ROMÂNIA
MINISTERUL APĂRĂRII NAŢIONALE
ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”

FACULTATEA DE SISTEME INFORMATICE ȘI SECURITATE CIBERNETICĂ

Specializarea: Calculatoare și sisteme informatice pentru apărare și securitate națională

RECUNOAȘTEREA AUTOMATĂ A ACTIVITĂȚILOR ÎN SECVENȚE VIDEO

ABSOLVENT:
Vasilache Cosmin George

CONDUCĂTOR ŞTIINŢIFIC:
Lect. dr. ing. Stelian Spînu

Conține _____ file

Inventariat sub nr. _____

Poziția din indicator: _____

Termen de păstrare: _____

BUCUREŞTI
2025

Abstract

Human action recognition in video sequences is an increasingly important topic, with applications in areas such as security, healthcare, and human-computer interaction. In this project, several machine learning models were analyzed to classify actions from video sequences, using both raw video frames and 3D landmark data describing human posture, extracted with the help of MediaPipe Pose Landmarker. The dataset used was UCF50, a widely used dataset in the research literature, which includes 50 types of actions recorded in real-world environments.

Each video was divided into 15 uniform temporal windows across the sequence of frames, and from each window, a representative frame was selected along with the complete set of 33 body landmarks. Each landmark was represented by spatial coordinates (x, y, z) and confidence scores related to visibility and presence in the frame. The coordinates were normalized to fit within the range [-1, 1], ensuring consistency between samples and facilitating the learning process.

Five different architectures were tested: RNN, 1D CNN, a sequential CNN-RNN model, and two multi-input models (CNN + RNN and CNN + 1D CNN). The simplest model, the RNN, which uses only the posture landmarks, achieved an accuracy of 57.33%, while the 1D CNN, also trained from scratch on the same data, reached a significantly better performance of 69.06% accuracy. The sequential CNN-RNN model, which uses a pre-trained ResNet50 to extract visual features from video frames, improved accuracy to 83.20%. The best performance was achieved by the Multi-Input CNN-RNN model, which combined visual information (frames) with postural information (landmarks), reaching an accuracy of 88.19% and an F1 score of 0.9019.

The results show that integrating multiple types of data—spatial and temporal—as well as using pre-trained components, provides a clear advantage in the task of human action recognition. It is also worth noting the performance of the 1D CNN model, which, despite being trained from scratch, delivered remarkable results.

Rezumat

Recunoașterea acțiunilor umane în secvențe video este o temă tot mai importantă, cu aplicații în domenii precum securitate, sănătate și interacțiuni om-calculator. În acest proiect, au fost analizate mai multe modele de învățare automată pentru a clasifica acțiunile din secvențe video, utilizând atât cadre video brute, cât și date de tip repere 3D care descriu postura unei persoane, extrase cu ajutorul MediaPipe Pose Landmarker. Datasetul utilizat a fost UCF50, un set de date larg răspândit în literatura de specialitate, care include 50 de tipuri de acțiuni filmate în medii reale.

Fiecare videoclip a fost împărțit în 15 ferestre temporale uniforme pe parcursul succesiunii de cadre, iar din fiecare fereastră a fost selectat câte un cadru reprezentativ, împreună cu setul complet de 33 de repere (landmarks) asociate corpului uman, fiecare reprezentat prin coordonate spațiale (x, y, z) și scoruri de încredere legate de vizibilitate și prezență în cadru. Coordonatele au fost normalizate pentru a se încadra în intervalul [-1, 1], asigurând consistență între eșantioane și facilitând procesul de învățare.

Au fost testate cinci arhitecturi diferite: RNN, 1D CNN, un model secvențial CNN-RNN, precum și două modele de tip multi-input (CNN + RNN și CNN + 1D CNN). Cel mai simplu model, RNN, care utilizează doar reperele ce descriu postura unei persoane, a obținut o acuratețe de 57,33%, în timp ce 1D CNN, antrenat tot de la zero pe aceleași date, a atins o performanță semnificativ mai bună, cu 69,06% acuratețe. Modelul CNN-RNN secvențial, care folosește un ResNet50 preantrenat pentru extragerea caracteristicilor vizuale din cadrele video, a crescut acuratețea la 83,20%. Cea mai bună performanță a fost obținută de modelul Multi-Input CNN-RNN, care a combinat informația vizuală (cadre) și cea posturală (landmarks), atingând o acuratețe de 88,19% și un scor F1 de 0,9019.

Rezultatele obținute arată că integrarea mai multor tipuri de date, spațiale și temporale, precum și utilizarea de componente preantrenate, oferă un avantaj clar în sarcina de recunoaștere a acțiunilor umane. Totodată, este de remarcat performanța modelului 1D CNN, care, deși a fost antrenat de la zero, a oferit rezultate notabile.

Cuprins

1. Introducere

Recunoașterea activităților umane (HAR-Human Activity Recognition) se referă la procesul de utilizare a algoritmilor de învățare automată și a datelor provenite de la diferiți senzori pentru a detecta și a clasifica activitățile umane, cum ar fi mersul, alergatul sau săritul [1].

Această temă reprezintă un domeniu de cercetare din ce în ce mai important datorită varietății largi de aplicații practice pe care le are și faptului că poate oferi un impact major în diverse domenii. Odată cu avansul tehnologic în domeniul senzorilor și al inteligenței artificiale, acest domeniu devine tot mai relevant, deschizând noi posibilități pentru îmbunătățirea eficienței în numeroase aplicații.

Una dintre cele mai relevante aplicații ale HAR este în domeniul securității publice și al supravegherii video. Sistemele automate pot detecta comportamente anormale sau potențial periculoase cum ar fi altercațiile și pot alerta în timp real operatorii sau autoritățile competente. Astfel, recunoașterea acțiunilor umane contribuie semnificativ la prevenirea incidentelor și la intervenții rapide în situații critice.

În contextul îmbătrânirii populației și al dezvoltării locuințelor inteligente, HAR poate fi folosit în asigurarea independenței și siguranței persoanelor vulnerabile. Sistemele de monitorizare pot detecta în mod automat evenimente precum căderile, lipsa activității sau comportamentele neobișnuite, trimițând alerte către persoanele responsabile sau personalul medical.

În domeniul sportiv, HAR poate fi utilizat pentru a analiza performanțele sportivilor pentru a detecta eventuale greșeli sau a identifica mișcări riscante care pot duce la accidentări.

Sistemele HAR integrate în vehicule pot monitoriza comportamentul șoferului pentru a detecta oboseala, distragerea atenției sau alte semnale care ar putea conduce la accidente. De asemenea, în cazul vehiculelor autonome, recunoașterea acțiunilor pietonilor și ale altor participanți la trafic este esențială pentru luarea deciziilor în timp real.

Obiectivul acestei lucrări este de a testa diferite abordări din domeniul învățării automate pentru a putea clasifica diverse acțiuni umane care au loc în videoclipuri sau în fluxuri video în timp real. Scopul este de a identifica cele mai eficiente modele care pot detecta și clasifica cu acuratețe acțiunile umane în contexte variate și în condiții diverse de iluminare, unghi sau fundal. Rezultatele

obținute pot contribui la dezvoltarea unor sisteme mai robuste și mai precise de recunoaștere a activităților umane.

2. Stadiul actual

În sursa [2] se prezintă o abordare similară cu cea a proiectului descris în acest proiect. Procesul începe cu colectarea informațiilor din două surse diferite, denumite S1 și S2. Datele preluate sunt apoi supuse unui proces atent de pregătire, care include curățarea, formatarea și integrarea acestora într-un set de date unitar, ce reunește informații relevante din ambele surse.

Un accent important este pus pe procesarea imaginilor din acest set de date. Se aplică tehnici precum normalizarea, scalarea și augmentarea pentru a obține imagini uniforme și de calitate, potrivite pentru analiza ulterioară. Setul de date este apoi adnotat prin etichetarea comportamentelor umane suspecte, pregătindu-l pentru antrenarea modelelor de învățare supervizată.

Pentru a evalua corect performanța modelelor, setul de date este împărțit în două părți: una pentru antrenare și cealaltă pentru testare. Această separare asigură că modelele învață dintr-o gamă variată de date, dar sunt evaluate pe informații noi, nevăzute anterior, garantând astfel obiectivitatea rezultatelor.

Rețelele neuronale convoluționale (CNN) sunt folosite pentru a extrage caracteristici relevante din imaginile prelucrate, oferind o reprezentare esențială a conținutului vizual, care este apoi utilizată în modelele de învățare automată.

Pentru identificarea comportamentului uman suspect, sunt testate mai multe arhitecturi de învățare profundă, cum ar fi modelul LSTM, CNN folosind TimeDistributed, și Conv3D. Fiecare dintre aceste modele utilizează caracteristicile extrase de CNN pentru a învăța tipare comportamentale și pentru a genera predicții.

Modelele antrenate sunt apoi aplicate în contexte reale, precum analiza videoclipurilor de pe YouTube sau interpretarea unor secvențe video necunoscute. Această etapă evidențiază aplicabilitatea practică a metodei propuse și potențialul ei de utilizare în situații diverse.

În concluzie, această abordare contribuie la îmbunătățirea sistemelor de supraveghere și securitate, sporind capacitatea de detectare și prevenire a comportamentelor suspecte.

După cum este bine cunoscut, videoclipurile reprezintă o succesiune de cadre (imagini) din care se pot extrage atât informații spațiale, cât și de mișcare. Fluxul spațial se ocupă doar de informația statică, precum aspectul imaginilor și

recunoașterea obiectelor, în timp ce fluxul temporal se concentrează pe mișcare, analizând schimbările dintre cadre pentru a detecta dinamica obiectelor.

În sursa [3] s-a propus o arhitectură de tip CNN cu două fluxuri (two-stream CNN) pentru procesarea separată a informațiilor spațiale și temporale, folosind două rețele convoluționale distincte. Fluxul spațial primește ca input imagini RGB individuale, pentru a recunoaște obiectele statice, iar fluxul temporal primește cadre de tip flux vizual, care reflectă mișcarea în cadrul secvenței. Fiecare flux este antrenat independent, iar rezultatele sunt combinate și introduse într-o rețea Convolutional LSTM (ConvLSTM). Scorul final de clasificare este obținut printr-un strat SoftMax aplicat peste ieșirea ConvLSTM-ului.

Abordarea din sursa [4] prezintă o manieră similară cu cea din sursa precedentă. În modelul propus, fluxurile spațial și temporal sunt implementate folosind rețele convoluționale urmate de un CLSTM, astfel încât să se păstreze informația spațială și să se îmbunătățească detecția temporală. Rezultatele obținute indică o acuratețe superioară față de arhitectura pe 2 fluxuri în varianta standard.

Pentru antrenarea celor două fluxuri, s-au folosit rețele reziduale (ResNet) și anume variantele mai complexe constituite din mai multe straturi, care extrag caracteristici discriminative mai eficiente decât variantele cu puține straturi. O problemă frecventă la rețelele cu foarte multe straturi este degradarea performanței. Pentru a se evita această problemă autorii articolului din sursa [5] au folosit o functie de mapare reziduala diferita de cea folosita in mod normal de retele ResNet. Blocul rezidual este implementat deoarece acționează ca o legătură directa care interconectează primul strat cu oricare alt strat din rețea, evitând astfel metoda tradițională de conectare secvențială între straturi unde fiecare strat se conectează doar la cel precedent. Această abordare ajută la evitarea problemei reducerii gradientului, prin ocolirea unor straturi și transferul direct al erorii de învățare către straturi mai superficiale. Deasemenea evită creșterea numărului de parametri și a complexității computaționale. În rețelele reziduale ResNet, normalizarea pe loturi (Batch Normalization - BN) este aplicată înainte de fiecare strat de activare și după fiecare operație de convoluție. Această procedură accelerează convergența rețelei și rezolvă problema shiftului co-variant (covariate shift). În final, în locul straturilor complet conectate (fully connected) și al straturilor SoftMax, se utilizează o combinație între pooling mediu global (global average pooling) și funcția Softmax. Această reducere simplifică modelul, micșorând considerabil numărul de parametri. De asemenea structura de tip

bottleneck contribuie la reducerea complexitatii și asigură în același timp o performanță ridicată a modelului.

Inputul pentru fluxul temporal este format din cadre de tip optical flow, extrase din imaginile RGB. Aceste cadre sunt prelucrate astfel încât fluxul temporal să poată fi antrenat utilizând rețele pre-antrenate. Pentru acest lucru, este necesară preprocesarea imaginilor RGB pentru a obține cadrele de flux optic. Pentru fiecare imagine se generează două cadre: unul care capturează componentele verticale ale mișcării și unul pentru cele orizontale. Exista mai multe metode de generare al acestui tip de cadre. Există două metode populare pentru extragerea acestor cadre:

- Metoda Brox [6]
- Metoda Total Variation-L1 (TV-L1) [7]

În studiul realizat în articolul [8], autorii au demonstrat că metoda TV-L1 oferă rezultate mai bune decât metoda Brox.

Se generează un set de 20 de cadre (10 verticale și 10 orizontale) pentru fiecare secvență, iar imaginile sunt scalate în intervalul [0, 255] pentru a putea fi procesate de rețelele pre-antrenate.

O altă abordare pe baza căreia s-a realizat proiectul curent este cea din sursa [9]. Autorii articolului propun o abordare prin care se poate estima postura corpului uman, atât în 2D, cât și în 3D, iar apoi aceste estimări să fie folosite în procesul de recunoaștere al acțiunilor umane.

Pentru estimarea posturii în plan 2D, după literatura de specialitate, există 2 familii de metode, și anume metode de detecție și metode de regresie.

Metodele bazate pe detecție tratează estimarea poziției ca pe o problemă de predicție de hartă de căldură (heat map), unde fiecare pixel dintr-o astfel de hartă reprezintă scorul de detecție pentru o articulație corespunzătoare. Totuși, abordările bazate pe detecție nu oferă direct coordonatele articulațiilor. Pentru a obține poziția (x, y), se aplică de obicei funcția argmax ca un pas de post-procesare.

Pe de altă parte, metodele bazate pe regresie folosesc o abordare neliniară care mapează direct intrarea către ieșirea dorită, adică coordonatele articulațiilor. O limitare a metodelor de regresie o reprezintă funcția de regresie, care deseori poate fi sub-optimală. Pentru a se depăși acest neajuns, s-a propus funcția Soft-argmax [10], care permite conversia directă a hărților de căldură în coordonate ale

articulațiilor și, implicit, transformarea metodelor de detecție în metode de regresie.

În cazul punctelor din spațiul tridimensional, s-a folosit o arhitectură volumetrică care oferea rezultate bune, inspirată dintr-o abordare în care s-a folosit o arhitectură în formă de clepsidră.

Pentru a recunoaște acțiunile umane, autorii au propus o metodă formată din 2 componente. Prima componentă o reprezintă recunoașterea acțiunilor pe baza posturii detectate în imagini, iar a doua componentă, recunoașterea pe baza caracteristicilor vizuale.

Recunoașterea pe baza posturilor persoanelor se realizează prin rearanjarea a T posturi, care conțin NJ puncte/articulații, într-o structură matricială tridimensională, în următoarea manieră:

- axa verticală corespunde timpului (cadrelor),
- axa orizontală corespunde articulațiilor,
- iar canalele reprezintă coordonatele (x, y) pentru 2D sau (x, y, z) pentru 3D

Această abordare permite utilizarea rețelelor convoluționale pentru a extrage tipare direct din secvențele temporale de posturi ale corpului. Deoarece metoda de estimare a posturii se bazează pe imagini statice, se poate folosi clasa wrapper TimeDistributed pentru a procesa secvențe video.

