دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشگاه صنعتی خواجه نصیرالدین طوسی
دانشکدهٔ مهندسی برق

درس مبانی مکاترونیک

استاد: دکتر مهدی دلربایی

گزارش کار پروژه پایانی

نام و نام خانوادگی	محمدامین محمدیون شبستری
شمارهٔ دانشجویی	۴۰۱۲۲۵۰۳
نام و نام خانوادگی	یکتا خلیلی
شمارهٔ دانشجویی	۴۰۱۱۷۹۰۳
نام و نام خانوادگی	سوگل سلامت
شمارهٔ دانشجویی	۴۰۱۱۹۳۷۳
تاریخ	شهریور ۱۴۰۴

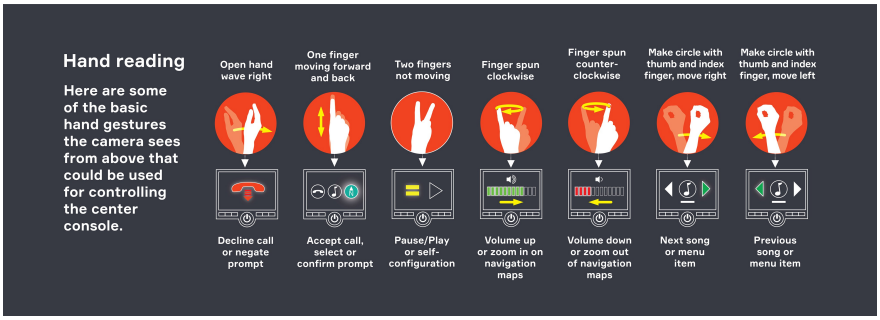
۱ مقدمه

۱.۱ کلیات و اهمیت موضوع

با گسترش روزافزون فناوری‌های نوین در حوزه هوش مصنوعی و بینایی ماشین، ارتباط و تعامل میان انسان و ماشین وارد مرحله‌ای تازه شده است. در گذشته، تعامل با سامانه‌های مکانیکی و الکترونیکی عمدتاً از طریق واسطهای فیزیکی مانند کلیدها، دکمه‌ها و صفحه‌کلیدها صورت می‌گرفت. با پیشرفت فناوری، روش‌های جدیدتری همچون صفحات لمسی، فرمان‌های صوتی و اخیراً تشخیص حرکات بدن و دست معرفی شده‌اند که می‌توانند تجربه کاربری را به طور چشمگیری بهبود بخشند.

۲.۱ کاربردهای تشخیص حرکات دست

یکی از مهم‌ترین شاخه‌های این حوزه، تشخیص حرکات دست (Hand Gesture Recognition) است که امکان صدور فرمان به دستگاه‌ها تنها با استفاده از حرکات طبیعی دست را فراهم می‌کند. این فناوری کاربردهای متنوعی در علوم و صنایع مختلف دارد؛ از کنترل ربات‌های صنعتی در محیط‌های حساس گرفته تا استفاده در سیستم‌های واقعیت مجازی (VR) و واقعیت افزوده (AR). در محیط‌های پزشکی، امکان کنترل تجهیزات بدون لمس می‌تواند از آلودگی متقاطع جلوگیری کند و در حوزه سرگرمی، به بازی‌های ویدئویی و سیستم‌های تعاملی سطح بالاتری از واقع‌گرایی می‌بخشد.



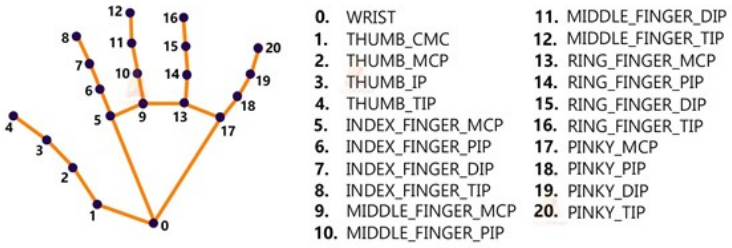
شکل ۱: نمونه‌ای از نگاشت حرکات دست به فرمان‌های کنترلی در یک سیستم تعاملی

۳.۱ نقش مکاترونیک در پروژه

در مهندسی مکاترونیک که ترکیبی از مکانیک، الکترونیک، کنترل و علوم رایانه است، چنین فناوری‌هایی به عنوان بخشی از سیستم‌های هوشمند نقش مهمی ایفا می‌کنند. این پروژه یک نمونه کاربردی از تلفیق بینایی ماشین و یادگیری ماشین است و نشان‌دهنده رویکرد میان‌رشته‌ای مکاترونیک در ترکیب فناوری‌های نرم‌افزاری و سخت‌افزاری برای ایجاد سیستم‌های هوشمند و کارآمد می‌باشد. مزیت مهم آن این است که پیاده‌سازی به صورت نرم‌افزاری انجام شده و در عین حال امکان یکپارچه‌سازی با سخت‌افزارهای مکاترونیکی نظیر ربات‌ها را نیز دارد.

