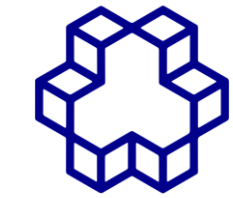
به نام خدا



دانشگاه صنعتی خواجه نصیرالدین طوسی

دانشکده برق

مبانی مکاترونیک

گزارش پروژه

آموزش مدل با مدیاپایپ و LSTM

سبحان سخایی

مبینا یوسفی مقدم

یزدان بیات

استاد : آقای دکتر مهدی دلربایی

خرداد ماه 1404

فهرست مطالب

چکیده

در این پروژه سامانه‌ای برای تشخیص حرکات دست مبتنی بر یادگیری عمیق طراحی و پیاده‌سازی شد. بدین منظور از مجموعه‌داده‌ی **Hand Gesture Detection System** شامل پنج ژست اصلی استفاده گردید. پس از انجام مراحل پیش‌پردازش، ویژگی‌های سه‌بعدی انگشتان دست با بهره‌گیری از کتابخانه‌ی **Mediapipe** استخراج شد و دنباله‌هایی متشکل از ۳۰ فریم متوالی به منظور آموزش مدل تهیه گردید. در ادامه، یک شبکه‌ی بازگشتی **LSTM** برای دسته‌بندی ژست‌ها آموزش داده شد که در داده‌های اعتبارسنجی دقتی در حدود ۹۷ تا ۹۸ درصد به دست آورد.

در بخش اینفرنس، ژست‌های آموزش‌دیده برای کنترل محتوای ویدئویی به کار گرفته شدند: حرکت دست به سمت چپ و راست برای جابه‌جایی در ویدئو، ژست «شست به بالا» و «شست به پایین» برای افزایش و کاهش صدا، و ژست مشت بسته برای توقف یا پخش ویدئو مورد استفاده قرار گرفت.

نتایج حاصل نشان می‌دهد که سامانه‌ی طراحی‌شده می‌تواند به عنوان یک رابط کاربری طبیعی، کارآمد و قابل اعتماد در کاربردهای چندرسانه‌ای مورد استفاده قرار گیرد.

.مقدمه

بیان مسئله

در سال‌های اخیر، با گسترش سامانه‌های هوشمند و افزایش سطح تعامل انسان با ماشین، ضرورت طراحی رابط‌های کاربری طبیعی (**Natural User Interfaces**) بیش از پیش مورد توجه قرار گرفته است. تعامل متداول با رایانه‌ها و دستگاه‌های دیجیتال عمدتاً از طریق ابزارهایی نظیر صفحه‌کلید، ماوس و ریموت کنترل انجام می‌شود. هرچند این ابزارها در بسیاری از کاربردها کارآمد بوده‌اند، اما محدودیت‌های مشخصی نیز به همراه دارند.

به عنوان نمونه، در هنگام تماشای یک ویدئو، تغییر حجم صدا یا جابه‌جایی در زمان پخش معمولاً مستلزم استفاده از دکمه‌های فیزیکی یا ابزارهای کنترلی جانبی است. این وابستگی به تجهیزات سخت‌افزاری در شرایطی که کاربر دسترسی سریع به ابزار کنترلی ندارد یا ترجیح می‌دهد از شیوه‌های غیرتماسی بهره گیرد، می‌تواند موجب کاهش کارایی و تجربه‌ی کاربری شود.

از سوی دیگر، ژست‌های دست به عنوان یکی از روش‌های طبیعی و شهودی ارتباط انسانی، ظرفیت قابل توجهی برای جایگزینی یا تکمیل ابزارهای ورودی سنتی دارند. بهره‌گیری از حرکات دست نه تنها نیازمند تماس فیزیکی نیست، بلکه می‌تواند تجربه‌ای روان‌تر، سریع‌تر و منعطف‌تر برای کاربر فراهم آورد.

با این حال، شناسایی دقیق و بلادرنگ ژست‌های دست همواره با چالش‌های متعددی همراه است. تغییر شرایط نوری محیط، تفاوت در ابعاد و فرم دست افراد، و همچنین تنوع حرکات از جمله عواملی هستند که فرآیند شناسایی را پیچیده و دشوار می‌سازند. یکی از راهکارهای مؤثر برای غلبه بر این چالش‌ها بهره‌گیری از روش‌های یادگیری عمیق است.

در این پروژه، با استفاده از کتابخانه‌ی **Mediapipe** موقعیت سه‌بعدی نقاط کلیدی دست استخراج شد و یک شبکه‌ی بازگشتی **LSTM** به منظور شناسایی توالی حرکات در طول زمان آموزش داده شد. مسئله‌ی اصلی تحقیق حاضر آن است که آیا می‌توان از ترکیب این روش‌ها برای طراحی سامانه‌ای بهره گرفت که قادر باشد ژست‌های پرکاربردی همچون «شست رو به بالا»، «شست رو به پایین»، «حرکت دست به چپ»، «حرکت دست به راست» و «مشت بسته» را با دقت بالا تشخیص دهد و امکان کنترل سامانه‌های چندرسانه‌ای را از طریق آن‌ها فراهم آورد.

