοριοποίηση, δηλαδή την εμπρόσθια, την προς τα πίσω, την προς τα αριστερά, την προς τα δεξιά, την τυφλή εμπρόσθια και την τυφλή προς τα πίσω πτώση. Ακολουθώντας μια άλλη οπτική, οι Noury et al. [24] κατηγοριοποίησαν το συμβάν της πτώσης σε τέσσερις φάσεις: την φάση πριν την πτώση (pre-fall phase), την κρίσιμη φάση (critical phase), την φάση μετά την πτώση (post-fall phase) και την φάση αποκατάστασης (recovery phase). Στην pre-fall φάση ένας άνθρωπος κάνει κανονικές δραστηριότητες, για παράδειγμα, περπατάει, κάθεται, σκύβει. Η κρίσιμη φάση περιλαμβάνει την αρχή της πτώσης, την κατάσταση στην οποία το σώμα κατευθύνεται προς μια χαμηλή επιφάνεια και το τέλος της πτώσης, όπου το σώμα προσκρούει στην επιφάνεια. Η περίοδος αδράνειας μετά την πτώση αναγνωρίζεται ως post-fall φάση, και τέλος, η φάση αποκατάστασης αφορά τις καταστάσεις όπου ο άνθρωπος σηκώνεται μετά το post-fall phase.

Σκοπός του μοντέλου που κληθήκαμε να εκπαιδεύσουμε είναι να ανιχνεύει πτώσεις σε συσκευές περιορισμένων πόρων. Καθώς, για την χρήση ενός μοντέλου που αναγνωρίζει μεγάλο εύρος κινήσεων απαιτούνται αρκετοί υπολογιστικοί πόροι,



Σχήμα 7: Αρχιτεκτονική Προτεινόμενου Μοντέλου.

καταλήξαμε στον ορισμό του προβλήματος ως πρόβλημα binary classification με κατηγορίες 'fall' και 'no_fall'. Όλοι οι τρόποι πτώσεων όπως αναλύθηκαν από τους Berndary et al. και Putra et al. ανήκουν στην κατηγορία 'fall', ενώ ως προς τους Noury et al., οι pre-fall phase και recovery phase ανήκουν στην κατηγορία 'no_fall', και οι critical phase και post-fall phase ανήκουν στην κατηγορία 'fall'. Θα μπορούσε να ανήκει στην κατηγορία 'fall' μόνο το critical phase ή μονο το post-fall phase , ώστε το μοντέλο να διακρίνει την κατάσταση πτώσης από υπόλοιπες ενέργειες, ή ώστε το μοντέλο να επικεντρώνεται στην κατάσταση αδράνειας. Αντιθέτως επιλέχθηκαν και οι δύο, ώστε το μοντέλο να είναι ικανό να αποτυπώσει τη συνολική επίδραση μιας πτώσης.

4.2 Αρχιτεκτονική Μοντέλου

Σε αυτήν την εργασία, προτείνουμε ένα μοντέλο ανίχνευσης πτώσης σε πραγματικό χρόνο για συσκευές IoT περιορισμένων υπολογιστικών πόρων, που ακολουθεί τα εξής στάδια: Η εικόνα που λαμβάνεται από την κάμερα του μικροελεγκτή προωθείται σε ένα μοντέλο εκτίμησης σωματικής στάσης. Τα αποτελέσματα στην συνέχεια, δηλαδή οι συντεταγμένες των αρθρώσεων και οι πιθανότητες ύπαρξης αυτών, γίνονται inputs ενός MLP. Αυτό λειτουργεί ως ταξινομητής και δίνει αποτελέσματα 'fall' ή 'no_fall'. Μια αναπαράσταση του μοντέλου δίνεται στο σχήμα 7.

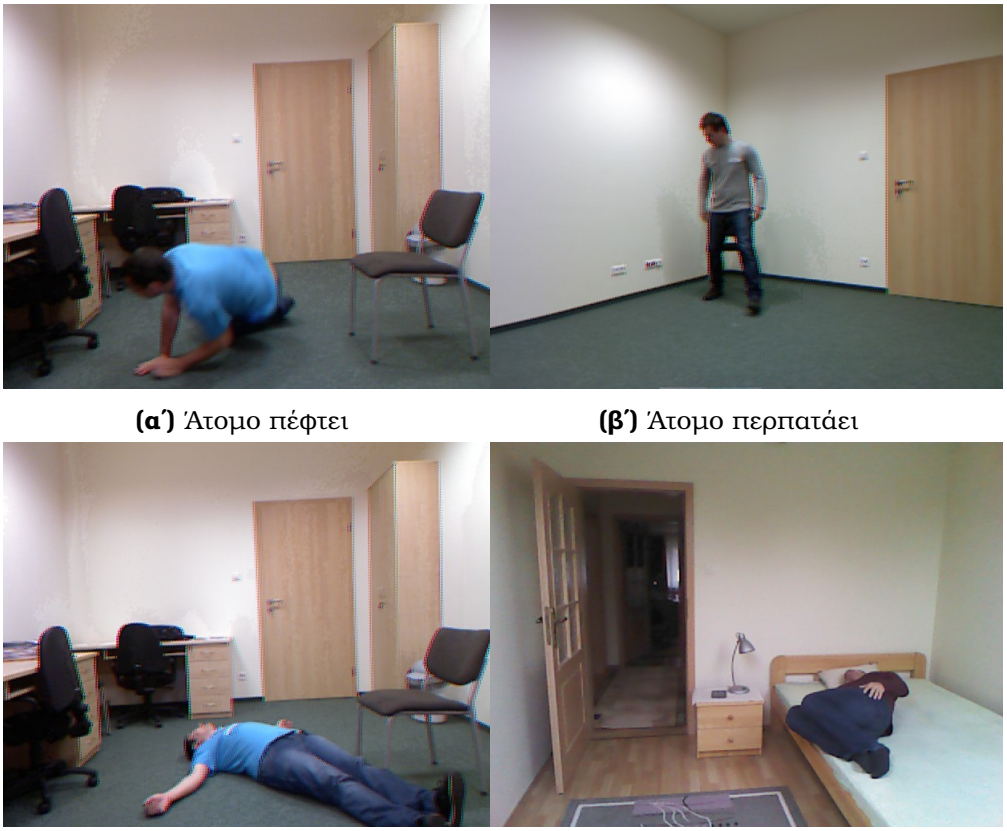
Το μοντέλο που επιλέχθηκε για την εκτίμηση της σωματικής στάσης είναι το PoseNet [43]. Το PoseNet, ανεπτυγμένο από την Google, είναι ένα μοντέλο machine learning που επιτρέπει την εκτίμηση σωματικής στάσης ενός ή πολλαπλών ατόμων σε browsers. Είναι μια υλοποίηση του PersonLab ανεπτυγμένη σε TensorFlow.js [43] και μπορεί να χρησιμοποιήσει για CNN ένα MobileNetV1 ή ένα ResNet [12]. Το αποτέλεσμα που δίνει το PoseNet κατά την εφαρμογή του είναι 17 keypoints αρθρώσεων και μια πιθανότητα για κάθε άρθρωση, δηλαδή έναν τανυστή $(17, 3)$. Το PoseNet μπορεί να βρεθεί διαθέσιμο στην ιστοσελίδα TFHub [44]. Για την ανάπτυξη των μοντέλων, επιλέξαμε την εκδοχή με το MobileNetV1, διότι, όπως αναλύσαμε στην ενότητα 3.3.5, προσφέρει στο μοντέλο την

δυνατότητα να απαιτεί λιγότερους υπολογιστικούς πόρους, καθιστώντας το πιο αποδοτικό και ελαφρύ. Ακόμη, περιοριστήκαμε στην εκτίμηση σωματικής στάσης ενός ατόμου για να μην επιβαρύνεται η συσκευή, όπως είναι λογικό να συμβαίνει στο multiple pose estimation. Τα width multiplier που επιλέχθηκαν ήταν 0.75 και 0.50, ενώ οι αναλύσεις των inputs διαλέξαμε να είναι 192×192, 224×224 και 256×256.

4.3 Σύνολο Δεδομένων

Οι μικροελεγκτές, λόγω της χαμηλού μεγέθους μνήμης RAM που διαθέτουν, αδυνατούν να χρησιμοποιήσουν εικόνες υψηλής ανάλυσης. Συνεπώς για τις ανάγκες της διπλωματικής, απαιτήθηκε να χρησιμοποιηθεί ένα σύνολο δεδομένων με εικόνες, όσο το δυνατόν μικρότερης διάστασης. Το σύνολο δεδομένων που επιλέχθηκε είναι το UR Fall Detection Dataset [20]. Αυτό το dataset περιέχει 70 ακολουθίες, 30 ακολουθίες πτώσεων και 40 ακολουθίες από δραστηριότητες της καθημερινής ζωής (active daily life - ADL). Τα βίντεο καταγράφηκαν με δύο Microsoft Kinect κάμερες, η πρώτη δίνει μια πλάγια όψη του δωματίου και η δεύτερη τοποθετήθηκε στην οροφή δίνοντας μια όψη από ψηλά. Το dataset περιέχει RGB εικόνες, εικόνες βάθους (depth images) καθώς και δεδομένα από επιταχυνσιόμετρο. Κάποιες εικόνες του dataset παρουσιάζονται στο σχήμα 8.

Για τους στόχους της διπλωματικής, χρησιμοποιήθηκαν οι RGB εικόνες της πρώτης κάμερας, μεγέθους 640×480, ενώ από αυτές αφαιρέθηκαν εικόνες που δεν περιείχαν ανθρώπους ή σχεδόν όλο το σώμα δεν φαινόταν. Οι δημιουργοί του dataset κατηγοριοποίησαν τις εικόνες των ακολουθιών πτώσης σε εικόνες που ο άνθρωπος δεν ξαπλώνει, σε εικόνες που ξαπλώνει και σε εικόνες όπου το άτομο πέφτει ή έχει στάση σώματος που θυμίζει πτώση, με συμβολισμούς '-1', '1' και '0' αντίστοιχα. Έτσι με βάση τον ορισμό της πτώσης και μη πτώσης που δόθηκε προηγουμένως, η κατηγορία 'fall' περιέχει τις '1' και '0', ενώ η κατηγορία 'no_fall' περιέχει την '-1'. Ακόμη, απορρίφθηκαν εικόνες από τις ακολουθίες ADL που δείχνουν άτομα να ξαπλώνουν και εικόνες πριν ξαπλώσουν, όπως στην εικόνα 8δ΄. Αυτό έγινε ώστε η διαφορά μεταξύ των δύο κατηγοριών να είναι διακριτή, και να μην προκαλείται σύγχυση στο μοντέλο κατά την εκπαίδευση.