În procesarea acestei structuri, autorii au folosit o rețea neuronală complet convoluțională, care extrage caracteristici din structura de mai sus și produce hărți de activare pentru acțiuni.

Pentru a obține probabilitatea finală a fiecărei acțiuni pentru o secvență video, se aplică o operație de pooling pe hărțile de acțiune. Scoaterea în evidență a celor mai bune predicții se face folosind o combinație de pooling maxim + minim, urmată de o activare Softmax.

Recunoașterea pe baza caracteristicilor vizuale se realizează într-o manieră similară. O serie de cadre sunt procesate de o rețea convoluțională similară, din care se extrag caracteristici vizuale.

Unele acțiuni sunt dificil de diferențiat doar pe baza reprezentării datelor legate de postura.

De exemplu acțiunile "*a bea apă"* și "*a vorbi la telefon"* sunt foarte similare dacă luăm în considerare doar articulațiile corpului, însă pot fi ușor de separat dacă avem și informația vizuală corespunzătoare obiectelor cană și telefon.

Pe de altă parte, alte acțiuni nu sunt legate direct de informația vizuală ci mai degrabă de mișcările corpului cum ar fi salutul sau atingerea pieptului iar în aceste cazuri, informația legată de postură poate oferi informații complementare.

Pentru a valorifica contribuția ambelor modele, autorii au decis sa combine predicțiile respective folosind un strat complet conectat cu funcție de activare Softmax, care oferă predicția finală a modelului.

Autorii sustin ca aceasta abordare pe 2 fluxuri de date prezinta 2 avantaje clare.

În primul rând este foarte eficientă din punct de vedere computațional deoarece majoritatea calculelor sunt partajate între sarcini.

În al doilea rând caracteristicile extrase sunt mai robuste, deoarece sunt antrenate simultan pentru sarcini diferite și pe seturi de date diferite.

3. Noțiuni teoretice

3.1. Învățarea automată

Învățarea automată (machine learning) reprezintă un subdomeniu esențial al inteligenței artificiale care permite sistemelor informatice să analizeze date, să identifice tipare și să ia decizii cu un grad ridicat de autonomie, fără a fi programate în mod explicit pentru fiecare scenariu posibil. Aceasta funcționează pe baza unor algoritmi care, prin expunerea la volume mari de date, își ajustează performanța și acuratețea în timp [11].

Procesul de învățare automată implică trei componente principale [11]:

- Procesul decizional: Algoritmii de învățare automată sunt utilizați pentru a face predicții sau clasificări pe baza unor date de intrare, care pot fi etichetate sau neetichetate.
- Funcția de eroare: Aceasta evaluează predicția modelului, comparând-o cu exemple cunoscute pentru a determina acuratețea modelului. Practic, se face o diferență între predicția modelului și rezultatul care se dorește a fi obținut.
- Procesul de optimizare a modelului: Pe parcursul învățării, modelul își ajustează parametrii pentru a reduce diferența dintre predicțiile sale și valorile reale, repetând acest proces până când se atinge un anumit nivel de acuratețe.

Modelele de învățare automată se încadrează în mai multe categorii.

Învățarea supervizată (supervised learning) este una dintre cele mai utilizate metode în domeniul învățării automate. Aceasta implică utilizarea unor seturi de date etichetate, unde fiecare instanță de antrenare este asociată cu o etichetă sau o valoare de ieșire dorită. Modelul este antrenat să prezică aceste ieșiri în funcție de datele de intrare. Pe măsură ce datele sunt introduse în algoritm, modelul își ajustează parametrii interni pentru a minimiza eroarea de predicție.

Învățarea nesupervizată (unsupervised learning) se bazează pe analiza și gruparea datelor neetichetate. În acest caz, algoritmii nu primesc ieșiri cunoscute pentru antrenament, ci trebuie să descopere singuri tipare, structuri sau relații ascunse între date. Această metodă este ideală pentru explorarea datelor (exploratory data analysis), segmentarea clienților, strategii de cross-selling, recunoaștere de imagini și detectarea anomaliilor.

Învățarea semi-supervizată (semi-supervised learning) reprezintă o abordare intermediară între învățarea supervizată și cea nesupervizată. Aceasta utilizează un volum mic de date etichetate, alături de un set mai mare de date neetichetate, pentru a ghida procesul de învățare. Este o metodă extrem de utilă atunci când etichetarea datelor este costisitoare, consumatoare de timp sau dificil de realizat.

Învățarea prin întărire (reinforcement learning) este un model de învățare automată asemănător cu învățarea supervizată, însă algoritmul nu este antrenat pe baza unor date etichetate. Acest model învață treptat, prin încercare și eroare. Secvențele de acțiuni care duc la rezultate reușite sunt întărite, contribuind la dezvoltarea celei mai bune strategii sau recomandări pentru o anumită problemă.

Învățarea automată are aplicabilități în diferite scenarii din lumea reală, cum ar fi:

- Analiza imaginilor medicale: Modelele de învățare automată analizează scanările medicale (precum radiografii, CT-uri, RMN-uri) pentru a asista radiologii în detectarea anomaliilor cum ar fi identificarea tumorilor sau a altor patologii. Acest lucru contribuie adesea la un diagnostic mai precis și la îmbunătățirea rezultatelor.
- Vehicule autonome: Mașinile autonome se bazează intens pe învățarea automată pentru sarcini precum percepția mediului, detectarea obiectelor (pietoni, alte vehicule, semne de circulație), menținerea benzii de circulație și navigarea. Companii precum Tesla demonstrează aplicabilitatea acestei tehnologii.

3.2. Rețele neuronale

O rețea neuronală este un program sau model de învățare automată care ia decizii într-un mod similar cu cel al creierului uman, folosind procese care imită modul în care neuronii biologici colaborează pentru a identifica fenomene, a evalua opțiuni și a ajunge la concluzii [12].

Fiecare rețea neuronală este formată din mai multe straturi de noduri (sau neuroni artificiali): un strat de intrare, unul sau mai multe straturi ascunse și un strat de ieșire.

Rețelele neuronale se bazează pe date de antrenament pentru a învăța și a-și îmbunătăți acuratețea în timp. Odată ce sunt ajustate corespunzător, devin

instrumente foarte puternice în domeniul informaticii și al inteligenței artificiale, permițând clasificarea și gruparea rapidă a datelor. Sarcini precum recunoașterea vocală sau recunoașterea imaginilor pot fi realizate în câteva minute, comparativ cu ore întregi necesare pentru identificarea manuală de către un expert uman. Unul dintre cele mai cunoscute exemple de rețea neuronală este algoritmul de căutare al Google.

Rețelele neuronale mai sunt numite și rețele neuronale artificiale (ANNs – Artificial Neural Networks) sau rețele neuronale simulate (SNNs – Simulated Neural Networks). Ele fac parte din domeniul învățării automate și se află la baza modelelor de învățare profundă (deep learning).

Rețelele neuronale pot fi clasificate în diferite tipuri, fiecare fiind utilizat în scopuri diferite. Deși nu este o listă completă, cele de mai jos reprezintă cele mai comune tipuri de rețele neuronale întâlnite în funcție de cazurile de utilizare:

Perceptronul este cea mai veche rețea neuronală, creată de Frank Rosenblatt în 1958.

Rețelele neuronale feedforward, sau perceptronii multistrat (MLP – Multi-Layer Perceptrons), sunt cele asupra cărora ne-am concentrat în principal în acestei lucrări. Acestea sunt formate dintr-un strat de intrare, unul sau mai multe straturi ascunse și un strat de ieșire. Deși aceste rețele sunt adesea denumite MLP-uri este important de menționat că ele sunt compuse de fapt din neuroni sigmoidali, nu din perceptroni simpli, deoarece majoritatea problemelor din lumea reală sunt neliniare. Datele sunt, de obicei, introduse în aceste modele pentru a le antrena, iar ele stau la baza viziunii computerizate, procesării limbajului natural și a altor tipuri de rețele neuronale.

Rețelele neuronale convoluționale (CNN – Convolutional Neural Networks) sunt asemănătoare cu rețelele feedforward, însă sunt utilizate în mod obișnuit pentru recunoașterea imaginilor, recunoașterea de tipare și/sau viziune computerizată. Aceste rețele se bazează pe principii din algebra liniară, în special înmulțirea matricilor, pentru a identifica tipare într-o imagine.

Rețelele neuronale recurente (RNN – Recurrent Neural Networks) se disting prin buclele lor de feedback. Acești algoritmi de învățare sunt utilizați în principal în lucrul cu date secvențiale (time-series) pentru a face predicții despre rezultate viitoare, cum ar fi previziunile bursiere sau prognozele de vânzări.

3.3. LSTM

LSTM (Long Short-Term Memory) este o variantă îmbunătățită a rețelelor neuronale recurente (RNN), dezvoltată de Hochreiter și Schmidhuber, special concepută pentru a învăța dependențe pe termen lung din date secvențiale. Acest tip de rețea este ideal pentru sarcini precum traducerea automată a limbajului, recunoașterea vocală sau prognoza seriilor temporale [13].

Rețelele neuronale recurente (RNN) sunt concepute pentru a procesa date secvențiale păstrând o stare ascunsă care reflectă informațiile din pașii anteriori ai secvenței. Totuși RNN-urile tradiționale se confruntă cu dificultăți atunci când trebuie să învețe relații între puncte temporale îndepărtate din secvență. Acest lucru este cauzat de două probleme principale:

- Gradientul care dispare (vanishing gradient): în timpul antrenării, valorile gradientului pot deveni foarte mici pe măsură ce se propagă înapoi în timp. Astfel, modelul „uită" informațiile din pașii anteriori, ceea ce îl face incapabil să învețe modele pe termen lung.
- Gradientul care explodează (exploding gradient): în unele cazuri, valorile gradientului pot deveni extrem de mari, ducând la instabilitate și actualizări necontrolate ale modelului.

Ambele fenomene îngreunează învățarea eficientă a relațiilor de lungă durată în secvențe deoarece gradientul arată cât de mult și în ce direcție trebuie să fie ajustate valorile din rețea pentru ca eroarea să scadă.

Arhitectura LSTM introduce un nou element esențial: celula de memorie, care este controlată de trei tipuri de porți:

- Poarta de intrare (Input gate) – decide ce informații noi sunt adăugate în celula de memorie.
- Poarta de uitare (Forget gate) – stabilește ce informații din celulă trebuie șterse.
- Poarta de ieșire (Output gate) – determină ce informații din celulă sunt transmise mai departe către ieșire.

Această structură permite rețelelor LSTM să selecteze în mod inteligent ce informație să rețină, ce să uite și ce să utilizeze mai departe, în funcție de contextul actual și de istoricul datelor. Astfel, LSTM-urile pot învăța dependențe pe termen lung mult mai eficient decât RNN-urile clasice.

Pe lângă celula de memorie LSTM păstrează și o stare ascunsă care poate fi considerată o memorie pe termen scurt actualizată continuu cu ajutorul intrării curente, stării ascunse anterioare și conținutului celulei de memorie.

3.4. ResNet-50

ResNet-50 este un model de învățare preantrenat de tip CNN care face parte din familia ResNet, o serie de modele concepute pentru a rezolva dificultățile întâmpinate în antrenarea rețelelor neuronale. A fost dezvoltat de cercetători de la Microsoft Research Asia. Arhitecturile ResNet sunt disponibile în diferite variante, în funcție de complexitate, cum ar fi ResNet-18, ResNet-32 și altele, ResNet-50 fiind o variantă de dimensiune medie. Această familie de modele este cunoscută pentru performanțele excepționale în clasificarea imaginilor. ResNet-50 a fost antrenat pe o parte din setul de date ImageNet, care este constituit din peste 14 milioane de imagini clasificate în 1000 de clase [14][15].

3.5. TimeDistributed

TimeDistributed este un wrapper ce permite aplicarea unui strat pe fiecare element temporal al unei secvențe de date colectate periodic în timp.

Fiecare intrare trebuie să aibă cel puțin 3 dimensiuni, iar prima dimensiune va fi considerată dimensiunea temporală.

Să luăm în considerare un lot de 32 de videoclipuri de câte 10 cadre, unde fiecare cadru este o imagine RGB cu dimensiunea 128 × 128 pixeli. Forma inputului va fi următoarea: (32, 10, 128, 128, 3).

TimeDistributed poate fi folosit pentru a aplica același strat fiecărui cadru dintre cele 10, în mod independent [16].

3.6. MediaPipe

MediaPipe Solutions oferă un set complet de biblioteci și unelte ce permit aplicarea rapidă a tehnicilor de inteligență artificială (AI) și învățare automată în aplicații. Aceste soluții pot fi integrate direct în proiectele existente, personalizate în funcție de cerințele specifice și utilizate pe diverse platforme de dezvoltare.

În esență, MediaPipe funcționează ca un sistem de programare orientat pe fluxuri de date unde informația parcurge o rețea de componente interconectate, numite calculators. Fiecare calculator realizează o sarcină specifică (de exemplu, detectarea feței, extragerea punctelor de interes sau filtrarea datelor) și trimite rezultatul mai departe în flux. Acest model permite construirea de pipeline-uri flexibile, eficiente și ușor de personalizat.

Un avantaj major al MediaPipe este faptul că oferă componente predefinite și optimizate, care pot fi combinate rapid pentru a crea aplicații funcționale. Datorită suportului multiplatformă (desktop, mobile și web) și performanțelor în timp real, MediaPipe este folosit pe scară largă în domenii precum analiza posturii sau recunoașterea feței.

3.7. YOLO

YOLO (You Only Look Once) este un model bine cunoscut de detectare a obiectelor și segmentare a imaginilor, dezvoltat de Joseph Redmon și Ali Farhadi la Universitatea Washington. Modelul a fost lansat în 2015 și a câștigat rapid popularitate pentru viteza și acuratețea sa.

De-a lungul anilor modelul a evoluat și au apărut mai multe variante, cea mai recenta fiind YOLOv11 care oferă performanțe de ultimă generație în diverse sarcini, inclusiv detectarea obiectelor, segmentarea, estimarea posturii, urmărirea și clasificarea, valorificând capabilitățile inteligenței artificiale în aplicații și domenii variate.

Modelul YOLO abordează detecția obiectelor ca o problemă unică de regresie, estimând direct atât pozițiile obiectelor (bounding boxes), cât și probabilitățile claselor pentru întreaga imagine într-o singură etapă de procesare. Această strategie inovatoare i-a permis să devină considerabil mai rapid decât modelele clasice care utilizează pași separați pentru propunerea și clasificarea obiectelor fără a compromite precizia rezultatelor.

Fiecare versiune nouă a modelului YOLO a adus îmbunătățiri la nivel de arhitectură și metodologie, contribuind la performanțe superioare pe o varietate de metrici. Versiunea YOLO11 continuă această tendință integrând cele mai recente descoperiri din domeniul viziunii artificiale pentru a oferi un echilibru și mai eficient între viteză și acuratețe.

Ultralytics YOLO este cea mai recentă versiune a seriei YOLO, dedicată detecției obiectelor și segmentării imaginilor în timp real. Față de versiunile anterioare, aceasta aduce funcționalități noi, performanțe optimizate și o flexibilitate crescută. Modelul este compatibil cu numeroase sarcini din sfera inteligenței artificiale vizuale, precum detecția, segmentarea, estimarea posturii, urmărirea obiectelor și clasificarea. Datorită arhitecturii moderne, oferă rezultate rapide și precise fiind potrivit atât pentru dispozitive edge cât și pentru aplicații bazate pe cloud [18].

