Speech:

Speech:

Transcript Making:

ASR: automatic speech recognition

Basically what we are trying to do here is to 1) get voice of 2 people 2) and then transcript it.
For number 1 we are goona use speaker diarization. That will ask question that who's voice is this, and it will give output like "Speaker 1 spoke from 0.0s to 5.2s," and "Speaker 2 spoke from 5.3s to 8.9s."
After that for that particular amount of time we will use ASR model to transcribe the speech. =

For diarization we will use Pyannote.audio and for ASR we will use OpenAI's whisper
Below is speaker diarizaiton using assemblyAI
TO get the key:
https://www.assemblyai.com/?utm_source=youtube&utm_medium=referral&utm_campaign=yt_smit_25

Code: https://www.assemblyai.com/docs/speech-to-text/pre-recorded-audio/speaker-diarization

After getting the transcripte we can use a self supervised model to learn the tone and all of our learners.
For that we can use a model like Wav2vec.2. This could be used for extract features.

https://www.assemblyai.com/docs/speech-to-text/pre-recorded-audio/speaker-diarization

Table 1 (continued)

Construct	Marker	Behavior
Audio Features: Speech		
Communication	Speech Activity	The average percentage of time a team was speaking over the entire duration of the task
Communication	Silence	The average percentage of time a team was silent over the entire duration of the task
Communication	Short Pauses	The average percentage of time a team pauses briefly (0.15 sec) over their speech activity
Communication	Long Pauses	The average percentage of time a team made long pauses (1.5 sec) over their entire speech activity
Communication	Speech Overlap	The average percentage of time the speech of the team members overlapped over the entire duration of the task
Communication	Overlap to Speech Ratio	The ratio of the speech overlap over the entire activity

Tone:

To decide the tone we can use wav2vec library.

- **Pitch (Fundamental Frequency or F0):** This is the highness or lowness of the voice. A higher pitch can indicate excitement, anxiety, or questions, while a lower pitch can indicate seriousness or sadness.
- **Loudness (Intensity or Energy):** This measures how soft or loud the voice is. Changes in loudness can signal emphasis or changes in emotional state.
- **Speaking Rate:** The pace at which someone speaks. A fast pace might suggest urgency or enthusiasm, while a slow pace can indicate careful thought or hesitation.

You can also look at the **prosody**, which is the rhythm, stress, and intonation of a person's speech.

Wav2Vec can be used for 1) Feature extraction and after that for 2) classification.
Wav2Vec gives us a vector representation of our speech. That will mostly have the imp parts of the speech. After that to decide the tone we need to create our own classification model and label some data on our own and then train this model and then testing will give us the tone.

Basically we can use different models, libraries, softwares to get the features that are imp. Then after extracting theses features we can feed them into a simple classifier and get the tone.

Libraries that we could use: Wav2vec, Praat, Librosa.

If you use Librosa: It will give you an average pitch, loudness, and MFCCs for that 5-second clip.
If you use Wav2vec: It will give you a single feature vector representing the overall acoustic qualities of that segment.
Output: A data table with each row containing the speaker, timestamp, and the extracted features.

Speech overlapping :

First just do speech diarization, get all the stamps. The main advantage by using pyannote.audio is that we will get to know that 2 speakers are talking. After that for that stamp of time we can get the segment of both speakers and then decide.

Tone: Use `Librosa` or `wav2vec` to analyze the tone of the overlapping speech. Is the pitch high and loud (competitive) or low and soft (cooperative)?
Content: You can transcribe the overlapping audio as a single block of text and look for keywords. Does the transcript show phrases of agreement ("yeah, exactly") or disagreement ("no, that's not right")?
Pauses and Pace: Analyze if the overlapping speech occurs during a pause (cooperative) or in the middle of a continuous sentence (competitive).

Short pause and long pause:

- **Short Pauses (e.g., < 1.5 seconds):** These are often a normal part of speech. They can be for breathing, for punctuation (like a comma in a sentence), or for slight hesitation as the speaker plans their next word. They are a sign of normal cognitive processing and can actually improve a listener's understanding.
- **Long Pauses (e.g., > 1.5 seconds):** These are much more significant. A long pause can indicate **cognitive effort**, as a student tries to think deeply or solve a difficult problem. It can also signal **uncertainty** or **confusion** as they struggle to find the right words or figure out the next step. In a collaborative setting, a long pause can also be a signal that a student is waiting for their partner to take a turn.

You can use the duration of a pause to directly infer meaning. For example, a student who uses a lot of long pauses might be a "calm thinker" who is working through a problem internally.

Analyzing both short and long pauses is a great way to understand a student's cognitive state. The duration of a pause reveals a lot about the speaker's thought process. You can extract these features by identifying periods of silence that occur *within* a single speaker's turn.

The Meaning of Pauses

- **Short Pauses (e.g., < 1.5 seconds):** These are often a normal part of speech. They can be for breathing, for punctuation (like a comma in a sentence), or for slight hesitation as the speaker plans their next word. They are a sign of normal cognitive processing and can actually improve a listener's understanding.
- **Long Pauses (e.g., > 1.5 seconds):** These are much more significant. A long pause can indicate **cognitive effort**, as a student tries to think deeply or solve a difficult problem. It can also signal **uncertainty** or **confusion** as they struggle to find the right words or figure out the next step. In a collaborative setting, a long pause can also be a signal that a student is waiting for their partner to take a turn.

You can use the duration of a pause to directly infer meaning. For example, a student who uses a lot of long pauses might be a "calm thinker" who is working through a problem internally.

How to Extract Pauses with Code

You can use a library like **Pydub** to easily detect pauses within an audio segment after you have used speaker diarization to isolate each person's turns.

Here is a conceptual workflow:

1. **Isolate a speaker's audio turn:** Use the timestamps from your diarization model (`pyannote.audio`) to get a short audio clip containing only one speaker.
2. **Detect silence within that turn:** Use `pydub.silence.detect_silence()` to find all the silence periods within that clip. You can set a threshold for what you consider to be "silent" and a minimum silence length.
3. **Categorize the pauses:** Loop through the detected silence periods. If a silence period is shorter than your threshold (e.g., 1.5 seconds), you would label it as a "short pause." If it's longer, you label it a "long pause."

Using librosa to give handcrafted features, here we are using the waveform to extract features.

How MFCC is calculated: basically, we do fourier transforms of the audio and then it is passed through mel scale filters .

MFCC (Mel-Frequency Cepstral Coefficients) is one of the most common and powerful audio features used to represent voice patterns in speech and audio analysis.

It transforms a raw audio signal (waveform) into a compact set of features that describe how the human ear perceives sound — focusing on the spectral shape rather than raw frequencies.

MFCCs capture these spectral envelopes — i.e., how energy is distributed across frequencies — and thus represent voice identity, tone, and pronunciation.

So MFCCs essentially describe "what your voice sounds like", not just the pitch or loudness.

Basically, it gives us the waveform kind of thing, which shows how energy is distributed across the frequency.

