

Homework 3 ChatGPT

Process video to extract feature

I got this code by coping and modifying Medaipipe official guide:

```
def extract_features(video_path, label, frame_interval):

    # Eye gaze features (8 features)
    eye_gaze_features = [
        'eyeLookUpLeft', 'eyeLookUpRight',
        'eyeLookDownLeft', 'eyeLookDownRight',
        'eyeLookInLeft', 'eyeLookInRight',
        'eyeLookOutLeft', 'eyeLookOutRight'
    ]

    # Eye blink features (2 features)
    eye_blink_features = ['eyeBlinkLeft', 'eyeBlinkRight']

    logging.info(f"Start extract feture from video {video_path}")
    # Create FaceLandmarker

    base_options = python.BaseOptions(model_asset_path=model_path)
    options = vision.FaceLandmarkerOptions(
        base_options=base_options,
        output_face_blendshapes=True, # Main feature
        output_facial_transformation_matrixes=True, # transformation matrix
        num_faces=1,
    )
```

Then I ask AI to keep write my code to archive my goal.

📄 Video feature extraction with face landmarks ▾

Share

RH I want to extract blendshape and transformation matrixies, and write in a csv or other format file:

```
def extract_features(video_path, label, frame_interval):  
    # Eye gaze features (8 features)  
    eye_gaze_features = [  
        'eyeLookUpLeft', 'eyeLookUpRight',  
        'eyeLookDownLeft', 'eyeLookDownRight',  
        'eyeLookInLeft', 'eyeLookInRight',  
        'eyeLookOutLeft', 'eyeLookOutRight'  
    ]  
  
    # Eye blink features (2 features)  
    eye_blink_features = ['eyeBlinkLeft', 'eyeBlinkRight']  
  
    logging.info(f"Start extract feture from video {video_path}")  
    # Create FaceLandmarker  
  
    base_options = python.BaseOptions(model_asset_path=model_path)  
    options = vision.FaceLandmarkerOptions(  
        base_options=base_options,  
        output_face_blendshapes=True, # Main feature  
        output_facial_transformation_matrixes=True, # transformation matrix  
        num_faces=1,  
    )
```

I'll help you extract blendshapes and transformation matrices from video frames and save them to a CSV file. Here's a complete solution:

Extract Face Blendshapes and Transformation Matrices

Code



I've created a complete solution for extracting face blendshapes and transformation

Then I got the remain code:

```
fps = cap.get(cv2.CAP_PROP_FPS)  
total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))  
logging.info(f"Video FPS: {fps}, Total frames: {total_frames}")  
  
# Storage for extracted data  
data_rows = []  
  
frame_count = 0  
processed_count = 0
```

```

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    # Process every nth frame
    if frame_count % frame_interval != 0:
        frame_count += 1
        continue

    # Convert BGR to RGB
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    mp_image = mp.Image(image_format=mp.ImageFormat.SRGB, data=rgb_frame)

    # Detect face landmarks
    detection_result = detector.detect(mp_image)

    # Initialize row data
    row_data = {
        'video_path': video_path,
        'label': label,
        'frame_number': frame_count,
        'timestamp': frame_count / fps
    }

    # Extract blendshapes
    if detection_result.face_blendshapes:
        blendshapes = detection_result.face_blendshapes[0]

        # Extract all blendshapes
        for blendshape in blendshapes:
            row_data[blendshape.category_name] = blendshape.score

    # Extract transformation matrix
    if detection_result.facial_transformation_matrixes:
        matrix = detection_result.facial_transformation_matrixes[0]

        # Flatten 4x4 transformation matrix
        # Matrix is typically 4x4 for homogeneous coordinates
        for i in range(len(matrix)):
            for j in range(len(matrix[i])):
                row_data[f'transform_matrix_{i}_{j}'] = matrix[i][j]

    data_rows.append(row_data)
    processed_count += 1

    if processed_count % 100 == 0:
        logging.info(f"Processed {processed_count} frames")

    frame_count += 1

cap.release()

```

```

logging.info(f"Total frames processed: {processed_count}")

# Convert to DataFrame and save
if data_rows:
    df = pd.DataFrame(data_rows)
    df.to_csv(output_csv, index=False)
    logging.info(f"Saved features to {output_csv}")
    logging.info(f"Shape: {df.shape}")
    logging.info(f"Columns: {list(df.columns)}")
    return df
else:
    logging.warning("No data extracted")
    return None

```

I note the matrix format might not be good to train model, so I ask AI to modify them, to calculate the matrix to three values:

RH the transformation matrix should be calculated to 3 numbers (like x, y, z, but I don't care which feature's real label, just give me 3 features)

I'll modify the code to extract 3 values from the transformation matrix. I'll use the translation components (x, y, z position) from the matrix:

Extract Face Blendshapes and Transformation Matrices

Code • Version 2



Perfect! I've updated the code to extract just 3 features from the transformation matrix:

- **transform_x**: First translation component
- **transform_y**: Second translation component
- **transform_z**: Third translation component

These 3 values represent the 3D spatial information from the transformation matrix.