(α΄) Άτομο πέφτει **(β΄)** Άτομο περπατάει

(γ΄) Άτομο μετά από πτώση **(δ΄)** Άτομο ξαπλώνει

Σχήμα 8: Εικόνες από το UR Fall Detection Dataset.

4.4 Μετρικές Αξιολόγησης

Από την οπτική της δυαδικής ταξινόμησης το σύστημα πρέπει να αποφασίσει αν στην εικόνα που δέχεται ως είσοδο εμφανίζεται πτώση ατόμου ή όχι. Για να αξιολογήσουμε τα μοντέλα χαρακτηρίζουμε τις προβλέψεις τους ως:

- Αληθώς θετική - TP (True Positive): 'fall' εντοπίστηκε ως 'fall'
- Ψευδώς θετική - FP (False Positive): 'fall' εντοπίστηκε ως 'no_fall'
- Αληθώς αρνητική - TN (True Negative): 'no_fall' εντοπίστηκε ως 'no_fall'
- Ψευδώς αρνητική - FN (False Negative): 'no_fall' εντοπίστηκε ως 'fall'

Με αυτούς τους όρους μπορούμε να ορίσουμε τις παρακάτω μετρικές αξιολόγησης:

- Ορθότητα (Accuracy): αναφέρεται στο ποσοστό των σωστών προβλέψεων που έκανε το μοντέλο συνολικά σε σχέση με όλες τις προβλέψεις που έκανε.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FP} \tag{19}$$

- Ακρίβεια (Precision): αναφέρεται στο ποσοστό των προβλέψεων πτώσης που πραγματικά ήταν πτώσεις σε σχέση με όλες τις προβλέψεις πτώσης που έκανε το μοντέλο. Δείχνει πόσο καλά το μοντέλο αναγνώρισε τις πραγματικές πτώσεις.

$$Precision = \frac{TP}{TP + FP} \tag{20}$$

- Ανάκληση ή Ευαισθησία(Recall/Sensitivity): αναφέρεται στο ποσοστό των πραγματικών πτώσεων που το μοντέλο κατάφερε να ανιχνεύσει.

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

- Τιμή F1 (F1 Score): είναι ο αρμονικός μέσος της ακρίβειας και της ανάκλησης του μοντέλου. Συνδυάζει συμμετρικά τις δυο προηγούμενες μετρικές σε μια. Η υψηλότερη δυνατή τιμή είναι 1 και υποδεικνύει τέλεια ακρίβεια και ανάκληση, ενώ

η χαμηλότερη δυνατή τιμή είναι 0, εάν είτε η ακρίβεια είτε η ανάκληση είναι μηδέν. Χρησιμοποιείται για να βαθμολογήσει τη συνολική απόδοση του μοντέλου ως προς τις πτώσεις.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{22}$$

4.5 Εργαλεία

Οι πειραματισμοί και οι εκπαιδεύσεις των μοντέλων πραγματοποιήθηκαν σε περιβάλλον Windows 10, σε συνδυασμό με την γλώσσα προγραμματισμού Python έκδοσης 3.10. Η βιβλιοθήκη TensorFlow 2.10 χρησιμοποιήθηκε για την υλοποίηση των μοντέλων. Η εκπαίδευση έγινε σε desktop PC με GPU NVIDIA GTX 1060 6GB, CPU AMD Ryzen 5 2400G 3.6 GHz και 24GB μνήμη RAM. Για τις διεργασίες επιτάχυνσης, χρησιμοποιήθηκε το NVIDIA CUDA Toolkit 11.2 και το cuDNN 8.1. Τα Visual Studio Code και Jupyter Colab χρησιμοποιήθηκαν για την ανάπτυξη και την εκτέλεση του κώδικα.

4.5.1 Tensorflow

Το TensorFlow [42] είναι ένα framework μηχανικής μάθησης ανοιχτού κώδικα που αναπτύχθηκε από την Google και δίνει τη δυνατότητα σε ερευνητές και προγραμματιστές να δημιουργήσουν και να αναπτύξουν αποτελεσματικά διάφορα μοντέλα μηχανικής μάθησης. Προσφέρει ένα ολοκληρωμένο οικοσύστημα εργαλείων, βιβλιοθηκών και κοινοτικών πόρων που διευκολύνουν την ανάπτυξη νευρωνικών δικτύων και άλλων αλγορίθμων μηχανικής μάθησης. Το TensorFlow παρέχει μια ευέλικτη πλατφόρμα για τη δημιουργία, την εκπαίδευση και την ανάπτυξη μοντέλων σε διαφορετικές πλατφόρμες hardware, συμπεριλαμβανομένων των CPU, των GPU και των TPU. Το πλούσιο σύνολο προκατασκευασμένων επιπέδων, μοντέλων και τεχνικών βελτιστοποίησης απλοποιεί τη διαδικασία ανάπτυξης, ενώ η επεκτασιμότητα και η συμβατότητά του επιτρέπουν την απρόσκοπτη μετάβαση από την έρευνα στην παραγωγή.

Το TensorFlow Lite είναι μια εξειδικευμένη έκδοση του TensorFlow που έχει σχεδιαστεί για την ανάπτυξη μοντέλων machine learning σε συσκευές με

περιορισμένους πόρους, όπως κινητά τηλέφωνα, συσκευές IoT και ενσωματωμένα συστήματα. Επικεντρώνεται στη βελτιστοποίηση του inference process, επιτρέποντας στα μοντέλα να εκτελούνται αποτελεσματικά ακόμη και σε συσκευές με περιορισμένη υπολογιστική ισχύ και μνήμη. Το TensorFlow Lite παρέχει εργαλεία για μετατροπή, κβαντισμό και βελτιστοποίηση μοντέλων, επιτρέποντας σε πολύπλοκα μοντέλα να μετατρέπονται σε μορφές που ευνοούν την εκτέλεση σε πραγματικό χρόνο, στη συσκευή. Αυτή η τεχνολογία δίνει τη δυνατότητα στους προγραμματιστές να φέρουν τις δυνατότητες μηχανικής μάθησης σε edge συσκευές, επιτρέποντας εφαρμογές που κυμαίνονται από image recognition και επεξεργασία φυσικής γλώσσας έως διάφορες εργασίες που βασίζονται σε αισθητήρες, χωρίς να εξαρτώνται σε μεγάλο βαθμό από το cloud.

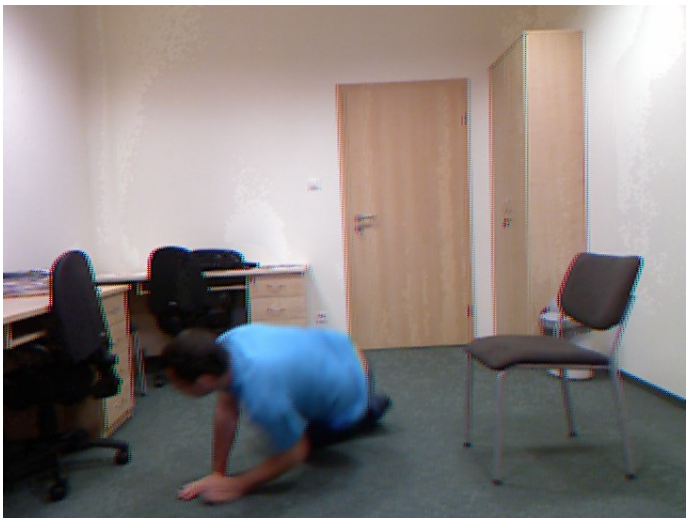
Το TensorFlow for Microcontrollers οδηγεί την ιδέα της ανάπτυξης μοντέλων μηχανικής εκμάθησης στα άκρα στοχεύοντας συσκευές εξαιρετικά περιορισμένων πόρων, όπως μικροελεγκτές. Αυτές οι μικροσκοπικές συσκευές έχουν συχνά περιορισμένη μνήμη, ισχύ επεξεργασίας και προϋπολογισμό ενέργειας. Έχει σχεδιαστεί για να καλύπτει αυτούς τους περιορισμούς, διατηρώντας παράλληλα τη δυνατότητα εκτέλεσης μοντέλων απευθείας σε τέτοιες συσκευές. Προσφέρει εξαιρετικά βελτιστοποιημένους πυρήνες και εργαλεία που επιτρέπουν τη μεταγλώττιση και την εκτέλεση μοντέλων εντός των μικροελεγκτών. Αυτό ανοίγει νέες δυνατότητες για εφαρμογές όπως η αναγνώριση χειρονομιών, η ανίχνευση ανωμαλιών και η παρακολούθηση της υγείας απευθείας στο επίπεδο της συσκευής. Το TensorFlow for Microcontrollers ενσαρκώνει την ιδέα να φέρει τη τεχνητή νοημοσύνη στην άκρη του δικτύου, επιτρέποντας στις συσκευές να λαμβάνουν έξυπνες αποφάσεις αυτόνομα χωρίς να βασίζονται σε servers.

Για τους σκοπούς της εργασίας επιστρατεύσαμε το TensorFlow για την ανάπτυξη προγνωστικών μοντέλων προσαρμοσμένα για την ανίχνευση πτώσεων. Εκμετελλεύοντας την ευελιξία που προσφέρει, τα μοντέλα μετατράπηκαν σε TensorFlow Lite format και εφαρμόστηκε η τεχνική quantization για την μείωση των μεγεθών τους και, τελικά, για την προετοιμασία τους για το deployment σε συσκευές περιορισμένων πόρων.

