ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή Εργασία

Ανάπτυξη Εφαρμογής για Αναγνώριση Ανθρώπινων Δραστηριοτήτων

'Application Development for Human Activity Recognition'

Παναγιωτάτου Αγάθη Ανδρονίκη

Επιβλέπων Καθηγητής: Αναστάσιος Γούναρης

Σεπτέμβριος 2024

Θεσσαλονίκη

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, η ανάπτυξη και υλοποίηση ευφυών σπιτιών έχει προσελκύσει σημαντικά το ενδιαφέρον των ερευνητών, καθώς οι εφαρμογές τους εκτείνονται σε πολλούς τομείς, από τη διαχείριση οικιακών συσκευών και τη βελτίωση της ενεργειακής απόδοσης, μέχρι την ενίσχυση της ασφάλειας, την παρακολούθηση και υποστήριξη ευπαθών ομάδων, όπως ηλικιωμένοι, ασθενείς και άτομα με ειδικές ανάγκες. Για την επίτευξη αυτών των στόχων, η ακριβής αναγνώριση ανθρώπινων δραστηριοτήτων είναι αναγκαία. Οι τεχνολογίες που χρησιμοποιούνται για τον σκοπό αυτό περιλαμβάνουν οπτικά δεδομένα από κάμερες, αισθητήρες που ανιχνεύουν την κίνηση και τις περιβαλλοντικές συνθήκες, καθώς και φορητές συσκευές, όπως έξυπνα ρολόγια και κινητά τηλέφωνα, τα οποία μπορούν να παρακολουθούν τη φυσική κατάσταση και τις καθημερινές δραστηριότητες των χρηστών. Αυτές οι πληροφορίες αξιοποιούνται για τη δημιουργία ενός εξατομικευμένου, ευφυούς περιβάλλοντος, που προσαρμόζεται στις ανάγκες και τις συνήθειες των κατοίκων.

Παρά τη σημαντική πρόοδο που έχει σημειωθεί στον τομέα της αναγνώρισης ανθρώπινων δραστηριοτήτων, υπάρχουν ακόμα πολλά περιθώρια βελτίωσης και περαιτέρω ανάπτυξης. Οι τεχνικές προκλήσεις περιλαμβάνουν τη διαχείριση μεγάλου όγκου δεδομένων σε πραγματικό χρόνο, την προσαρμογή των συστημάτων στις ανάγκες των χρηστών καθώς και τη διαλειτουργικότητα μεταξύ διαφόρων συσκευών .

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην αξιολόγηση διαφόρων μοντέλων αναγνώρισης ανθρώπινων δραστηριοτήτων σε οικιακά περιβάλλοντα, παρακολούθηση ασθενών μέσω οπτικών δεδομένων. Αρχικά, θα παρουσιαστούν τα θεωρητικά θεμέλια και θα αναλυθούν συναφείς εργασίες που σχετίζονται με τη συγκεκριμένη περιοχή έρευνας, ενώ στη συνέχεια θα γίνει αξιολόγηση μοντέλων και το καταλληλότερο μοντέλο θα επανεκπαιδευτεί για την επίτευξη καλύτερων αποτελεσμάτων.

ABSTRACT

In recent years, the development and implementation of smart homes have significantly attracted researchers' interest, as their applications cover multiple areas, from managing household appliances and improving energy efficiency to enhancing security, monitoring, and supporting vulnerable groups such as the elderly, patients, and individuals with disabilities. To achieve these goals, accurate recognition of human activities is necessary. The technologies used for this purpose include visual data from cameras, sensors that detect movement and environmental conditions, as well as wearable devices such as smartwatches and mobile phones, which can monitor users' physical condition and daily activities. This information is utilized to create a personalized, intelligent environment that adapts to the needs and habits of the residents.

Despite the significant progress made in the field of human activity recognition, there is still much room for improvement and further development. Technical challenges include managing large volumes of real-time data, adapting systems to user needs, and ensuring interoperability between different devices.

This thesis focuses on evaluating various models for human activity recognition in domestic environments, monitoring patients through visual data. Initially, the theoretical foundations will be presented, and related research work in this specific research area will be analyzed. Subsequently, models will be evaluated, and the most suitable model will be retrained to achieve better results.

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της πτυχιακής μου εργασίας, θα ήθελα να εκφράσω τις ειλικρινείς ευχαριστίες μου στον επιβλέποντα καθηγητή κ. Αναστάσιο Γούναρη για την πολύτιμη βοήθεια και καθοδήγηση που μου προσέφερε καθ' όλη τη διάρκεια της εργασίας. Η εμπιστοσύνη που μου έδειξε ήταν πολύτιμη και καθοριστική για την επιτυχία αυτής της προσπάθειας.

Επίσης, θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου και τους φίλους μου για την στήριξή τους κατά τη διάρκεια αυτής της διαδικασίας.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΟΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΚΩΔΙΚΑ

ΚΕΦΑΛΑΙΟ 1

1. ΕΙΣΑΓΩΓΗ

1.1 Εισαγωγικά Στοιχεία

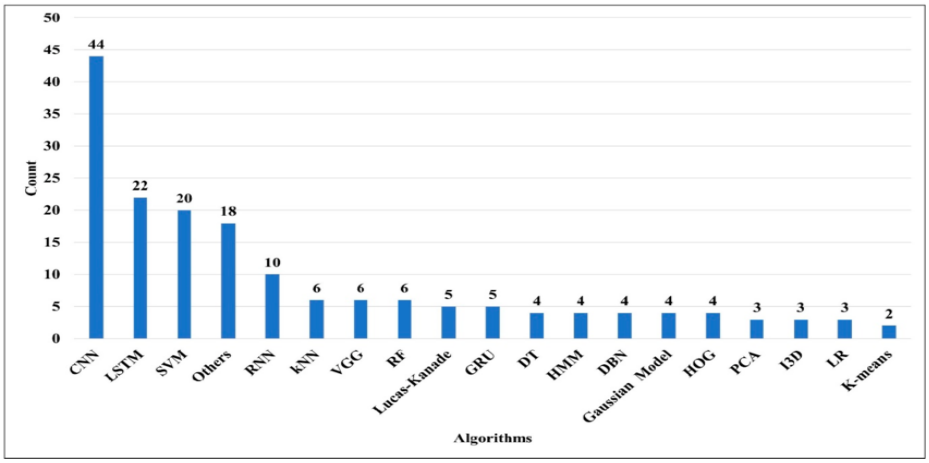
Η αναγνώριση ανθρώπινης δραστηριότητας (Human Activity Recognition-HAR) έχει εξελιχθεί σε ένα κρίσιμο πεδίο έρευνας με πολυάριθμες εφαρμογές, όπως η διαχείριση έξυπνων σπιτιών, η υγειονομική περίθαλψη, και η ασφάλεια. Οι τεχνολογίες IoT και τα συστήματα τεχνητής νοημοσύνης έχουν ενισχύσει σημαντικά τις δυνατότητες παρακολούθησης και ανάλυσης ανθρώπινων δραστηριοτήτων σε πραγματικό χρόνο. Συγκεκριμένα, οι τεχνικές μηχανικής μάθησης, όπως τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network-CNN), τα Δίκτυα Μακράς Βραχύχρονης Μνήμης (Long Short-Term Memory -LSTM), τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Network-RNN) έχουν γίνει αναπόσπαστα εργαλεία για την ακριβή και αξιόπιστη αναγνώριση δραστηριοτήτων[5] [6] [7] . Όπως αναφέρεται και στην πηγή [9], η εικόνα 1.1.α παρουσιάζει τη συχνότητα των αλγορίθμων που χρησιμοποιούνται στη για HAR.

Τα CNN είναι ιδιαίτερα αποτελεσματικά στην ανάλυση οπτικών δεδομένων, ενώ τα LSTM και RNN αξιοποιούνται για την ανάλυση χρονοσειρών δεδομένων, όπως αυτά που προέρχονται από αισθητήρες ή φορητές συσκευές. Αυτές οι τεχνικές επιτρέπουν την επεξεργασία μεγάλων όγκων δεδομένων, προσφέροντας εξατομικευμένες λύσεις που προσαρμόζονται στις ανάγκες των χρηστών και διασφαλίζουν την αποτελεσματική λειτουργία των συστημάτων HAR.

Παρά την πρόοδο, το πεδίο της HAR αντιμετωπίζει σοβαρές προκλήσεις, όπως η ανάγκη για διαχείριση και ανάλυση δεδομένων από διάφορες πηγές, καθώς και η διασφάλιση της ιδιωτικότητας των χρηστών. Η ακρίβεια στην αναγνώριση των δραστηριοτήτων είναι κρίσιμη για την επιτυχή εφαρμογή των συστημάτων αυτών, ειδικά όταν τα δεδομένα συλλέγονται σε περιβάλλοντα με ποικίλες συνθήκες ή από χρήστες με διαφορετικές συνήθειες και ανάγκες.

Οι μελλοντικές εξελίξεις στην τεχνητή νοημοσύνη και τη μηχανική μάθηση αναμένεται να βελτιώσουν ακόμα περισσότερο την απόδοση των συστημάτων HAR. Ωστόσο, για να επιτευχθεί η πλήρης δυναμική των εφαρμογών αυτών, είναι απαραίτητη η συνεχής έρευνα και ανάπτυξη, με έμφαση στην ενίσχυση της διαλειτουργικότητας, της ασφάλειας και της προσαρμοστικότητας των συστημάτων.

Τα συστήματα HAR έχουν τη δυνατότητα να προσφέρουν σημαντικές βελτιώσεις στην ποιότητα ζωής, ιδιαίτερα για ευπαθείς ομάδες πληθυσμού, όπως ηλικιωμένοι ή άτομα με ειδικές ανάγκες. Με την ενσωμάτωση των σύγχρονων τεχνικών, τα έξυπνα σπίτια μπορούν να γίνουν πιο φιλικά προς τον χρήστη, προσφέροντας εξατομικευμένες λύσεις που ανταποκρίνονται στις ανάγκες των κατοίκων, ενώ παράλληλα προάγουν την ασφάλεια και την ευελιξία.



Εικόνα 1.1.α: Συχνότητα τεχνικών/αλγορίθμων που χρησιμοποιούνται για την Αναγνώριση Ανθρώπινης Δραστηριότητας (HAR) [9]

1.2 Εφαρμογές της αναγνώρισης δραστηριότητας

Η ικανότητα ενός συστήματος να κατανοεί και να αναγνωρίζει σύνθετες ανθρώπινες ενέργειες έχει πολυάριθμες εφαρμογές. Παρακάτω θα αναφερθούν μερικές από τις κύριες εφαρμογές που τονίζουν τη σημασία αυτού του ερευνητικού πεδίου.

Υγεία

Η εφαρμογή της αναγνώρισης ανθρώπινης δραστηριότητας (HAR) στον τομέα της υγείας έχει αποδειχθεί εξαιρετικά σημαντική, ιδιαίτερα για την υποστήριξη ηλικιωμένων ατόμων και ατόμων με χρόνιες ασθένειες. Η αύξηση του προσδόκιμου ζωής, σύμφωνα με την έρευνα της “Our World in Data” (βλ εικόνα 1.1.β), μαζί με τις τεχνολογικές εξελίξεις, έχει οδηγήσει σε έναν αυξανόμενο αριθμό ηλικιωμένων που χρειάζονται συνεχή φροντίδα. Η HAR, σε συνδυασμό με τεχνολογίες όπως η τεχνητή νοημοσύνη (AI), το IoT, δημιουργούν τις προϋποθέσεις για την ανάπτυξη έξυπνων συστημάτων φροντίδας κατ’ οίκον, τα οποία όχι μόνο βελτιώνουν την ποιότητα ζωής, αλλά και περιορίζουν τα έξοδα υγειονομικής περίθαλψης.

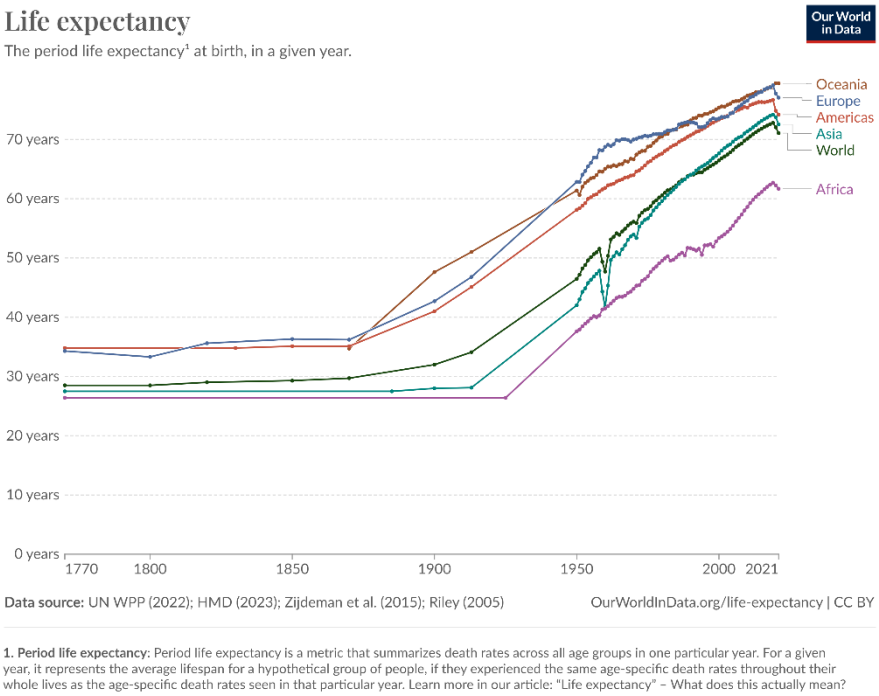
Η HAR μπορεί να παρακολουθεί συνεχώς τις δραστηριότητες και τις φυσιολογικές παραμέτρους των ασθενών, επιτρέποντας στους παρόχους υγειονομικής περίθαλψης να εντοπίζουν έγκαιρα κρίσιμες καταστάσεις και να παρέχουν άμεση επέμβαση. Συσκευές όπως τα έξυπνα ρολόγια που καταγράφουν δεδομένα όπως την αρτηριακή πίεση και τον καρδιακό ρυθμό, ενσωματώνονται σε έξυπνα περιβάλλοντα για την καθημερινή παρακολούθηση των χρηστών.

Η δυνατότητα παρακολούθησης των ασθενών από απόσταση είναι ένα ακόμα σημαντικό πλεονέκτημα της HAR. Αυτή η τεχνολογία επιτρέπει στους ιατρούς να αξιολογούν την υγεία των ασθενών και να προσαρμόζουν τα θεραπευτικά πλάνα χωρίς την ανάγκη φυσικής παρουσίας, κάτι που είναι εξαιρετικά χρήσιμο για άτομα με περιορισμένη κινητικότητα ή για ασθενείς που ζουν σε απομακρυσμένες περιοχές.

Επιπλέον, η χρήση ρομπότ για κοινωνική φροντίδα αποτελεί άλλη μία καινοτομία που συνδέεται με την HAR. Αυτά τα ρομπότ μπορούν να αλληλεπιδρούν με τους ασθενείς, να τους υπενθυμίζουν να παίρνουν τα φάρμακά τους ή να εκτελούν συγκεκριμένες δραστηριότητες, ενώ παράλληλα παρέχουν συναισθηματική υποστήριξη.

Τα έξυπνα περιβάλλοντα και τα συστήματα φροντίδας που βασίζονται στην HAR προσφέρουν αυτονομία στους ηλικιωμένους και στους ασθενείς, επιτρέποντάς τους να ζουν με μεγαλύτερη ανεξαρτησία, ενώ ταυτόχρονα διασφαλίζεται ότι λαμβάνουν την αναγκαία φροντίδα. Αυτός ο συνδυασμός τεχνολογιών καθιστά δυνατή τη δημιουργία ενός ενεργού, ασφαλούς και ευφυούς περιβάλλοντος διαβίωσης, που προσαρμόζεται στις ατομικές ανάγκες των χρηστών.

Οι τεχνολογίες αυτές έχουν τη δυνατότητα να μειώσουν την ανάγκη για ανθρώπινο δυναμικό στη φροντίδα των ηλικιωμένων, εξοικονομώντας πόρους και μειώνοντας την πίεση στα εθνικά συστήματα υγείας. Παράλληλα, ενισχύεται η ποιότητα ζωής των ασθενών, καθώς τους επιτρέπεται να διατηρούν μια πιο ενεργή και αυτόνομη καθημερινότητα. [1] [13]



Εικόνα 1.2.α: Το προσδόκιμο ζωής έχει αυξηθεί σημαντικά σε παγκόσμιο επίπεδο τα τελευταία χρόνια [8]

Ασφάλεια

Η χρήση της αναγνώρισης ανθρώπινης δραστηριότητας (HAR) για την ασφάλεια και επιτήρηση των σπιτιών έχει εξελιχθεί σημαντικά με την ανάπτυξη έξυπνων συσκευών παρακολούθησης. Αυτές οι συσκευές, επιτρέπουν την εύκολη εγκατάσταση και τη συνεχή παρακολούθηση του σπιτιού από οπουδήποτε μέσω smartphone ή υπολογιστή. Παρέχουν ειδοποιήσεις σε πραγματικό χρόνο όταν ανιχνεύεται κίνηση, προσφέροντας ηρεμία στους χρήστες, αποτρέποντας εισβολές και βοηθώντας στη μείωση των εξόδων μιας κατοικίας. Η χρήση HAR σε συνδυασμό με αυτές τις συσκευές προσφέρει προηγμένη επιτήρηση, επιτρέποντας την αναγνώριση ύποπτων κινήσεων ή δραστηριοτήτων εντός

και εκτός της οικίας. Αυτό μπορεί να περιλαμβάνει τη διάκριση μεταξύ φυσιολογικών δραστηριοτήτων, όπως η παρουσία κατοικίδιων, και δυνητικά επικίνδυνων καταστάσεων, όπως μια εισβολή. Το σύστημα μπορεί να επεκταθεί με την προσθήκη περισσότερων συσκευών, καλύπτοντας έτσι μεγαλύτερες περιοχές και παρέχοντας πιο ολοκληρωμένη προστασία. Παράλληλα, αυτές οι τεχνολογίες είναι προσβάσιμες και για τους απλούς καταναλωτές, καθιστώντας την ασφάλεια και την παρακολούθηση σπιτιών προσιτή και εύκολη στη χρήση.[2]

Η αναγνώριση ανθρώπινης δραστηριότητας (HAR) έχει καταστεί και κρίσιμη για την ασφάλεια στις πόλεις, καθώς επιτρέπει την αυτόματη παρακολούθηση και ανάλυση μεγάλου όγκου βίντεο από συστήματα CCTV. Η χειροκίνητη παρακολούθηση αυτών των δεδομένων είναι δύσκολη και συχνά αναποτελεσματική, οδηγώντας σε πιθανές παραλείψεις. Ωστόσο, η HAR μπορεί να εντοπίζει αυτόματα ανωμαλίες ή ύποπτες δραστηριότητες σε πραγματικό χρόνο, προειδοποιώντας τις αρχές άμεσα και μειώνοντας τον χρόνο αντίδρασης σε περιστατικά, βελτιώνοντας έτσι την ασφάλεια σε δημόσιους χώρους.[3] [14]

Διαχείριση Οικιακών Συσκευών και Ενεργειακή Διαχείριση

Η αναγνώριση ανθρώπινης δραστηριότητας αποτελεί βασικό παράγοντα στη διαχείριση οικιακών συσκευών και τη βελτίωση της ενεργειακής αποδοτικότητας στα σύγχρονα έξυπνα σπίτια και συστήματα διαχείρισης ενέργειας . Με την ανάπτυξη και τη χρήση τέτοιων τεχνολογιών, οι εφαρμογές έξυπνων σπιτιών και τα συστήματα διαχείρισης ενέργειας κερδίζουν ολοένα και μεγαλύτερη προσοχή στην ερευνητική κοινότητα, χάρη στα πλεονεκτήματα που προσφέρουν. Αυτές οι τεχνολογίες στοχεύουν στην αυτοματοποίηση και βελτιστοποίηση της λειτουργίας των οικιακών συσκευών, προσφέροντας σημαντικά οφέλη τόσο για τους χρήστες όσο και για την ενεργειακή απόδοση των σπιτιών.

Η αναγνώριση της ανθρώπινης δραστηριότητας, όπως η ανίχνευση της παρουσίας, των κινήσεων και των δραστηριοτήτων των κατοίκων, είναι καθοριστική για την ανάπτυξη και λειτουργία αυτών των έξυπνων συστημάτων. Μέσω της χρήσης αισθητήρων και αλγορίθμων ανάλυσης δεδομένων, οι σύγχρονες τεχνολογίες μπορούν να παρακολουθούν και να καταγράφουν την ενεργειακή κατανάλωση των συσκευών, εντοπίζοντας τη χρήση τους και αναγνωρίζοντας πρότυπα συμπεριφοράς. Αυτό επιτρέπει την ανάπτυξη εφαρμογών που βελτιώνουν τη διαχείριση των οικιακών συσκευών και συμβάλλουν στη μείωση της κατανάλωσης ενέργειας.

Επιπλέον, η ανάλυση δεδομένων από τις συσκευές μπορεί να παρέχει πολύτιμες πληροφορίες για τα πρότυπα συμπεριφοράς των κατοίκων. Αυτά τα δεδομένα περιλαμβάνουν πληροφορίες για τις συνήθειες τους, όπως οι ώρες απουσίας από το σπίτι, οι ώρες ύπνου και άλλες καθημερινές δραστηριότητες. Αυτή η γνώση επιτρέπει τη δημιουργία εξατομικευμένων προγραμμάτων διαχείρισης ενέργειας, τα οποία προσαρμόζονται στις ανάγκες και τις συνήθειες των χρηστών, βελτιώνοντας την ενεργειακή αποδοτικότητα και μειώνοντας τα έξοδα ενέργειας.[4] [15]

Αλληλεπίδραση Ανθρώπου Υπολογιστή

Η αναγνώριση ανθρώπινων δραστηριοτήτων παίζει σημαντικό ρόλο στην αλληλεπίδραση ανθρώπου υπολογιστή (Human Computer Interaction) και συμβάλλει σημαντικά στην ανάπτυξη πιο έξυπνων και προσαρμοστικών συστημάτων. Η σημασία της HAR στο HCI έγκειται σε αρκετούς τομείς. Η HAR επιτρέπει στα συστήματα να κατανοούν καλύτερα τις προθέσεις και τις ανάγκες των χρηστών, προσφέροντας εξατομικευμένες εμπειρίες. Με την αναγνώριση των δραστηριοτήτων του χρήστη, όπως η εργασία, η άσκηση ή η ανάπαυση, τα συστήματα μπορούν να προσαρμόζουν τις λειτουργίες τους αυτόματα, βελτιώνοντας την αλληλεπίδραση και την άνεση του χρήστη.

1.3 Σκοπός της εργασίας

Όπως έχει αναφερθεί, η αναγνώριση ανθρώπινων δραστηριοτήτων παίζει σημαντικό ρόλο στην επιτήρηση και παρακολούθηση ασθενών και ατόμων με ειδικές ανάγκες. Στην παρούσα εργασία, θα επικεντρωθούμε στην αναγνώριση απλών, καθημερινών δραστηριοτήτων που εκτελούνται σε οικιακό περιβάλλον, οι οποίες είναι καίριες για την παρακολούθηση των ασθενών. Σκοπός μας είναι να χρησιμοποιήσουμε έτοιμα συστήματα που έχουν εκπαιδευτεί για πιο σύνθετες δραστηριότητες και να αξιολογήσουμε την απόδοσή τους σε αυτό το πλαίσιο. Το σύστημα με την καλύτερη απόδοση θα επανεκπαιδευτεί για να εξετάσουμε τις επιπλέον δυνατότητές του.

Η δομή των επόμενων κεφαλαίων της εργασίας είναι η εξής: Στο 2ο κεφάλαιο θα παρουσιαστεί το επιστημονικό υπόβαθρο των τεχνολογιών που χρησιμοποιούνται, με σκοπό την καλύτερη κατανόηση του θέματος. Στο 3ο κεφάλαιο θα εξεταστούν συναφείς ερευνητικές εργασίες που εστιάζουν στη χρήση της αναγνώρισης ανθρώπινων δραστηριοτήτων (HAR) για την παρακολούθηση ασθενών. Στο 4ο κεφάλαιο θα γίνει η αξιολόγηση των επιδόσεων των έτοιμων συστημάτων πάνω στο σύνολο δεδομένων που θα χρησιμοποιήσουμε. Τέλος, στο 5ο κεφάλαιο θα αναλυθούν τα βήματα και τα αποτελέσματα από την επανεκπαίδευση του μοντέλου με την καλύτερη απόδοση.