این پروژه با هدف پاسخ‌گویی به مسئله‌ی مذکور طراحی و اجرا شد. در سامانه‌ی پیشنهادی، پس از آموزش مدل، ژست‌های دست شناسایی‌شده مستقیماً در یک کاربرد عملی به کار گرفته شدند: حرکت دست به سمت چپ و راست برای عقب و جلو بردن ویدئو، ژست «شست رو به بالا» و «شست رو به پایین» برای افزایش یا کاهش حجم صدا، و ژست «مشت بسته» برای توقف یا پخش ویدئو. به این ترتیب، سامانه‌ی ارائه‌شده نه تنها به عنوان یک الگوریتم تشخیص ژست عمل می‌کند، بلکه امکان تعامل غیرتماسی، طبیعی و کارآمد با سامانه‌های چندرسانه‌ای را نیز فراهم می‌سازد..

رویکرد پیشنهادی

داده‌های استفاده‌شده

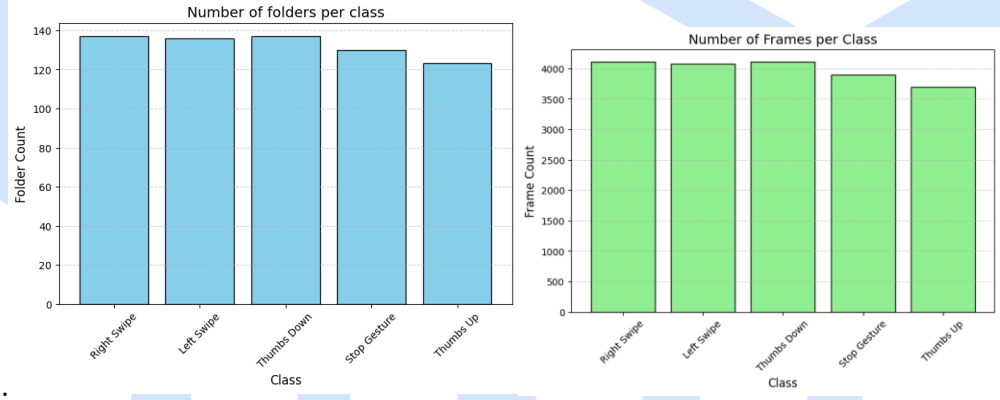
در این پروژه از **Hand Gesture Detection System dataset** استفاده شد که شامل ویدئوهایی از پنج gesture اصلی دست است. این gesture ها عبارت‌اند از: «انگشت شست رو به بالا»، «انگشت شست

رو به پایین»، «حرکت دست به چپ»، «حرکت دست به راست» و «علامت توقف». ساختار داده‌ها به‌صورت پوشه‌های مجزا برای هر class در دو بخش **training** و **validation** سازمان‌دهی شده بود.

همچنین برای هرclass ، الگوی نام‌گذاری پوشه‌ها به‌طور دقیق مشخص گردید تا فرآیند دسته‌بندی داده‌ها به‌درستی انجام گیرد.

به‌منظور بررسی کیفیت داده‌ها، شمارش تعداد پوشه‌ها و فریم‌های تصویری هر class انجام شد. نتایج نشان داد تعداد پوشه‌ها به‌ترتیب حدود ۱۲۳(**Thumbs Up**) ، ۱۳۶(**Left Swipe**) ، ۱۳۷(**Right Swipe**) ، ۱۳۰ (**Stop**) و ۱۳۷ (**Thumbs Down**) است. همچنین تعداد فریم‌ها برای کلاس‌ها به‌ترتیب حدود ۳۶۹۰، ۴۰۸۰، ۴۱۱۰، ۳۹۰۰ و ۴۱۱۰ برآورد شد.

برای آموزش مدل ترتیبی، طول sequence ورودی برابر با ۳۰ frame در نظر گرفته شد و داده‌ها با استفاده از sliding window به توالی‌های زمانی تقسیم گردید. در نهایت، شکل نهایی داده‌های **training** و **validation** به‌ترتیب (663, 30, 63) و (100, 30, 63) به دست آمد. در این ساختار، ۶۳ بعد ویژگی متناظر با مختصات سه‌بعدی ۲۱ keypoint اصلی دست است.



روش

در این پروژه برای پیاده‌سازی سامانه‌ی تشخیص gesture دست، چند گام اصلی دنبال شد،**datapreprocessing**، **feature extraction** با استفاده از **Mediapipe** ، ساخت توالی‌های زمانی، طراحی مدل یادگیری عمیق (**LSTM**) و در نهایت **training** و **evaluation** مدل. در ادامه، هر مرحله همراه با بخش‌هایی از کد اصلی مورد بررسی و توضیح قرار می‌گیرد.