4. Implementare

4.1. Setul de date

Pentru antrenarea modelelor am folosit setul de date UCF50 [19]. UCF50 este un set de date larg utilizat în cercetarea recunoașterii acțiunilor umane în videoclipuri. A fost dezvoltat de University of Central Florida și conține videoclipuri colectate de pe internet (în special YouTube), organizate în 50 de categorii de acțiuni. Aceste clase de actiuni sunt organizate în 25 de grupuri, fiecare grup conținând peste 4 secvențe video. Clipurile din cadrul aceluiași grup pot prezenta similarități semnificative la nivel de caracteristici vizuale, cum ar fi persoana surprinsa in videoclip, decorul de fundal sau perspectiva camerei.

Setul de date UCF50 este organizat fizic pe disc într-o structură de directoare, unde folderul numit UCF50 reprezinta directorul rădăcină. În interiorul acestuia, se găsesc 50 de subdirectoare, fiecare corespunzător unei clase de acțiuni (de exemplu: Basketball, PlayingPiano, Archery etc.). Fiecare subdirector poartă numele acțiunii pe care o reprezintă și conține mai multe fișiere video (.avi), care sunt clipuri etichetate corespunzător.

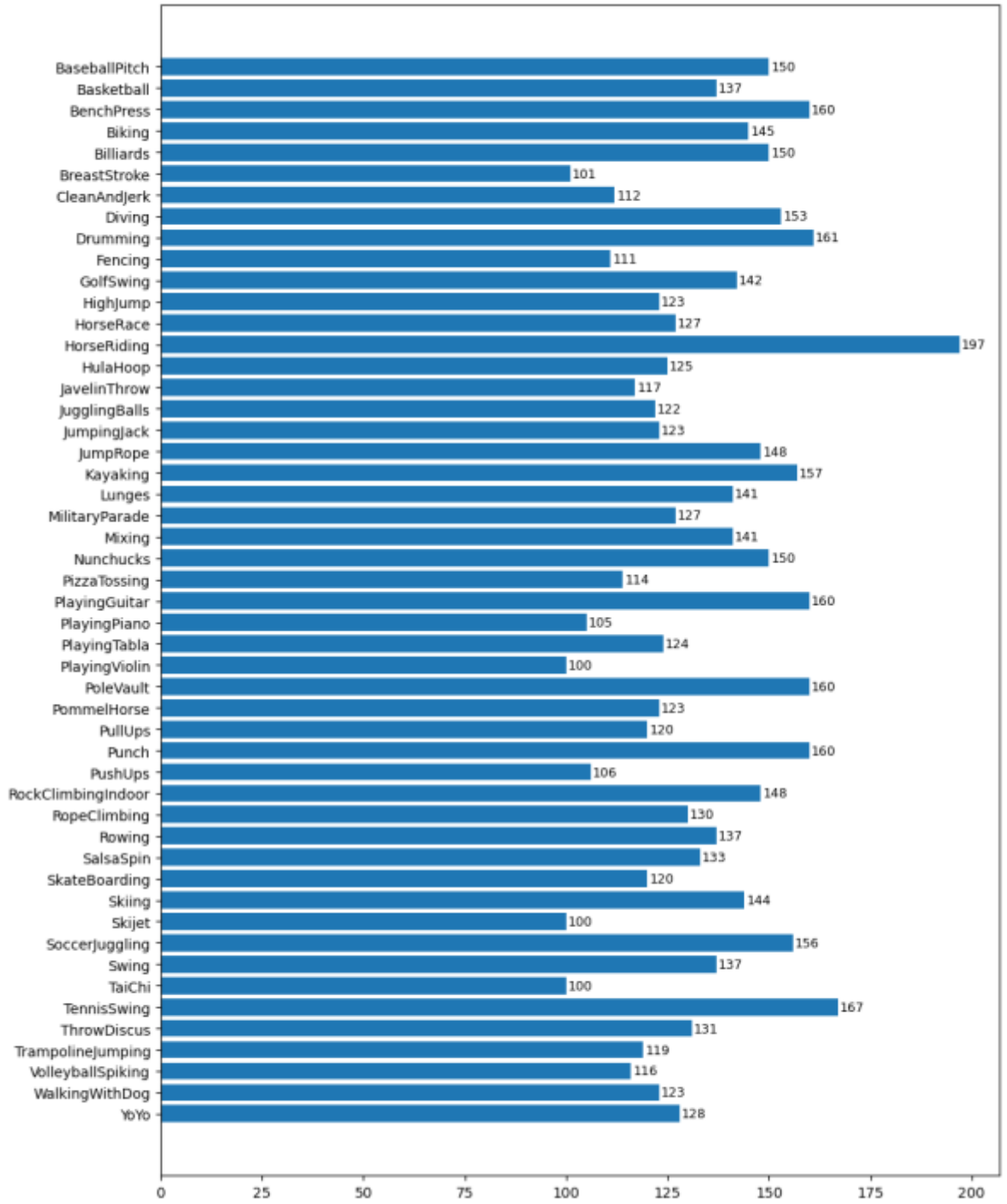


Fig. 4.1.1 Histograma numărului de videoclipuri din fiecare clasă

Pentru denumirea videoclipurilor s-a folosit un șablon specific, care facilitează identificarea mai ușoară a fiecărui videoclip, a acțiunii prezentate și a grupului din care face parte. De exemplu, pentru videoclipul cu numele *v_PlayingPiano_g01_c01*, interpretarea denumirii este următoarea:

- v – indică faptul că fișierul este un clip video.

- PlayingPiano – reprezintă acțiunea care are loc în clip (numele directorului părinte).
- g01 – grupul 01; face referire la grupul din care face parte videoclipul. Numarul grupului este relativ la clasa de actiuni in care se afla clipul.
- c01 – clipul 01; indică faptul că este primul videoclip din acel grup pentru acea acțiune.

Potrivit creatorilor setului de date, majoritatea seturilor existente pentru recunoașterea acțiunilor nu reflectă scenarii reale, fiind în mare parte compuse din secvențe regizate cu actori, în medii controlate. Ei afirmă că obiectivul principal al acestui set este de a oferi comunității de viziune computerizată un set de date realist, compus din videoclipuri preluate de pe YouTube. Conform acestora, setul este deosebit de provocator din cauza variațiilor semnificative în mișcarea camerei, apariția și poziția obiectelor, scara acestora, unghiurile de filmare, fundalurile aglomerate și condițiile diverse de iluminare.

4.2. Preprocesarea setului de date

MediaPipe este un framework open-source dezvoltat de Google, conceput pentru a construi și rula pipeline-uri de învățare automată care procesează date multimedia precum video, text sau audio în timp real.

Pentru a avea acces la mai multe tipuri de date pentru antrenare, nu doar cadre video, am aplicat MediaPipe Pose Landmarker peste toate videoclipurile din setul de date.

MediaPipe Pose Landmarker [20] este un model oferit de cadrul MediaPipe, specializat în detectarea și urmărirea reperelor spatiale (pose landmarks) care descriu postura corpului uman într-o imagine sau secvență video. În urma detectării unei persoane într-o imagine sau într-un cadru video, sunt generate 33 de puncte specifice distribuite pe întregul corp, care aproximează postura persoanei în spațiul imaginii. Aceste puncte sunt cunoscute sub denumirea de pose landmarks (repere de postura) și corespund unor articulații sau regiuni anatomice esențiale, precum: capul, umerii, coatele, încheieturile, șoldurile, genunchii și gleznele, precum și unele puncte de pe trunchi și degetele membrelor superioare și inferioare.

Fiecare landmark este reprezentat printr-un set de 5 valori:

- coordonate tridimensionale (x, y, z)

- visibility – scor de încredere care indică probabilitatea ca punctul să fie vizibil în imagine
- presence – scor care reflectă certitudinea prezenței acelui punct, chiar dacă el nu este vizibil din cauza ocluziilor.

Fig. 4.2.1 Rezultatul folosirii MediPipe Pose Landmarker [20]

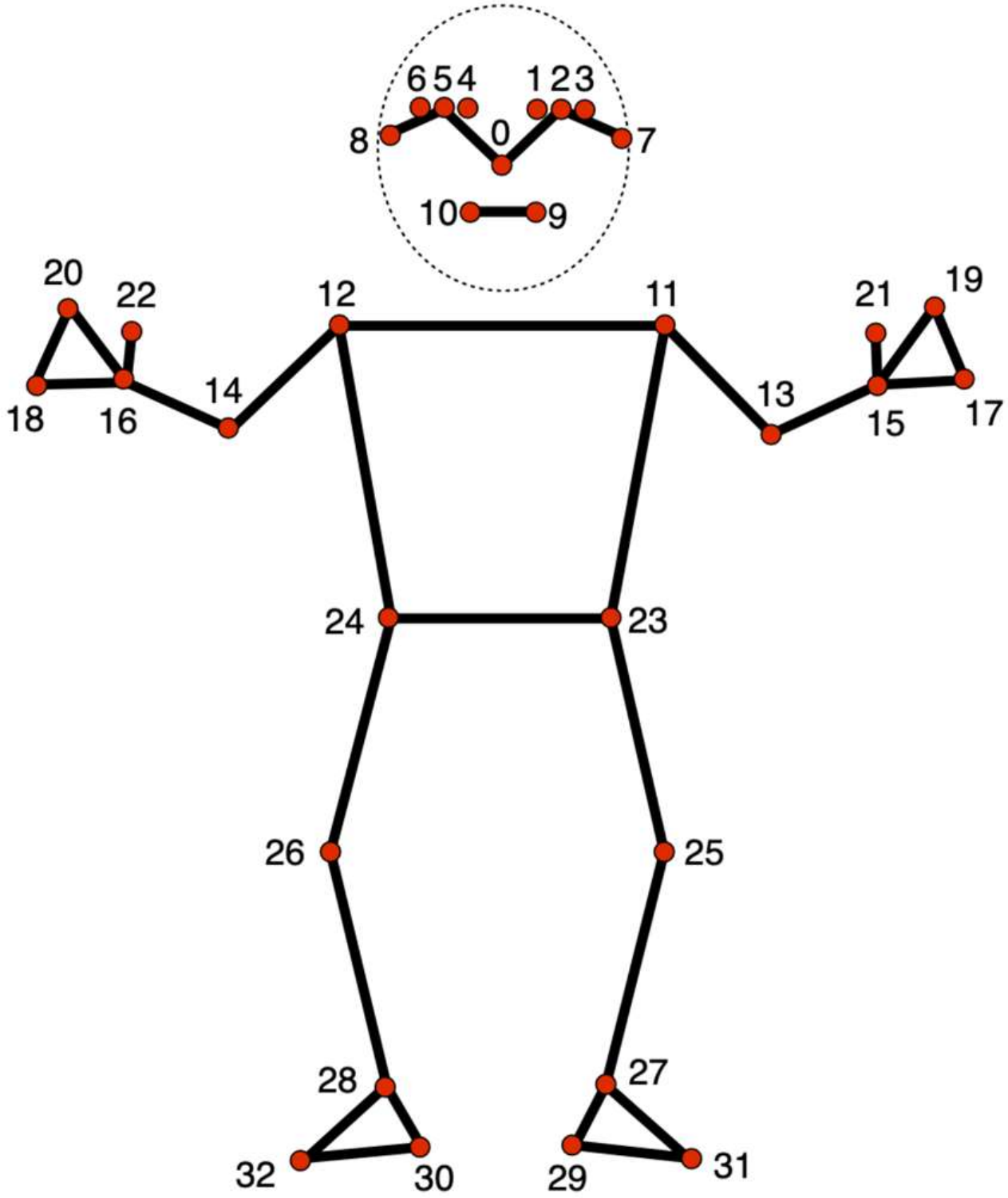


Fig. 4.2.2 Reprezentarea reperelor reprezentative posturii unei persoane [20]

Coordonatele x și y ale fiecărui landmark sunt exprimate sub formă de valori normalizate în raport cu dimensiunile imaginii de intrare, având în mod tipic valori cuprinse în intervalul [0, 1]. Aceste coordonate reflectă poziția relativă

a punctului în planul bidimensional al imaginii, unde (0, 0) corespunde colțului stânga-sus, iar (1, 1) colțului dreapta-jos. În anumite situații, atunci când modelul estimează că un punct este prezent în afara limitelor imaginii, valorile x sau y pot depăși acest interval, devenind mai mici decât 0 sau mai mari decât 1.

Coordonata z descrie adâncimea relativă a punctului față de planul camerei, exprimată într-un sistem de unități arbitrare. Aceasta nu are o corespondență directă cu unitățile fizice (precum metri), ci oferă o măsură relativă a distanței față de cameră. În general, valori negative ale coordonatei z indică puncte mai apropiate de cameră, în timp ce valori pozitive semnalează puncte mai îndepărtate.

Datele returnate de model sunt structurate ca o listă de 33 de obiecte, fiecare având valorile x, y, z, visibility, presence.

4.3. Extragerea landmark-urilor din videoclipuri

Procesul de extragere a reperelor posturii are ca scop convertirea fiecărui videoclip într-o reprezentare numerică standardizată care poate fi ulterior utilizată ca input pentru modele de învățare automată. Fiecare videoclip este privit ca o succesiune de imagini statice (cadre), care surprind mișcarea corpului uman în timp. Pentru a obține o descriere coerentă a întregului videoclip, se aplică o strategie de eșantionare a conținutului video în 15 ferestre temporale.

Concret, toate cadrele unui videoclip sunt împărțite în 15 subseturi consecutive (ferestre). Aceste ferestre au dimensiuni egale (număr de cadre egal), cu excepția ultimei care poate conține un număr diferit de cadre în cazul în care totalul cadrelor nu este divizibil exact la 15. Această metodă asigură acoperirea uniformă a întregii durate a videoclipului și reduce variația temporală în cadrul setului de date.

În cadrul fiecărei ferestre algoritmul MediaPipe Pose Landmarker este aplicat asupra fiecărui cadru în ordine secvențială. Scopul este de a detecta prezența unei persoane și de a extrage un set valid de landmark-uri din fiecare fereastra. Imediat ce într-un cadru din fereastră se reușește detecția unei persoane, acel set de puncte este reținut ca reprezentant al ferestrei respective iar procesul continuă cu următoarea fereastră. Astfel, pentru un videoclip rezultă exact 15 seturi de landmark-uri (daca a fost găsit un set in fiecare fereastră). În cazul în care într-un videoclip nu au fost detectate suficiente seturi de puncte, acel videoclip nu mai este luat in considerare.

Fiecare set este compus din 33 de landmark-uri, fiecare reprezentând un punct anatomic semnificativ al corpului uman (precum umeri, coate, șolduri, genunchi, glezne etc.). Fiecare punct este descris prin cinci atribute(x, y, z, visibility și presence). Rezultatul final al acestui proces de preprocesare este, pentru fiecare videoclip în care au fost detectate 15 seturi de puncte, un tensor de dimensiune (15, 33, 5).

4.4. Normalizarea coordonatelor

Procesul de normalizare reprezintă o transformare matematică aplicată datelor astfel încât acestea să fie aduse într-un interval standard sau într-o formă comparabilă, indiferent de unitățile, dimensiunile sau variațiile originale ale datelor.

După cum am precizat și mai sus, coordonata z exprimă adâncimea punctului în plan, iar aceasta nu are un interval fix de valori. Valorile negative ale acestei coordonate semnifică o distanță mică față de cameră a punctului respectiv, iar valorile pozitive o distanță mai mare. Toate aceste valori z au ca origine punctul median din zona șoldurilor, adică mijlocul segmentului format de punctele 23 și 24 din cadrul unui set de 33 de puncte care constituie postura unei persoane detectate.