۴.۱ معرفی ابزار کلیدی: MediaPipe

یکی از ابزارهای پیشرفته در این زمینه، کتابخانه MediaPipe است که توسط گوگل توسعه یافته و به صورت متن‌باز در اختیار پژوهشگران قرار گرفته است. این کتابخانه مجموعه‌ای از الگوریتم‌های از پیش‌ساخته را برای پردازش بلادرنگ تصویر و صوت ارائه می‌دهد. در بخش شناسایی دست، MediaPipe قادر است با دقت بالا ۲۱ نقطه کلیدی (**Landmarks**) هر دست را شناسایی کند. این نقاط شامل محل دقیق مفاصل، کف دست و نوک انگشتان بوده و ساختار هندسی کامل دست را بازسازی می‌کنند (تصویر ۲). مزیت اصلی این روش، عدم نیاز به تجهیزات گران‌قیمت و استفاده از یک دوربین معمولی برای استخراج داده است.



شکل ۲: ۲۱ نقطه کلیدی دست (Landmarks) که توسط MediaPipe شناسایی می‌شوند.

۵.۱ اهداف پروژه

هدف این پروژه، طراحی و پیاده‌سازی یک سامانه تشخیص حرکات دست با استفاده از MediaPipe و مدل‌های یادگیری ماشین است. فرآیند شامل مراحل زیر است:

- تهیه و بارگذاری یک مجموعه داده شامل تصاویر حرکات دست.
- استخراج مختصات نقاط کلیدی دست از هر تصویر با MediaPipe.
- آموزش یک مدل طبقه‌بندی‌کننده برای شناسایی الگوهای حرکتی.
- ارزیابی عملکرد مدل برای سنجش دقت و کارایی سیستم.
- پیاده‌سازی یک برنامه نهایی برای کنترل بی‌درنگ مدیاپلیر با استفاده از مدل آموزش‌دیده.

۲ فرآیند آموزش مدل

۱.۲ بارگذاری و آماده‌سازی داده‌ها

۱.۱.۲ فراخوانی کتابخانه‌ها و دانلود مجموعه داده

در گام نخست، کتابخانه‌های اصلی مورد نیاز پروژه شامل numpy، pandas و os فراخوانی شدند.

```python
import numpy as np
import pandas as pd
import os
```

Code 1: اولیه کتابخانه‌های فراخوانی

سپس، با استفاده از کتابخانه‌های kagglehub و mediapipe، مجموعه داده hand-gesture-detection-system از Kaggle دانلود و در محیط اجرایی بارگذاری شد.

```python
import kagglehub
import mediapipe as mp
path = kagglehub.dataset_download("marusagar/hand-gesture-detection-system")
print("Path to dataset files:", path)
```

Code 2: Kaggle از داده مجموعه دانلود

۲.۱.۲ تحلیل و اصلاح ساختار اولیه داده‌ها

پس از بارگذاری، مشخص شد که فایل train.csv ساختاری نامناسب دارد و تمام اطلاعات هر سطر در یک ستون با جداکننده «؛» قرار گرفته‌اند (تصویر ۳).

	WIN_20180925_17_08_43_Pro_Left_Swipe_new;Left_Swipe_new;0
0	WIN_20180925_17_18_28_Pro_Left_Swipe_new;Left_...
1	WIN_20180925_17_18_56_Pro_Left_Swipe_new;Left_...
2	WIN_20180925_17_19_51_Pro_Left_Swipe_new;Left_...
3	WIN_20180925_17_20_14_Pro_Left_Swipe_new;Left_...
4	WIN_20180925_17_21_28_Pro_Left_Swipe_new;Left_...
...	...
657	WIN_20180907_16_38_23_Pro_Thumbs Up_new;Thumbs...
658	WIN_20180907_16_41_09_Pro_Thumbs Up_new;Thumbs...
659	WIN_20180907_16_42_05_Pro_Thumbs Up_new;Thumbs...
660	WIN_20180907_16_42_55_Pro_Thumbs Up_new;Thumbs...
661	WIN_20180907_16_43_39_Pro_Thumbs Up_new;Thumbs...

662 rows × 1 columns

شکل ۳: ساختار اولیه و نامناسب فایل train.csv پیش از پردازش.

