### UNIVERSITATEA DIN BUCUREȘTI

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

# LUCRARE DE LICENȚĂ

Absolvent

Hirica Ioan Alexandru

**Coordonator științific**

**Prof. Dr. Radu Ionescu**

**București, iunie 2023**

### UNIVERSITATEA DIN BUCUREȘTI

**FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ**

**SPECIALIZAREA INFORMATICĂ**

# Rețele neuronale pentru estimare de statistici din videoclipuri cu meciuri din tenis

Absolvent

Hirica Ioan Alexandru

**Coordonator științific**

**Prof. Dr. Radu Ionescu**

**București, iunie 2023**

**Rezumat**

Statisticile mișcărilor dintr-un meci de tenis pot oferi un mare avantaj în pregătirea unui jucător. O cunoaștere detaliata a unui oponent poate fi benefică în antrenarea unui jucător, oferind un avantaj in înțelegerea punctelor forte și a celor slabe ale unui adversar, ceea ce aduce un plus pentru desfășurarea antrenamentelor.

In cadrul acestei lucrări am ales să creez un model care poate prezice mișcarea efectuată de un jucător de tenis dintr-o filmare. Modelul pentru predicția mișcărilor unui jucător de tenis este făcută folosind coordonatele articulațiilor oferite de către un model de YOLO pre antrenat, acestea fiind oferite unei rețele LSTM. Acest model poate distinge 12 mișcări distincte pe care un jucător de tenis le poate efectua.

Pe setul de date THETIS care conține videoclipuri cu jucători care efectuează diferite mișcări, aceasta abordare oferă o acuratețe de 93%.

Acest model a fost integrat într-o aplicație având o interfața grafica, făcând utilizarea de către un utilizator mult mai ușoara.

**Abstract**

The statistics of the movements in a tennis match can offer a great advantage in the preparation of a player. A detailes knowledge of an opponent can be beneficial in the preparation of a player, offering an advantage in understanding the strengths and weakness of opponent, which brings a plus for training.

In this study, I have chosen to create a model that can predict the movement performed by a tennis player from a video. The model for predicting the movements of a tennis player, is made using the coordinates of the joints provided by a pre-trained YOLO model, these being provided to an LSTM network.

On the THETIS dataset containing videos of players performing various moves, this approach provides an accuracy of 93%.

This model was integrated into an app that has a user interface, so that it is easier for an user to operate.

Cuprins

[1. Introducere 6](#_Toc137600654)

[1.1 Motivație 6](#_Toc137600655)

[1.2 Structura lucrării 7](#_Toc137600656)

[1.2.1: Detecția jucătorului 7](#_Toc137600657)

[1.2.2: Predicția acțiunii 7](#_Toc137600658)

[2. Concepte teoretice 8](#_Toc137600659)

[2.1: Neuronul 8](#_Toc137600660)

[2.2: Rețele feed-forward 8](#_Toc137600661)

[2.3: Antrenarea unei rețele feed-forward 9](#_Toc137600662)

[2.4: LSTM 10](#_Toc137600663)

[2.5 YOLO 11](#_Toc137600664)

[2.6 YOLO pose 11](#_Toc137600665)

[3. Abordări recente 12](#_Toc137600666)

[4. Tehnologii folosite 14](#_Toc137600667)

[4.1 Python 14](#_Toc137600668)

[4.2PyTorch 14](#_Toc137600669)

[4.3 Matplotlib 14](#_Toc137600670)

[4.4 OpenCV 14](#_Toc137600671)

[4.5 YOLOv7 15](#_Toc137600672)

[5. Abordarea propusa 16](#_Toc137600673)

[5.1 Descrierea setului de date 16](#_Toc137600674)

[5.2 Configurarea modelului YOLO 17](#_Toc137600675)

[5.3 Transformarea setului de date 18](#_Toc137600676)

[5.4 Normalizarea datelor 19](#_Toc137600677)

[5.5 Încărcarea datelor și crearea secvențelor 20](#_Toc137600678)

[5.6 Crearea modelului 22](#_Toc137600679)

[5.7 Antrenarea modelului 24](#_Toc137600680)

[6. Experimente și rezultate 27](#_Toc137600681)

[6.1. Prezentarea metricilor de acuratețe și primele rezultate 27](#_Toc137600682)

[6.2 Interpretarea rezultatelor 31](#_Toc137600683)

[7. Descrierea aplicației 33](#_Toc137600684)

[7.1 Front-end 33](#_Toc137600685)

[7.2 Back-end 35](#_Toc137600686)

[7.2.1 Baza de date 35](#_Toc137600687)

[7.2.2 Semnătura digitala a unui fișier video 37](#_Toc137600688)

[7.2.3 Generarea videoclipului ce conține loviturile 38](#_Toc137600689)

[7.2.4 Generarea fișierelor de statistici 40](#_Toc137600690)

[8. Concluzii și dezvoltări ulterioare 42](#_Toc137600691)

[9. Bibliografie 44](#_Toc137600692)

# Introducere

### 1.1 Motivație

Analiza video a sporturilor este o soluție pentru a extrage diferite informații care pot fi folosite atât de profesioniști cât și de telespectatori pentru a avea o privire de ansamblu analitică a ceea ce se întâmpla pe parcursul jocului.

O astfel de tehnologie este sistemul „Hawk-Eye Live” care poate semnala dacă mingea a fost data in afara suprafeței de joc. Însă instalarea acestui sistem este dificilă și costisitoare. Conform [Espn, 2023] costul pentru instalarea acestui sistem pe un teren este între $60.000 și $70.000, făcând folosirea acesteia imposibilă pentru antrenamente sau pentru turneele mici.

Scopul acestei lucrări este de a oferi o soluție accesibila din punct de vedere computațional, fiind ușor de rulat pe un GPU.

Lucrarea propune o modalitate de a vedea ce mișcări precise efectuează un jucător de tenis în cadrul unei filmări, pentru a putea fi analizate. Prin intermediul acestei abordări se poate analiza atent jocul jucătorului, antrenorii putând îmbunătăți strategiile, tehnica de joc, precum și pregătirea fizica înaintea meciurilor.

Abordarea aleasă vautiliza limbajul Python pentru implementare și tehnici inovatoare pentru a detecta jucătorul și coordonatele pozițiilor membrelor. Astfel, pentru detecția jucătorului vom folosi YOLOv7, care conform [Wang et al., 2022] întrece toți detectorii in privința vitezei și a acurateței, oferind o acuratețe de 56.8% pe setul de date MS COCO. Pentru obținerea coordonatelor membrelor vom folosi un detector de pose estimation antrenat pe YOLOv7 având la baza tehnica prezentata in [Maji et al., 2022] care oferă o acuratețe buna, de 68.5 pe setul de date COCO 2017 și are o performanta computațională foarte buna.

### 1.2 Structura lucrării

* **Detecția jucătorului**: selectarea jucătorului care are dimensiunile cele mai mari din fiecare cadru, extragerea coordonatelor membrelor și normalizarea datelor pentru a putea fi prelucrate.
* **Predicția acțiunii**: antrenarea, proiectarea și inferența într-un videoclip, bazându-se pe coordonatele membrelor pentru a oferi predicția mișcării efectuate, pe un segment de video, având numărul de date prestabilit ca hiperparametru.

### 1.2.1: Detecția jucătorului

Pentru a detecta jucătorul vom folosi modelul YOLOv7 pre antrenat pentru detecția pozițiilor membrelor, pentru a detecta toate persoanele dintr-o imagine. După aceasta detecție vom selecta numai persoana care are dimensiunea cea mai mare din cadru (daca ar fi feed-ul unei camere din transmisia TV, acest model va selecta numai jucătorul din partea de jos). Pentru fiecare membru din „cutia” dominanta ca și dimensiuni, coordonatele acestuia se vor normaliza in intervalul [0, 1]. Aceasta operațiune se va repeta pentru fiecare cadru dintr-o filmare.

### 1.2.2: Predicția acțiunii

Vom lua datele de la punctul anterior, ne vom folosi de un set de date adnotat cu videoclipuri cu lovituri din tenis pentru a antrena un model care să poată da o predicție asupra mișcării efectuate de către un jucător, oferind ca ieșire una din 12 mișcări elementare jocului de tenis.

# Concepte teoretice

### 2.1: Neuronul

Neuronul este o unitate neuronala simplă care poate rezolva probleme de clasificare binară. Peceptronul Rosenblatt, descris in [Rosenblatt et al., 1958], primește un set de caracteristici de intrare și produce o ieșire binară pe baza unui set de ponderi și a unei funcții de activare. Fiecare caracteristica de intrare are o pondere asociată, care este utilizată pentru a calcula suma ponderata a intrărilor. Ponderile indică contribuția fiecărei caracteristici. Suma ponderată este apoi trecuta printr-o funcție de activare care are rolul de a introduce non-liniaritate in procesul de decizie.

Ecuația de calcul a unui perceptron este: , unde reprezintă valoarea fiecărei intrări, ponderea fiecărei intrări, iar funcția este functia de activare. Cateva exemple de functii de activare sunt: Sigmoid(x), ReLU(x), tanh(x).

### 2.2: Rețele feed-forward

Rețelele neuronale de tipul feed-forward sunt formate din mai multe straturi de neuroni a căror ieșire este conectata la alte straturi de neuroni. Aceasta rețea este alcătuita din 3 componente: stratul de intrare, straturile ascunse și stratul de ieșire. Stratul de intrare este primul strat care primește datele de intrare. Straturile ascunse sunt situate intre stratul de intrare și cel de ieșire, fiecare strat conținând un număr de neuroni care poate să fie distinct. Acestea au rolul de a extrage și transforma caracteristicile de intrare, învățând reprezentări complexe. Stratul de ieșire este ultimul strat și produce ieșirea rețelei. Numărul de neuroni de pe acest strat pentru o problema de clasificare poate reprezintă numărul de clase pentru care dorim să facem clasificarea. Comparativ cu o rețea care este alcătuita dintr-un singur neuron, rețelele de tip feed-forward sunt capabile să învețe și să modeleze relații complexe. Cu cât adăugam mai multe straturi ascunse acestea devin mai puternice, putând să construiască reprezentări ale datelor mai sofisticate.

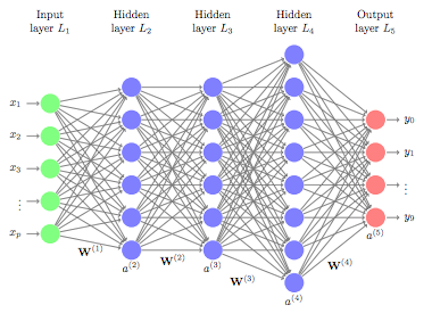


Figura 2.1: Imagine descriptivă a unei rețele feed-forward [University of Cincinnati, 2023]

In figura 2.1 este evidențiata o arhitectura de tipul de feed-forward având 3 straturi ascunse, care este folosită pentru a rezolva o problema de clasificare cu 10 clase (numărul de neuroni din stratul de ieșire este de 10). Aceasta rețea primește ca intrare un număr de „p” caracteristici.

### 2.3: Antrenarea unei rețele feed-forward

La început ponderile unei rețele feed-forward sunt inițializate aleatoriu sau folosind o metoda specifica cum ar fi inițializarea Xavier descrisă in [Glorot and Xavier, 2010]. In pasul următor datele sunt trecute prin rețea strat cu strat într-un mod secvențial, pana când se obține ieșirea finala. Este apoi calculată eroarea rețelei ce reprezintă diferența dintre predicția rețelei și eticheta reala. Pasul care urmează este cel de propagare înapoi in care eroarea este propagata înapoi in rețea pentru a calcula gradientul funcției de pierdere in raport cu ponderile. Procesul începe de la ultimul strat și continua către straturile anterioare. Actualizarea ponderilor se face folosind gradientul funcției de pierdere pentru fiecare pondere, folosind un algoritm de optimizare. Un astfel de algoritm este algoritmul coborârii pe gradient, in care ponderile sunt actualizate scăzând din acestea gradientul înmulțit cu un termen, denumit rata de învățare. Rata de învățare este un termen care este în intervalul [0, 1] și determină cat de mult să actualizăm ponderea.