These features (MFCC, Chroma, Spectral Centroid, Spectral Bandwidth, ZCR, RMS Energy) are chosen because **they together capture the main acoustic properties that describe how humans speak and how emotions or personality are reflected in voice**.

Feature	Technical Description	What It Captures in Voice	Why It's Important
MFCC (Mel-Frequency Cepstral Coefficients)	Represents short-term **power spectrum** on a **mel scale**, mimicking how humans perceive sound frequencies. Extracted from the **log-mel spectrogram** after applying DCT(Discrete Cosine Transform).	Describes **timbre** (tone quality), **vocal texture**, and **phoneme characteristics** — the "fingerprint" of how someone sounds.	Because MFCCs encode *how the shape of the vocal tract* changes, they can distinguish **speakers, emotions, and phonetic content**. Smooth MFCC → calm voice Abrupt MFCC changes → emotional, excited, or stressed voice.
Chroma (Chroma-STFT)	Maps energy into **12 pitch classes** (C, C#, D, …, B), ignoring octave differences.	Represents **harmonic and pitch content** — tonal aspects of speech or music.	Useful for **melodic patterns**, **intonation**, and **emotional speech cues**. Rising pitch (questions, surprise) Falling pitch (sadness, calm).
Spectral Centroid	The **center of mass** of the spectrum — weighted average frequency of the sound's power distribution.	Indicates **brightness** or **sharpness** of voice.	Higher centroid → **bright, sharp, energetic** (anger, excitement). Lower centroid → **warm, dark, relaxed** (sadness, calm).
Spectral Bandwidth	Measures **spread of frequencies** around the spectral centroid.	Reflects **complexity** or **noisiness** of sound.	High bandwidth → **noisy, harsh, stressed** tone. Low bandwidth → **smooth, steady, composed** tone.
Zero Crossing Rate (ZCR)	Counts how often the signal crosses zero amplitude per frame — i.e., how frequently the signal changes sign.	Indicates **noisiness** or **unvoiced energy**.	High ZCR → **breathy, tense, or agitated** speech (anger, fear). Low ZCR → **steady voiced** sounds (neutral, calm).
RMS Energy	Root-mean-square of amplitude (signal power).	Represents **loudness** and **overall energy**.	High RMS → **energetic, loud, emotional** speech. Low RMS → **soft, low-energy** tone (sadness, fatigue).

Category	Feature Examples	Captured Aspect
Spectral Features	MFCC, Spectral Centroid, Spectral Bandwidth	Shape and texture of the sound spectrum — tells *what kind of sound* it is (timbre, brightness, roughness).
Pitch-related Features	Chroma	Captures *intonation*, pitch variation — important for detecting *emotion*, *stress*, or *questioning tone*.
Temporal Features	ZCR, RMS Energy	Capture *temporal dynamics* and *rhythmic energy* — how the voice evolves in time (smooth or abrupt, calm or agitated).

action

1. A Real-Time 3-Dimensional Object Detection Based Human Action Recognition Model

This paper builds a real-time human action recognition system. Here's what it does:

- Detects humans in each video frame using YOLOv6.
- Extracts a short video clip around the detected human.
- Feeds the clip into a 3D CNN, which learns both:
- Spatial features (e.g., shape, pose),
- Temporal features (how things change over time in the clip).
- The features are passed to a multiplicative LSTM, which models longer-term action patterns.

They also use a skeleton-based module that explicitly analyzes the human joints and body movement patterns.
The outputs are combined and passed to a classifier to predict the action (e.g., walking, waving, running).
They evaluate on standard action datasets and show high accuracy + fast runtime.
They handle *occlusions, shadows, blur, background clutter, imbalanced data*.
Their solution:

Four-phase HAR pipeline with:

- Object identification (YOLOv6 finds humans).
- Skeleton articulation (joint-based pose features). - OpenPose
- 3D CNN (spatiotemporal features from clips). - The first module is 3D CNN with multiplicative LSTM (Fig. 1), a model that extracts features with CNN and provides them as input to layers of multiplicative LSTM. LSTM learns long-term and short-term dependencies
- Multiplicative LSTM (temporal sequence modeling).

Innovation: They combine 3D CNN + multiplicative LSTM (special RNN with factorized hidden transitions, more efficient than vanilla LSTM).

Datasets used:

- KITTI: real-world driving dataset, here used for object detection.
- NTU-RGB+D: ~57k videos, 60 action classes.
- NTU-RGB+D 120: ~114k videos, 120 classes.
- UCF101: ~13k YouTube videos, 101 action classes.

Fusion: They combine these into one giant training set

I don't think this aligns much with our dataset but we can build it from scratch for only hand movements. Accuracy is 96% on UCF101 dataset

Model is heavy (combines CNN + LSTM + skeleton → computational cost).
Needs multiple large datasets fused not always practical.
Focused mostly on full-body actions, not fine-grained ones like only hand gestures.

Class	Action
Assisting	Getting up, walking, holding another person, passing something
Baby Crying/Crawling	Baby is crawling or crying
Drinking	Drinking something
Eating	Eating or cutting food or meal
Exercising	clapping, bowing, jumping, boxing, shaking, bending
Massage	Massaging head, body
Moving	Entering or moving out of room or lawn
Opening something	Opening or closing something
Personal care	Brushing, drying hair, manicuring
Playing	Playing some game
Putting cloths	Putting on or off clothes
Reading/writing	Reading or writing on paper
Watching TV/Rest	Doing rest or watching TV
Sewing	Sewing something
Standing/smoking	Smoking while standing
Throwing	Throwing something
Listening music	Listening music on headphone
Washing something	Washing dish, washing face
Wearing	Wearing glasses, gloves, shoes

First, you **spot the person** (YOLOv6).

Then, you **trace their body outline and joints** (Skeleton articulation).

You also **watch a short clip** of what they're doing to see the motion (3D CNN).

You **remember the sequence of movements** (LSTM).

You **combine all this evidence** (Fusion).

Finally, you **decide what action they're doing** (Classifier).

2. YOLO Series for Human Hand Action Detection and Classification from Egocentric Videos

- The paper focuses on hand detection and classification.
    - 3D hand pose estimation (figuring out finger positions in 3D space),
    - Hand activity recognition (e.g., waving, pointing, grasping).

They work on egocentric vision datasets → video recorded from the wearer's perspective (like from a head-mounted camera).

- EV datasets they used:
    - FPHAB (First-Person Hand Action Benchmark),
    - HOI4D (Hand-Object Interaction 4D dataset),
    - RehabHand (rehabilitation-focused dataset).

Egocentric data is tricky because Hands appear very close to the camera, Frequent occlusions (objects covering the hand), Different angles and lighting.
They propose a study comparing YOLO networks for hand detection. Specifically:

1. Review all versions YOLOv1 → YOLOv7, highlighting their architectures, advantages, disadvantages.
2. Label hand bounding boxes in datasets for training/evaluation.
3. Adapt YOLO-family models for hand detection on EV datasets.
4. They take multiple versions of YOLO (v1–v7).
5. They fine-tune them on hand detection tasks.
6. They evaluate precision (P), speed (frames per second), and robustness across datasets.

