Preparation

1. Keep all videos in ".mp4" format in a folder named "videos"
2. Create target files with positive score, active score, and category score as follows (figure in the next page):

Time	Positive score (+/-)	Active score (+/-)	one pred
00>02	4	4	3
02>04	3	2	3

3. Keep target files in the "videos" folder and name them as follows:
    a. If video file name is "Team2_teamsession2.mp4", then name the target file as "Team2_teamsession2_target.csv"
4. Create audio files using *"AudioConversion.py"* python script and store them as follows:
    a. Files will be in ".wav" format
    b. Keep ".wav" audio files in "videos" folder
    c. If video file name is "Team2_teamsession2.mp4", then name the target file as "Team2_teamsession2.wav"
5. (Transcription)
    a. (Whisper) Run "TranscribeVideos.py" to generate transcript files for videos as follows:
        i. If video file name is "Team2_teamsession2.mp4", then the transcript file will be "Team2_teamsession2.txt"
    b. (DEPRECATED) Create transcript file (using Welder) and store them as follows:
        i. If file is in ".srt" format, convert to ".txt" format
        ii. Keep ".txt" transcript files in "videos" folder
        iii. If video file name is "Team2_teamsession2.mp4", then name the transcript file as "Team2_teamsession2.txt"
6. (Optional; run if "one pred" values in target files are not correct) Run *"target_correction_one_pred.py"* python script