Pentru a obține o reprezentare mai robustă și comparabilă între cadre și între videoclipuri, se dorește centrarea tuturor punctelor în origine, adică în punctul (0, 0, 0). Aceasta se realizează prin scăderea coordonatelor punctului median al șoldurilor din toate coordonatele punctelor dintr-un cadru.

În urma acestei centrări, valorile coordonatelor x și y pot depăși intervalul normalizat inițial [0, 1], întrucât poziția punctelor nu mai este exprimată în raport cu marginile imaginii, ci față de un punct intern al corpului.

Astfel, în acest stadiu, niciuna dintre axele de coordonate (x, y, z) nu mai este restrânsă la un interval fix. Pentru a uniformiza datele, ne propunem ca toate valorile să fie aduse în intervalul [-1, 1], o practică frecventă în preprocesarea datelor pentru rețele neuronale. În acest moment niciuna din axele de coordonate nu are un interval fix de valori. Ne propuneam ca toate valorile axelor de coordonate sa se afle in intervalul [-1,1].

Pentru a atinge acest obiectiv:

1. Se extrag cele 15 seturi de puncte (corespunzătoare celor 15 ferestre) pentru fiecare videoclip.
2. Pentru fiecare set se determină valorile minime și maxime ale fiecărei axe (x, y, z) rezultând 6 valori per set: x_min, x_max, y_min, y_max, z_min, z_max.
3. Dintre toate cele 15 seturi, pentru fiecare axă se identifică perechea de valori cu cea mai mare diferență absolută.
   $x\_size = max(\,|\, x\_max_i - x\_min_i \,|\,)$
   $y\_size = max(\,|\, y\_max_i - y\_min_i \,|\,)$
   $z\_size = max(\,|\, z\_max_i - z\_min_i \,|\,)$
   $i \in \{0,1,2,\ldots,14\}$

4. Se folosește această diferență drept factor de scalare global pentru normalizarea tuturor punctelor dintr-un videoclip pe acea axă, astfel încât valorile finale să se încadreze în intervalul [-1, 1].

Coordonate originale: $P = (x, y, z)$

Coordonate normalizate: $P = (\frac{x}{x\_size}, \frac{y}{y\_size}, \frac{z}{z\_size})$

4.5. Extragerea cadrelor din videoclipuri

Procesul de extragere a cadrelor din videoclipuri urmează o strategie similară cu cea utilizată pentru obținerea punctelor de interes (pose landmark-uri) asociate posturii umane. Fiecare videoclip este împărțit uniform în 15 secvențe (ferestre) consecutive, fiecare reprezentând un subset de cadre succesive.

În absența unui criteriu specific de selecție, se optează pentru păstrarea primului cadru din fiecare fereastră. Alternativ, în contextul aplicațiilor care necesită atât informații vizuale, cât și punctele de interes aferente posturii, se pot reține doar cadrele în care a fost detectată prezența unei persoane prin intermediul MediaPipe Pose Landmarker. Această abordare permite asocierea directă între cadrul video și setul de 33 de puncte corespunzătoare posturii persoanei detectate în acel moment.

Pentru a asigura consistența dimensională în procesarea ulterioară, toate cadrele extrase sunt redimensionate la o rezoluție uniforma de 128 × 128 pixeli.

Această etapă este esențială pentru compatibilitatea cu modelele de învățare automată care necesită dimensiuni fixe la nivelul intrărilor, reducând totodată variațiile cauzate de diferențele de rezoluție între videoclipuri.

4.6. Normalizarea valorilor pixelilor

Normalizarea valorilor pixelilor reprezintă un pas esențial în preprocesarea imaginilor, mai ales când acestea urmează să fie utilizate ca intrări pentru modele de învățare automată sau rețele neuronale. Scopul este de a aduce valorile pixelilor într-un interval numeric standard, de regulă între 0 și 1 sau între -1 și 1, pentru a facilita o învățare mai stabilă și mai eficientă a parametrilor modelului.

O imagine color este compusă din trei canale: roșu (R), verde (G) și albastru (B). Fiecare canal dintr-un pixel are, în mod obișnuit, o valoare între 0 și 255 (când imaginea este codificată pe 8 biți per canal). Pentru a normaliza aceste valori în intervalul [0,1], aplicăm următoarea transformare pentru fiecare pixel:

$$\text{canal}_{\text{normalizat}} = \frac{\text{canal}}{255}$$

4.7. Antrenarea modelelor neuronale

Scopul procesului de antrenare a unui model de învățare automată este de a permite algoritmului să învețe un model de decizie care să poată generaliza comportamentul observat în datele de instruire. În cazul de față, obiectivul constă în clasificarea acțiunilor umane pe baza unor secvențe de date extrase din videoclipuri, fie sub forma coordonatelor landmark-urilor corporale detectate de MediaPipe, fie sub forma cadrelor video procesate.

Pentru fiecare videoclip avem:

- (daca sunt detectate)15 seturi de 33 de puncte reprezentând postura unei persoane (cu 5 componente per punct: x, y, z, visibility, presence), rezultând o formă de intrare (15, 33, 5) per eșantion;
- 15 cadre video redimensionate la dimensiunea de 128×128 pixeli, cu 3 canale de culoare (RGB), ceea ce conduce la o formă de intrare (15, 128, 128, 3) per eșantion;

Pentru ca modelele de învățare automată să poată procesa corect datele de ieșire (etichetele), acestea trebuie transformate din format textual într-o formă numerică, compatibilă cu algoritmii de clasificare. În mod obișnuit, această etapă

se numește codificare a etichetelor (label encoding). Setul de date UCF50, utilizat frecvent în sarcini de recunoaștere a acțiunilor umane, conține 50 de clase de acțiuni, fiecare reprezentată printr-un nume textual, precum: Punch, PushUps etc. Pentru a putea fi folosite de model, aceste etichete sunt mapate la indici numerici unici. De exemplu, se poate construi o corespondență de tipul: Punch = 0, PushUps = 1, etc.

Această mapare se realizează, de regulă, cu ajutorul unei funcții sau al unui obiect de codificare automată, precum *LabelEncoder* din biblioteca *scikit-learn*.

În procesul de antrenare, modelul va învăța să asocieze un vector de caracteristici (de exemplu: o secvență de cadre video) cu una dintre aceste etichete numerice. La final, rezultatul inferenței poate fi reconvertit din format numeric în format text, pentru a putea fi interpretat de utilizator.

4.8. Abordări luate în antrenare

Pentru a analiza eficiența diferitelor paradigme arhitecturale în sarcina de clasificare a acțiunilor umane, au fost proiectate și testate cinci modele distincte. Aceste modele utilizează ca date de intrare fie secvențe de puncte de tip pose landmarks, fie cadre video brute, fie o combinație a ambelor tipuri de informație, integrând atât caracteristici spațiale, cât și temporale.

Întreaga lista de videoclipuri din setul de date a fost împărțită în trei subseturi disjuncte, după cum urmează:

- 60% pentru antrenare
- 25% pentru validare
- 15% pentru testare

Setul de antrenare este utilizat pentru învățarea efectivă a parametrilor modelului (ponderi și termeni aditivi). Modelul parcurge acest subset de mai multe ori pe parcursul antrenării, ajustându-și parametrii astfel încât să minimizeze eroarea dintre predicțiile sale și valorile reale.

Setul de validare este folosit pentru a evalua performanța modelului în timpul antrenării, fără ca acesta să învețe din acest subset. Pe baza rezultatelor obținute pe setul de validare, se pot ajusta hiperparametrii modelului (de exemplu: rata de învățare, numărul de epoci, arhitectura rețelei) și se poate detecta apariția

fenomenului de suprainvățare (overfitting), adică momentul în care modelul învață prea bine datele de antrenament și nu mai generalizează corect pe date noi.

Setul de testare este utilizat exclusiv după finalizarea procesului de antrenare pentru evaluarea obiectivă a performanței modelului pe date complet noi. Acest subset permite obținerea unor măsuri de performanță (precum acuratețea, precizia, etc.) care reflectă capacitatea modelului de a generaliza pe situații reale neîntâlnite anterior.

Toate modelele au fost antrenate folosind optimizatorul Adam, funcția de pierdere sparse_categorical_crossentropy și un batch size de 32.

Pentru crearea și antrenarea modelelor am utilizat biblioteca Keras care oferă o interfață simplificată pentru definirea, antrenarea și evaluarea rețelelor neuronale. Prevenirea fenomenul de supraînvățare asupra setului de date de antrenament se face prin integrarea unei funcționalitați disponibilă în Keras, numită *EarlyStopping*.

Acest mecanism permite oprirea automată a procesului de antrenare în cazul în care performanța modelului pe setul de validare nu se îmbunătățește într-un anumit interval de timp. Mai exact, EarlyStopping monitorizează o metrică specifică precum pierderea sau acuratețea și dacă aceasta nu mai prezintă îmbunătățiri de-a lungul unui număr de epoci antrenarea este întreruptă.

În cazul de fața, se monitorizează funcția de pierdere pe datele de validare (val_loss), iar dacă aceasta nu scade timp de 10 epoci consecutive (patience=10), antrenarea se oprește.

4.8.1. Retea neuronala recurenta (RNN)

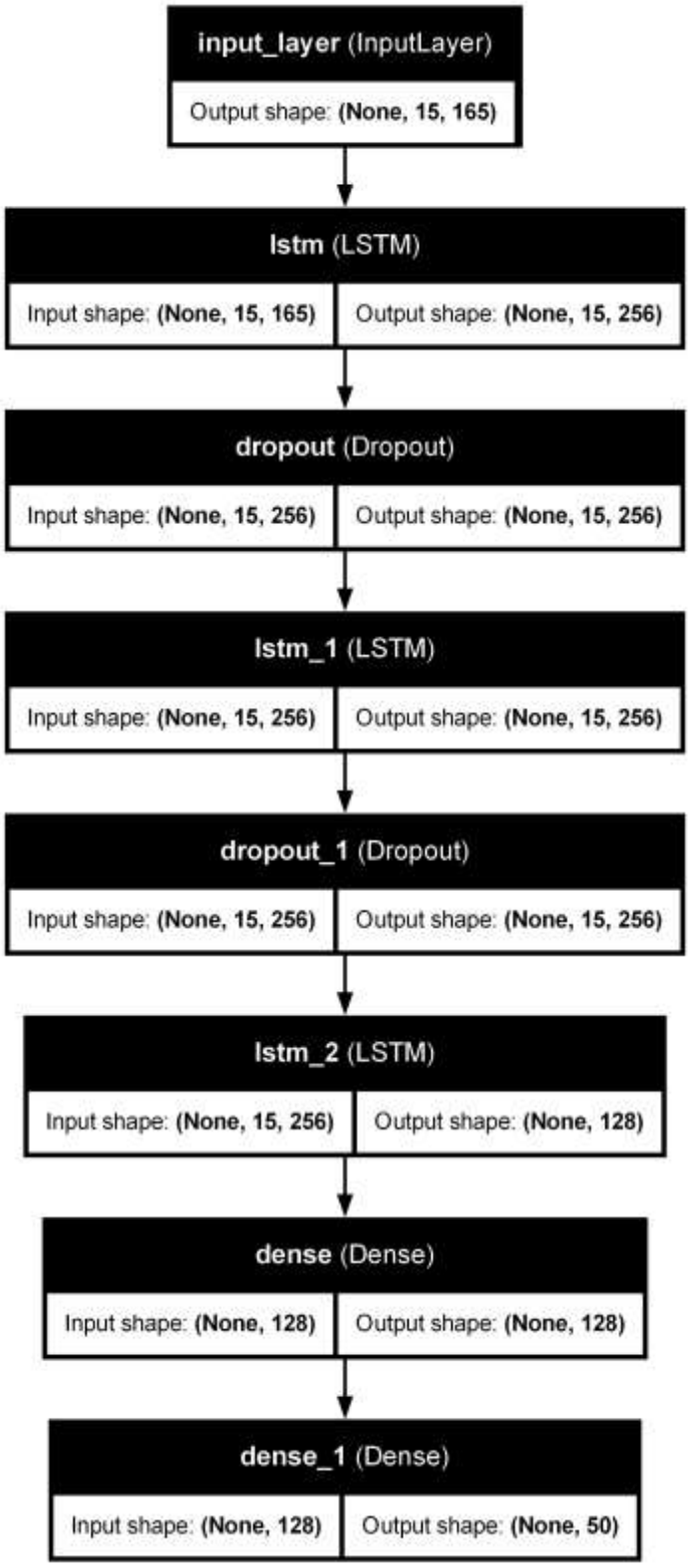


Fig. 4.8.1.1 RNN

Această arhitectură utilizează exclusiv secvențele de puncte de interes (landmark-uri) detectate cu ajutorul MediaPipe Pose Landmarker. Multimea de puncte de interes extrase din videoclipuri este structurată într-un tensor cu 4 dimensiuni, având forma (număr_videoclipuri, 15, 33, 5) unde:

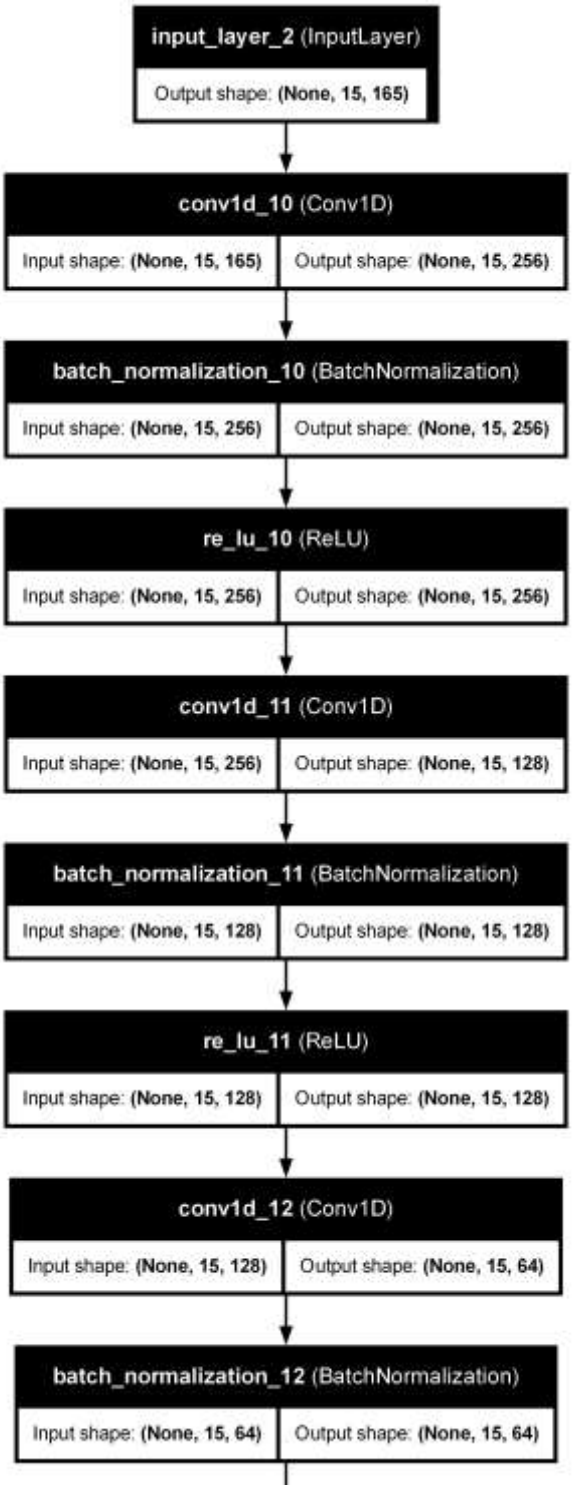
- 15 reprezintă numărul de cadre selectate per videoclip (fereastra temporală),
- 33 este numărul total de puncte de interes per cadru,
- 5 sunt atributele asociate fiecărui punct: x, y, z, visibility, presence.