### 2.4: LSTM

Rețelele de tip LSTM (Long short term memory) vin ca o evoluție a rețelelor RNN (Recurrent neural networks), care conform [Staudemeyer et al., 2019.] nu pot depăși mai mult de 5-10 pași de timp întrucât gradienții se pot micșora foarte tare (dispar) sau pot avea valori foarte mari (explodează). In cazul in care gradienții explodează, valoarea „w” oscilează, iar in cazul in care gradienții dispar, antrenarea durează foarte mult timp sau chiar nu funcționează.

Soluția acestei probleme este folosirea LSTM, care poate avea conform [Staudemeyer et al., 2019.] și mai mult de 1000 de pași de timp. Rețelele LSTM excelează pe sarcinile care necesită ca un set de date secvențial să fie ținut minte pentru o perioada lunga de timp. Aceasta proprietate este atribuita blocurilor de memorie, care pot accesa intrarea și produc o ieșire, prevenind astfel ca informațiile irelevante să intre sau să iasă din acest bloc. De asemenea, aceste blocuri mai au și o poarta de „uitare” astfel încât, daca o caracteristica devine irelevantă, aceasta nu mai este luata in calcul [Staudemeyer et al., 2019.].

Aceste rețele se pot suprapune pentru a forma o rețea deep LSTM, astfel ieșirea unei astfel de rețele, reprezintă intrarea următoarei. Aceasta abordare a adus o îmbunătățire dramatica față de o rețea cu un singur strat, in domeniul transcrierii mesajelor audio in text [Graves et al., 2013].

Pentru a accelera antrenarea putem folosi „Gradient clipping”, care menține gradienții într-un interval stabilit ca și hiperparametru. De exemplu, daca am setat valoarea de 0.25 ca hiperparametru și gradientul nostru are valoarea de 0.7, acesta va fi setat la valoarea de 0.25. Conform [Zhang et al., 2019] folosind aceasta tehnica, au testat pe o sarcina de prelucrarea limbajului natural, folosind modelul AWD-LSTM, și au observat o creștere a vitezei cu care modelul este antrenat.

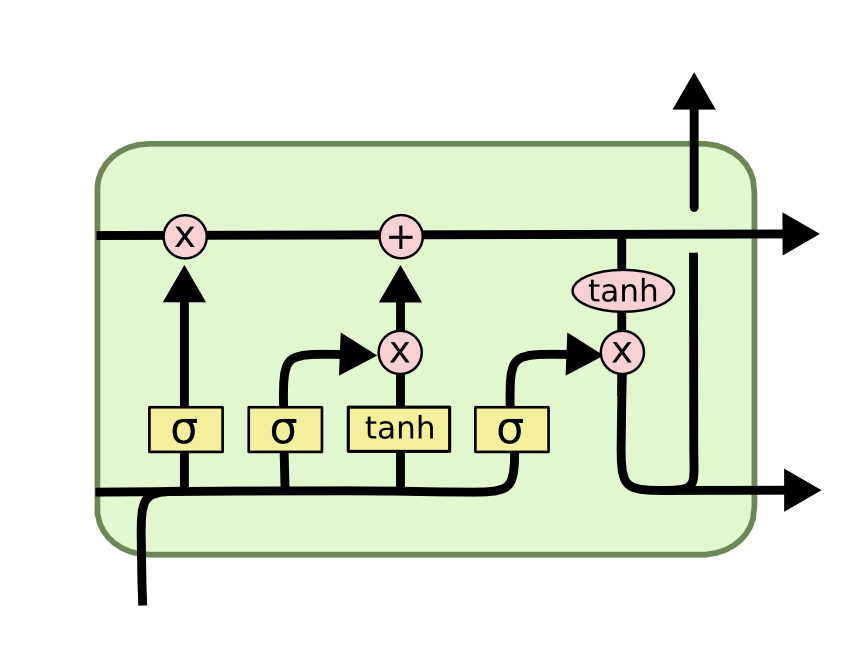


Figura 2.2: Imagine descriptivă a unui bloc LSTM [colah.github.io, 2023]

### 2.5 YOLO

YOLO (You only look once) este o rețea capabila să detecteze locațiile obiectelor in timp real din imagini. In aceasta lucrare vom folosi YOLOv7 care îmbunătățește acuratețea și viteza. Pentru antreneare acesta folosește setul de date MS COCO, antrenarea făcându-se de la 0, nefolosindu-se alte seturi de date sau rețele preantrenate. Comparativ cu alte metode, de exemplu YOLOv4, acest model are cu 75% mai puțini parametrii și aduce o îmbunătățire a preciziei medii de 1.5%. Comparativ cu modelul state-of-the-art YOLOR-CSP, autorii ne prezinta ca modelul are cu YOLOv7 are cu 43% mai putini parametrii și o precizie medie cu 0.4% mai mare [Wang et al., 2022].

### 2.6 YOLO pose

Sarcina acestui model este de a detecta coordonatele articulațiilor tuturor persoanelor dintr-o imagine. Pentru antrenarea acestui model, autorii au folosit setul de date COCO Keypoint Detection, unde au augumentat datele, folosind redimensionare, translatare, inversare, augumentari mozaic și diferite augumentari ale culorilor. Pe setul de date COCO val2017 precizia medie este de 68.5% iar cel mai bun model este HigherHRNet cu 70.5%. Conform autorilor la rezoluții mai mici, YOLOv5s6-pose are o performanta mult mai buna decât alte modele de complexitate mica, de exemplu EfficientHRNet, pe setul de date COCO val2017 [Maji et al., 2022].

# Abordări recente

O abordare propusă in estimarea acțiunii este folosindu-ne de caracteristici calculate de mana, precum in lucrarea [Wang et al., 2013], (care se folosește de caracteristici calculate de mana pentru a estima mișcarea). Pentru aceasta, autorii calculează homografia folosind algoritmul RANSAC pentru a stabiliza mișcarea camerei, intre 2 cadre. După aceea se folosesc de un detector uman pentru a detecta oamenii din imagine, folosindu-se de acele detecții pentru a îmbunătăți estimarea optical flow-ului, eliminând oamenii care pot avea o alta mișcare fata de cea a camerei, astfel stricând estimarea mișcării camerei. In următorul pas sunt calculați descriptorii HOF și MBH optical flow, iar descriptorul HOG sunt aplicați pe imaginea originala. Pentru clasificare autorii se folosesc de un SVM liniar pentru a face predicțiile, datele fiind codificate într-un vector Fisher.

La momentul apariției aceasta abordare s-a dovedit a fi cea mai buna pe seturile de date HMDB51 (57.2% acuratețe), UCF50 (91.2% acuratețe).

O alta abordare este cea [Varol et al., 2017] care se folosește de rețele convoluțiunile adânci pentru a estima acțiunea. Ca și date de intrare autorii experimentează cu diferite tehnici, de exemplu de a oferi imaginile ca RGB sau ca un optical flow. Rezultatele cele mai bune provin din folosirea metodei Brox de a obține optical flow-ul, aceasta fiind furnizata rețelei. Alte experimente au fost efectuate in găsirea numărului optim de cadre care să fie furnizat rețelei.

Când a apărut rezultatele pe seturile de date UCF101 (92.7% acuratețe) și pe HMDB1 (67.2% acuratețe) au fost cele mai bune.

O alta abordare este făcută in [Luvizon et al., 2018] care pentru o imagine se folosește de Inception-V4 pentru a extrage caracteristicile vizuale precum și estimarea locațiilor articulațiilor. Aceste 2 rezultate sunt agregate pentru a furniza predicția finala. Autorii folosesc aceste 2 informații deoarece unele acțiuni pot fi dificil de interpretat folosind doar una dintre metode. Ei dau ca exemplu acțiunea de a bea apă și acțiunea de a da un telefon care poate să fie dificil de interpretat doar din locațiile articulațiilor. Ei antrenează rețeaua folosind 16 cadre continue, obținând cele mai bune rezultate pe setul de date Penn Action for 2D action recognition (98.6% acuratețe).

Autorii setului de date ales pentru aceasta lucrare [Gourgari et al., 2013] denumit THETIS, au obținut pe filmările care oferă informații despre profunzime o acuratețe de 60.23% folosind 12 rețele SVM (pentru fiecare clasa), folosind caracteristici oferite de HOG (histogram of oriented gradients) și HOF (histogram of optical flow).

Pe setul de date ales pentru aceasta lucrare (THETIS) de tenis o abordare prezentata in [Vinyes et al., 2017] care se folosește de setul de date RGB presupune in a extrage caracteristicile dintr-o imagine folosind rețeaua Inception reantrenata pe ImageNet pentru a extrage caracteristicile fiecărei imagini. Aceste caracteristici sunt transmise unei rețele LSTM care este pe 3 straturi, fiecare strat având 90 de celule ascunse, lungimea unei secvențe folosite fiind de 100 de cadre. Acuratețea obținuta folosind aceasta tehnica este de 47.22%.

O alta abordare efectuata pe acest set de date este prezentata de [Vainstein et al., 2014] în care autorii se folosesc de silueta jucătorului dintr-un anumit cadru și imaginea de unde provine aceasta pentru a segmenta jucătorul, și a putea decupa din aceasta doar jucătorul. Aceste imagini decupate sunt grupate 2 cate 2 și formează optical flow-ul. Acest optical flow este apoi clasificat folosind CRF (conditional random fields) obținând o acuratețe de 86.44% conform autorilor.

# Tehnologii folosite

### 4.1 Python

Pentru realizarea aplicației am folosit limbajul Python care oferă o flexibilitate mare in privința alegerii unei biblioteci de deep learning, oferind cele mai populare 2 optiuni: TensorFlow si Pytorch. De asemenea oferă biblioteca Open-CV care este de mare ajutor pentru extrage cadrele din videoclipuri.

### 4.2PyTorch

Pytorch este o biblioteca de deep learming care se ocupa de operații cu tensori și cu o diferențiere automata, utilizându-se de accelerarea plăcii grafice. Principiile de proiectare sunt de a fi cat mai ușor de utilizat pentru un dezvoltator familiarizat cu limbajul Python, să pună oamenii din cercetare pe primul loc, oferind un mijloc ușor de a dezvolta modele, optimizatoare și un mijloc de încărcare al datelor ușor de folosit. Complexitatea operațiunilor din model trebuie să fie ascunsă de dezvoltator, el trebuind să folosească doar API-uri intuitive. De asemenea trebuie să aibă o performanta pragmatică, implementarea acceptând puțină complexitate in plus pentru a oferi o experiența mai intuitiva. Aceasta biblioteca este scrisă in Python cu cod optimizat de C++ [Paszke et al., 2018]. Această bibliotecă a fost folosită pentru a construi modelul de LSTM folosit pentru recunoașterea acțiunii.

### 4.3 Matplotlib

Matplotlib este o biblioteca de Python care are rolul de a crea grafice statice, animate și interactive. In cadrul proiectului aceasta a fost folosita pentru a afișa metricele de performanta, precum acuratețea modelului, și funcțiile de pierdere.

### 4.4 OpenCV

Este o biblioteca open-source care are o funcționalitate potrivita pentru sarcinile de vedere artificială in timp real. Este scrisă in principal in C++ si este oferita pentru Python, Java și MATLAB/Octave. Începând cu anul 2010 aceasta biblioteca oferă accelerare de pe placa grafica. In cadrul acestui proiect această biblioteca a fost folosita pentru a citi videoclipurile cadru cu cadru pentru a putea fi prelucrate de modelul YOLO.