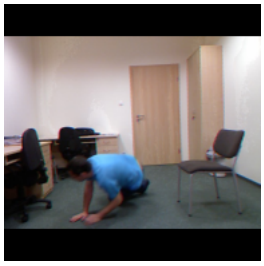
Tfjs-to-tf. Όπως προηγουμένως αναφέρθηκε, το PoseNet έχει format TensorFlow.js, μια εκδοχή του TensorFlow σε Javascript που διευκολύνει την χρήση μοντέλων machine learning σε web browsers, αλλά και σε servers, μέσω του Node.js. Τα μοντέλα μπορεί, είτε να κατασκευαστούν και να εκπαιδευτούν απευθείας σε Javascript με την βιβλιοθήκη TensorFlow.js, είτε να εκπαιδευτούν σε Python με την βιβλιοθήκη TensorFlow και στην συνέχεια να μετατραπούν σε format TensorFlow.js. Ωστόσο, η Tensorflow δεν παρέχει εργαλεία για την αντίστροφη μετατροπή (από Javascript σε Python). Προκειμένου να μετατραπεί το PoseNet, σε TensorFlow, εκμεταλλευτήκαμε τη βιβλιοθήκη TensorFlow.js Graph Model Converter (tfjs-to-tf), διαθέσιμη στο Github [45].

4.5.2 OpenCV

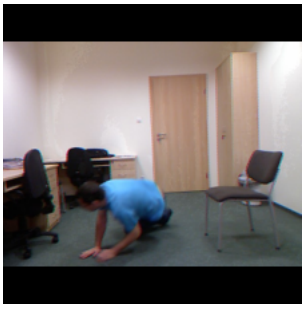
Το OpenCV (Open Source Computer Vision Library) [46] είναι μια ανοιχτού κώδικα βιβλιοθήκη υπολογιστικής όρασης που αναπτύχθηκε αρχικά από την Intel και συνεχίζεται από την open-source κοινότητα. Η βιβλιοθήκη αυτή παρέχει ένα ευρύ φάσμα λειτουργιών για επεξεργασία εικόνων και βίντεο, καθώς και για αναγνώριση προτύπων και ανάλυση υπολογιστικής όρασης. Η βιβλιοθήκη χρησιμοποιήθηκε για την αλλαγή της ανάλυσης των εικόνων του συνόλου δεδομένων. Από την αρχική τους ανάλυση 640×480, συρρικνώθηκαν σε 192×192, 224×224, και 256×256. Ακόμη, επειδή η αλλαγή ανάλυσης δεν ήταν άμεση, προστέθηκαν ορισμένα pixel για να συμπληρωθούν τα απαιτούμενα μεγέθη. Ένα παράδειγμα με τις αλλαγές εμφανίζεται στο σχήμα 9. Ο κώδικας είναι διαθέσιμος στο παράρτημα Α΄.



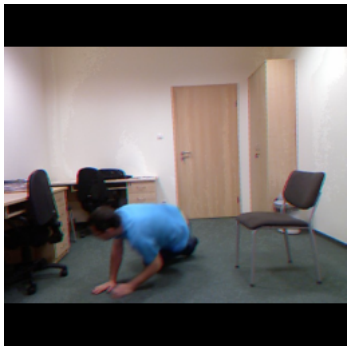
(α΄) Εικόνα χωρίς επεξεργασία 640×480



(β΄) Εικόνα 192×192



(γ΄) Εικόνα 224×224



(δ΄) Εικόνα 256×256

Σχήμα 9: Επεξεργασία εικόνων.

5 Εκπαίδευση - Αποτελέσματα

Στο πέμπτο κεφάλαιο αναλύονται εκτενώς τα βήματα που πάρθηκαν για την εκπαίδευση των μοντέλων. Ακόμη, παρατίθενται τα αποτελέσματα των μετρικών αξιολόγησης για τα μοντέλα πριν και μετά τον κβαντισμό.

5.1 Εκπαίδευση

Σύμφωνα με όσα αναλύσαμε στην ενότητα 4.2, η αρχιτεκτονική του μοντέλου ανίχνευσης πτώσης περιέχει τον αλγόριθμο pose estimation PoseNet και έναν MLP, που λειτουργεί ως ταξινομητής των 51 (17×3) εξόδων του PoseNet στις κατηγορίες 'fall' και 'no_fall'. Σχεδιάστηκαν δυο MLP, έναν (normal) με τέσσερα hidden layers και έναν (wide) με πέντε. Η είσοδος που δέχονται είναι ένας τανυστής μεγέθους (17×3) (η έξοδος του Posenet) στην οποία εφαρμόζουν flattening. Οι 51 τιμές διέρχονται από τα hidden layers ώστε τελικά να ταξινομηθούν σε 'fall' ή 'no_fall'. Στους πίνακες 4 και 5 παρουσιάζονται οι αρχιτεκτονικές των MLP. Επομένως, δημιουργήθηκαν και εκπαιδεύτηκαν δυο μοντέλα MLP για κάθε εκδοχή του PoseNet (0.50, 0.75), για κάθε ανάλυση (192×192, 224×224, 256×256), συνολικά 12 μοντέλα.

Πίνακας 4: Αρχιτεκτονική normal MLP.

Input 17×3			
Layer	# Neurons	Dropout rate	Activation function
Flatten	-	-	-
Dense	64	0.5	ReLU
Dense	32	0.5	ReLU
Dense	16	0.25	ReLU
Dense	1	-	Sigmoid

Το κάθε σύνολο δεδομένων (7,968 δείγματα) μετά την μετατροπή της ανάλυσης χωρίστηκε, αναθέτοντας το 70% για training (5,578 δείγματα), το 10% για validation (790 δείγματα), και το 20% για testing(1,600 δείγματα). Τα training και validation χρησιμοποιήθηκαν για εκπαίδευση και αξιολόγηση κατά την εκπαίδευση αντίστοιχα, ενώ το testing dataset για την εξέταση των μοντέλων σε ξένα δείγματα. Για να

Πίνακας 5: Αρχιτεκτονική wide MLP.

Input 17×3			
Layer	# Neurons	Dropout rate	Activation function
Flatten	-	-	-
Dense	64	0.5	ReLU
Dense	128	0.5	ReLU
Dense	32	0.5	ReLU
Dense	16	0.25	ReLU
Dense	1	-	Sigmoid

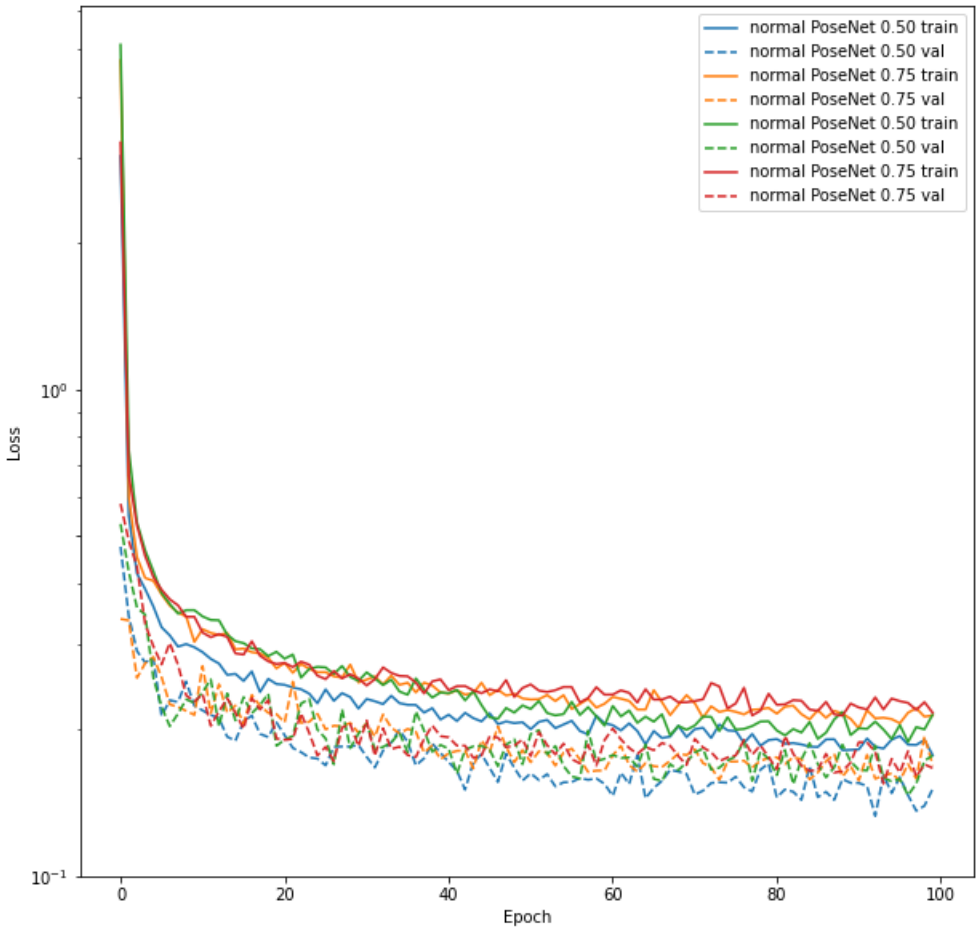
εξοικονομηθεί χρόνος κατά την εκπαίδευση, το PoseNet εφαρμόστηκε στα δείγματα και εκπαιδεύτηκαν μόνο τα MLP.

Κατά την εκπαίδευση των MLP επιλέχθηκε να εφαρμοστεί η τεχνική Dropout στα hidden layers. Αυτή η τεχνική θέτει τυχαία τις εισόδους των νευρώνων του εκάστοτε layer ως 0 με μια συχνότητα dropout_rate, για να αποτραπεί το overfitting. Η εκπαίδευση του κάθε μοντέλου έγινε σε 100 εποχές (epochs), με 16 batches οι normal και με 32 batches οι wide. Για όλα τα μοντέλα χρησιμοποιήθηκε ο Adam optimizer, με παραμέτρους $a = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, και η συνάρτηση κόστους binary cross-entropy loss. Ο κώδικας της εκπαίδευσης είναι διαθέσιμος στο παράρτημα Α΄.

