

Sign Language Recognition Application

Autor: Achim Rareş, 331AB

An:2023- 2024

Table of Contents

Introducere:	3
Prezentarea pe scurt:	3
Prezentare tehnică:	4
Hands Media Pipe și recunoaștere mainii / palme	4
Random Forest Classifier - Clasificatorul pentru ASL.....	6
OpenCV Python - biblioteca pentru tratarea imaginilor și fișierelor video.....	7
Pickle - Data Serialization	7
RFC (Random Forest Classifier) vs MLP (Multilayer Perceptron)	7
Prezentare utilizare	8
Pașii de utilizare:.....	8
Concluzii.....	10
Bibliografie.....	Error! Bookmark not defined.

Introducere:

"Sign Language Recognition" este o aplicație de machine learning care își propune să spargă granița dintre persoanele care cunosc limbajul semnelor (o să fie referit mai departe ca ASL = American Sign Language [7], întrucât semnele folosite în cadrul aplicației fac parte din acest alfabet) și cele care nu îl cunosc. Tehnologia de învățare automată pentru a recunoaște și a interpreta limbajul semnelor, transformându-l în text scris, a fost antrenată să învețe și să prezică în mod corect și valid folosind diferite seturi de date care au fost preluate cu ajutorul web-cam-ului. (camera laptopului)

- **Tema:** Limbajul semnelor este o formă esențială de comunicare pentru multe persoane, dar este adesea înțeles doar de cei care îl folosesc regulat. Aplicația noastră își propune să reducă această barieră, permitând oricui să înțeleagă limbajul semnelor prin intermediul unei interfețe (format video).
- **Prezentarea obiectivelor:** Obiectivul principal al aplicației "Sign Language Recognition" este de a face limbajul semnelor accesibil unui public mai larg. Prin utilizarea tehnologiei de învățare automată, aplicația poate recunoaște și interpreta semnele, permitând utilizatorilor să comunice mai eficient cu cei care folosesc limbajul semnelor.

De menționat este faptul că limbajul semnelor conține 4500+ de semne și simboluri distincte, astfel că folosirea acestuia pentru a crea un mediu optim de comunicare între persoane este dificil de implementat în stadiul actual întrucât asta ar echivala la captura și colectarea pentru fiecare semn un set de date suficient de mare, lucru care ar putea reprezenta obiectul unui alt proiect de nivel mai complex.

Prezentarea pe scurt:

Proiectul a fost realizat în limbajul Python folosind mai multe modele de machine learning deja antrenate, testate și deployed pentru o gama foarte largă de aplicații. Două dintre aceste modele fac parte din librăria Media Pipe Hands [2] și sunt folosite pentru a identifica atât mâinile din cadrul imaginii dar și pentru a desena, localiza și accesa obiectele definite de aceasta librărie numite *landmarks*. Detalii mai multe o să fie prezentate în secțiunea următoare despre cum funcționează aceste modele precum și cum au fost acestea folosite în aplicația mea. Al treilea model folosit este modelul de tip Random Forest Classifier din librăria sklearn [1] iar motivul pentru care a fost folosit este pentru a implementa funcția de recunoaștere (clasificare) a semnelor din ASL. De ce a fost ales acest model în defavoarea altor modele de învățare automată o să fie prezentat de asemenea în secțiunea următoare.

O alta bibliotecă importantă folosită în acest proiect este cv2, o librărie extrem de puternică pentru procesarea imaginilor și a videoclipurilor. Aceasta a fost folosită pentru a captura setul de date, a le prelucra și a folosi modelul cu scopul de a prezice semnele prezentate pe ecran.

Inspirație pentru acest proiect reprezintă proiectul open-source “Sign Language Detector” care este de asemenea atașat în bibliografie dezvoltat de @computervisioneng (repository-ul de pe github) [3]. În cadrul aplicației mele există inspirații ce țin de antrenarea modelului, colectarea imaginilor și crearea de dataset împreună cu utilizarea modelului. Un feature reprezentativ adăugat de mine este colectarea semnelor recunoscute de către model într-un fișier .txt în cadrul aplicației cu scopul de a încuraja folosirea acestei aplicații pentru o comunicare mai bună între persoanele care cunosc acest limbaj și cei care nu. De asemenea, anumite optimizări care țin de antrenarea modelului, alegerea tipului corect de model de învățare automată, folosirea ambelor mâini în timpul utilizării aplicației, afișarea unui mesaj corespunzător atunci când rezultatul prezicerii este *Inconclusive* și adăugarea de suport infografic la colectarea datelor pentru antrenare sunt feature-uri ce contribuie la o mai bună experiență a utilizatorului și a persoanei care creează modelul.