### 4.5 YOLOv7

Pentru a prelua informațiile despre coordonatele persoanelor, respectiv a coordonatelor articulațiilor m-am folosit de implementarea oficiala a [Wang et al., 2022] care se poate regăsi la [github.com/yolov7, 2023]. Din implementările oferite am folosit modelul pre antrenat pe găsirea coordonatelor articulațiilor, folosindu-mă de funcționalitatea oferită de biblioteca. De exemplu o funcție folosită a fost cea care îmi translatează coordonatele detecției de la formatul in care se oferă coordonatele detecției, aceasta fiind oferita in locația x și y a punctului din mijloc, in lungimea și înălțimea detecției in pixeli, in cea care reprezintă punctul din stânga sus și cel din dreapta jos.

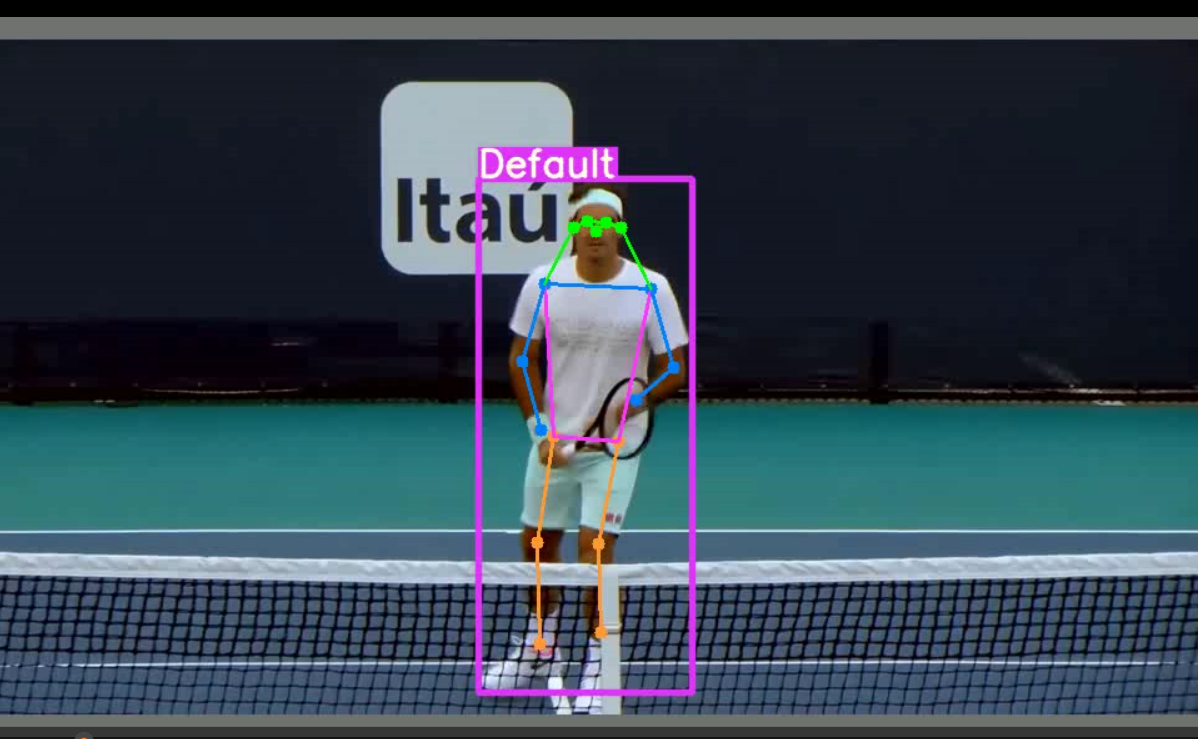


Figura 4.1: Exemplu de output al modelului YOLO pre antrenat in a detecta coordonatele membrelor

# Abordarea propusa

### 5.1 Descrierea setului de date

Setul de date folosit pentru acest proiect este THETIS [Gourgari et al., 2013], care este alcătuit din 8734 din videoclipuri, 1980 fiind in formatul RGB, executate de 31 de amatori și 24 de jucători experimentați de tenis. Durata totala a videoclipurilor este de 7 ore și 15 minute. Fiecare persoana repeta cate o lovitura de 3-4 ori. Acest set de date conține mai multe tipuri de videoclipuri, fiind captate cu o camera care are senzor de profunzime. Pentru aceasta lucrare am folosit videoclipurile RGB, cu scopul ca mai apoi modelul să fie rulat pe videoclipuri cu tenis captate cu o camera video obișnuita. Setul conține 12 lovituri din tenis care sunt adnotate. Loviturile sunt:

* Backhand with two hands
* Backhand
* Backhand Slice
* Backhand volley
* Forehand open stands
* Forehand slice
* Forehand volley
* Service flat
* Service kick
* Service slice
* Smash

Aceste 12 lovituri sunt executate in lipsa unei mingi de tenis.

Pentru dezvoltarea modelului am ales la întâmplare aproximativ 10% din videoclipurile din fiecare clasă rezultând 192 de videoclipuri. Restul de 1788 au fost folosite pentru antrenament.



Figura 5.1: Exemplu de imagini preluate din setul de date THETIS

### 5.2 Configurarea modelului YOLO

Implementarea modelului YOLOv7 este cea oficiala a lucrării [Wang et al., 2022] care a fost adaptată pentru a fi compatibila cu cerințele mele. Am definit o clasă intitulata „YoloModel” care se ocupa cu extragerea coordonatelor persoanelor și a articulațiilor. Pentru aceasta am definit funcția „inference\_on\_image”, care primește ca parametru o imagine și are ca ieșire coordonatele fiecărei persoane cât și pozițiile fiecărui membru. Astfel, ieșirea acestei funcții are dimensiunea număr\_persoane x 58. Aceasta funcție este apelata in funcția principala „read\_from\_video” care primește ca parametru calea către un videoclip. Aceasta funcție procesează videoclipul cadru cu cadru, luând in considerare persoana care are dimensiunile maxime. Aceasta limitare este făcută deoarece in setul de date sunt prezente și alte persoane, persoana care trebuie să fie luată în calcul este doar cea de dimensiunea cea mai mare. De asemenea, in cadrul videoclipurilor reale, neavând informații despre teren, va fi luat in considerare doar persoana cea mai apropiată de camera. Pentru a transforma coordonatele detecțiilor într-un format mai ușor de procesat, am creat fișierul „utils\_detection.py”, care conține mai multe funcții ajutătoare. Deoarece algoritmul ne furnizează rezultatul coordonatelor unde se afla persoana alipit de rezultatul coordonatelor membrelor, am creat funcția „get\_detection\_box\_yolo” care primește ca parametru detecția întreaga. Din aceasta extrag coordonatele aferente acestui tip de detecție și îl transform din formatul {x, y, w, h} care reprezintă coordonatele centrului, lungimea și lățimea dreptunghiului, in formatul {x1, y1, x2, y2} care semnifica coordonatele coltului din stânga sus, respectiv dreapta jos. Pentru formatarea coordonatelor membrelor, care vin într-un vector care are dimensiunea 1, fiecare articulație din cele 17 fiind de forma {x, y, încredere} x, y reprezentând punctul unde se afla membrul, iar „încredere” reprezentând încrederea modelului ca membrul respectiv să fie in acea locație. Pentru fiecare detecție de persoana am ales să schimb dimensiunea acestui vector astfel acesta să aibă dimensiunea 17x2, 17 reprezentând numărul de articulații detectate, iar 2 fiind punctele respective. Pentru fiecare articulație care are valoarea „încredere” < 0.5 am decis să fac coordonatele nule. Folosind aceasta formatare pentru a fi mai ușor de a normaliza datele.

### 5.3 Transformarea setului de date

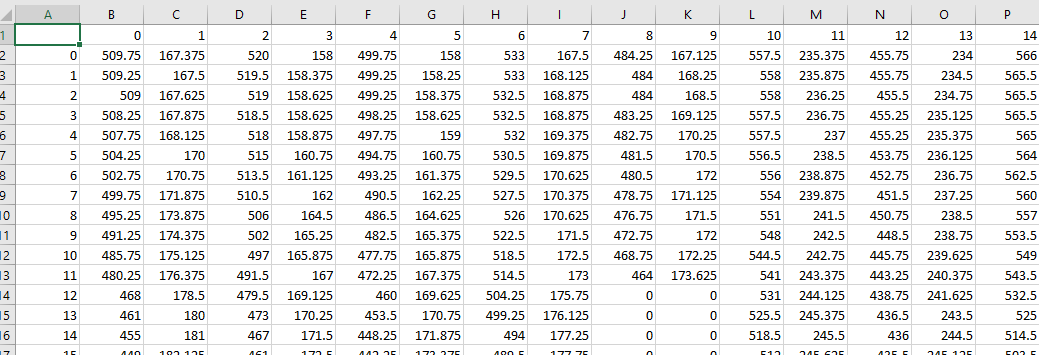
Pentru a transforma setul de date din videoclipuri in secvențe de numere care pot fi procesate am creat fișierul „create\_dataset.py”, care pentru fiecare videoclip din setul de date aplica modelul YOLO și salvează coordonatele articulațiilor și al dreptunghiului unde se afla jucătorul in cate 2 fișiere csv pentru fiecare videoclip.

Cele doua fișiere sunt denumite după formatul:

* „**nume\_original\_limbs.csv**” pentru coordonatele articulațiilor;
* „**nume\_original\_yolo.csv**” pentru salvarea coordonatelor dreptunghiurilor;

Astfel fiecare fișier, numărul de linii va fi egal cu numărul de cadre al clipului. Coordonatele articulațiilor vor fi in ordinea formatului COCO, grupate 2 cate doua in funcție de ce articulație reprezintă. Pentru formatul YOLO se salvează cate 5 numere care reprezintă, coordonatele punctului din stânga sus al dreptunghiului, coordonatele dreapta jos, și încrederea pe care o are modelul in predicția făcută. Fișierele sunt grupate in foldere in funcție de ce clasă reprezintă, astfel o să avem 12 foldere care conțin fișierele csv după formatul enunțat mai sus. In aceste fișiere datele nu sunt normalizate. Crearea acestor fișiere csv este benefică deoarece evita rularea algoritmului YOLO de fiecare dată când dorim să antrenam algoritmul de recunoaștere a acțiunii, algoritmul YOLO fiind o operațiune dificilă din punct de vedere computațional.

Crearea acestui set de date a durat aproximativ 7 ore pentru toate cele 1980 de videoclipuri. In figura 5.2 este exemplificat un astfel de fișier.

Figura 5.2: Exemplu de fișier produs pentru antrenarea modelului LSTM care conține coordonatele nenormalizate din cadrul unui videoclip

### 5.4 Normalizarea datelor

O etapa importanta în încărcarea datelor este preprocesarea lor, întrucât dorim ca toate caracteristicile să folosească o scala similara. Întrucât pentru antrenarea modelului vrem să folosim doar coordonatele articulațiilor, dorim să le aducem in intervalul [0, 1] pe fiecare. Pentru aceasta, le vom normaliza in funcție de poziția lor relativa din dreptunghiul de detecție, folosind normalizarea min-max. De exemplu dacă avem dreptunghiul de detecție de coordonate (10, 10), (100, 100) și punctul care reprezintă una din articulații ca fiind (55, 55), dorim ca acesta să aibă coordonatele normalizate ca fiind (0.5, 0.5).