ΚΕΦΑΛΑΙΟ 2

2.ΕΠΙΣΤΗΜΟΝΙΚΟ ΥΠΟΒΑΘΡΟ

2.1 Εξόρυξη Δεδομένων

2.1.1 Εισαγωγή

Η εξόρυξη δεδομένων (Data Mining) είναι η διαδικασία ανακάλυψης χρήσιμων πληροφοριών, μοτίβων, τάσεων και γνώσεων μέσα από μεγάλες δεξαμενές δεδομένων. Χρησιμοποιεί τεχνικές από το πεδίο της μηχανικής μάθησης, της στατιστικής και της βάσης δεδομένων για να ερευνήσει σε βάθος τα δεδομένα και να αποκαλύψει νέα και χρήσιμα πρότυπα, τα οποία διαφορετικά θα παρέμεναν άγνωστα. Στόχος της είναι η μετατροπή των ακατέργαστων δεδομένων σε σημαντικές και χρήσιμες πληροφορίες, οι οποίες μπορούν να αξιοποιηθούν για τη λήψη αποφάσεων, την πρόβλεψη μελλοντικών τάσεων και τη βελτίωση επιχειρηματικών και άλλων διαδικασιών.

Η εξόρυξη δεδομένων αποτελεί ένα αναπόσπαστο κομμάτι της Ανακάλυψης Γνώσης από τις Βάσεις Δεδομένων, μια συνολική διεργασία που περιλαμβάνει μια σειρά βημάτων από την προεπεξεργασία δεδομένων μέχρι την επεξεργασία των αποτελεσμάτων. Κατά την προεπεξεργασία, τα ακατέργαστα δεδομένα μετατρέπονται σε μορφή κατάλληλη για ανάλυση, μέσω διαδικασιών όπως η συγχώνευση πληροφοριών από πολλαπλές πηγές, ο καθαρισμός δεδομένων για την εξάλειψη θορύβου και διπλότυπων, και η επιλογή σχετικών εγγραφών και χαρακτηριστικών. [23]

2.1.2 Σημασία στην Αναγνώριση Ανθρώπινης Δραστηριότητας

Η εξόρυξη δεδομένων παίζει σημαντικό ρόλο στην αναγνώριση ανθρώπινης δραστηριότητας (HAR - Human Activity Recognition) διότι επιτρέπει την ανάλυση μεγάλων και ποικιλόμορφων συνόλων δεδομένων για την εξαγωγή χρήσιμων μοτίβων και γνώσεων. Αυτή η διαδικασία είναι σημαντική για την ανάπτυξη ακριβών και αποδοτικών συστημάτων που μπορούν να αναγνωρίσουν και να ταξινομήσουν ανθρώπινες δραστηριότητες από δεδομένα που συλλέγονται μέσω αισθητήρων και άλλων συσκευών.

2.2 Μηχανική Μάθηση

2.2.1 Εισαγωγή

Η μηχανική μάθηση (Machine Learning) είναι ένας κλάδος της τεχνητής νοημοσύνης που επικεντρώνεται στην ανάπτυξη αλγορίθμων και μοντέλων που επιτρέπουν στους υπολογιστές να μαθαίνουν από δεδομένα και να λαμβάνουν αποφάσεις ή να κάνουν προβλέψεις χωρίς να είναι ρητά προγραμματισμένοι γι' αυτό. Η βασική αρχή της μηχανικής μάθησης είναι η βελτίωση της απόδοσης των μοντέλων μέσω της εμπειρίας, δηλαδή με την επεξεργασία και ανάλυση μεγάλων συνόλων δεδομένων.

Η μηχανική μάθηση αξιοποιεί τις τεχνικές της εξόρυξης δεδομένων για να αναλύσει δεδομένα, να ανιχνεύσει μοτίβα και να εκπαιδεύσει αλγορίθμους που επιτρέπουν στους υπολογιστές να μαθαίνουν και να βελτιώνουν την απόδοσή τους. [27]

2.2.2 Κατηγορίες Μηχανικής Μάθησης

Η μηχανική μάθηση διακρίνεται σε τρεις βασικές κατηγορίες:

Επιβλεπόμενη Μάθηση (Supervised Learning)

Στην επιβλεπόμενη μάθηση, τα μοντέλα εκπαιδεύονται με δεδομένα που περιλαμβάνουν τόσο τις εισόδους όσο και τις αντίστοιχες εξόδους. Ο στόχος είναι να μάθουν τη σχέση μεταξύ των εισόδων και των εξόδων, ώστε να προβλέπουν σωστά την έξοδο για νέες, άγνωστες εισόδους. Τυπικές μέθοδοι περιλαμβάνουν τη Γραμμική και Λογιστική Παλινδρόμηση, τις Μηχανές Διανυσμάτων Υποστήριξης, τα Δέντρα Απόφασης και τα Νευρωνικά Δίκτυα.

Μη Επιβλεπόμενη Μάθηση (Unsupervised Learning)

Στη μη επιβλεπόμενη μάθηση, τα δεδομένα εκπαίδευσης περιέχουν μόνο εισόδους χωρίς αντίστοιχες εξόδους. Εδώ, ο στόχος είναι η ανακάλυψη κρυφών μοτίβων ή δομών στα δεδομένα. Κοινές τεχνικές περιλαμβάνουν την Ανάλυση Κύριων Συνιστωσών (Principal Component Analysis-PCA), την Ομαδοποίηση k-Means, την Ιεραρχική Ομαδοποίηση, τους Αυτοκωδικοποιητές (Autoencoders) και τον Αλγόριθμο DBSCAN.

Ενισχυτική Μάθηση (Reinforcement Learning)

Στην ενισχυτική μάθηση, οι πράκτορες μαθαίνουν να λαμβάνουν αποφάσεις αλληλεπιδρώντας με το περιβάλλον τους. Μέσα από ανταμοιβές ή ποινές για τις ενέργειές τους, οι πράκτορες προσπαθούν να μεγιστοποιήσουν τη συνολική ανταμοιβή, μαθαίνοντας συνεχώς από την εμπειρία τους. Δημοφιλείς μέθοδοι περιλαμβάνουν το Q-Learning και τα Deep Q-Networks (DQN).

Επιπλέον, τα προβλήματα μηχανικής μάθησης μπορούν να κατηγοριοποιηθούν με βάση το επιθυμητό αποτέλεσμα. Συγκεκριμένα:

Ταξινόμηση

Η Ταξινόμηση είναι μια μέθοδος επιβλεπόμενης μάθησης, όπου το μοντέλο προσπαθεί να προβλέψει τη σωστή ετικέτα για ένα δεδομένο σύνολο εισόδου. Στη διαδικασία αυτή, το μοντέλο εκπαιδεύεται πλήρως με τα δεδομένα εκπαίδευσης και στη συνέχεια αξιολογείται με τα δεδομένα δοκιμής, πριν χρησιμοποιηθεί για προβλέψεις σε νέα, άγνωστα δεδομένα.

Παλινδρόμηση

Η παλινδρόμηση είναι επίσης ένα πρόβλημα επιβλεπόμενης μάθησης, όπου τα αποτελέσματα είναι συνεχή και όχι διακριτά.

Συσταδοποίηση

Στη συσταδοποίηση, ένα σύνολο εισόδων διαχωρίζεται σε ομάδες ή συστάδες. Σε αντίθεση με την ταξινόμηση, οι ομάδες δεν είναι προκαθορισμένες, καθιστώντας τη συσταδοποίηση τυπικό πρόβλημα μη επιβλεπόμενης μάθησης.

2.2.3 Κατηγοριοποίηση

Όπως αναφέρθηκε προηγουμένως, η κατηγοριοποίηση ή αλλιώς ταξινόμηση αναφέρεται στην ανάθεση των δεδομένων εισόδου σε μία ή περισσότερες προκαθορισμένες κατηγορίες ή κλάσεις. Υπάρχουν διάφορες μέθοδοι για την υλοποίηση της ταξινόμησης, οι οποίες θα παρουσιαστούν παρακάτω.

Κ – Πλησιέστερους Γείτονες (K-Nearest Neighbors)

Ο αλγόριθμος k-Nearest Neighbors (k-NN) είναι μια μέθοδος κατηγοριοποίησης που βασίζεται στην εύρεση των k πλησιέστερων δειγμάτων εκπαίδευσης σε σχέση με ένα δεδομένο δείγμα ελέγχου. Κάθε δείγμα αναπαρίσταται ως ένα σημείο δεδομένων σε έναν χώρο πολλών διαστάσεων, όπου οι διαστάσεις αντιστοιχούν στα χαρακτηριστικά του δείγματος. Για να καθοριστεί η ετικέτα κατηγορίας του δείγματος ελέγχου, υπολογίζεται η απόσταση του δείγματος από τα υπόλοιπα σημεία δεδομένων στο σύνολο εκπαίδευσης, και τα k πλησιέστερα σημεία χρησιμοποιούνται για να προβλεφθεί η κατηγορία του. Στην εικόνα 2.2.3.α απεικονίζεται ένα απλό παράδειγμα κατηγοριοποίησης δύο κλάσεων, χρησιμοποιώντας τον αλγόριθμο k-NN. Οι μετρικές απόστασης που χρησιμοποιούνται συνήθως στον αλγόριθμο k-NN περιλαμβάνουν την Ευκλείδεια απόσταση, την απόσταση Manhatan, και την απόσταση Minkowski.[23]

- Ευκλείδια απόσταση (Euclidean Distance) :
  Η ευκλείδια απόσταση μεταξύ δύο σημείων x,y σε έναν χώρο μίας ή και περισσότερων διαστάσεων δίνεται από τον τύπο:

$$d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

  όπου n το πλήθος των διαστάσεων και όπου k είναι τα k-οστά χαρακτηριστικά των x,y

- Απόσταση Minkowski:
  Το μέτρο της ευκλείδιας απόστασης γενικεύεται από το μέτρο της απόστασης Minkowski, η οποία δίνεται από την παρακάτω συνάρτηση:

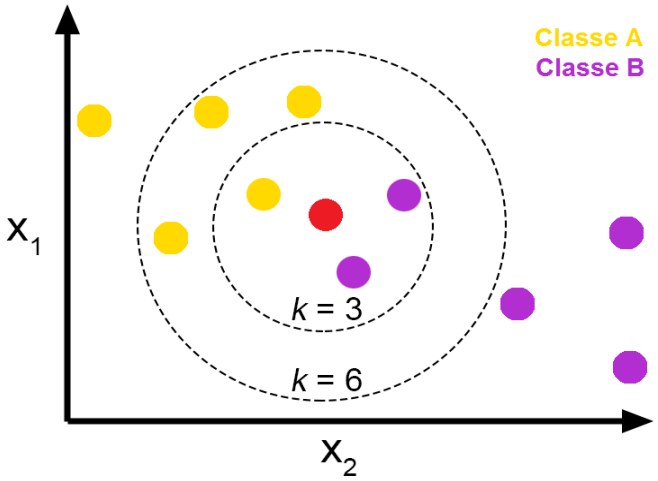
$$d(x,y) = \left(\sum_{k=1}^{n}|(x_k - y_k)|^r\right)^{\frac{1}{r}}$$

  Για r=1 , είναι η απόσταση Manhatan ή αλλιώς L_1 νόρμα
  Για r=2, είναι η ευκλείδια απόσταση ή αλλιώς L_2 νόρμα
  Για r = ∞ είναι L_{max} ή L_∞ νόρμα

Αλγόριθμος:

Ο αλγόριθμος k-Nearest Neighbors (k-NN) υπολογίζει την απόσταση (ή ομοιότητα) μεταξύ κάθε δείγματος ελέγχου και όλων των δειγμάτων εκπαίδευσης για να εντοπίσει τους πλησιέστερους γείτονες. Ωστόσο, αυτός ο υπολογισμός μπορεί να είναι υπολογιστικά απαιτητικός όταν το πλήθος των δειγμάτων ελέγχου είναι μεγάλο. Για την αντιμετώπιση αυτού του προβλήματος, υπάρχουν τεχνικές που μειώνουν τον αριθμό των υπολογισμών που απαιτούνται για τον εντοπισμό των πλησιέστερων γειτόνων.

Μόλις καθοριστεί η λίστα των πλησιέστερων γειτόνων, το δείγμα ελέγχου κατηγοριοποιείται με βάση την κατηγορία στην οποία ανήκει η πλειοψηφία αυτών των γειτόνων. Η απόφαση αυτή μπορεί να ληφθεί είτε με ίσο βάρος για κάθε γείτονα είτε με σταθμισμένη πλειοψηφία, όπου η απόσταση κάθε γείτονα από το σημείο ελέγχου επηρεάζει το βάρος της ψήφου του.



Εικόνα 2.2.3.α: Παράδειγμα για k-NN [32]

Δένδρα Απόφασης

Τα δέντρα απόφασης αποτελούν ένα ισχυρό εργαλείο για την ανάλυση και την κατηγοριοποίηση δεδομένων, βασισμένο στη δομή και τη λογική των δέντρων. Αυτοί οι αλγόριθμοι χρησιμοποιούν μια ιεραρχική προσέγγιση για να εξετάσουν τα δεδομένα, οργανώνοντας τις αποφάσεις σε μορφή δέντρου. Κάθε επίπεδο του δέντρου αντιπροσωπεύει μια απόφαση βασισμένη σε χαρακτηριστικά των δεδομένων, με τους κόμβους να απεικονίζουν χαρακτηριστικά προς ανάλυση και τα κλαδιά να αντιπροσωπεύουν τις δυνατές τιμές αυτών των χαρακτηριστικών. Τα φύλλα του δέντρου εκφράζουν τις τελικές αποφάσεις ή κατηγορίες στις οποίες κατατάσσονται τα δεδομένα.

Η δημιουργία του δέντρου περιλαμβάνει την επαναλαμβανόμενη εφαρμογή κριτηρίων που αξιολογούν την ποιότητα των διαχωρισμών στα δεδομένα. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να επιτευχθεί ένα σημείο στο οποίο δεν είναι δυνατόν να επιτευχθούν σημαντικές βελτιώσεις ή να πληρούνται οι συνθήκες τερματισμού, οι οποίες καθορίζονται μέσω διαφόρων μετρικών. Αυτές οι μετρικές παίζουν σημαντικό ρόλο στην

απόφαση για το πότε και πώς να σταματήσει η ανάπτυξη νέων κλαδιών του δέντρου, εξασφαλίζοντας ότι το τελικό μοντέλο είναι τόσο ακριβές όσο και γενικεύσιμο.[23]

Κόμβος Ρίζα

Ο κόμβος ρίζα είναι ο πρώτος και κεντρικός κόμβος ενός δέντρου απόφασης, ο οποίος περιέχει όλα τα δεδομένα εκπαίδευσης. Από αυτόν τον κόμβο ξεκινούν όλοι οι άλλοι κόμβοι, και η διαδικασία διαχωρισμού των δεδομένων σε επόμενους κόμβους βασίζεται στην ανομοιογένεια των χαρακτηριστικών.

Εσωτερικοί Κόμβοι

Οι εσωτερικοί κόμβοι, που βρίσκονται μεταξύ του κόμβου ρίζας και των φύλλων, αντιπροσωπεύουν σημεία όπου γίνονται έλεγχοι ή αποφάσεις για το διαχωρισμό των δεδομένων. Κάθε εσωτερικός κόμβος εξετάζει ένα χαρακτηριστικό και χωρίζει τα δεδομένα σε υποσύνολα με βάση τις τιμές αυτού του χαρακτηριστικού.

Φύλλα ή Τερματικοί Κόμβοι

Τα φύλλα ή τερματικοί κόμβοι είναι οι τελικοί κόμβοι του δέντρου απόφασης. Κάθε φύλλο αντιπροσωπεύει μια συγκεκριμένη κατηγορία ή ετικέτα, δηλαδή το αποτέλεσμα της απόφασης για το συγκεκριμένο μονοπάτι στο δέντρο.

Δημιουργία του Δέντρου Απόφασης

Η δημιουργία ενός δέντρου απόφασης συνήθως περιλαμβάνει τις εξής βασικές διαδικασίες:

1. **Επιλογή Χαρακτηριστικού για Διαχωρισμό:**

   Ο στόχος είναι να επιλέξουμε το καλύτερο χαρακτηριστικό για τη διάκριση των δεδομένων σε κάθε εσωτερικό κόμβο. Αυτό γίνεται με τη χρήση κριτηρίων αξιολόγησης, όπως η εντροπία, ο λόγος κέρδους (gain ratio) και ο δείκτης Gini .

2. **Διαχωρισμός των Δεδομένων:**

   Αφού επιλεγεί το χαρακτηριστικό, τα δεδομένα διαχωρίζονται σε υποσύνολα βάσει της τιμής του χαρακτηριστικού. Κάθε υποσύνολο οδηγεί σε νέο κόμβο (είτε εσωτερικό είτε φύλλο).

3. **Επανάληψη της Διαδικασίας:**

   Η διαδικασία επαναλαμβάνεται για κάθε νέο κόμβο μέχρι να πληρούνται οι συνθήκες τερματισμού. Οι συνθήκες τερματισμού μπορεί να περιλαμβάνουν την επίτευξη ενός προκαθορισμένου βάθους δέντρου, την ύπαρξη ενός ελάχιστου αριθμού παραδειγμάτων σε ένα φύλλο, ή τη μη σημαντική βελτίωση της απόδοσης.

Κριτήρια Αξιολόγησης για τον Διαχωρισμό

Τα μέτρα που αναπτύχθηκαν για την επιλογή του καλύτερου διαχωρισμού βασίζονται περισσότερο στο βαθμό ανομοιογένειας των κόμβων παιδιών. Μερικές μετρικές αναφέρονται παρακάτω:

Εντροπία (Entropy):

$$Entropy(t) = -\sum_{i=1}^{k} p(i|t) log_2 p(i|t)$$

όπου k είναι το πλήθος των διαχωρισμών και p_i είναι η πιθανότητα της κατηγορίας i στον κόμβο t

Δείκτης Gini :

$$Gini(t) = 1 - \sum_{i=1}^{k} [p(i|t)]^2$$

όπου k είναι το πλήθος των διαχωρισμών και p_i είναι η πιθανότητα της κατηγορίας i στον κόμβο t

Αναλογία Κέρδους (Gain Ratio):

Υπολογίζεται ως:

$$Gain\ Ration = \frac{Information\ Gain}{Split\ Information}$$

όπου Information Gain είναι η διαφορά στην εντροπία και όπου $Split\ Information = -\sum_{i=1}^{k} P(u_i) log_2 P(u_i)$, όπου u_i είναι οι διαφορετικές θέσεις διαχωρισμού, k είναι ο συνολικός αριθμός των διαχωρισμών και $P(u_i)$ είναι η πιθανότητα κάθε θέσης διαχωρισμού u_i. Για παράδειγμα, αν κάθε τιμή χαρακτηριστικού έχει το ίδιο πλήθος εγγραφών, τότε για κάθε i: $P(u_i) = \frac{1}{k}$.

Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines)

Η βασική ιδέα πίσω από τα Support Vector Machines (SVMs) είναι δοθέντος ενός δείγματος εκπαίδευσης να κατασκευαστεί ένα υπερεπίπεδο ως επιφάνεια απόφασης με τρόπο ώστε το περιθώριο διαχωρισμού μεταξύ θετικών και αρνητικών διανυσμάτων να μεγιστοποιείται [23] [16]. Ακολουθεί μια αναλυτική περιγραφή της λειτουργίας των SVMs:

Βασικές Αρχές των SVMs

Υπερ-Επίπεδο (Hyperplane):

Στην κλασική περίπτωση διαχωρισμού σε δύο κατηγορίες, ένα SVM προσπαθεί να βρει ένα υπερ-επίπεδο που διαχωρίζει τα δεδομένα σε δύο κατηγορίες με την καλύτερη δυνατή απόσταση (margin) μεταξύ των δεδομένων των δύο κατηγοριών.

Περιθώριο(Margin):

Το margin είναι η απόσταση μεταξύ του υπερ-επίπεδου και τα κοντινότερα σημεία κάθε κατηγορίας. Το SVM προσπαθεί να μεγιστοποιήσει αυτό το margin, έτσι ώστε το υπερ-επίπεδο να είναι όσο το δυνατόν πιο "σταθερό" και οι κατηγορίες να είναι όσο το δυνατόν πιο διακριτές.

Διανύσματα Υποστήριξης (Support Vectors):

Τα διανύσματα υποστήριξης είναι ένα κρίσιμο υποσύνολο των δεδομένων στο σύνολο εκπαίδευσης, τα οποία έχουν καθοριστική σημασία για τον προσδιορισμό της θέσης του υπερ-επίπεδου (hyperplane) σε ένα Support Vector Machine (SVM). Συγκεκριμένα, τα διανύσματα υποστήριξης είναι τα δεδομένα που βρίσκονται πιο κοντά στο υπερ-επίπεδο και επηρεάζουν άμεσα την τοποθέτησή του. Ενώ ο αλγόριθμος μάθησης χρησιμοποιεί όλα τα σημεία δεδομένων κατά τη διάρκεια της εκπαίδευσης, μόνο αυτά τα κρίσιμα σημεία, δηλαδή τα διανύσματα υποστήριξης, συμμετέχουν στον υπολογισμό του τελικού υπερ-επίπεδου. Έτσι, αυτά τα σημεία είναι ουσιαστικά για τον καθορισμό της βέλτιστης διάκρισης των κατηγοριών και τον προσδιορισμό του margin του SVM.

Διαδικασία Εκπαίδευσης

Γραμμικός Διαχωρισμός:

Για γραμμικά διαχωρίσιμα δεδομένα, το SVM προσπαθεί να βρει το υπερ-επίπεδο που μεγιστοποιεί το margin. Αν τα δεδομένα μπορούν να διαχωριστούν με μια ευθεία γραμμή (σε 2 διαστάσεις) ή μια υπερ-επίπεδη επιφάνεια (σε υψηλότερες διαστάσεις), τότε το πρόβλημα είναι γραμμικά διαχωρίσιμο. Η επίλυση του προβλήματος του SVM περιλαμβάνει τη λύση του παρακάτω προβλήματος βελτιστοποίησης:

$$Minimize\ \frac{\|w\|^2}{2}$$

υπό τους περιορισμούς:

$$y_i(w^T\ x_i + b) \geq 1$$

για όλα τα δείγματα εκπαίδευσης $\{(x_i, y_i)\}_{i=1}^N$ όπου w είναι τα βάρη και b η πόλωση

Μη Γραμμικός Διαχωρισμός:

Όταν τα δεδομένα δεν είναι γραμμικά διαχωρίσιμα, δεν είναι δυνατό να κατασκευαστεί ένα υπερεπίπεδο διαχωρισμού χωρίς σφάλματα ταξινόμησης. Ωστόσο, είναι σημαντικό να βρεθεί ένα υπερ-επίπεδο που ελαχιστοποιεί την πιθανότητα λάθους κατά την ταξινόμηση, υπολογισμένο με βάση το σύνολο των δεδομένων εκπαίδευσης. Έτσι προκύπτει το παρακάτω πρόβλημα βελτιστοποίησης:

$$Minimize\ \Phi(w, \xi) = \frac{1}{2}w^T w + C\left(\sum_{i=1}^{N} \xi_i\right)^k$$

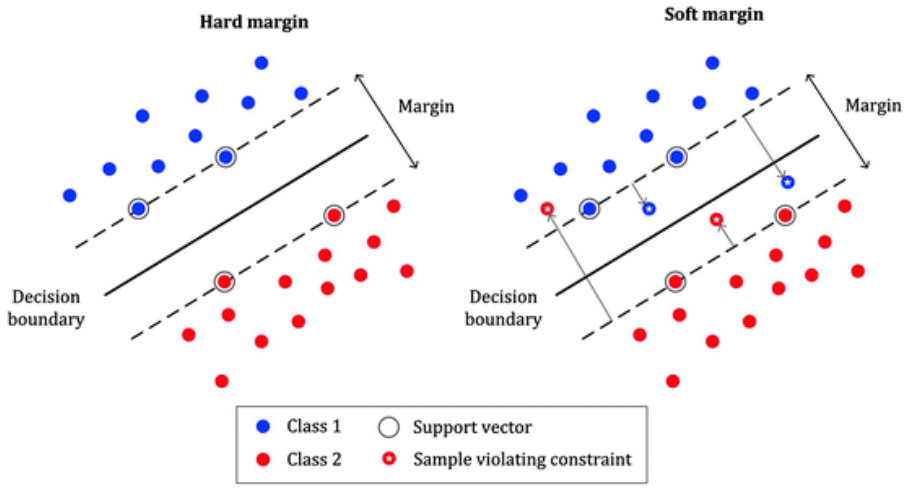
όπου ξ_i είναι οι μεταβλητές χαλάρωσης που υπολογίζουν την απόσταση ενός δεδομένου σημείου από την ιδανική συνθήκη διαχωρισιμότητας προτύπων, ενώ οι παράμετροι C και k είναι ορισμένοι από τον χρήστη και αντιπροσωπεύουν την ποινή για την εσφαλμένη κατηγοριοποίηση, υπό τους περιορισμούς:

$$y_i(w^T\ x_i + b) \geq 1 - \ \xi_i$$

για όλα τα δείγματα εκπαίδευσης $\{(x_i, y_i)\}_{i=1}^N$ όπου w είναι τα βάρη και b η πόλωση

Κατηγοριοποίηση Νέων Δεδομένων:

Όταν το μοντέλο εκπαιδευτεί, μπορεί να χρησιμοποιηθεί για να ταξινομήσει νέα δεδομένα, απλώς υπολογίζοντας σε ποια πλευρά του υπερ-επίπεδου ανήκουν τα νέα δεδομένα. Στην εικόνα 2.2.3.β, απεικονίζονται παραδείγματα για γραμμικά και μη γραμμικά SVMs.



