Emotional Mario: A Games Analytics Challenge

Mediaeval 2021

Link

Introduction

- A grand challange from MediaEval
- Motivation: game affects users' emotions → the connection between emotion and games
- **Task 1: Event detection**

- **Task 2: Gameplay summarization**

Introduction

- A grand challange from MediaEval
- Motivation: game affects users' emotions → the connection between emotion and games
- **Task 1: Event detection**

- **Task 2: Gameplay summarization**

Dataset: Toadstool

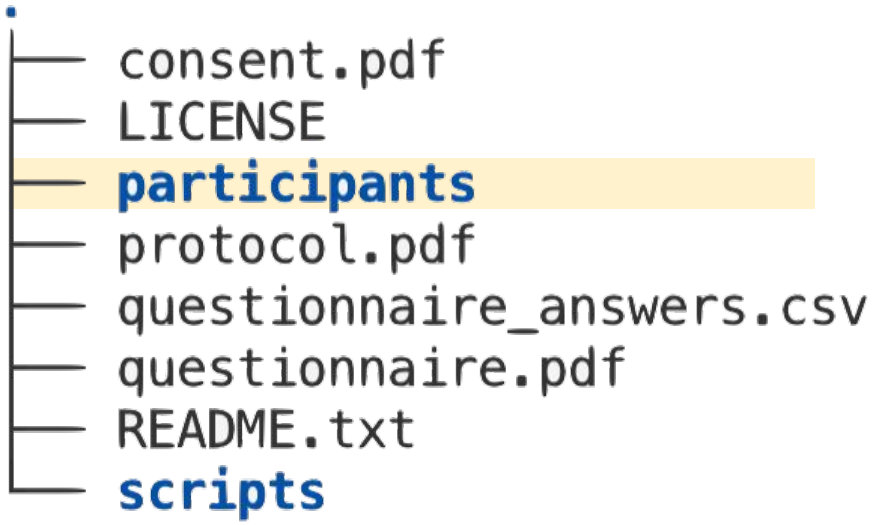
- The dataset consists of video, biometric data collected from 10 participants playing Super Mario Bros
- Participants wore wristband to collect the biometric data
- The dataset is separated into two main folders:

```
.
├── consent.pdf
├── LICENSE
├── participants
├── protocol.pdf
├── questionnaire_answers.csv
├── questionnaire.pdf
├── README.txt
└── scripts
```

- **participants**: facial videos, biometric data, and game data from the ten participants
- **scripts**:
  - Scripts about how they synchronize the game data with other data
  - A script to get mario game frame

Dataset: Toadstool

- The dataset consists of video, biometric data collected from 10 participants playing Super Mario Bros
- Participants wore wristband to collect the biometric data
- The dataset is separated into two main folders:



- **participants**: facial videos, biometric data, and game data from the ten participants
- **scripts**:
  - Scripts about how they synchronize the game data with other data
  - A script to get mario game frame

Dataset: Toadstool-Participants

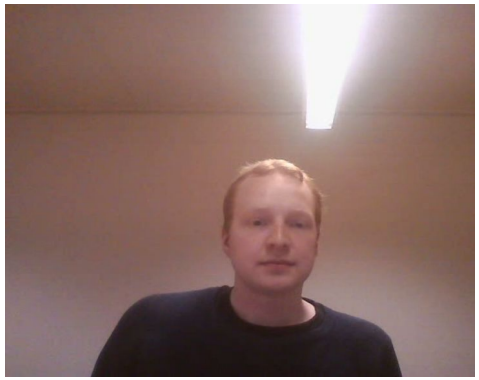
- In the participants folder, we have data for the 10 participants
- E.g., data in participant_0

```
.
├── participant_0_gap_info.json
├── participant_0_sensor
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   └── TEMP.csv
├── participant_0_sensor_raw
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   ├── tags.csv
│   └── TEMP.csv
├── participant_0_session.json
├── participant_0_video.avi
└── participant_0_video_info.json
```

Dataset: Toadstool-Facial video in Participants

- In the participants folder, we have data for the 10 participants
- E.g., data in participant_0

```
.
├── participant_0_gap_info.json
├── participant_0_sensor
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   └── TEMP.csv
├── participant_0_sensor_raw
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   ├── tags.csv
│   └── TEMP.csv
├── participant_0_session.json
├── participant_0_video.avi
└── participant_0_video_info.json
```