In fișierul „utils\_detection.py” am creat funcția „normalize\_detection\_limbs” care se ocupă cu acest lucru:

def normalize\_detection\_limbs(yolo\_boxes, detection\_limbs):

    detection\_for\_limbs = detection\_limbs.copy()

    for idx in range(yolo\_boxes.shape[0]):

        x\_std\_normalization = yolo\_boxes[idx, 2] - yolo\_boxes[idx, 0]

        y\_std\_normalization = yolo\_boxes[idx, 3] - yolo\_boxes[idx, 1]

        detection\_for\_limbs[idx, :, 0] = (detection\_for\_limbs[idx, :, 0] - yolo\_boxes[idx, 0]) / x\_std\_normalization

        detection\_for\_limbs[idx, :, 1] = (detection\_for\_limbs[idx, :, 1] - yolo\_boxes[idx, 1]) / y\_std\_normalization

    for idx in range(detection\_for\_limbs.shape[0]):

        for jdx in range(detection\_for\_limbs.shape[1]):

            if detection\_for\_limbs[idx,jdx,0] < 0 or detection\_for\_limbs[idx,jdx,1] < 0:

                detection\_for\_limbs[idx,jdx,0] = 0

                detection\_for\_limbs[idx,jdx,1] = 0

    return detection\_for\_limbs

Aceasta primește ca parametru dreptunghiurile de detecție și coordonatele articulațiilor. Cele doua sunt denumite intuitiv, ele fiind de tipul „numpy array” pentru a putea face operațiile într-un timp cat mai scurt. Astfel pentru fiecare dreptunghi dintr-un videoclip, se calculează pentru fiecare dimensiune diferența intre punctul maxim și punctul minim. Întrucât variabila „detection\_for\_limbs” are dimensiunile numar\_cadre x 17 x 2 ne este ușor să normalizam, putând cu ușurința să scădem de pe coordonata x, respectiv y punctul de coordonate minim, ca apoi să putem să normalizăm folosind diferența dintre maxim și minim. Ultimele 2 parcurgeri sunt in cazul in care coordonatele normalizate sunt negative, caz in care le aducem la valoarea 0.

### 5.5 Încărcarea datelor și crearea secvențelor

Pentru a menține datele in memorie am creat clasa „DatasetLoader” care are ca părinte clasa „torch.utils.data.Dataset” care va facilita antrenarea modelului. Aceasta clasă se ocupa cu încărcarea tuturor datelor din toate fișierele csv și de crearea secvențelor de date. Pentru a crea secvențele in fișierul „utils\_detection.py” am creat funcția „get\_all\_sequences\_from\_2D\_format(matrix, sequence\_length, y\_value, step = 1)” care primește ca parametru coordonatele membrelor, lungimea pe care să o reprezintă o secvență și valoarea codificată a clasei care este reprezentata in videoclip.

Crearea secvențelor poate să fie reprezentata in modul următor:

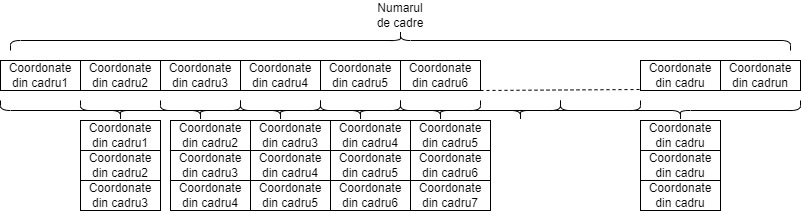


Figura 5.3: Ilustrare a modului cum sunt încărcate datele in memorie pentru a fi procesate de către rețeaua LSTM

In acest exemplu lungimea unei secvențe este de 3, iar pasul este 1. Acest lucru înseamnă ca intre fiecare 2 secvențe consecutive, numărul de coordonate intercalate este de lungime secvență – 1, in exemplul de mai sus, numărul de cadre intercalate fiind de 2. Pentru reprezentarea datelor in cadrul proiectului, pentru fiecare videoclip am ales să mențin pasul cu valoarea 1, pentru a avea un număr de date cat mai mare, stabilirea lungimii optime a unei secvențe fiind un hiper-parametru. Toate secvențele de cadre vor fi adunate într-o singura variabila, ea având dimensiunile numar\_secvente x lungime\_secventa x 34. De exemplu pentru lungimea unei secvențe de 20, vom avea 95915 de secvențe in total pentru antrenare. Întrucât părintele acestei clase este „torch.utils.data.Dataset” am supraîncărcat metodele \_\_len\_\_(self) care returnează numărul de secvențe in total și \_\_getitem\_\_(self, index) care returnează secvența indicată de index și valoarea codificată a acțiunii care este reprezentată. La final, datele sunt transformate in tensori, care vor fi încărcați pe GPU, in cazul in care acesta exista, in caz contrar fiind încărcat pe CPU.

Pentru ca încărcarea in model să fie cat mai facila, aceasta clasă este intializată pentru setul de date de antrenare cat și pentru setul de date de testare. Apoi ne folosim de clasa DataLoader din Pytorch care are ca scop „împachetarea” datelor având o lungime specificata ca hiper-parametrul „BATCH-SIZE”. Pentru antrenare datele sunt amestecate pentru a oferi o diversitate in cadrul unei actualizări a ponderilor modelului.

### 5.6 Crearea modelului

Pentru crearea modelului am creat o clasă care să fie ușor de personalizat, primind ca parametrii in constructor numărul de caracteristici al unei intrări, in cazul de față 34, numărul de straturi, numărul de unități hidden al rețelei LSTM precum și lungimea secvenței pe care o va primi ca input. Codul prin care este definita rețeaua este:

class LSTM(nn.Module):

    def \_\_init\_\_(self, num\_features, hidden\_units, seq\_length):

        super(LSTM, self).\_\_init\_\_()

        self.num\_features = num\_features

        self.num\_classes = 12

        self.hidden\_units = hidden\_units

        self.seq\_length = seq\_length

        self.num\_layers = 3

        self.lstm = nn.LSTM(input\_size = self.num\_features,

                            hidden\_size = self.hidden\_units, num\_layers = self.num\_layers,

                            batch\_first = True,

                            dropout = 0.7)

        self.fc\_1 = nn.Linear(self.hidden\_units, 256)

        self.fc\_final = nn.Linear(256, self.num\_classes)

        self.dropout = nn.Dropout(0.7)

        self.relu = nn.ReLU()

    def forward(self, x):

        batch\_size = x.shape[0]

        h0 = torch.zeros(self.num\_layers, batch\_size, self.hidden\_units).to("cuda:0").requires\_grad\_()

        c0 = torch.zeros(self.num\_layers, batch\_size, self.hidden\_units).to("cuda:0").requires\_grad\_()

        outputs, (hn, \_) = self.lstm(x, (h0, c0))

        outputs = outputs[:, -1, :]

        outputs = self.dropout(outputs)

        out = self.relu(outputs)

        out = self.fc\_1(out)

        out = self.dropout(out)

        out = self.relu(out)

        out = self.fc\_final(out)

        return out

    def return\_train\_data():

        SEQUENCE\_LENGTH = 23

        INPUT\_SIZE = 34

        HIDDEN\_SIZE = 512

        return SEQUENCE\_LENGTH, INPUT\_SIZE, HIDDEN\_SIZE

Aceasta clasă moștenește torch.nn.Module pentru a permite antrenarea și construcția modelului. In constructor se inițializează parametrii modelului.

Aceasta este compusă din:

* **Rețeaua LSTM**: input\_size reprezintă numărul de caracteristici al unei intrări (34), hidden\_size numărul de unități de LSTM, num\_layers numărul de straturi LSTM
* **Rețeaua MLP**: este alcătuita din 2 unități, prima având dimensiunea numarul\_unitati\_LSTM x 256, cea de-a doua având 256 x 12 (12 fiind numărul de clase care trebuie prezis)

Aceste 2 componente au un probabilitate de „dropout” in cadrul antrenării de 0.7. Conform [Srivastava et al., 2014] „dropout” presupune in dezactivarea anumitor neuroni pentru a reduce rata de supra învățare a modelului, cea in care modelul se descurca foarte bine pe datele de antrenare dar eșuează in a face predicții corecte pe alte date. In această lucrare, folosind această tehnica autorii ne prezintă eroarea pe toate seturile de date testare scade. De exemplu, pe setul de date CIFAR-100, cu rata de dropout aplicata pe toate componentele rețelei eroarea a scăzut de la 43.48% la 37.20% fără să aducă alte îmbunătățiri asupra setului de date sau să augmenteze datele.

Funcția „return\_train\_data()” care este statica are rolul de a facilita inițializarea modelului in timpul inferenței, pentru fiecare model antrenat, am definit cate o funcție separată.

Funcția „forward(self, x)” este apelata pentru a oferi predicțiile pentru fiecare intrare. Variabila „x” reprezintă datele care urmează a fi parcuse, având forma batch\_size x lungime\_secventa\_34, astfel oferindu-se „batch\_size” predicții.

În această funcție se mai initializează și „starea ascunsa” (variabila h0) care retine informațiile importante din intrările anterioare ale secvenței. Aceasta poate „uita” sau „retine” informațiile. O alta variabila necesara care este inițializată este „starea celulei” (variabila c0), care reține informațiile pe termen lung din intrările secvenței.

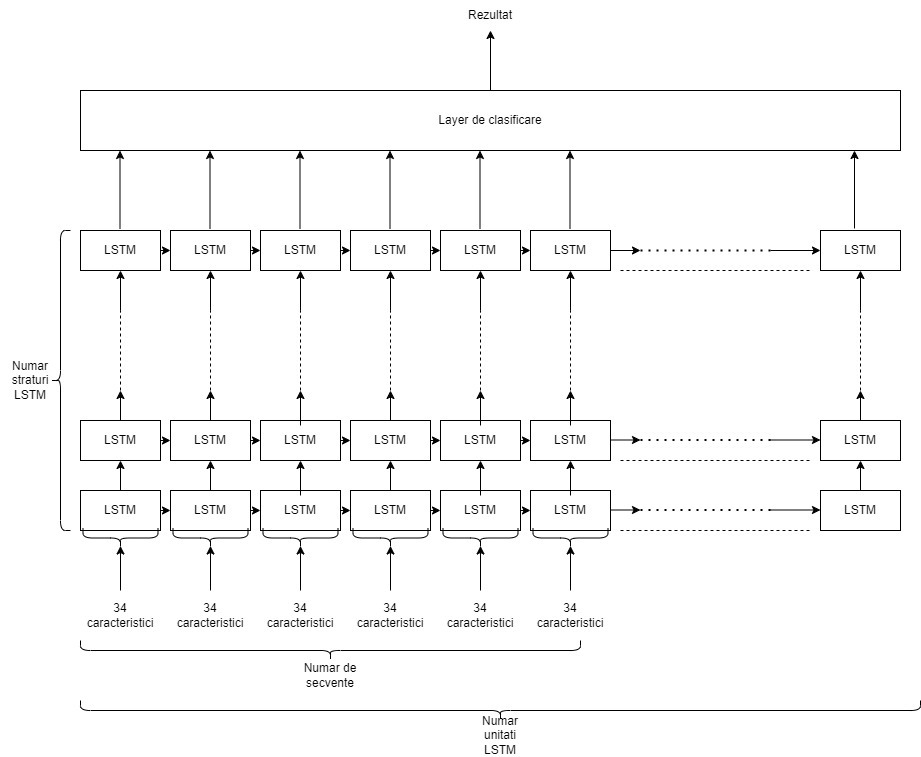


Figura 5.3: Prezentarea structurii modelului și ce reprezintă fiecare notație într-un mod grafic

### 5.7 Antrenarea modelului

Pentru antrenarea modelului mă folosesc de funcționalitatea PyTorch pentru antrenarea sa. Pentru antrenare mă folosesc de optimizatorul Adam și funcția de pierdere „CrossEntropyLoss” care are formula: . Din aceasta ecuație simbolizează valoarea prezisă de modelul nostru, iar reprezintă valoarea reală a acelei clase.