Modelul este construit pe baza unei înșiruiri de straturi LSTM (Long Short-Term Memory), special concepute pentru a gestiona date secvențiale și a învăța dependențe temporale pe termen lung. În documentație se precizează că această rețea trebuie să primească ca intrare un tensor tridimensional de forma: (batch, timesteps, features) unde:

- batch reprezintă numărul de secvențe procesate simultan în timpul antrenării (adică numărul de videoclipuri dintr-un mini-lot);
- timesteps corespunde lungimii secvenței temporale — în cazul nostru, cele 15 cadre extrase din fiecare videoclip;
- features este numărul de caracteristici extrase din fiecare timestep ; pentru datele noastre de tip pose landmarks, fiecare cadru conține 33 de puncte, fiecare având 5 atribute (x, y, z, visibility, presence), rezultând în total 33 × 5 = 165 caracteristici.

De aici rezultă că, înainte de a antrena rețeaua, este necesară o redimensionare a datelor de intrare pentru a le adapta cerințelor arhitecturii LSTM.Pentru a obține forma necesară de (batch, timesteps, features), datele sunt transformate astfel încât ultima dimensiune să devină 33 × 5 = 165, ceea ce corespunde tuturor caracteristicilor concatenate pentru un cadru.

4.8.2. Rețea neuronală convoluțională 1D (1D CNN)



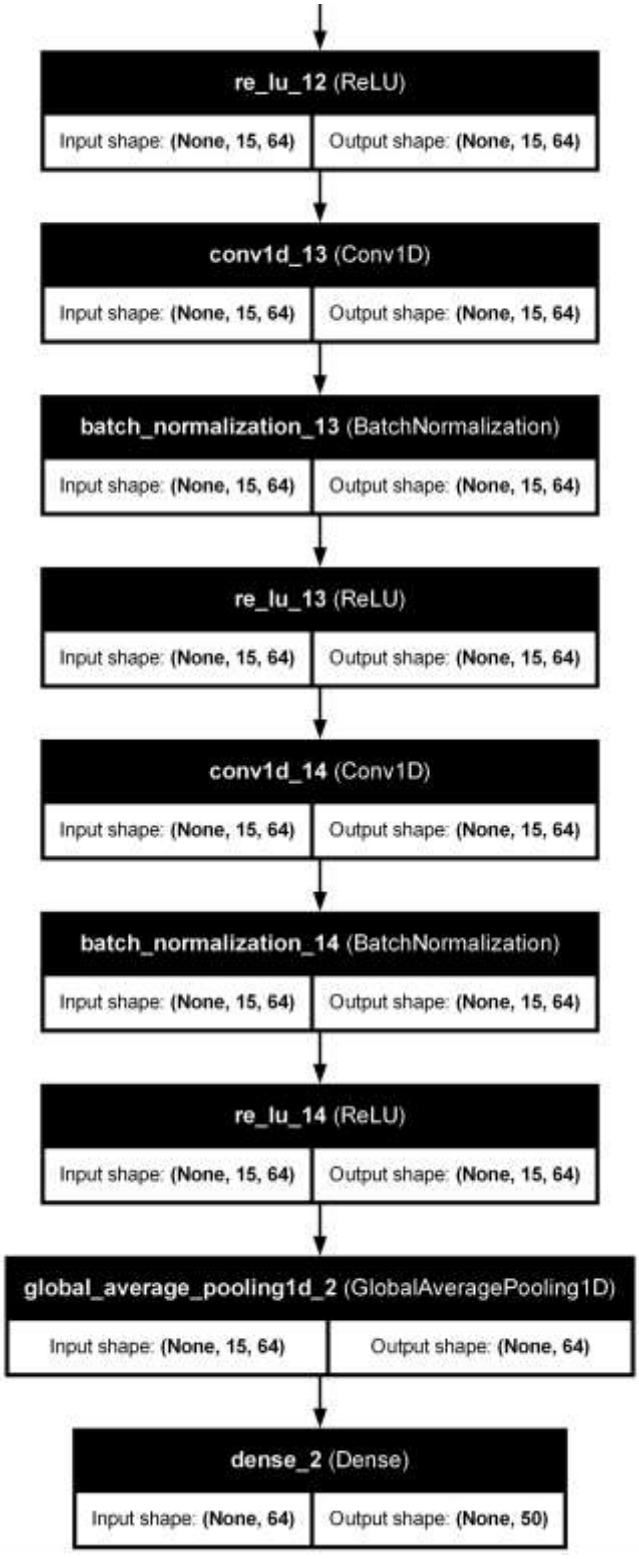


Fig. 4.8.2.1 1D CNN

Această arhitectură utilizează, la fel ca modelul RNN, exclusiv secvențele de puncte de interes (landmark-uri) extrase cu ajutorul MediaPipe Pose Landmarker. Implementarea relețelei este bazată pe cea din sursa [21]. Rețeaua

este alcătuită dintr-o succesiune de straturi convoluționale unidimensionale (1D CNN), special concepute pentru procesarea datelor secvențiale. Acest tip de rețea este deosebit de potrivit pentru analiza informațiilor care variază în timp și sunt măsurate la intervale regulate, exact caracteristicile pe care le au secvențele de landmark-uri extrase din videoclipuri.

Similar cu cazul prezentat anterior pentru rețeaua RNN, și în cazul arhitecturii 1D CNN, modelul necesită ca datele de intrare să fie structurate sub forma unui tensor tridimensional de dimensiune (batch, timesteps, features).

Pentru a asigura compatibilitatea cu cerințele arhitecturii CNN 1D, este necesară o preprocesare a setului de date, constând într-o redimensionare a tensorului original de formă (număr_videoclipuri, 15, 33, 5) într-un tensor de formă (număr_videoclipuri, 15, 165).

4.8.3 Rețea neuronală convoluțională (CNN) urmată de o rețea recurenta (RNN)

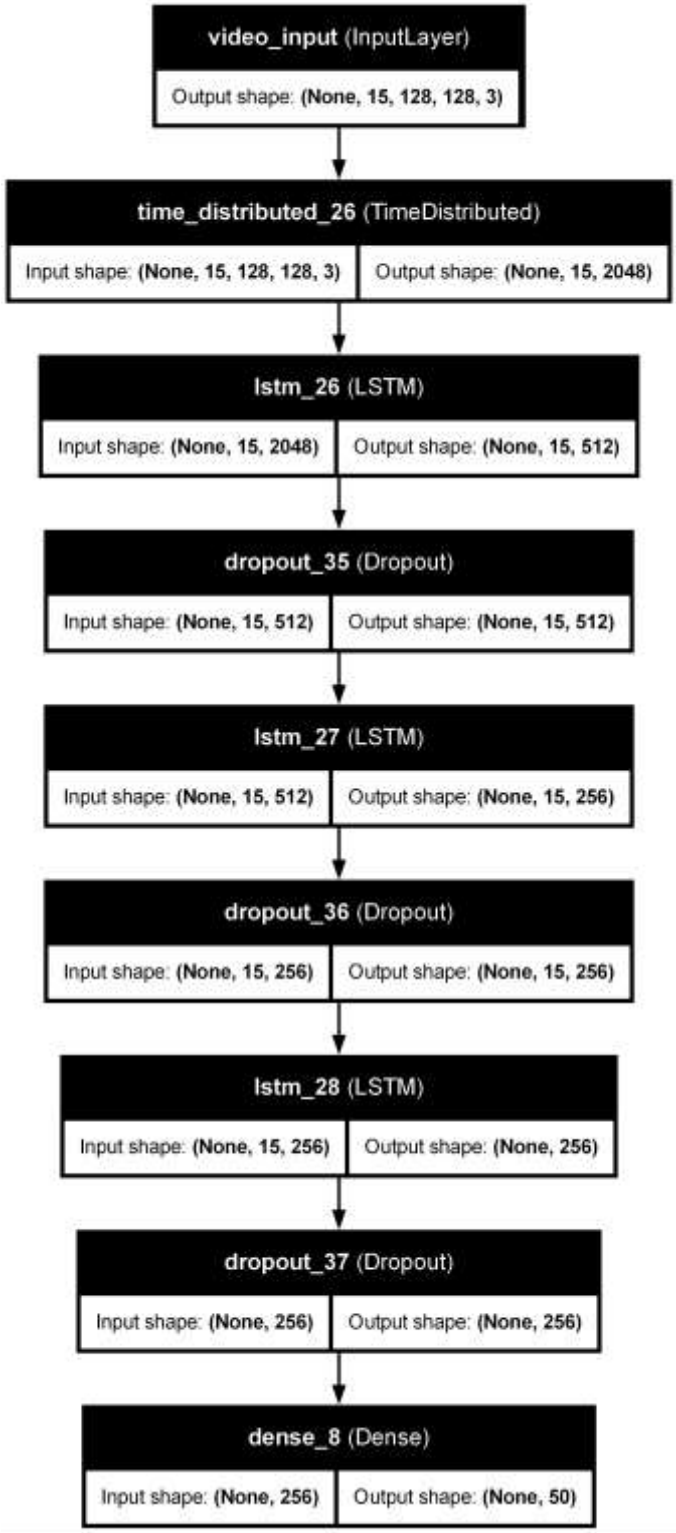


Fig. 4.8.3.1 CNN urmată de RNN

Această arhitectură utilizează ca date de intrare cadrele brute extrase din videoclipuri, organizate sub forma unui tensor cu dimensiunile (număr_videoclipuri, 15, 128, 128, 3), unde:

- 15 reprezintă numărul de cadre selectate per videoclip,
- 128 × 128 este dimensiunea fiecărui cadru,
- 3 corespunde canalelor de culoare RGB.

Modelul combină o rețea convoluțională preantrenată pentru clasificarea imaginilor cu o rețea neuronală recurentă (RNN) de tip LSTM.

Rețeaua preantrenată utilizată pentru extragerea caracteristicilor spațiale din imagini color este ResNet50, un model bine cunoscut pentru performanța sa în sarcinile de clasificare a imaginilor statice care a fost prezentat anterior. Totuși, în cazul de față, nu avem de-a face cu o singură imagine, ci cu o succesiune de 15 cadre per videoclip.

Cum ResNet50 a fost concepută pentru a procesa imagini individuale, pentru a o putea aplica asupra fiecărui cadru dintr-un videoclip, este necesară utilizarea stratului TimeDistributed. Acest strat permite aplicarea aceleiași rețele (în cazul nostru, ResNet50 fără ultimul strat de clasificare) independent și în paralel pe fiecare cadru din secvență.

Astfel, pentru fiecare videoclip reprezentat ca un tensor de formă (batch, 15, 128, 128, 3), stratul TimeDistributed (ResNet50) va procesa fiecare dintre cele 15 imagini în mod separat, generând pentru fiecare un vector de caracteristici spațiale. Rezultatul este un nou tensor de formă (batch, 15, features), care poate fi apoi trimis către rețeaua recurentă pentru a învăța dependențele temporale dintre aceste cadre.

4.8.4 Model Multi-Input: CNN + RNN

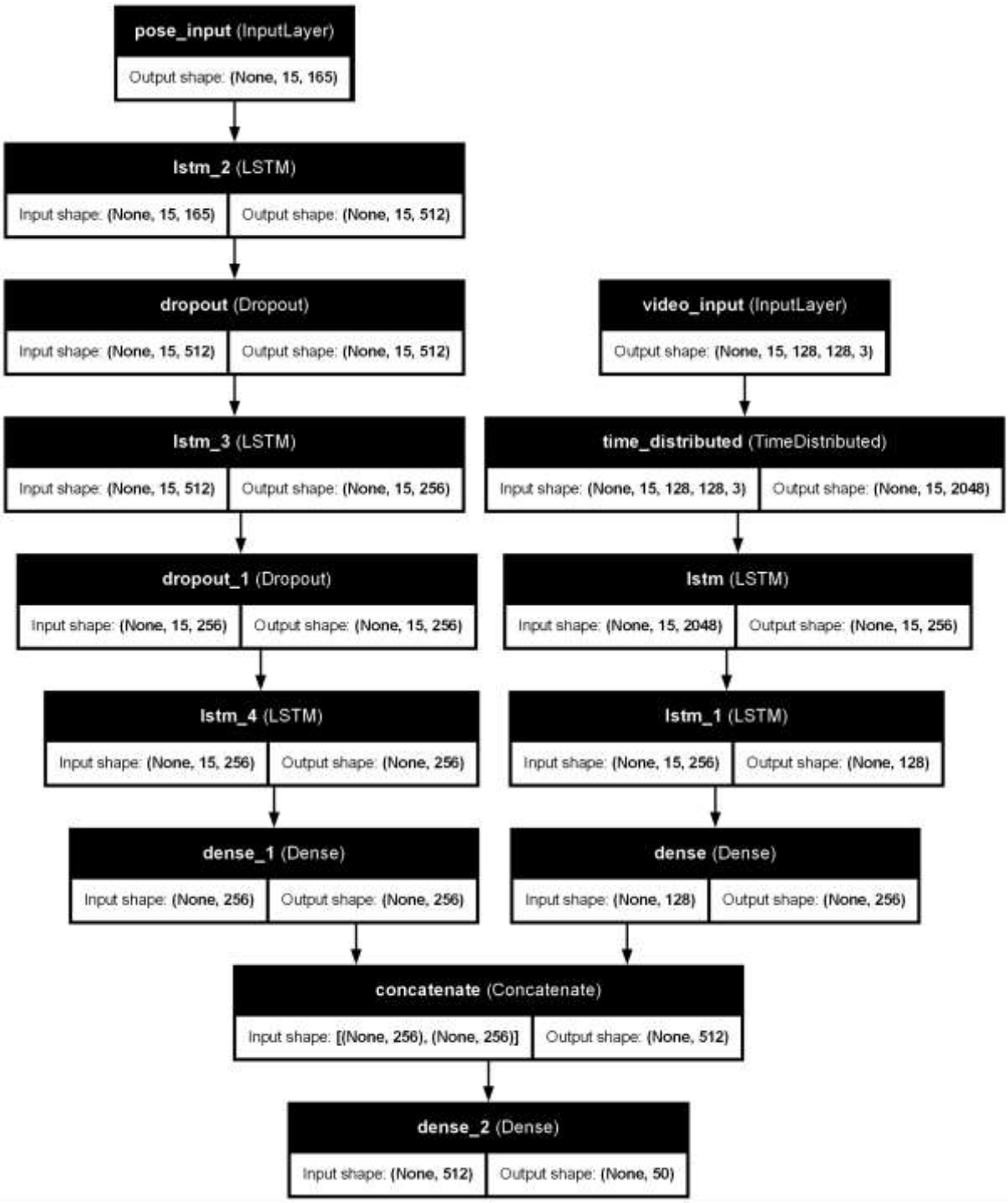


Fig. 4.8.4.1 Model Multi-Input CNN + RNN

Această arhitectură combină informația vizuală brută (cadre video) cu reperele spațiale extrase din detectia posturii corpului (pose landmarks), cu scopul de a valorifica complementaritatea dintre caracteristicile spațiale și cele de poziționare/anatomice. Modelul primește ca input două surse de date:

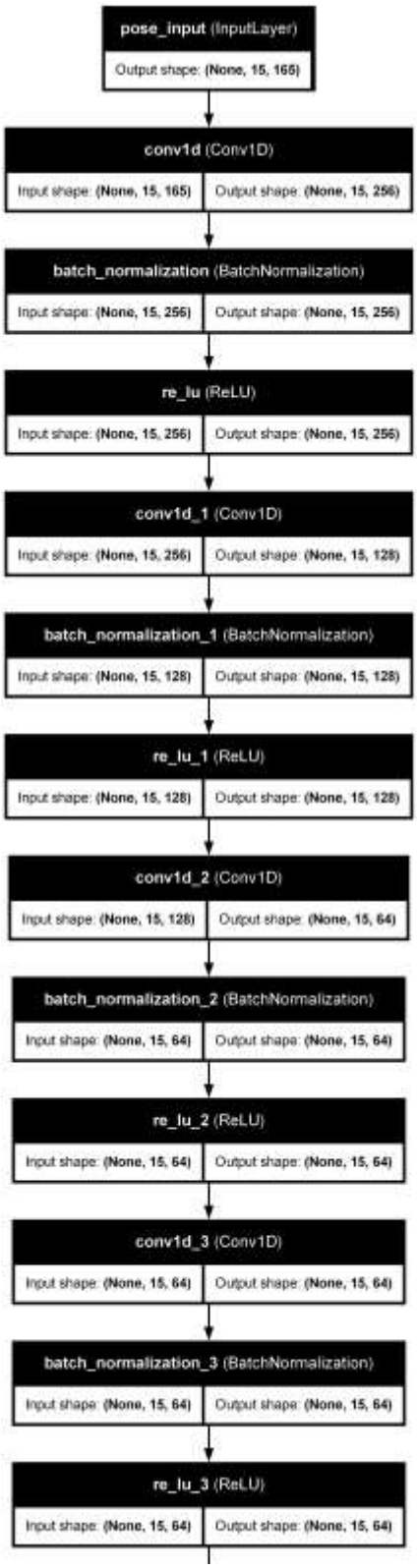
- Cadre video structurate într-un tensor de forma (număr_videoclipuri, 15, 128, 128, 3), reprezentând 15 cadre color per videoclip.
- Date de tip pose landmarks structurate într-un tensor de forma (număr_videoclipuri, 15, 165), unde 165 rezultă din concatenarea a 33 de puncte × 5 atribute (x, y, z, visibility, presence) pentru fiecare cadru.

Modelul este împărțit în două ramuri principale:

- Ramura 3D CNN pentru cadre video: Fiecare cadru este procesat cu o rețeaua convoluțională ResNet50 aplicată cu ajutorul wrapperului TimeDistributed.
- Ramura RNN pentru: Landmark-urile normalizate și redimensionate sunt transmise direct către o rețea LSTM, care extrage dependențe temporale dintre acestea.

La final, vectorii de caracteristici generați de cele două ramuri sunt concatenați și folosiți clasificare. Acest tip de model permite rețelei să combine detaliile vizuale cu informațiile despre poziția și mișcarea corpului uman.

4.8.5. Model Multi-Input: CNN + 1D CNN



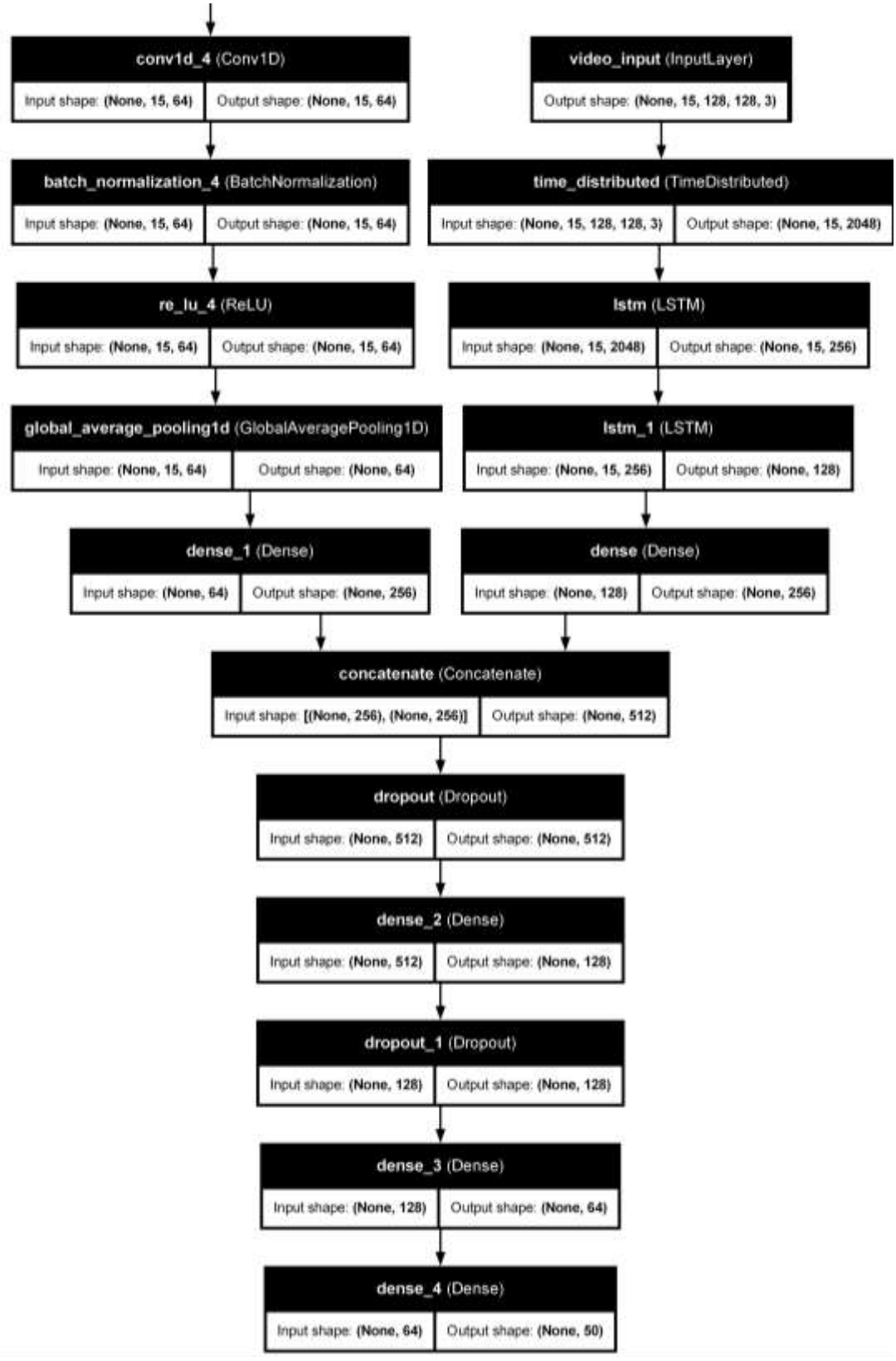


Fig. 4.8.5.1 Model Multi-Input CNN + 1D CNN

La fel ca în cazul anterior, acest model este structurat pe două ramuri principale, fiecare procesând o sursă diferită de informație din videoclipuri:

- Ramura CNN pentru cadrele video: Această ramură utilizează o rețea convoluțională tridimensională, bazată pe arhitectura preantrenată ResNet50, adaptată pentru a procesa secvențe video. Rolul său este de a extrage caracteristici spațiale și temporale din cadrele succesive, capturând contextul vizual general al acțiunii.
- Ramura 1D CNN pentru pose landmarks: În paralel, punctele de interes (landmark-urile) extrase din fiecare cadru sunt transmise către o rețea convoluțională 1D, prezentată anterior. Aceasta analizează evoluția în timp a pozițiilor corporale și extrage tipare relevante din datele secvențiale.

În etapa finală, vectorii de caracteristici extrași din cele două ramuri sunt concatenați și transmiși unui strat dens care realizează clasificarea.

4.9. Integrarea unui model in aplicatie

După finalizarea procesului de antrenare și validare, următorul pas esențial constă în integrarea modelului într-o aplicație practică, astfel încât acesta să poată fi utilizat pentru clasificarea acțiunilor în timp real sau asupra unor fișiere video preînregistrate.

4.10 Salvarea modelului antrenat

După finalizarea cu succes a procesului de antrenare și evaluare, modelul de învățare profundă trebuie salvat pentru a putea fi reutilizat ulterior, fără a fi necesară reantrenarea sa de la zero. Salvarea modelului permite integrarea acestuia într-o aplicație practică și utilizarea pe date noi în scenarii de producție.În cazul folosirii bibliotecii *Keras*, modelul este salvat sub forma unui fișier .keras , care conține atât arhitectura, cât și parametrii rețelei neuronale:

model.save("action_recognition_model.keras")

Unde model este obiectul care reprezinta modelul neuronal antrenat.

Pe lângă modelul în sine, este esențial să fie salvat și obiectul folosit pentru codificarea etichetelor claselor, de obicei realizată cu ajutorul clasei LabelEncoder din sklearn.preprocessing care transformă etichetele textuale (de exemplu: "alergare", "sarituri") în valori numerice (0, 1, 2 etc.), necesare pentru antrenarea

modelului. Pentru ca predicțiile generate de model să poată fi interpretate corect este necesară restaurarea exactă a aceleiași mapări dintre etichete și indecși. Exemplu de salvare a obiectului *LabelEncoder* folosind biblioteca *pickle*:

```
import pickle
with open("label_encoder.pkl", "wb") as f:
    pickle.dump(label_encoder, f)
```

4.11. Încărcarea modelului salvat

Pentru a putea utiliza modelul antrenat într-o aplicație practică sau într-un modul de testare, este necesară încărcarea acestuia din fișier, împreună cu obiectul LabelEncoder folosit la codificarea claselor. Această etapă asigură reproducerea fidelă a comportamentului modelului și interpretarea corectă a rezultatelor.

Modelul salvat anterior într-un fișier *.keras* poate fi reîncărcat cu ajutorul funcției *keras.models.load_model()*:

```
import keras
action_model=keras.models.load_model("action_recognition_model.keras")
```

Obiectul *LabelEncoder*, salvat anterior cu ajutorul bibliotecii *pickle*, trebuie de asemenea restaurat pentru a traduce rezultatul numeric al predicțiilor în etichete lizibile:

```
import pickle
with open("label_encoder.pkl", "rb") as f:
label_encoder = pickle.load(f)
```

4.12. Utilizarea YOLO

Pentru a putea utiliza modelele de clasificare a activităților în scenarii din viața reală, este esențial să fim capabili să detectăm persoanele în videoclipuri preînregistrate sau în fluxuri video în direct.

Detecția persoanelor reprezintă primul pas in procesul de recunoastere a activitților efectuate de acestea deoarece doar după identificarea prezenței și

poziției acestora în cadru se pot extrage informațiile necesare, cum ar fi punctele de interes (pose landmarks) sau imagini cu persoanele detectate decupate din cadru necesare pentru clasificarea activităților.

Pentru a rezolva aceasta problema am folosit YOLO. YOLO (You Only Look Once) este un algoritm de detecție a obiectelor în timp real prezentat anterior.

Autorii propun o abordare diferita, tratând problema detecției obiectelor ca o sarcină de regresie, nu ca una de clasificare. În loc să analizeze imaginea în mai multe etape, modelul YOLO împarte imaginea în regiuni spațiale, prezicând pentru fiecare regiune chenare de delimitare (bounding boxes) și probabilități asociate prezenței obiectelor, folosind o singură rețea neuronală convoluțională (CNN). Această arhitectură unificată permite detecții extrem de rapide și precise, ceea ce face ca YOLO să fie ideal pentru aplicații în timp real.

Modelul YOLO este încărcat într-un mod similar cu celelalte componente ale aplicației, folosind interfața furnizată de biblioteca *ultralytics*:

```
from ultralytics import YOLO

yolo_model = YOLO("yolo11n.pt")
```

După încărcare, modelul poate fi utilizat pentru a efectua detecția obiectelor sau, mai precis, a persoanelor din imagini sau videoclipuri. În aplicație, YOLO este apelat în modul de urmărire (tracking), astfel conform sursei [22]:

```
video_path = "path/to/video.mp4"

cap = cv2.VideoCapture(video_path)

while cap.isOpened():

    success, frame = cap.read()

    if success:

        results = yolo_model.track(frame, persist=True)

        annotated_frame = results[0].plot()

        cv2.imshow("YOLO11 Tracking", annotated_frame)

        if cv2.waitKey(1) & 0xFF == ord("q"):

            break
```

```
    else:
        break
cap.release()
cv2.destroyAllWindows()
```

Apelul funcției *track()* are următoarele funcționalități:

- Detecția obiectelor: identificarea tuturor obiectelor din imagine, inclusiv persoanele
- Urmărirea obiectelor (tracking): alocarea unui identificator unic (ID) fiecărui obiect detectat, menținut pe parcursul mai multor cadre video consecutive. Această funcționalitate este activată prin argumentul *persist=True*.

Modelul YOLO returnează, în urma apelului funcției *track()*, un obiect de tip *Results*, care conține informații detaliate despre toate detecțiile efectuate într-un anumit cadru video. Aceste informații sunt stocate în atributul *boxes* al rezultatului și includ următoarele elemente esențiale:

- *results[0].boxes.cls* – o listă care conține clasele obiectelor detectate, reprezentate prin indici numerici. De exemplu, valoarea 0 corespunde clasei „person”.
- *results[0].boxes.xyxy* – coordonatele fiecărei casete delimitatoare (bounding box) asociate obiectelor detectate, sub forma [x1, y1, x2, y2], unde (x1, y1) reprezintă colțul din stânga sus, iar (x2, y2) reprezintă colțul din dreapta jos al fiecărei regiuni în care a fost identificat un obiect.
- *results[0].boxes.id* – identificatorii unici de urmărire atribuiți fiecărei instanțe detectate, utilizați pentru a menține continuitatea detecției de la un cadru la altul.
- *results[0].boxes.conf* – scorurile de încredere asociate fiecărei detecții, care exprimă nivelul de certitudine al modelului în privința clasificării obiectului.

4.13 Logica de funcționare a aplicației

Aplicația este proiectată să funcționeze atât cu fluxuri video în timp real, cât și cu videoclipuri preînregistrate. Scopul principal este detectarea persoanelor în fiecare cadru video și, ulterior, recunoașterea acțiunilor realizate de acestea pe baza datelor culese de-a lungul fluxului video.

Pentru detectarea persoanelor în fiecare cadru, aplicația utilizează modelul YOLO, care este aplicat cadru cu cadru in maniera prezentata anterior, iar pentru fiecare frame procesat se obține un obiect de tip Results, care conține toate detecțiile efectuate (persoane, obiecte etc.).

După obținerea obiectului Results, se aplică o filtrare pentru a reține doar:

- detecțiile care au un scor de încredere mai mare de 60%;
- detecțiile care aparțin clasei persoană (clasa 0 în modelul YOLO).

Fiecare persoană detectată este încadrată într-un dreptunghi reprezentat de următoarele valori.

- x1, y1 – colțul stânga sus,
- x2, y2 – colțul dreapta jos.

Aceste coordonate sunt folosite ulterior pentru a extrage regiunea de interes din imagine, adică zona în care se află persoana detectată.