- Best model = YOLOv7 and its variants.
- YOLOv7-w6 (a wider version of YOLOv7) gave the strongest results:
    - FPHAB → Precision = 97% @ IoU = 0.5
    - HOI4D → Precision = 95% @ IoU = 0.5
    - RehabHand → >95% @ IoU = 0.5
- Speed trade-off:
    - YOLOv7-w6 → 60 fps (slower, but high accuracy, input 1280×1280).
    - YOLOv7 → 133 fps (much faster, smaller resolution 640×640, still strong accuracy).

They design a **3-step pipeline** for hand rehab evaluation:

1. **Input** → Video sequence from *egocentric vision (EV)* dataset.

    - EV = recorded from **first-person view** (wearable cameras).
2. **Processing** → Multiple steps:
    - **Hand detection** → finding where the hand is in the frame.
    - **Hand tracking** → following the hand across frames.
    - **2D pose estimation** → finger positions in 2D image.
    - **3D pose estimation** → hand/finger structure in 3D.
    - **Hand activity recognition** → classify what movement is happening (e.g., grasping, opening).
3. **Output** → Quantified action ability of the hand (numerical assessment of movement capacity).

EV datasets are collected from **the patient's view**.This means: Fingers are often **occluded** (covered by the hand itself or by objects). Sometimes only the **back of the hand** is visible. Most older research used **third-person datasets** (NYU, ICVL, MSRA) where the camera has a **clear view** of the hand. In EV datasets, data is **messier and harder** → poses are harder to estimate.

They tried using **Google MediaPipe (GM)** for hand detection/classification.On EV datasets (FPHAB, HOI4D): **Pre-trained GM models performed poorly.** Sometimes **failed to detect the hand entirely**. So existing tools don't work reliably for EV data.

YOLO's Idea: Do everything in **one stage** → detect + classify objects in a single pass.
Process: Split image into a grid. Each grid cell predicts bounding boxes + confidence + class.
Train the model to minimize error in box position, size, confidence, and class.
Clean up duplicates with **Non-Maximum Suppression**.
Rank all boxes by confidence score.
Keep the highest one.
Remove other boxes that overlap too much (measured by **IOU = Intersection Over Union**). Repeat until only the best boxes remain.
YOLOv1: First version, fast but struggled with small objects.
YOLOv2: Added batch norm, higher-resolution training, anchor boxes, and Darknet-19 backbone → much better accuracy and speed.
YOLOv3: Deeper Darknet-53 + multi-scale detection + better anchor boxes → very good at detecting objects of all sizes.

Each version of YOLO improves the backbone (feature extractor), neck (feature fusion), and head (final predictions). YOLOv7 is currently the best balance between speed and accuracy.YOLOv7 introduced **4 key upgrades** (E-ELAN, MSCM, re-parameterization, new label assignment). These make it the most efficient and accurate YOLO yet. The paper then applies this family (all versions) to **EV datasets** for hand detection & classification, and shows YOLOv7 wins.**Datasets**:**FPHAB** → evaluate detection + classify action hands/background/objects.
HOI4D & RehabHand → evaluate detection + classify left/right hands, background, objects.Datasets Used
The paper evaluates YOLO (v1–v7) on three EV (Egocentric View) datasets:

1. FPHAB (First Person Hand Action Benchmark)
    - Captured using an Intel RealSense SR300 camera (RGB + Depth).
    - Resolution: 1920×1080 RGB, 640×480 depth.
    - Hand pose tracked with 6 magnetic sensors, annotated in 3D (21 joints per hand).
    - 6 subjects, each performing 45 actions, repeated 3–9 times.
    - Dataset split:
        - Test: 1st sequence of each subject (27k samples).
        - Validation: 2nd sequence (25k samples).
        - Train: Remaining sequences (53k samples).
        - Approx ratio → 1:2.5 (test:train).

    2. HOI4D

    - Large-scale Human-Object Interaction dataset.
    - Captured with Kinect v2 + Intel RealSense D455.
    - 2.4M frames, 4000+ sequences, 9 participants.
    - 800 object instances, 16 categories, 610 indoor rooms.
    - Provides rich annotations:
        - Panoptic segmentation
        - Motion segmentation
        - 3D hand pose (21 joints)
        - Object pose & hand action
        - Scene point clouds
    - For evaluation → bounding boxes are derived from 2D hand keypoints in the 3D pose annotations.
3. RehabHand
    - Collected at Hanoi Medical University Hospital, Vietnam.
    - First-person videos captured with GoPro Hero4 (1080p, 30fps).
    - 15 patients, 4 rehab exercises (ball, bottle, cube, cylinder).
    - Each patient performs each exercise 5 times.
    - Dataset size: 53 GB, 10 video files, ~4 hrs total.
    - Split:
        - Train: 2220 images (60%)
        - Validation: 740 images (20%)
        - Test: 740 images (20%).

Evaluation Metrics

They use standard object detection metrics:

1. IOU (Intersection over Union) → measures overlap between predicted & ground-truth bounding boxes.
    - IOU = Area of overlap ÷ Area of union.
    - If IOU ≥ threshold → True detection, else false.

2. Precision (P), Recall (R), F1-score (F1) → for classification (hand action, left/right hand, background).
    - Precision (P) = TP ÷ (TP + FP) → how many predicted hands are correct.
    - Recall (R) = TP ÷ (TP + FN) → how many true hands were detected.
    - F1 = harmonic mean of P and R.
3. mAP (mean Average Precision) → averages precision across all classes.
    - Used widely in detection benchmarks (like COCO, Pascal VOC).

Training Setup

- Hardware: NVIDIA RTX 2080 Ti (12 GB).
- Frameworks/Libraries: Python 3.7+, CUDA 11.2, cuDNN 8.1.0, OpenCV, Numpy, Scipy, Pillow, Cython, Matplotlib, Scikit-image, TensorFlow ≥1.3.

Object Interaction Hand Actions

(Hands interacting with objects)

1. Open book
2. Close book
3. Flip pages
4. Pour water from bottle
5. Drink from bottle
6. Open bottle cap
7. Close bottle cap
8. Push button
9. Type on keyboard
10. Press light switch
11. Turn on tap
12. Turn off tap
13. Pick up phone
14. Put down phone
15. Answer phone (hold to ear)
16. Hang up phone
17. Pick up cup
18. Put down cup
19. Drink from cup
20. Pour from cup
21. Pick up spoon
22. Stir with spoon
23. Eat with spoon
24. Pick up pen
25. Write with pen
26. Put down pen
27. Pick up ball

28. Throw ball
29. Catch ball
30. Squeeze ball

Hand-only Actions (no object)

(Only gestures / free-hand movements)
31. Wave hand
32. Point with finger
33. Thumbs up
34. Thumbs down
35. OK gesture
36. Stop gesture (palm open)
37. Clap hands
38. Scratch head
39. Rub hands
40. Cross arms
41. Snap fingers
42. Pinch fingers
43. Expand fingers (spread out)
44. Rotate wrist
45. Fist (make a fist)

Meanwhile, the **HOI4D** and **RehabHand** datasets do **not** have 45 actions — they just classify **left hand / right hand / background / object**.