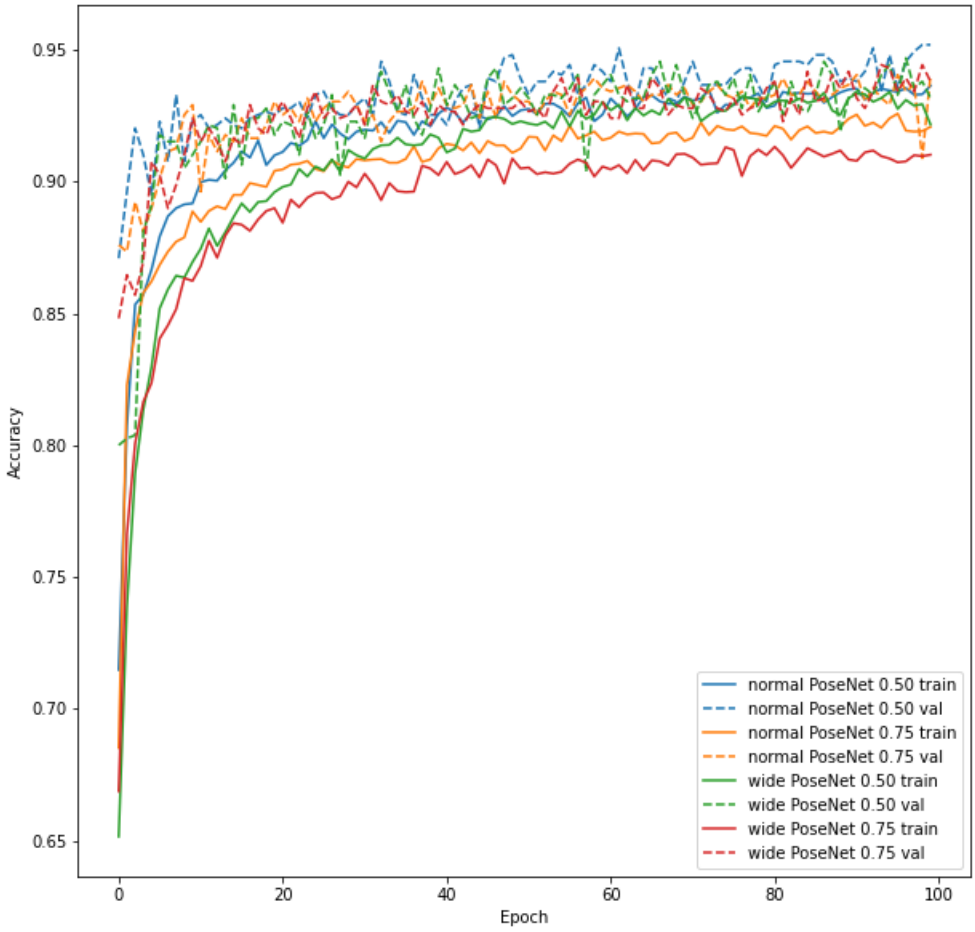
Παρατηρούμε από τα σχήματα 10, 12 και 14 ότι οι συναρτήσεις κόστους των μοντέλων για τα training και validation δεδομένα αναπτύσσονται ομοιόμορφα, γεγονός που αποδεικνύει ότι δεν υπάρχει overfitting κατά την εκπαίδευση. Ακόμη, από τα σχήματα 11, 13 και 15, συμπεραίνουμε ότι στις τελευταίες εποχές, τα μοντέλα δεν βελτιώθηκαν σημαντικά, και συνεπώς περαιτέρω εκπαίδευση δεν χρειάζεται, καθώς μπορεί να οδηγήσει σε overfitting.

Να σημειώσουμε ότι τα μοντέλα wide MLP για τα PoseNet 0.50 και 0.75 για την ανάλυση 256×256 απορρίφθηκαν ύστερα από την εκπαίδευση, και για αυτό δεν εμφανίζονται στα σχήματα, ούτε στα αποτελέσματα, που παρουσιάζονται στην επόμενη ενότητα. Ο λόγος είναι διότι από τα πρώτα epochs οι συναρτήσεις κόστους αυξάνονταν απότομα και αδυνατούσαν να βελτιωθούν. Τα accuracies παρέμεναν σταθερά (περίπου 0.785) ενώ τα recalls και precisions παρέμεναν κοντά στο 0.

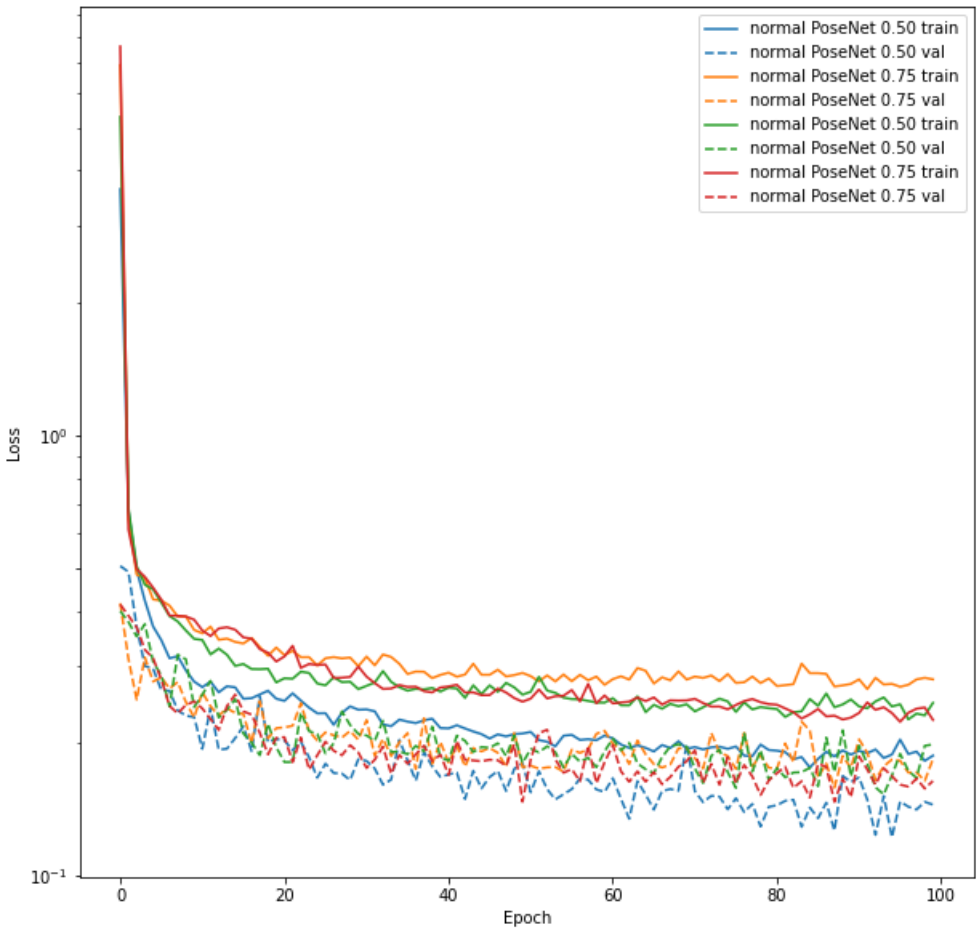
Ύστερα από το training, τα μοντέλα MLP προστέθηκαν στα μοντέλα PoseNet που



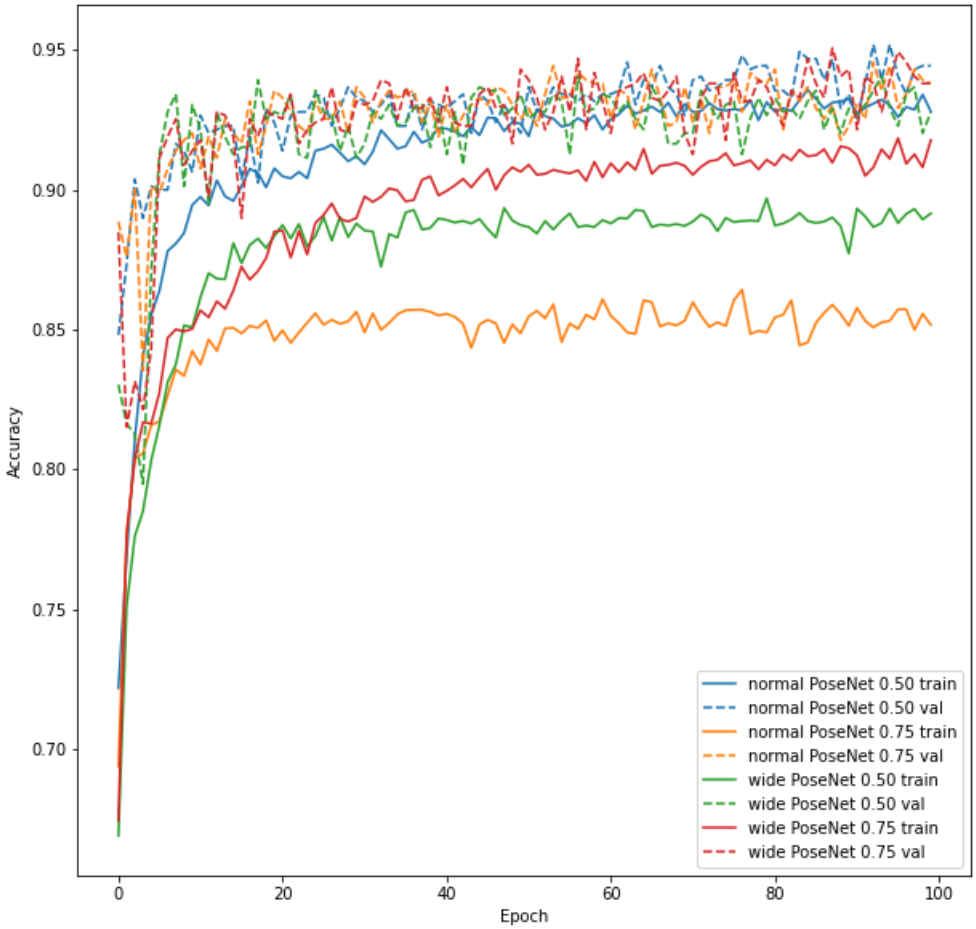
Σχήμα 10: Συναρτήσεις κόστους των μοντέλων για ανάλυση 192×192.