Prezentare tehnică:

Datorită faptului ca este o gama destul de variată de tehnologii, biblioteci modele de machine learning și instrumente care au ajutat la dezvoltarea acestei aplicații, am decis să împart aceasta secțiune în mai multe parti.

Hands Media Pipe și recunoașterea mâinii / palmei

Pentru a putea înțelege mai bine cum a fost folosită aceasta librărie în cadrul proiectului, este recomandat să se înțeleagă cum funcționează acest pachet de modele și pentru ce este recomandat să fie folosit. Pentru început, acestea folosesc în spate 2 modele separate care optimizează detecția mâinii. Primul model se ocupă cu identificarea palmei în imaginea inițială iar al doilea model se ocupă cu identificarea elementelor de tip *landmark*.

Modelul a fost antrenat pe aproximativ 30.000 de imagini din lumea reală, precum și pe mai multe modele sintetice de mâini redate pe diverse fundaluri. Modelul de detectare a palmelor localizează mâinile în cadrul imaginii de intrare, iar modelul de detectare a reperelor mâinii identifică reperele specifice ale mâinii pe imaginea mâinii decupate definită de modelul de detectare a palmelor. Deoarece rularea modelului de detectare a palmelor este foarte heavy când vine vorba de timp, **Hand Landmarker** (modelul de detecție a reperelor) folosește caseta de delimitare definită de modelul de repere ale palmelor pentru a localiza regiunea mâinilor pentru cadrele ulterioare. Hand Landmarker declanșează modelul de detectare a palmelor doar



dacă acesta nu mai identifică prezența mâinilor sau nu reușește să urmărească mâinile în cadrul. Acest lucru reduce semnificativ latenta din punct de vedere al timpului și al performanței.

Mai sus sunt atașate cele 21 de tipuri de *landmarks* extrase de Hand Landmarker și semnificația lor în contextul fizionomiei mâinii. Coordonatele acestora sunt folosite ca date de intrare pentru fiecare dintre semnele folosite în aplicație.

În cadrul aplicației mele, acest model este utilizat atât pentru a extrage landmark-urile din pozele extrase și a stoca cele 21 de repere sub forma de coordonate x și y dar și pentru a afișa pe ecran aceste repere atât la extragerea setului de date dar și la testarea modelului adică la rularea aplicației propriu zise de recunoaștere a semnelor, cu scopul de a oferi o experiență cât mai vizuală

Mai jos este prezentat modul cum cele 2 modele au fost folosite pentru a crea setul de date și pentru a reprezenta pe ecran informațiile dorite. Au fost utilizate cele 2 modele, cu mențiunea că este setat *static_image_mode=True* pentru a preciza modelului faptul că imaginile trebuie considerate de sine stătătoare.



```
mp_hands_service = mp.solutions.hands
mp_drawing_service = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
hands_processor =
mp_hands_service.Hands(static_image_mode=True,
min_detection_confidence=0.3)
```

În cea de a 2 poza este prezentat modul cum funcția *draw_landmarks* desenează reperele și conexiunile între acestea pe imaginea dată, permitând vizualizarea reperelor detectate de model. La final, coordonatele x și y sunt normalizate și ulterior adăugate în setul de date care urmează să fie transmis modelului pentru recunoașterea semnelor.