این مشکل با خواندن مجدد فایل و تعیین sep=';' و همچنین تعریف نام ستون‌ها (IMAGE_NAME، LABEL، CLASS) برطرف گردید.

```python
df = pd.read_csv('.../train.csv', sep=';', names=['IMAGE_NAME', 'LABEL', 'CLASS'])
```

Code 3: صحیح ستون‌های نام و جداکننده با CSV فایل خواندن

	IMAGE_NAME	LABEL	CLASS
0	WIN_20180925_17_08_43_Pro_Left_Swipe_new	Left_Swipe_new	0
1	WIN_20180925_17_18_28_Pro_Left_Swipe_new	Left_Swipe_new	0
2	WIN_20180925_17_18_56_Pro_Left_Swipe_new	Left_Swipe_new	0
3	WIN_20180925_17_19_51_Pro_Left_Swipe_new	Left_Swipe_new	0
4	WIN_20180925_17_20_14_Pro_Left_Swipe_new	Left_Swipe_new	0
...	...	...	...
658	WIN_20180907_16_38_23_Pro_Thumbs Up_new	Thumbs Up_new	4
659	WIN_20180907_16_41_09_Pro_Thumbs Up_new	Thumbs Up_new	4
660	WIN_20180907_16_42_05_Pro_Thumbs Up_new	Thumbs Up_new	4
661	WIN_20180907_16_42_55_Pro_Thumbs Up_new	Thumbs Up_new	4
662	WIN_20180907_16_43_39_Pro_Thumbs Up_new	Thumbs Up_new	4

663 rows × 3 columns

شکل ۴: ساختار اصلاح‌شده دیتافریم پس از بارگذاری صحیح.

در این دیتافریم، هر سطر به یک ویدیو کلیپ کوتاه از یک ژست خاص اشاره دارد. ستون LABEL شامل نام ژست مانند Left_Swipe_new، Thumbs_Up_new، Stop_Gesture_new و غیره است.

۲.۲ پاک‌سازی و پیش‌پردازش داده‌ها

۱.۲.۲ استانداردسازی برچسب‌های متنی

داده‌های خام معمولا نیازمند پاک‌سازی هستند تا برای مدل‌های یادگیری ماشین قابل استفاده باشند. در اینجا، برچسب‌های موجود در ستون LABEL حاوی اطلاعات اضافی و فرمت‌های ناهمگون بودند. عباراتی مانند new_ و کاراکتر _ حذف شدند تا نام ژست‌ها خواناتر و استاندارد شوند.

```
df['LABEL'] = df['LABEL'].str.replace('_', " ")
df['LABEL'] = df['LABEL'].str.replace('new', "")
df['LABEL'].value_counts()
```

Code 4: برچسب‌ها از اضافی کاراکترهای و عبارات حذف

LABEL	count
Right Swipe	137
Thumbs Down	137
Thumbs Up	123
Left Swipe	96
Stop	93
Left Swipe Left Swipe	40
Stop Gesture	37

dtype: int64

شکل ۵: توزیع اولیه برچسب‌ها که شامل موارد تکراری است.

۲.۲.۲ یکپارچه‌سازی کلاس‌های مشابه

همانطور که در تصویر ۵ مشخص است، برخی ژست‌ها با نام‌های مختلفی ثبت شده بودند. برای مثال، "Left Swipe Left Swipe" و "Stop Gesture" وجود داشتند. این موارد به ترتیب به "Left Swipe" و "Stop" تبدیل شدند تا هر کلاس معنای منحصربه‌فردی داشته باشد. این کار از ایجاد کلاس‌های اضافی و سردرگمی مدل جلوگیری می‌کند.

```
df['LABEL'] = df['LABEL'].str.replace('Left Swipe  Left Swipe ', "Left Swipe ")
df['LABEL'] = df['LABEL'].str.replace('Stop Gesture ', "Stop ")
```

Code 5: یکپارچه‌سازی برچسب‌های مشابه

LABEL	count
Right Swipe	137
Thumbs Down	137
Left Swipe	136
Stop	130
Thumbs Up	123

dtype: int64

شکل ۶: توزیع نهایی و پاک‌سازی‌شده برچسب‌ها در مجموعه داده آموزشی.

این فرآیند برای مجموعه داده اعتبارسنجی (val.csv) نیز تکرار شد.

LABEL	count
Right Swipe	23
Stop	22
Thumbs Down	21
Left Swipe	18
Thumbs Up	16

dtype: int64

شکل ۷: توزیع پاک‌سازی‌شده برچسب‌ها در مجموعه داده اعتبارسنجی.

۳.۲.۲ مقایسه توزیع کلاس‌ها در داده‌های آموزش و اعتبارسنجی

یک مجموعه داده اعتبارسنجی خوب باید توزیع کلاس‌های مشابهی با مجموعه داده آموزشی داشته باشد. با مقایسه تصاویر ۶ و ۷ می‌توان دید که:

- در هر دو مجموعه، ۵ کلاس نهایی (Right Swipe, Thumbs Down, Left Swipe, Stop, Thumbs Up) وجود دارند.
- توزیع داده‌ها در هر دو مجموعه نسبتاً متوازن (Balanced) است، یعنی تعداد نمونه‌های هر کلاس تفاوت فاحشی با دیگری ندارد. این موضوع به مدل کمک می‌کند تا به تمام کلاس‌ها اهمیت یکسانی بدهد و از سوگیری (Bias) به سمت کلاس‌های پرتعدادتر جلوگیری شود.
- نسبت تعداد نمونه‌ها در داده اعتبارسنجی نیز منعکس‌کننده توزیع در داده آموزشی است که نشان می‌دهد این تقسیم‌بندی برای ارزیابی عملکرد مدل مناسب است.