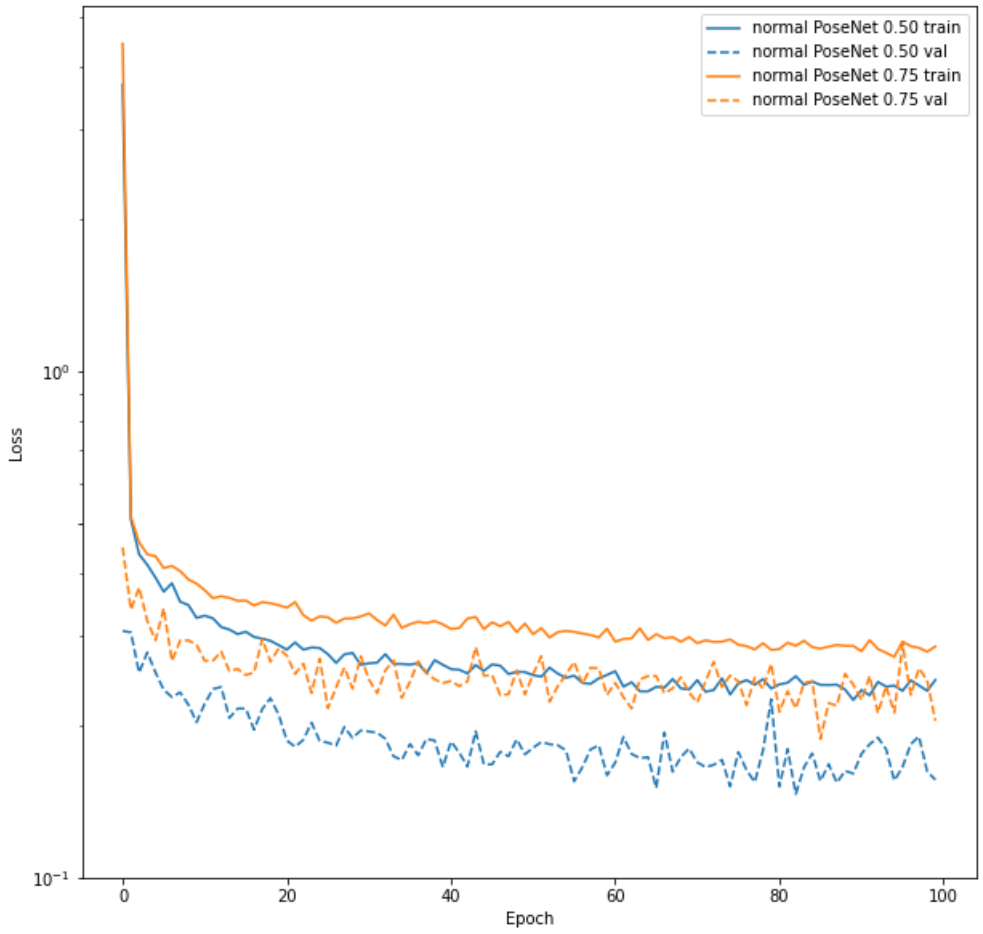
Σχήμα 11: Accuracy των μοντέλων για ανάλυση 192×192.



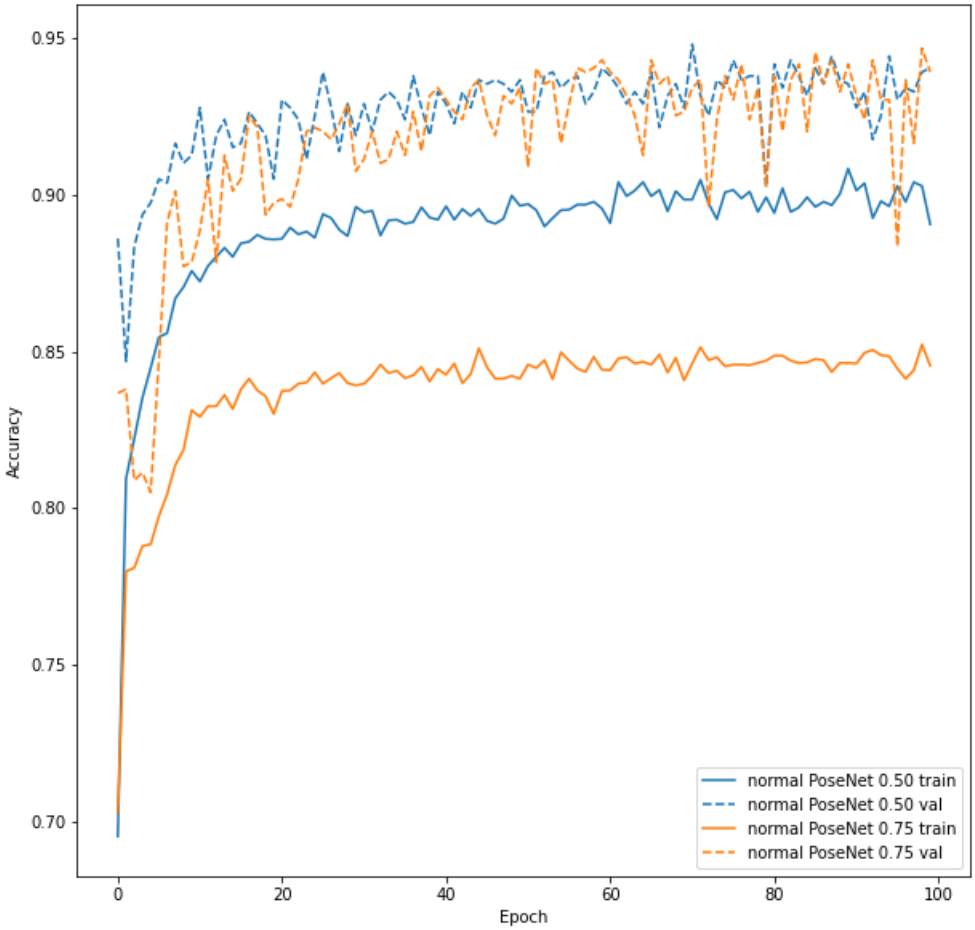
Σχήμα 12: Συναρτήσεις κόστους των μοντέλων για ανάλυση 224×224.



Σχήμα 13: Accuracy των μοντέλων για ανάλυση 224×224.



Σχήμα 14: Συναρτήσεις κόστους των μοντέλων για ανάλυση 256×256.



Σχήμα 15: Accuracy των μοντέλων για ανάλυση 256×256.

τους αναλογούσαν, και μετατράπηκαν σε μοντέλα Tensorflow Lite. Στην συνέχεια εφαρμόστηκε full integer quantization στα μοντέλα για να μειωθεί το μέγεθος τους. Ο κώδικας είναι διαθέσιμος στο παράρτημα Α΄.

5.2 Αποτελέσματα

Τα αποτελέσματα των μοντέλων πριν το quantization, ως προς τις μετρικές που ορίσαμε στην ενότητα 4.4, βρίσκονται στον πίνακα 6. Όλα τα μοντέλα εμφανίζουν ικανοποιητικά accuracy και precision, όμως οι μετρικές recall είναι μέτριες. Από τα precision και recall συμπεραίνουμε ότι τα μοντέλα δεν ανιχνεύουν με τρομερή επιτυχία πτώσεις, αλλά όταν προβλέπουν ότι μια ενέργεια αντιστοιχεί σε πτώση, η πιθανότητα να είναι όντως πτώση είναι αρκετά μεγάλη. Τα πιο ικανοποιητικά αποτελέσματα παρουσιάζει το normal MLP σε συνδυασμό με PoseNet 0.50 σε ανάλυση 192×192, ύστερα το normal MLP με PoseNet 0.50 σε ανάλυση 224×224 και ακολουθούν το normal MLP με PoseNet 0.75 σε ανάλυση 224×224, το normal MLP με PoseNet 0.50 σε ανάλυση 256×256, και το normal MLP σε συνδυασμό με PoseNet 0.75 σε ανάλυση 192×192. Τα wide MLP παρουσίασαν τα χειρότερα αποτελέσματα, γεγονός που μας φέρνει στο συμπέρασμα ότι το παραπάνω layer υπεραναλύει τα inputs σε σημείο που το μοντέλο χάνει πολύτιμες πληροφορίες.

Πίνακας 6: Αποτελέσματα αξιολόγησης μοντέλων πριν το quantization.

Resolution	PoseNet multiplier	MLP type	Accuracy	Precision	Recall	F1 Score	Size
192×192	0.50	normal	94.7%	90.7%	83.5%	87.0%	2.49 MB
		wide	92.6%	95.6%	68.9%	80.0%	2.50 MB
	0.75	normal	93.4%	97.6%	70.8%	82.0%	5.12 MB
		wide	93.1%	98.0%	69.5%	81.3%	5.13 MB
224×224	0.50	normal	94.6%	96.0%	77.6%	85.8%	2.49 MB
		wide	91.7%	96.9%	63.4%	76.6%	2.50 MB
	0.75	normal	93.9%	99.2%	71.8%	83.3%	5.12 MB
		wide	93.0%	98.7%	68.0%	80.5%	5.13 MB
256×256	0.50	normal	93.2%	90.8%	75.6%	82.5%	2.49 MB
	0.75	normal	93.6%	99.1%	67.9%	80.6%	5.12 MB

Στον πίνακα 7 παρουσιάζονται τα αποτελέσματα της αξιολόγησης των μοντέλων μετά το quantization. Τα μοντέλα χάνουν κατά μέσο όρο 4 ποσοστιαίες μονάδες από το accuracy,

12.2 μονάδες από το precision, 9.7 μονάδες από το recall και 10.8 μονάδες από το f1 score. Πιο αποδοτικά παραμένουν το normal MLP σε συνδυασμό με PoseNet 0.50 σε ανάλυση 192×192 και το normal MLP σε συνδυασμό με PoseNet 0.50 σε ανάλυση 224×224. Όσον αφορά τα μεγέθη, τα μοντέλα μετά τον κβαντισμό είναι έως και 3.5 φορές μικρότερα από τα αρχικά. Τα μικρότερα είναι τα μοντέλα που χρησιμοποιούν το PoseNet 0.50.

Πίνακας 7: Αποτελέσματα αξιολόγησης μοντέλων μετά το quantization.