Pentru a urmări fiecare persoană detectată și pentru a-i stoca datele asociate (landmark-uri și/sau imagini), aplicația utilizează o structură de tip dicționar în care:

- cheia este id-ul unic al persoanei atribuit de algoritmul de urmărie al modelului YOLO;
- valoarea este un obiect de tip Box, definit astfel:

```
class Box:
    def __init__(self, id):
        self.id = id
        self.pl_queue = deque(maxlen=NUMBER_OF_FRAMES)
        self.frames_queue = deque(maxlen=NUMBER_OF_FRAMES)
```

Componentele clasei Box:

- id: identificator unic al persoanei urmărite;
- pl_queue: coadă în care se stochează landmark-urile scheletice detectate pentru acea persoană (folosite pentru clasificare pe bază de poziție);
- frames_queue: coadă în care se stochează imaginile decupate și redimensionate corespunzătoare fiecărui cadru (folosite pentru clasificare vizuală).

În funcție de modelul de clasificare utilizat, aplicația poate folosi doar una dintre aceste cozi sau pe amândouă. Variabila *NUMBER_OF_FRAMES* are valoarea 15 și se referă la numărul de ferestre în care a fost împărțit fiecare videoclip.

Atunci când pentru un anumit chenar de detecție au fost introduse date noi și în cozile *pl_queue* și/sau *frames_queue* s-au acumulat suficiente informații, adică câte 15 imagini sau seturi de pose landmark-uri, se poate realiza o predicție referitoare la activitatea desfășurată de persoana detectată în acel chenar. În cazul în care pentru efectuarea predicției sunt necesare seturile de pose landmark-uri, înainte de realizarea predicției este necesară normalizarea coordonatelor punctelor, așa cum a fost prezentată în etapa de antrenare a modelelor de clasificare.

4.14 Prezentarea aplicației

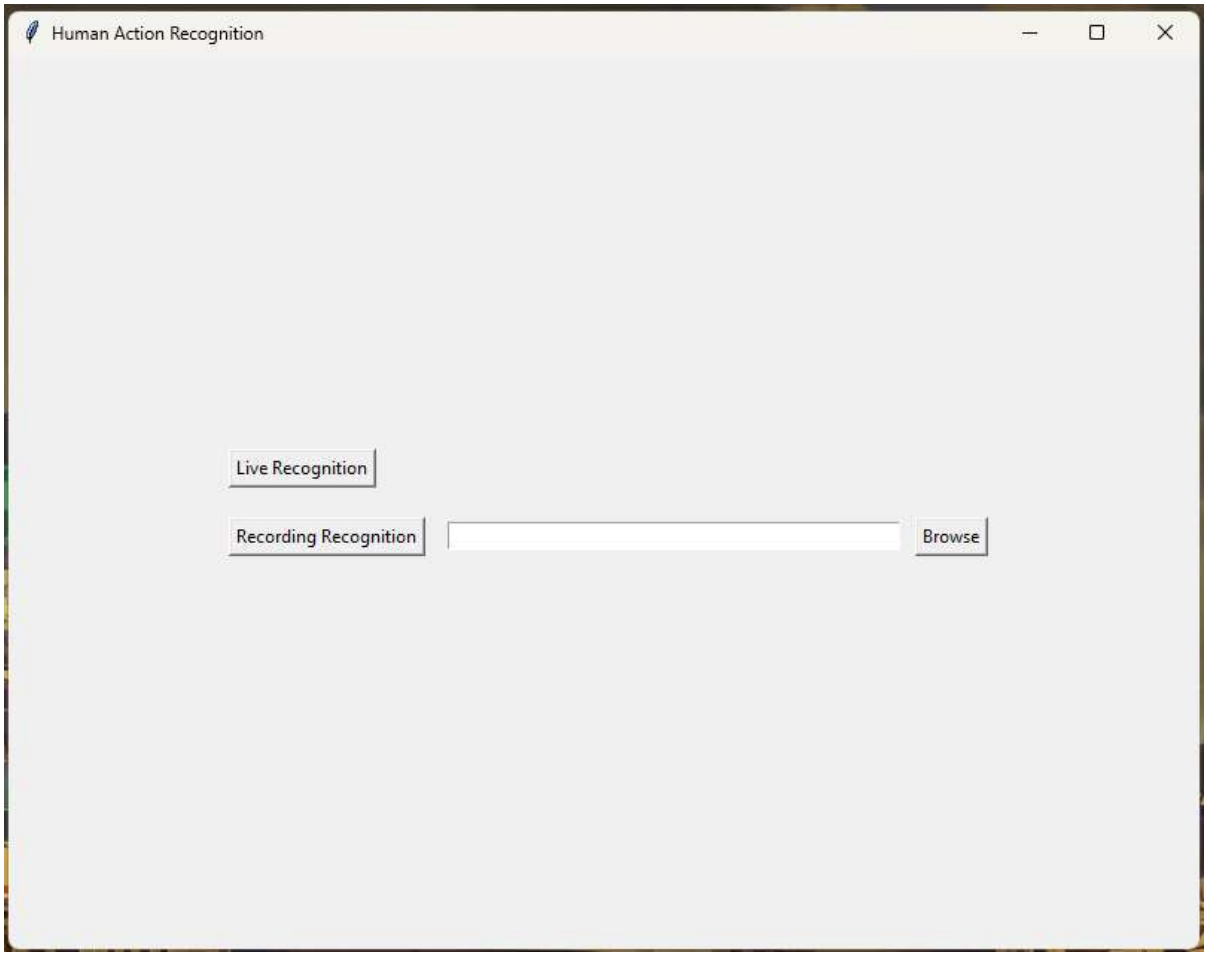


Fig. 4.14.1 Meniul principal

Aplicația permite detectarea acțiunilor efectuate de persoane identificate într-un flux video live, prin intermediul butonului *Live Recognition*, sau într-un videoclip preînregistrat, selectând un fișier video de pe dispozitiv prin butonul *Browse*, apoi apăsând butonul *Recording Recognition*.

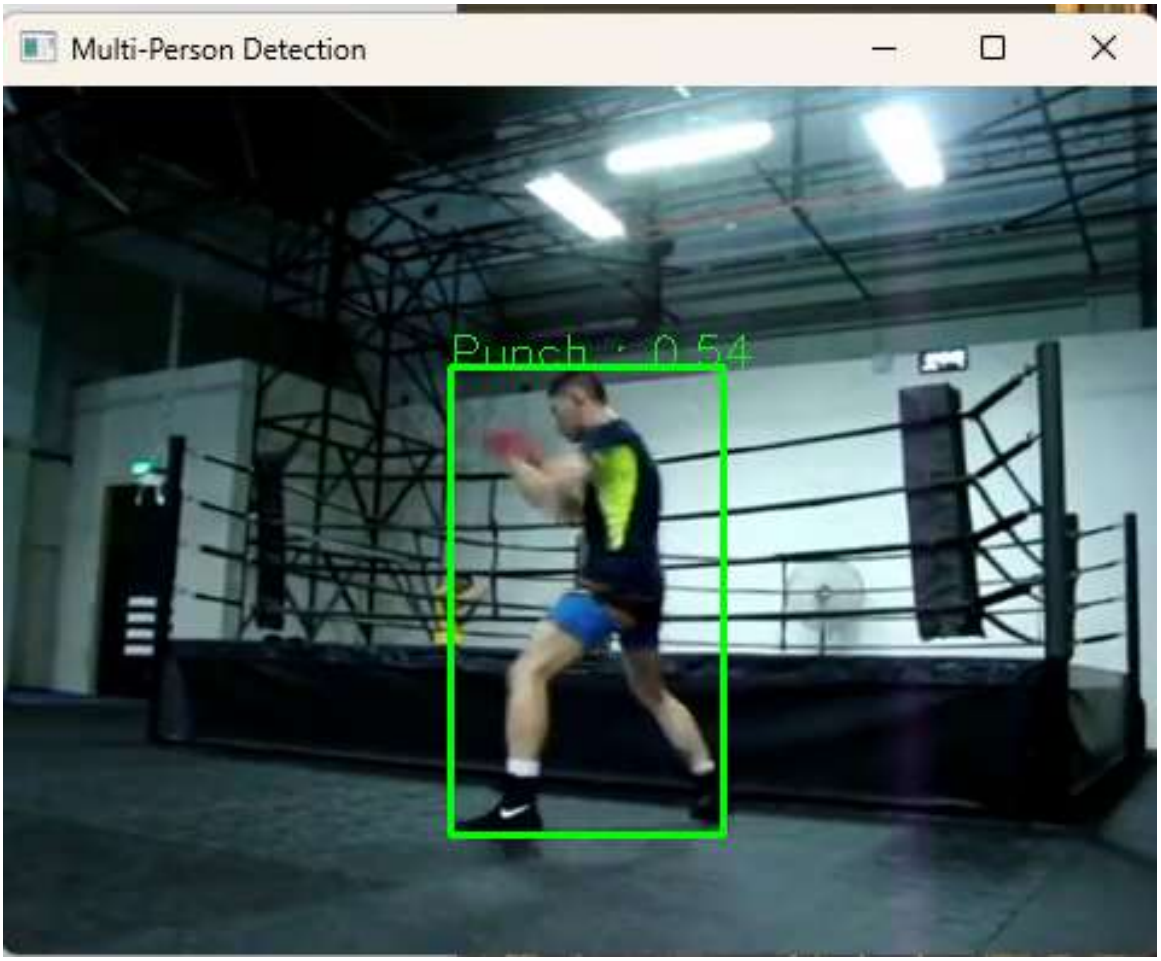


Fig. 4.14.2 Recunoașterea activităților cu ajutorul aplicației

5. Rezultate experimentale

În această secțiune sunt prezentate rezultatele obținute în urma antrenării și evaluării modelelor propuse pentru clasificarea acțiunilor umane pe baza secvențelor video. După cum am precizat anterior, am folosit un procent de 15% de videoclipuri din setul de date UCF50.

Trebuie subliniat faptul că performanța modelelor de învățare automată poate fi semnificativ influențată de modul în care este realizată împărțirea setului de date în subseturi de antrenare, validare și testare. În cazul setului de date UCF50, videoclipurile sunt organizate în grupuri, însă această organizare nu este una explicită, în sensul că fișierele aparținând aceluiași grup nu sunt plasate într-un director comun.

Conform sursei [19] videoclipurile care aparțin aceluiași grup provin, de fapt, dintr-un singur videoclip de dimensiuni mari, care a fost segmentat în mai multe secvențe video mai scurte. Acest lucru înseamnă că acele videoclipuri sunt extrem de similare din punct de vedere vizual și contextual, având același decor, aceleași mișcări și aceiași actori. Ca urmare, dacă aceste secvențe sunt distribuite arbitrar între subseturile de antrenare, validare și testare, modelele pot obține rezultate artificial ridicate. Aceste performanțe nu reflectă cu acuratețe capacitatea modelului de a generaliza la date complet noi, deoarece evaluarea se face pe mostre foarte asemănătoare cu cele folosite la antrenare.

Pentru a preveni acest fenomen între subseturi este esențial ca videoclipurile din același grup să nu fie împărțite între seturile de date. O soluție practică este evitarea amestecării aleatorii a videoclipurilor înainte de împărțire. Astfel procedura aplicată în acest proiect a constat în extragerea listei tuturor videoclipurilor din setul de date original fără permutarea ordinii acestora urmată de o împărțire secvențială:

- primele 60% dintre videoclipuri au fost utilizate pentru antrenare,
- următoarele 25% au fost alocate validării,
- iar ultimele 15% au fost rezervate pentru testare.

Această abordare reduce semnificativ riscul ca videoclipurile aparținând aceluiași grup să fie împărțite între subseturi diferite. În cel mai rău caz, doar două grupuri pot fi afectate de o astfel de împărțire: unul aflat la granița dintre subsetul de antrenare și cel de validare, și altul aflat între subsetul de validare și cel de

testare. Impactul acestora asupra generalizării este considerabil mai mic decât în cazul unei împărțiri complet aleatorii.

Prin urmare, această metodă de împărțire contribuie la o evaluare mai corectă și realistă a performanței modelelor testate, reducând influența datelor redundante sau similare în procesul de validare și testare.

O altă componentă importantă care trebuie luată în considerare în măsurarea performanței modelelor este acuratețea detecțiilor realizate de MediaPipe. Acest aspect este relevant doar pentru modelele care se bazează pe informații extrase din pose landmark-uri. După cum a fost menționat anterior, din fiecare videoclip sunt extrase câte 15 seturi de landmark-uri, corespunzătoare unor cadre selectate în mod uniform pe durata videoclipului. Totuși, MediaPipe nu garantează detecția completă și corectă a punctelor de interes în toate cadrele. Această limitare apare frecvent în cazul imaginilor în care persoana este parțial ieșită din cadru, este într-o poziție neobișnuită sau fundalul este complex ori întunecat. Astfel, numarul seturilor de pose landmark-uri poate sa fie mai mic decat cel dorit. Pentru a rezolva cazul în care există un număr insuficient de detecții, dar totuși mai mare ca zero, a fost implementată o schemă de padding. Astfel, pentru fiecare listă de seturi de puncte detectate, dacă aceasta conține mai puțin de 15 elemente, se adaugă în continuare elemente din aceeași listă, dar în ordine inversă. Dacă toate elementele au fost deja duplicate și dimensiunea dorită nu a fost atinsă, procesul continuă prin alternarea ordinii pentru a completa lista. În plus se respectă condiția ca să nu existe seturi identice de puncte pe poziții succesive.

Pentru a exemplifica, vom folosi o listă de numere care inițial este:

$$list = [1,2,3]$$

Considerând că dorim ca lista să aibă 15 elemente, după aplicarea procesului de padding, aceasta va arăta astfel:

$$list = [1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3]$$

Am folosit această abordare de padding deoarece prin adăugarea în ordine inversă a elementelor din lista de seturi de puncte se păstrează într-o anumită măsură dinamica temporală a mișcării. Practic în loc să completăm secvența cu valori arbitrare sau cu repetări simple ale ultimului cadru, am ales să alternăm cadrele deja existente, dar în ordine inversă, ceea ce oferă o continuitate mai naturală și mai realistă a secvenței.

Ca analogie, este ca și cum unui videoclip i-am adăuga, la final, o redare a propriei sale secvențe în ordine inversa. Dacă am vizualiza un astfel de videoclip, am observa că după ce persoana își termină mișcarea aceasta pare să revină gradual în poziția inițială într-un mod coerent din punct de vedere vizual. Acest comportament păstrează un anumit ritm al mișcării, ceea ce este important pentru rețelele neuronale care trebuie să învețe nu doar poziții statice, ci și tranziții între posturi.

În plus, pentru a evita introducerea de redundanță evidentă sau secvențe nerealiste, algoritmul de padding are grijă ca două seturi identice de puncte să nu apară consecutiv, asigurând astfel o varietate minimă în cadrul secvenței extinse. Această tehnică contribuie la păstrarea coerenței datelor de intrare și ajută modelul să învețe mai eficient tiparele temporale relevante pentru clasificarea acțiunilor.

Pentru a măsura diferențele de performanță, toate modelele care folosesc detecțiile făcute de MediaPipe au fost antrenate atât pe date asupra cărora s-a aplicat padding, cât și pe date unde detecțiile insuficiente au fost eliminate.

5.1. Rețea neuronală recurentă (RNN)

Acest model a fost antrenat exclusiv pe detecții realizate de MediaPipe.