Εικόνα 2.2.3.β: Παράδειγμα για γραμμικά και μη γραμμικά SVMs [33]

Νευρωνικά Δίκτυα

Ορισμός

Τα νευρωνικά δίκτυα (ή τεχνητά νευρωνικά δίκτυα, ANN) είναι υπολογιστικά συστήματα που εμπνέονται από τη βιολογική δομή του εγκεφάλου και ειδικότερα από τον τρόπο λειτουργίας των βιολογικών νευρώνων. Αυτά τα δίκτυα αποτελούνται από στρώματα συνδεδεμένων μονάδων, που ονομάζονται τεχνητοί νευρώνες ή κόμβοι, οι οποίοι μιμούνται τη λειτουργία των νευρώνων στον ανθρώπινο εγκέφαλο. [16] [28]

Λειτουργία

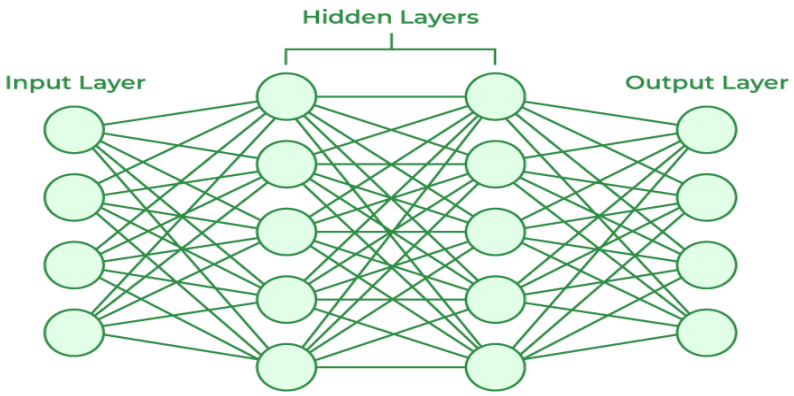
Ένα νευρωνικό δίκτυο αποτελείται από διάφορα επίπεδα (layers) νευρώνων:

Επίπεδο Εισόδου (Input Layer): Αυτό είναι το πρώτο επίπεδο, όπου οι νευρώνες λαμβάνουν τα δεδομένα εισόδου. Κάθε νευρώνας στο επίπεδο εισόδου αντιστοιχεί σε μια χαρακτηριστική τιμή από το σύνολο δεδομένων που χρησιμοποιείται για την ανάλυση ή την εκπαίδευση του δικτύου.

Κρυφά Επίπεδα (Hidden Layers): Αυτά τα ενδιάμεσα επίπεδα επεξεργάζονται τις εισερχόμενες πληροφορίες από το επίπεδο εισόδου. Οι νευρώνες στα κρυφά επίπεδα εκτελούν υπολογισμούς βασισμένους στα βάρη και τις συναρτήσεις ενεργοποίησης (activation functions), δημιουργώντας νέες αναπαραστάσεις των δεδομένων. Ένα νευρωνικό δίκτυο μπορεί να έχει ένα ή περισσότερα κρυφά επίπεδα, τα οποία το καθένα να περιλαμβάνει από ένα ή και περισσότερους νευρώνες και η ύπαρξη πολλών κρυφών επιπέδων το χαρακτηρίζει ως "βαθύ" νευρωνικό δίκτυο (deep neural network).

Επίπεδο Εξόδου (Output Layer): Το τελευταίο επίπεδο του δικτύου παράγει την τελική έξοδο ή πρόβλεψη, που μπορεί να είναι μια ταξινόμηση, μια αριθμητική τιμή ή άλλες μορφές αποτελεσμάτων, ανάλογα με την εφαρμογή του δικτύου.

Η εικόνα 2.2.3.γ παρουσιάζει την βασική δομή ενός τεχνητού νευρωνικού δικτύου.



Εικόνα 2.2.3.γ: Αναπαράσταση Νευρωνικού Δικτύου [34]

Εκπαίδευση

Τα νευρωνικά δίκτυα μαθαίνουν μέσω της διαδικασίας της εκπαίδευσης, κατά την οποία το δίκτυο προσαρμόζει τα βάρη (weights) των συνδέσεων μεταξύ των νευρώνων ώστε να ελαχιστοποιήσει το σφάλμα στις προβλέψεις του.

Συναρτήσεις Ενεργοποίησης (Activation Functions): Κάθε νευρώνας εφαρμόζει μια συνάρτηση ενεργοποίησης, η οποία καθορίζει την έξοδο του νευρώνα με βάση τα συνολικά ερεθίσματα που λαμβάνει. Οι συνηθέστερες συναρτήσεις είναι η ReLU, η sigmoid και η tanh.

Παρουσίαση Εξειδικευμένων Νευρωνικών Δικτύων

Συνελικτικά Νευρωνικά Δίκτυα

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Network-CNN) είναι μια εξελιγμένη μορφή τεχνητών νευρωνικών δικτύων (ANNs) που χρησιμοποιούνται κυρίως για την εξαγωγή χαρακτηριστικών από δεδομένα σε μορφή πλέγματος, όπως εικόνες και βίντεο. Τα CNNs είναι σχεδιασμένα για να αναγνωρίζουν και να εξάγουν σημαντικά χαρακτηριστικά από τέτοιες δομές δεδομένων, αξιοποιώντας την χωρική διάρθρωση που έχουν.

Ένα CNN συνήθως περιλαμβάνει τα εξής βασικά επίπεδα (βλ. εικόνα 2.2.3.δ):

- Convolutional Layer:
  Στο Convolutional Layer, εφαρμόζονται φίλτρα στην είσοδο του δικτύου, δηλαδή την εικόνα, για να εξάγουν χαρακτηριστικά. Κάθε φίλτρο διασχίζει την εικόνα και υπολογίζει το εσωτερικό γινόμενο μεταξύ των βαρών του φίλτρου και της αντίστοιχης περιοχής της εικόνας, δημιουργώντας έτσι έναν νέο χάρτη χαρακτηριστικών (feature map) που αποτυπώνει τα χαρακτηριστικά που ανιχνεύονται από το φίλτρο.

- Pooling Layer:
  Το Pooling Layer χρησιμοποιείται για να μειώσει τη διάσταση του χάρτη χαρακτηριστικών, μειώνοντας έτσι την υπολογιστική πολυπλοκότητα και αποτρέποντας την υπερπροσαρμογή. Συνήθως χρησιμοποιούνται τεχνικές όπως το max pooling ή το average pooling. Για παράδειγμα, η χρήση max pooling με φίλτρα 2x2 και βήμα 2 (stride 2) θα μειώσει τις διαστάσεις του χάρτη χαρακτηριστικών κατά το ήμισυ.

- ReLU Layer:
  Εφαρμόζει τη συνάρτηση ενεργοποίησης $f(x) = \max(0, x)$, η οποία εισάγει μη γραμμικότητα στο νευρωνικό δίκτυο. Αυτή η συνάρτηση αφαιρεί τις αρνητικές τιμές από έναν χάρτη ενεργοποίησης, θέτοντάς τες στο μηδέν. Άλλες συναρτήσεις που μπορούν να χρησιμοποιηθούν είναι η tahn και η sigmoid. Η ReLU έχει αποδειχθεί ότι επιτρέπει καλύτερη εκπαίδευση βαθύτερων δικτύων, ενώ προτιμάται συχνά έναντι άλλων συναρτήσεων λόγω της ταχύτερης εκπαίδευσης, χωρίς σημαντική επίπτωση στην ακρίβεια της γενίκευσης.

- Flattening:
  Στο Flattening layer, οι τελικοί χάρτες χαρακτηριστικών μετατρέπονται σε ένα μονοδιάστατο διάνυσμα, προετοιμάζοντας τα δεδομένα για τα fully connected layers.

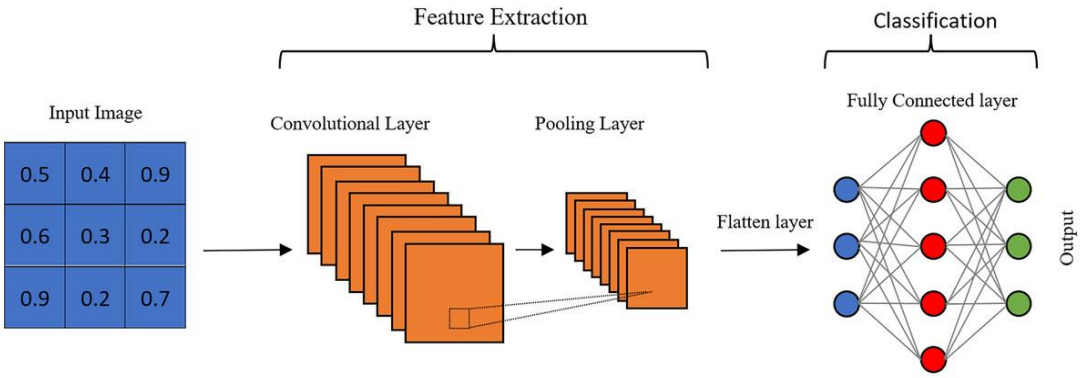
- Fully Connected Layers:
  Τα Fully Connected Layers συνδέονται με όλους τους νευρώνες της προηγούμενης στρώσης και χρησιμοποιούνται για την τελική ταξινόμηση ή πρόβλεψη. Οι έξοδοι των fully connected layers συνήθως καταλήγουν σε ένα τελικό επίπεδο εξόδου.

- Output Layer:
  Στο Output Layer, η έξοδος από τα fully connected layers εισάγεται σε μια συνάρτηση logistic, όπως η sigmoid ή η softmax, η οποία μετατρέπει την έξοδο κάθε κατηγορίας σε πιθανότητα για κάθε κατηγορία.

Λειτουργία Συνελικτικών Στρώσεων:

Τα CNNs λειτουργούν χρησιμοποιώντας φίλτρα με μικρές διαστάσεις (π.χ. 3x3 ή 5x5), τα οποία εφαρμόζονται σε ολόκληρη την είσοδο για την εξαγωγή χαρακτηριστικών. Τα φίλτρα

κινούνται πάνω από την είσοδο, και το εσωτερικό γινόμενο υπολογίζεται για κάθε θέση του φίλτρου. Το αποτέλεσμα είναι ένας νέος χάρτης χαρακτηριστικών με βάθος ίσο με τον αριθμό των φίλτρων που χρησιμοποιούνται. Καθώς το δίκτυο εκπαιδεύεται, τα φίλτρα προσαρμόζονται για να εντοπίσουν τα πιο χρήσιμα χαρακτηριστικά για την εργασία που εκτελείται. Τα CNNs είναι ιδιαίτερα αποδοτικά για αναγνώριση προτύπων σε οπτικά δεδομένα και είναι ιδιαίτερα χρήσιμα σε πολλές σύγχρονες εφαρμογές υπολογιστικής όρασης και ανάλυσης εικόνας. [19] [29]



Εικόνα 2.2.3.δ: Συνελικτικά Νευρωνικά Δίκτυα [36]

Επαναλαμβανόμενα Νευρωνικά Δίκτυα

Τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα (Recurrent Neural Networks -RNNs) είναι ένας τύπος τεχνητού νευρωνικού δικτύου που χρησιμοποιείται για την επεξεργασία σειριακών δεδομένων, δηλαδή δεδομένων όπου η χρονική σειρά ή η αλληλουχία έχει σημασία. Αυτό τα καθιστά ιδιαίτερα κατάλληλα για εφαρμογές όπως η ανάλυση κειμένου, η αναγνώριση ομιλίας και η επεξεργασία βίντεο. Σε αντίθεση με τα παραδοσιακά νευρωνικά δίκτυα, τα RNNs έχουν "μνήμη", καθώς κάθε κρυφή κατάσταση (hidden state) σε ένα RNN εξαρτάται όχι μόνο από την τρέχουσα είσοδο αλλά και από την προηγούμενη κρυφή κατάσταση. Αυτό επιτρέπει στα RNNs να "θυμούνται" πληροφορίες από προηγούμενα βήματα στην αλληλουχία.

Δομή και Αρχιτεκτονική των RNNs:

Η δομή και αρχιτεκτονική των επαναλαμβανόμενων νευρωνικών δικτύων (RNNs) χαρακτηρίζεται από μια βρόγχου δομή (loop), η οποία επιτρέπει την επιστροφή της εξόδου από έναν κόμβο της κρυφής στρώσης στην είσοδο του ίδιου κόμβου ή άλλων κόμβων της κρυφής στρώσης. Αυτή η δομή επιτρέπει στο δίκτυο να μεταδίδει πληροφορίες μέσω των χρονικών βημάτων, καθιστώντας το ικανό να επεξεργάζεται αλληλουχίες δεδομένων. Μια τυπική είσοδος σε ένα RNN είναι μια αλληλουχία δεδομένων, και το δίκτυο επεξεργάζεται κάθε στοιχείο της αλληλουχίας διαδοχικά. Καθώς προχωρά μέσω της αλληλουχίας, το RNN ενημερώνει την κρυφή του κατάσταση, η οποία λειτουργεί ως μια μορφή "μνήμης" που μεταφέρει πληροφορίες από τις προηγούμενες εισόδους, επιτρέποντας έτσι στο δίκτυο να κατανοεί και να αναλύει τη χρονική αλληλουχία των δεδομένων.[20] [30]

Κατηγορίες RNNs:

Οι κατηγορίες των Επαναλαμβανόμενα Νευρωνικών Δικτύων (RNNs) αναλύονται με βάση τους τύπους αλληλουχιών εισόδου και εξόδου. Υπάρχουν τέσσερις κύριες κατηγορίες, όπως φαίνεται και στην εικόνα 2.2.3.ε, οι οποίες περιγράφουν πώς τα RNNs μπορούν να χρησιμοποιηθούν ανάλογα με τη φύση των δεδομένων και των εφαρμογών:

1. One-to-One (Ένα-προς-Ένα)
   Ο τύπος One-to-One (Ένα-προς-Ένα) είναι ο πιο απλός, όπου αντιστοιχεί μια μοναδική είσοδος σε μια μοναδική έξοδο. Πρόκειται για έναν παραδοσιακό τύπο νευρωνικού δικτύου που δεν εκμεταλλεύεται την ακολουθιακή φύση των δεδομένων. Ένα χαρακτηριστικό παράδειγμα εφαρμογής αυτού του τύπου είναι η αναγνώριση εικόνων.

2. One-to-Many (Ένα-προς-Πολλά)
   Ο τύπος One-to-Many (Ένα-προς-Πολλά) αναφέρεται σε μια περίπτωση όπου ένα μοναδικό στοιχείο εισόδου παράγει μια αλληλουχία εξόδων. Ένα παράδειγμα εφαρμογής αυτής της κατηγορίας είναι η περιγραφή εικόνας, όπου μια μεμονωμένη εικόνα (μία είσοδος) δημιουργεί μια σειρά λέξεων (πολλαπλές έξοδοι) που περιγράφουν το περιεχόμενό της.

3. Many-to-One (Πολλά-προς-Ένα)
   Στην κατηγορία Many-to-One (Πολλά-προς-Ένα), μια αλληλουχία εισόδων παράγει μια μοναδική έξοδο. Αυτός ο τύπος είναι χρήσιμος όταν απαιτείται η αναγνώριση ή κατηγοριοποίηση βασισμένη σε μια σειρά δεδομένων. Ένα κοινό παράδειγμα εφαρμογής είναι η ανάλυση συναισθημάτων, όπου μια ακολουθία λέξεων (πολλαπλές είσοδοι) αποδίδει μια κατηγορία συναισθήματος (μία έξοδος).

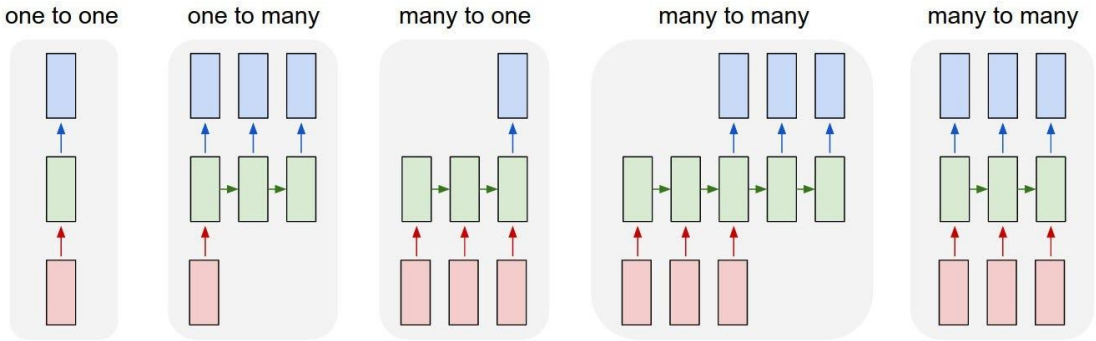
4. Many-to-Many (Πολλά-προς-Πολλά)
   Ο τύπος Many-to-Many (Πολλά-προς-Πολλά) αφορά την περίπτωση όπου μια ακολουθία εισόδων παράγει μια ακολουθία εξόδων. Υπάρχουν δύο υποκατηγορίες σε αυτήν την κατηγορία: οι ακολουθίες ίδιου μήκους, όπου οι ακολουθίες εισόδων και εξόδων έχουν το ίδιο μήκος, και οι ακολουθίες διαφορετικού μήκους, όπου το μήκος της εισόδου και της εξόδου μπορεί να διαφέρει. Ένα παράδειγμα εφαρμογής είναι η μετάφραση κειμένου, όπου μια ακολουθία λέξεων σε μια γλώσσα (πολλές είσοδοι) μετατρέπεται σε μια ακολουθία λέξεων σε άλλη γλώσσα (πολλές έξοδοι).

Προβλήματα των RNNs:

- Vanishing and Exploding Gradient Problem (Πρόβλημα Εκμηδένισης και Εκρηκτικής Ανόδου των Gradient):
  Κατά την εκπαίδευση των RNNs με τη μέθοδο της οπισθοδιάδοσης (backpropagation), τα RNNs αντιμετωπίζουν συχνά προβλήματα με τις gradients. Εάν οι gradients γίνονται πολύ μικρές (εξασθένιση), το δίκτυο αδυνατεί να μάθει, ενώ αν γίνουν πολύ μεγάλες (έκρηξη), η εκπαίδευση μπορεί να αποτύχει.

- Long-term Dependencies (Μακροχρόνιες Εξαρτήσεις):
  Ένα από τα μεγάλα προβλήματα των παραδοσιακών RNNs είναι ότι δυσκολεύονται να διατηρήσουν πληροφορίες από προηγούμενα χρονικά βήματα

για μεγάλες αλληλουχίες. Αυτό σημαίνει ότι δεν είναι καλοί στο να θυμούνται πληροφορίες που βρίσκονται πολλά βήματα πίσω στην αλληλουχία.



Εικόνα 2.2.3.ε: Κατηγορίες των RNNs [35]

Δίκτυα Μακράς Βραχύχρονης Μνήμης

Τα Δίκτυα Μακράς Βραχύχρονης Μνήμης (Long Short-Term Memory -LSTM) δίκτυα είναι μια εξειδικευμένη μορφή επαναλαμβανόμενων νευρωνικών δικτύων (RNNs). Τα LSTM έχουν σχεδιαστεί για να αντιμετωπίζουν το πρόβλημα της "μακροχρόνιας εξάρτησης" που αντιμετωπίζουν τα παραδοσιακά RNNs, όπου η πληροφορία από προηγούμενα βήματα μπορεί να χάνεται ή να αποδυναμώνεται καθώς προχωράμε στην επεξεργασία ακολουθιών.[21] [31]

Βασική Δομή και Λειτουργία των LSTM:

Τα LSTM αποτελούνται από ένα κελί (cell), μια πύλη εισόδου (input gate), μια πύλη εξόδου (output gate) και μια forget gate, όπως απεικονίζονται και στην εικόνα 2.2.3.ζ . Αυτά τα στοιχεία συνεργάζονται για να διατηρήσουν και να τροποποιήσουν την πληροφορία μέσα στο κελί, επιτρέποντας στα LSTM να "θυμούνται" πληροφορίες για μεγάλα χρονικά διαστήματα. Αναλυτικότερα:

- Cell State:
  Είναι ο πυρήνας του LSTM που μεταφέρει πληροφορίες σε όλο το χρονικό διάστημα της ακολουθίας. Το κελί επιτρέπει την ροή της πληροφορίας χωρίς να υφίσταται μεγάλες αλλαγές. Οι πύλες (gates) μπορούν να προσθέσουν ή να αφαιρέσουν πληροφορίες από το κελί.

- Forget Gate:
  Αυτή η πύλη φροντίζει για την αφαίρεση πληροφοριών που δεν είναι πλέον χρήσιμες από την κατάσταση του κελιού. Λαμβάνει δύο εισόδους: την τρέχουσα είσοδο x_t και την έξοδο του προηγούμενου κελιού h_{t-1}. Αυτές οι είσοδοι συνδυάζονται με βάρη και bias, και το αποτέλεσμα περνά από μια συνάρτηση ενεργοποίησης, συνήθως την sigmoid. Η έξοδος της συνάρτησης είναι δυαδική: εάν είναι 0, οι σχετικές πληροφορίες διαγράφονται, ενώ εάν είναι 1, οι πληροφορίες διατηρούνται για μελλοντική χρήση.

- Input Gate:
  Η πύλη εισόδου (Input Gate) προσθέτει χρήσιμες πληροφορίες στην κατάσταση του κελιού. Αρχικά, χρησιμοποιεί τη συνάρτηση sigmoid για να ρυθμίσει και να φιλτράρει τις τιμές που πρέπει να αποθηκευτούν, με βάση τις εισόδους h_{t-1} και x_t. Στη συνέχεια, δημιουργείται ένα διάνυσμα χρησιμοποιώντας τη συνάρτηση tanh, το οποίο παράγει τιμές από -1 έως +1, περιλαμβάνοντας όλες τις δυνατές τιμές που προκύπτουν από τις εισόδους h_{t-1} και x_t. Τέλος, οι τιμές του διανύσματος πολλαπλασιάζονται με τις φιλτραρισμένες τιμές, προκειμένου να προκύψουν οι χρήσιμες πληροφορίες που θα προστεθούν στην κατάσταση του κελιού.

- Output Gate:
  Η πύλη εξόδου (Output Gate) έχει την ευθύνη εξαγωγής χρήσιμων πληροφοριών από την τρέχουσα κατάσταση του κελιού προκειμένου να παρασταθούν ως έξοδος. Αρχικά, δημιουργείται ένα διάνυσμα εφαρμόζοντας τη συνάρτηση tanh στην κατάσταση του κελιού. Στη συνέχεια, η πληροφορία ρυθμίζεται με τη χρήση της συνάρτησης sigmoid και φιλτράρεται με βάση τις τιμές που πρέπει να αποθηκευτούν, χρησιμοποιώντας τις εισόδους h_{t-1} και x_t . Τέλος, οι τιμές του διανύσματος πολλαπλασιάζονται με τις φιλτραρισμένες τιμές και το αποτέλεσμα αποστέλλεται ως έξοδος και είσοδος στο επόμενο κελί.