۳.۲ استخراج ویژگی با MediaPipe

۱.۳.۲ نقاط کلیدی (Landmarks) چه هستند؟

به جای استفاده مستقیم از پیکسل‌های تصویر که ابعاد بسیار بزرگی دارند و شامل اطلاعات اضافی (مانند پس‌زمینه) هستند، از یک رویکرد مبتنی بر ویژگی استفاده می‌شود. Landmarks یا نقاط کلیدی، مجموعه‌ای از نقاط با مختصات مشخص روی یک شیء هستند که ساختار هندسی آن را توصیف می‌کنند. برای دست انسان، MediaPipe ۲۱ نقطه را تعریف کرده که متناظر با مفاصل انگشتان، نوک انگشتان و مچ است (همانطور که در تصویر ۲ نشان داده شد). هر نقطه دارای سه مختصات (x, y, z) است. این مختصات نرمال‌سازی شده‌اند:

- x و y: نسبت به عرض و ارتفاع تصویر نرمال شده و مقداری بین ۰ و ۱ دارند.
- z: عمق نقطه را نسبت به مچ دست نشان می‌دهد. مقدار کوچکتر به معنای نزدیک‌تر بودن به دوربین است.

با اتصال این ۲۱ نقطه، یک اسکلت سه‌بعدی از دست به دست می‌آید که اطلاعات بسیار غنی و فشرده‌ای از ژست دست را بدون توجه به پس‌زمینه یا رنگ پوست ارائه می‌دهد.

۲.۳.۲ پیکربندی ماژول MediaPipe Hands

ماژول mp.solutions.hands.Hands یک پایپ‌لاین پیچیده از مدل‌های یادگیری عمیق است. فرآیند کار آن به طور خلاصه به این صورت است: ابتدا یک مدل سبک به نام Palm Detector کل تصویر را برای پیدا کردن ناحیه کف دست جستجو می‌کند. پس از یافتن دست، ناحیه تصویر برش خورده و به یک مدل سنگین‌تر به نام Hand Landmark Model داده می‌شود که ۲۱ نقطه کلیدی را با دقت بالا روی آن مشخص می‌کند. این رویکرد دو مرحله‌ای باعث افزایش سرعت و دقت می‌شود.

```python
import mediapipe as mp
hands = mp.solutions.hands.Hands(
    static_image_mode=True,
    max_num_hands=1,
    min_detection_confidence=0.5
)
```

Code 6: دست تشخیص مدل پیکربندی MediaPipe

۳.۳.۲ تابع استخراج نقاط کلیدی و تبدیل فرمت

خروجی MediaPipe یک شیء پیچیده است. تابع extract_landmarks این شیء را به یک فرمت استاندارد و قابل استفاده برای مدل‌های یادگیری ماشین تبدیل می‌کند. این تابع، مختصات x, y, z هر ۲۱ نقطه را پشت سر هم قرار داده و یک بردار یک‌بعدی با طول $21 \times 3 = 63$ ایجاد می‌کند. برای مثال، خروجی برای یک نقطه به صورت [x0, y0, z0, x1, y1, z1, ..., x20, y20, z20] خواهد بود.

```python
def extract_landmarks(result):
    if result.multi_hand_landmarks:
        hand_landmarks = result.multi_hand_landmarks[0]
        landmarks_ = []
        for lm in hand_landmarks.landmark:
            landmarks_.extend([lm.x, lm.y, lm.z])
        return np.array(landmarks_)
    return None
```

Code 7: دست کلیدی نقاط مسطح‌سازی و استخراج تابع

۴.۳.۲ اجرای فرآیند استخراج ویژگی روی داده‌ها

یک حلقه بر روی تمام تصاویر مجموعه داده اجرا شد. در هر تکرار، تصویر با OpenCV خوانده شده و به فرمت رنگی RGB تبدیل می‌شد. دلیل این تبدیل این است که OpenCV تصاویر را به صورت پیش‌فرض در فرمت BGR (آبی، سبز، قرمز) می‌خواند، در حالی که مدل‌های MediaPipe بر روی تصاویر با فرمت RGB (قرمز، سبز، آبی) آموزش دیده‌اند. عدم تطابق این فرمت‌ها می‌تواند منجر به تشخیص نادرست شود.