A facial video

{"start_time": 1579785027.233421, "stop_time": 1579787135.8056052}

Dataset: Toadstool-Biometric Data in Participants

- In the participants folder, we have data for the 10 participants
- E.g., data in participant_0

```
.
├── participant_0_gap_info.json
├── participant_0_sensor
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   └── TEMP.csv
├── participant_0_sensor_raw
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   ├── tags.csv
│   └── TEMP.csv
├── participant_0_session.json
├── participant_0_video.avi
└── participant_0_video_info.json
```

Raw biometric data collected from the wristband:

- **ACC**: the movement of the wearer
- **EDA**: the electrical conductivity of the skin
- **BVP**: blood Volume Pulse
- **IBI**: the time interval between individual heartbeats → the instantaneous heart rate/heart rate variability
- **HR**: the average heart rate values
- **TEMP**: the temperature of the person

Dataset: Toadstool-Biometric Data in Participants

- Data for the 10 participants
- E.g., data in participant_0

```
.
├── participant_0_gap_info.json
├── participant_0_sensor
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   └── TEMP.csv
├── participant_0_sensor_raw
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   ├── tags.csv
│   └── TEMP.csv
├── participant_0_session.json
├── participant_0_video.avi
└── participant_0_video_info.json
```

Synchronized biometric data collected from the wristband to the game frame:

- **ACC**: the movement of the wearer
- **EDA**: the electrical conductivity of the skin
- **BVP**: blood Volume Pulse
- **IBI**: the time interval between individual heartbeats → the instantaneous heart rate/heart rate variability
- **HR**: the average heart rate values
- **TEMP**: the temperature of the person

Dataset: Toadstool-Game Data in Participants

- In the participants folder, we have data for the 10 participants
- E.g., data in participant_0

```
.
├── participant_0_gap_info.json
├── participant_0_sensor
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   └── TEMP.csv
├── participant_0_sensor_raw
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   ├── tags.csv
│   └── TEMP.csv
├── participant_0_session.json
├── participant_0_video.avi
└── participant_0_video_info.json
```

Where is the game data?

Dataset: Toadstool-Game Data in Participants

- Use replay_game_session.py in *scripts* folder with participant_[num]_session.json in *participants* folder to get the game frame

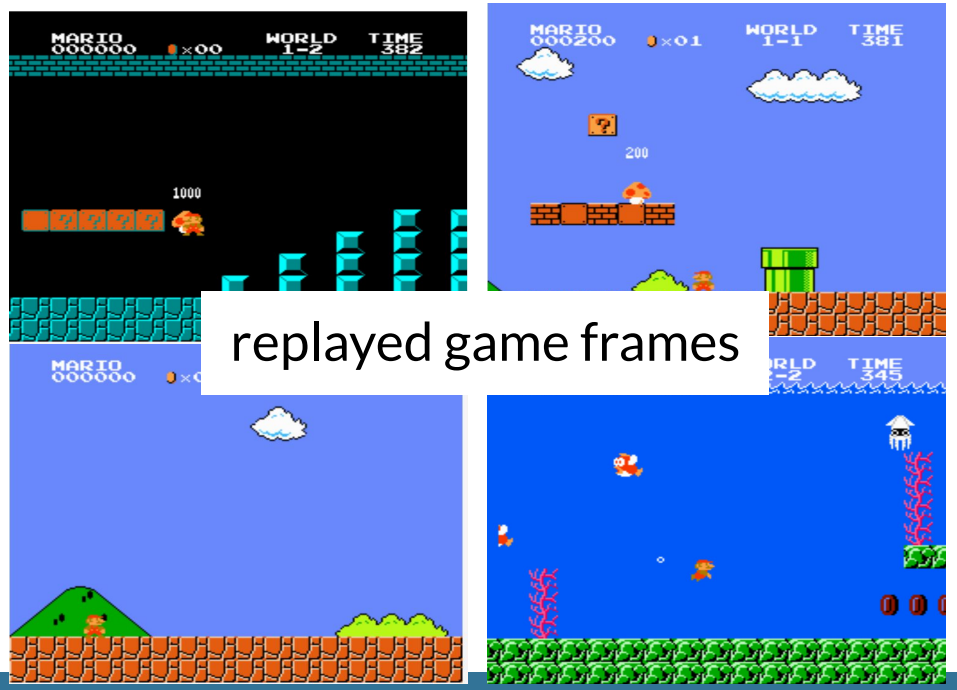
```
.
├── participant_0_gap_info.json
├── participant_0_sensor
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   └── TEMP.csv
├── participant_0_sensor_raw
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   ├── tags.csv
│   └── TEMP.csv
├── participant_0_session.json
├── participant_0_video.avi
└── participant_0_video_info.json
```