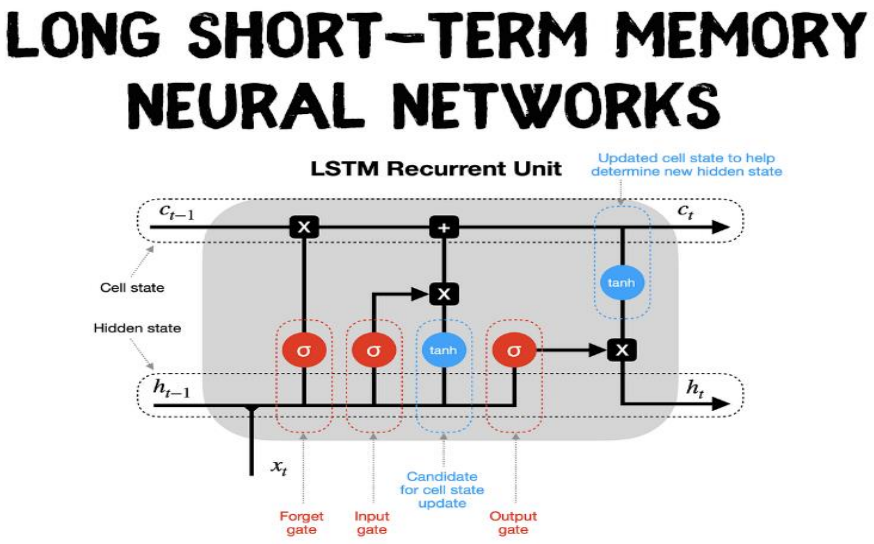
Αναλυτικά Βήματα Λειτουργίας των LSTM:

1. Απόφαση για διαγραφή (Forget Gate):
   Αρχικά, το LSTM αποφασίζει ποιες πληροφορίες από την προηγούμενη κατάσταση του κελιού θα διαγραφούν με τη χρήση της forget gate.

2. Ενημέρωση της κατάστασης του κελιού (Input Gate):
   Στη συνέχεια, αποφασίζει ποιες νέες πληροφορίες θα αποθηκευτούν στην κατάσταση του κελιού και ενημερώνει την τρέχουσα κατάσταση με αυτές τις πληροφορίες.

3. Ενημέρωση της κατάστασης του κελιού (Cell State Update):
   Η τρέχουσα κατάσταση του κελιού ενημερώνεται εν μέρει από τις πληροφορίες που αποφάσισε να κρατήσει η forget gate και τις νέες πληροφορίες που αποφάσισε να προσθέσει η input gate.

4. Απόφαση για έξοδο (Output Gate):
   Τέλος, η output gate αποφασίζει ποια τμήματα της κατάστασης του κελιού θα χρησιμοποιηθούν ως τελική έξοδος.



Εικόνα 2.2.3.ζ: Αναπαράσταση μιας μονάδας LSTM νευρωνικού δικτύου [37]

2.3 Αναγνώριση Ανθρώπινης Δραστηριότητας

2.3.1 Εισαγωγή

Η αναγνώριση ανθρώπινης δραστηριότητας είναι ένα πρόβλημα κατηγοριοποίησης, όπου ο στόχος είναι να ταξινομήσουμε ένα βίντεο στην αντίστοιχη κατηγορία του. Εφόσον η διαδικασία απαιτεί την αντιστοίχιση εισόδων με τις αντίστοιχες εξόδους κατά την εκπαίδευση, η HAR ανήκει στην κατηγορία της επιβλεπόμενης μάθησης. Είναι ένας τομέας της μηχανικής μάθησης και της επεξεργασίας σήματος που εστιάζει στην αναγνώριση των δραστηριοτήτων ή των δράσεων ενός ατόμου ή μιας ομάδας ατόμων από δεδομένα που προέρχονται από διάφορους αισθητήρες.

Η αναγνώριση δραστηριότητας επικεντρώνεται στην αναγνώριση και κατηγοριοποίηση διαφόρων ενεργειών, οι οποίες μπορεί να κυμαίνονται από απλές κινήσεις, όπως το περπάτημα ή το τρέξιμο, μέχρι πιο σύνθετες δραστηριότητες, όπως το μαγείρεμα ή η συμμετοχή σε αθλητικές εκδηλώσεις. Αυτές οι δραστηριότητες μπορούν να είναι είτε ατομικές, όπως όταν ένα άτομο περπατάει, είτε ομαδικές, όπως όταν πολλοί άνθρωποι συμμετέχουν σε ένα ομαδικό άθλημα.

Τα δεδομένα που χρησιμοποιούνται για την αναγνώριση δραστηριοτήτων προέρχονται από διάφορους αισθητήρες, οι οποίοι μπορεί να είναι φορητοί ή να είναι ενσωματωμένοι στο περιβάλλον. Αυτά τα δεδομένα υφίστανται επεξεργασία προκειμένου να εξάγουν χαρακτηριστικά που θα χρησιμοποιηθούν για την ταξινόμηση των δραστηριοτήτων.

Η αναγνώριση δραστηριοτήτων βρίσκει εφαρμογές σε πολλούς τομείς, όπως η υγειονομική περίθαλψη, όπου μπορεί να χρησιμοποιηθεί για την παρακολούθηση ηλικιωμένων και ασθενών, στην ασφάλεια για την ανίχνευση ύποπτης συμπεριφοράς και στα έξυπνα σπίτια για την αυτοματοποίηση ενεργειών με βάση τις δραστηριότητες των χρηστών.

Ωστόσο, η διαδικασία αναγνώρισης δραστηριοτήτων αντιμετωπίζει σημαντικές προκλήσεις. Η πολυπλοκότητα και η ποικιλομορφία των δραστηριοτήτων απαιτούν την εφαρμογή προηγμένων τεχνικών μηχανικής μάθησης, καθώς η ίδια δραστηριότητα

μπορεί να εκτελείται με διαφορετικούς τρόπους από διαφορετικούς ανθρώπους. Επίσης, κάποιες δραστηριότητες μπορεί να έχουν παρόμοια χαρακτηριστικά, γεγονός που καθιστά δύσκολο τον διαχωρισμό τους.

Η συγχώνευση δεδομένων από πολλαπλούς αισθητήρες αποτελεί επίσης πρόκληση, καθώς απαιτεί την ενοποίηση ετερογενών δεδομένων για την ακριβή αναγνώριση των δραστηριοτήτων. Επιπλέον, ένα σύστημα αναγνώρισης δραστηριοτήτων πρέπει να είναι ικανό να προσαρμόζεται σε νέες συνθήκες και χρήστες χωρίς να χρειάζεται επανεκπαίδευση, κάτι που απαιτεί ευελιξία και δυνατότητα γενίκευσης.

2.3.2 Τύποι Συστημάτων Αναγνώρισης Ανθρώπινης Δραστηριότητας

Αναγνώριση Δραστηριότητας με Βάση Αισθητήρες (Sensor-based Activity Recognition)

Η αναγνώριση ανθρώπινης δραστηριότητας (Human Activity Recognition - HAR) βασισμένη σε αισθητήρες περιλαμβάνει τη χρήση αισθητήρων για τη συλλογή δεδομένων που καταγράφουν τη συμπεριφορά των ανθρώπων κατά την εκτέλεση καθημερινών δραστηριοτήτων. Οι προσεγγίσεις που βασίζονται σε αισθητήρες χωρίζονται σε τρεις κύριες κατηγορίες, ανάλογα με τον τρόπο εγκατάστασης των αισθητήρων[22]:

1. *Wearable Sensors:*

   Οι αισθητήρες φοριούνται από τους χρήστες κατά τη διάρκεια της εκτέλεσης των δραστηριοτήτων. Έχει γίνει σημαντική έρευνα για την αναγνώριση δραστηριοτήτων με τη χρήση Wearable Sensors , και η αύξηση της δημοτικότητας των φορετών αισθητήρων έχει οδηγήσει στην ανάπτυξη της τεχνολογίας του Body Sensor Network (BSN). Τα BSNs συλλέγουν δεδομένα από τους φορετούς αισθητήρες και τα επεξεργάζονται για να εξάγουν χρήσιμες πληροφορίες. Αυτή η τεχνολογία έχει εφαρμογές στην υγειονομική περίθαλψη, τη φροντίδα ηλικιωμένων, την παρακολούθηση φυσικής κατάστασης και αθλητικών δραστηριοτήτων. Ωστόσο, ένα σημαντικό πρόβλημα είναι ότι η χρήση φορετών αισθητήρων μπορεί να μην είναι πάντα πρακτική, καθώς οι ηλικιωμένοι μπορεί να ξεχάσουν να φορέσουν τους αισθητήρες ή να αρνηθούν να τους φορέσουν.

2. *Αισθητήρες Συνδεδεμένοι με Αντικείμενα (Object-Tagged Sensors):*

   Σε αυτήν την προσέγγιση, οι αισθητήρες τοποθετούνται σε αντικείμενα καθημερινής χρήσης και η αναγνώριση των δραστηριοτήτων βασίζεται στην αλληλεπίδραση του χρήστη με αυτά τα αντικείμενα. Αυτή είναι μια προσέγγιση που απαιτεί από τους χρήστες να χρησιμοποιούν μόνο συγκεκριμένα αντικείμενα που φέρουν αισθητήρες. Όπως και με τα wearable sensors, αυτή η προσέγγιση μπορεί επίσης να μην είναι πάντα πρακτική, καθώς οι χρήστες περιορίζονται στη χρήση συγκεκριμένων αντικειμένων.

3. *Αισθητήρες Εγκατεστημένοι στο Περιβάλλον:*

   Στις πιο πρόσφατες προσεγγίσεις, οι αισθητήρες τοποθετούνται στο περιβάλλον και όχι πάνω στο άτομο ή στα αντικείμενα. Αυτό επιτρέπει στους χρήστες να εκτελούν τις δραστηριότητές τους χωρίς να χρειάζεται να φορούν ή να αλληλεπιδρούν με συγκεκριμένα αντικείμενα. Αυτή η προσέγγιση είναι πιο

πρακτική, καθώς δεν απαιτεί από τον χρήστη να φέρει ή να χρησιμοποιεί συγκεκριμένες συσκευές κατά την εκτέλεση των δραστηριοτήτων. Ωστόσο, υπάρχουν προκλήσεις, όπως η παρεμβολή από το περιβάλλον που μπορεί να προκαλέσει θόρυβο στα δεδομένα που καταγράφονται από τους αισθητήρες.

Αναγνώριση Δραστηριότητας με Βάση Βίντεο (Vision-based Activity Recognition)

Η αναγνώριση ανθρώπινης δραστηριότητας μέσω βίντεο αποτελεί μια προηγμένη εφαρμογή της υπολογιστικής όρασης, που χρησιμοποιεί οπτικά δεδομένα για την παρακολούθηση και ανάλυση φυσικών δραστηριοτήτων. Σε αυτήν την προσέγγιση, κάμερες τοποθετούνται σε διάφορα σημεία ενός επιβλεπόμενου χώρου, ώστε να συλλέγονται εικόνες από πολλαπλές γωνίες και να καταγράφονται οι δραστηριότητες που εκτελούνται. Τα δεδομένα μπορεί να προέρχονται είτε από ζωντανή ροή είτε από αποθηκευμένα βίντεο, τα οποία αποτελούνται από ακολουθίες εικόνων που αναπαριστούν την εκάστοτε δραστηριότητα. Μέσω τεχνικών υπολογιστικής όρασης και αλγορίθμων βαθιάς μάθησης, αυτή η μέθοδος επιτρέπει την ανάλυση βίντεο σε πραγματικό χρόνο, αναγνωρίζοντας και καταγράφοντας διάφορες ανθρώπινες ενέργειες και συμπεριφορές. Το σύστημα επεξεργάζεται τα οπτικά δεδομένα για να κατηγοριοποιεί και να εντοπίζει ενέργειες, να ανιχνεύει την έναρξη και το τέλος μιας δραστηριότητας, καθώς και να παρακολουθεί την κίνηση και τις αλληλεπιδράσεις των ανθρώπων.

Η εφαρμογή αυτής της τεχνολογίας δεν περιορίζεται μόνο σε επαγγελματικά και ελεγχόμενα περιβάλλοντα, αλλά επεκτείνεται και σε οικιακά περιβάλλοντα. Στα σπίτια, η αναγνώριση δραστηριοτήτων μπορεί να χρησιμοποιηθεί για την παρακολούθηση καθημερινών δραστηριοτήτων, την ενίσχυση της ασφάλειας και την αυτοματοποιημένη διαχείριση του σπιτιού. Ωστόσο, σε οικιακά περιβάλλοντα, η απόδοση της τεχνολογίας μπορεί να επηρεαστεί από μεταβολές στο φωτισμό και τις συνθήκες του περιβάλλοντος, όπως η παρουσία σκιάς.

Η συνεχής πρόοδος στην τεχνητή νοημοσύνη και την βαθιά μάθηση έχει ενισχύσει σημαντικά τις δυνατότητες της αναγνώρισης δραστηριοτήτων μέσω οπτικών δεδομένων. Οι αλγόριθμοι μπορούν να ενσωματώσουν προηγμένες μεθόδους, όπως η εκτίμηση θέσης και η ανάλυση βάθους, για να βελτιώσουν την ακρίβεια. Ωστόσο, η ανάγκη για μεγάλο όγκο δεδομένων εκπαίδευσης και η απαιτούμενη υπολογιστική ισχύς για την ανάλυση αυτών των δεδομένων σε πραγματικό χρόνο συνεχίζουν να αποτελούν σημαντικές προκλήσεις. Επιπλέον, η ενσωμάτωσή της σε οικιακά περιβάλλοντα απαιτεί την αντιμετώπιση διαφορών στις συνθήκες φωτισμού και την προσαρμογή σε ποικιλόμορφα περιβάλλοντα, προκειμένου να επιτευχθεί η καλύτερη δυνατή απόδοση και αξιόπιστη αναγνώριση δραστηριοτήτων.

Συνδυασμός των Προσεγγίσεων

Συχνά, τα συστήματα αναγνώρισης δραστηριότητας συνδυάζουν δεδομένα από διαφορετικές πηγές (π.χ. αισθητήρες και βίντεο) και χρησιμοποιούν τεχνικές όπως η βαθιά μάθηση για να αυξήσουν την ακρίβεια και την αξιοπιστία της αναγνώρισης. Αυτό επιτρέπει την καλύτερη κατανόηση των δραστηριοτήτων σε πραγματικό χρόνο και σε πιο ποικίλα περιβάλλοντα.

2.4 Συστήματα Επεξεργασίας Ροών Δεδομένων

2.4.1 Εισαγωγή

Τα Συστήματα Επεξεργασίας Ροών Δεδομένων (Complex Event Processing -CEP) είναι μια τεχνολογία επικεντρώνεται στην ανάλυση και επεξεργασία δεδομένων σε πραγματικό χρόνο για την αναγνώριση σύνθετων γεγονότων ή μοτίβων. Ο βασικός σκοπός του CEP είναι η ανάλυση γεγονότων και η αναγνώριση καταστάσεων που δεν μπορούν να ανιχνευθούν απλώς με τη μεμονωμένη εξέταση μεμονωμένων γεγονότων. Το CEP συνδυάζει δεδομένα και γεγονότα από διάφορες πηγές για να αναγνωρίσει πολύπλοκες καταστάσεις, εντοπίζοντας μοτίβα που υποδεικνύουν σημαντικά γεγονότα.

Η διαδικασία του CEP ξεκινά με τη συλλογή δεδομένων από ποικίλες πηγές, όπως αισθητήρες, συστήματα καταγραφής ή ροές δεδομένων. Αυτά τα δεδομένα αναλύονται σε πραγματικό χρόνο, γεγονός που επιτρέπει την αναγνώριση σύνθετων μοτίβων και γεγονότων καθώς συμβαίνουν. Μέσω αυτής της διαδικασίας, το CEP μπορεί να εντοπίσει γεγονότα που προκύπτουν από τον συνδυασμό πολλών δεδομένων, ελέγχοντας αν αυτά πληρούν προδιαγεγραμμένες συνθήκες. Μόλις τα σύνθετα γεγονότα αναγνωριστούν, το σύστημα μπορεί να αντιδράσει άμεσα, είτε στέλνοντας ειδοποιήσεις είτε ενεργοποιώντας αυτοματοποιημένες διαδικασίες.

Το CEP είναι ειδικά σχεδιασμένο για την επεξεργασία συνεχών ροών δεδομένων, δίνοντάς του τη δυνατότητα να αναλύει μεγάλες ποσότητες πληροφορίας καθώς αυτή ρέει αδιάκοπα. Μπορεί να εντοπίσει σύνθετα γεγονότα που συνδυάζουν απλά γεγονότα από διαφορετικές πηγές και είναι ιδιαίτερα ευαίσθητο στον χρόνο, επιτρέποντας στο σύστημα να αντιδρά ταχύτατα σε κρίσιμες καταστάσεις καθώς αυτές αναπτύσσονται.

Οι εφαρμογές του CEP είναι πολυάριθμες και ποικίλες. Στις χρηματοοικονομικές αγορές, χρησιμοποιείται για την ανίχνευση και ανάλυση χρηματιστηριακών συναλλαγών, εντοπίζοντας πιθανές ανωμαλίες ή ευκαιρίες. Στον τομέα της ασφάλειας και της διαχείρισης κινδύνων, παρακολουθεί και αναλύει τους κινδύνους σε πραγματικό χρόνο, ενώ στη διαχείριση ενέργειας, το CEP συμβάλλει στην παρακολούθηση της κατανάλωσης ενέργειας και την ανάλυση δεδομένων από αισθητήρες, βελτιστοποιώντας έτσι τη χρήση των πόρων. [17] [18]

2.4.2 Ανάλυση και Επεξεργασία Ροών Δεδομένων για HAR

Η συνδυασμένη χρήση Complex Event Processing (CEP) και Activity Recognition (HAR) μπορεί να προσφέρει σημαντικά πλεονεκτήματα για την παρακολούθηση ασθενών. Συγκεκριμένα, η αναγνώριση ανθρώπινης δραστηριότητας μπορεί να εντοπίζει και να αναγνωρίζει διάφορες ανθρώπινες δραστηριότητες, όπως περπάτημα, μια πτώση, και άλλες συμπεριφορές. Το CEP μπορεί να συνδυάσει αυτές τις δραστηριότητες με άλλες παραμέτρους ή δεδομένα (π.χ., ιατρικά δεδομένα, περιβαλλοντικές συνθήκες) για την αναγνώριση σύνθετων γεγονότων και ανωμαλιών. Επομένως, μπορεί να εντοπίσει αν κάτι πάει στραβά, όπως αν ο ασθενής δεν κινείται για μεγάλο χρονικό διάστημα ή παρουσιάζει ασυνήθιστες δραστηριότητες, προειδοποιώντας το προσωπικό υγείας ή ενεργοποιώντας αυτοματοποιημένες διαδικασίες

ΚΕΦΑΛΑΙΟ 3

3.ΑΝΑΛΥΣΗ ΣΥΝΑΦΩΝ ΕΡΓΑΣΙΩΝ

Η αναγνώριση ανθρώπινων δραστηριοτήτων (Human Activity Recognition - HAR) αποτελεί ένα από τα βασικά πεδία έρευνας στο χώρο της υπολογιστικής όρασης και της τεχνητής νοημοσύνης με σημαντικές εφαρμογές σε πεδία όπως η ασφάλεια, η αλληλεπίδραση ανθρώπου-υπολογιστή, και η βελτίωση της ποιότητας ζωής ασθενών και ατόμων με ειδικές ανάγκες, όπως έχει προαναφερθεί και σε προηγούμενα κεφάλαια. Είναι επομένως αναμενόμενο ότι υπάρχει ένας μεγάλος αριθμός ερευνητικών εργασιών που εξετάζουν διαφορετικές τεχνικές αναγνώρισης δραστηριοτήτων, συγκρίνοντας την αποδοτικότητά τους και προσπαθώντας να βελτιώσουν τα υπάρχοντα μοντέλα. Οι μετρικές αξιολόγησης που χρησιμοποιούνται περιλαμβάνουν την ακρίβεια της αναγνώρισης, τη δυνατότητα σε πραγματικό χρόνο επεξεργασίας, καθώς και την ικανότητα των συστημάτων να γενικεύουν σε νέες και άγνωστες δραστηριότητες.

Μια ενδιαφέρουσα εργασία είναι αυτή με τίτλο "Vision-based human activity recognition: a survey" [24], η οποία παρέχει μια εκτενή ανασκόπηση των τεχνικών αναγνώρισης ανθρώπινων δραστηριοτήτων βασισμένες σε οπτικά δεδομένα, όπως εικόνες και βίντεο. Η μελέτη κατηγοριοποιεί τις μεθόδους αναγνώρισης δραστηριοτήτων σε δύο κύριες κατηγορίες: τις μεθόδους που βασίζονται σε handcrafted features και τις feature learning μεθόδους. Αναλύεται η αποτελεσματικότητα κάθε προσέγγισης, με παρουσίαση των πλεονεκτημάτων και των περιορισμών τους. Η εργασία διαχωρίζει τις δραστηριότητες σε δύο κατηγορίες: τις στατικές (που περιγράφονται με βάση την κατεύθυνση και τη θέση στο χώρο) και τις δυναμικές (που περιγράφονται ως κινήσεις των στατικών δραστηριοτήτων). Εξετάζονται μέθοδοι όπως οι μέθοδοι βασισμένες σε σχήματα (shape-based methods), οι μέθοδοι βασισμένες σε κίνηση (motion-based methods) και οι υβριδικές μέθοδοι (hybrid methods).Η εργασία αξιολογεί την απόδοση των τεχνικών σε διάφορα datasets που περιλαμβάνουν εικόνες και βίντεο, με ποικιλία δραστηριοτήτων από απλές κινήσεις, όπως το περπάτημα, έως πιο σύνθετες αλληλεπιδράσεις μεταξύ ατόμων. Τα αποτελέσματα της μελέτης προσφέρουν σημαντικές πληροφορίες για την επιλογή της κατάλληλης μεθόδου αναγνώρισης σε διαφορετικά σενάρια. Τέλος, τονίζονται προκλήσεις όπως η κατανόηση καθημερινών δραστηριοτήτων σε μεγάλα βίντεο, η ακριβής αναγνώριση δραστηριοτήτων σε συστήματα επιτήρησης που απαιτούν βελτίωση της ακρίβειας σε πραγματικό χρόνο, και η μείωση του υπολογιστικού κόστους για την ευρεία προσβασιμότητα των συστημάτων.

Το άρθρο "Accelerometer-Based Human Activity Recognition for Patient Monitoring Using a Deep Neural Network" που δημοσιεύτηκε στο MDPI [25], εστιάζει στην αναγνώριση ανθρώπινων δραστηριοτήτων μέσω accelerometer με τη βοήθεια βαθιών νευρωνικών δικτύων. Το κύριο αντικείμενο της μελέτης είναι η ανάπτυξη και αξιολόγηση ενός μοντέλου βαθιάς μάθησης για την ανάλυση δεδομένων από accelerometer με στόχο την παρακολούθηση ασθενών. Το σύστημα που προτείνεται βασίζεται σε ένα βαθύ νευρωνικό δίκτυο, το οποίο εκπαιδεύεται για την αναγνώριση διαφορετικών τύπων δραστηριοτήτων μέσω δεδομένων κίνησης. Το άρθρο περιγράφει τη διαδικασία εκπαίδευσης του μοντέλου, συμπεριλαμβανομένων των χαρακτηριστικών που χρησιμοποιούνται για την εκπαίδευση, όπως οι μετρήσεις από τα accelerometer και οι τεχνικές προεπεξεργασίας δεδομένων. Η βαθιά αρχιτεκτονική του δικτύου περιλαμβάνει πολλαπλά επίπεδα που επιτρέπουν την κατηγοριοποίηση δραστηριοτήτων με βάση τα δεδομένα κίνησης. Η

αξιολόγηση του μοντέλου δείχνει υψηλή ακρίβεια στην αναγνώριση των δραστηριοτήτων, με συγκριτική ανάλυση της απόδοσης του δικτύου σε σχέση με άλλες μεθόδους αναγνώρισης δραστηριοτήτων. Το άρθρο καταγράφει τα αποτελέσματα από τις δοκιμές με δεδομένα από accelerometer και επισημαίνει τις βελτιώσεις που επιτεύχθηκαν σε σύγκριση με προηγούμενες προσεγγίσεις. Η μελέτη καταλήγει στην καταλληλότητα του βαθιού νευρωνικού δικτύου για την παρακολούθηση των δραστηριοτήτων των ασθενών, προτείνοντας περαιτέρω έρευνα για τη βελτίωση της γενίκευσης και της ευαισθησίας του μοντέλου. Η εφαρμογή της τεχνολογίας αυτής έχει σημαντικές προοπτικές για την παρακολούθηση της υγείας σε πραγματικό χρόνο και τη βελτίωση της ποιότητας ζωής των ασθενών.