	Image_name	Folder	Label	Class	Landmarks
0	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180925_17_08_43_Pro_Left_Swipe_new	Left Swipe	0	[0.29479125142097473, 0.8378140330314636, 3.55...
1	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180925_17_08_43_Pro_Left_Swipe_new	Left Swipe	0	[0.26317331194877625, 0.7459539771080017, 2.65...
2	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180925_17_08_43_Pro_Left_Swipe_new	Left Swipe	0	[0.2492646127939224, 0.6994234323501587, 2.24...
3	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180925_17_08_43_Pro_Left_Swipe_new	Left Swipe	0	[0.2189014554023747, 0.6593414545059204, 2.04...
4	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180925_17_08_43_Pro_Left_Swipe_new	Left Swipe	0	[0.21465922892093658, 0.6445009708404541, 2.08...
...	...	...	...	...	...
19885	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180907_16_43_39_Pro_Thumbs Up_new	Thumbs Up	4	None
19886	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180907_16_43_39_Pro_Thumbs Up_new	Thumbs Up	4	None
19887	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180907_16_43_39_Pro_Thumbs Up_new	Thumbs Up	4	None
19888	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180907_16_43_39_Pro_Thumbs Up_new	Thumbs Up	4	None
19889	/root/.cache/kagglehub/datasets/marusagar/hand...	WIN_20180907_16_43_39_Pro_Thumbs Up_new	Thumbs Up	4	None

19890 rows × 5 columns

شکل ۸: دیتافریم نهایی شامل ستون Landmarks که ویژگی‌های استخراج‌شده را نگهداری می‌کند.

۵.۳.۲ مدیریت داده‌های فاقد ویژگی و دلیل آن

در این فرآیند، MediaPipe در ۵۰۴۶ تصویر آموزشی و ۶۷۳ تصویر اعتبارسنجی موفق به تشخیص دست نشد و مقدار None برگرداند. دلیل اصلی این اتفاق، پارامتر min_detection_confidence=0.5 است. این بدان معناست که مدل تشخیص کف دست (Palm Detector) در این تصاویر، یک ناحیه را با اطمینان کمتر از ۵۰٪ به عنوان دست شناسایی کرده و در نتیجه آن را نادیده گرفته است. دلایل

دیگر می‌تواند شامل تاری شدید تصویر، نور بسیار کم یا زیاد، یا قرار گرفتن دست در زاویه‌ای نامتعارف باشد. از آنجایی که این سطرها فاقد ویژگی ورودی برای مدل بودند، با دستور dropna از دیتافریم‌ها حذف شدند. این کار برای جلوگیری از بروز خطا و آموزش مدل با داده‌های ناقص ضروری است.

```
df_images.dropna(subset=['Landmarks'], inplace=True)
df_valid.dropna(subset=['Landmarks'], inplace=True)
```

Code 8: ویژگی فاقد سطر های حذف

۳ تعریف، پیاده‌سازی و ارزیابی مدل‌ها

۱.۳ مدل ماشین بردار پشتیبان (SVM)

۱.۱.۳ دلیل انتخاب SVM به عنوان مدل اولیه

مدل SVM به چند دلیل به عنوان اولین انتخاب در نظر گرفته شد:

- کارایی در فضای با ابعاد بالا: داده‌های ما یک بردار ۶۳ بعدی هستند. SVM در چنین فضاهایی عملکرد بسیار خوبی دارد.
- سرعت و کارایی: برای مجموعه داده‌هایی با اندازه متوسط (مانند پروژه ما با حدود ۱۵۰۰۰ نمونه)، SVM معمولاً سریع‌تر از شبکه‌های عصبی عمیق آموزش می‌بیند و نیاز به منابع محاسباتی کمتری دارد.
- مقاومت در برابر بیش‌برازش: با تنظیم صحیح پارامتر C، می‌توان توازن خوبی بین دقت و تعمیم‌پذیری مدل ایجاد کرد.

۲.۱.۳ بهینه‌سازی هایپرپارامترها با GridSearchCV

برای یافتن بهترین ترکیب هایپرپارامترها، از GridSearchCV استفاده شد. پارامترهای بررسی‌شده در جدول زیر آمده‌اند:

پارامتر	مقادیر آزمایشی	توضیح
kernel	rbf, linear, poly	نوع هسته برای نگاشت داده‌ها به فضای با ابعاد بالاتر.
C	0.1, 1, 10, 100	پارامتر تنظیم (Regularization). مقدار بالاتر، خطای کمتری را در داده آموزشی مجاز می‌داند.
gamma	0.001, 0.01, 0.1, 1, 'scale'	ضریب هسته برای rbf و poly. تأثیر یک نمونه آموزشی را مشخص می‌کند.
degree	2, 3, 4	درجه چندجمله‌ای برای هسته poly.

جدول ۱: پارامترهای مورد استفاده در فرآیند جستجوی شبکه‌ای.