Data in *scripts* folder:

```
.
├── replay_game_session.py
├── synchronize_game_with_face_data.py
├── synchronize_game_with_sensor_data.py
└── synchronize.sh
```

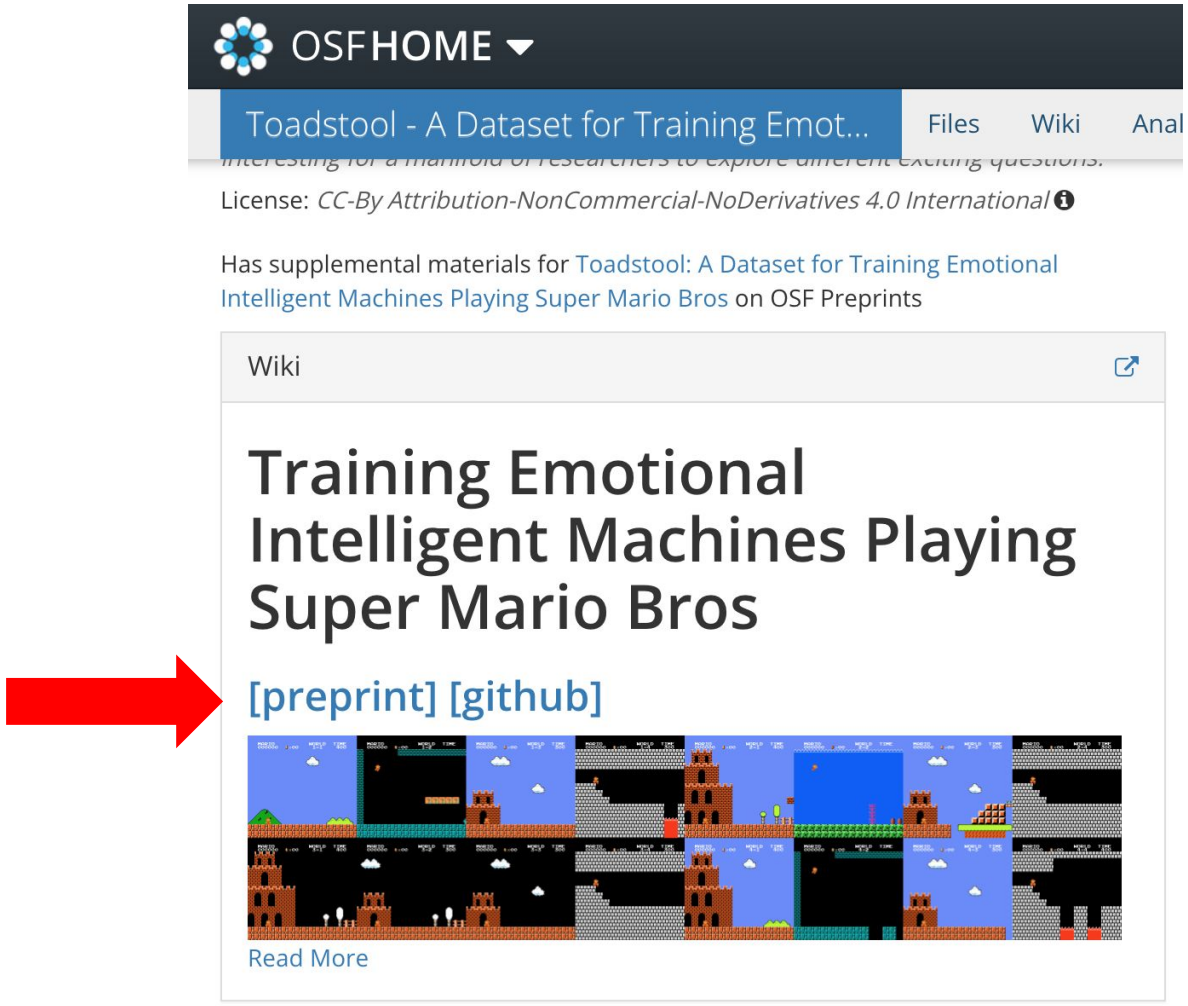
Command:

python3 replay_game_session.py -i [path]/participant_[idx]_session.json -o [folder path to store the generated frame images]