If we fine tune this we can also calculate duration using the LSTM/transformer on this pipeline maybe by seeing in how many frames the object is held or the action is done.

FPHAB Dataset **YOLOv7 & variants > 95% accuracy** even at **IoU = 0.95** (very strict threshold). Precision (P) > Recall (R) in most cases, because sometimes background gets misclassified as hand.

HOI4D Dataset **Precision (P): 90.55%**
Recall (R): 89.85%
mAP@0.5: 88.9% (mean Average Precision at IoU=0.5).

RehabHand Dataset
YOLOv7 (Left Hand): P = 100%, R = 92.1%, mAP@0.5 = 94%
YOLOv7-X (Right Hand): P = 87.7%, R = 92.5%, mAP@0.5 = 96.7%

3. Hand Detection Tracking in Python using OpenCV and MediaPipe

Image → Segmentation → Tracking → Feature Extraction → Classification → Action Output.

The system first detects the palm → maps hand landmarks → classifies gesture.

No accuracy or anythin mentioned justa small hand tracking project with opencv and mediapipe. Gave functions and code to use.
https://gautamaditee.medium.com/hand-recognition-using-opencv-a7b109941c88

3. Res3ATN - Deep 3D Residual Attention Network for Hand Gesture Recognition in Videos

- They use a 3D residual attention network (Res3ATN).
- This network has stacked attention blocks (each block focuses on different discriminative features).
- It is end-to-end trainable and can be scaled to deep layers.

Comparisons: They test Res3ATN against popular 3D networks:

- C3D (basic 3D CNN),
- ResNet-10,
- ResNeXt-101.
  Findings: With 3 attention blocks, their network is more robust and accurate.
  Outperforms existing networks on three public datasets.

Methodology:

3D CNNs: Capture **spatial and temporal features** from video sequences. Important because gestures are dynamic and need temporal modeling.

Residual Network (ResNet): Allows **very deep networks** without vanishing gradient problems. Helps in learning more complex features, which is essential for gestures.

Residual Attention Network (Res3ATN): Uses **3D attention blocks** to focus on regions of interest in each frame. Each attention block has a **trunk** (main residual features) and a **mask layer** (soft attention weights). Multiple blocks help **correct wrong attention predictions**, making it robust. Outputs weighted features to identify the gesture in space and time.

Output and Training Fully connected layers at the end classify gestures like moving the hand **up, down, left, or right**, etc. Soft attention ensures the network focuses on relevant hand regions.

1. Input Video

2. Preprocessing

3. 3D Convolutional Layers

4. Residual Blocks

5. Attention Blocks (3D Residual Attention) Each block has two parts:

1. Trunk Layer
    - Main residual network output features.
2. Mask Layer
    - Soft attention mask that weights the trunk features to focus on important regions, e.g., the moving hand.

- Multiple attention blocks are stacked to:
    - Focus on different regions at different levels.
    - Correct wrong attention predictions from previous blocks.

6. Weighted Feature Output

7. Fully Connected Layers

8. Softmax Classifier

9. Output

Accuracy: Top 5 - 81
Top 1- 62

emotions

1. Facial expression recognition in videos using hybrid CNN & ConvLSTM

propose a 3D-CNN + ConvLSTM pipeline:

- Feature Extraction: 3D-CNN extracts spatiotemporal features from short video clips (e.g., sequences of 16 frames).
- Sequence Modeling: The extracted features are fed into a ConvLSTM that models long-term temporal dependencies across frames without losing spatial details.
- Classification: A final classifier (usually softmax) predicts the facial expression class.

This hybrid design ensures: 3D-CNN handles local motion + appearance (smiles forming, eyebrows raising). ConvLSTM handles longer temporal context (subtle transitions in emotions). ConvLSTM preserves spatial information in videos, making it a natural choice for VFER.

Typical FER Pipeline has **three major stages**:

1. Face Detection (and Alignment)
2. Feature Extraction: traditional hand-crafted methods : These hand-engineered features are **sensitive to lighting, occlusion, and ethnicity variations** don't generalize well.Deep learning solves this by learning **trainable features** end-to-end.
3. Feature Classification

Early methods: Relied on **hand-crafted features** (LBP, LBP-TOP, HOG) + ML classifiers.
Intermediate methods: Hybrid CNN + RNN (or LSTM) networks better temporal modeling.
Recent methods: Advanced hybrids (CNN + RNN + GCN, attention, fusion of modalities) more accurate but computationally **complex**.

Pipeline:
Input Pre-Processing Unit

1. Face Detection: Uses Single Shot Multibox Detector (SSD) with MobileNetV1 backbone. Trained using transfer learning on the WIDER FACE dataset. Runs at 60 FPS with high accuracy.
2. Facial Landmark Detection: Detects 68 key landmarks (eyes, nose, mouth, jawline). Uses the open-source implementation from Dlib (King [38]) with regression trees.
3. Face Alignment: Align faces based on landmarks (e.g., rotate/shift so eyes are horizontally aligned). Ensures spatial symmetry across frames.
4. Normalization: Faces are cropped + resized to 64×64 pixels.Output is a standardized sequence of aligned face images ready for feature extraction.

Hybrid 3D-CNN + ConvLSTM Model

1. 3D-CNN Backbone: **3D-CNN** adds a **temporal dimension**, enabling it to capture **motion** across frames (e.g., how a smile gradually forms). Input: A sequence of video frames → forms a **video cube** I(x,y,t). Kernel: 3D filter W(a,b,c) slides over x (width), y (height), and t (time). Output: Feature map Z(x,y,t) containing **spatiotemporal features**.
2. ConvLSTM Block: Input: Feature maps from 3D-CNN. Internals: Uses convolutional gates (input gate It, forget gate Ft, output gate Ot, and cell state Ct). Maintains **spatial + temporal memory**.

3. Model Architecture: **3D Convolution Layer 1** → ReLU; **3D Convolution Layer 2** → ReLU → **3D Max Pooling; 3D Convolution Layer 3** → ReLU; **ConvLSTM Block (16 units); Flatten** → **Fully Connected Layer; Softmax Classifier** → Outputs one of 7 classes: **Anger, Contempt, Disgust, Fear, Happiness, Sadness, Surprise**

4. Video Input → Face Detection + Alignment → Resized Faces → 3D-CNN → ConvLSTM → FC → Softmax → Expression Class.
5. They use **three datasets**: CK+, SAVEE, and AFEW
6. For AFEW they have done some extra pre processing because of extra noise.Training details

- **Framework**: Keras, running on **NVIDIA Tesla K80 GPU**.
- **Optimizer**: Adam, learning rate = **0.0001** (small LR = stable training).
- **Loss function**: **Categorical cross-entropy** (standard for multi-class classification).
- **Metric**: **Accuracy**.
- **Training schedule**: 80 epochs → select model with **highest validation accuracy**.