نتایج نشان داد که بهترین عملکرد با کرنل rbf، C=100 و gamma=0.001 به دست آمد.

۳.۱.۳ اهمیت نرمال‌سازی داده‌ها

قبل از ورود داده‌ها به مدل SVM، از StandardScaler برای نرمال‌سازی استفاده شد. این کار مقیاس تمام ۶۳ ویژگی را یکسان می‌کند (میانگین صفر و واریانس یک). از آنجایی که SVM بر اساس محاسبه فاصله بین نمونه‌ها کار می‌کند، اگر ویژگی‌ها مقیاس‌های متفاوتی داشته باشند، ویژگی‌هایی با مقادیر بزرگتر تأثیر نامتناسبی بر مدل خواهند گذاشت. نرمال‌سازی این مشکل را حل کرده و به همگرایی سریع‌تر و عملکرد بهتر مدل کمک می‌کند.

۴.۱.۳ ارزیابی دقیق عملکرد مدل **SVM**

مدل نهایی دقت کلی **۸۷.۳۲**% را کسب کرد. تحلیل دقیق‌تر معیارها در تصویر ۹ نشان می‌دهد:

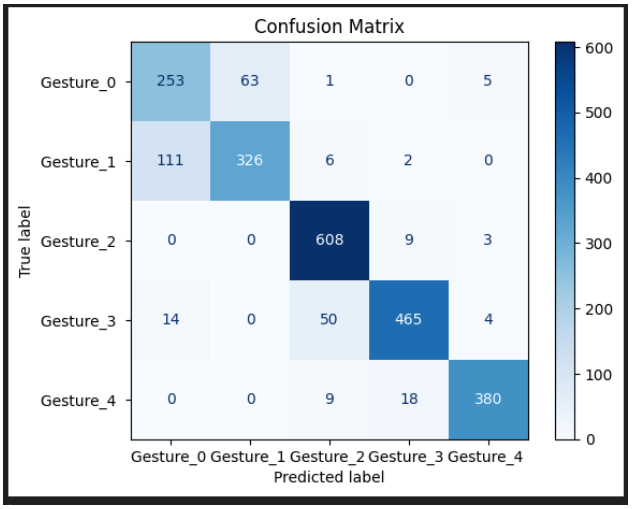
- Precision (دقت پیش‌بینی): این معیار نشان می‌دهد که از میان نمونه‌هایی که مدل به عنوان یک کلاس خاص پیش‌بینی کرده، چه درصدی واقعاً درست بوده‌اند. برای مثال، Precision ۰.۹۷ برای کلاس ۴ (Thumbs Up) به این معناست که اگر مدل ژستی را به عنوان Thumbs Up تشخیص دهد، در ۹۷% موارد درست گفته است.
- Recall (بازخوانی): این معیار نشان می‌دهد که مدل چه درصدی از نمونه‌های واقعی یک کلاس را توانسته به درستی شناسایی کند. برای مثال، Recall ۰.۹۸ برای کلاس ۲ (Stop) یعنی مدل ۹۸% از تمام ژست‌های Stop واقعی را پیدا کرده است.
- F۱-Score: این معیار میانگین هماهنگ بین Precision و Recall است و یک دید کلی از توازن بین این دو ارائه می‌دهد.

تحلیل ماتریس درهم‌ریختگی (تصویر ۱۰) نشان می‌دهد که ۱۱۱ نمونه از کلاس ۱ (Right Swipe) به اشتباه کلاس ۰ (Left Swipe) تشخیص داده شده‌اند که بزرگترین منبع خطا است. این خطا به دلیل شباهت ساختاری و قرینه بودن این دو ژست کاملاً قابل انتظار است.

	precision	recall	f1-score	support
0	0.67	0.79	0.72	322
1	0.84	0.73	0.78	445
2	0.90	0.98	0.94	620
3	0.94	0.87	0.91	533
4	0.97	0.93	0.95	407
accuracy			0.87	2327
macro avg	0.86	0.86	0.86	2327
weighted avg	0.88	0.87	0.87	2327

Accuracy: 0.87322733132789

شکل ۹: نتایج بهترین مدل svm



شکل ۱۰: ماتریس در هم ریختگی مربوط به svm

۲.۳ مدل پرسپترون چندلایه (MLP)