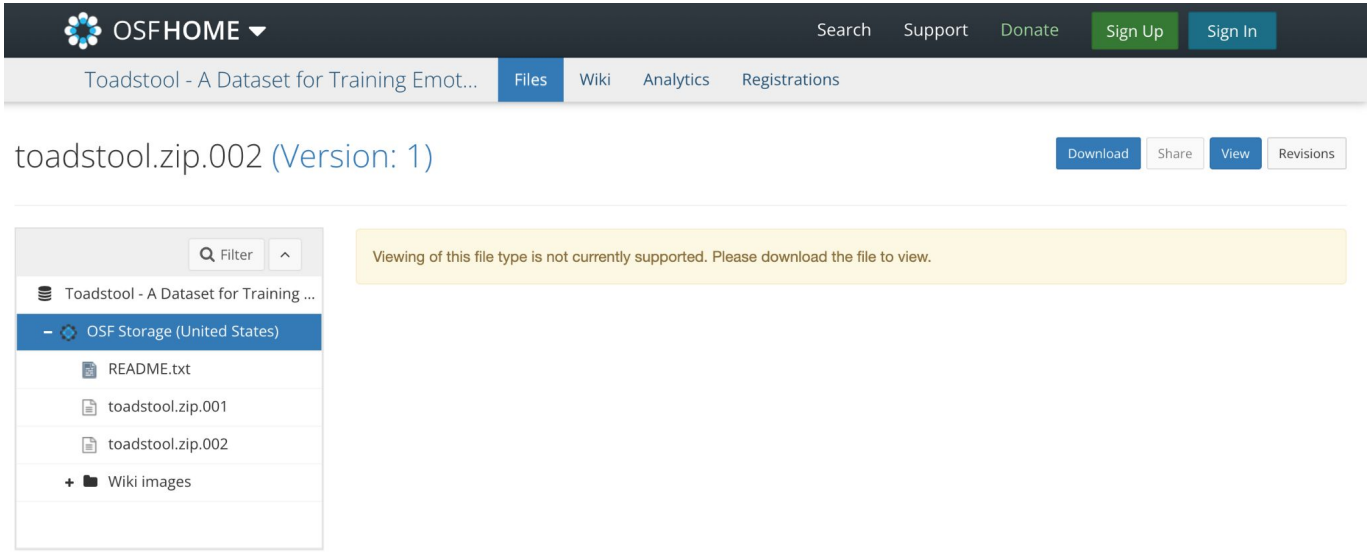
Dataset: Toadstool

Check out their perprint and github for more information



How to Download the Dataset?

- Click the dataset link
- Download toadstool.zip.001 and toadstool.zip.002
- Use 7 zip to unzip the dataset
- Ubuntu command:
  - sudo apt install p7zip-full
  - 7z x toadstool.zip.001