Τέλος μία ενδιαφέρουσα εργασία με τίτλο "Patient Monitoring by Abnormal Human Activity Recognition Based on CNN Architecture" [26] εστιάζει στη χρήση του αλγορίθμου YOLO (You Only Look Once) για την ανίχνευση ανωμαλιών στις ανθρώπινες δραστηριότητες σε βίντεο. Η μελέτη αποσκοπεί στη βελτίωση της αναγνώρισης ανωμαλιών σε πραγματικό χρόνο, χρησιμοποιώντας ένα μεγάλο σύνολο δεδομένων με βίντεο ασθενών για εκπαίδευση του μοντέλου CNN.Ο αλγόριθμος YOLO επιλέχθηκε λόγω της ικανότητάς του να συνδυάζει την ταξινόμηση και τον εντοπισμό αντικειμένων σε ένα ενιαίο δίκτυο, επιτυγχάνοντας ταχύτητα επεξεργασίας 40-90 καρέ ανά δευτερόλεπτο και ακριβή αναγνώριση ανωμαλιών με ακρίβεια 96,8%. Ο προτεινόμενος αλγόριθμος κατόρθωσε να διακρίνει ανώμαλες δραστηριότητες με F1-Score 89,2. Η εργασία επικεντρώνεται στη διάκριση ανώμαλων δραστηριοτήτων σε βίντεο με ένα μόνο άτομο στην οπτική γωνία της κάμερας, γεγονός που περιορίζει την εφαρμογή της σε περιβάλλοντα με πολλούς ανθρώπους. Η έρευνα προτείνει την εξέταση της αναγνώρισης ανωμαλιών σε σκηνές με πολλαπλά άτομα και την ανάλυση αλληλεπιδράσεων μεταξύ τους, προκειμένου να επεκταθεί η χρησιμότητα του συστήματος σε ποικιλόμορφα περιβάλλοντα παρακολούθησης, όπως νοσοκομεία.

Συνολικά, η αναγνώριση ανθρώπινων δραστηριοτήτων αποτελεί ένα δυναμικό και αναπτυσσόμενο πεδίο, με τεράστιες δυνατότητες εφαρμογής σε ποικίλες περιοχές. Οι πρόσφατες μελέτες που αναλύθηκαν, περιλαμβάνοντας τεχνικές βασισμένες σε οπτικά δεδομένα, accelerometer και CNN αλγόριθμους, αναδεικνύουν την πρόοδο που έχει επιτευχθεί στην αναγνώριση δραστηριοτήτων, αλλά και τις προκλήσεις που παραμένουν, όπως η βελτίωση της ακρίβειας σε πραγματικό χρόνο και η προσαρμογή σε ποικιλόμορφα περιβάλλοντα. Αυτές οι εξελίξεις υπογραμμίζουν την ανάγκη για περαιτέρω έρευνα σε πιο εξειδικευμένα μοντέλα, ιδίως στον τομέα των οπτικών δεδομένων, τα οποία μπορούν να προσφέρουν αξιόπιστες λύσεις σε πραγματικές εφαρμογές. Σκοπός στα επόμενα κεφάλαια είναι να βρεθεί ένα κατάλληλο μοντέλο για την αναγνώριση ανθρώπινης δραστηριότητας μέσω οπτικών δεδομένων, ώστε να μπορεί να χρησιμοποιηθεί μελλοντικά για την παρακολούθηση ασθενών σε πραγματικό χρόνο.

ΚΕΦΑΛΑΙΟ 4

4.ΑΞΙΟΛΟΓΗΣΗ ΕΤΟΙΜΩΝ ΣΥΣΤΗΜΑΤΩΝ

4.1 Παρουσίαση Συνόλου Δεδομένων

Εισαγωγή

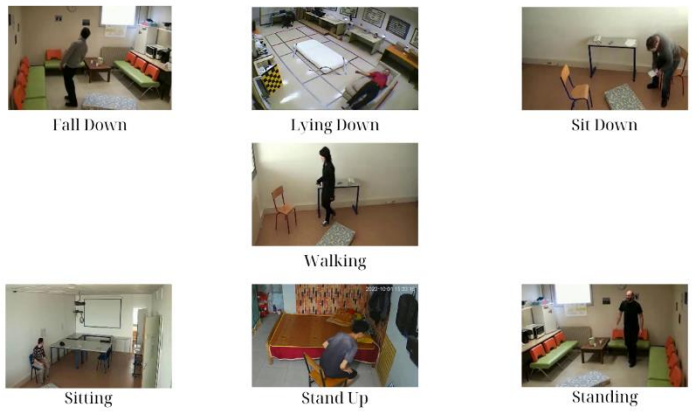
Το σύνολο δεδομένων "Video for Human Action Recognition" [10] παρέχει βίντεο για την αναγνώριση ανθρώπινων δράσεων, χρησιμοποιούμενο για την ανάπτυξη και εκπαίδευση μοντέλων τεχνητής νοημοσύνης σε εφαρμογές ανάλυσης βίντεο. Το σύνολο αυτό είναι κρίσιμο για τη δημιουργία συστημάτων που μπορούν να ανιχνεύσουν και να αναγνωρίσουν διάφορες ανθρώπινες ενέργειες σε βίντεο, με εφαρμογές σε τομείς όπως η παρακολούθηση ασθενών, η ασφάλεια και η ανάλυση δραστηριοτήτων.

Περιγραφή του Συνόλου Δεδομένων

Το σύνολο δεδομένων περιλαμβάνει συνολικά 4030 βίντεο, με διάρκεια περίπου 1-6 δευτερόλεπτα το καθένα. Τα βίντεο έχουν ανάλυση 240p-720p και είναι διαθέσιμα σε μορφή avi. Τα βίντεο συνοδεύονται από αρχεία κειμένου σε μορφή csv που περιγράφουν τις ετικέτες των δράσεων που εμφανίζονται σε κάθε βίντεο.

Κλάσεις και Ετικέτες

- *Κατηγορίες*: Το σύνολο δεδομένων περιλαμβάνει 7 κλάσεις δράσεων. Κάθε κλάση αντιστοιχεί σε μια συγκεκριμένη ανθρώπινη ενέργεια. Οι κλάσεις περιλαμβάνουν:
    - Fall Down: 415, 177 βίντεο για train και test αντίστοιχα
    - Lying Down: 380, 162 βίντεο για train και test αντίστοιχα
    - Sit Down: 187, 79 βίντεο για train και test αντίστοιχα
    - Sitting: 315, 135 βίντεο για train και test αντίστοιχα
    - Stand up: 296, 126 βίντεο για train και test αντίστοιχα
    - Standing: 566,242 βίντεο για train και test αντίστοιχα
    - Walking: 665, 285 βίντεο για train και test αντίστοιχα



Εικόνα 4.1.α: Σύνολο Δεδομένων

Τέλος από το dataset επιλέγουμε να χρησιμοποιήσουμε μόνο 6 από τα 7 διαθέσιμα labels, αποκλείοντας τα βίντεο με το label "standing" λόγω σφαλμάτων στην

κατηγοριοποίηση. Στη συνέχεια, μεταφέρουμε όλα τα βίντεο του train σε έναν κοινό φάκελο, κατηγοριοποιώντας τα ανάλογα με την ετικέτα τους, και κάνουμε το ίδιο και για τα βίντεο του test. Κατόπιν, τροποποιούμε τα αρχεία train.csv και test.csv ώστε να περιέχουν τα νέα ονόματα των βίντεο και την αντιστοίχιση κάθε κατηγορίας με τον αντίστοιχο αριθμό. Τέλος μετατρέπουμε τα αρχεία από avi σε mp4.

4.2 Αποτίμηση Μοντέλων

4.2.1 Εισαγωγή

Πριν ξεκινήσει η διαδικασία εκπαίδευσης των μοντέλων, ορίζονται βασικές συναρτήσεις και κλάσεις οι οποίες αφορούν την προετοιμασία, φόρτωση και επεξεργασία των δεδομένων, καθώς και την εκπαίδευση των μοντέλων.

1. Συνάρτηση custom_collate_fn

Αυτή η συνάρτηση χρησιμοποιείται για τη συγκέντρωση των δεδομένων (collating) σε batches κατά τη φόρτωση των δεδομένων από DataLoader.

- Φιλτράρισμα None αντικειμένων:

  Αρχικά, η συνάρτηση φιλτράρει τα δείγματα που είναι None στο batch. Αυτό γίνεται μέσω της filter, η οποία κρατάει μόνο τα δείγματα που δεν είναι None.

- Έλεγχος αν το batch είναι κενό:

  Αν μετά το φιλτράρισμα το batch είναι κενό, η συνάρτηση επιστρέφει None, παραλείποντας έτσι το συγκεκριμένο batch.

- Συγκέντρωση των υπόλοιπων δεδομένων:

  Αν υπάρχουν έγκυρα δεδομένα στο batch, η συνάρτηση χρησιμοποιεί την προεπιλεγμένη collate function (default_collate) για να συγκεντρώσει τα δείγματα σε ένα batch.

2. Κλάση CustomVideoDataset

Αυτή η κλάση δημιουργεί ένα προσαρμοσμένο Dataset για φόρτωση και προεπεξεργασία βίντεο από ένα σύνολο δεδομένων.

- __init__ Μέθοδος:

  Φορτώνει τις αναφορές (π.χ., τα paths των βίντεο και τις ετικέτες τους) από ένα CSV αρχείο και αποθηκεύει το root directory (root_dir) όπου βρίσκονται τα βίντεο. Αποθηκεύει τις μετασχηματιστικές λειτουργίες (transform) που θα εφαρμοστούν σε κάθε frame του βίντεο.

- __len__ Μέθοδος:

  Επιστρέφει το πλήθος των δειγμάτων στο dataset, το οποίο είναι ίσο με το πλήθος των γραμμών στο CSV αρχείο.

- __getitem__ Μέθοδος:

  Αυτή η μέθοδος παίρνει έναν δείκτη (idx) και επιστρέφει το αντίστοιχο βίντεο και την ετικέτα του. Αρχικά, φορτώνει το βίντεο μέσω της load_video μεθόδου. Αν το βίντεο έχει λιγότερα από 32 frames, επιστρέφει None (το οποίο θα φιλτραριστεί αργότερα).Αν το βίντεο έχει περισσότερα από 32 frames, γίνεται δειγματοληψία (sampling) ώστε να κρατηθούν μόνο 32 frames, χρησιμοποιώντας τη μέθοδο sample_frames.Αν υπάρχει μετασχηματισμός (transform), εφαρμόζεται σε κάθε frame του βίντεο. Τέλος, επιστρέφει το βίντεο ως tensor και την αντίστοιχη ετικέτα.

- load_video Μέθοδος:

  Φορτώνει όλα τα frames από το βίντεο σε ένα numpy array. Χρησιμοποιεί την OpenCV (cv2.VideoCapture) για να διαβάσει το βίντεο και να συλλέξει τα frames.

- sample_frames Μέθοδος:

  Δειγματοληπτεί ένα συγκεκριμένο αριθμό frames (num_frames) από το συνολικό αριθμό των frames του βίντεο. Αυτό γίνεται με ομοιόμορφη δειγματοληψία χρησιμοποιώντας τη np.linspace.

Ο παρακάτω κώδικας περιλαμβάνει τη συνάρτηση και την κλάση, μαζί με τις αντίστοιχες μεθόδους της:

```python
def custom_collate_fn(batch):
    # Filter out samples that are None
    batch = list(filter(lambda x: x is not None, batch))

    # If the batch is empty, return None to skip it
    if len(batch) == 0:
        return None

    # Use the default collate function to combine the remaining samples
    return torch.utils.data.dataloader.default_collate(batch)

class CustomVideoDataset(Dataset):
    def __init__(self, csv_file, root_dir, transform=None):
        """
        Initialize the dataset with CSV file, root directory, and optional transformations.

        Args:
            csv_file (str): Path to the CSV file containing video file names and labels.
            root_dir (str): Directory with all the videos.
            transform (callable, optional): A function/transform to apply to the videos.
        """
        self.annotations = pd.read_csv(csv_file)
        self.root_dir = root_dir
        self.transform = transform
```

Κώδικας 4.2.1.α: Παρουσίαση της συνάρτησης custom_collate_fn και ορισμός της κλάσης CustomVideoDataset και των μεθόδων της

```
def __len__(self):
    """
    Return the total number of samples in the dataset.
    """
    return len(self.annotations)

def __getitem__(self, idx):
    """
    Retrieve the video and label at the specified index.
    Args:
        idx (int): Index of the sample to retrieve.
    Returns:
        tuple: (video, label) where video is a tensor of frames and label is the corresponding label.
    """
    video_file = self.annotations.iloc[idx, 0]
    label = self.annotations.iloc[idx, 1]
    video_path = os.path.join(self.root_dir, video_file)

    video = self.load_video(video_path)

    # Check if the video has fewer than 32 frames; return None if so
    if len(video) < 32:
        return None
    # Sample 32 frames from the video if it has more than 32 frames
    elif len(video) > 32:
        video = self.sample_frames(video, num_frames=32)

    # Apply transformations if provided
    if self.transform:
        video = [transforms.ToPILImage()(frame) for frame in video]
        video = [self.transform(frame) for frame in video]
        video = torch.stack(video, dim=0)

    return video, label
```

Κώδικας 4.2.1.β: Συνέχεια των μεθόδων της κλάσης CustomVideoDataset

```
def load_video(self, path):
    """
    Load a video from the specified path.
    Args:
        path (str): Path to the video file.
    Returns:
        np.array: Array of frames extracted from the video.
    """
    video_frames = []
    cap = cv2.VideoCapture(path)
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        video_frames.append(frame)
    cap.release()
    video_frames = np.array(video_frames)
    return video_frames

def sample_frames(self, video, num_frames):
    """
    Sample a specified number of frames from the video.
    Args:
        video (np.array): Array of video frames.
        num_frames (int): Number of frames to sample.
    Returns:
        np.array: Array of sampled video frames.
    """
    total_frames = len(video)
    indices = np.linspace(0, total_frames - 1, num_frames, dtype=int)
    sampled_video = video[indices]
    return sampled_video
```

Κώδικας 4.2.1.γ: Συνέχεια των μεθόδων της κλάσης CustomVideoDataset

3. Συνάρτηση load_dataset

Η συνάρτηση load_dataset δημιουργεί τα σύνολα δεδομένων για εκπαίδευση και δοκιμή. Αρχικά, δημιουργεί ένα αντικείμενο της κλάσης CustomVideoDataset για το σύνολο εκπαίδευσης, το οποίο φορτώνει δεδομένα από ένα αρχείο CSV με ετικέτες και έναν φάκελο που περιέχει βίντεο. Κατά τη φόρτωση, εφαρμόζονται μετασχηματισμοί (όπως αλλαγή μεγέθους και κανονικοποίηση) σε κάθε καρέ του βίντεο (transform). Στη συνέχεια, δημιουργείται ένας DataLoader που διαχειρίζεται τη φόρτωση των δεδομένων σε batches με μέγεθος 8, χρησιμοποιώντας τη συνάρτηση custom_collate_fn. Τα δεδομένα ανακατεύονται σε κάθε εποχή και η φόρτωση επιταχύνεται με τη χρήση 4 παράλληλων επεξεργαστών. Ομοίως, για το σύνολο δεδομένων δοκιμής, δημιουργείται ένα αντίστοιχο αντικείμενο CustomVideoDataset και DataLoader με τα ίδια χαρακτηριστικά: μετασχηματισμοί, batch size 8, shuffle δεδομένων και 4 παράλληλοι επεξεργαστές για βέλτιστη απόδοση. Παρακάτω παρατίθεται ο κώδικας της συνάρτησης:

```python
def load_dataset(transform, custom_collate_fn, num_workers, batch_size):
    # Create an instance of the CustomVideoDataset for training
    dataset = CustomVideoDataset(
        csv_file='train.csv',  # Path to the CSV file with training labels
        root_dir='train',  # Directory containing training video files
        transform=transform  # Transformations to apply to each video frame
    )

    # Create a DataLoader for the training dataset
    dataloader = DataLoader(
        dataset,  # Dataset to load data from
        batch_size=8,  # Number of samples per batch
        shuffle=True,  # Shuffle the data at the start of each epoch
        collate_fn=custom_collate_fn,  # Custom function to handle batching of data
        num_workers=4  # Number of subprocesses to use for data loading
    )

    # Create an instance of the CustomVideoDataset for testing
    test_dataset = CustomVideoDataset(
        csv_file='test.csv',  # Path to the CSV file with testing labels
        root_dir='test',  # Directory containing testing video files
        transform=transform  # Transformations to apply to each video frame
    )

    # Create a DataLoader for the testing dataset
    test_dataloader = DataLoader(
        test_dataset,  # Dataset to load data from
        batch_size=8,  # Number of samples per batch
        shuffle=True,  # Shuffle the data at the start of each epoch
        collate_fn=custom_collate_fn,  # Custom function to handle batching of data
        num_workers=4  # Number of subprocesses to use for data loading
    )
```

Κώδικας 4.2.1.δ: Ορισμός της συνάρτησης load_dataset

4. Συνάρτηση train_model

Η συνάρτηση train_model εκπαιδεύει ένα μοντέλο και αξιολογεί την απόδοσή του σε σύνολα δεδομένων εκπαίδευσης και δοκιμής. Ως είσοδο δέχεται το μοντέλο (model), το σύνολο δεδομένων εκπαίδευσης (dataloader), το σύνολο δοκιμής (test_dataloader), τη συνάρτηση απώλειας (criterion), τον βελτιστοποιητή (optimizer), τον αριθμό των εποχών (num_epochs) και τη μεταβλητή slowfast_alpha για την εκπαίδευση του μοντέλου SlowFast. Σε κάθε εποχή, το μοντέλο τίθεται σε λειτουργία εκπαίδευσης, και επεξεργάζεται κάθε batch δεδομένων από τον dataloader. Τα δεδομένα μεταφέρονται

στη σωστή συσκευή (GPU ή CPU), και αν η slowfast_alpha είναι μεγαλύτερη από το μηδέν, εφαρμόζεται το μοντέλο SlowFast. Αυτό περιλαμβάνει το διαχωρισμό του βίντεο σε δύο διαδρομές: την αργή ροή (slow path) που δειγματοληπτεί καρέ με μειωμένη συχνότητα, και τη γρήγορη ροή (fast path), που κρατά τα καρέ αμετάβλητα. Στη συνέχεια, γίνεται πρόβλεψη, υπολογίζεται η απώλεια, και οι παράμετροι του μοντέλου ενημερώνονται μέσω του βελτιστοποιητή. Τέλος, οι μετρήσεις απόδοσης (ακρίβεια, precision, recall, F1-score) καταγράφονται τόσο για τα δεδομένα εκπαίδευσης όσο και για τα δεδομένα δοκιμής. Ακολουθεί ο κώδικας της συνάρτησης:

```python
def train_model(model, dataloader, test_dataloader, criterion, optimizer, num_epochs, slowfast_alpha):
    # Lists to store accuracy and other metrics
    train_accuracies = []
    train_precisions = []
    train_recalls = []
    train_f1_scores = []
    test_accuracies = []
    test_precisions = []
    test_recalls = []
    test_f1_scores = []
    st_time = time.time()  # Start time of training

    # Training loop
    for epoch in range(num_epochs):
        model.train()  # Set model to training mode
        running_loss = 0.0
        correct_predictions = 0
        total_samples = 0
        all_labels = []
        all_predictions = []
        # Training process
        for batch in tqdm(dataloader, desc=f'Epoch {epoch+1}/{num_epochs}', unit='batch'):
            if batch is None:
                continue  # Skip empty batches
            videos, labels = batch
            videos = videos.to(device)  # Move videos to the appropriate device (GPU/CPU)
            labels = labels.to(device)  # Move labels to the appropriate device
            # Rearrange dimensions from [batch_size, frames, channels, height, width] to [batch_size, channels, frames, height, width]
            videos = videos.permute(0, 2, 1, 3, 4)  # Permute dimensions
            optimizer.zero_grad()  # Zero the parameter gradients
            if slowfast_alpha > 0:
                fast_path = videos  # Fast path: frames remain unchanged
                # Slow path: Uses frames with reduced frequency (slowfast_alpha is usually 4)
                slow_path = videos[:, :, ::slowfast_alpha, :, :]
                # Create a list of tensors as required by the model
                videos = [slow_path, fast_path]
            outputs = model(videos)  # Forward pass
            loss = criterion(outputs, labels)  # Compute loss
            loss.backward()  # Backward pass
            optimizer.step()  # Update model parameters
            running_loss += loss.item()  # Accumulate loss
            # Calculate accuracy
            _, predicted = torch.max(outputs, 1)  # Get predicted class
            correct_predictions += (predicted == labels).sum().item()  # Count correct predictions
            total_samples += labels.size(0)  # Total number of samples
            all_labels.extend(labels.cpu().numpy())  # Store true labels
            all_predictions.extend(predicted.cpu().numpy())  # Store predicted labels
```

Κώδικας 4.2.1.ε: Ορισμός της συνάρτησης train_model

```python
        # Calculate training metrics
        train_accuracy = 100 * correct_predictions / total_samples
        train_accuracies.append(train_accuracy)
        train_precision = precision_score(all_labels, all_predictions, average='macro', zero_division=0)
        train_precisions.append(train_precision)
        train_recall = recall_score(all_labels, all_predictions, average='macro', zero_division=0)
        train_recalls.append(train_recall)
        train_f1 = f1_score(all_labels, all_predictions, average='macro', zero_division=0)
        train_f1_scores.append(train_f1)

        # Evaluate on the test set
        model.eval()  # Set model to evaluation mode
        correct_predictions = 0
        total_samples = 0
        all_labels = []
        all_predictions = []

        # Use tqdm for progress bar
        with torch.no_grad():  # Disable gradient calculation
            for batch in tqdm(test_dataloader, desc='Evaluating', unit='batch'):
                if batch is None:
                    continue  # Skip empty batches
                videos, labels = batch
                videos = videos.to(device)  # Move videos to the appropriate device
                labels = labels.to(device)  # Move labels to the appropriate device
                videos = videos.permute(0, 2, 1, 3, 4)  # Permute dimensions
                if slowfast_alpha > 0:
                    fast_path = videos  # Fast path: frames remain unchanged
                    # Slow path: Uses frames with reduced frequency (slowfast_alpha is usually 4)
                    slow_path = videos[:, :, ::slowfast_alpha, :, :]
                    # Create a list of tensors as required by the model
                    videos = [slow_path, fast_path]
                outputs = model(videos)  # Forward pass
                _, predicted = torch.max(outputs, 1)  # Get predicted class
                correct_predictions += (predicted == labels).sum().item()  # Count correct predictions
                total_samples += labels.size(0)  # Total number of samples
                all_labels.extend(labels.cpu().numpy())  # Store true labels
                all_predictions.extend(predicted.cpu().numpy())  # Store predicted labels

        # Calculate test metrics
        test_accuracy = 100 * correct_predictions / total_samples
        test_accuracies.append(test_accuracy)
        test_precision = precision_score(all_labels, all_predictions, average='macro', zero_division=0)
        test_precisions.append(test_precision)
        test_recall = recall_score(all_labels, all_predictions, average='macro', zero_division=0)
        test_recalls.append(test_recall)
        test_f1 = f1_score(all_labels, all_predictions, average='macro', zero_division=0)
        test_f1_scores.append(test_f1)
```

Κώδικας 4.2.1.ζ: Συνέχεια της συνάρτησης train_model

4.2.2 Αξιολόγηση 1ου Μοντέλου

Παρουσίαση Μοντέλου

Το μοντέλο R(2+1)D[11] είναι ένα νευρωνικό δίκτυο που σχεδιάστηκε ειδικά για την ανάλυση βίντεο, συνδυάζοντας τη δύναμη των δικτύων 2D συνελικτικών (CNNs) με τα 3D συνελικτικά δίκτυα (3D CNNs). Χρησιμοποιεί μια στρατηγική που ονομάζεται "Factorized Convolutions" για να βελτιώσει την απόδοση και να μειώσει την υπολογιστική πολυπλοκότητα.