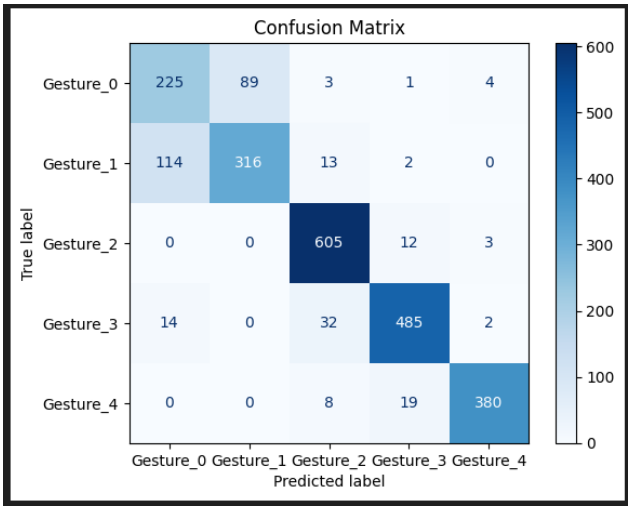
۱.۲.۳ معماری و هایپرپارامترهای مدل

یک مدل MLP به عنوان جایگزین SVM پیاده‌سازی شد. معماری آن شامل یک لایه ورودی با ۶۳ نورون، دو لایه مخفی با ۲۵۶ و ۱۲۸ نورون و تابع فعال‌سازی ReLU بود. برای جلوگیری از بیش‌برازش، پس از هر لایه مخفی یک لایه Dropout با نرخ ۰.۳ قرار داده شد که به طور تصادفی ۳۰% از نورون‌ها را در هر مرحله آموزش غیرفعال می‌کرد. هایپرپارامترهای آموزش به شرح زیر بودند:

- بهینه‌ساز: AdamW
- نرخ یادگیری: 1e-4
- تابع هزینه: CrossEntropyLoss (مناسب برای مسائل طبقه‌بندی چندکلاسه)
- اندازه دسته: ۶۴
- مکانیسم توقف: توقف زودهنگام با patience=5

۲.۲.۳ ارزیابی عملکرد مدل MLP

این مدل به دقت کلی **۸۶.۴۲**% دست یافت. تحلیل ماتریس درهم‌ریختگی آن (تصویر ۱۱) نیز نتایج مشابهی با SVM نشان داد؛ یعنی بیشترین خطا در تفکیک بین ژست‌های ۰ و ۱ بود. هرچند عملکرد کلی خوب بود، اما اندکی ضعیف‌تر از مدل SVM عمل کرد.



شکل ۱۱: ماتریس در هم ریختگی mlp

Classification Report:	precision	recall	f1-score	support
0	0.64	0.70	0.67	322
1	0.78	0.71	0.74	445
2	0.92	0.98	0.94	620
3	0.93	0.91	0.92	533
4	0.98	0.93	0.95	407
accuracy			0.86	2327
macro avg	0.85	0.85	0.85	2327
weighted avg	0.87	0.86	0.86	2327

شکل ۱۲: نتایج ساختار mlp

۳.۳ مقایسه و انتخاب مدل نهایی

جدول زیر دو مدل را بر اساس معیارهای مختلف مقایسه می‌کند:

معیار	مدل **SVM**	مدل **MLP**
دقت (Accuracy)	٪۸۷.۳۲	٪۸۶.۴۲
F1-Score (وزنی)	۰.۸۷	۰.۸۶
پیچیدگی مدل	سبک و ساده	پیچیده‌تر
سرعت پیش‌بینی (Inference)	بسیار بالا	بالا (اما کندتر از SVM)
زمان آموزش	طولانی (به دلیل GridSearch)	نسبتاً سریع (با توقف زودهنگام)
وابستگی‌ها	Scikit-learn	PyTorch (سنگین‌تر)

جدول ۲: مقایسه جامع بین مدل‌های SVM و MLP.

با توجه به دقت بالاتر و همچنین سبک‌تر و سریع‌تر بودن در مرحله پیش‌بینی، مدل SVM به عنوان گزینه نهایی برای پیاده‌سازی در برنامه بی‌درنگ انتخاب شد.

۴ پیاده‌سازی نهایی و کاربرد بی‌درنگ

۱.۴ معماری سیستم بی‌درنگ و ابزارها

این سیستم با هدف پردازش لحظه‌ای تصویر وب‌کم، تشخیص ژست و ارسال فرمان به سیستم‌عامل طراحی شده است. کتابخانه‌های کلیدی در جدول قبل معرفی شدند.

۲.۴ بارگذاری مدل آموزش‌دیده

در ابتدای اسکریپت، مدل SVM ذخیره‌شده با استفاده از joblib.load بارگذاری می‌شود.

```
import joblib
try:
    model = joblib.load('final_model_v2_svm (1).pkl')
except FileNotFoundError:
    print("Error: Model file not found.")
    exit()
```

Code 9: بارگذاری مدل SVM ذخیره‌شده

۳.۴ نگاشت ژست به فرمان (Gesture-to-Command Mapping)

دیکشنری GESTURE_MAP کلاس‌های خروجی مدل را به فرمان‌های قابل فهم و کلیدهای کیبورد نگاشت می‌دهد.

```
GESTURE_MAP = {
    0: "Rewind",        # ->
    1: "Fast Forward",  # ->
    2: "Play/Pause",    # ->    Space
    3: "Volume Down",   # ->
    4: "Volume Up"      # ->
{
```

Code 10: کنترلی فرمان‌های به پیش‌بینی کلاس‌های نگاشت

۴.۴ حلقه اصلی و پردازش فریم

برنامه وارد یک حلقه بی‌نهایت می‌شود که در هر تکرار، یک فریم از وب‌کم می‌خواند، آن را برمی‌گرداند و به فرمت RGB تبدیل می‌کند.