1. استخراج نقاط کلیدی(Feature) دست با Mediapipe

در این مرحله از کتابخانه‌ی **Mediapipe** استفاده شد. این ابزار توانایی شناسایی دست و استخراج ۲۱ نقطه‌ی کلیدی سه‌بعدی از آن را دارد. هر نقطه شامل مختصات (x,y,z) است و در نتیجه هر فریم به یک بردار ۶۳ بعدی تبدیل می‌شود.

```
import cv2
import mediapipe as mp


# Initialize Mediapipe Hands
mp_hands = mp.solutions.hands
hands = mp_hands.Hands(
    max_num_hands=1,
    min_detection_confidence=0.7
)


def extract_landmarks(image):
    """
    Input: BGR image (from cv2.VideoCapture)
    Output: List of 63 values [x1,y1,z1, ..., x21,y21,z21]
            If no hand is detected: 63 zeros.
    """
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = hands.process(image_rgb)


    if results.multi_hand_landmarks:
        hand_lms = results.multi_hand_landmarks[0].landmark
        lm_list = []
        for lm in hand_lms:
            lm_list.extend([lm.x, lm.y, lm.z])
        return lm_list


    return [0.0] * 63
```

توضیح: در این تابع ابتدا تصویر به RGB تبدیل می‌شود و سپس به مدل Mediapipe داده می‌شود. اگر دست شناسایی شود، مختصات ۲۱ نقطه استخراج و در قالب یک بردار۶۳ بعدی بازگردانده می‌شوند. در غیر این صورت یک بردار صفر جایگزین می‌شود تا انسجام داده‌ها حفظ گردد.

2. ساخت توالی‌های زمانی

برای اینکه مدل بتواند پویایی حرکات دست را در طول زمان یاد بگیرد، هر ۳۰ فریم متوالی به عنوان یک توالی انتخاب شد. این کار با حرکت یک «پنجره‌ی لغزان» روی داده‌ها انجام شد.

```
IMPORT OS
IMPORT CV2
IMPORT NUMPY AS NP

SEQUENCE_LENGTH = 30

DEF LOAD_DATASET(BASE_PATH):
  SEQUENCES, LABELS = [], []

  FOR GESTURE_NAME, PATTERNS IN CLASS_PATTERNS.ITEMS():
    GESTURE_FOLDERS = [
      F FOR F IN OS.LISTDIR(BASE_PATH)
      IF ANY(P IN F FOR P IN PATTERNS)
    ]

    FOR FOLDER_NAME IN GESTURE_FOLDERS:
      FOLDER_PATH = OS.PATH.JOIN(BASE_PATH, FOLDER_NAME)
      FRAMES = SORTED(OS.LISTDIR(FOLDER_PATH))

      LANDMARKS_SEQ = []
      FOR F IN FRAMES:
        IMG = CV2.IMREAD(OS.PATH.JOIN(FOLDER_PATH, F))
```

```
            LANDMARKS = EXTRACT_LANDMARKS(IMG)
            LANDMARKS_SEQ.APPEND(LANDMARKS)

        FOR I IN RANGE(LEN(LANDMARKS_SEQ) - SEQUENCE_LENGTH + 1):
            SEQ = LANDMARKS_SEQ[I:I + SEQUENCE_LENGTH]
            SEQUENCES.APPEND(SEQ)
            LABELS.APPEND(CLASS_MAP[GESTURE_NAME])

    RETURN NP.ARRAY(SEQUENCES), NP.ARRAY(LABELS)
```

توضیح :این تابع پوشه‌های مربوط به هر ژست را می‌خواند، نقاط کلیدی فریم‌ها را استخراج می‌کند و سپس توالی‌های ۳۰فریمی می‌سازد. هر توالی یک نمونه‌ی آموزشی با برچسب مشخص است.

3. طراحی مدل LSTM

مدل طراحی‌شده یک شبکه‌ی بازگشتی **LSTM** است که برای یادگیری وابستگی‌های زمانی در داده‌ها بسیار مناسب است. ساختار شبکه شامل دو لایه LSTM با لایه‌های Dropout برای جلوگیری از بیش‌برازش و در نهایت یک لایه Dense برای دسته‌بندی پنج‌کلاسه است.

```
FROM TENSORFLOW.KERAS.MODELS IMPORT SEQUENTIAL
FROM TENSORFLOW.KERAS.LAYERS IMPORT LSTM, DENSE, DROPOUT

MODEL = SEQUENTIAL()

MODEL.ADD(LSTM(
  64, RETURN_SEQUENCES=TRUE,
  INPUT_SHAPE=(SEQUENCE_LENGTH, 63)
))
MODEL.ADD(DROPOUT(0.5))

MODEL.ADD(LSTM(64))
MODEL.ADD(DROPOUT(0.5))
```

```
model.add(Dense(64, activation='relu'))
model.add(Dense(NUM_CLASSES, activation='softmax'))


model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy']
)


model.summary()
```

توضیح:

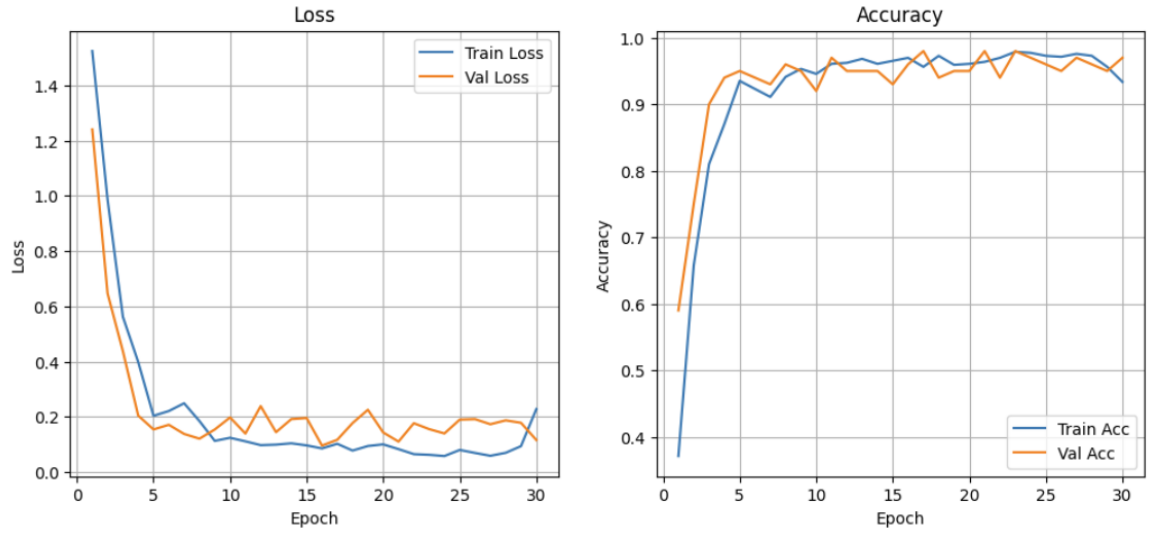
- لایه اول LSTM با ۶۴ نورون، توالی‌ها را پردازش کرده و الگوهای زمانی اولیه را استخراج می‌کند.
- لایه Dropout با نرخ ۰.۵ برای کاهش بیش‌برازش استفاده شد.
- لایه دوم LSTM با ۶۴ نورون ویژگی‌های پیچیده‌تر زمانی را یاد می‌گیرد.
- یک لایه Dense با ۶۴ نورون و تابع ReLU به‌عنوان نگاشت غیرخطی اضافه شد.
- در نهایت لایه خروجی با ۵ نورون و Softmax قرار گرفت که احتمال تعلق هر توالی به یکی از کلاس‌های ژست را محاسبه می‌کند.

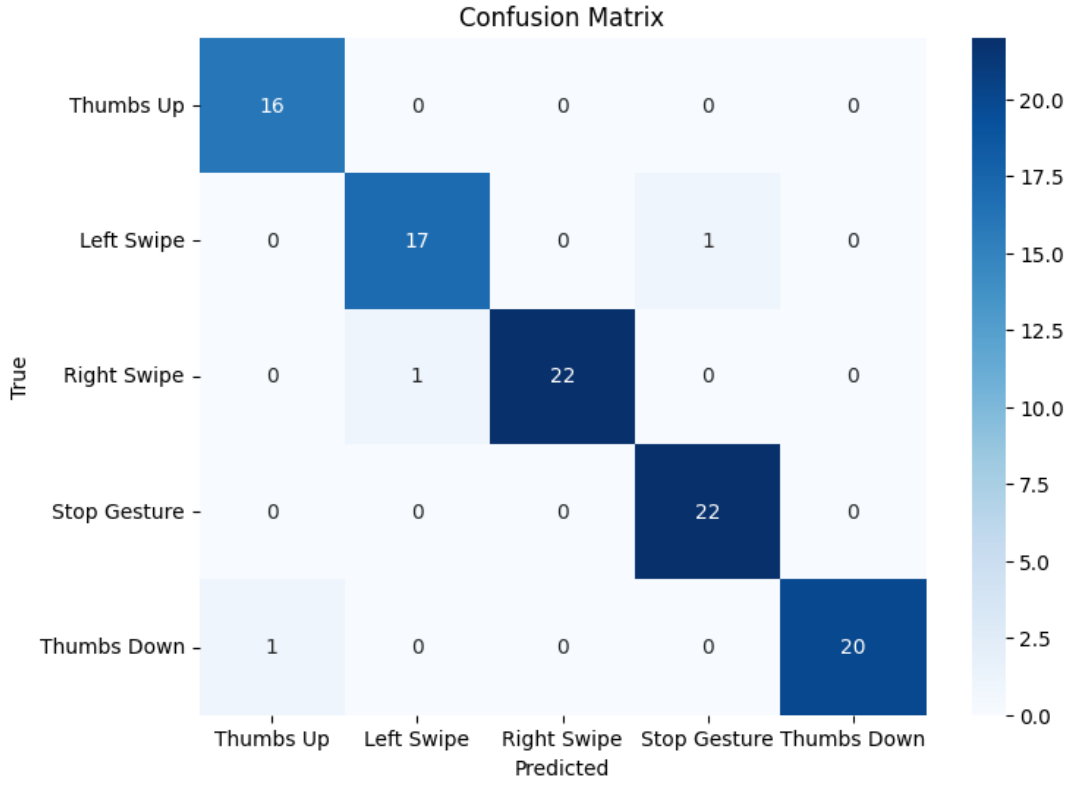
4. آموزش و ارزیابی مدل

مدل به‌مدت ۳۰ **Epoch** و با اندازه‌ی بچ ۳۲ روی داده‌های آموزشی آموزش داده شد. در طول آموزش، مقادیر دقت و خطا در هر دوره ثبت و برای ارزیابی عملکرد، مجموعه اعتبارسنجی مورد استفاده قرار گرفت.

```
HISTORY = MODEL.FIT (
    X_TRAIN, Y_TRAIN,
    EPOCHS=30,
    BATCH_SIZE=32,
    VALIDATION_DATA=(X_VAL, Y_VAL_CAT)
)
```

نتایج :مدل توانست در داده‌های آموزشی به دقت بالای ۹۷ درصد و در داده‌های اعتبارسنجی به حدود ۹۸ درصد دست یابد. نمودار تغییرات Loss و Accuracy نشان داد که مدل به‌خوبی همگرایی داشته است.