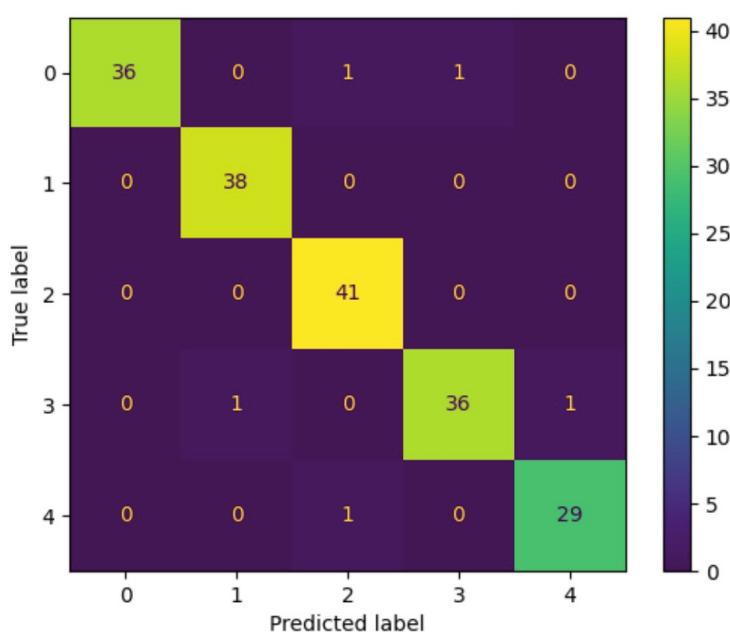
```
for hand_landmarks in results.multi_hand_landmarks:
    mp_drawing_service.draw_landmarks(
        img_rgb,
        hand_landmarks,
        mp_hands_service.HAND_CONNECTIONS,
        mp_drawing_styles.get_default_hand_landmarks_style(),
        mp_drawing_styles.get_default_hand_connections_style()
    )

    min_y = min(landmark.y for landmark in hand_landmarks.landmark)
    min_x = min(landmark.x for landmark in hand_landmarks.landmark)

    data_aux = [(landmark.y - min_y, landmark.x - min_x) for landmark in hand_landmarks.landmark]
    data.append(data_aux)
```

Random Forest Classifier - Clasificatorul pentru ASL

Așa cum am precizat mai sus, clasificatorul RFC a fost ales pentru a facilita recunoașterea semnelor ce fac parte din alfabetul ASL. Principalul motiv pentru care acesta a fost ales este rezultatele foarte bune atunci când vine vorba de fenomenul numit *overfitting*. Acesta este cauzat de mai multe aspecte, principalul fiind numărul oarecum mediu de set de date de antrenare, aspect pe care RFC îl ia în calcul și chiar excedează la acest capitol. Astfel, **Random Forest Classifier** este mai robust la overfitting comparativ cu un model de rețea neuronală. Acest lucru se datorează faptului că RFC construiește mai mulți arbori de decizie și face predicții prin votarea majorității, ceea ce ajută la reducerea overfitting-ului.



Overfitting-ul [9] este un concept din domeniul învățării automate și se referă la situația în care un model de învățare este prea complex și învăță prea bine datele de antrenament, la punctul în care începe să "memoreze" datele, în loc să învețe să generalizeze din ele. Acest lucru se poate întâmpla la seturi de date de dimensiuni reduse aşa cum este cazul aplicației noastre. Modelul performând foarte bine pe datele de antrenament însă rezultate pentru setul de test nefiind la fel de satisfăcător. Astfel, modelul de RFC dispune de anumite opțiuni de configurare pentru a

combate fenomenul de overfitting cum ar fi definirea numărului de estimatori și adâncimea maxima a arborilor de decizie, opțiuni ce sunt oferite estimatorului pentru ca el să aleagă cea mai bună combinație de valori pentru setul de date oferit. O consecință și în același timp o dovadă a lipsei overfitting-ului în prezicerile modelului sunt valorile scăzute sau chiar nule ale termenilor de **False Positive** și **False Negative** din matricea de covarianță al datelor de test.

De asemenea, modelul RFC este cel mai ușor de folosit atunci când vine vorba de prelucrare a datelor. Folosind modelele de la Hands Media Pipe și extrăgând coordonatele folosind librăria CV2 (care va fi descrisă anterior mai jos), coordonatele pot fi direct folosite pentru a antrena modelul.

OpenCV Python - biblioteca pentru tratarea imaginilor și fișierelor video

CV2 este un modul din biblioteca OpenCV (Open Source Computer Vision Library) [4], care este o bibliotecă de procesare a imaginilor și a videoclipurilor în timp real. Este utilizată în principal pentru detectarea și recunoașterea obiectelor, recunoașterea facială, analiza imaginilor, extragerea caracteristicilor, etc. În contextul aplicației, cv2 este utilizat în mai multe moduri, principalul motiv pentru care a fost utilizat este pentru a captura frame-ul din videoul oferit de webcam și pentru a putea salva anumite aspecte ale imaginii. Două exemple de funcții care au fost folosite din aceasta librărie sunt:

- `cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)`
- `cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)`

Convertirea de culori sub formatul BGR -> RGB a fost deosebit importantă întrucât modelul de recunoaștere al palmelor din biblioteca Hands Media Pipe lucrează cel mai bine cu imagini cu pixel de tip rgb în loc de bgr.