CK+ Dataset : The proposed model achieved **>95% accuracy**
SAVEE Dataset (Surrey Audio-Visual Expressed Emotion): Achieved **state-of-the-art accuracy**
AFEW Dataset (Acted Facial Expressions in the Wild): Accuracy: 83.12%.

Deepface: basically a pretrained cnn model.
-face detection
- region or interest
-preprocessing

-model selection(mostly vgg or something)
-feature extraction for the given frame
-prediction

State	Key Facial Markers (AUs)	Link to Learning
Engagement	**AU06/12** (Cheek Raiser/Smile)	Frequent, brief smiles can indicate *aha* moments or satisfaction with progress.

Confusion/Effort	**AU04** (Brow Lowerer), **AU43** (Eyes Closed)	**Sustained AU04** is a primary marker of high cognitive load, effort, or being stuck ("Hmm" moment).
Frustration/Anger	**AU09** (Nose Wrinkler), **AU17** (Chin Raiser)	Indicates a negative affective state in response to an obstacle, often leading to disengagement if prolonged.
Attention	**AU45** (Blink Rate) and **Gaze**	A low blink rate and sustained gaze toward the task suggest active attention; high deviation suggests distraction or *silent wandering*.

When you run **MediaPipe Face Mesh**, it gives you a **list of 468 3D landmark points** for each detected face.

Each landmark corresponds to a **fixed point on the human face** — for example:

- the tip of the nose,
- the corner of the mouth,
- the edge of the eye,
- etc.

These landmarks are indexed from **0 to 467**, and their positions are **standardized** by MediaPipe's face mesh model.

What do the 468 coordinates represent?

Each frame gives you:

- 468 × (x, y, z) = **1,404 numbers**
- Each triplet corresponds to a **specific facial landmark**, defined by MediaPipe's 3D face mesh model.

Region	Approximate Landmark Range	What It Represents
Face outline / jawline	0–16	Face contour / head movement
Eyebrows	70–105 (left), 300–334 (right)	Brow raise / frown
Eyes	33–133 (left), 362–263 (right)	Eye openness / blinking
Nose	1–10, 168–200	Center reference for pose
Mouth / Lips	13–14, 61–291, 308–402	Smiling, talking, mouth open
Cheeks	~50–55, 280–285	Cheek puff or smile lift
Chin	152	Lower face reference point
Forehead	10–19	Eyebrow and upper movement

So each coordinate traces a specific part of facial motion and shape.

Each facial "action unit" (AU) corresponds to one or more muscles:

- AU12 = Lip Corner Puller → **Smiling**
- AU4 = Brow Lowerer → **Frowning**
- AU25, 26 = Mouth Open
- AU5, 7 = Eye Opening/Widening
- AU1, 2 = Brow Raising

Now:

- **Positive Valence (pleasant)**: ↑ AU12 (smile), ↓ AU4 (frown)

- **Negative Valence (unpleasant)**: ↑ AU4, ↑ AU9/10 (nose/mouth tension)
- **High Arousal**: ↑ AU5 (wide eyes), ↑ AU26 (open mouth)
- **Low Arousal**: ↓ AU5, ↓ AU26 (relaxed face)

AU	Proxy Landmark Distances	Affects
AU12 (smile)	Distance(61, 291) – lip corners	↑ Valence
AU4 (brow lowerer)	Distance(70, 300) – inner brows	↓ Valence, ↑ Frustration
AU25/26 (mouth open)	Distance(13, 14) – lips	↑ Arousal
AU5 (eye open)	Dist(159,145), (386,374) – lids	↑ Arousal, ↑ Engagement
AU1/2 (brow raise)	Distance(brow–eye vertical)	↑ Surprise, ↑ Arousal

work_done_in last 15 days

Work done in the last 2 weeks

SPEECH

Have written the final code.
We are getting 1) handcrafted vectors from librosa library for pitch, energy and other things
2) code for getting the audio snippet without silence, so using VAD(voice activity detection)
3) after getting the trimmed versions of audio we are using openAI/Wishper medium model to get the final transcript. Mostly, we have hindi and English audio for which this model works fine. So, we are transcribing everything in english. When they are talking in hindi, we are simply translating it and then transcribing.

4) after that, I am using sentence-transformers/paraphrase-mpnet-base-v2 to get the sentence embedding of the transcript. This model gives a 768-dimensional vector for each sentence. We are only getting semantic embedding here. No emotions are extracted from the speech from the transcript. But that shld be fine as we are using librosa to do that on a lower level.

After getting the embeddings we can do cosine similarity, or pca to see the similarity between the sentence. Which can kind of give us the evaluation of our model.

Where can we improve:
I am doing the transcript using openAI whisper's medium model, we can use the quantized version of the large model. I tried with that but not able to get any results. Because of the GPU memory storage issues. So to get accurate transcripts we can use that model.

Then, I am using sentence-transformers/paraphrase-mpnet-base-v2 for converting the transcript sentences into embedding. Its sentence embedding here. It just has the semantic embedding. It doesnt contain any expression embedding. So we can add some expression embedding also.

ACTION

EMOTIONS

EYE GAZE

Planning to do: go to the lab, run this for all the audio files during this week.
After this week: do the final multimodal evaluation.

Tab 5

FOR POSTER

Emotion Feature Extraction (Valence–Arousal Model)

Library used: media pipe

Why Valence & Arousal?