5. استفاده در بخش ایتفرنس و کاربرد عملی

پس از ذخیره‌ی مدل، از آن در بخش inference استفاده شد. داده‌ی ورودی (ویدئو یا دوربین) با همان روش مرحله‌ی preprocessing آماده‌سازی گردید و توالی‌های ۳۰ فریمی به مدل داده شد. خروجی مدل که معادل کلاس gesture بود، با بهره‌گیری از کتابخانه‌ی PyAutoGUI به فرمان کنترلی متناظر نگاشت شد.

- حرکت دست به چپ/راست ← عقب یا جلو بردن ویدیو
- شست به بالا/پایین ← افزایش یا کاهش صدا
- مشت بسته ← توقف یا پخش ویدیو

بحث و نتیجه‌گیری

نتایج حاصل از آموزش مدل نشان داد که به‌کارگیری ترکیب **Mediapipe** برای **feature extraction** و شبکه‌ی بازگشتی **LSTM** برای تحلیل توالی‌های زمانی، راهکاری کارآمد برای تشخیص حرکات دست محسوب می‌شود. مدل طراحی‌شده پس از ۳۰ دوره **training** به دقتی در حدود ۹۷ تا ۹۸ درصد در داده‌های **validation** دست یافت که بیانگر توانایی بالای آن در شناسایی صحیحgesture هاست. علاوه بر این، نمودارهای روند **loss** و **accuracy** نشان دادند که مدل به‌طور پایدار به همگرایی رسیده و توانسته است تعادل مناسبی میان یادگیری داده‌های آموزشی و تعمیم به داده‌های جدید برقرار سازد.

علاوه بر نتایج عددی، آزمایش عملی مدل در بخش **inference** اهمیت ویژه‌ای داشت. خروجی مدل به کمک کتابخانه‌ی **PyAutoGUI** به فرمان‌های کنترلی سیستم **mapped** شد و بدین ترتیب، حرکات دست کاربر به‌طور مستقیم برای کنترل پخش ویدیو به کار گرفته شدند. در این پیاده‌سازی، حرکت دست به چپ و راست برای عقب و جلو بردن ویدیو، ژست شست به بالا و پایین برای افزایش یا کاهش صدا و ژست مشت بسته برای توقف یا پخش ویدیو استفاده گردید. این آزمایش نشان داد که مدل آموزش‌دیده نه‌تنها در محیط آزمایشگاهی، بلکه در یک کاربرد واقعی نیز عملکرد مطلوبی دارد.

با این حال، پروژه با محدودیت‌هایی نیز مواجه است. یکی از چالش‌های اصلی، وابستگی سیستم به شرایط نوری و کیفیت تصویر است؛ زیرا شناسایی دقیق **landmark** های دست در محیط‌های با نور نامناسب یا پس‌زمینه‌ی شلوغ می‌تواند با خطا همراه شود. علاوه بر این، سیستم صرفاً برای پنج gesture از پیش‌تعریف‌شده آموزش داده شد و گسترش آن به مجموعه‌ی گسترده‌تری ازgesture ها نیازمند توسعه‌ی بیشتر **dataset** و به‌کارگیری مدل‌های پیچیده‌تر است.

در مجموع، نتایج این پروژه نشان داد که می‌توان با بهره‌گیری از روش‌های **deep learning** و **computer vision**، سامانه‌ای برای **natural user interface** مبتنی بر حرکات دست طراحی کرد که قادر است کنترل‌های متداول چندرسانه‌ای را بدون نیاز به تماس فیزیکی فراهم آورد. چنین سامانه‌ای می‌تواند زمینه‌ساز توسعه‌ی کاربردهای گسترده‌تری نظیر کنترل هوشمند دستگاه‌های خانگی، تعامل در محیط‌های **virtual reality** و حتی دستیارهای **robotics** باشد.

Inference(استنتاج) :

در بخش **inference**، تصویر از **webcam** دریافت و در هر فریم با استفاده از **Mediapipe Hands** مختصات ۲۱ **landmark** سه‌بعدی دست استخراج شد. این داده‌ها به بردارهای ۶۳ بعدی تبدیل و در یک **sliding window** با طول ۱۵ فریم ذخیره گردید. پس از تکمیل توالی، ورودی(63, 15, 1) به مدل **LSTM** داده شد و خروجی آن ـ پس از اعمال **confidence threshold** و مکانیزم‌های تثبیت ـ کلاس gesture را تعیین کرد. در نهایت، کلاس پیش‌بینی‌شده به کمک **PyAutoGUI** به فرمان‌های کنترلی ویدئو **mapped** شد؛ از جمله جابه‌جایی در ویدئو، تغییر صدا و پخش/توقف.