Codul pentru antrenarea modelului:

for epoch in range(num\_epochs):

    time\_begin = time.perf\_counter()

    model.train()

    total\_loss = 0

    epochs.append(epoch)

    for X, y in train\_loader:

        output = model(X)

        loss = loss\_function(output, y)

        optimizer.zero\_grad()

        loss.backward()

        torch.nn.utils.clip\_grad\_norm\_(model.parameters(), 0.5)

        optimizer.step()

        total\_loss += loss.item()

    time\_end = time.perf\_counter()

    avg\_loss = total\_loss / num\_batches

    print(f"Train loss for epoch {epoch}: {avg\_loss}, duration {time\_end-time\_begin} seconds")

    train\_loss.append(avg\_loss)

    test\_loss, precision, precision2 = test\_model(test\_loader, model, loss\_function)

    if test\_loss < minimum\_testing\_error:

        minimum\_testing\_error = test\_loss

        minimum\_model = copy.deepcopy(model.state\_dict())

        minimum\_epoch = epoch

    torch.save(model.state\_dict(), f"intermediary\_results\\saved\_checkpoint\_{model.\_\_class\_\_.\_\_name\_\_}\_{epoch}\_epoch.pth")

    validation\_loss.append(test\_loss)

    precision\_validation.append(precision)

    precision2\_validation.append(precision2)

Modelul este antrenat pentru un număr de epoci stabilit ca și parametru. Pentru a începe antrenamentul este necesar să punem modelul in modul „train”, acest pas fiind făcut la începutul fiecărei epoci deoarece funcția „test\_model(test\_loader, model, loss\_function)” trece modelul in modul de test. Rolul acestei funcții este să facă predicții pe setul de date destinat testării, care nu este inclus in antrenare și de a returna eroarea pe acesta precum și precizia modelului. Variabilele „train\_loss”, „validation\_loss”, „precision\_validation” și „precision2\_validation” sunt de tip lista și au scopul de a retine rezultatele pe care modelul le obține de-a lungul antrenării. Variabilele „X” și „y” reprezintă date extrase din setul de date de antrenament, fiind extrase sub forma de batch. De exemplu variabila X o să aibă dimensiunile batch\_size x lungime\_secventa x 34, iar pentru y o să fie de forma batch\_size x 1. Apelul funcției „optimizer.zero\_grad()” are rolul de a reseta toți gradienții modelului. Daca nu am folosi aceasta funcție comportamentul implicit al PyTorch este de a acumula gradientul curent cu cei de la pașii anteriori.

Apelul funcției „loss.backward()” face ca gradienții modelului să fie calculați in funcție de eroarea obținuta pe setul de date. După acest pas, pentru a actualiza parametrii modelului trebuie apelata funcția „optimizer.step()”, care trece prin fiecare parametru al modelului și îl actualizează folosindu-se de valoarea gradientului calculata anterior. Pentru a limita gradienții care depășesc o anumită valoare, folosesc funcția „torch.nn.utils.clip\_grad\_norm(model.parameters(), valoare)”. In acest exemplu am limitat parametrii la valoarea 0.5, asta simbolizând faptul că în momentul în care aceasta valoare este depășită, gradientul folosit va avea valoarea 0.5.

In cadrul antrenării, după fiecare epoca afișez statisticile aferente setului de date de antrenare precum și a celui de test (acest lucru făcându-se in functia „test\_model”). De asemenea la finalul fiecărei epoci se salvează modelul in memoria calculatorului, precum și modelul care a avut cea mai mica eroare într-o variabilă.

# Experimente și rezultate

### Prezentarea metricilor de acuratețe și primele rezultate

Pentru antrenarea modelului am folosit optimizatorul Adam având o rata de învățare de 1e-4. O rata de învățare mai mică, ducea la o antrenare mult mai lenta, iar o rata de învățare mai mare, ducea la rezultate mai proaste. Am realizat modelul de clasificare ca in capitolul 5.6, rezultatele optime fiind obținute cu aceasta rețea.

Pentru a evalua modelele am folosit doua metrici de acuratețe:

* **Metrica 1**: pe fiecare secvență de date este făcută predicția, acuratețea finala fiind reprezentat de formula unde reprezintă numărul de secvențe prezise corect, reprezentând numarul de secvențe care au fost prezise incorect
* **Metrica 2**: pentru ca rezultatele să poate fi comparate in mod direct cu [Vinyes et al., 2017], pentru fiecare videoclip din setul de testare, am făcut predicțiile necesare pe fiecare secvența, iar apoi predicția finala pentru acest videoclip a fost reprezentata de predicția majoritara. Aceasta predicție finala este comparata cu valoarea de adevăr pe care o reprezintă videoclipul, iar apoi se face calculul prezentat in formula de la Metrica 1.

Experimentele au fost realizate oferind o predicție o data la un număr de cadre predefinit. Numarul acesta de cadre a fost denumit și „secvența” iar valorile testate au fost următoarele: 16, 18, 20, 23. Pe parcursul antrenării graficul funcției de pierdere a arătat asemănător pentru oricare din lungimile testate de secvențe. Un exemplu este prezentat în figura 6.1

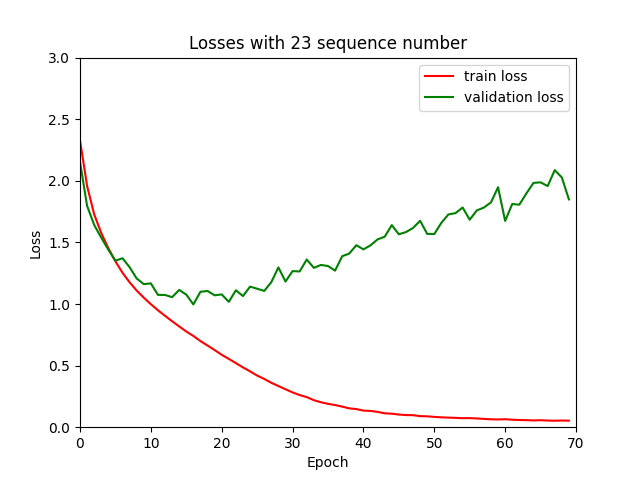


Figura 6.1: Graficul evoluției funcției de pierdere, modelul LSTM având un input de secvențe de 23

Pentru doua din aceste lungimi de secvență graficele de îmbunătățire a acurateței sunt:

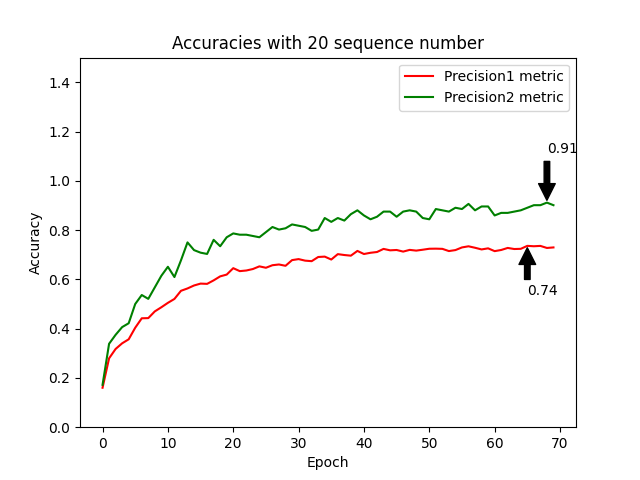


Figura 6.2: Graficul evoluției acurateței, modelul LSTM având un input de secvențe de 20

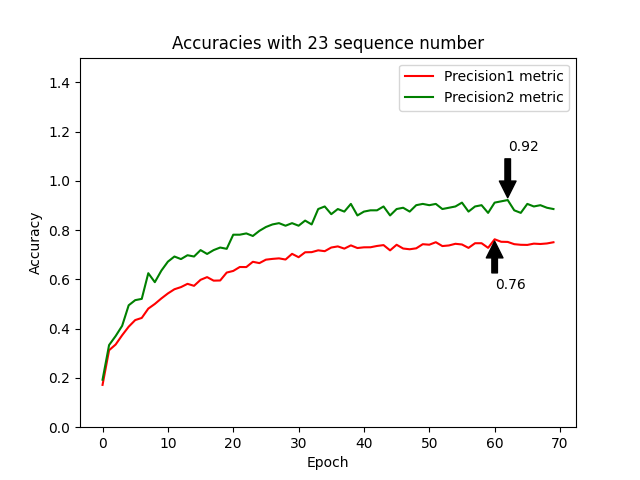


Figura 6.3: Graficul evoluției acurateței, modelul LSTM având un input de secvențe de 23

Putem observa din figura 6.1 ca funcția de pierdere pentru setul de testare începe să crească in jurul epocii 10, simbolizând apariția fenomenului de overfit. Din figurile 6.2 și 6.3 putem observa ca acuratețea pentru setul de date de testare continuă să crească asemănător pentru cele 2 metrice. Îmbunătățirile asupra acurateței par să atingă un platou in jurul epocii 40, valorile maxime fiind atinse in jurul epocii 60. Mai putem observa ca performanța metricii 1 este îmbunătățita în momentul în care avem un număr mai mare de secvențe. Având un număr mai mare de secvențe, modelul are mai multe informații despre ceea ce se întâmplă. Acest lucru devine mai aparent in momentul in care ne uitam la figura 6.4 care prezinta acuratețile folosind mai multe lungimi de secvență.

Modelul care are cele mai bune rezultate are ca și hiperparametrii:

* **LSTM**: 3 straturi ale rețelei cu 512 hidden units cu o rata de dropout de 0.7
* **Layer de clasificare**: 2 straturi de MLP, primul având dimensiunea de 512 x 256, iar cel de-al doilea de 256 x 12. Se aplica o rata de dropout de 0.7

|  |  |  |
| --- | --- | --- |
| Performanta modelului in funcție de numărul de secvențe ca input | | |
| Număr secvențe | Acuratețe maxima metrica 1 | Acuratețe maxima metrica 2 |
| 23 | 0.76 | 0.92 |
| 20 | 0.74 | 0.91 |
| 18 | 0.73 | 0.92 |
| 16 | 0.72 | 0.93 |

Figura 6.4

In cadrul acestui proiect metrica cea mai importanta este metrica 1 întrucât pentru inferența din cadrul unui meci real de tenis vom dori să rulam predicția pe secvențe de cadre și să afișam mișcarea efectuată. Putem observa din tabel ca metrica 1 creste progresiv in funcție de lungimea secvenței pe care modelul o primește, valoarea maxima fiind de 0.76.

Comparativ cu [Vinyes et al., 2017], aceasta abordare pare să aducă o îmbunătățire a acurateței, acuratețea prezentata de autor fiind de 0.47.

In lucrarea [Gourgari et al., 2013] autorii au executat experimente similare folosind metrica de acuratețe 2, folosind videoclipuri cu date despre profunzime și coordonatele articulațiilor in trei dimensiuni comparativ cu metoda prezentata care se folosește doar de videoclipuri RGB. Astfel ei au obținut rezultate de 0.60.

O alta abordare se poate găsi in [Vainstein et al., 2014] unde autorii s-au folosit de optical flow-ul oferit de siluetele jucătorilor, care apoi a fost clasificat folosind CRF. Acuratețea obținuta de aceștia a fost de 0.86.

Alte configurații ale rețelei nu au avut rezultate la fel de bune. De exemplu alte experimente au constat într-o rețea in care numărul de straturi LSTM este 1, rezultatul maxim al metricii 1 a fost de 0.70, metrica 2 având acuratețea de 0.88. Alt experiment a fost să micșorez numărul de hidden units a rețelei LSTM la valoarea de 256, neschimbând numărul de straturi, rezultatul maxim pentru metrica 1 fiind 0.70, iar pentru metrica 2 de 0.84. Pentru aceste 2 experimente numărul de secvențe a fost păstrat constant și anume la valoarea de 23. Astfel pentru a interpreta rezultatele am păstrat modelul cu cea mai bună performanță și anume cel care are numărul de hidden units pentru rețeaua LSTM de 512 și numărul de straturi al rețelei LSTM a fost păstrat la 3.