Pickle - Data Serialization

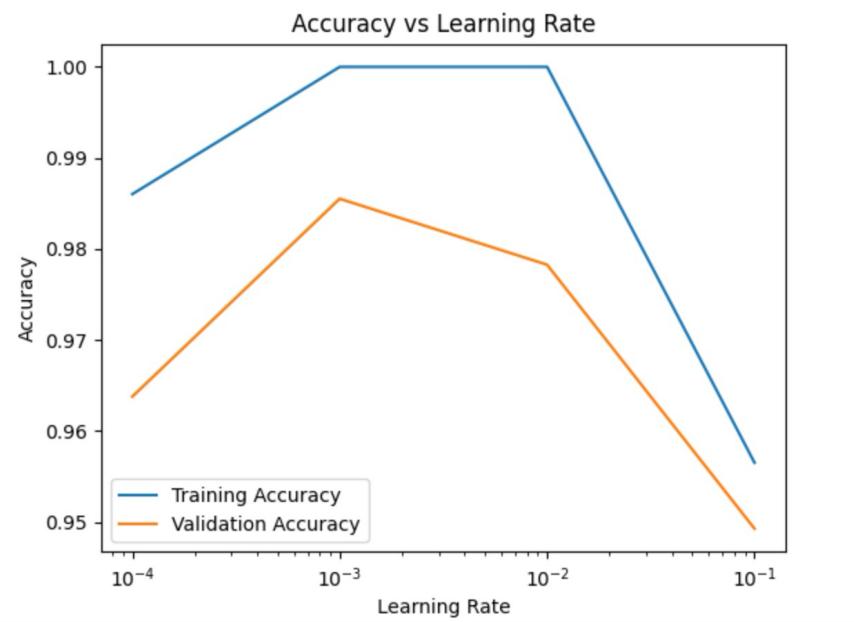
Pickle [5] este un modul din Python care implementează protocoale binare pentru serializarea și deserializarea unei structuri de obiecte Python. În cadrul aplicației mele, s-au folosit amândouă sensurile de prelucrare a datelor cu ajutorul acestei librării: *Pickling* și *Unpickling*. Astfel ca, "Pickling" este termenul folosit pentru a descrie procesul de serializare a unui obiect Python, iar "unpickling" este termenul opus, adică procesul de deserializare a datelor pentru a obține înapoi un obiect Python. Serializarea se referă la procesul de transformare a obiectelor din memorie într-un format care poate fi stocat pe disc sau transmis peste rețea. Acest format păstrează informațiile necesare pentru a reconstrui obiectul într-o altă instanță de Python. În cazul aplicației mele, pickle a fost folosit pentru a salva setul de date după ce acesta a fost convertit dar și pentru a salva modelul după ce acesta a fost antrenat și testat.

RFC (Random Forest Classifier) vs MLP (Multilayer Perceptron)

Ultimul lucru pe care vreau să-l menționez la aceasta secțiune este modul cum a fost realizata alegerea modelului de învățare automată, mai exact, motivul pentru care nu a fost ales un model de neural networks și modul cum am realizat ca acest lucru ar fi în defavoarea aplicației. Aceasta alegere s-a simplificat la alegere unui learning rate (rate de învățare) pentru modelul de neural networks destinat prezicerii de semne.

Luând în calcul mai multe posibilități pentru coeficientul de learning rate al modelului, acesta a fost antrenat pentru fiecare folosind anumite caracteristici. De cele mai multe ori, pentru un model de neural network, în cazul nostru un Multilayer Perceptron (MLP) [6] este necesar studierea și alegerea variabilelor de reglare într-un mod deosebit de detaliat, lucru care este încercat, deși minimal, în acest caz. De asemenea, Rețelele neuronale, inclusiv MLP-urile, sunt predispuse la supra adaptare dacă nu sunt corect regularizate. Acest lucru înseamnă că ar putea funcționa bine pe datele de antrenament, dar slab pe datele nevăzute. O alt avantaj al RFC-ului este faptul ca funcționează adesea bine pe seturi de date cu multe caracteristici și un amestec de date numerice și categorice.