در ابتدای کد، پارامترهای اصلی تعیین می‌شوند: طول توالی(**SEQUENCE_LENGTH=15**) ، مسیر مدل ذخیره‌شده(**MODEL_PATH**) ، آستانه‌ی اطمینان(**CONF_THRESH=0.80**) ، پارامترهای تثبیت ژست توقف (**STABLE_N & COOLDOWN_SEC**) و شمارنده‌ی نبود دست (**ABSENT_RESET**).این مقادیر حساسیت سیستم را کنترل می‌کنند؛ به‌عنوان مثال، با افزایش **CONF_THRESH** سیستم محافظه‌کارتر شده و دیرتر فرمان صادر می‌کند. سپس مدل با **load_model** بارگذاری می‌شود و **Mediapipe Hands** با تنظیمات max_num_hands=1 و min_detection_confidence=0.7 مقداردهی می‌شود تا در هر فریم حداکثر یک دست با حداقل اطمینان ۰.۷ شناسایی گردد

.کلاس‌ها و نگاشت برچسب‌ها

آرایه‌ی class_names ترتیب کلاس‌های خروجی مدل را تعریف می‌کند و class_mapاندیس کلاس به نام آن را نگاشت می‌کند. لازم است این ترتیب با ترتیب آموزش مدل یکسان باشد تا پیش‌بینی‌ها درست تفسیر شوند.

استخراج لندمارک‌ها (Extract_Landmark)

در هر فریم، تصویر از BGR به RGB تبدیل و به Mediapipe داده می‌شود. اگر دست شناسایی شود، مختصات ۲۱ لندمارک سه‌بعدی (x, y, z) استخراج و به صورت یک بردار ۶۳ بعدی بازگردانده می‌شود. همچنین برای بازخورد بصری، اتصالات و نقاط دست روی فریم رسم می‌شود. در صورت عدم شناسایی دست، تابع None برمی‌گرداند تا جریان منطقی پایین‌دست بداند در این فریم پیش‌بینی انجام ندهد.

بافر زمانی توالی‌ها (deque) و منطق نبود دست

برای نگهداری آخرین ۱۵ فریم معتبر، یک **deque** با طول ثابت **(max len = SEQUENCE_LENGTH)** تعریف شد. در صورت عدم شناسایی دست در یک فریم، شمارنده‌ی **absent_frames** افزایش می‌یابد. اگر این شمارنده به مقدار **ABSENT_RESET** برسد، بافر توالی و متغیرهای حالت ریست می‌شوند تا داده‌های قدیمی **(stale windows)** باعث تولید فرمان اشتباه نشوند. این سازوکار از فعال‌سازی ناخواسته در شرایطی که دست برای مدتی از کادر خارج می‌شود جلوگیری می‌کند.

شرط آمادگی برای پیشبینی

فرآیند **prediction** تنها زمانی انجام می‌شود که در یک فریم، دست شناسایی و به بافر افزوده شده باشد و طول بافر دقیقاً به ۱۵ برسد. در این حالت، داده به فرم (1,T,63) ـ شامل یک نمونه، طول توالی T=15ویژگی ـ آماده شده و به **model.predict** داده می‌شود. خروجی مدل به‌صورت توزیع احتمال روی کلاس‌هاست که از آن، کلاس با بیشترین احتمال **(argmax)** و مقدار اطمینان متناظر استخراج می‌گردد.

طراحی «یک‌بارفعال‌سازی» برای ژست توقف (Stop → Play/Pause)

برای ژست «Stop Gesture» (مشت برای توقف/پخش) از یک مکانیزم ضدلرزش و ضدتکرار استفاده شده است تا با یک بار نشان دادن ژست، فقط **یک** بار کلید Space شبیه‌سازی شود:

- اگر کلاس «Stop Gesture» با اطمینان بیشتر از CONF_THRESH دیده شود، شمارنده‌ی stop_consec یکی افزایش می‌یابد؛ در غیر این صورت به صفر برمی‌گردد و متغیر prev_stop_armedمجدداً «مسلح» می‌شود.
- وقتی stop_consec به STABLE_N برسد (یعنی چند فریم پیاپی ژستِ واقعی بوده) و هم‌زمان prev_stop_armed درست باشد و از آخرین اقدام حداقل COOLDOWN_SEC گذشته باشد، یک بار pyautogui.press("space") اجرا می‌شود. سپس prev_stop_armed=False می‌شود تا تا وقتی ژست از تصویر خارج نشده، Space دوباره تکرار نشود.
- به محض خروج از حالت Stop ، دوباره «مسلح» می‌شویم تا اجرای بعدی اجازه داشته باشد.

این الگو از «یک‌بارفعال‌سازی» جلوی اسپم شدن Space را می‌گیرد و تجربه‌ی کاربری طبیعی‌ای می‌دهد: نگه داشتن دست در حالت Stop باعث تکرار شدن Space نمی‌شود؛ باید Stop را برداری و دوباره نشان دهی.

خنثی‌سازی اسپم برای ژست‌های دیگر (Cool Down)

برای ژست‌های غیر از Stop (شست بالا/پایین و چپ/راست) نیز شرط اطمینان (confidence >= CONF_THRESH) برقرار است؛ افزون بر آن، یک «وقفه‌ی زمانی کوتاه» (۰.۱۵ ثانیه برای صدا و ۰.۲۵ ثانیه برای جلو/عقب) اعمال می‌شود تا با باقی ماندن ژست برای چند فریم، فرمان‌ها بی‌وقفه تکرار نشوند:

- pyautogui.press("volumeup") ← Thumbs Up با حداقل فاصله‌ی ۰.۱۵ ثانیه
- pyautogui.press("volumedown") ← Thumbs Down با حداقل فاصله‌ی ۰.۱۵ ثانیه
- pyautogui.press("left") ← Left Swipe با حداقل فاصله‌ی ۰.۲۵ ثانیه
- pyautogui.press("right") ← Right Swipe با حداقل فاصله‌ی ۰.۲۵ ثانیه

متغیر last_action_time آخرین زمان اجرای یک عمل را نگه می‌دارد و جلوی اجرای متوالی بی‌وقفه را می‌گیرد.