Introduction

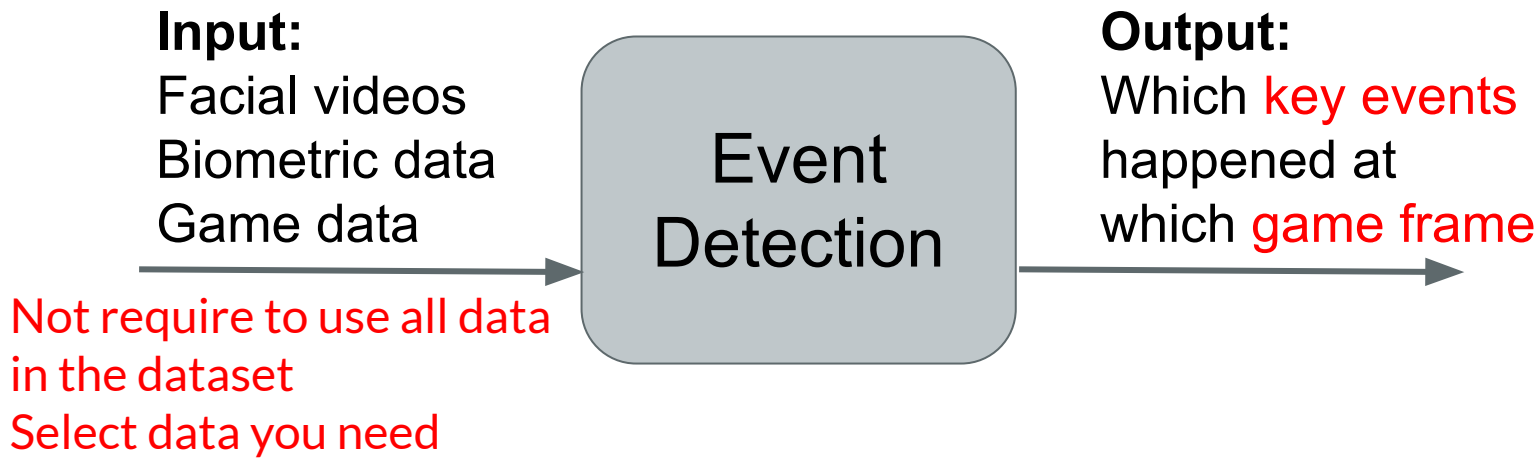
- A grand challange from MediaEval
- Motivation: game affects users' emotions → the connection between emotion and games
- **Task 1: Event detection**

Not require to use all data in the dataset
Select data you need

- **Task 2: Gameplay summarization**

Introduction

- A grand challange from MediaEval
- Motivation: game affects users' emotions → the connection between emotion and games
- **Task 1: Event detection**



- **Task 2: Gameplay summarization**

Output

Frame number: 1

Select numbers within "obs_n" in participant_{idx}_session.json

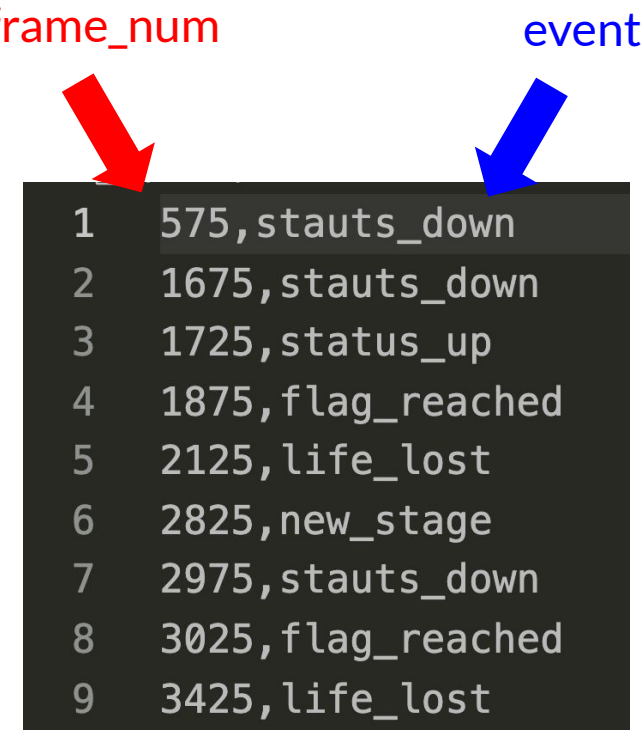
```
.
├── participant_0_gap_info.json
├── participant_0_sensor
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   └── TEMP.csv
├── participant_0_sensor_raw
│   ├── ACC.csv
│   ├── BVP.csv
│   ├── EDA.csv
│   ├── HR.csv
│   ├── IBI.csv
│   ├── info.txt
│   ├── tags.csv
│   └── TEMP.csv
├── participant_0_session.json
├── participant_0_video.avi
└── participant_0_video_info.json
```