Resolution	PoseNet multiplier	MLP type	Accuracy	Precision	Recall	F1 Score	Size
192×192	0.50	normal	90.5%	82.1%	70.7%	75.9%	708 KB
		wide	87.5%	79.1%	57.3%	66.5%	719 KB
	0.75	normal	87.0%	74.2%	59.1%	65.8%	1,427 KB
		wide	88.5%	82.0%	59.0%	68.6%	1,438 KB
224×224	0.50	normal	90.1%	86.4%	63.6%	72.7%	708 KB
		wide	87.2%	83.7%	52.2%	64.4%	719 KB
	0.75	normal	88.9%	82.7%	60.1%	69.6%	1,427 KB
		wide	87.2%	77.2%	56.7%	65.4%	1,438 KB
256×256	0.50	normal	89.2%	82.8%	63.3%	71.65%	708 KB
	0.75	normal	89.2%	86.2%	58.9%	69.9%	1,427 KB

5.3 Προβλήματα κατά την Υλοποίηση

Κατά την μετατροπή του PoseNet από TensorFlow.js σε TensorFlow, συνειδητοποιήσαμε ότι τα αρχεία που πήραμε από το TFHub [44] ήταν corrupted. Αντ'αυτού, πήραμε τα αρχεία από ένα XML αρχείο που περιέχει αρχεία για παραδείγματα των μοντέλων του TensorFlow[41]. Μετά την μετατροπή παρατηρήθηκε ότι τα αποτελέσματα αυτού του PoseNet ήταν τα heatmaps και offset vectors προτού την αποκωδικοποίηση. Έτσι γράψαμε δικό μας κώδικα για αυτήν την διαδικασία, ο οποίος βρίσκεται στο παράρτημα Α'. Ακόμη υπήρξε η επιθυμία αξιολόγησης του μοντέλου σε ένα ακόμα σύνολο δεδομένων, το Multiple cameras fall dataset [2], όμως κατά την εφαρμογή του PoseNet σε αυτά, διαπιστώθηκε ότι ο αλγόριθμος pose estimation αδυνατούσε να κάνει προβλέψεις, και έτσι δεν χρησιμοποιήθηκε το σύνολο δεδομένων. Επίσης, περιοριστήκαμε μόνο στον MLP ως classifier, διότι, τα αποτελέσματα των διάφορων πειραματισμών που έγιναν με κλασικούς αλγορίθμους μηχανικής μάθησης,

δηλαδή τους Random Forest, Support Vector Machines, k-Nearest Neighbors, ήταν άκρως απογοητευτικά και σε καμία των περιπτώσεων δεν μπορούσαν να ανταγωνιστούν τα αποτελέσματα των MLP.

6 Συμπεράσματα

6.1 Συμπεράσματα

Το διάχυτο ζήτημα των πτώσεων, ιδιαίτερα μεταξύ των ηλικιωμένων και των ατόμων με ειδικές ανάγκες, δεν μπορεί να υποτιμηθεί. Οι επιζήμιες συνέπειες των πτώσεων στην υγεία, την ποιότητα ζωής και η πίεση που επιβάλλουν στα συστήματα υγειονομικής περίθαλψης και τους φροντιστές υπογραμμίζουν την επείγοντα ανάγκη για αποτελεσματικές λύσεις ανίχνευσης πτώσης. Οι πτώσεις αποτελούν σημαντική πρόκληση για τη δημόσια υγεία, συμβάλλοντας σε υψηλότερα ποσοστά νοσηρότητας και θνησιμότητας, παρατεταμένες νοσηλεύσεις και συναισθηματική πίεση τόσο στα άτομα όσο και στις οικογένειές τους. Ως εκ τούτου, η ανάπτυξη και η εφαρμογή ισχυρών συστημάτων ανίχνευσης πτώσης είναι επιτακτική για τον μετριασμό αυτών των προκλήσεων και τη βελτίωση της ευημερίας όσων κινδυνεύουν. Η αξιοποίηση τεχνολογιών αιχμής, όπως το TinyML, προσφέρει μια ελπιδοφόρα πορεία για την επίλυση αυτών των προκλήσεων, καθώς ενδυναμώνει την ανάπτυξη, αποδοτικών και αξιόπιστων λύσεων ανίχνευσης πτώσης σε πραγματικό χρόνο.

Ανακεφαλαιώνοντας, στην παρούσα διπλωματική εργασία παρουσιάστηκε ένα σύστημα ανίχνευσης πτώσης χαμηλού μεγέθους. Το σύστημα βασίστηκε στο PoseNet, ένα lightweight state-of-the-art μοντέλο εκτίμησης σωματικής στάσης για την εξαγωγή χαρακτηριστικών κατά την πτώση, τα οποία ταξινομήθηκαν με έναν Multilayer Perceptron. Αναπτύχθηκαν μοντέλα για τρεις διαφορετικές αναλύσεις εικόνων, 192×192, 224×224 και 256×256, και δυο width multiplier του PoseNet, 0.5 και 0.75. Στα μοντέλα που εκπαιδεύτηκαν εφαρμόστηκε η τεχνική κβαντισμού για την μείωση των μεγεθών τους, με αποτέλεσμα να είναι μικρότερα από 1.5 MB. Το μικρότερο μοντέλο και συνάμα το πιο αποδοτικό ήταν αυτό που αξιοποίησε το PoseNet με width multiplier 0.5, εκπαιδευμένο σε ανάλυση εικόνων 192×192, παρουσιάζοντας 90.5% accuracy, 82.1% precision, 70.7% recall, 75.9% f1 score και 708 KB μέγεθος.

6.2 Μελλοντικές Επεκτάσεις

Για το κλείσιμο της εργασίας αναφέρονται κάποιες πιθανές επεκτάσεις που προέκυψαν ως σκέψεις κατά την εκπόνηση της. Για να γίνει μια αξιολόγηση σε πραγματικές συνθήκες, τα μοντέλα πρέπει να εγκατασταθούν σε μικροελεγκτές και να μετρηθεί η απόκρισή τους και η ενέργεια που καταναλώνεται. Επίσης, μετά το training θα μπορούσε να εφαρμοστεί quantization-aware training στα μοντέλα για μερικές εποχές, ώστε μετά τον κβαντισμό να μην εμφανιστούν τόσες μεγάλες μειώσεις στην απόδοσή τους. Η συμπεριφορά των μοντέλων θα μπορούσε να ερευνηθεί και με άλλα σύνολα δεδομένων και άλλους classifiers. Τέλος, αν και είναι ίσως το πιο lightweight σύστημα για pose estimation, θα μπορούσε να χρησιμοποιηθεί η τεχνική εύρεσης αρχιτεκτονικής (Neural Architecture Search) στο PoseNet για να σχεδιαστεί ένα ακόμα μικρότερο μοντέλο.

Βιβλιογραφία

[1] Alam E, Sufian A, Dutta P, Leo M. Vision-based human fall detection systems using deep learning: A review. Comput Biol Med. 2022 Jul;146:105626. doi: 10.1016/j.compbiomed.2022.105626. Epub 2022 May 27. PMID: 35680453.

[2] E. Auvinet, C. Rougier, J.Meunier, A. St-Arnaud, J. Rousseau, "Multiple cameras fall dataset", Technical report 1350, DIRO - Université de Montréal, July 2010.

[3] El-Bendary, N., Tan, Q., Pivot, F.C., Lam, A. (2013). Fall Detection and Prevention for the Elderly: A Review of Trends and Challenges. International Journal on Smart Sensing and Intelligent Systems, 6, 1230 - 1266.

[4] Z. Cao, G. Hidalgo, T. Simon, S. -E. Wei and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 172-186, 1 Jan. 2021, doi: 10.1109/TPAMI.2019.2929257.

[5] J. Clemente, F. Li, M. Valero and W. Song, "Smart Seismic Sensing for Indoor Fall Detection, Location, and Notification," in IEEE Journal of Biomedical and Health Informatics, vol. 24, no. 2, pp. 524-532, Feb. 2020, doi: 10.1109/JBHI.2019.2907498.

[6] Dutta, L., Bharali, S. (2021). TinyML Meets IoT: A Comprehensive Survey. Internet Things, 16, 100461.

[7] Faes MC, Reelick MF, Joosten-Weyn Banningh LW, Gier Md, Esselink RA, Olde Rikkert MG. Qualitative study on the impact of falling in frail older persons and family caregivers: foundations for an intervention to prevent falls. Aging Ment Health. 2010 Sep;14(7):834-42. doi: 10.1080/13607861003781825. PMID: 20635232.

[8] Fang, H., Li, J., Tang, H., Xu, C., Zhu, H., Xiu, Y., Li, Y., Lu, C. (2022). AlphaPose: Whole-Body Regional Multi-Person Pose Estimation and Tracking in Real-Time. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45, 7157-7173.

[9] H. Foroughi, B. S. Aski and H. Pourreza, "Intelligent video surveillance for monitoring fall detection of elderly in home environments," 2008 11th International Conference on Computer and Information Technology, Khulna, Bangladesh, 2008, pp. 219-224, doi: 10.1109/ICCITECHN.2008.4803020.

[10] Gutiérrez J, Rodríguez V, Martin S. Comprehensive Review of Vision-Based Fall Detection Systems. Sensors. 2021; 21(3):947. https://doi.org/10.3390/s21030947

[11] K. He, G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 2980-2988, doi: 10.1109/ICCV.2017.322.

[12] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[13] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv, abs/1704.04861.

[14] F. Hussain, F. Hussain, M. Ehatisham-ul-Haq and M. A. Azam, "Activity-Aware Fall Detection and Recognition Based on Wearable Sensors," in IEEE Sensors Journal, vol. 19, no. 12, pp. 4528-4536, 15 June15, 2019, doi: 10.1109/JSEN.2019.2898891.

[15] A. Jefiza, E. Pramunanto, H. Boedinoegroho and M. H. Purnomo, "Fall detection based on accelerometer and gyroscope using back propagation," 2017 4th International Conference on Electrical Engineering, Computer

Science and Informatics (EECSI), Yogyakarta, Indonesia, 2017, pp. 1-6, doi: 10.1109/EECSI.2017.8239149.