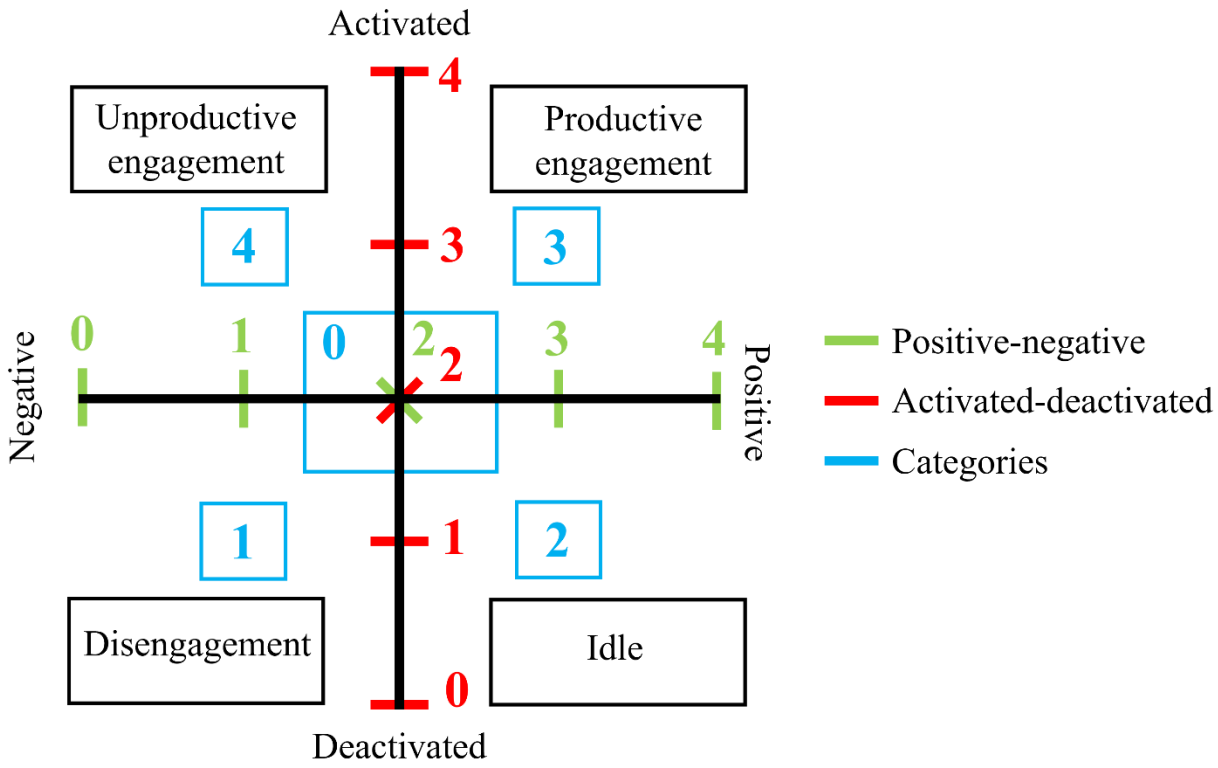


Figure: Positive-negative, activated-deactivated, and category codes. Positive-negative scoring codes: "0", "1", "2", "3", "4" respectively denote n = 0, n = 1, n = 2, n = 3, and n = 4 students are positive. Activated-deactivated scoring codes: "0", "1", "2", "3", "4" respectively denote n = 0, n = 1, n = 2, n = 3, and n = 4 students are activated. Category codes: based on the positive-negative and the activated-deactivated scoring codes. Category "0" was ignored for ML-based processing.

Text

1. Create an empty folder “Text_Dataset”
2. Run *“Create_Text_Dataset.py”* python script
3. Run *“Text_Remove_NA.py”* python script
4. (Optional) Run *“TextInspection.py”* python script
    a. This will help you check the dataset and how many samples in each category
5. Text data augmentation
    a. Keep this folder in the root directory
    b. Run *“TextAugmentation.py”* python script
    c. If you want to delete the augmented audio clips, run *“DeleteTextAug.py”* python script
6. Run *“Fasttext_dataset_creation_one_pred.py”* python script
    a. This will generate two files “Text_Dataset_LATEST_train.txt” and “Text_Dataset_LATEST_test.txt” in the folder “Text_Dataset/one pred”
7. Run *“fasttext_train_one_pred.py”* python script
    a. This will generate the trained text model file “fasttext_one_pred_model.bin” in the root directory
8. Run *“TextConfMatrix.py”* python script
    a. You will get the results of the training performance

Audio

1. Create empty folder named “Audio Processing”
2. Create empty folder named “audioData” inside the folder “Audio processing”
3. Run *“AudioDatasetCreation.py”* python script (time-consuming)
    a. This will populate the directory “Audio processing/audioData” with raw audio clips from the full audio files in “.wav” format
4. (Optional) Run *“AudioInspection.py”* python script
    a. This will help you check the dataset and how many samples in each category
5. Audio data augmentation (time-consuming)
    a. Keep this folder inside the “Audio processing” folder
    b. Run *“AudioAugmentation.py”* python script
    c. If you want to delete the augmented audio clips, run *“DeleteAudioAug.py”* python script
6. Run *“AudioDatasetPreparation.py”* python script
    a. This will create a file “AudioDataset_excel.csv” inside “audioData” folder
7. Run *“AudioPreprocessing.py”* python script
    a. This will create a file “NN_data.pickle” inside “audioData” folder
8. Run *“AudioTrainAll_NEW.py”* python script (time-consuming)
    a. This will generate pickle files containing the training and testing data split inside the “audioData” folder
    b. Also, it will generate model weight files “*{epoch}*_weights_CNN-*{acc}*” every 10 epochs inside “audioData” folder
    c. Run the command *“tensorboard --logdir logs_CNN”* to monitor the training progress
9. Run *“AudioModel_Evaluate.py”* python script
    a. You will get the results of the training performance in the file “AudioModel_EvaluationResults.xlsx” inside “audioData” folder
10. Select the best performing model file and name it “Audio_Model_CNN.hdf5”

Video

1. Create empty folder “faceData” in root directory
2. Create empty folder “faceDataLabels” in root directory
3. Create an empty csv file named “vidsDone.csv” inside “faceDataLabels” folder in the following format (keep the header names in the csv file):