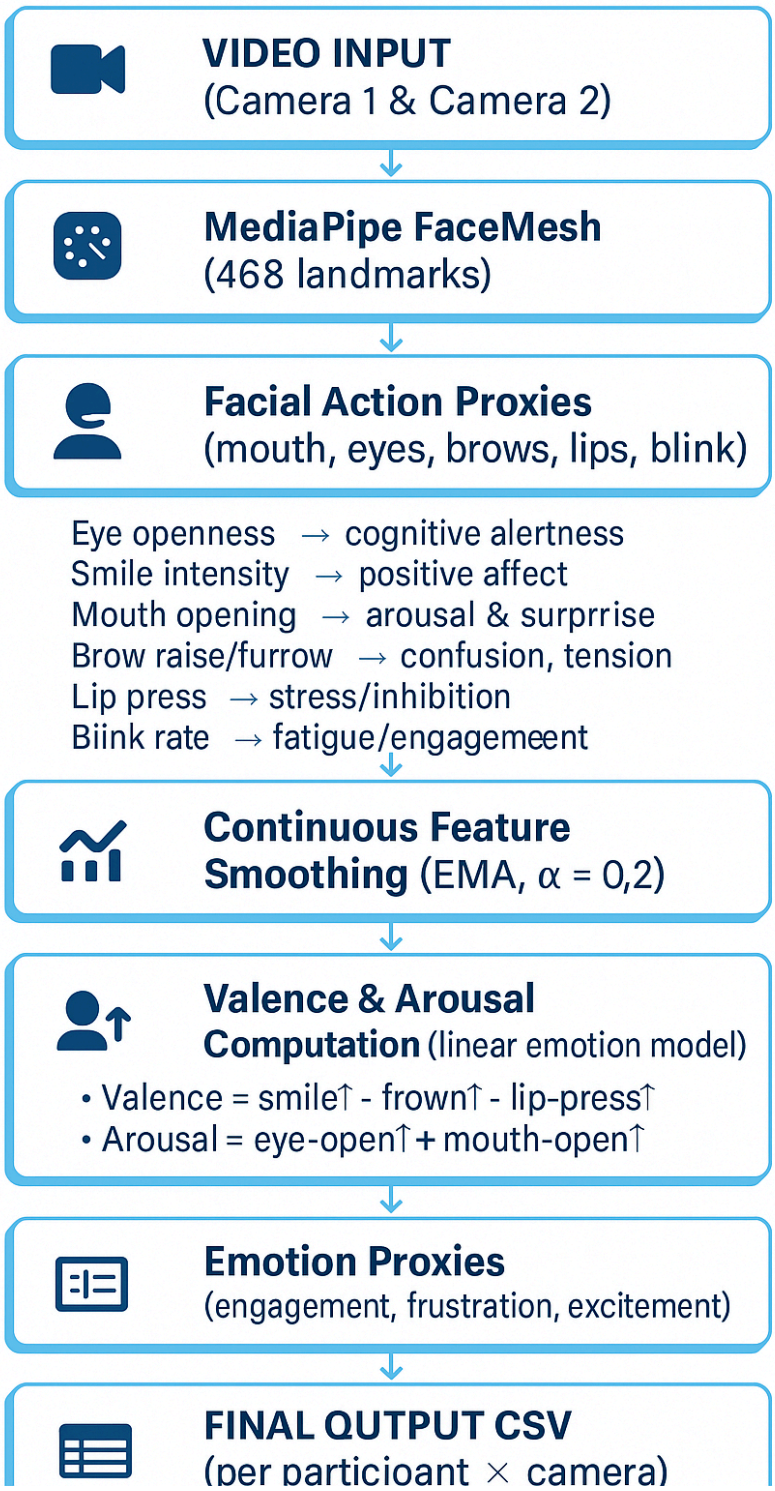
- **Compact & Comprehensive:** The valence–arousal space is a widely used, low-dimensional representation that captures the full emotional spectrum without needing many discrete labels.

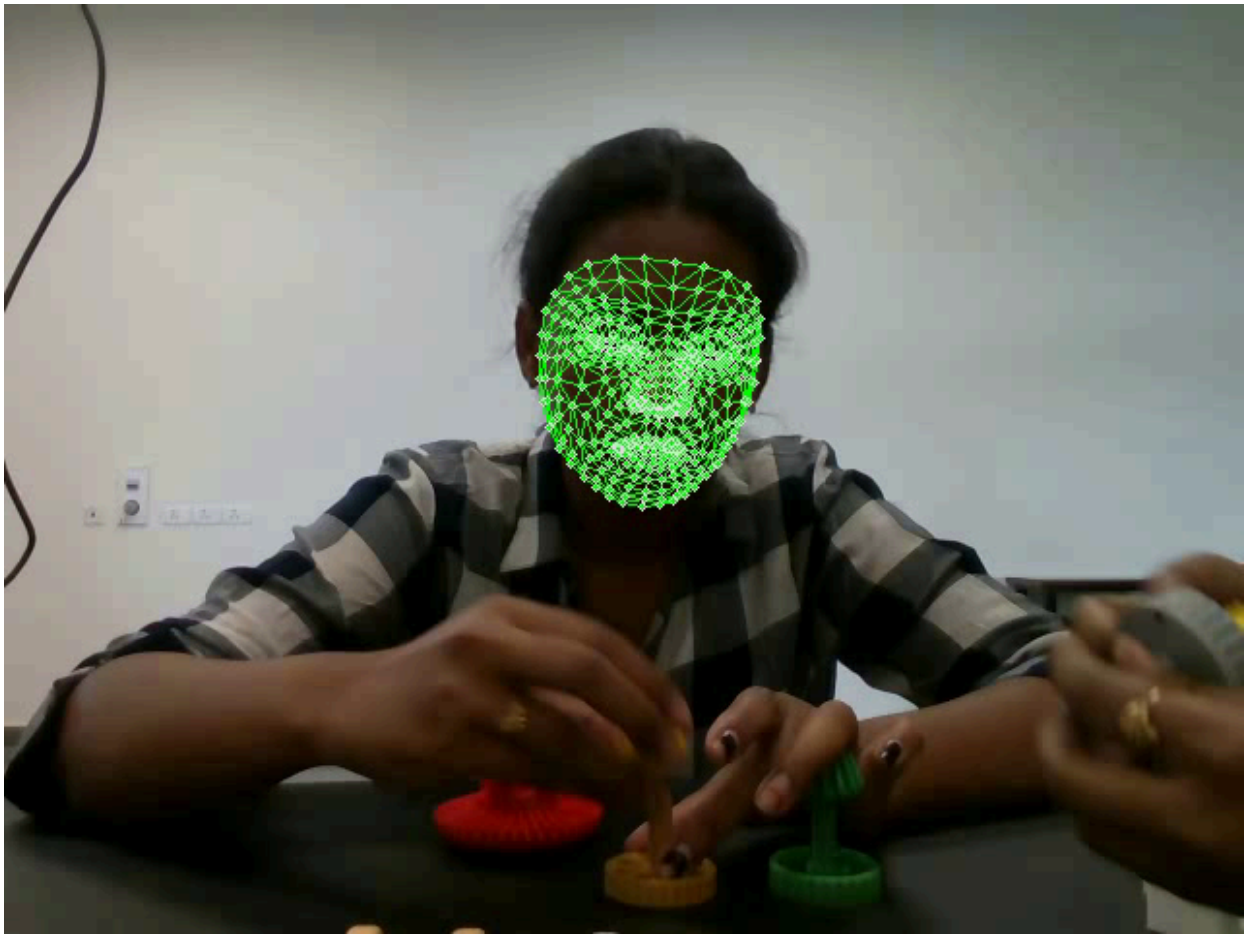
- **Stable Over Time:** Mean valence and arousal summarise the *overall affective climate* of an interaction, reducing noise from moment-to-moment facial fluctuations.

- **Reveals Learning Dynamics:**

    - Higher valence → positive affect, comfort, and smooth collaboration.

    - Higher arousal → active engagement, cognitive effort, or heightened focus.
      These metrics help us interpret *how participants react, learn, and coordinate* during collaborative tasks.