[16] Jeon, S., Nho, Y., Park, S., Kim, W., Tcho, I., Kim, D., Kwon, D., Choi, Y. (2017). Self-powered fall detection system using pressure sensing triboelectric nanogenerators. Nano Energy, 41, 139-147.

[17] O. Keskes and R. Noumeir, "Vision-Based Fall Detection Using ST-GCN," in IEEE Access, vol. 9, pp. 28224-28236, 2021, doi: 10.1109/ACCESS.2021.3058219.

[18] Khan, M.S., Yu, M., Feng, P., Wang, L., Chambers, J.A. (2015). An unsupervised acoustic fall detection system using source separation for sound interference suppression. Signal Process., 110, 199-210.

[19] Khan, S.S., Hoey, J. (2016). Review of Fall Detection Techniques: A Data Availability Perspective. Medical engineering physics, 39, 12-22.

[20] Bogdan Kwolek and Michal Kepski. 2014. Human fall detection on embedded platform using depth maps and wireless accelerometer. Comput. Methods Prog. Biomed. 117, 3 (December 2014), 489–501. https://doi.org/10.1016/j.cmpb.2014.09.005

[21] Chien-Liang Liu, Chia-Hoang Lee, and Ping-Min Lin. 2010. A fall detection system using k-nearest neighbor classifier. Expert Syst. Appl. 37, 10 (October, 2010), 7174–7181. https://doi.org/10.1016/j.eswa.2010.04.014

[22] De Miguel K, Brunete A, Hernando M, Gambao E. Home Camera-Based Fall Detection System for the Elderly. Sensors. 2017; 17(12):2864. https://doi.org/10.3390/s17122864

[23] Muhammad Mubashir, Ling Shao, and Luke Seed. 2013. A survey on fall detection: Principles and approaches. Neurocomput. 100 (January, 2013), 144–152. https://doi.org/10.1016/j.neucom.2011.09.037

[24] Noury, A. Fleury, P. Rumeau, A. Bourke, G. Laighin, V. Rialle, J. Lundy, Fall detection-principles and methods, in: Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, IEEE, 2007, pp. 1663–1666.

[25] Papandreou, G., Zhu, T., Chen, LC., Gidaris, S., Tompson, J., Murphy, K. (2018). PersonLab: Person Pose Estimation and Instance Segmentation with a Bottom-Up, Part-Based, Geometric Embedding Model. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science(), vol 11218. Springer, Cham. https://doi.org/10.1007/978-3-030-01264-9_17

[26] Emanuele Principi, Stefano Squartini, Roberto Bonfigli, Giacomo Ferroni, and Francesco Piazza. 2015. An integrated system for voice command recognition and emergency detection based on audio signals. Expert Syst. Appl. 42, 13 (August 2015), 5668–5683. https://doi.org/10.1016/j.eswa.2015.02.036

[27] Putra IPES, Brusey J, Gaura E, Vesilo R. An Event-Triggered Machine Learning Approach for Accelerometer-Based Fall Detection. Sensors. 2018; 18(1):20. https://doi.org/10.3390/s18010020

[28] H. Ramirez, S. A. Velastin, I. Meza, E. Fabregas, D. Makris and G. Farias, "Fall Detection and Activity Recognition Using Human Skeleton Features," in IEEE Access, vol. 9, pp. 33532-33542, 2021, doi: 10.1109/ACCESS.2021.3061626.

[29] Rubenstein LZ. Falls in older people: epidemiology, risk factors and strategies for prevention. Age Ageing. 2006 Sep;35 Suppl 2:ii37-ii41. doi: 10.1093/ageing/afl084. PMID: 16926202.

[30] . Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. International Journal of Computer Vision, 115(3):211–252, 2015.

[31] P. S. Sase and S. H. Bhandari, "Human Fall Detection using Depth Videos," 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2018, pp. 546-549, doi: 10.1109/SPIN.2018.8474181.

[32] A. Shahzad and K. Kim, "FallDroid: An Automated Smart-Phone-Based Fall Detection System Using Multiple Kernel Learning," in IEEE Transactions on Industrial Informatics, vol. 15, no. 1, pp. 35-44, Jan. 2019, doi: 10.1109/TII.2018.2839749.

[33] Tanwar R, Nandal N, Zamani M, Manaf AA. Pathway of Trends and Technologies in Fall Detection: A Systematic Review. Healthcare. 2022; 10(1):172. https://doi.org/10.3390/healthcare10010172

[34] J. Wang et al., "Deep High-Resolution Representation Learning for Visual Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 10, pp. 3349-3364, 1 Oct. 2021, doi: 10.1109/TPAMI.2020.2983686.

[35] Zerrouki, N., Houacine, A. Combined curvelets and hidden Markov models for human fall detection. Multimed Tools Appl 77, 6405–6424 (2018). https://doi.org/10.1007/s11042-017-4549-5

[36] Zhong Zhang, Christopher Conly, and Vassilis Athitsos. 2015. A survey on vision-based fall detection. In Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '15). Association for Computing Machinery, New York, NY, USA, Article 46, 1–7. https://doi.org/10.1145/2769493.2769540

[37] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall Press, USA, 3rd edition, 2009.

[38] European Commission, The impact of demographic change in Europe. Available at: <https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/new-push-european-democracy/impact-demographic-change-europe>

[39] World Health Organization, 2021. Falls. Available at: <https://www.who.int/news-room/fact-sheets/detail/falls>.

[40] World Health Organization, 2022. Ageing and health. Available at:<https://www.who.int/news-room/fact-sheets/detail/ageing-and-health>

[41] Google, stoorage.googleapis, XML file. Available at: <https://storage.googleapis.com/tfjs-models>.

[42] TesnorFlow. Available at: <https://www.tensorflow.org>.

[43] TensorFlow, Real-time Human Pose Estimation in the Browser with TensorFlow.js. Available at: <https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html>.

[44] TensorFlowHub. Available at: <https://tfhub.dev>

[45] TensorFlow.js Graph Model Converter. Available at: <https://github.com/patlevin/tfjs-to-tf>

[46] OpenCV. Available at: <https://opencv.org>

[47] Bendersky E. Depthwise separable convolutions for machine learning. Available at: <https://eli.thegreenplace.net/2018/depthwise-separable-convolutions-for-machine-learning>.

[48] Shervine A. Teachings. Available at: <https://stanford.edu/shervine/teaching>.

Α΄ Παράρτημα – Κώδικες

Κώδικας 1: Επεξεργασία εικόνων

```
1  import cv2
2  import os
3  import numpy as np
4  from PIL import Image
5  import os
6
7  # 640*0.3 = 192, 480*0.3=144 +24+24 = 192
8  # 640*0.35 = 224, 480*0.35=168 +28+28 = 224
9  # 640*0.4 = 256, 480*0.4=192 +32+32 = 256
10
11 def rescale_images(scale, shape, path):
12   if shape == 192:
13     pixel = 24
14   elif shape == 224:
15     pixel = 28
16   elif shape == 256:
17     pixel = 32
18
19 for dirpath, _, files in os.walk(path): # <--
20   for file in files:
21     image = Image.open(os.path.join(dirpath, file)) # <--
22     arim = np.array(image)
23     cvimage = cv2.imread(path + '/' + str(file))
24     height = int(cvimage.shape[0]*scale)
25     width = int(cvimage.shape[1]*scale)
26     dimention = (width, height)
27     newimage = cv2.resize(cvimage, dimention,
          interpolation=cv2.INTER_AREA)
```

```
28      newpadimage = cv2.copyMakeBorder(newimage, pixel, pixel, 0, 0,
            cv2.BORDER_CONSTANT, None, value = 0)
29      newimagename = str(file)
30      os.remove(os.path.join(dirpath, file))
31      cv2.imwrite(path + '/' + newimagename, newpadimage)
```

Κώδικας 2: Κατασκευή PoseNet

```
1  import tensorflow as tf
2  from sklearn.metrics import accuracy_score, precision_score,
       recall_score, f1_score, confusion_matrix
3  import numpy as np
4
5  module = tf.saved_model.load("/posenet_75_16/")
6  file_posenet_75 = module.signatures['serving_default']
7
8  module = tf.saved_model.load("/posenet_50_16/")
9  file_posenet_50 = module.signatures['serving_default']
10
11 output_stride = 16
12
13 class BasePosenet_50(tf.keras.layers.Layer):
14   def __init__(self, file_posenet_50, **kwargs):
15     super(BasePosenet_50, self).__init__(**kwargs)
16     self.file_posenet_50 = file_posenet_50
17
18   def call(self, inputs):
19     return file_posenet_50(inputs)
20
21 class BasePosenet_75(tf.keras.layers.Layer):
22   def __init__(self, file_posenet_75, **kwargs):
23     super(BasePosenet_75, self).__init__(**kwargs)
```

```
24      self.file_posenet_75 = file_posenet_75
25
26    def call(self, inputs):
27      return file_posenet_75(inputs)
28
29  class PoseNetOutputLayer(tf.keras.layers.Layer):
30    def __init__(self, output_stride= 16, **kwargs):
31      super(PoseNetOutputLayer, self).__init__(**kwargs)
32      self.output_stride = output_stride
33
34  #
       https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html
35    def call(self, inputs):
36      heatmaps, offsets = tf.squeeze(inputs['heatmaps']),
           tf.squeeze(inputs['offsets'])
37
38      scores = tf.sigmoid(heatmaps)
39
40      height, width, depth = scores.shape
41      reshaped_heatmap = tf.reshape(scores, [height * width, depth])
42      coords = tf.argmax(reshaped_heatmap, axis=0, output_type =
           tf.int32)
43      y_coords = tf.math.floordiv(coords,width)
44      x_coords = tf.math.floormod(coords, width)
45      heat_coords = tf.concat([tf.expand_dims(y_coords, 1),
           tf.expand_dims(x_coords, 1)], axis=1)
46
47      keypoint_indices = tf.range(depth)
48
49      keypoint_confidence = tf.gather_nd(scores, tf.stack([y_coords,
           x_coords, keypoint_indices], axis=-1))
50
```

```
51      offset_y = tf.gather_nd(offsets, tf.stack([y_coords, x_coords,
            keypoint_indices], axis=1))
52      offset_x = tf.gather_nd(offsets, tf.stack([y_coords, x_coords,
            keypoint_indices + depth], axis=1))
53
54      offest_vectors = tf.stack([offset_y, offset_x], axis=1)
55
56      scaled_heatmap = tf.cast(heat_coords * output_stride, dtype=
            tf.float32)
57      keypoint_positions = scaled_heatmap + offest_vectors
58
59      final_result = tf.concat([keypoint_positions,
            tf.expand_dims(keypoint_confidence, axis=1)], axis=1)
60
61      return final_result
62
63  def create_posenet(version, size):
64    output_stride = 16
65
66    if size == 192:
67      inputs = tf.keras.layers.Input(shape=(192,192,3))
68    elif size == 224:
69      inputs = tf.keras.layers.Input(shape=(224,224,3))
70    elif size == 256:
71      inputs = tf.keras.layers.Input(shape=(256,256,3))
72
73    if version == 75:
74      x = BasePosenet_75(file_posenet_75= file_posenet_75)(inputs)
75    elif version == 50:
76      x = BasePosenet_50(file_posenet_50= file_posenet_50)(inputs)
77
78    outputs = PoseNetOutputLayer(output_stride= 16)(x)
```

```
79
80    posenet_whole = tf.keras.Model(inputs= inputs, outputs= outputs)
81
82    return posenet_whole
```