FileName

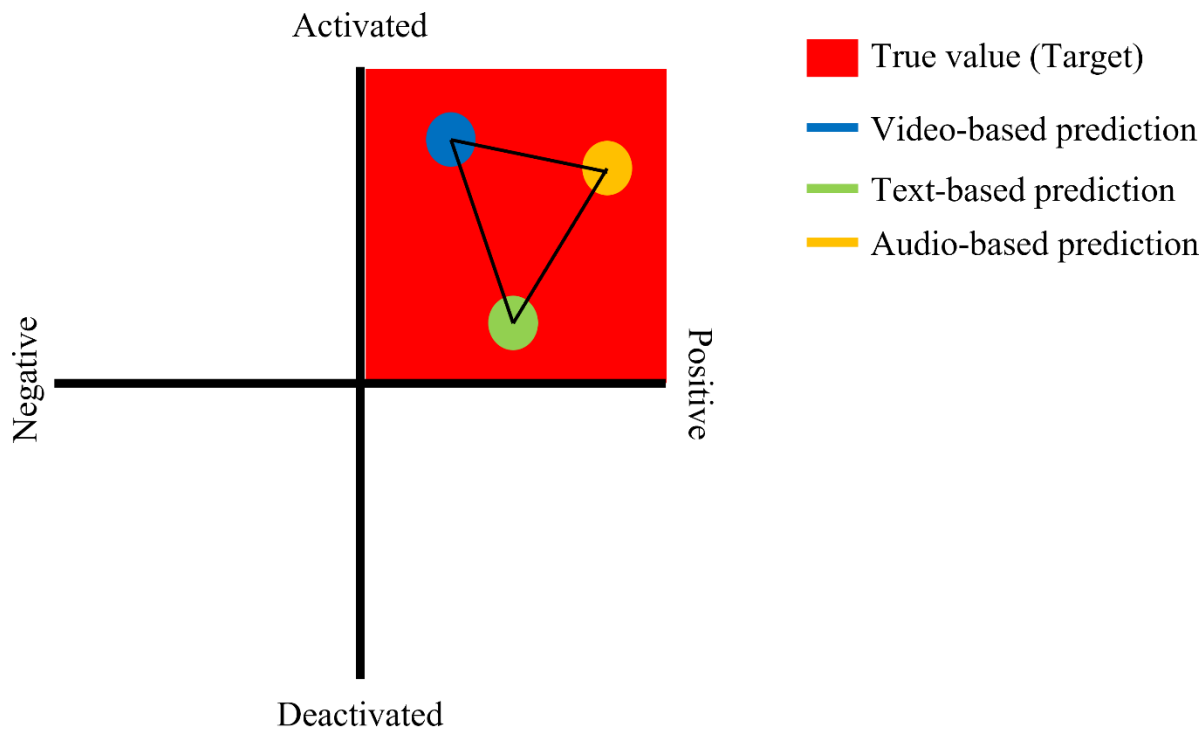
4. Create an empty csv file named “faceLabels.csv” inside “faceDataLabels” folder in the following format (keep the header names in the csv file):

FileName	Label

5. Run *“SaveFaces_All.py”* python script (very time-consuming)
6. (Optional) Run *“VideoInspection.py”* python script
    a. This will help you check the dataset and how many samples in each category
7. Data augmentation (very time-consuming):
    a. Run *“FacesAug.py”* python script (only augments cat 1 and cat 2 faces)
    b. If you want to delete the augmented images, run *“DeleteFacesAug.py”* python script
    c. If there are any errors in labelling augmented images, run *“FacesAugCorrection.py”* python script (very time-consuming)
8. Run *“VideoDatasetPrep.py”* python script
    a. This will create three files “train.txt” (with the training samples), “test.txt” (with testing samples), and “faceLabels.txt” (with all samples) inside “faceDataLabels” folder.
9. Keep *“core.py”*, *“VideoPreprocess.py”*, and *“utils.py”* python scripts in the root directory
10. Create an empty folder “saved_video_models” in the root directory.
11. Run *“VideoTraining.py”* python script (very time-consuming)
    a. Run the command *“tensorboard --logdir runs”* to monitor the training progress
12. Select the best performing model from the folder “saved_video_models” and rename it to “test_model_alpha_0”. Copy it to the root directory.
13. Manually input the confusion matrix values from tensorboard to *“VideoConfMatrix.py”* python script and run it (couldn’t automate it)
    a. You will get the results of the training performance

Processing (Validation)