VIDEO INPUT
(Camera 1 & Camera 2)
MediaPipe FaceMesh
(468 landmarks)
Facial Action Proxies
(mouth, eyes, brows, lips, blink)
Eye openness → cognitive alertness
Smile intensity → positive affect
Mouth opening → arousal & surprrise
Brow raise/furrow → confusion, tension
Lip press → stress/inhibition
Biink rate → fatigue/engagemeent
Continuous Feature
Smoothing (EMA, α = 0,2)
Valence & Arousal
Computation (linear emotion model)
• Valence = smile↑ - frown↑ - lip-press↑
• Arousal = eye-open↑ + mouth-open↑
Emotion Proxies
(engagement, frustration, excitement)
FINAL QUTPUT CSV
(per particioant × camera)



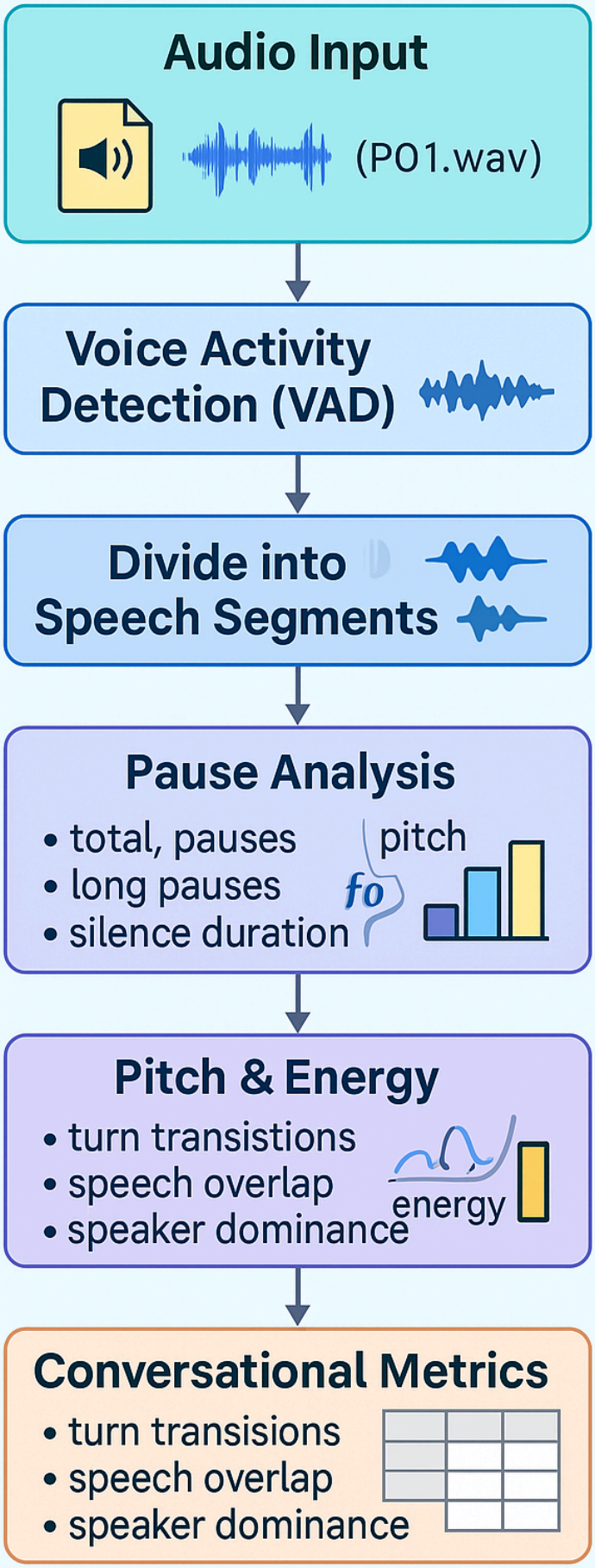
SPEECH

silero-vad model to get the time stamps

After that library used: librosa

Feature Group	Features Used	Why It Matters in Collaborative Learning

Pause-Based Prosody	- Total pauses- Short pauses- Long pauses- Avg pause duration- Total silence	• Indicates cognitive load, planning, hesitation,• Long pauses → potential confusion or low engagement• Silence patterns show active vs passive participation
Vocal Affect (Pitch & Energy)	- Avg pitch (Hz)- Avg energy	• Reflects arousal, confidence, or stress• High pitch/energy → excitement or urgency• Low values → calmness or reduced engagement
Participation Dynamics	- Total speech time- Dominance	• Measures balance of participation in the pair• Dominance asymmetry suggests unequal roles or confidence levels
Interaction Flow	- Overlap duration- Turns A→B- Turns B→A	• Captures coordination, responsiveness, synchrony• Smooth turn-taking = effective collaboration• High overlap = interruption or competition


Audio Input
(P01.wav)
Voice Activity Detection (VAD)
Divide into Speech Segments
Pause Analysis
• total, pauses
• long pauses
• silence duration
pitch
fo
Pitch & Energy
• turn transistions
• speech overlap
• speaker dominance
energy
Conversational Metrics
• turn transisions
• speech overlap
• speaker dominance

Eye Gaze

Libraries used:

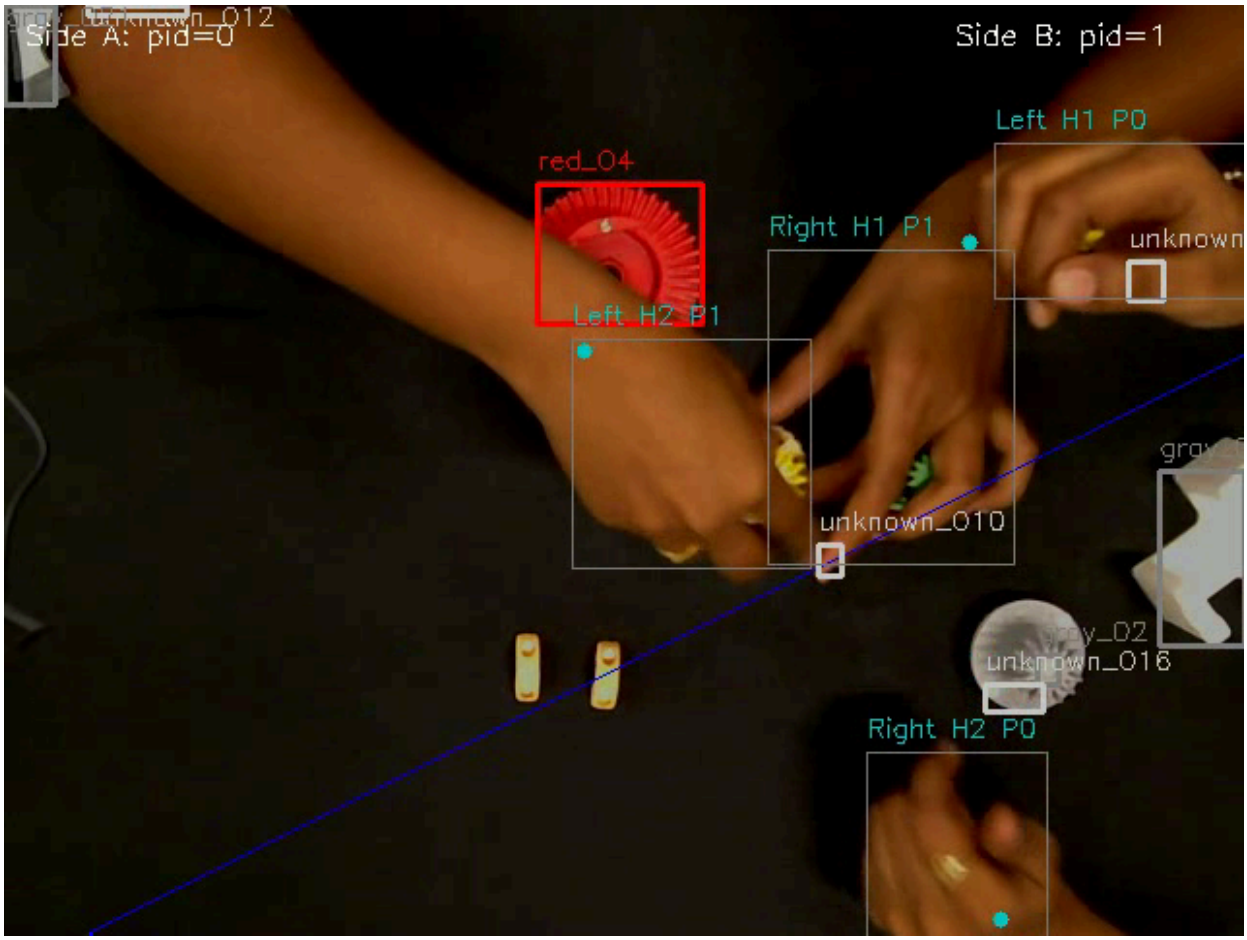
OpenFace →Used to detect face bounding boxes, head pose, and eye region positions.
It provides stable face localisation for every frame.