بازخورد بصری و مدیریت منابع

روی هر فریم، وضعیت سیستم روی تصویر چاپ می‌شود: اگر آماده‌ی پیش‌بینی نیستیم یا دست پیدا نشده، پیام در حال جستجوی دست .../Warming up...نمایش داده می‌شود؛ اگر پیش‌بینی شده، نام کلاس و اطمینان آن روی تصویر می‌آید. در پایان، وب‌کم و پنجره‌ها به‌درستی آزاد و بسته می‌شوند.

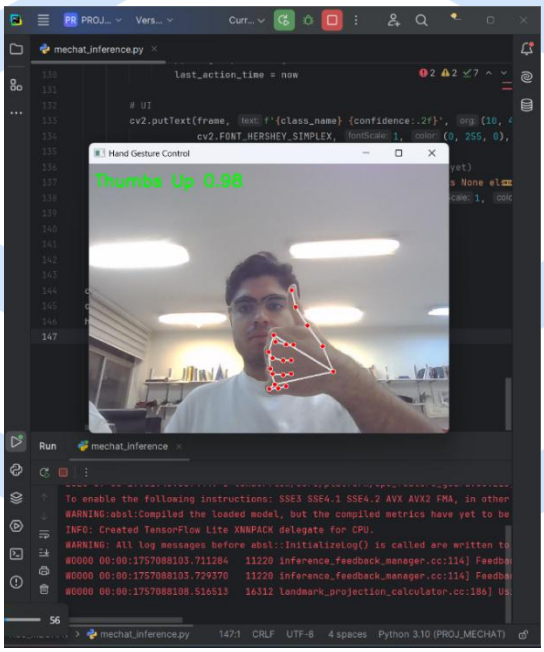
نکات عملی و تنظیم حساسیت

1. **هم‌خوانی ترتیب کلاس‌ها**: ترتیب آرایه‌ی class_names باید کاملاً منطبق با ترتیب خروجی مدل آموزش‌دیده باشد؛ در غیر این صورت، احتمال اجرای دستورات اشتباه وجود دارد.
2. **آستانه‌ها را با محیط میتوان تنظیم کرد** :اگر فرمان‌ها دیر فعال می‌شوند، می‌توانی CONF_THRESH را کمی پایین‌تر بیاوری؛ اگر اشتباه زیاد است، آن را بالاتر بگذار. همین‌طور STABLE_N را بسته به لرزش دست و سرعت وب‌کم تنظیم کن.
3. **نور و پس‌زمینه**: کیفیت تشخیص Mediapipe به نور و پس‌زمینه حساس است. نور یکنواخت و پس‌زمینه‌ی ساده نرخ خطا را کم می‌کند.
4. **کارایی** :کاهش SEQUENCE_LENGTH پاسخ‌دهی را سریع‌تر می‌کند اما ممکن است پایداری را کم کند؛ افزایش آن برعکس. نرخ فریم وب‌کم و توان پردازشی هم اثرگذارند.

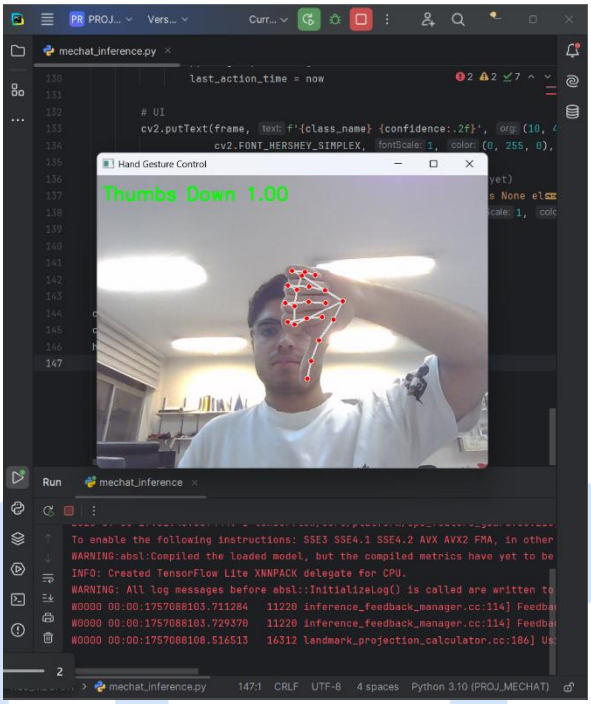
5. **ایمنی** PyAutoGUI: می‌تواند با سیستم تعامل کامل داشته باشد؛ در حین آزمایش، از شورتکات‌های ناخواسته جلوگیری کن و کلید خروج (q) را در دسترس داشته باش.