- **Αρχιτεκτονική του R(2+1)D:** Η βασική ιδέα πίσω από το R(2+1)D είναι η διάσπαση των 3D συνελικτικών στρωμάτων σε 2D και 1D συνελικτικά στρώματα:
    - *2D Convolutional Layers*: Τα πρώτα στρώματα του μοντέλου χρησιμοποιούν 2D συνελικτικά φίλτρα για την εξαγωγή χαρακτηριστικών από κάθε καρέ του βίντεο ξεχωριστά. Αυτά τα στρώματα επεξεργάζονται την χωρική πληροφορία εντός κάθε καρέ.
    - *1D Temporal Convolutional Layers*: Στη συνέχεια, χρησιμοποιούνται 1D συνελικτικά φίλτρα για την επεξεργασία της χρονικής διάστασης, δηλαδή την ανάλυση της αλλαγής μεταξύ διαδοχικών καρέ. Αυτό βοηθά στο να μάθει το μοντέλο πώς οι ενέργειες εξελίσσονται με την πάροδο του χρόνου.

Η αρχιτεκτονική R(2+1)D είναι, επομένως, αποτελεσματική στη συνδυασμένη χρήση των χαρακτηριστικών που εξάγονται από τα καρέ των βίντεο και τις χρονικές σχέσεις μεταξύ αυτών.

- **Λειτουργία του R(2+1)D**
    - *Εξαγωγή Χωρικών Χαρακτηριστικών*: Το μοντέλο αρχικά χρησιμοποιεί 2D συνελικτικά φίλτρα για την ανάλυση της χωρικής πληροφορίας σε κάθε καρέ του βίντεο. Αυτά τα φίλτρα ανιχνεύουν χαρακτηριστικά όπως άκρα, υφές, και σχήματα μέσα σε κάθε καρέ. Στη συνέχεια, τα 1D συνελικτικά φίλτρα αναλύουν την αλλαγή αυτών των χαρακτηριστικών μέσω του χρόνου, δηλαδή αναγνωρίζουν πώς τα χαρακτηριστικά αλλάζουν μεταξύ διαδοχικών καρέ.
    - *Συνδυασμένη Ανάλυση*: Η συνδυασμένη χρήση 2D και 1D συνελικτικών φίλτρων επιτρέπει στο μοντέλο να κατανοήσει καλύτερα τις σύνθετες ενέργειες και τις αλληλεπιδράσεις στο βίντεο.

- **Πλεονεκτήματα του R(2+1)D**
    - *Μειωμένη Υπολογιστική Πολυπλοκότητα*: Το R(2+1)D συνδυάζει τη δύναμη των 3D CNNs με τη μειωμένη πολυπλοκότητα των 2D και 1D CNNs. Αυτό καθιστά το μοντέλο πιο αποδοτικό σε υπολογιστικούς πόρους.
    - Αυξημένη Απόδοση: Η διάσπαση των συνελικτικών στρωμάτων βοηθά στην καλύτερη κατανόηση τόσο των χωρικών όσο και των χρονικών σχέσεων, βελτιώνοντας την ακρίβεια της ανάλυσης βίντεο.

- **Μειονεκτήματα**

  Ένα μειονέκτημα του μοντέλου R(2+1)D είναι ότι απαιτεί σταθερό αριθμό εισόδων (Frames), γεγονός που το καθιστά μη ευέλικτο όσον αφορά την προσαρμογή σε διαφορετικά μήκη ακολουθιών βίντεο. Αυτό σημαίνει ότι δεν μπορεί να προσαρμοστεί αυτόματα σε βίντεο με διαφορετικό αριθμό καρέ, περιορίζοντας έτσι την ευελιξία του σε εφαρμογές με μεταβαλλόμενη είσοδο.

Fine-tuning

Ο κώδικας ξεκινά με τη φόρτωση του προεκπαιδευμένου μοντέλου R(2+1)D-18 από τη βιβλιοθήκη torchvision, το οποίο έχει εκπαιδευτεί στο σύνολο δεδομένων Kinetics-400 και αποτελείται από 18 στρώματα. Τα αρχικά στρώματα του μοντέλου "παγώνουν", διατηρώντας τα προκαθορισμένα χαρακτηριστικά τους χωρίς να επιτρέπεται η εκπαίδευση τους. Στη συνέχεια, το τελικό επίπεδο ταξινόμησης τροποποιείται ώστε να προσαρμοστεί για ταξινόμηση σε 6 κατηγορίες.

```
# Load the pretrained R(2+1)D-18 model for video action recognition
model = torchvision.models.video.r2plus1d_18(pretrained=True)

# Freeze the parameters of the initial layers to prevent updates during training
for param in model.parameters():
    param.requires_grad = False

# Modify the final classification layer
model.fc = nn.Linear(model.fc.in_features, out_features=6)

# Move the model to the GPU if available, otherwise use the CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Print the model's architecture to review changes
print(model)
```

Κώδικας 4.2.2.α: Φόρτωση προεκπαιδευμένου μοντέλου

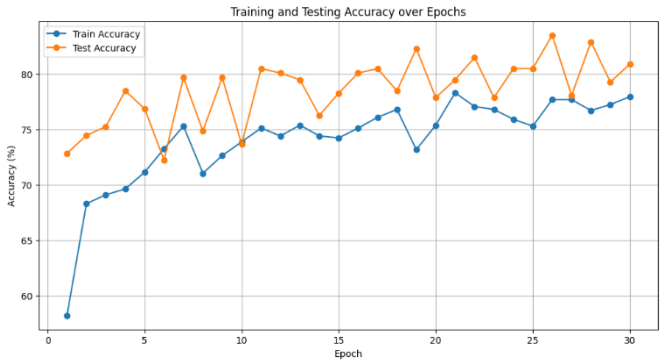
Ο κώδικας ορίζει μετασχηματισμούς για τα καρέ βίντεο [12]. Συγκεκριμένα, αλλάζει το μέγεθος των frames σε 128x171 pixels, κάνει κεντρικό crop στα frames για να έχουν μέγεθος 112x112 pixels, μετατρέπει τα frames σε tensors και τα κλιμακώνει στο εύρος [0, 1], και τέλος κανονικοποιεί τα frames σύμφωνα με τις συγκεκριμένες μέσες τιμές και τυπικές αποκλίσεις που χρησιμοποιεί το μοντέλο. Οι μετασχηματισμοί αυτοί υλοποιούνται στον παρακάτω κώδικα:

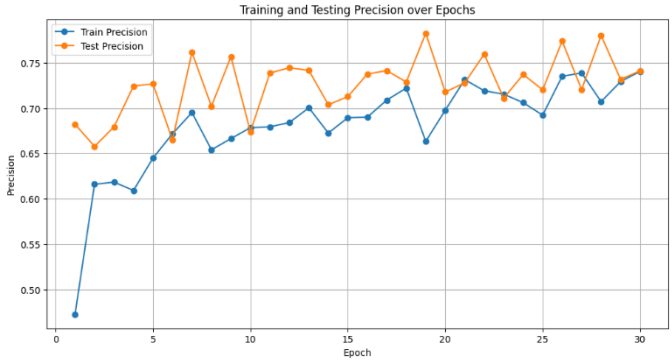
```
# Create transformations for the video frames
transform = transforms.Compose([
    transforms.Resize(size=[128, 171], interpolation=InterpolationMode.BILINEAR),  # Resize the frames
    transforms.CenterCrop(size=[112, 112]),  # Apply central crop
    transforms.ToTensor(),  # Convert frames to tensors and scale to [0, 1]
    transforms.Normalize(mean=[0.43216, 0.394666, 0.37645], std=[0.22803, 0.22145, 0.216989]),  # Normalize
])
```

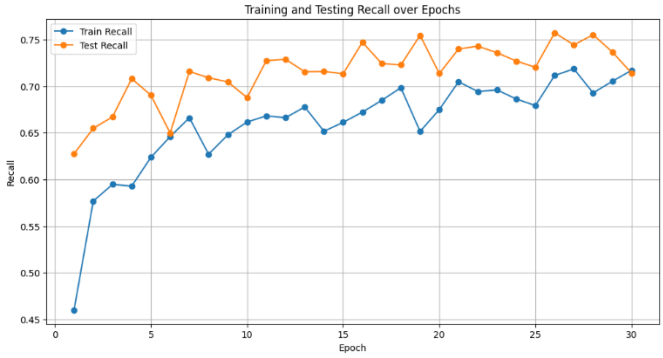
Κώδικας 4.2.2.β: Ορισμός των μετασχηματισμών για τα καρέ των βίντεο

Στη συνέχεια, φορτώνεται το σύνολο δεδομένων χρησιμοποιώντας την συνάρτηση load_dataset και δημιουργούνται οι αντίστοιχοι dataloaders για την εκπαίδευση και τη δοκιμή, χρησιμοποιώντας προκαθορισμένες μετατροπές (transform), συνάρτηση συγκέντρωσης δεδομένων (custom_collate_fn) για τη διαχείριση των batches, με μέγεθος batch_size = 8 και 4 παράλληλους επεξεργαστές (num_workers = 4). Η συνάρτηση απώλειας ορίζεται ως CrossEntropyLoss, κατάλληλη για προβλήματα ταξινόμησης, ενώ ως βελτιστοποιητής χρησιμοποιείται ο Adam με ρυθμό μάθησης 0.001. Ο αριθμός των εποχών εκπαίδευσης καθορίζεται στις 30. Τέλος, η συνάρτηση train_model καλείται για την εκπαίδευση του μοντέλου με τα δεδομένα του dataloader, αξιοποιώντας την καθορισμένη συνάρτηση απώλειας και τον βελτιστοποιητή, και με slowfast_alpha=0, ενώ υπολογίζονται δείκτες απόδοσης όπως ακρίβεια, precision, recall και F1 score τόσο για την εκπαίδευση όσο και για τη δοκιμή.

Αποτελέσματα Μοντέλου







Εικόνα 4.2.2.α: Train and Test accuracy over epochs for R(2+1)D

Εικόνα 4.2.2.β: Train and Test precision over epochs for R(2+1)D

Εικόνα 4.2.2.γ: Train and Test recall over epochs for R(2+1)D

Εικόνα 4.2.2.δ: Train and Test F1-score over epochs for R(2+1)D

	10 Εποχές	20 Εποχές	30 Εποχές
Train Accuracy	69.00%	74.49%	74.22%
Train Precision (macro)	0.63	0.68	0.69
Train Recall(macro)	0.58	0.65	0.66
Train F1-score(macro)	0.59	0.66	0.67
Test Accuracy	78.47%	82.70%	82.49%
Test Precision(macro)	0.80	0.77	0.81
Test Recall(macro)	0.70	0.75	0.73
Test F1-score(macro)	0.70	0.76	0.75

Πίνακας 1: Αποτελέσματα του μοντέλου R(2+1)D

Χρόνος εκτέλεσης: 61 min

4.2.3 Αξιολόγηση 2ου Μοντέλου

Παρουσίαση Μοντέλου

Το μοντέλο SlowFast [38] για αναγνώριση βίντεο είναι μια αρχιτεκτονική που συνδυάζει δύο διαφορετικές ταχύτητες ανάλυσης για να εκμεταλλευτεί καλύτερα τις χωρικές και χρονικές πληροφορίες σε βίντεο. Αναλυτικότερα:

Διπλή Ροή (Two-Stream method)

Το SlowFast χρησιμοποιεί δύο παράλληλα δίκτυα (streams) για να αναλύσει το βίντεο:

- Αργή Ροή (Slow Pathway): Επικεντρώνεται στη λεπτομερή ανάλυση των καρέ, αλλά με χαμηλότερο ρυθμό καρέ (frame rate). Αυτό επιτρέπει την εκμετάλλευση των χωρικών πληροφοριών με μεγάλη ακρίβεια, διότι το δίκτυο εξετάζει πιο αναλυτικά κάθε καρέ.
- Γρήγορη Ροή (Fast Pathway): Εξετάζει τα καρέ με υψηλότερο ρυθμό (frame rate) και έχει χαμηλότερη χωρική ανάλυση. Αυτό επιτρέπει την καλή ανάλυση των γρήγορων κινήσεων και των χρονικών αλλαγών που συμβαίνουν γρήγορα στο βίντεο.

Ενοποίηση Χωρικών και Χρονικών Πληροφοριών

Η βασική ιδέα του SlowFast είναι να συνδυάσει αυτές τις δύο ροές για να επωφεληθεί τόσο από τις λεπτομερείς χωρικές πληροφορίες όσο και από τις γρήγορες χρονικές μεταβολές:

- Η Αργή Ροή εστιάζει στην κατανόηση των μακροχρόνιων και στατικών χαρακτηριστικών του βίντεο.
- Η Γρήγορη Ροή εστιάζει στις γρήγορες αλλαγές και κινήσεις στο βίντεο.

Τα αποτελέσματα από τις δύο ροές συνδυάζονται σε ένα κοινό επίπεδο, το οποίο επιτρέπει στο μοντέλο να κατανοεί και να αναγνωρίζει τις αλληλεπιδράσεις και τις κινήσεις με μεγαλύτερη ακρίβεια.

Αρχιτεκτονική

- Αργή Ροή: Χρησιμοποιεί συνήθως ένα κλασικό σύνολο αρχιτεκτονικών CNNs (όπως ResNet) με χαμηλό ρυθμό καρέ για να ενσωματώσει λεπτομερείς χωρικές πληροφορίες.
- Γρήγορη Ροή: Χρησιμοποιεί ένα άλλο σύνολο CNNs με υψηλό ρυθμό καρέ, που είναι λιγότερο λεπτομερή χωρικά αλλά παρέχουν πιο γρήγορη ανάλυση των χρονικών πληροφοριών.

Συνδυασμός και Εκπαίδευση

Τα δεδομένα από τις δύο ροές συνδυάζονται προκειμένου να δημιουργηθεί ένα ενιαίο χαρακτηριστικό που περιλαμβάνει τόσο τις χωρικές όσο και τις χρονικές πληροφορίες. Το μοντέλο εκπαιδεύεται σε αυτή τη συνδυασμένη αναπαράσταση για να κάνει προβλέψεις σχετικά με τις δραστηριότητες ή τις κατηγορίες του βίντεο.

Fine-tuning

Ο κώδικας ξεκινά με τη φόρτωση του προεκπαιδευμένου μοντέλου SlowFast R50. Τα αρχικά στρώματα του μοντέλου "παγώνουν" και το τελικό επίπεδο ταξινόμησης τροποποιείται ώστε να προσαρμοστεί για ταξινόμηση σε 6 κατηγορίες.

```
# Load the pretrained SlowFast R50 model
model = slowfast_r50(pretrained=True)

# Freeze the parameters of the initial layers
for param in model.parameters():
    param.requires_grad = False

# Modify the final classification layer
# Access the 'blocks' module and the last block within it
model.blocks[6].proj = nn.Linear(model.blocks[6].proj.in_features, out_features=6)

# Move the model to the GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Print the model's architecture to review changes
print(model)
```

Κώδικας 4.2.3.α: Φόρτωση προεκπαιδευμένου μοντέλου SlowFast

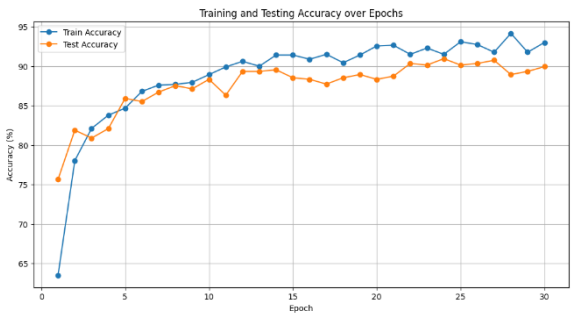
Ο κώδικας ορίζει μετασχηματισμούς για τα καρέ βίντεο [39]. Η μεταβλητή slowfast_alpha ορίζεται με τιμή 4 και χρησιμοποιείται για τον καθορισμό του ρυθμού δειγματοληψίας των καρέ στο πλαίσιο του μοντέλου SlowFast. Συγκεκριμένα, αυτή η παράμετρος ελέγχει πόσο πιο αραιά δειγματοληπτούνται τα καρέ στην «αργή» ροή του μοντέλου, επιτρέποντας την επεξεργασία καρέ με χαμηλότερη συχνότητα, ενώ παράλληλα διατηρείται η «γρήγορη» ροή με πλήρη συχνότητα καρέ.

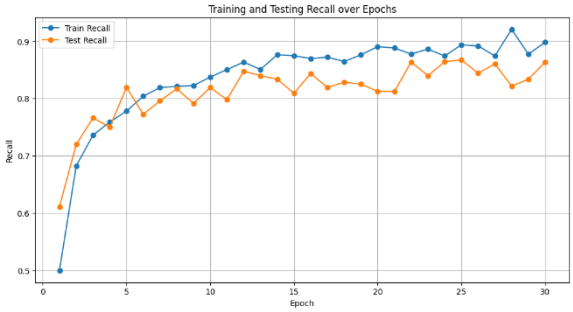
```python
# Create transformations for the video frames
slowfast_alpha = 4
transform = transforms.Compose([
    transforms.Resize(size=[256, 256]),  # Resize the frames
    transforms.CenterCrop(size=256),  # Central crop
    transforms.ToTensor(),  # Convert frames to tensor and scale to [0, 1]
    transforms.Normalize(mean=[0.45, 0.45, 0.45], std=[0.225, 0.225, 0.225]),  # Normalize
])
```

Κώδικας 4.2.3.β: Ορισμός των μετασχηματισμών για τα καρέ των βίντεο

Στη συνέχεια, φορτώνεται το σύνολο δεδομένων και δημιουργούνται οι αντίστοιχοι dataloaders για την εκπαίδευση και τη δοκιμή. Τέλος, καλείται η συνάρτηση train_model για την εκπαίδευση του μοντέλου με τα δεδομένα του dataloader, την συνάρτηση απώλειας και τον βελτιστοποιητή και υπολογίζονται οι δείκτες απόδοσης.

Αποτελέσματα Μοντέλου





Εικόνα 4.2.3.α: Train and Test accuracy over epochs for SlowFast-R50

Εικόνα 4.2.3.β: Train and Test precision over epochs for SlowFast-R50

Εικόνα 4.2.3.γ: Train and Test recall over epochs for SlowFast-R50

Εικόνα 4.2.3.δ: Train and Test F1-score over epochs for SlowFast-R50

	10 Εποχές	20 Εποχές	30 Εποχές
Train Accuracy	88.93%	92.56%	93.00%
Train Precision (macro)	0.85	0.89	0.91
Train Recall(macro)	0.84	0.89	0.90
Train F1-score(macro)	0.84	0.89	0.90
Test Accuracy	88.33%	88.33%	89.94%
Test Precision(macro)	0.84	0.87	0.86
Test Recall(macro)	0.82	0.81	0.86
Test F1-score(macro)	0.82	0.81	0.86

Πίνακας 2: Αποτελέσματα του μοντέλου SlowFast-R50

Χρόνος εκτέλεσης: 77 min

4.2.4 Αξιολόγηση 3ου Μοντέλου

Παρουσίαση Μοντέλου

Το μοντέλο MC3-18 (Multi-Clip 3D ResNet-18) είναι μια παραλλαγή του ResNet-18 που έχει σχεδιαστεί ειδικά για αναγνώριση βίντεο. Αυτό το μοντέλο συνδυάζει τις αρχές των 3D συνελικτικών δικτύων με την αρχιτεκτονική ResNet [41] για να επεξεργάζεται δεδομένα βίντεο, ενσωματώνοντας πληροφορίες και από τις χωρικές και χρονικές διαστάσεις των βίντεο.

Βασικά Στοιχεία του MC3-18 μοντέλου

3D Συνελικτικά Δίκτυα:

Το MC3-18 χρησιμοποιεί 3D συνελικτικά φίλτρα αντί για τα κλασικά 2D φίλτρα. Τα 3D φίλτρα επεξεργάζονται τα δεδομένα βίντεο σε τρεις διαστάσεις: δύο χωρικές (ύψος και πλάτος) και μία χρονική διάσταση. Αυτό επιτρέπει στο δίκτυο να εξάγει χαρακτηριστικά που ενσωματώνουν πληροφορίες και από τις δύο διαστάσεις.

Αρχιτεκτονική ResNet:

Το MC3-18 βασίζεται στην αρχιτεκτονική ResNet, η οποία εισάγει residual blocks. Τα residual blocks περιλαμβάνουν skip connections (παράκαμψη) που επιτρέπουν στο δίκτυο να διατηρεί πληροφορίες που διαφορετικά θα χάνονταν κατά τη διάρκεια της εκπαίδευσης.

Multi-Clip Approach:

Η "Multi-Clip" προσέγγιση αναφέρεται στο γεγονός ότι το MC3-18 χρησιμοποιεί πολλαπλά κλιπ βίντεο για να κατανοήσει καλύτερα τις χρονικές αλληλουχίες. Τα κλιπ βίντεο αναλύονται ξεχωριστά και στη συνέχεια συνδυάζονται για να δημιουργήσουν μια πλήρη αναπαράσταση του βίντεο.

Λειτουργία του MC3-18 μοντέλου

Εισαγωγή Δεδομένων:

Το μοντέλο εισάγει βίντεο ως τρισδιάστατους όγκους (3D volumes), όπου κάθε καρέ του βίντεο έχει δύο χωρικές διαστάσεις και μία χρονική διάσταση.

Εφαρμογή Συνελικτικών Επιπέδων:

Τα 3D συνελικτικά φίλτρα εφαρμόζονται στα βίντεο για την εξαγωγή χαρακτηριστικών. Αυτά τα φίλτρα μπορούν να εντοπίσουν τόσο χωρικές όσο και χρονικές δομές μέσα στο βίντεο.

Χρήση Residual Blocks:

Η αρχιτεκτονική περιλαμβάνει residual blocks που συνδυάζουν τα εξαγόμενα χαρακτηριστικά με τις εισόδους των προηγούμενων επιπέδων μέσω skip connections. Αυτό βοηθά στη διατήρηση της πληροφορίας και στην αποφυγή της εξαφάνισης των gradients κατά την εκπαίδευση.

Ενοποίηση και Κατηγοριοποίηση:

Τα χαρακτηριστικά που εξάγονται από τα διάφορα κλιπ του βίντεο συνδυάζονται για να δημιουργηθεί μια συνολική αναπαράσταση του βίντεο. Το μοντέλο χρησιμοποιεί αυτή την αναπαράσταση για την αναγνώριση δραστηριοτήτων.

Τεχνικά Στοιχεία

Δομή:

Το MC3-18 περιλαμβάνει 18 επίπεδα με 3D συνελικτικά φίλτρα. Η αρχιτεκτονική είναι σχεδιασμένη έτσι ώστε να εξάγει χαρακτηριστικά από τις τρεις διαστάσεις του βίντεο (δύο χωρικές και μία χρονική).

Εκπαίδευση και Βελτιστοποίηση:

Το μοντέλο εκπαιδεύεται χρησιμοποιώντας σύνολα δεδομένων βίντεο με στόχο την κατηγοριοποίηση ή την ανίχνευση δραστηριοτήτων. Η εκπαίδευση περιλαμβάνει τη βελτιστοποίηση των παραμέτρων του δικτύου για την καλύτερη αναγνώριση των δραστηριοτήτων.