Mai jos este atașat o comparare directă în acuratețea din setul de antrenament și cea din setul de validare (test). Se poate observa că modelul de MLP nu performează foarte bine cu nicio valoarea a learning rate-ului încercat. Aceasta comparare s-a realizat cu ajutorul documentației oficiale din librărie de modele sci-kit learn [8]



Prezentare utilizare

Majoritatea aplicațiilor de învățare automată / prezicere a anumitor informații / recunoaștere a anumitor caracteristici tend să ducă lipsa de o interfață user-friendly cu utilizatorul. Așadar, am considerat ca este necesar în contextul acestui proiect o comunicare minimală cu utilizatorul pentru a putea crea o experiență plăcută.

```
Please enter the number corresponding to the function you want to run:  
1. Generate Images  
2. Create Dataset  
3. Train Model  
4. Run Model  
5. Exit
```

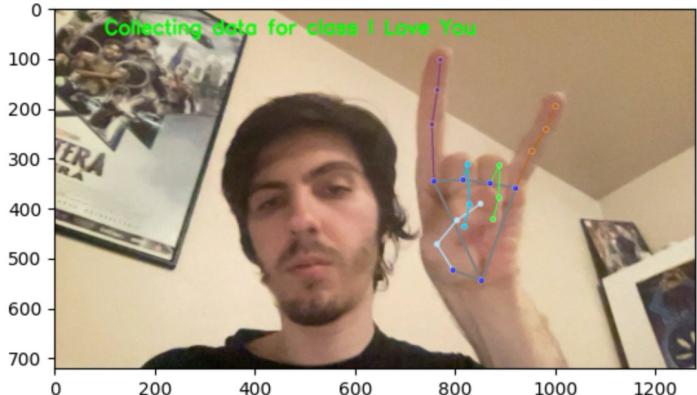
Astfel, cu ajutorul unui script și a liniei de comandă, aplicație interacționează cu utilizatorul printr-un set de instrucțiuni pe care acesta le poate adresa aplicației de prezicere.

Pașii de utilizare:

Observație: Primii 3 pași sunt obligatorii pentru utilizatorii care trebuie să colecteze datele, să le convertească și să antreneze modelul. În schimb, dacă modelul este deja antrenat din rulările anterioare ale aplicației, acești pași sunt opționali și se poate sări direct la pasul 4 unde se rulează aplicația propriu-zisă.

1. Colectarea de imagini

- Prima interacțiune pe care un utilizator îl are cu funcția de video al aplicației se află în acest pas. După selectarea acestei opțiuni, o să apară o fereastră în care se transmite video live ce conține imagini captureate de camera webcam a dispozitivului.
- Utilizatorului o să ii se informeze printr-un text aflat în partea de sus a ecranului instrucțiunile următoare. Acesta trebuie să apese tastă "Q" pentru a începe procesul de colectare al frame-urilor.
- În funcție de frame rate-ul fiecărui dispozitiv, aplicația o să colecteze 200 de frame-uri pentru fiecare semn în intervalul de timp necesar.



2. Creare dataset de imagini

- În urma colectării de imagini din pasul anterior, aceasta instrucțiune rulează cele 2 modele din libraria MediaPipe pentru a identifica landmark-urile pentru fiecare frame
- După ce sunt identificate, acestea sunt convertite în coordonate X și Y, sunt normalizate pentru a avea date la aceeași scală (coordonatele să fie în aceeași zona de proximitate)
- Obiectul cu coordonate este salvat într-un fișier pickle împreună cu label-urile respective pentru fiecare (numele semnelor din ASL)

3. Antrenare Model

- Antrenarea modelului este procesul critic și cel mai important al acestei aplicații. De cele mai multe ori acesta este un proces îndelungat însă datorită anumitor optimizări descrise mai sus, am reușit să scurtez acest timp la câteva secunde (intervalul de timp depinde de numărul de frame-uri și numărul de semne colectate)
- La finalul acestui proces, modelul este încărcat tot cu ajutorul librăriei pickle într-un fișier ce urmează să fie folosit în rularea efectiva a aplicației la pasul următor