### 6.2 Interpretarea rezultatelor



Figura 6.5: Matricea de confuzie oferita de modelul care are ca input un număr de 23 secvențe

In figura 6.5 este prezentata matricea de confuzie a predicțiilor pentru modelul standard care folosește 23 de cadre pentru efectuarea predicțiilor. Vizualizarea matricei de confuzie este importanta deoarece ne oferă o vizualizare a performanței algoritmului per clasa. Axa orizontala reprezintă valoarea prezisă de model, iar axa verticala reprezintă valoarea reală a clasei. Această matrice corespunde metricii 1.

Putem observa ca rezultatele pot fi interpretate. De exemplu diferența intre „kick service” și „slice service” este minoră, ele diferind doar în direcția data mingii. Clasa „smash” diferă de toate celelalte tipuri de serve prin faptul ca nu este prima lovitura in cadrul unui joc. Modelul tinde să prezică smash când clasele reale erau unele de serviciu. Videoclipurile din setul de date THETIS nu conțin mingea de tenis, acest lucru ar putea explica de ce aceste confuzii sunt făcute.

Pentru mișcările care sunt mai distincte putem observa ca modelul are o acuratețe buna, 0.94 pentru „forehand openstands”, 0.88 pentru clasa „backhand2hands” sau 0.87 pentru „backhand”.

Alcatuirea unui set de date ce contine videoclipuri din cadrul unor meciuri de tenis, in care loviturile jucătorilor sunt adnotate, pot aduce o îmbunătățire modelului, deoarece consider ca putem include un model precum cel prezentat in [Huang et al., 2019] care ne poate furniza coordonatele mingii de tenis. Aceste coordonate le-as include alături de cele ce conțin membrele pentru a crea un model ce are o perspectiva mai amplă.

# Descrierea aplicației

Scopul acestei aplicații este de a oferi unui utilizator o interfață grafică intuitivă, pe care o poate folosi pentru a afla statisticile referitoare la loviturile pe care un jucător le-a făcut in cadrul unui meci de tenis. Aplicația are ca scop să ruleze pe desktop.

Aplicația este împărțită in 2 părți:

* **Front-end**: interfața grafica a aplicației care a fost dezvoltata cu librăria tkinter din python
* **Back-end**: se ocupa cu prelucrarea videoclipurilor, gestiunea unei baze de date, și oferirea rezultatelor

### 7.1 Front-end

Interfața grafica a fost construita folosind biblioteca tkinter din python, fiind singura biblioteca de interfață grafica care vine preinstalata cu python. Funcționalitatea principală constă în facilitarea selectării unui videoclip de către utilizator. Intenția utilizatorului de a selecta un videoclip este solicitată prin apăsarea butonului „Select file” care va deschide o alta fereastra, in care acesta va selecta videoclipul pe care dorește să-l analizeze. Interfața grafica va afișa doar fișierele care au formatul „avi” și „mp4”, formatul compatibil cu acest program. In momentul in care detecția persoanei este in progres, butonul „Select file” este dezactivat, astfel utilizatorul poate selecta un alt videoclip in momentul in care cel curent s-a terminat de analizat. De asemenea pe parcursul analizei, utilizatorul primește informații despre ceea ce este executat, stările acesteia fiind:

* „**The file is found at ‚location’**” – în cazul în care fișierul video a fost deja analizat, „location” reprezentând locația fișierului video deja analizat.
* „**Finding the person in the video ‚location’**” – simbolizează faptul ca detecția YOLO este in curs de desfășurare, „location” reprezentând locația fișierului video care este in curs de analiza.
* „**Predicting the action from ‚location’**” – simbolizează faptul ca predicția mișcărilor executate este in curs de desfășurare, „location” reprezentând același lucru ca mai sus
* „**Creating the zip file**” – salvarea fișierelor este in curs de executare.
* „**Finished, zip file at location ‚location’**” – programul a terminat de analizat videoclipul, locația unde a fost salvat fișierul zip fiind indicata.

De asemenea finalul detecției, programul va deschide arhiva „zip”, care conține fișierele oferite de acest program, folosind aplicația prestabilita a sistemului de operare. In figura 7.1 este prezentata pagina principala a programului.

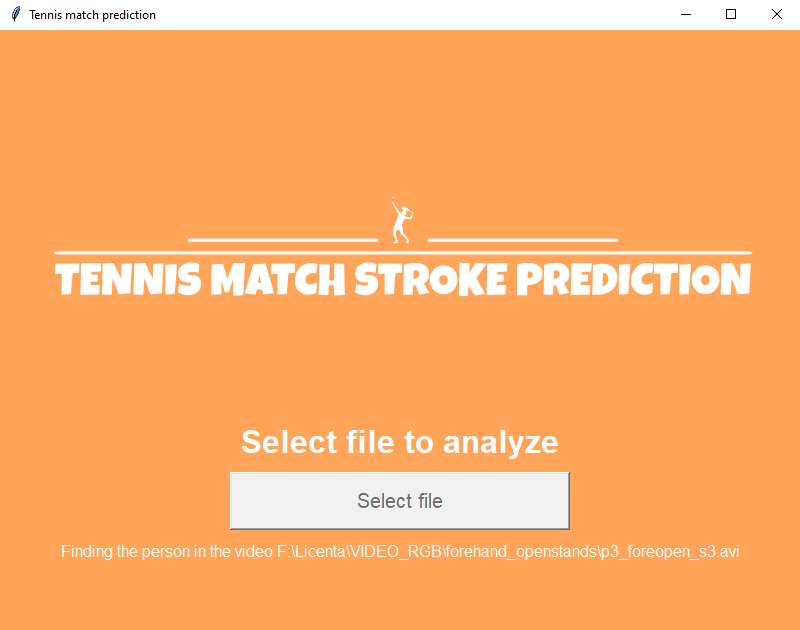


Figura 7.1: Interfața grafica a aplicației

Funcția care va fi executata in momentul in care este apăsat butonul pentru a selecta un fișier este:

def browse\_file\_run\_inference():

    file\_types = [

        ("Avi files", "\*.avi"),

        ("Mp4 files", "\*.mp4")

    ]

    filename = filedialog.askopenfilename(initialdir = "/",

                                          title = "Select a File",

                                          filetypes = file\_types)

    if len(filename) > 0:

        SELECT\_BUTTON["state"] = "disabled"

        STATUS\_LABEL['text'] = ""

        filename = filename.replace('/', '\\')

        thread = Thread(target=make\_predictions, args=[filename])

        thread.start()

Aceasta funcție va deschide o ferestră de file explorer unde utilizatorul va putea selecta videoclipul care dorește să îl analizeze. Utilizatorul va alege ce tip de fișier să ți fie afișat, acesta putând să aleagă între fișiere mp4 sau avi. Daca un videoclip a fost ales, butonul va fi blocat și nu o să mai poată fi apăsat pana când fișierul nu este analizat complet. Va fi apelata funcția make\_predictions care primește ca parametru un fișier video într-un nou thread, lucru necesar pentru ca interfață grafica să nu fie blocata cat timp fișierul video este analizat.

### 7.2 Back-end

Sarcina aplicației Back-end este de a lega aplicația grafica cu modelul LSTM. Are rolul de a optimiza procesul de calcul, detectând daca un videoclip a fost deja analizat sau nu. O baza de date este conectată cu rolul de a tine evidenta asupra videoclipurilor care au fost analizate.

Back-end-ul este alcătuit din următoarele componente:

* **Baza de date**
* **Semnătura digitala a unui fișier video**
* **Generarea videoclipului ce conține loviturile**
* **Generarea fișierelor de statistici**

### 7.2.1 Baza de date

In cadrul acestui proiect baza de date folosita este „Microsoft SQL Server” și are rolul de a reține fișierele video care au fost deja analizate de către utilizator. Legătura dintre server și python a fost făcută utilizând biblioteca „pyodbc”. Baza de date conține o singura tabela, care are ca proprietăți:

* **HASH** (PK) – semnătura hash a fișierului, generat folosind algoritmul md5
* **VIDEOPATH** – conține calea către fișierul zip generat

Pentru fiecare execuție a algoritmului, programul adaugă in aceasta tabela, modificările pe care utilizatorul le face. Actualizarea acesteia se face la fiecare analiza a unui videoclip. Valoarea HASH este calculată folosind conținutul videoclipului, astfel fiind independenta de numele și locația acestui fișier. In cazul în care utilizatorul introduce din nou un videoclip care deja a fost analizat, programul îl va împiedica și i se va indica locația rezultatelor deja efectuate. In modul predefinit, programul salvează clipurile in folderul „saved\_videos” pe care acesta îl creează in locația de unde programul este executat. Acest pas are ca rol împiedicarea analizei de doua ori a aceluiași fișier video, întrucât aceasta operațiune este costisitoare. Dacă intre timp fișierul zip a fost șters și utilizatorul dorește să reanalizeze același videoclip, in baza de date se va reactualiza locația și numele fișierului zip oferit de program. În figura 7.2 este un exemplu de cum arată datele in baza de date.

Codul care face conexiunea cu baza de date este:

def create\_database\_connection():

    cnxn\_str = ("Driver={SQL Server};"

                f"Server={platform.node()}\SQLEXPRESS;"

                "Database=VideoEntry;"

                "Trusted\_Connection=yes;")

    cnxn = pyodbc.connect(cnxn\_str)

    cursor = cnxn.cursor()

    return cnxn, cursor

Pentru funcționarea acestui program, utilizatorul trebuie să aibă o instanță a „Microsoft SQL Server” și o baza de date denumită „VideoEntry”, ce conține o tabela denumită “EntryVideo”. Funcția „create\_database\_connection()” va returna conexiunea și cursorul către această baza de date, cursorul fiind folosit pentru a o interoga.

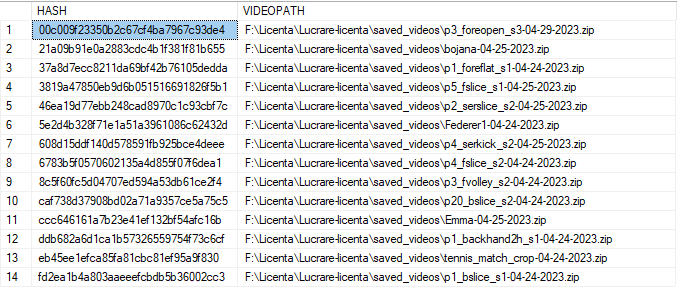


Figura 7.2: Exemplu de conținut al bazei de date

### 7.2.2 Semnătura digitala a unui fișier video

Pentru a genera semnătura digitală a unui fișier video, am calculat valoarea hash MD5. Pentru aceasta am folosit biblioteca hashlib. Codul care efectuează aceasta operație este:

def get\_hash\_of\_file(file\_path):

    hash\_md5 = hashlib.md5()

    with open(file\_path, "rb") as f:

        for chunk in iter(lambda: f.read(4096), b""):

            hash\_md5.update(chunk)

    return hash\_md5.hexdigest()

Acesta primește ca parametru calea către un fișier video, care este deschis folosind modul „rb” care permite citirea fișierelor binare. Fișierul este citit in blocuri de cate 4096 de octeți pentru a nu umple memoria calculatorului în cazul in care fișierul este de dimensiuni foarte mari. Valoarea hash este actualizată pentru fiecare bloc citit, iar apoi funcția va returna valoarea hash-ului in format de sir de caractere in baza 16. Aceasta codificare este unică pentru fiecare videoclip, fiind bazată doar pe conținutul acestuia.