Fine-tuning

Ο κώδικας αρχίζει με τη φόρτωση του προεκπαιδευμένου μοντέλου MC3-18. Τα πρώτα στρώματα του μοντέλου "παγώνουν" και το τελικό επίπεδο ταξινόμησης τροποποιείται για ταξινόμηση σε 6 κατηγορίες.

```python
# Load pretrained model
model = mc3_18(pretrained=True)

# Freeze the parameters of the initial layers
for param in model.parameters():
    param.requires_grad = False

# Modify the final classification layer
model.fc = nn.Linear(model.fc.in_features, out_features=6)

# Move the model to the GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Print the model's architecture to review changes
print(model)
```

Κώδικας 4.2.4.α: Φόρτωση προεκπαιδευμένου μοντέλου MC3-18

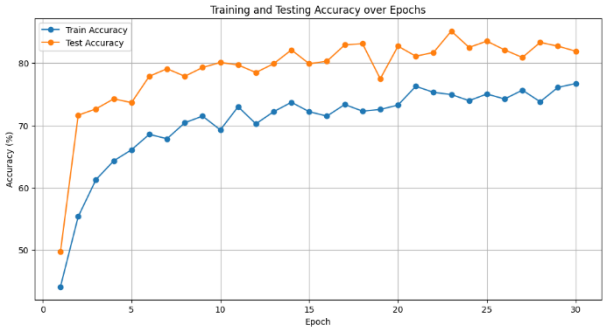
Επιπλέον ορίζονται οι παρακάτω μετασχηματισμοί για τα καρέ βίντεο [40]:

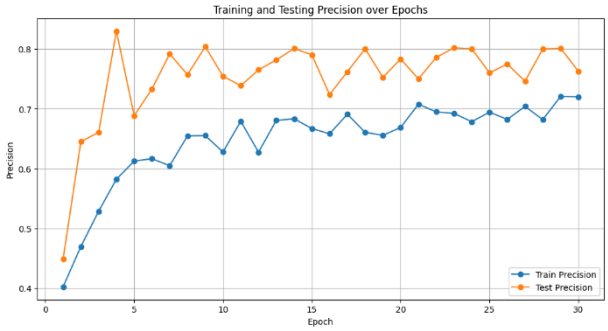
```python
# Create transformations for the video frames
transform = transforms.Compose([
    transforms.Resize(size=[128, 171], interpolation=InterpolationMode.BILINEAR),  # Resize the frames
    transforms.CenterCrop(size=[112, 112]),  # Central crop
    transforms.ToTensor(),  # Convert frames to tensor and scale to [0, 1]
    transforms.Normalize(mean=[0.43216, 0.394666, 0.37645], std=[0.22803, 0.22145, 0.216989]),  # Normalize
])
```

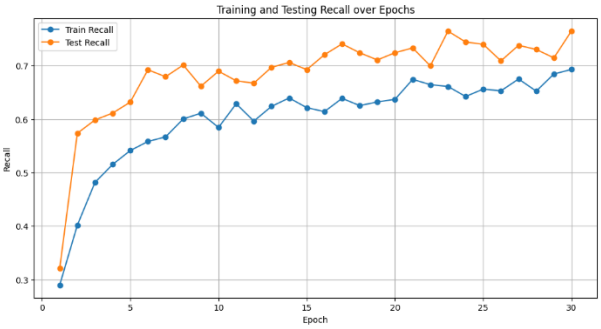
Κώδικας 4.2.4.β: Ορισμός των μετασχηματισμών για τα καρέ των βίντεο για το μοντέλο MC3-18

Αφού δημιουργηθούν τα σύνολα δεδομένων και οι αντίστοιχοι dataloaders, καλείται η συνάρτηση train_model για την εκπαίδευση του μοντέλου, χρησιμοποιώντας τα δεδομένα του dataloader, την καθορισμένη συνάρτηση απώλειας και τον βελτιστοποιητή.

Αποτελέσματα Μοντέλου







Εικόνα 4.2.4.α: Train and Test accuracy over epochs for MC3-18

Εικόνα 4.2.4.β: Train and Test precision over epochs for MC3-18

Εικόνα 4.2.4.γ: Train and Test recall over epochs for MC3-18

Εικόνα 4.2.4.δ: Train and Test F1-score over epochs for MC3-18

	10 Εποχές	20 Εποχές	30 Εποχές
Train Accuracy	69.26%	73.25%	76.71%
Train Precision (macro)	0.63	0.67	0.72
Train Recall(macro)	0.58	0.64	0.69
Train F1-score(macro)	0.59	0.65	0.70
Test Accuracy	80.08%	82.70%	81.89%
Test Precision(macro)	0.75	0.78	0.76
Test Recall(macro)	0.69	0.72	0.77
Test F1-score(macro)	0.70	0.72	0.76

Πίνακας 3: Αποτελέσματα του μοντέλου MC3-18

Χρόνος εκτέλεσης: 53 min

4.2.5 Αξιολόγηση 4ου Μοντέλου

Παρουσίαση Μοντέλου

Το μοντέλο R3D-18 (ResNet-3D 18 layers) είναι και αυτό μια παραλλαγή των ResNet που σχεδιάστηκε ειδικά για την επεξεργασία βίντεο μέσω της χρήσης 3D συνελικτικών φίλτρων. Το "R3D" σημαίνει "Residual 3D", και το "18" αναφέρεται στο βάθος του δικτύου. Το R3D-18 ενσωματώνει τις αρχές της αρχιτεκτονικής ResNet με επεκτάσεις για τρισδιάστατες συνελικτικές διαδικασίες, οι οποίες είναι ιδανικές για τη διαχείριση των χωρικών και χρονικών διαστάσεων σε δεδομένα βίντεο.

Fine-tuning

Αρχικά, γίνεται φόρτωση του προεκπαιδευμένου μοντέλου R3D-18. Όπως και στα προηγούμενα μοντέλα, τα πρώτα στρώματα του μοντέλου "παγώνουν" και το τελικό επίπεδο ταξινόμησης τροποποιείται για την ταξινόμηση σε 6 κατηγορίες.

```
# Load pretrained model
model = video.r3d_18(pretrained=True)

# Freeze the parameters of the initial layers
for param in model.parameters():
    param.requires_grad = False

# Modify the final classification layer
model.fc = nn.Linear(model.fc.in_features, out_features=6)

# Move the model to the GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Print the model's architecture to review changes
print(model)
```

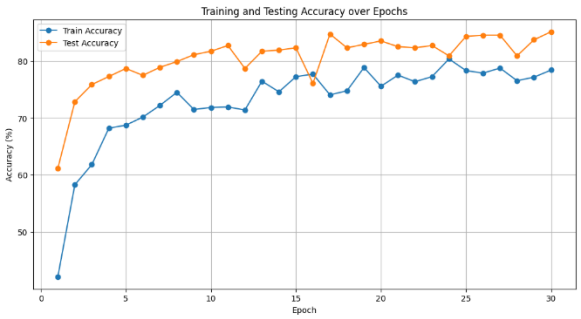
Κώδικας 4.2.5.α: Φόρτωση προεκπαιδευμένου μοντέλου R3D-18

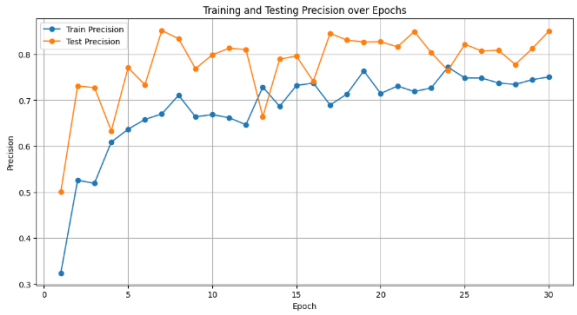
Ορίζονται επίσης οι παρακάτω μετασχηματισμοί [42]:

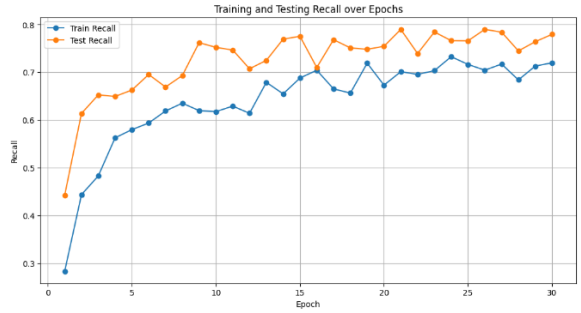
```
# Create transformations for the video frames
transform = transforms.Compose([
    transforms.Resize(size=[128, 171], interpolation=InterpolationMode.BILINEAR),  # Resize the frames
    transforms.CenterCrop(size=[112, 112]),  # Central crop
    transforms.ToTensor(),  # Convert frames to tensor and scale to [0, 1]
    transforms.Normalize(mean=[0.43216, 0.394666, 0.37645], std=[0.22803, 0.22145, 0.216989]),  # Normalize
])
```

Κώδικας 4.2.5.β: Ορισμός των μετασχηματισμών για τα καρέ των βίντεο

Αποτελέσματα Μοντέλου







Εικόνα 4.2.5.α: Train and Test accuracy over epochs

Εικόνα 4.2.5.β: Train and Test precision over epochs

Εικόνα 4.2.5.γ: Train and Test recall over epochs

Εικόνα 4.2.5.δ: Train and Test F1-score over epochs

	10 Εποχές	20 Εποχές	30 Εποχές
Train Accuracy	71.83%	75.55%	78.39%
Train Precision (macro)	0.67	0.71	0.75
Train Recall(macro)	0.62	0.67	0.72
Train F1-score(macro)	0.63	0.69	0.73
Test Accuracy	81.69%	83.50%	85.11%
Test Precision(macro)	0.80	0.83	0.85
Test Recall(macro)	0.75	0.75	0.78
Test F1-score(macro)	0.76	0.77	0.79

Πίνακας 4: Αποτελέσματα του μοντέλου R3D-18

Χρόνος εκτέλεσης: 52 min

4.2.6 Αξιολόγηση 5ου Μοντέλου

Το Swin3D [43] είναι μια επέκταση του Swin Transformer ειδικά σχεδιασμένη για τρισδιάστατα δεδομένα, όπως βίντεο και τρισδιάστατα αντικείμενα. Χρησιμοποιεί μια ιεραρχική αρχιτεκτονική με βάση τα τρισδιάστατα παράθυρα (3D windows) που επιτρέπουν την αποτελεσματική ανάλυση τοπικών περιοχών του 3D χώρου.

Βασικά Χαρακτηριστικά του Swin3D

3D Ιεραρχική Δομή:

Το Swin3D οργανώνει τα δεδομένα σε τρισδιάστατα παράθυρα, τα οποία καλύπτουν όγκους δεδομένων. Η ιεραρχική δομή επιτρέπει την επεξεργασία του 3D χώρου σε διαφορετικές κλίμακες, από τοπικές περιοχές με λεπτομερή χαρακτηριστικά έως μεγαλύτερες περιοχές με πιο συνολική πληροφορία, βελτιώνοντας έτσι την κατανόηση της δομής σε πολλαπλά επίπεδα.

Self-Attention σε Τρεις Διαστάσεις:

Αντί να εφαρμόζεται το self-attention σε δύο διαστάσεις (όπως στις εικόνες), το Swin3D επεξεργάζεται τρεις διαστάσεις, καλύπτοντας τόσο το χώρο όσο και το χρόνο. Το self-attention υπολογίζεται τοπικά μέσα σε κάθε 3D παράθυρο, καθιστώντας το κατάλληλο για δεδομένα με χωροχρονικές σχέσεις, όπως σειρές από frames βίντεο ή τρισδιάστατες αναπαραστάσεις αντικειμένων.

Μετατοπισμένα Παράθυρα σε 3D (Shifted 3D Windows):

Για να εξασφαλίσει την κάλυψη πληροφοριών σε ολόκληρο τον τρισδιάστατο χώρο, το Swin3D μετατοπίζει τα παράθυρα του σε κάθε επίπεδο του δικτύου. Έτσι, επιτυγχάνει μεγαλύτερη πληροφόρηση για τις σχέσεις μεταξύ γειτονικών περιοχών του 3D χώρου χωρίς να αυξάνει σημαντικά το υπολογιστικό κόστος.

Λειτουργία του Swin3D

Το Swin3D εφαρμόζει την ίδια τεχνική των 3D shifted windows, αλλά επεκτείνεται στη χρονική και χωρική διάσταση, επιτρέποντας την κατανόηση των σχέσεων τόσο στο χώρο όσο και στο χρόνο. Αυτή η προσέγγιση είναι ιδιαίτερα αποδοτική για την επεξεργασία

δεδομένων που απαιτούν 3D ανάλυση, όπως αναγνώριση αντικειμένων σε βίντεο ή 3D κατανόηση περιβάλλοντος.

Fine-tuning

Αρχικά γίνεται η φόρτωση του προεκπαιδευμένου μοντέλου swined-b. Τα αρχικά στρώματα του μοντέλου "παγώνουν" και τροποποιείται το τελικό επίπεδο ταξινόμησης.

```python
# Load pretrained model
model = video.swin3d_b(pretrained=True)

# Freeze the parameters of the initial layers
for param in model.parameters():
    param.requires_grad = False

# Modify the final classification layer
model.head = nn.Linear(model.head.in_features, out_features=6)

# Move the model to the GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Print the model's architecture to review changes
print(model)
```

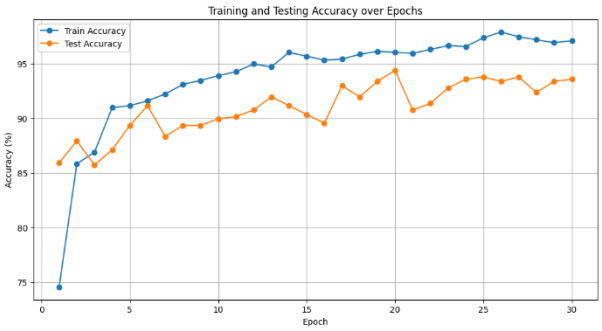
Κώδικας 4.2.6.α: Φόρτωση προεκπαιδευμένου μοντέλου Swin3d

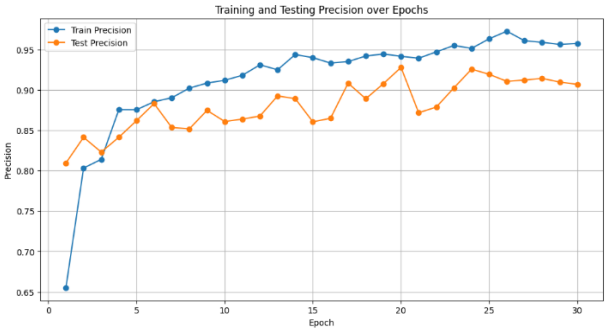
Ορίζονται οι παρακάτω μετασχηματισμοί [44]:

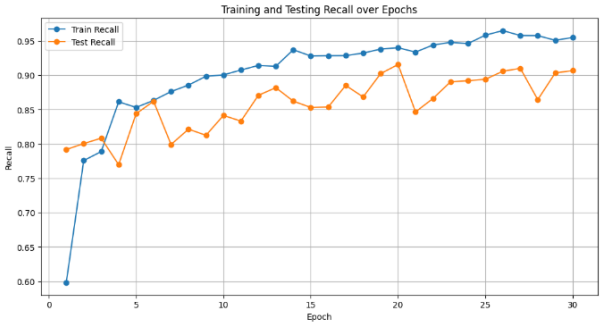
```python
# Create transformations for the video frames
transform = transforms.Compose([
    transforms.Resize(size=[256, 256], interpolation=InterpolationMode.BILINEAR),  # Resize the frames
    transforms.CenterCrop(size=[224, 224]),  # Central crop
    transforms.ToTensor(),  # Convert frames to tensor and scale to [0, 1]
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),  # Normalize
])
```

Κώδικας 4.2.6.β: Ορισμός των μετασχηματισμών για τα καρέ των βίντεο για το μοντέλο Swin3d

Αποτελέσματα Μοντέλου







Εικόνα 4.2.6.α: Train and Test accuracy over epochs for Swin3d

Εικόνα 4.2.6.β: Train and Test precision over epochs for Swin3d

Εικόνα 4.2.6.γ: Train and Test recall over epochs for Swin3d

Εικόνα 4.2.6.δ: Train and Test F1-score over epochs for Swin3d

	10 Εποχές	20 Εποχές	30 Εποχές
Train Accuracy	93.89%	96.01%	97.08%
Train Precision (macro)	0.91	0.94	0.96
Train Recall(macro)	0.90	0.94	0.96
Train F1-score(macro)	0.91	0.94	0.96
Test Accuracy	89.94%	90.74%	93.56%
Test Precision(macro)	0.86	0.87	0.91
Test Recall(macro)	0.84	0.85	0.91
Test F1-score(macro)	0.84	0.85	0.91

Πίνακας 5: Αποτελέσματα του μοντέλου Swin3d-b

Χρόνος εκτέλεσης: 267 min

4.2.7 Σύγκριση και Ανάλυση Αποτελεσμάτων

Τα συστήματα εκτελούνται στο Google Colab, χρησιμοποιώντας μια NVIDIA T4 GPU (16GB VRAM) και μια CPU πολλαπλών πυρήνων με ταχύτητα 2.2-2.3 GHz. Το σύστημα διαθέτει 25GB RAM, προσφέροντας επαρκή υπολογιστική ισχύ για παράλληλες διεργασίες και εφαρμογές βαθιάς μάθησης.

Όλα τα συστήματα εκπαιδεύτηκαν και αξιολογήθηκαν χρησιμοποιώντας τα ίδια σύνολα εκπαίδευσης (train set) και δοκιμής (test set). Πιο συγκεκριμένα, η εκπαίδευση πραγματοποιήθηκε αποκλειστικά σε βίντεο που περιέχουν τουλάχιστον 32 frames. Συνεπώς, κάθε κατηγορία περιλάμβανε τον εξής αριθμό βίντεο:

- Fall Down: 139, 57 βίντεο για train και test αντίστοιχα
- Lying Down: 178, 82 βίντεο για train και test αντίστοιχα
- Sit Down: 95, 43 βίντεο για train και test αντίστοιχα
- Sitting: 218, 96 βίντεο για train και test αντίστοιχα
- Stand up: 97, 45 βίντεο για train και test αντίστοιχα
- Walking: 402, 174 βίντεο για train και test αντίστοιχα

Παρακάτω παρουσιάζεται ένας αναλυτικός πίνακας με τα αποτελέσματα όλων των μοντέλων.

ΣΥΝΟΛΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΟΝΤΕΛΩΝ					
	R(2+1)D-18	**SlowFast-R50**	**MC3-18**	**R3D-18**	**Swin3d**
10 Εποχές					
Train Accuracy	69.00%	88.93%	69.26%	71.83%	93.89%
Train Precision (macro)	0.63	0.85	0.63	0.67	0.91
Train Recall (macro)	0.58	0.84	0.58	0.62	0.90
TrainF1-score (macro)	0.59	0.84	0.59	0.63	0.91
Test Accuracy	78.47%	88.33%	80.08%	81.69%	89.94%
Test Precision (macro)	0.63	0.84	0.75	0.80	0.86
Test Recall (macro)	0.58	0.82	0.69	0.75	0.84
Test F1-score (macro)	0.59	0.82	0.70	0.76	0.84
20 Εποχές					
Train Accuracy	74.49%	92.56%	73.25%	75.55%	96.01%
Train Precision (macro)	0.68	0.89	0.67	0.71	0.94
Train Recall (macro)	0.65	0.89	0.64	0.67	0.94
TrainF1-score (macro)	0.66	0.89	0.65	0.69	0.94
Test Accuracy	82.70%	88.33%	83.70%	83.50%	90.74%
Test Precision (macro)	0.77	0.87	0.78	0.83	0.87
Test Recall (macro)	0.75	0.81	0.72	0.75	0.85
Test F1-score (macro)	0.76	0.81	0.72	0.77	0.85
30 Εποχές					
Train Accuracy	74.49%	93.00%	76.71%	78.39%	97.08%
Train Precision (macro)	0.69	0.91	0.72	0.75	0.96
Train Recall (macro)	0.66	0.90	0.69	0.72	0.96
TrainF1-score (macro)	0.67	0.90	0.70	0.73	0.96
Test Accuracy	82.49%	89.94%	81.89%	85.11%	93.56%
Test Precision (macro)	0.81	0.86	0.76	0.85	0.91
Test Recall (macro)	0.73	0.86	0.77	0.78	0.91
TrainF1-score (macro)	0.75	0.86	0.76	0.79	0.91
Χρόνοι Εκτέλεσης					

Χρόνος εκτέλεσης σε λεπτά (min)	61 min	77 min	53 min	52 min	267 min

Πίνακας 6: Συνολικά αποτελέσματα μοντέλων

Επιπλέον, τα μοντέλα αξιολογήθηκαν σε εντελώς νέα, άγνωστα δεδομένα, τα οποία περιλάμβαναν 60 βίντεο συνολικά, κατανεμημένα σε 10 βίντεο για κάθε μία από τις 6 κατηγορίες. Τα βίντεο λήφθηκαν σε δύο διαφορετικά εσωτερικά περιβάλλοντα οικιακών χώρων, ώστε να διασφαλιστεί η ποικιλία στις συνθήκες φωτισμού και διάταξης του χώρου. Κάθε βίντεο έχει διάρκεια 2-3 δευτερολέπτων, με ανάλυση 1280x720 pixels και ρυθμό ανανέωσης 30 frames ανά δευτερόλεπτο, επιτρέποντας την καταγραφή λεπτομερών κινήσεων.

Η αξιολόγηση των συστημάτων πραγματοποιήθηκε με βάση το accuracy, precision, recall και f1-score προκειμένου να προσδιοριστεί η συνολική απόδοση τους. Αξίζει να σημειωθεί ότι τα δεδομένα των βίντεο ήταν πρωτόγνωρα για τα μοντέλα, προκειμένου να αξιολογηθεί η γενική ικανότητά τους να αναγνωρίζουν και να κατηγοριοποιούν νέα σενάρια. Η απόδοση των μοντέλων συνοψίζεται στον παρακάτω πίνακα:

ΑΞΙΟΛΟΓΗΣΗ ΜΟΝΤΕΛΩΝ ΣΕ ΑΓΝΩΣΤΑ ΔΕΔΟΜΕΝΑ					
	R(2+1)D-18	*SlowFast-R50*	*MC3-18*	*R3D-18*	*Swin3d*
Accuracy	46.67%	60.00%	41.67%	33.33%	41.67%
Precision (macro)	0.41	0.73	0.38	0.37	0.48
Recall (macro)	0.47	0.60	0.42	0.33	0.42
F1-score (macro)	0.41	0.60	0.36	0.31	0.37

Πίνακας 7: Αξιολόγηση μοντέλων σε άγνωστα δεδομένα

Παρατηρούμε ότι το μοντέλο SlowFast-R50 παρουσιάζει την καλύτερη απόδοση στα άγνωστα δεδομένα. Συνεπώς, στο επόμενο κεφάλαιο θα επικεντρωθούμε στην επανεκπαίδευσή του.

ΚΕΦΑΛΑΙΟ 5

5. ΕΠΑΝΕΚΠΑΙΔΕΥΣΗ

5.1 Προεπεξεργασία Δεδομένων

Για την επανεκπαίδευση του μοντέλου SlowFast-R50, είναι απαραίτητη η περαιτέρω προεπεξεργασία του συνόλου δεδομένων. Αρχικά, θα υπολογιστούν οι μέσες τιμές (mean) και οι τυπικές αποκλίσεις (std) για το συγκεκριμένο σύνολο δεδομένων που χρησιμοποιείται στην εκπαίδευση του μοντέλου. Ο κώδικας που ακολουθεί παρουσιάζει τη διαδικασία αυτών των υπολογισμών:

```python
def compute_mean_and_std(video_dir):
    # Lists to store the mean and std values for each video
    means = []
    stds = []
    # Get all video files in the directory with extensions .mp4 and .avi
    video_files = [os.path.join(video_dir, f) for f in os.listdir(video_dir) if f.endswith(('.mp4', '.avi'))]
    # Variables to accumulate the total sum of means and squared means for all frames
    total_frames = 0
    sum_means = np.zeros(3)  # To store the cumulative mean for R, G, B channels
    sum_squared_means = np.zeros(3)  # To store the cumulative squared mean for std calculation

    # Loop through all the video files
    for video_file in tqdm(video_files):
        # Open the video file
        cap = cv2.VideoCapture(video_file)

        # Read frames until the video ends
        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break
            # Convert frame to float32 for precise calculations and normalize to [0, 1]
            frame = frame.astype(np.float32) / 255.0
            # Calculate mean and std for each channel (R, G, B) in the current frame
            mean_per_frame = np.mean(frame, axis=(0, 1))  # Mean for each channel
            std_per_frame = np.std(frame, axis=(0, 1))    # Std for each channel
            # Accumulate the sums of means and squared stds for all frames
            sum_means += mean_per_frame
            sum_squared_means += std_per_frame ** 2
            total_frames += 1

        # Release the video capture object after reading all frames
        cap.release()

    # Calculate final mean and std for all frames in all videos
    mean = sum_means / total_frames
    std = np.sqrt(sum_squared_means / total_frames)

    return mean, std

# Define the video directory
video_directory = '/content/train'
# Compute the mean and standard deviation for all videos in the directory
mean, std = compute_mean_and_std(video_directory)
```

Κώδικας 5.1.α: Υπολογισμός mean και std του συνόλου δεδομένων

```python
# Create transformations for the video frames
slowfast_alpha = 4
transform = transforms.Compose([
    transforms.Resize(size=[256, 256]),  # Resize the frames
    transforms.CenterCrop(size=256),  # Central crop
    transforms.ToTensor(),  # Convert frames to tensor and scale to [0, 1]
    transforms.Normalize(mean=[0.36581896, 0.43348778, 0.46818488], std=[0.22157381, 0.21832773, 0.20730222]),  # Normalize
])
```

Κώδικας 5.1.β: Ορισμός μετασχηματισμών για τα καρέ των βίντεο

Επιπλέον, ορίζεται η κλάση CustomVideoDatasetWithDetectron, η οποία διαχειρίζεται ένα dataset και αξιοποιεί τη βιβλιοθήκη Detectron2, ένα σύστημα για την ανίχνευση και κατηγοριοποίηση αντικειμένων (object segmentation) [45]. Η κλάση αυτή εφαρμόζει ανίχνευση ανθρώπων σε κάθε βίντεο, φιλτράροντας τα καρέ στα οποία εντοπίζεται ανθρώπινη παρουσία. Ακολουθεί η εφαρμογή προκαθορισμένων μετασχηματισμών (transformations) στις εικόνες. Παρακάτω περιγράφεται αναλυτικά η λειτουργία των βασικών συναρτήσεων της κλάσης:

1. __init__ (Constructor)

- csv_file (str): Μονοπάτι του αρχείου CSV που περιέχει τα ονόματα των βίντεο και τις αντίστοιχες ετικέτες.
- root_dir (str): Ο κατάλογος που περιέχει όλα τα βίντεο.
- preprocessed_dir (str): Ο κατάλογος με τα προεπεξεργασμένα βίντεο.
- preprocessed(Boolean): Αν τα δεδομένα είναι προεπεξεργασμένα
- cfg_path (str): Το μονοπάτι προς το αρχείο διαμόρφωσης της Detectron2 (για το προ-εκπαιδευμένο μοντέλο).
- device (str): Η συσκευή στην οποία τρέχει το μοντέλο ('cuda' ή 'cpu').