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)
    img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    result = hands.process(img_rgb)
    # ...       ...
```

ویدئویی فریم‌های پردازش حلقه اصلی بخش :Code 11

۵.۴ پیش‌بینی و اجرای فرمان با PyAutoGUI

پس از تشخیص دست، ویژگی‌ها استخراج و ژست توسط مدل SVM پیش‌بینی می‌شود. برای جلوگیری از ارسال فرمان‌های تکراری، یک تأخیر یک ثانیه‌ای (ACTION_DELAY) در نظر گرفته شده است.

```
current_time = time.time()
if current_time - last_action_time > ACTION_DELAY:
    if action_name == "Play/Pause":
        pyautogui.press('space')
        video_is_playing = not video_is_playing
        last_action_time = current_time
    elif video_is_playing:
        if action_name == "Fast Forward":
            pyautogui.press('right')
            last_action_time = current_time
        # ...         ...
```

تشخیص‌دادهشده ژست اساس بر فرمان اجرای منطق :Code 12

۶.۴ بازخورد بصری و خاتمه برنامه

نقاط کلیدی دست، وضعیت پخش و نام ژست تشخیص‌دادهشده به صورت زنده روی فریم ویدئو ترسیم می‌شوند. با فشردن کلید ‘q‘، برنامه خاتمه می‌یابد.

۵ نتیجه‌گیری و پیشنهادات

۱.۵ جمع‌بندی نهایی

این پروژه با موفقیت یک سیستم کامل از آموزش تا پیاده‌سازی برای تشخیص حرکات دست را اجرا کرد. نتایج نشان داد که استخراج ویژگی‌های ساختاری دست با استفاده از MediaPipe و طبقه‌بندی آن‌ها با یک مدل یادگیری ماشین کلاسیک و سبک مانند SVM، یک رویکرد بسیار مؤثر و کارآمد است. مدل نهایی با دستیابی به دقت ۸۷.۳۲%، توانایی خود را در تفکیک پنج ژست مختلف به اثبات رساند و کاربرد عملی آن در کنترل مدیاپلیر، پتانسیل این فناوری را در تعاملات انسان و ماشین به خوبی نمایش داد.

۲.۵ پیشنهادات برای کارهای آتی

برای بهبود و گسترش این پروژه در آینده، موارد زیر پیشنهاد می‌شود:

- استفاده از مدل‌های زمانی (**Temporal Models**): برای تشخیص بهتر حرکات پویا که در طول زمان تعریف می‌شوند (مانند Swipe)، می‌توان از شبکه‌های عصبی بازگشتی (RNN) مانند LSTM یا GRU استفاده کرد که توالی فریم‌ها را تحلیل می‌کنند.
- افزایش داده (**Data Augmentation**): می‌توان با اعمال تغییرات جزئی مانند چرخش، تغییر مقیاس یا افزودن نویز به بردارهای ویژگی Landmarks، داده‌های آموزشی را غنی‌تر کرده و مدل را در برابر شرایط مختلف مقاوم‌تر نمود.
- گسترش مجموعه ژست‌ها: با جمع‌آوری داده‌های بیشتر، می‌توان تعداد ژست‌های قابل تشخیص را افزایش داد و سیستم را برای کاربردهای پیچیده‌تر، مانند کنترل یک بازوی رباتیک، توسعه داد.
- بهینه‌سازی برای محیط‌های نویزی: می‌توان مدل را با داده‌هایی که در شرایط نوری مختلف یا با پس‌زمینه‌های شلوغ گرفته شده‌اند، آموزش داد تا پایداری آن در محیط‌های واقعی افزایش یابد.
- یکپارچه‌سازی با سخت‌افزار و پیاده‌سازی : به عنوان فاز بعدی، پروژه بر روی یک برد Raspberry Pi 3 پیاده‌سازی خواهد شد. در این مرحله، عملگر (actuator) سیستم از کنترل نرم‌افزاری مدیاپلیر به کنترل سخت‌افزارهای فیزیکی تغییر خواهد یافت. اهداف این فاز شامل کنترل یک الگوی نوری با استفاده از آرایه‌ای از LEDها یا کنترل جهت و سرعت یک موتور الکتریکی بر اساس ژست‌های تشخیص داده شده می‌باشد. این گام، کاربرد عملی پروژه در سیستم‌های مکاترونیکی را به نمایش می‌گذارد.