Κώδικας 3: Δημιουργία μοντέλων

```
1  def create_mlp():
2  #Define the classification model
3  model = tf.keras.Sequential([
4    tf.keras.layers.Flatten(input_shape= (17,3)),
5    tf.keras.layers.Dense(64, activation='relu'),
6    tf.keras.layers.Dropout(rate= 0.5),
7    tf.keras.layers.Dense(32, activation='relu'),
8    tf.keras.layers.Dropout(rate= 0.5),
9    tf.keras.layers.Dense(16, activation='relu'),
10   tf.keras.layers.Dropout(rate= 0.25),
11   tf.keras.layers.Dense(1, activation='sigmoid') # 2 output
       classes: fall or not fall
12 ])
13
14 return model
15
16 def create_wide_mlp():
17 #Define the classification model
18 model = tf.keras.Sequential([
19   tf.keras.layers.Flatten(input_shape= (17,3)),
20   tf.keras.layers.Dense(64, activation='relu'),
21   tf.keras.layers.Dropout(rate= 0.5),
22   tf.keras.layers.Dense(128, activation='relu'),
23   tf.keras.layers.Dropout(rate= 0.5),
24   tf.keras.layers.Dense(32, activation='relu'),
```

```
25    tf.keras.layers.Dropout(rate= 0.5),
26    tf.keras.layers.Dense(16, activation='relu'),
27    tf.keras.layers.Dropout(rate= 0.25),
28    tf.keras.layers.Dense(1, activation='sigmoid') # 2 output
          classes: fall or not fall
29  ])
30
31  return model
32
33  def create_fall_detection_model(version, size):
34
35  fd_model = tf.keras.Sequential()
36
37  if version == 50:
38    if size == 192:
39      posenet = posenet_50_192
40      mlp = mlp_50_192
41    elif size == 224:
42      posenet = posenet_50_224
43      mlp = mlp_50_224
44    elif size == 256:
45      posenet = posenet_50_256
46      mlp = mlp_50_256
47  elif version == 75:
48    if size == 192:
49      posenet = posenet_75_192
50      mlp = mlp_75_192
51    elif size == 224:
52      posenet = posenet_75_224
53      mlp = mlp_75_224
54    elif size == 256:
55      posenet = posenet_75_256
```

```
56      mlp = mlp_75_256
57
58  fd_model.add(posenet)
59  fd_model.add(tf.keras.layers.Lambda(lambda x: tf.expand_dims(x,
        axis=0)))
60  fd_model.add(mlp)
61
62  return fd_model
```

Οι ακόλουθοι κώδικες είναι παρόμοιοι για όλα τα μοντέλα. Για λόγους συντομίας παρουσιάζεται μόνο το normal mlp με Posenet 0.50 σε ανάλυση 192×192.

Κώδικας 4: Εκπαίδευση και testing μοντέλων

```
1   posenet_50_192 = create_posenet(version=50, size=192)
2
3   # train:70% validation:10% test:20%,
4
5   def create_dataset_splits(size):
6     seed = 42
7     shuffle_value = True
8     validation_split = 0.3
9
10    if size == 192:
11      dir = '/urfall_192/'
12      image_size = (192,192)
13    elif size == 224:
14      dir = '/urfall_224/'
15      image_size = (224,224)
16    if size == 256:
17      dir = '/urfall_256/'
18      image_size = (256,256)
19
```

```
20    train_ds = tf.keras.utils.image_dataset_from_directory(
21      directory = dir,
22      image_size = image_size,
23      label_mode ='binary',
24      class_names = ('no_fall', "fall"),
25      batch_size= 16,
26      validation_split = validation_split,
27      subset = "training",
28      seed = seed,
29      shuffle = shuffle_value)
30
31    val_ds = tf.keras.utils.image_dataset_from_directory(
32      directory = dir,
33      image_size = image_size,
34      label_mode ='binary',
35      class_names = ('no_fall', "fall"),
36      batch_size= 16,
37      validation_split = validation_split,
38      subset = "validation",
39      seed = seed,
40      shuffle = shuffle_value)
41
42    val_batches = tf.data.experimental.cardinality(val_ds)
43    test_ds = val_ds.take((2*val_batches) // 3)
44    val_ds = val_ds.skip((2*val_batches) // 3)
45
46    return train_ds, val_ds, test_ds
47
48 train_50_192, val_50_192, test_50_192 = create_dataset_splits(size=
      192)
49
50 # apply posenet on images
```

```
51 def preprocess_and_apply_posenet_vectorized(image, label,
       posenet_model):
52   def preprocess_and_apply_single(image, posenet_model):
53     image = image/255.0
54     pose_output = posenet_model(tf.expand_dims(image, axis=0))
55   return pose_output
56
57   pose_outputs = tf.vectorized_map(lambda img:
        preprocess_and_apply_single(img, posenet_model), image)
58   return pose_outputs, label
59
60 train_50_192 = train_50_192.map(lambda image, label:
     preprocess_and_apply_posenet_vectorized(image, label,
     posenet_50_192))
61 val_50_192 = val_50_192.map(lambda image, label:
     preprocess_and_apply_posenet_vectorized(image, label,
     posenet_50_192))
62 test_50_192 = test_50_192.map(lambda image, label:
     preprocess_and_apply_posenet_vectorized(image, label,
     posenet_50_192))
63
64 mlp_50_192 = create_mlp()
65
66 mlp_50_192.compile(
67   optimizer = tf.keras.optimizers.Adam(learning_rate=0.001),
68   loss = tf.keras.losses.BinaryCrossentropy(),
69   metrics = ['accuracy',
70         tf.keras.metrics.Precision(),
71         tf.keras.metrics.Recall(),
72         tf.keras.metrics.TruePositives(),
73         tf.keras.metrics.FalsePositives(),
74         tf.keras.metrics.TrueNegatives(),
```

```
75          tf.keras.metrics.FalseNegatives()
76          ])
77
78 his_mlp_50_192 = mlp_50_192.fit(
79     train_50_192,
80     validation_data=val_50_192,
81     epochs=100)
82
83 # testing
84 mlp_50_192.evaluate(test_50_192)
```

Κώδικας 5: Quantization μοντέλων

```
1  def representative_50_192():
2    data = train_50_192
3
4    for image_batch, _ in data.take(200):
5      for image in image_batch:
6        exp_images = tf.expand_dims(image, axis=0)
7        yield [exp_images]
8
9  def quantize_model(model,representative_dataset):
10   converter = tf.lite.TFLiteConverter.from_keras_model(model)
11   converter.optimizations = [tf.lite.Optimize.DEFAULT]
12   converter.representative_dataset = representative_dataset
13   converter.target_spec.supported_ops =
       [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
14   converter.inference_input_type = tf.int8 # or tf.uint8
15   converter.inference_output_type = tf.int8 # or tf.uint8
16   tflite_quant_model = converter.convert()
17
18   return tflite_quant_model
```

```
19
20 fd_50_192 = create_fall_detection_model(50, 192)
21
22 quant_fd_50_192 = quantize_model(saved_fd_50_192,
      representative_50_192)
```

Κώδικας 6: Testing quantized μοντέλα

```
1  # now we need the images
2  train_50_192, val_50_192, test_50_192 = create_dataset_splits(size=
       192)
3  test_50_192 = test_50_192.batch(1)
4  train_50_192 = val_50_192 = None
5
6  x_test_50_192 = np.asarray(list(test_50_192.map(lambda x, y: x)))
7  y_test_50_192 = np.asarray(list(test_50_192.map(lambda x, y: y)))
8
9  def calc_metr(path, images, labels):
10   interpreter = tf.lite.Interpreter(model_path = path)
11   interpreter.allocate_tensors()
12   input_details = interpreter.get_input_details()
13   output_details = interpreter.get_output_details()
14
15   input_images = images.astype(np.int8)
16
17   predictions = []
18
19   for image in input_images:
20     # Assuming your image is in the appropriate format, preprocess
         it
21     # to match the input shape expected by the TFLite model
22     int8_image = image.astype(np.int8)
```

```
23
24      # Set input tensor
25      interpreter.set_tensor(input_details[0]['index'], int8_image)
26
27      # Run inference
28      interpreter.invoke()
29
30      # Get output tensor and post-process predictions
31      output = interpreter.get_tensor(output_details[0]['index'])
32      predictions.append(output)
33
34    predicted_labels = [1 if pred >= 0 else 0 for pred in predictions]
35    labels = labels.astype(np.int32)
36
37    accuracy = accuracy_score(labels, predicted_labels)
38    precision = precision_score(labels, predicted_labels)
39    recall = recall_score(labels, predicted_labels)
40    f1 = f1_score(labels, predicted_labels)
41
42    print("Accuracy:", accuracy)
43    print("Precision:", precision)
44    print("Recall:", recall)
45    print("F1-Score:", f1)
46
47  calc_metr(path = '/fd_50_192/int8/model.tflite', images=
      x_test_50_192, labels= y_test_50_192)
```