Inserția in baza de date a semnăturii digitale este făcută in funcția:

def insert\_value(hash, video\_path):

    global cursor, cnxn

    cursor.execute(f"INSERT INTO EntryVideo VALUES('{hash}', '{video\_path}');")

    cnxn.commit()

Funcția primește ca și parametrii șirul de caractere ce reprezintă semnătura digitala și calea către unde urmează să fie salvata arhiva cu rezultatele. Este utilizat cursorul pentru a rula comanda care ne va introduce valoarea semnăturii și calea către fișierul cu rezultate.

### 7.2.3 Generarea videoclipului ce conține loviturile

Pentru generarea predicțiilor fiecărui videoclip primul pas a fost ca pentru fiecare cadru din clip să fie detectata persoana care are câmpul de detecție cel mai mare, pentru a analiza persoana dominantă din acel cadru. In cazul unui meci real de tenis, fiind analizat jucătorul cel mai apropiat de camera. In pasul următor, din fiecare cadru analizat, sunt extrase coordonatele membrelor, folosind algoritmul YOLO. Acestea, care sunt 34 la număr pentru fiecare cadru, sunt apoi structurate in grupuri consecutive de cate 23 de cadre, care nu se intersectează, facându-se pe fiecare grup predicția mișcării efectuate, folosind rețeaua LSTM enunțată in capitolul 5.6.

In pasul următor este creat un videoclip pornind de la cel original, care pentru fiecare cadru va avea adnotat dreptunghiul de detecție și mișcarea efectuata de jucător.

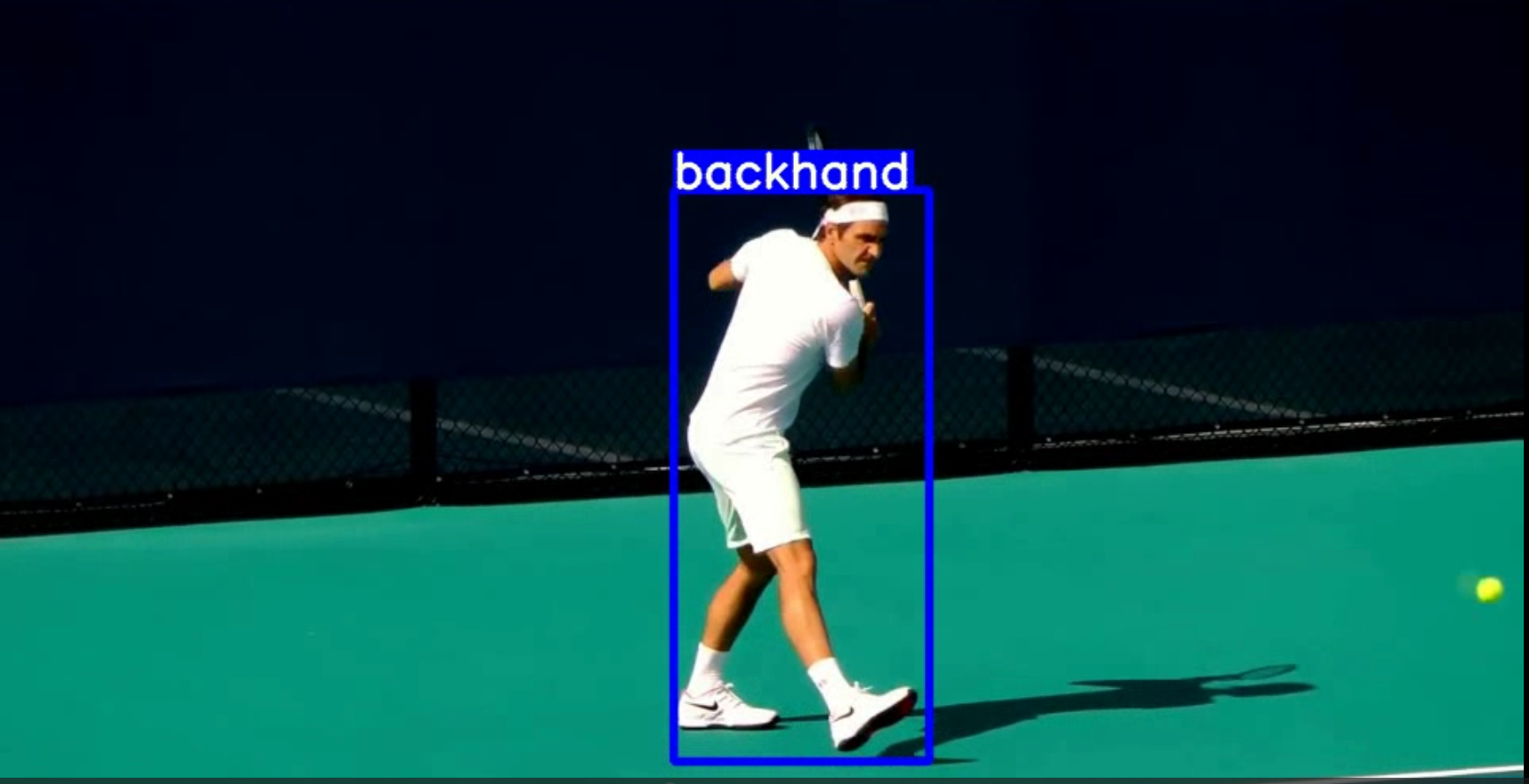


Figura 7.3: Cadru extras in urma analizei unui videoclip

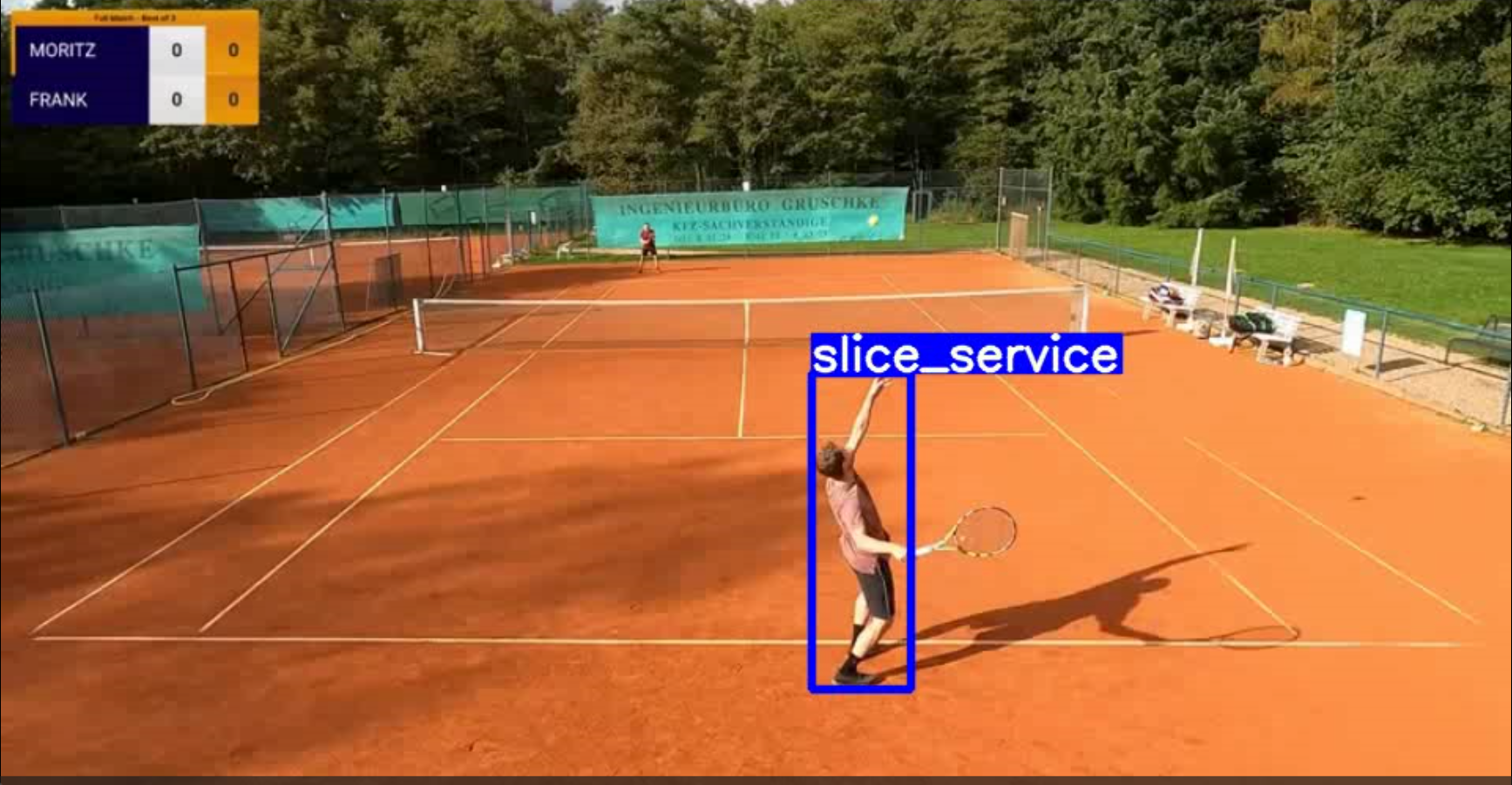


Figura 7.4: Cadru extras in urma analizei unui videoclip

Codul care se ocupa cu rularea logicii principale pentru a face predicții pe un videoclip este:

def make\_predictions(video\_path):

    hash\_of\_file = get\_hash\_of\_file(video\_path)

    hash\_file = search\_hash\_in\_database(hash\_of\_file)

    name\_of\_output = NAME\_OF\_OUTPUT

    if hash\_file != False :

        if not(os.path.exists(hash\_file)):

            #if the file path is no longer existing we need to delete this from the database entry

            delete\_entry\_database(hash\_of\_file)

        else:

            print(f"It is found at {hash\_file}")

            STATUS\_LABEL['text'] = f"The file if found at {hash\_file}"

            SELECT\_BUTTON["state"] = "normal"

            return False

    outputs = make\_predictions\_video(video\_path, name\_of\_output)

    fill\_outputs\_value(outputs)

    name\_of\_output\_video = create\_output\_name\_of\_video(video\_path)

    STATUS\_LABEL['text'] = "Creating the zip file"

    make\_csv\_and\_graph(name\_of\_output\_video)

    path\_to\_video = os.path.abspath(f"saved\_videos/{name\_of\_output\_video}.zip")

    insert\_value(hash\_of\_file, path\_to\_video)

    STATUS\_LABEL['text'] = f"Finished, zip file at location {path\_to\_video}"

    SELECT\_BUTTON["state"] = "normal"

    os.startfile(path\_to\_video)

    return True

Funcția primește ca și argument calea către videoclipul care trebuie analizat, fiind oferit de selecția utilizatorului din interfața grafică. In primul pas variabilei hash\_of\_file îi este atribuită semnătura digitala a videoclipului. Funcția “search\_hash\_in\_database(hash)” va căuta in baza de date dacă hash-ul exista, in caza afirmativ returnând locația fișierului in care sunt ținute rezultatele, iar în cazul contrar va returna valoarea fals. Variabila “name\_of\_output” conține calea către folderul temporar in care vor fi stocate rezultatele. În cazul in care semnătura apare in baza de date, programul verifica daca fișierul cu rezultate mai exista. In cazul in care fișierul exista, funcția va returna fals, iar utilizatorului i se va afișa faptul ca fișierul cu rezultate exista, și calea către acesta. In cazul in care exista o intrare in baza de date, dar fișierul nu mai exista, din baza de date va fi ștearsă intrarea căruia ii corespunde semnătura pentru a reanaliza videoclipul. Variabila outputs, va conține valorile prezise pentru fiecare secvență din videoclip. Funcția „fill\_outputs\_value(outputs)” are scopul de a aduna într-o lista declarata global, fiecare tip de lovitura executat, statistica care va fi folosita pentru generarea unui grafic și a unui fișier csv. Funcția „create\_output\_name\_of\_video(video\_path)” adaugă numelui videoclipului data la care analiza a fost efectuata pe acesta. Funcția „make\_csv\_and\_graph(name\_of\_output\_video)” creează fișierele de statistici. La final este inserata locația fișierului .zip generat in baza de date și utilizatorului i se va deschide fișierul cu statistici.