GazeLLE (DINOv2 ViT-L14) → A transformer-based model that generates high-resolution gaze heatmaps. It predicts where each participant is looking inside the frame.

Feature Group	Features Used	Why It Matters in Collaborative Learning
Fixation-Based Attention	**– Focus Duration on One Object (s)** **– Number of Fixations** **– Average Focus Area Spread**	• Long fixations → deeper concentration, sustained attention, problem-solving. • Number of fixations shows visual search behaviour. • Spread indicates precision vs uncertainty in attention.
Gaze Dynamics	**– Gaze Shifts per Second**	• High shift rate → active searching, exploration. • Too many shifts → distraction, confusion. • Low shifts → stable focus or cognitive load.
Task Engagement (Where They Look)	**– % Time Looking at Workspace**	• Primary indicator of task-focused engagement. • Higher % means sustained involvement in the making activity.
Social Engagement	**– % Time Looking at Partner**	• Reflects coordination and understanding of collaborator's actions • Indicates turn-taking, teaching moments, and joint planning.

Off-Task Behaviour	**– % Time Looking Elsewhere**	• Shows distraction, disengagement, or breaks in focus. • Useful for detecting cognitive fatigue or confusion.
Joint Attention & Coordination	**– Shared Attention Overlap Ratio**	• Measures similarity of gaze heatmaps between partners. • High overlap → strong synchrony, common focus, efficient collaboration. • Most direct marker of joint attention.

Hands



Component	Method Used	Notes
Hand Detection	MediaPipe Hands	Robust 21-landmark tracking
Object Detection	HSV color segmentation	More stable than YOLO for small colored cubes
Skin Removal	Subtract MediaPipe hand mask from HSV mask	Prevents skin being detected as an object
Distance Calculation	Landmark centroid → object centroid	Used for touch/reach detection
Interaction Classification	Threshold-based logic	Touch detected reliably; reach/grasp inconsistent

Feature Name	Description	How It Was Computed
total_interactions	Total number of hand–object interactions	Count of frames where hand-to-object distance < threshold
avg_interaction_duration	Average duration of each interaction	Mean length (in seconds) of continuous contact sequences
interaction_frequency	Interactions per minute	total_interactions / total_time_min
object_switch_rate_per_minute	How often participants switched objects	Count of transitions between different object IDs / minute

simultaneous_object_touch_ratio	How often both hands touched objects at the same time	(# frames both hands touching same objects) / (total frames)
workspace_coverage_ratio	How much of the workspace area the hands moved across	Area covered by hand centroid path / total workspace area

Challenges

YOLO failed to detect the small colored objects reliably.

Interaction types (reach/touch/grasp) could not be classified due to varied hand-holding styles and occlusions.

Eye gaze
Slow Face Detection with RetinaFace

Heavy GPU Memory Load

Speech

- **Background noise** makes it difficult to detect clean speech signals.

emotions

Subtle facial expressions are difficult to detect, especially during natural tasks.

Occlusions (hands, tools, head turns) often hide key facial cues.

Fast, dynamic movements reduce the accuracy of frame-level emotion detection.

Ambiguous expressions make it hard to classify emotions reliably.

Lighting variations affect consistency of face and

- **Overlapping speech** between partners complicates feature extraction (e.g., pauses, turn-taking).
- **Variable speaking styles** (fast, slow, soft, loud) affect prosodic features.
- **Microphone distance and quality** cause inconsistent audio levels.
- **Emotion and tone cues** are subtle and hard to capture reliably.
- **Alignment with video** (hand/face actions) requires precise synchronization.

expression recognition.

Model bias can misinterpret emotions across different individuals.

Cluster ID	Average Learning Gain (LG)	Assigned Name	Hypothesized Behavioral Profile
Cluster 2	0.75	**Productive Learners**	High engagement, high tone variability, balanced talk time (2 pairs)
Cluster 1	0.72	**Effective Collaborators**	Low initial activity, high pause time for reflection, high prosodic entrainment (9 pairs) .
Cluster 0	0.42	**Passive Learners**	Low total talk time, low facial expression, high speech overlap/interruption. (6 pairs)

Tab 6

Points for Tomorrow:

Speech: Model: silero VAD , and librosa library (it gives accurate features for audio)

Stage 1 — Voice Activity Detection (VAD)
Stage 2 — Pause statistics per speaker
Stage 3 — Pairwise interaction metrics
Stage 4 — Pitch & Energy per speaker (used librosa for this)

Emotions: Mediapipe face mesh 468 landmarks 3D (x,y,z) values

Stage 1 - Face landmark detection
Stage 2- Compute Raw Facial Features
Stage 3 - EMA smoothing
Stage 4 - Valence & Arousal Estimation

Valence- how +ve or -ve the emotion is
High valence = partner is enjoying the task, cooperative, relaxed
Low valence = confusion, disagreement, stress, frustration

Arousal- how activated or energised a person is.

In collaboration:
High arousal = active involvement, dynamic interaction
Low arousal = disengagement, slow responses, low energy

Stage 3 — Smooth Features (EMA smoothing)

- To reduce noise from fast movements, we apply **exponential moving average** (α=0.2).
- Smoothing makes emotional trends stable across frames.