Rezultate pe date asupra cărora NU s-a aplicat padding:

- Acurațețe: 0.5733
- Precizie: 0.6283
- Recall: 0.6064
- Scor F1: 0.6003

Rezultate pe date asupra cărora s-a aplicat padding:

- Acurațețe: 0.4298
- Precizie: 0.4592
- Recall: 0.4594
- Scor F1: 0.4462

5.2. Rețea neuronală convoluțională 1D (1D CNN)

Acest model folosește detecțiile făcute de MediaPipe. Este important de menționat că acest model a fost antrenat de la 0, nu include nicio componentă preantrenată și are o performanță notabiă.

Rezultate pe date asupra cărora NU s-a aplicat padding:

- Acurațețe: 0.6906
- Precizie: 0.7681
- Recall: 0.7420
- Scor F1: 0.7436

Rezultate pe date asupra cărora s-a aplicat padding:

- Acurațețe: 0.4331
- Precizie: 0.5227
- Recall: 0.4816
- Scor F1: 0.4807

5.3. Rețea neuronală convoluțională 3D urmată de o rețea RNN

Acesta este singurul model care foloseste ca și date doar cadre video.

Rezultate:

- Acurațețe: 0.8320
- Precizie: 0.8720
- Recall: 0.8636
- Scor F1: 0.8618

5.4 Model Multi-Input:CNN + RNN

Rezultate pe date asupra cărora NU s-a aplicat padding:

- Acurațețe: 0.8819
- Precizie: 0.9106
- Recall: 0.9021
- Scor F1: 0.9019

Rezultate pe date asupra cărora s-a aplicat padding:

- Acurațețe: 0.8685
- Precizie: 0.8894
- Recall: 0.8788
- Scor F1: 0.8792

5.5. Model Multi-Input:CNN + 1D CNN

Rezultate pe date asupra cărora NU s-a aplicat padding:

- Acurațețe: 0.8056
- Precizie: 0.8730
- Recall: 0.8644
- Scor F1: 0.8606

Rezultate pe date asupra cărora s-a aplicat padding:

- Acurațețe: 0.7704
- Precizie: 0.8160
- Recall: 0.7956
- Scor F1: 0.7974

Din performanțele prezentate anterior se poate concluziona că un factor esențial în eficiența modelului este calitatea și prelucrarea inițială a datelor. În special, se observă că anumite arhitecturi oferă rezultate semnificativ mai bune atunci când sunt antrenate și testate pe date brute, fără nicio formă de padding.

Padding-ul, deși util pentru uniformizarea dimensiunilor între secvențe sau cadre, poate introduce zgomot sau informații artificiale care afectează negativ procesul de învățare al modelului. Aceasta se reflectă în scăderea acurateței observate pe seturile de date cu padding, chiar dacă diferența nu este întotdeauna foarte mare.

6. Concluzii

În acest proiect au fost explorate și prezentate diverse abordări prin care modelele de învățare profundă pot fi utilizate pentru a recunoaște și clasifica acțiunile realizate de persoane în cadrul secvențelor video. Deși la prima vedere s-ar putea presupune că informația vizuală conținută într-un videoclip este suficientă pentru această sarcină în realitate performanța modelelor depinde în mare măsură de cât de bine sunt reprezentate și corelate informațiile extrase din aceste date.

Astfel s-a evidențiat importanța folosirii unor surse complementare de informații precum coordonatele punctelor de pe corp care descriu postura unei persoane (pose landmark-uri), viteza de mișcare, orientarea în spațiu sau alte atribute temporale. Aceste date, obținute prin instrumente precum MediaPipe, permit construirea unui context mai complet și mai coerent în jurul fiecărei acțiuni, ajutând modelul să înțeleagă nu doar ce se vede într-un cadru, ci și cum evoluează mișcarea în timp.

Prin construirea și testarea mai multor arhitecturi de rețele neuronale, proiectul a urmărit identificarea unor corelații între caracteristicile vizuale statice (extrase din imagini sau cadre individuale) și caracteristicile temporale (care reflectă dinamica acțiunii pe parcursul videoclipului). În locul unei abordări simple, bazate pe imagini individuale, s-au folosit secvențe de cadre sau date derivate din acestea, precum cele oferite de MediaPipe, cu scopul de a obține o reprezentare mai fidelă a videoclipului și, implicit, o clasificare mai precisă a acțiunilor.

Această combinație între caracteristici spațiale (ce se întâmplă într-un cadru) și temporale (cum se desfășoară acțiunea în timp) poate duce la dezvoltarea unor sisteme mult mai performante de recunoaștere a acțiunilor umane în videoclipuri.

7. Directii viitoare de cercetare

7.1. Padding

Pentru datele prezentate anterior în cazul în care nu se realizează niciun tip de padding la nivelul detecțiilor efectuate de MediaPipe se pierd aproximativ 45% din setul total de date disponibil. Această pierdere semnificativă afectează în mod direct performanța modelelor de învățare profundă, deoarece acestea dispun de mai puține exemple din care pot învăța. Prin urmare, este esențială identificarea unui algoritm de padding mai eficient care să permită utilizarea unui număr cât mai mare de secvențe video fără a compromite calitatea sau relevanța temporală a datelor introduse în rețea.

Algoritmul de padding prezentat anterior presupune completarea secvențelor scurte prin repetarea inversată a seturilor de puncte deja existente. Cu toate acestea, în cazul în care o secvență conține un număr foarte redus de seturi de landmark-uri, acest algoritm introduce o redundanță extrem de mare, afectând negativ capacitatea modelului de a învăța tipare semnificative.

De exemplu, să presupunem că avem următorul caz simplificat, transpus ca o listă de numere întregi:

$$list = [1]$$

După aplicarea algoritmului de padding (pentru a atinge o lungime de 15 elemente), lista va deveni:

$$list = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$$

Acesta este cel mai nefavorabil caz posibil. Dacă transpunem acest exemplu în contextul unei liste de seturi de landmark-uri detectate de MediaPipe, rezultatul este o secvență complet redundantă lipsită de variație temporală. Practic, nu se mai transmite nicio informație despre mișcarea efectuată ceea ce face imposibilă învățarea unei acțiuni de către rețea. Astfel de secvențe nu contribuie în niciun fel la antrenarea modelului și mai mult decât atât, pot introduce zgomot sau informații eronate în procesul de învățare.

Pe de altă parte dacă avem o secvență aproape completă:

$$list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]$$

După aplicarea padding-ului, aceasta devine:

$$list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,13]$$

În acest caz, redundanța este minimă, iar structura temporală este în mare parte păstrată. Modelul poate interpreta secvența ca fiind coerentă din punct de vedere al mișcării, ceea ce contribuie pozitiv la procesul de învățare.

Ținând cont de cele două extreme, este evident că trebuie identificat un prag optim, adica un număr minim de seturi de landmark-uri care trebuie să fie disponibile într-o secvență astfel încât după aplicarea padding-ului calitatea datelor să rămână ridicată iar performanța modelelor să nu fie afectată semnificativ.

Acest prag ar putea fi stabilit empiric, în urma unui proces de validare pe un subset de date analizând impactul padding-ului asupra acurateței și capacității de generalizare a modelului. O altă direcție viabilă de cercetare ar putea fi dezvoltarea unui padding adaptiv care în loc să repete datele să folosească o formă de interpolare temporală sau chiar generare de secvențe intermediare prin rețele specializate, cum ar fi autoencodere sau GAN-uri.

7.2. Augmentarea detecțiilor MediaPipe

Una dintre limitările majore în procesul de antrenare a modelelor de recunoaștere a acțiunilor este dimensiunea și diversitatea setului de date. Chiar dacă MediaPipe oferă o soluție eficientă pentru extragerea automată a punctelor de reper care descriu postura unei persoane, detecțiile obținute sunt strict corelate cu orientarea și dinamica specifică a persoanei în videoclip. Pentru a îmbunătăți generalizarea modelului și a-l face mai robust în fața variațiilor din lumea reală, augmentarea acestor detecții poate deveni o etapă esențială.

Să considerăm exemplul unui videoclip în care o persoană aleargă spre cameră. După aplicarea MediaPipe și normalizarea punctelor, vom obține o secvență de 15 seturi de landmark-uri care descriu mișcarea persoanei. Dacă aceste puncte sunt proiectate într-un spațiu tridimensional (3D), se va forma o reprezentare scheletală care se deplasează de-a lungul axei Z, în direcția camerei.

Presupunând că avem un al doilea videoclip, în care o persoană aleargă în direcția opusă, îndepărtându-se de cameră, apare întrebarea daca modelele de învățare automată vor interpreta aceste două secvențe ca aparținând aceleiași acțiuni (alergare), sau le vor încadra în clase diferite datorită direcției și poziției în spațiu.

În lipsa unei diversități suficiente în setul de antrenament, există riscul ca modelul să devină sensibil la astfel de variații și să nu generalizeze corect. Acesta este motivul pentru care augmentarea detecțiilor MediaPipe este o soluție eficientă pentru creșterea robusteții modelului.

Augmentarea propusă presupune aplicarea unor transformări geometrice asupra coordonatelor punctelor detectate. În particular, ne concentrăm pe rotirea tridimensională a seturilor de puncte în jurul unei axe verticale, considerată perpendiculară pe planul solului. Această rotație are ca efect schimbarea direcției de deplasare percepută în reprezentarea tridimensională. De exemplu, după o rotire cu 30°, aceeași secvență de mișcări care descria alergarea direct spre cameră poate fi percepută ca o alergare în diagonală. Repetând acest proces cu pași constanți (ex. 15°, 30°, 45°, etc.), se pot genera multiple versiuni ale aceleiași acțiuni, cu variații de unghi și orientare.

7.3. Vision Transformer

De-a lungul istoriei recente, rețelele convoluționale 3D au fost cele mai dominante modele folosite pentru a procesa imagini și secvențe video în scopul diferitelor sarcini.

Deși aceste modele s-au dovedit destul de eficiente în recunoașterea corectă a acțiunilor din videoclipuri, ele prezentau totuși anumite limitări. Cea mai mare problemă a arhitecturii 3DCNN, mai ales în contextul recunoașterii acțiunilor, era incapacitatea acesteia de a învăța cu adevărat dependențele pe termen lung.

Pentru a înțelege mai bine ce înseamnă o dependență pe termen lung, ne putem gândi la acțiuni de durată dintr-un videoclip, precum mersul unei persoane sau faptul că aceasta ține un obiect în mână pentru mai mult timp. Modelele 3DCNN nu reușesc să surprindă eficient astfel de tipare, deoarece filtrele lor, care determină câte cadre pot fi analizate simultan, trebuie menținute la o adâncime redusă pentru a funcționa corespunzător.

În ultimii ani, arhitectura Transformer a devenit una dintre cele mai utilizate structuri în domeniul rețelelor neuronale, cu aplicații de succes nu doar în procesarea limbajului natural, ci și în viziunea computerizată. Un exemplu important în acest sens este Vision Transformer (ViT), care adaptează mecanismul de atenție al transformerelor la sarcini precum clasificarea imaginilor sau recunoașterea acțiunilor în secvențe video.

Unul dintre marile avantaje ale ViTs este mecanismul de atenție, care permite modelului să se concentreze asupra diferitelor părți ale unei imagini sau ale unei secvențe video. În cazul videoclipurilor, acest lucru se traduce prin capacitatea de a analiza relații temporale complexe între cadre, fără a fi limitat de o fereastră fixă de timp, așa cum se întâmplă în cazul arhitecturilor 3DCNN. Practic, ViT poate lua în considerare cadre aflate la distanțe mari în timp pentru a înțelege contextul complet al unei acțiuni, un aspect esențial pentru recunoașterea acțiunilor de durată, cum ar fi mersul, alergatul sau gesturi mai subtile.

8. Bibliografie

[1] Nweke, H. F., Teh, Y. W., Al-garadi, M. A., & Alo, U. R. (2018). Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. Expert Systems with Applications, 105, 233–261. https://doi.org/10.1016/j.eswa.2018.03.056

[2] Mohamed Zaidi, M., Avelino Sampedro, G., Almadhor, A., Alsubai, S., Al Hejaili, A., Gregus, M., & Abbas, S. (2024). Suspicious human activity recognition from surveillance videos using deep learning. *IEEE Access: Practical Innovations, Open Solutions*, *12*, 105497–105510. https://doi.org/10.1109/access.2024.3436653

[3] Sarabu, A., & Santra, A. K. (2021). Human action recognition in videos using Convolution Long Short-Term Memory network with Spatio-temporal networks. Emerging Science Journal, 5(1), 25–33. https://doi.org/10.28991/esj-2021-01254

[4] Simonyan, Karen, and Andrew, Zisserman. "Two-Stream Convolutional Networks for Action Recognition in Videos.". In Proceedings of the 27th International Conference on Neural Information Processing Systems -Volume 1 (pp. 568–576). MIT Press, 2014. doi:10.5555/2968826.2968890

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778.

[6] Brox T., Bruhn A., Papenberg N., Weickert J. (2004) High Accuracy Optical Flow Estimation Based on a Theory for Warping. In: Pajdla T., Matas J. (eds) Computer Vision -ECCV 2004. ECCV 2004. Lecture Notes in Computer Science, vol 3024. Springer, Berlin, Heidelberg. doi:10.1007/978-3-540-24673-2_3

[7] Zach, C., T. Pock, and H. Bischof. “A Duality Based Approach for Realtime TV-L 1 Optical Flow.” Pattern Recognition (n.d.): 214–223. doi:10.1007/978-3-540-74936-3_22

[8] Ma, C.-Y., Chen, M.-H., Kira, Z., & AlRegib, G. (2019). TS-LSTM and temporal-inception: Exploiting spatiotemporal dynamics for activity recognition. Signal Processing. Image Communication, 71, 76–87. https://doi.org/10.1016/j.image.2018.09.003

[9] Diogo, C. L., David, P., & Hedi, T. (2018). 2D/3D pose estimation and action recognition using multitask deep learning. In arXiv [cs.CV]. http://arxiv.org/abs/1802.09232

[10] Luvizon, D. C., Tabia, H., & Picard, D. (2017). Human Pose regression by combining indirect part detection and contextual information. In arXiv [cs.CV]. http://arxiv.org/abs/1710.02322

[11] What is machine learning (ML)? (2025, June 4). Ibm.com. https://www.ibm.com/think/topics/machine-learning

[12] What is a neural network? (2025, June 4). Ibm.com. https://www.ibm.com/think/topics/neural-networks

[13] aakarshachug Follow Improve. (2019, January 16). What is LSTM - long short term memory? GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

[14] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. In arXiv [cs.CV]. http://arxiv.org/abs/1512.03385

[15] Petru, P. (2024, March 13). What is ResNet-50? Roboflow Blog. https://blog.roboflow.com/what-is-resnet-50/

[16] TimeDistributed layer. (n.d.). Keras.Io. https://keras.io/api/layers/recurrent_layers/time_distributed/

[17] MediaPipe solutions guide. (n.d.). Google AI for Developers. https://ai.google.dev/edge/mediapipe/solutions/guide

[18] Ultralytics. (2023, November 12). Home. Ultralytics.com. https://docs.ultralytics.com/

[19] UCF Center for Research. (n.d.). CRCV. Ucf.edu. https://www.crcv.ucf.edu/data/UCF50.php

[20] Pose landmark detection guide. (n.d.). Google AI for Developers. https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker

[21] Timeseries classification from scratch. (n.d.). Keras.Io. https://keras.io/examples/timeseries/timeseries_classification_from_scratch/

[22] Ultralytics. (2023b, November 12). Multi-object tracking with Ultralytics YOLO. Ultralytics.com. https://docs.ultralytics.com/modes/track/