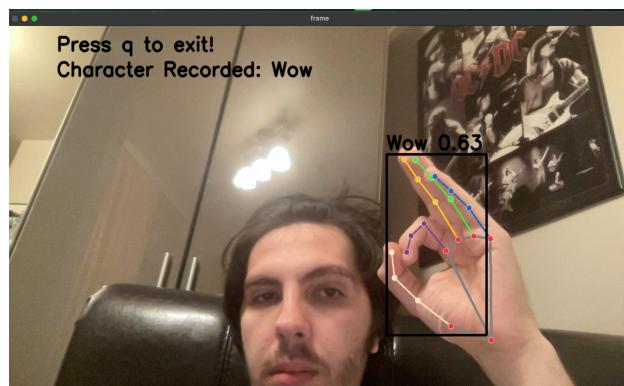
4. Rulare aplicație propriu-zisa

- Acest proces este poate cel mai important contact dintre utilizator și aplicație. Așa că am depus eforturi pentru ca experiența să fie una cat mai plăcută.
- Așadar, fără alte inputuri sau configurații suplimentare pe care utilizatorul ar fi nevoie să le facă, imediat după rularea comenzii, se deschide o fereastră, într-un mod similar cu cea de la pasul 1 și se afișează frame-urile captureate de video-cam.
- Atunci când modelul detectează o mana pe ecran, acesta returnează ca output label-ul (în cazul nostru semnul din ASL) asociat predicției acestuia împreună cu un coeficient care reprezintă valoarea de la 0 la 1 al încrederei prezicerii.

- De menționat este faptul ca în eventualitatea în care un simbol este incloncent, aplicație știe să recunoască acest lucru și să afișeze un mesaj corespunzător

5. Finalizare program

- La rularea acestei comenzi, programul se închide însă toate fișierele salvate precum datele colectate, setul de date și modelul antrenat sunt salvate astfel ca la următoarea rulare se poate trece direct la pasul 4.



Concluzii

- Acest proiect a avut ca obiectiv principal dezvoltarea unui sistem de recunoaștere a semnelor de mâină folosind tehnici de învățare automată. Utilitatea acestui sistem este evidentă în multe domenii, inclusiv în educație, unde poate fi folosit pentru a ajuta la învățarea limbajului semnelor ori pentru comunicare între persoane din diferite parti ale lumii, sau în tehnologie, unde poate fi folosit pentru a permite interacțiunea cu dispozitivele prin gesturi. Aplicațiile multimedia conținute în acest proiect se pot reduce la prelucrarea fluxului video venit din camera web, prelucrarea acestor imagini / frame-uri folosind diverse metode de detecția a mâinii și afișarea rezultatului modelului în timp real în concordanță cu poziția mâinii
- În ceea ce privește performanța, sistemul pare să funcționeze bine, deși nu avem date specifice despre acuratețea sau viteza de predicție. Cu toate acestea, faptul că sistemul poate face predicții în timp real sugerează că este suficient de rapid pentru multe aplicații.
- Din punctul de vedere al obiectivelor, sistemul pare să îndeplinească toate obiectivele propuse. Aceasta poate genera imagini, crea un set de date, antrena un model și face predicții pe baza acestuia. Impactul acestui sistem poate fi semnificativ, mai ales în domeniile în care comunicarea non-verbală este importantă. De asemenea, acest sistem poate fi folosit pentru a dezvolta noi metode de interacțiune cu tehnologia.
- Îmbunătățirile care se pot face la aceasta aplicație sunt numeroase astăzi ca există mai multe direcții posibile. În primul rând, performanța modelului ar putea fi îmbunătățită prin utilizarea unor tehnici de învățare automată mai avansate sau prin creșterea dimensiunii setului de date. În al doilea rând, sistemul ar putea fi extins pentru a recunoaște mai multe gesturi, pentru a funcționa în condiții de iluminare variabile sau pentru a recunoaște atât imagini dar și gesturi / secvențe video. În al treilea rând, interfața cu utilizatorul ar putea fi îmbunătățită pentru a face sistemul mai ușor de utilizat.

Bibliografie

- [1] „MediaPipe,” [Interaktiv]. Available:
https://developers.google.com/mediapipe/solutions/vision/hand_landmarker.
- [2] „Random Forest Classifier sklearn,” [Interaktiv]. Available: <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>.
- [3] computervisioneng, „Sign Language Detector Python,” 2023. [Interaktiv]. Available:
<https://github.com/computervisioneng/sign-language-detector-python>.
- [4] „opencv-python,” 31 December 2023. [Interaktiv]. Available:
<https://pypi.org/project/opencv-python/>.
- [5] „Pickle Python Documentation,” [Interaktiv]. Available:
<https://docs.python.org/3/library/pickle.html>.
- [6] „Multilayer Perceptron Wikipedia,” [Interaktiv]. Available:
https://en.wikipedia.org/wiki/Multilayer_perceptron.
- [7] „Wikipedia American Sign Language,” [Interaktiv]. Available:
https://en.wikipedia.org/wiki/American_Sign_Language.
- [8] „Multi-Layer Perceptron scikit-learn,” [Interaktiv]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html.
- [9] „Overfitting Wikipedia,” [Interaktiv]. Available: <https://en.wikipedia.org/wiki/Overfitting>.