Key events: 2

- **new_stage:** at the very beginning of a new stage, except the first one, which starts at frame_number 1
- **flag_reached**: when the flag, i.e. the level end is reached
- **status_up**: when a mushroom or flower (power up) is consumed by the player
- **status_down**: when a player encounters a monster and looses a power up
- **life_lost**: when a player looses one of Mario's lifes, note that the game is in endless mode

Sample Output

- Output your results into a csv files in format: ***frame_num*,*event***
- There must be a “,” between frame_num and event



- Name the output csv file as “**participant_[num]_events.csv**”, replace [num] with the index of participants
- **Do not** put headers in the csv file

Groundtruth

- Download the groudtruth here
- We only provide the groundtruth of 7 participants
- The rest of the groundtruth (particiapnt 2, 4, 7) are hidden testcases

```
[
    {"event": "status_up", "frame_number": 812},
    {"event": "flag_reached", "frame_number": 4229},
    {"event": "new_stage", "frame_number": 4230},
    {"event": "status_up", "frame_number": 4719},
    {"event": "status_down", "frame_number": 5849},
    ...
]
```

Evaluation Script

- We will provide an evaluation script
- Usage: **python3 evaluate.py**
  -i [input file folder]
  -t [groundtruth folder]
  -n [index of participant]
  - You can put multiple number behind -n to evaluate multiple results at one time
  - Put the python files and the two file folders under the same folder
- You need to name your input files in the format of "**participant_[num]_events.csv**"

```
run
├── participant_2_events.csv
├── participant_4_events.csv
├── participant_7_events.csv
└── README.md
truth
├── participant_0_events.json
├── participant_1_events.json
├── participant_2_events.json
├── participant_3_events.json
├── participant_4_events.json
├── participant_5_events.json
├── participant_6_events.json
├── participant_7_events.json
├── participant_8_events.json
├── participant_9_events.json
└── README.md
```

```
› python3 evaluate.py -i run -t truth -n 2 4 7
[2, 4, 7]
run identifier,precision,recall,f1
participant_2_events.json,0.0076804915514592 93,0.025380710659898477,0.011792452830188 68
participant_4_events.json,0.007877813504823151,0.30434782608695654,0.015358094342579535
participant_7_events.json,0.006752411575562701,0.2937062937062937,0.0132013201320132
```

Evaluation Details and Grading

1. Search the frame number shown in groundtruth in your input file (Frame number **+/- 25** is accepted)
2. Compare the event title in ground truth and your results
3. Calculate f1 score

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Grading:

- After your presentation, you will need to generate the results of participants 2, 4, and 7 and send the csv files to TAs right away
- We will use the avg. ***f1 score*** **of participants 2, 4, and 7** to grade your project. The number will be rounded to four decimal places.
  E.g. 0.3456 → 0.346
- **Benchmark** Scores
  - top ⅓: 10 points
  - next ⅓: 7 points
  - last ⅓: 4 points
  - Note: if your f1 score is lower than basline (0.02), you will get 0 points

Baseline

- We will provide the baseline code: baseline1.py
- It can be separated into 2 steps:
  - Find event may happen at which frame
    - Read BVP.csv for the participants as an array
    - Find the local maxima of the BVP array
    - Select the frames with local maxima as the target frames
  - Guess the event of the target frames randomly
    - We have 5 events. We set each event a number (0 ~ 4)
    - Randomly select a number from 0~9 for each target frame
    - If the number is between 0 to 4, we set an event to the frame based on the number
- Average F1 score for the testing set: ~ 0.02
- Run the baseline1.py:

  python3 baseline1.py -p [path to participants directory in the dataset] -i [index of the participants] -o [location to save the output csv file]

Other Ideas

- Create meaningful input data from the dataset
  - Use FER (facial expression recognition) to recognize the emotion from facial videos
  - Output sample results: csv files with emotion analysis for each participant

```
,box,angry,disgust,fear,happy,sad,surprise,neutral
0,"(273, 269, 139, 139)",0.01,0.0,0.04,0.07,0.06,0.0,0.81
1,"(275, 270, 138, 138)",0.01,0.0,0.03,0.08,0.03,0.0,0.86
2,"(274, 272, 137, 137)",0.01,0.0,0.04,0.05,0.05,0.0,0.85
3,"(276, 273, 134, 134)",0.01,0.0,0.04,0.08,0.03,0.0,0.85
4,"(278, 275, 130, 130)",0.0,0.0,0.04,0.07,0.02,0.0,0.86
```