Now your CSV will have:

- Metadata columns (video_path, label, frame_number, timestamp)
- All blendshape features (~52 columns)
- 3 transformation features (transform_x, transform_y, transform_z)

Much cleaner and easier to work with!

```
# Extract transformation matrix - convert to 3 features
if detection_result.facial_transformation_matrixes:
    matrix = detection_result.facial_transformation_matrixes[0]

    # Extract 3 values from transformation matrix
    # Using translation components (last column, first 3 rows)
    # These represent the 3D position/translation of the face
    row_data['transform_x'] = matrix[0][3] if len(matrix) > 0 and len(matrix[0]) > 3 else 0
    row_data['transform_y'] = matrix[1][3] if len(matrix) > 1 and len(matrix[1]) > 3 else 0
    row_data['transform_z'] = matrix[2][3] if len(matrix) > 2 and len(matrix[2]) > 3 else 0
```

I studied the snippet code and realized something wrong, the transformation matrix should be calculated to the value represent the rotation. So I ask AI to modify again.

RH the transformation matrix should be calculate in rotation, and there's are the feature I want:

```
def extract_features(video_path, label, frame_interval, model_path,
output_csv):
```

```
    """
```

Extract face blendshapes and transformation matrices from video frames.

Args:

video_path: Path to input video file

label: Label for this video (e.g., 'truth', 'lie')

frame_interval: Process every nth frame

model_path: Path to MediaPipe face landmarker model

output_csv: Path to output CSV file

```
    """
```

Eye gaze features (8 features)

```
eye_gaze_features = [
    "eyeLookUpLeft",
    "eyeLookUpRight",
    "eyeLookDownLeft",
    "eyeLookDownRight",
    "eyeLookInLeft",
    "eyeLookInRight",
    "eyeLookOutLeft",
    "eyeLookOutRight",
]
```

Eye blink features (2 features)

```
eye_blink_features = ["eyeBlinkLeft", "eyeBlinkRight"]
```

I'll update the code to calculate rotation angles (pitch, yaw, roll) from the transformation matrix and extract only the specific blendshape features you need:

```
# Extract head rotation from transformation matrix (pitch, yaw, roll)
if detection_result.facial_transformation_matrixes:
    # Get the 4x4 transformation matrix
    matrix = np.array(detection_result.facial_transformation_matrixes[0]).reshape(4, 4)

    # Extract rotation angles from rotation matrix (top-left 3x3)
    R = matrix[:3, :3]
```

```
# Calculate Euler angles (pitch, yaw, roll) in radians
pitch = np.arctan2(-R[2, 0], np.sqrt(R[2, 1]**2 + R[2, 2]**2))
yaw = np.arctan2(R[1, 0], R[0, 0])
roll = np.arctan2(R[2, 1], R[2, 2])

row_data['head_pitch'] = pitch
row_data['head_yaw'] = yaw
row_data['head_roll'] = roll
```

I didn't know does AI work right because of my bad math, but let it go for now.

Then I paste the structure of dataset, ask AI to handle them:

I'll modify the code to automatically process your dataset structure:

Extract Face Blendshapes and Transformation Matrices

Code • Version 3



Perfect! I've added a `process_dataset()` function that:

Features:

1. **Automatically scans** your `dataset/yes/` and `dataset/no/` folders
2. **Labels videos** based on their folder ("yes" or "no")
3. **Processes all videos** and saves individual CSV files per video
4. **Combines everything** into one `all_videos_combined.csv` file
5. **Sanitizes filenames** (handles spaces and special characters in your video names)
6. **Error handling** - continues processing even if one video fails
7. **Progress logging** - shows which video is being processed

Output:

- Individual files: `output_features/yes_1-10-18_21-15-40_features.csv`

• Combined file: `output_features/all_videos_combined.csv`

Training SVM Classifier

After extracting all features from the videos, I need to train a classifier. I asked AI to write a training script for me.

I told AI that I want to use SVM with RBF kernel, and I need to:

- Filter out frames where eyes are closed
- Split data into training and testing sets

- Perform hyperparameter tuning with GridSearchCV
- Evaluate the model with 7 metrics (Confusion Matrix, Accuracy, Precision, Recall, F1 Score, ROC Curve, AUC)

AI wrote the complete `train_svm.py` script for me, which includes:

- Data loading and preprocessing
- Feature scaling with StandardScaler
- GridSearchCV for hyperparameter tuning (C and gamma)
- Model evaluation with all 7 metrics
- Visualization generation (confusion matrix and ROC curve)
- Model saving

The script worked very well and achieved 97.7% accuracy on the test set.

Real-time Testing Program

I was curious whether the model was overfitting, so I asked AI to write a real-time webcam prediction program.

I asked AI to create a program that:

- Loads the trained model and scaler
- Captures frames from webcam
- Extracts features in real-time using MediaPipe
- Predicts whether I'm watching the screen
- Displays the result with a nice UI

AI wrote the complete `realtime_prediction.py` script for me, which includes:

- Real-time face detection and feature extraction
- Prediction with smoothing (moving average)
- Visual feedback with color-coded status
- Probability display
- Keyboard controls (quit, screenshot, reset)

The real-time program works really well and confirmed that the model is not overfitting. It correctly detects when I'm watching the screen or looking away, even when my head is turned but my eyes compensate to look at the screen.