### 7.2.4 Generarea fișierelor de statistici

La începutul programului, va fi verificat daca există folderele denumite „saved-videos” și „.tmp”. Acestea vor fi folosite pentru scrierea rezultatelor. In folderul „.tmp” vor fi puse rezultatele înainte să fie arhivate, după care vor fi șterse din acest folder. Arhiva cu rezultate va fi salvată in folderul „saved-videos”, formatul numelui dat va fi dat de numele clipului original căruia i se va adaugă data la care detecția a fost făcută. Fișierul arhivat va conține 3 alte fișiere.

O imagine in formatul „.png” care va conține un grafic cu numărul de lovituri din fiecare clasa, astfel facilitând vizualizarea statisticilor pentru un utilizator. Un fișier csv care va conține numărul de lovituri din fiecare clasa, oferit cu scopul in care utilizatorul dorește să prelucreze aceste informații într-o maniera specifica. Ultimul fișier este videoclipul generat, in care fiecare lovitura este adnotata.. In figura 7.5 este prezentat graficul generat pentru un fișier aleator.

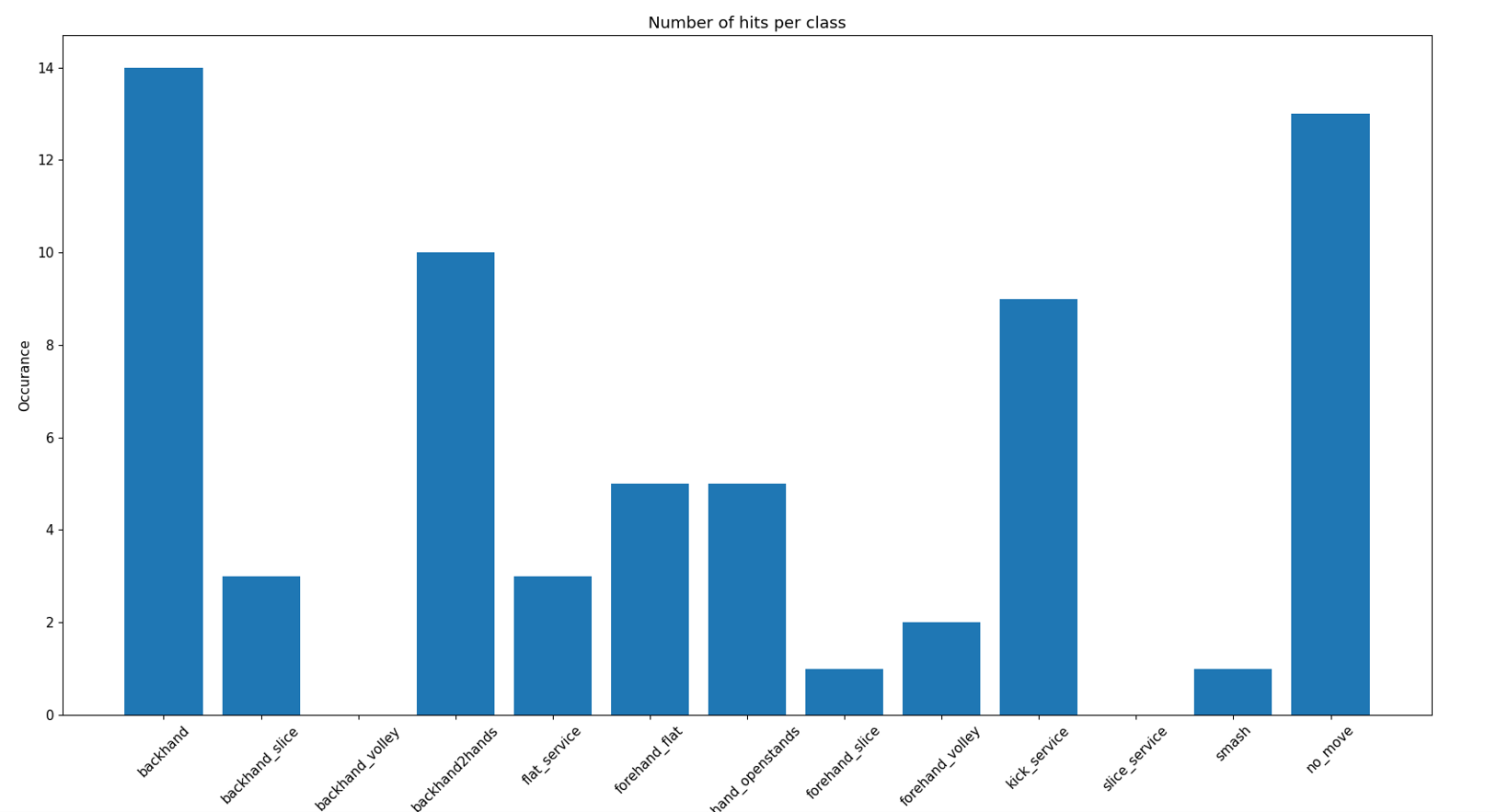


Figura 7.5: Grafic pe care aplicația îl oferă la finalul analizei unui videoclip

# Concluzii și dezvoltări ulterioare

Recunoașterea mișcărilor dintr-un meci de tenis este o componenta esențială in analiza și înțelegerea jocului. Această tehnologie utilizata permite identificarea și urmărirea mișcărilor jucătorilor efectuate în cadrul unui videoclip. În aceasta lucrare am abordat tema de a recunoașterea loviturii efectuate de un jucător. Utilizând o rețea de YOLO preantrenată care returnează coordonatele articulațiilor unui jucător dintr-un cadru, acestea fiind trimise unei rețele de LSTM, am putut oferi o soluție acestui tip de problema.

Pe setul de date THETIS care conține videoclipuri cu jucători care performează 12 tipuri diferite de mișcări, acuratețea generata de analiza completa a videoclipurilor (denumita in lucrare „metrica2”) fiind de 0.93, o îmbunătățire față de celelalte tehnici prezentate in [Vinyes et al., 2017] (0.47), [Gourgari et al., 2013] (0.60), [Vainstein et al., 2014] (0.86).

In cadrul acestei lucrări scopul a fost de a maximiza metrica de acuratețe care maximizează predicția pentru un număr de cadre predefinit pentru ca acest model să poate fi utilizat pe videoclipuri de tenis pentru a oferi predicții pentru fiecare lovitura. Pe setul de date THETIS pentru secvențe din clipuri care au 23 de cadre, acuratețea a fost de 0.76, aceasta metrica purtând numele in lucrare de „metrica1”.

Dezavantajul acestui set de date, este ca jucătorii nu executa aceste mișcări in prezenta unei mingi, producându-se erori pentru mișcările care sunt foarte asemănătoare in lipsa unei mingi. De exemplu „smash” și „smash service”, „smash service” fiind un serviciu, iar „smash” reprezentând o lovitura.

Alcătuirea unui set de date care conține și poziția mingii poate aduce îmbunătățiri in acuratețea modelului cat și eliminarea clasificării fals pozitivelor care se întâmpla in momentul in care un videoclip este analizat (in cadrul unui meci de tenis algoritmul detectează mișcări chiar și atunci când jucătorul nu executa una, deoarece modelul nu are informații despre poziția mingii).

In cadrul acestui proiect, modelul a fost înglobat într-o aplicație grafica pentru ca un utilizator să poată să își analizeze un videoclip cu ușurința. Întrucât operația de analiza este costisitoare din punct de vedere computațional, a fost adusa o optimizare pentru a evita analiza multipla a aceleași videoclip. Utilizatorul primește ca și rezultat un fișier arhivat care conține videoclipul adnotat, un fișier csv care conține numărul de mișcări din fiecare clasa și o poza cu un grafic cu aceste numere.

# Bibliografie

[Espn, 2023] URL: <https://www.espn.com/tennis/story/_/id/30877297/hawk-eye-live-gains-more-support-australian-open> (accesat la data 17.04.2023)

[Wang et al., 2022] Wang, Chien-Yao, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors." *arXiv preprint arXiv:2207.02696* (2022).

[Maji et al., 2022] Maji, Debapriya, Soyeb Nagori, Manu Mathew, and Deepak Poddar. "YOLO-Pose: Enhancing YOLO for Multi Person Pose Estimation Using Object Keypoint Similarity Loss." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2637-2646. 2022.

[Rosenblatt et al., 1958] Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65, no. 6 (1958): 386.

[University of Cincinnati, 2023] URL: <http://uc-r.github.io/feedforward_DNN> (accesat la data 13.06.2023)

[Glorot and Xavier, 2010] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249-256. JMLR Workshop and Conference Proceedings, 2010.

[Staudemeyer et al., 2019.] Staudemeyer, Ralf C., and Eric Rothstein Morris. "Understanding LSTM--a tutorial into long short-term memory recurrent neural networks." *arXiv preprint arXiv:1909.09586* (2019).

[Graves et al., 2013] Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." In *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 6645-6649. Ieee, 2013.

[Zhang et al., 2019] Zhang, Jingzhao, Tianxing He, Suvrit Sra, and Ali Jadbabaie. "Why gradient clipping accelerates training: A theoretical justification for adaptivity." *arXiv preprint arXiv:1905.11881* (2019).

[colah.github.io, 2023] URL: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> (accesat la data 17.04.2023)

[Wang et al., 2013] Wang, Heng, and Cordelia Schmid. "Action recognition with improved trajectories." In *Proceedings of the IEEE international conference on computer vision*, pp. 3551-3558. 2013.

[Varol et al., 2017] Varol, G., Laptev, I. and Schmid, C., 2017. Long-term temporal convolutions for action recognition. *IEEE transactions on pattern analysis and machine intelligence*, *40*(6), pp.1510-1517.

[Luvizon et al., 2018] Luvizon, Diogo C., David Picard, and Hedi Tabia. "2d/3d pose estimation and action recognition using multitask deep learning." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5137-5146. 2018.

[Gourgari et al., 2013] Gourgari, Sofia, Georgios Goudelis, Konstantinos Karpouzis, and Stefanos Kollias. "Thetis: Three dimensional tennis shots a human action dataset." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 676-681. 2013.

[Vinyes et al., 2017] Vinyes Mora, Silvia, and William J. Knottenbelt. "Deep learning for domain-specific action recognition in tennis." In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 114-122. 2017.

[Vainstein et al., 2014] Vainstein, Jonathan, José F. Manera, Pablo Negri, Claudio Delrieux, and Ana Maguitman. "Modeling video activity with dynamic phrases and its application to action recognition in tennis videos." In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications: 19th Iberoamerican Congress, CIARP 2014, Puerto Vallarta, Mexico, November 2-5, 2014. Proceedings 19*, pp. 909-916. Springer International Publishing, 2014.

[Paszke et al., 2018] Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019).

[wikipedia.com, 2023] URL: <https://en.wikipedia.org/wiki/OpenCV> (accesat la data 18.04.2023)

[github.com/yolov7, 2023] URL: <https://github.com/WongKinYiu/yolov7> (accesat la data 18.04.2023)

[Srivastava et al., 2014] Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15, no. 1 (2014): 1929-1958.

[Huang et al., 2019] Huang, Yu-Chuan, I-No Liao, Ching-Hsuan Chen, Tsì-Uí İk, and Wen-Chih Peng. "TrackNet: A deep learning network for tracking high-speed and tiny objects in sports applications." In *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1-8. IEEE, 2019.