خلاصه‌ی جریان داده

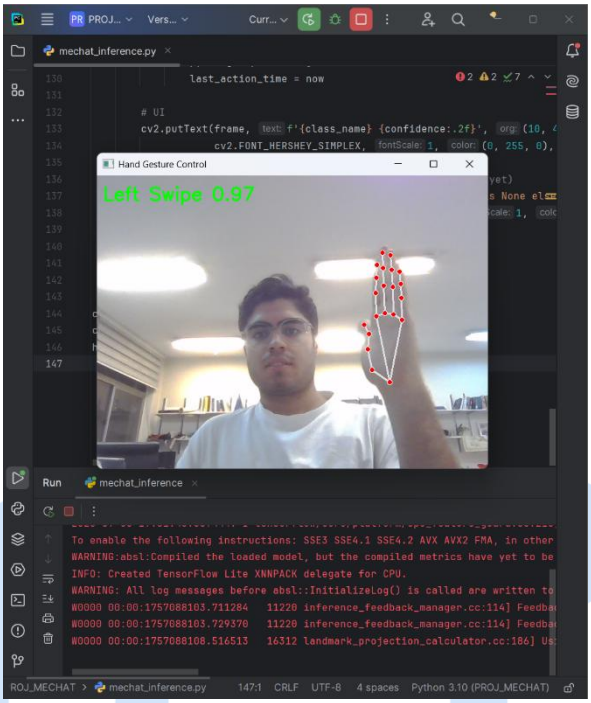
1. خواندن فریم از وب‌کم
2. استخراج لندمارک‌های دست با Mediapipe و تبدیل هر فریم به بردار ۶۳ بعدی
3. افزودن به deque تا رسیدن به ۱۵ فریم
4. ساخت ورودی (1, 15, 63) و پیش‌بینی کلاس + اطمینان
5. اعمال منطق «یک‌بارفعال‌سازی» برای Stop و «وقفه‌ی زمانی» برای سایر ژست‌ها
6. ارسال کلید مناسب با(PyAutoGUI (space/left/right/volumeup/volumedown
7. نمایش بازخورد روی تصویر و تکرار حلقه تا پایان



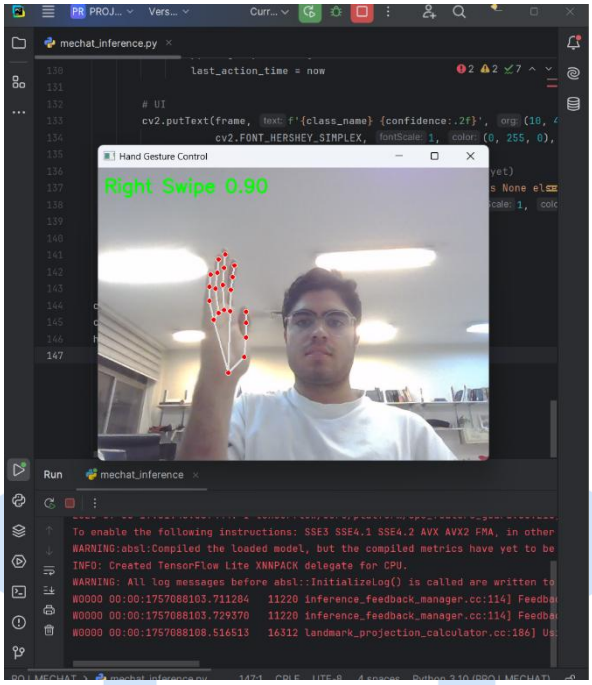
علامت انگشت به بالا که باید **Volume Up** را نشان دهد



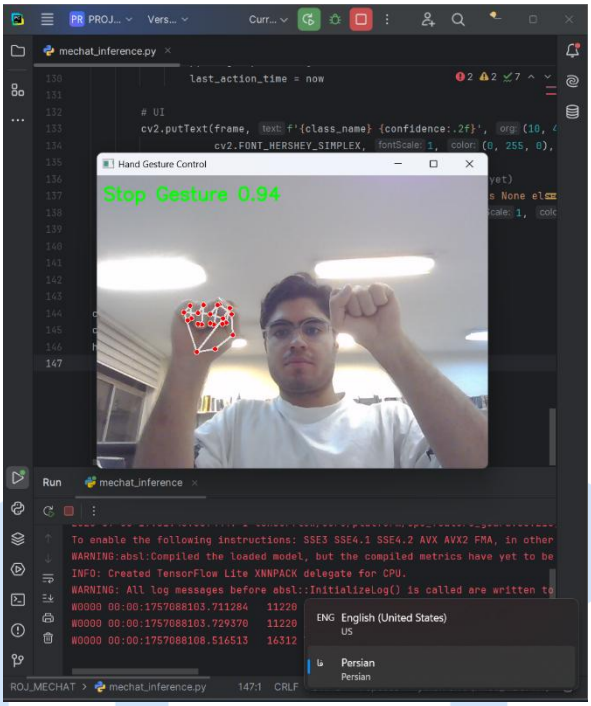
علامت انگشت به پایین که باید **Volume Down** را نشان دهد.



علامت کشیدت دست به چپ که **Left Swipe**



علامت کشیدت دست به چپ که **Right Swipe**



علامت مشت کردن دو دست که به معنی**Stop Gesture** و همانطور که میبینیم، تنها یک دست را قبول میکند.

مراجع

- Kaggle Dataset
- **A Real-time Hand Gesture Recognition System for Human-Computer and Human-Robot Interaction**
- **Hand Gesture Recognition Based on Computer Vision: A Review of Techniques**
- **ChatGPT**