Κατά την αρχικοποίηση, το πρόγραμμα φορτώνει τις ετικέτες (annotations) από το αρχείο CSV, φορτώνει το μοντέλο της Detectron2 για την ανίχνευση ατόμων στα καρέ των βίντεο και ορίζεται ο predictor για την αναγνώριση ατόμων στα καρέ των βίντεο.

2. __len__

Αυτή η συνάρτηση επιστρέφει το μήκος του dataset, δηλαδή πόσα στοιχεία περιέχει το dataset, βασισμένη στο μέγεθος του CSV αρχείου.

3. __getitem__

Αυτή η συνάρτηση χρησιμοποιείται για την ανάκτηση των στοιχείων του dataset. Για ένα συγκεκριμένο idx (δείκτη), εκτελεί τις εξής λειτουργίες:

- video_file, label: Ανακτά το όνομα του βίντεο και την ετικέτα από το CSV.
- video_path: Δημιουργεί το πλήρες μονοπάτι του βίντεο.

Η συνάρτηση στη συνέχεια ελέγχει αν υπάρχει προεπεξεργασμένη έκδοση του βίντεο. Εάν ναι, φορτώνει το προεπεξεργασμένο βίντεο. Διαφορετικά η συνάρτηση load_video φορτώνει το βίντεο από το δίσκο έπειτα η detect εφαρμόζει ανίχνευση ατόμων για κάθε καρέ και η sample_frames επιλέγει 32 καρέ από εκείνα στα οποία έχουν εντοπιστεί άτομα. Τέλος εφαρμόζονται τυχόν μετασχηματισμούς στα καρέ και τα επιστρέφει με την αντίστοιχη ετικέτα.

4. load_video

Αυτή η συνάρτηση φορτώνει τα καρέ ενός βίντεο από το αρχείο χρησιμοποιώντας το OpenCV. Διαβάζει κάθε καρέ μέχρι να τελειώσει το βίντεο και τα αποθηκεύει σε έναν πίνακα.

5. sample_frames

Αυτή η συνάρτηση δειγματοληπτεί έναν συγκεκριμένο αριθμό καρέ από το βίντεο, επιλέγοντας τα καρέ ομοιόμορφα σε όλη τη διάρκεια του βίντεο. Χρησιμοποιεί τη συνάρτηση np.linspace για την κατανομή των δειγμάτων.

6. detect

Αυτή η συνάρτηση εκτελεί ανίχνευση ατόμων σε ένα καρέ χρησιμοποιώντας το προ-εκπαιδευμένο μοντέλο της Detectron2. Αναγνωρίζει την παρουσία ατόμων και επιστρέφει μόνο τις εικόνες που εντοπίστηκε άτομο. Εάν δεν εντοπιστεί άτομο, επιστρέφει None.

- predictor(frame): Εκτελεί ανίχνευση στο καρέ.
- pred_classes: Ανακτά τις κλάσεις των ανιχνευμένων αντικειμένων (η κλάση 0 αντιπροσωπεύει άτομα).
- person_boxes: Λαμβάνει το κουτί περιγράμματος (bounding box) του πρώτου ατόμου που ανιχνεύτηκε.

Ακολουθεί η αναλυτική παρουσίαση του κώδικα της κλάσης:

```python
class CustomVideoDatasetWithDetectron(Dataset):
    def __init__(self, csv_file, root_dir, transform = None ,preprocessed_dir=None, preprocessed = False, cfg_path="COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml", device='cuda'):
        """
        Initialize the dataset with CSV file, root directory, and optional transformations.

        Args:
            csv_file (str): Path to the CSV file containing video file names and labels.
            root_dir (str): Directory with all the videos.
            transform (callable): A function/transform to apply to the videos.
            preprocessed_dir (str): Directory for preprocessed videos.
            preprocessed (bool): Whether to use preprocessed videos.
            cfg_path (str): The detectron2 config path for the pre-trained model.
            device (str): Device to run the model on ('cuda' or 'cpu').
        """

        # Load annotations from the provided CSV file
        self.annotations = pd.read_csv(csv_file)
        self.root_dir = root_dir
        self.transform = transform
        self.preprocessed_dir = preprocessed_dir
        self.preprocessed = preprocessed

        # Load Detectron2 model for person detection
        self.cfg = get_cfg()
        self.cfg.merge_from_file(model_zoo.get_config_file(cfg_path))
        self.cfg.MODEL.DEVICE = device  # Use the specified device (GPU or CPU)
        self.cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(cfg_path)
        self.predictor = DefaultPredictor(self.cfg)


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

Κώδικας 5.1.γ: Ορισμός της κλάσης CustomVideoDatasetWithDetectron και των μεθόδων της

```
def __getitem__(self, idx):

    # Get the video filename and corresponding label from the CSV
    video_file = self.annotations.iloc[idx, 0]
    label = self.annotations.iloc[idx, 1]
    video_path = os.path.join(self.root_dir, video_file)
    # Define the preprocessed video path (irrespective of preprocessed flag)
    preprocessed_video_path = os.path.join(self.preprocessed_dir, f'{os.path.splitext(video_file)[0]}.pt')

    # Check for a preprocessed video version and load if available
    if self.preprocessed:
        if os.path.exists(preprocessed_video_path):
            video_tensor = torch.load(preprocessed_video_path)
        else:
            return None
    else:
        video = self.load_video(video_path)
        if len(video) == 0:
            return None
        # Detect and crop the human in each frame
        frames_with_person = [self.detect(frame) for frame in video]
        frames_with_person = [frame for frame in frames_with_person if frame is not None]
        # If there are not enough frames with people, return None
        if len(frames_with_person) < 32:
            return None
        # Sample 32 frames from the frames with people
        sampled_frames = self.sample_frames(frames_with_person, num_frames=32)
        # Apply transformations if provided
        if self.transform:
            transformed_frames = [self.transform(frame) for frame in sampled_frames]
            video_tensor = torch.stack(transformed_frames, dim=0)
        else:
            video_tensor = torch.stack([transforms.ToTensor()(frame) for frame in sampled_frames], dim=0)

        # Save the preprocessed video tensor for future use
        os.makedirs(self.preprocessed_dir, exist_ok=True)
        torch.save(video_tensor, preprocessed_video_path)

    return video_tensor, label
```

Κώδικας 5.1.δ: : Συνέχεια των μεθόδων της κλάσης CustomVideoDatasetWithDetectron

```
def load_video(self, path):

    """
    Load a video from the given file path and return its frames.

    Args:
        path (str): Path to the video file.

    Returns:
        np.array: Array of video frames.
    """

    video_frames = []
    cap = cv2.VideoCapture(path)
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
        video_frames.append(frame)
    cap.release()

    return np.array(video_frames)

def sample_frames(self, video, num_frames):

    """
    Sample a specific number of frames evenly from the video.

    Args:
        video (np.array): Array of video frames.
        num_frames (int): The number of frames to sample.

    Returns:
        list: List of sampled frames.
    """

    total_frames = len(video)
    indices = np.linspace(0, total_frames - 1, num_frames, dtype=int).tolist()

    return [video[i] for i in indices]
```

Κώδικας 5.1.ε: Συνέχεια των μεθόδων της κλάσης CustomVideoDatasetWithDetectron

```
def detect(self, frame):

    """
    Detect the person in the frame using Detectron2 and crop the frame around the detected person.
    Args:
        frame (np.array): A single video frame (image).
    Returns:
        np.array: the frame with person or None if no person is detected.
    """

    # Run person detection on the frame
    outputs = self.predictor(frame)
    instances = outputs["instances"]
    pred_classes = instances.pred_classes.cpu().numpy()
    pred_boxes = instances.pred_boxes if instances.has("pred_boxes") else None

    # Check if any person is detected (class id for 'person' is 0)
    person_boxes = pred_boxes[pred_classes == 0] if pred_boxes is not None else []

    if len(person_boxes) > 0:
        """
          # Get the first detected person (it can be modified to handle multiple people if needed)
          #box = person_boxes[0].tensor.cpu().numpy()[0]

          #x1, y1, x2, y2 = map(int, box)
          #cropped_frame = frame[y1:y2, x1:x2]  # Crop the frame to the bounding box
        """
        return transforms.ToPILImage()(frame)  # Convert frame to PIL image

    return None
```

Κώδικας 5.1.ζ: Συνέχεια των μεθόδων της κλάσης CustomVideoDatasetWithDetectron

5.2 Εκπαίδευση Μοντέλου

Αρχικά, φορτώνουμε το προεκπαιδευμένο μοντέλο SlowFast-R50 και προσαρμόζουμε το τελευταίο επίπεδο ώστε να μπορεί να πραγματοποιήσει ταξινόμηση σε 6 κατηγορίες.

```
# Load the pretrained SlowFast R50 model
model = slowfast_r50(pretrained=True)

# Modify the final classification layer
model.output = nn.Linear(model.blocks[6].proj.out_features, 6)

# Move the model to the GPU if available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)
```

Κώδικας 5.2.α: Φόρτωση του μοντέλου SlowFast-R50

Για την εκπαίδευση του μοντέλου, αρχικά καθορίζεται η συνάρτηση απώλειας ως nn.CrossEntropyLoss, η οποία είναι κατάλληλη για ταξινόμηση, και ο optimizer ως torch.optim.Adagrad, με ρυθμό εκμάθησης (learning rate) 0.001 και παράμετρο κανονικοποίησης (weight decay) 0.0001. Στη συνέχεια, εκτελούνται 6 επαναλήψεις, όπου σε κάθε επανάληψη το πρόγραμμα παγώνει τα πρώτα 5-i blocks του μοντέλου (καθιστώντας τα μη εκπαιδεύσιμα) και επιτρέπει την εκπαίδευση μόνο στα υπόλοιπα i+1 blocks. Κατά τη διάρκεια κάθε επανάληψης, η συνάρτηση train_model εκπαιδεύει το

μοντέλο για 5 εποχές χρησιμοποιώντας τα δεδομένα εκπαίδευσης και δοκιμής, και υπολογίζει διάφορους δείκτες απόδοσης, όπως η ακρίβεια, το precision, το recall και το F1-score.

```
# Define the loss function and optimizer
criterion = nn.CrossEntropyLoss()  # Loss function for classification
optimizer = torch.optim.Adagrad(model.parameters(), lr=0.001, weight_decay=0.0001)
num_epochs = 5  # Number of epochs for training
start_time = time.time()
for i in range(6):
  print(f"Step {i+1}: Freezing the first {5-i} blocks and training the last {i+1} blocks.")
  for param in model.blocks[:5-i].parameters():
    param.requires_grad= False
  for param in model.blocks[5-i:].parameters():
    param.requires_grad = True
  train_accuracies, train_precisions, train_recalls, train_f1_scores, \
  test_accuracies, test_precisions, test_recalls, test_f1_scores = \
  train_model(model, dataloader, test_dataloader, criterion, optimizer, num_epochs, slowfast_alpha=slowfast_alpha)
finish_time = time.time()
print(f'Training time: {(finish_time - start_time)/60:.2f} minutes')  # Print total training time
```

Κώδικας 5.2.β: Επανεκπαίδευση του μοντέλου SlowFast-R50

5.3 Τελικά Αποτελέσματα

Το παραπάνω σύστημα εκτελείται στο Google Colab, χρησιμοποιώντας μια NVIDIA L4 GPU (24GB VRAM) και μια CPU πολλαπλών πυρήνων με ταχύτητα 2.6-2.8 GHz. Το σύστημα διαθέτει 45GB RAM, επιτρέποντας την ομαλή εκτέλεση μοντέλων μηχανικής μάθησης μεγάλης κλίμακας.

Το σύστημα εκπαιδεύτηκε και αξιολογήθηκε χρησιμοποιώντας τα ίδια σύνολα εκπαίδευσης (train set) και δοκιμής (test set). Πιο συγκεκριμένα, η εκπαίδευση πραγματοποιήθηκε αποκλειστικά σε βίντεο που περιέχουν τουλάχιστον 32 frames με ανθρώπινη παρουσία. Συνεπώς, κάθε κατηγορία περιλάμβανε τον εξής αριθμό βίντεο:

- Fall Down: 101, 39 βίντεο για train και test αντίστοιχα
- Lying Down: 170, 77 βίντεο για train και test αντίστοιχα
- Sit Down: 93, 43 βίντεο για train και test αντίστοιχα
- Sitting: 214, 96 βίντεο για train και test αντίστοιχα
- Stand up: 97, 45 βίντεο για train και test αντίστοιχα
- Walking: 396, 171 βίντεο για train και test αντίστοιχα

Παρακάτω παρουσιάζονται τα αποτελέσματα του μοντέλου:

ΑΠΟΤΕΛΕΣΜΑΤΑ ΤΗΣ ΕΠΑΝΕΚΠΑΙΔΕΥΣΗΣ ΤΟΥ ΜΟΝΤΕΛΟΥ SlowFast-R50						
	1ηΕπανάληψη 5η εποχή	2ηΕπανάληψη 5η εποχή	3ηΕπανάληψη 5η εποχή	4ηΕπανάληψη 5η εποχή	5ηΕπανάληψη 5η εποχή	6ηΕπανάληψη 5η εποχή
Train Accuracy	40.54%	100%	99.81%	100%	100%	100%
Train Precision(macro)	0.19	1.00	1.00	1.00	1.00	1.00
Train Recall (macro)	0.20	1.00	1.00	1.00	1.00	1.00
Train F1-Score(macro)	0.19	1.00	1.00	1.00	1.00	1.00
Test Accuracy	38.38%	98.08%	98.93%	98.93%	98.93%	98.72%

Test Precision(macro)	0.13	0.97	0.99	0.99	0.99	0.99
Test Recall (macro)	0.14	0.98	0.98	0.98	0.98	0.98
Test F1-Score(macro)	0.13	0.97	0.98	0.98	0.98	0.98

Πίνακας 8: Αποτελέσματα της επανεκπαίδευσης του SlowFast-R50

Χρόνος εκτέλεσης: 78 min (ο χρόνος εκπαίδευσης δεν περιλαμβάνει τον χρόνο προεπεξεργασίας των βίντεο)

Επιπλέον, όπως και στο προηγούμενο κεφάλαιο, το μοντέλο αξιολογήθηκε χρησιμοποιώντας άγνωστα δεδομένα, τα οποία δεν είχαν χρησιμοποιηθεί κατά τη φάση εκπαίδευσης. Αυτή η διαδικασία αξιολόγησης σε νέα δεδομένα είναι σημαντική για τη μέτρηση της ικανότητας γενίκευσης του μοντέλου, δηλαδή της δυνατότητάς του να εφαρμόζει τις γνώσεις που έχει αποκτήσει σε πραγματικά σενάρια, όπου τα δεδομένα ενδέχεται να διαφέρουν από αυτά της εκπαίδευσης. Στο παρακάτω πίνακα παρατίθενται τα αποτελέσματα του μοντέλου:

ΑΞΙΟΛΟΓΗΣΗ ΤΟΥ ΕΠΑΝΕΚΠΑΙΔΕΥΜΕΝΟΥ ΜΟΝΤΕΛΟΥ ΣΕ ΑΓΝΩΣΤΑ ΔΕΔΟΜΕΝΑ	
Accuracy	80%
Precision(macro)	0.82
Recall (macro)	0.80
F1-Score(macro)	0.80

Πίνακας 9: Αξιολόγηση του μοντέλου σε άγνωστα δεδομένα

Παρατηρούνται βελτιωμένα αποτελέσματα σε σύγκριση με προηγούμενες μετρήσεις. Ωστόσο, εξακολουθεί να υπάρχει σημαντική απόκλιση στην απόδοση μεταξύ των δεδομένων εκπαίδευσης και δοκιμής, σε σχέση με τα άγνωστα δεδομένα. Αυτό μπορεί να οφείλεται σε διαφορετικές συνθήκες φωτισμού ή στο γεγονός ότι το μοντέλο έχει προσαρμοστεί και έχει μάθει καλά τα περιβάλλοντα στα οποία εκπαιδεύτηκε, χωρίς να έχει αναπτύξει την ικανότητα να γενικεύει αποτελεσματικά.

ΚΕΦΑΛΑΙΟ 6

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Η αναγνώριση ανθρώπινης δραστηριότητας μέσω οπτικών δεδομένων, ιδιαίτερα σε οικιακούς μη ελεγχόμενους χώρους, είναι ένα πεδίο που βρίσκεται σε αρχικά στάδια ανάπτυξης. Οι προκλήσεις είναι αρκετές, όπως ότι οι οικιακοί χώροι είναι δυναμικοί και μεταβλητοί, με ποικιλία σκηνικών και συμπεριφορών που πρέπει να αναγνωρίζονται με ακρίβεια. Η τεχνολογία απαιτεί προηγμένους αλγορίθμους μηχανικής μάθησης και υπολογιστικής όρασης για να αναγνωρίσει και να καταγράψει τις ανθρώπινες δραστηριότητες σε πραγματικό χρόνο, λαμβάνοντας υπόψη τις διαφορετικές συνθήκες φωτισμού και τις αλλαγές στο περιβάλλον.

Στην συγκεκριμένη εργασία έγινε προσπάθεια για την εύρεση ενός προεκπαιδευμένου μοντέλου που μπορεί να προσαρμοστεί μέσω της τεχνικής του fine-tuning. Το fine-tuning επιτρέπει στο αρχικό μοντέλο να προσαρμοστεί στις ειδικές ανάγκες των οικιακών χώρων, ενσωματώνοντας τις ιδιαίτερες συνθήκες και τα χαρακτηριστικά τους. Αυτή η διαδικασία καθιστά το μοντέλο πιο ακριβές και αποτελεσματικό στην αναγνώριση δραστηριοτήτων, γεγονός που είναι σημαντικό για την ανάπτυξη έξυπνων σπιτιών. Μέσω της προσαρμογής του μοντέλου στις συνθήκες κάθε οικιακού χώρου, η τεχνολογία μπορεί να βελτιώσει την αλληλεπίδραση με τους κατοίκους και να προσαρμοστεί στις ατομικές τους ανάγκες και συνήθειες.

Επιπλέον, η συνδυασμένη χρήση του μοντέλου με ένα σύστημα Complex Event Processing (CEP) ανοίγει νέες δυνατότητες στην παρακολούθηση και ανάλυση της ανθρώπινης δραστηριότητας. Το CEP μπορεί να επεξεργάζεται και να αναλύει τα δεδομένα σε πραγματικό χρόνο, επιτρέποντας την άμεση ανταπόκριση σε κρίσιμα γεγονότα και καταστάσεις. Στην περίπτωση της παρακολούθησης ασθενών, για παράδειγμα, η συνδυασμένη εφαρμογή των τεχνολογιών αυτών μπορεί να ενισχύσει τη δυνατότητα ανίχνευσης επείγουσας ανάγκης ή επικίνδυνων καταστάσεων. Με αυτόν τον τρόπο, η τεχνολογία δεν προσφέρει μόνο τη δυνατότητα παρακολούθησης, αλλά και την ικανότητα παρέμβασης και υποστήριξης, βελτιώνοντας την ποιότητα ζωής και την ασφάλεια των χρηστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.mdpi.com/2076-3417/13/24/13009

[2] https://www.mdpi.com/1424-8220/17/4/737

[3] https://onlinelibrary.wiley.com/doi/full/10.1155/2022/8383461

[4] https://ieeexplore.ieee.org/abstract/document/9381213

[5] https://www.sciencedirect.com/science/article/pii/S1110016823007937

[6] https://www.nature.com/articles/s41598-023-49739-1

[7] https://link.springer.com/chapter/10.1007/978-3-319-66808-6_18

[8] https://ourworldindata.org/life-expectancy

[9] https://www.mdpi.com/1424-8220/22/17/6463

[10]https://www.kaggle.com/datasets/ngoduy/dataset-video-for-human-action-recognition

[11]https://openaccess.thecvf.com/content_cvpr_2018/papers/Tran_A_Closer_Look_CVPR_2018_paper.pdf

[12]https://pytorch.org/vision/main/models/generated/torchvision.models.video.r2plus1d_18.html

[13] https://www.sciencedirect.com/science/article/abs/pii/S0957417419302878

[14] https://onlinelibrary.wiley.com/doi/full/10.1155/2022/3454167

[15] https://www.sciencedirect.com/science/article/pii/S1877050916320609

[16] Neural Networks and Learning Machines, Simon Haykin

[17]https://www.databricks.com/glossary/complex-event-processing

[18]https://en.wikipedia.org/wiki/Complex_event_processing

[19] https://en.wikipedia.org/wiki/Convolutional_neural_network

[20] https://en.wikipedia.org/wiki/Recurrent_neural_network

[21] https://en.wikipedia.org/wiki/Long_short-term_memory

[22] https://www.sciencedirect.com/science/article/abs/pii/S1084804520302125

[23] Introduction to Data Mining, Pang-ning Tan, Michael Steinbach, Vipin Kumar

[24] https://link.springer.com/article/10.1007/s11042-020-09004-3#Sec13

[25] https://www.mdpi.com/1424-8220/20/22/6424

[26] https://www.mdpi.com/2079-9292/9/12/1993

[27]https://el.wikipedia.org/wiki/%CE%9C%CE%B7%CF%87%CE%B1%CE%BD%CE%B9%CE%BA%CE%AE_%CE%BC%CE%AC%CE%B8%CE%B7%CF%83%CE%B7

[28] https://en.wikipedia.org/wiki/Neural_network_(machine_learning)

[29] https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[30] https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[31] https://www.geeksforgeeks.org/understanding-of-lstm-networks/

[32] https://medium.com/@denizgunay/knn-algorithm-3604c19cd809

[33]https://ankitnitjsr13.medium.com/math-behind-svm-support-vector-machine-864e58977fdb

[34] https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

[35] https://victorzhou.com/blog/intro-to-rnns/

[36] https://peerj.com/articles/cs-1395/

[37]https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e

[38]https://openaccess.thecvf.com/content_ICCV_2019/papers/Feichtenhofer_SlowFast_Networks_for_Video_Recognition_ICCV_2019_paper.pdf

[39] https://pytorch.org/hub/facebookresearch_pytorchvideo_slowfast/

[40]https://pytorch.org/vision/stable/models/generated/torchvision.models.video.mc3_18.html

[41] https://en.wikipedia.org/wiki/Residual_neural_network

[42]https://pytorch.org/vision/main/models/generated/torchvision.models.video.r3d_18.html

[43] https://arxiv.org/pdf/2103.14030

[44]https://pytorch.org/vision/2.0/models/generated/torchvision.models.video.swin3d_b.html#torchvision.models.video.swin3d_b

[45] https://github.com/facebookresearch/detectron2