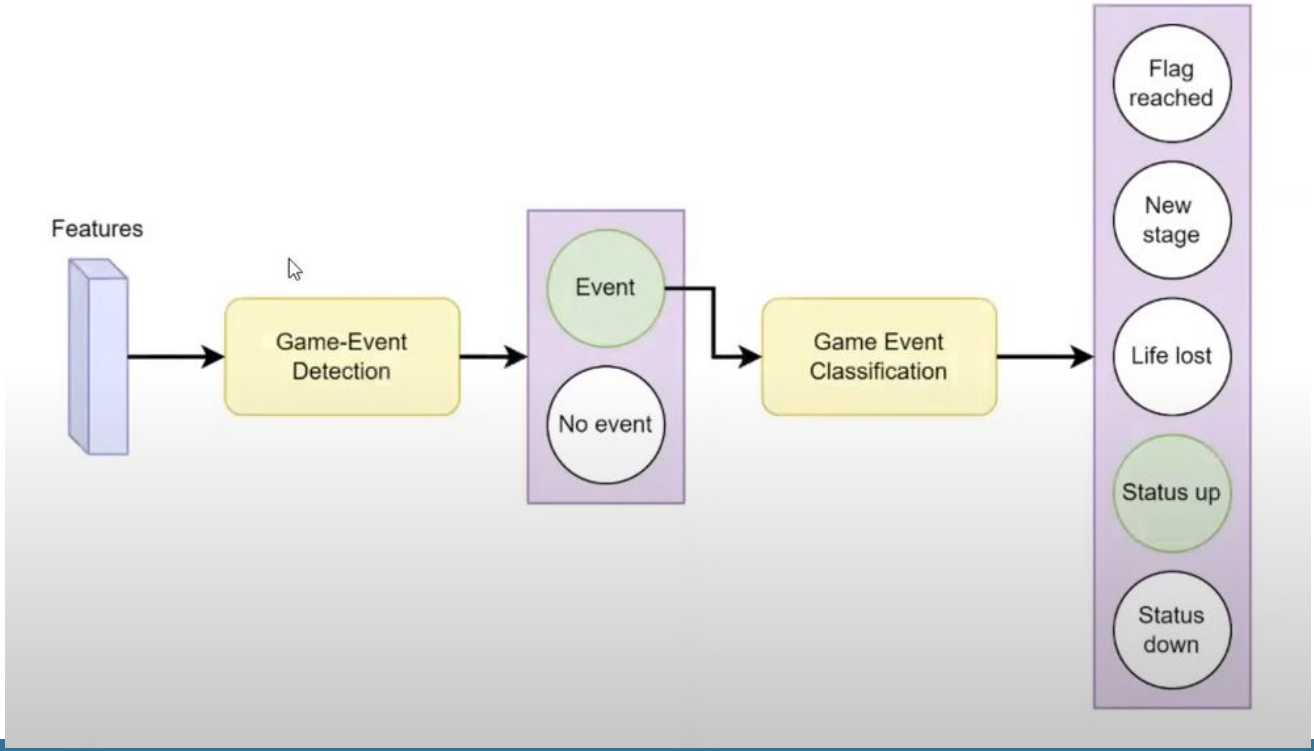
- Classifiers
  - Use random forest to classify each frame
  - Use RNN to consider contextual data
  - ...

Conclusion

- Get the input data from the dataset: toadstool
- The groundtruth of the training set: groundtruth
- We will provide the evaluation script, the baseline scripts, and a sample output: participant_0_events.csv on elearn or the website
- Demo to generate the results of the testing set (participants 2, 4, and 7) and send the csv files to TAs before the report deadline.
  We will anounce the detailed demo time later.
- The format of the csv files must follow what we just show in the previous slides
- Source code should be uploaded to elearn as well

Possible Solution

- Game-Event Detection:
  - Input: FER, BVP, Electrodermal Activity
- Game Event Classification:
  - Input: game played images, BVP
  - Use ResNet50 to classify the event



Any Questions?

Chia-Ying Hsieh

cyinghsieh@gmail.com

Tzu-Yi Fan

joyfan2@gmail.com