1. Copy the file "Audio_Model_CNN.hdf5" from the directory "Audio processing/audioData" to the root directory
2. Make sure to keep the files "test_model_alpha_0", "fasttext_one_pred_model.bin", and "Audio_Model_CNN.hdf5" in the root directory
3. Make sure to keep the python scripts *"core.py"*, *"emotion_detection_service.py"*, *"test.py"*, and *"utils.py"* in the root directory
4. Main run options (use either one):
    a. Run *"Processing_Validation_noGUI_FINAL.py"* python script (best option)
        i. No visualization
        ii. This will process all main video files in the "videos" folder automatically
        iii. After processing, two files will be generated in the "videos" folder for each video file:
            1. If video file name is "Team2_teamsession2.mp4", a processed video file named "Team2_teamsession2_processed.mp4" will be generated
            2. If video file name is "Team2_teamsession2.mp4", a processed data file named "Team2_teamsession2_processed_all_data.csv" will be generated
    b. Run *"Processing_Validation_GUI_FINAL.py"* python script (time-consuming)
        i. Allows you to visualize the processing
        ii. You have to manually select a video from the "videos" folder to process it
        iii. After processing, two files will be generated in the "videos" folder for each video file:
            1. If video file name is "Team2_teamsession2.mp4", a processed video file named "Team2_teamsession2_processed.mp4" will be generated
            2. If video file name is "Team2_teamsession2.mp4", a processed data file named "Team2_teamsession2_processed_all_data.csv" will be generated
5. (Optional) Run *"JoinVideoSound.py"* python script:
    a. This will generate a processed file with audio in the "videos" folder for each processed video:
        i. If video file name is "Team2_teamsession2.mp4", a processed video file named "Team2_teamsession2_processed_wSound.mp4" will be generated.


Activated
Negative
Positive
Deactivated
True value (Target)
Video-based prediction
Text-based prediction
Audio-based prediction

Post-processing 1: Results data analysis

1. Run *"calcResults_Validation.py"* python script (time-consuming)
    a. This will generate the file "Results_validation.xlsx" inside the "videos" folder
    b. You have to manually work with the data, create graphs, tables, etc. from the ".xlsx" file

Post-processing 2: Replaying processed video

1. Run *"InspectProcessedVideos_Validation.py"* python script
    a. Change the *filename* variable to the name of the video file you want to replay.
    b. (Work in progress) Need to add a GUI to select video to replay

RQ3: Applying the processing scripts to other videos that were not hand-coded:

1. Keep the new videos in folder "videos_RQ3" in the root directory
2. Transcription:
    a. (Whisper) Run "TranscribeVideos_RQ3.py" to generate transcript files for videos as follows:
        i. If video file name is "Team2_teamsession2.mp4", then the transcript file will be "Team2_teamsession2.txt"
3. Create audio files using *"AudioConversion_RQ3.py"* python script and store them as follows:
    a. Files will be in ".wav" format
    b. Keep ".wav" audio files in "videos_RQ3" folder
    c. If video file name is "Team2_teamsession2.mp4", then name the target file as "Team2_teamsession2.wav"
4. Main run options (use either one):
    a. Run *"Processing_RQ3_noGUI_FINAL_noVidOutput.py"* python script (best option)
        i. No visualization
        ii. This will process all main video files in the "videos_RQ3" folder automatically
        iii. After processing, one file will be generated in the "videos_RQ3" folder for each video file (no processed video file is generated):
            1. If video file name is "Team2_teamsession2.mp4", a processed data file named "Team2_teamsession2_processed_all_data.csv" will be generated
    b. Run *"Processing_RQ3_GUI_FINAL.py"* python script (time-consuming)
        i. Allows you to visualize the processing
        ii. You have to manually select a video from the "videos_RQ3" folder to process it
        iii. After processing, one file will be generated in the "videos_RQ3" folder for each video file (no processed video file is generated):
            1. If video file name is "Team2_teamsession2.mp4", a processed data file named "Team2_teamsession2_processed_all_data.csv" will be generated
5. Run *"calcResults_RQ3.py"* python script (time-consuming)
    a. This will generate the file "Results_RQ3.xlsx" inside the "videos_RQ3" folder
    b. You have to manually work with the data, create graphs, tables, etc. from the ".xlsx" file