Universitatea „Ștefan cel Mare” din Suceava

Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor

**Aplicație de recunoaștere și transpunere a gesturilor în chat.**

**1. Introducere**

**1. Introducere**

Recunoașterea gesturilor este un domeniu important al inteligenței artificiale, concentrându-se pe identificarea și interpretarea gesturilor umane. Aplicațiile sale variază de la interfețe om-calculator fără contact, controlul dispozitivelor electronice, limbajul semnelor și terapia fizică.

Acest raport analizează un cod Python pentru o aplicație de recunoaștere a gesturilor, descompunând funcționalitatea acestuia și evaluând eficacitatea sa. Scopul acestui proiect este de a dezvolta o aplicație care să permită utilizatorilor să comunice într-un chat prin intermediul gesturilor, oferind o alternativă intuitivă și accesibilă la introducerea textului tradițional.

**2. Analiza codului**

2.1. Biblioteci importate

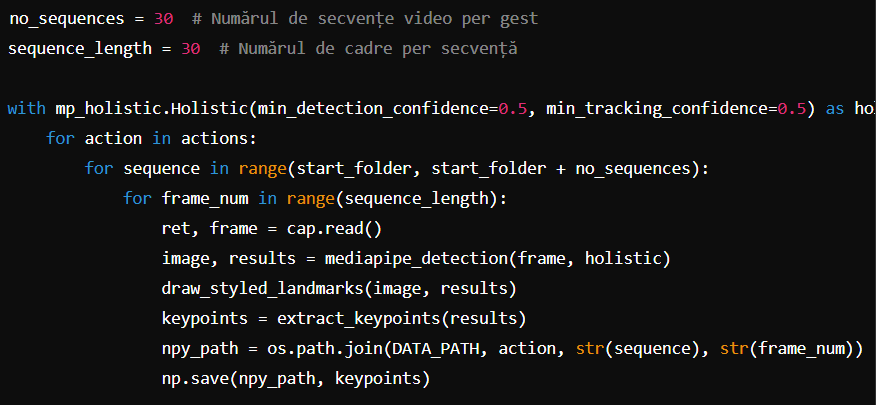
Codul folosește următoarele biblioteci Python:

* cv2 (OpenCV): Procesare imagini și videoclipuri.
* numpy (np): Calcul numeric și manipularea matricei.
* os: Gestionarea fișierelor și directoarelor.
* matplotlib.pyplot (plt): Vizualizarea datelor.
* time: Măsurarea timpului.
* mediapipe (mp): Framework Google pentru detectarea posturii, mâinii și irisului folosind modelul Holistic.
* tensorflow (tf): Platformă de învățare automată folosită pentru antrenarea și rularea rețelelor neuronale.

2.2. Modelul MediaPipe Holistic

Colectarea și antrenarea modelului

Procesul de colectare a datelor implică înregistrarea a 30 de secvențe video pentru fiecare dintre cele 7 gesturi definite. Fiecare secvență conține 30 de cadre, iar fiecare cadru este procesat pentru a extrage reperele cheie ale corpului uman utilizând modelul MediaPipe Holistic.



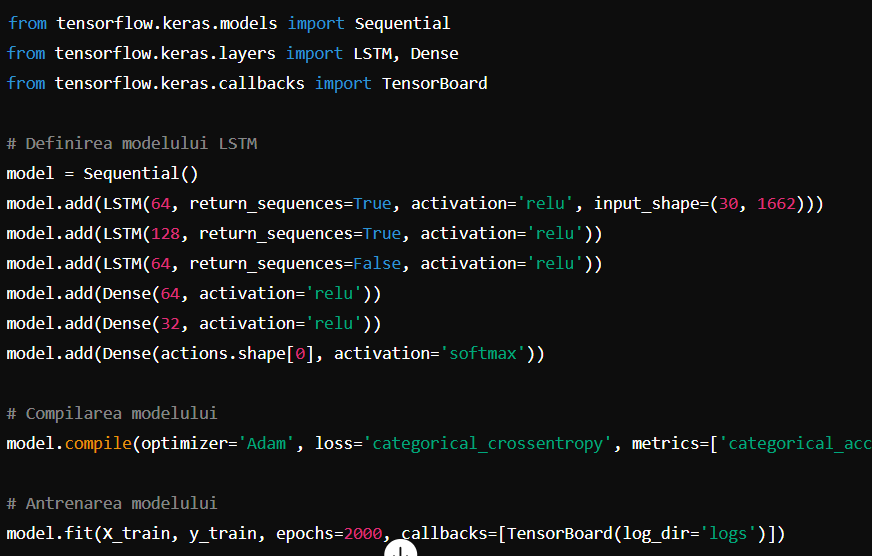
Explicație:

1. Inițializarea modelului MediaPipe Holistic.
2. Iterare prin fiecare gest, secvență și cadru video.
3. Citirea și procesarea fiecărui cadru pentru a extrage reperele cheie.
4. Salvarea reperelor cheie în fișiere NumPy (.npy).

Odată colectate, datele sunt preprocesate și folosite pentru a antrena rețeua neuronală LSTM, pentru recunoașterea gesturilor.

2.3. Antrenarea modelului cu TensorFlow

Pentru antrenarea modelului de recunoaștere a gesturilor, am utilizat TensorFlow. Codul implică încărcarea datelor, crearea și antrenarea unei rețele neuronale LSTM.

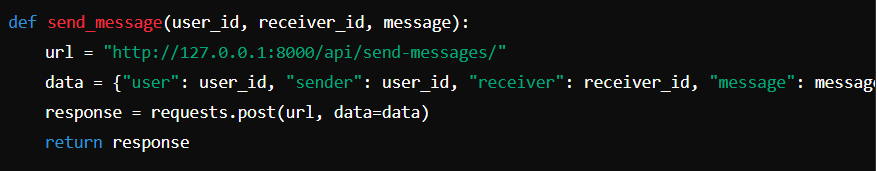


Explicație:

1. Definirea modelului LSTM: Modelul are trei straturi LSTM urmate de straturi Dense pentru clasificare.
2. Compilarea modelului: Folosind optimizatorul Adam și funcția de pierdere categorical\_crossentropy.
3. Antrenarea modelului: Antrenarea modelului cu datele de antrenament și utilizarea TensorBoard pentru monitorizarea antrenamentului.

2.4. API pentru recunoașterea gesturilor

După antrenarea modelului, se creează o legătură API pentru a permite comunicarea între aplicația de recunoaștere a gesturilor și backend-ul Django al aplicației de chat.



Explicație:

1. Funcția send\_message trimite o cerere POST către un endpoint API al backend-ului Django.
2. Datele includ ID-urile utilizatorilor și mesajul text extras din gest.
3. Backend-ul Django procesează și salvează mesajul în baza de date pentru a fi afișat în interfața de chat.

2.5. Captura video și detectarea reperelor cheie

* cap = cv2.VideoCapture(index): Accesarea camerei web.
* while cap.isOpened(): Buclă continuă pentru citirea cadrelor video.
* ret, frame = cap.read(): Citirea unui cadru video.
* image, results = mediapipe\_detection(frame, holistic): Aplicarea modelului MediaPipe pentru detectarea reperelor cheie.

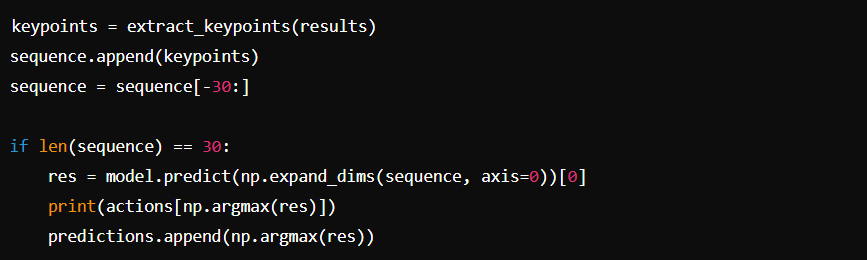
2.8. Încărcarea modelului pre-antrenat



Încarcă un model pre-antrenat pentru recunoașterea gesturilor.

2.9. Realizarea predicțiilor în timp real

Bucla principală continuă să citească cadre din captura video și să prezică gesturile în timp real utilizând modelul pre-antrenat.



**3. Implementare API**

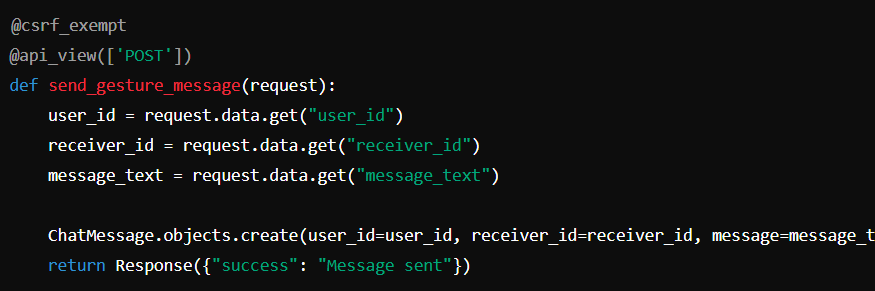
3.1. Configurația URL-urilor (Django - urls.py)



Definește legătura între URL și funcția send\_gesture\_message.

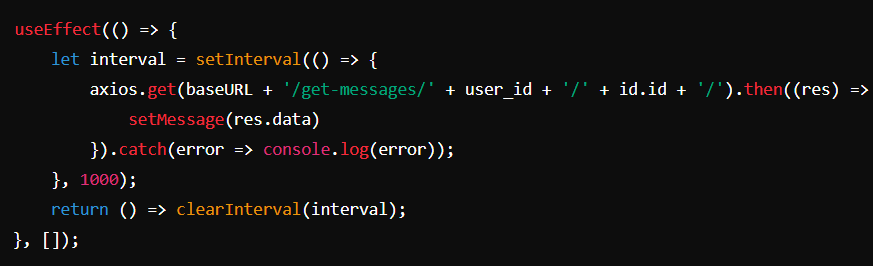
3.2. Endpoint-ul API (Django - views.py)

Funcția send\_gesture\_message procesează mesajele gestuale trimise de aplicația de recunoașt ere a gesturilor.



3.3. Încărcarea mesajelor în Frontend (React - MessageDetail.js)

Componenta React MessageDetail.js actualizează regulat chat-ul cu noile mesaje.



**4. Concluzii**

Aplicația analizată demonstrează o integrare eficientă între recunoașterea gesturilor, folosind MediaPipe și TensorFlow, și o aplicație de chat construită cu Django și React.js. Aceasta oferă o modalitate de comunicare, putând fi extinsă și îmbunătățită în viitor prin adăugarea de noi gesturi, îmbunătățirea acurateței recunoașterii și adăugarea de funcționalități suplimentare în aplicația de chat.

De asemenea, pot fi explorate și alte modele avansate de învățare profundă pentru a îmbunătăți performanța sistemului. Prin abordarea acestor aspecte, aplicația poate deveni un instrument bun pentru comunicarea nonverbală, oferind o experiență mai intuitivă și mai incluzivă pentru utilizatori.