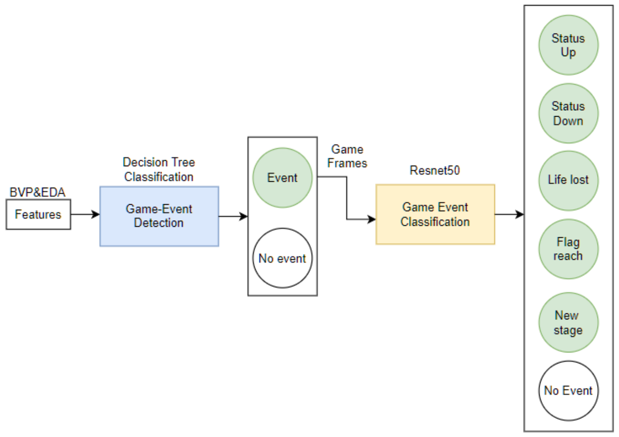
Introduction to Multimedia
Group12 Term Project Report

108020022 周昱宏
108062139 李哲
108062315 莊子郁

***Abstract* - By given biometric data and game session information, we are required to classify which frame has an event, and what kind of event it is. We tried to use Tesseract OCR to classify all the events.**

APPROACH 1: USING BVP AND EDA

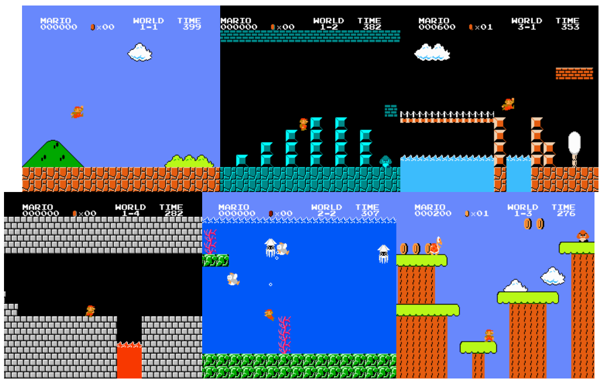
Considering the possible solution provided in the Mario slide, the diagram of our first structure about this project is:



We implement our Game-Event Detection by Decision Tree Classification. Our training dataset is the combination of 7 participants(participant 0, participant 1, participant 3, participant 5, participant 6, participant 8, participant 9) BVP data, EDA data and labeled groundtruth.

We concatenate BVP data and EDA data to a ndarray as the features of the Decision Tree Classifier, and we make a labeled ground truth where the frame becomes "1" when there is an event.

When we are testing participants in the training dataset, it shows high precision results to judge if there is an event or not. However, when we try to test participant 2, participant 4,and participant 7, who are not in the training dataset. After checking the result frames, we sadly find out that detection has very poor precision that hardly detects if the frame has an event or not.



Then we research some information online to see if we can come up with a new approach toward this. We find the original research, "Emotional Mario - Using Super Mario Bros. to Train Emotional Intelligent Machines." In this research, we found there is a description about BVP, " *the top 2%, meaning those with the highest associated prediction values, of the test images, and likely much of the top range, where mostly dominated by frames from levels using a dark background. These are either castle levels, or certain other late levels that use different background graphics from the rest of the levels.*" From this, we could find that BVP is less related with the events occurring in the game.

Besides, we found the link video in possible solution, they have a conclusion:

1. Using physiological signals and facial emotion recognition are not enough for event detection.
2. Some labels of the data are incorrect(e.g. flag reached).
3. User-independent model is not appropriate because people have different reactions.
4. Game-play image is a significant feature in the model.

Therefore, we completely give up with this biometric data approach. After observing the game frames, we find another method, which is simply using game frames.

Approach 2: Using Text Recognition

From the gameplay image, we found that game information text is the best identifier for us to detect the event frames. Thus, we use the Tesseract OCR package, which is an optical character recognition engine for various operating systems.

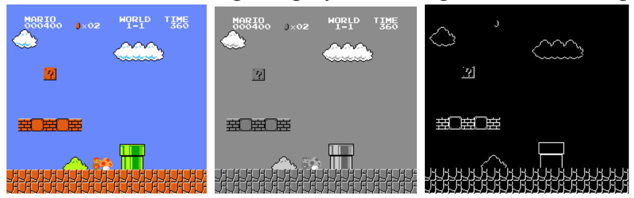
Since we have over 120,000 frames for each participant, we sample every 30 frames. We can largely reduce the dataset to 4000 while keeping the correctness. The reason we choose 30 frames are:

1. The game fps is 30.
2. The output frame number can be ground truth plus 25 or minus 25 frames, so we will not miss any correct frames.

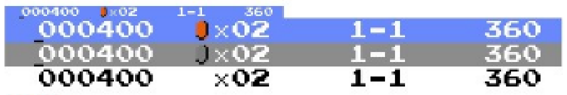
During the recognition, we met a problem, which is the image would have too many infomation that greatly affect the accuracy of recognition.

In order to solve this problem, we use a series of operations.

1. Transforming the image to grayscale
2. Transforming the grayscale image to thresholding.



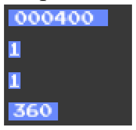
So, we first crop our original image to this. However, as we can see, it still has very poor results.



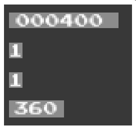
```
000400

xOoe

1-1

60
```

Then, we decide to separate the image into four parts and only contain integers: score, world_1, world_2, time.

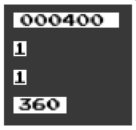
1. Crop the image:



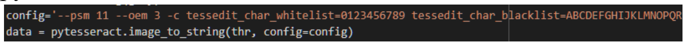
2. Transforming the image to grayscale:



3. Transforming the grayscale image to thresholding:



After preprocessing the images, we use the function in pytesseract:



```
config='--psm 11 --oem 3 -c tessedit_char_whitelist=0123456789 tessedit_char_blacklist=ABCDEFGHIJKLMNOPQR
data = pytesseract.image_to_string(thr, config=config)
```

In the config part, we use psm 11, which is for sparse text to find as much text as possible without any particular order. Why do we use this config? Because we tried many configurations, it is the best performance.

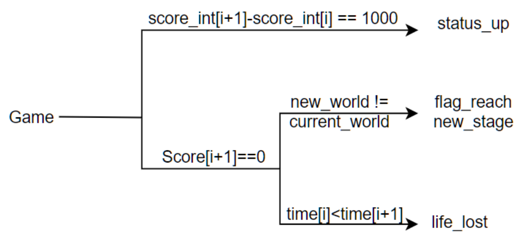
Now, we get four string arrays. we need to check which data is applicable. Because some images will generate wrong data, like '1' be recognized as 'L'.

The correction of data has following steps:

1. Check if the array has any empty string, and replace it with the former value. Or it would cause an error.
2. Replace all the characters with "", which means there are only numbers in strings.
3. Further check if the string is digit, then append it into the integer array.

After all these procedures, we now have pure integer arrays with score, world and time, and now we are allowed to classify events.

This diagram below shows our statements.



1. Status up: When score increases by 1000. Although the Super Mario Wiki shows there are still other situations that would cause the same result, we think it is still the most effective way to deter mine a "statu up" situation. It means that num of other situations where an increased 1000 score is too few, we can ignore them.
2. New stage: the next score equals 0 && a world change happens.
3. Flag reach: "new stage" 's frame number minus one.
4. New stage: the next score equals to 0 && next time is biggert than the current time

Partial Results

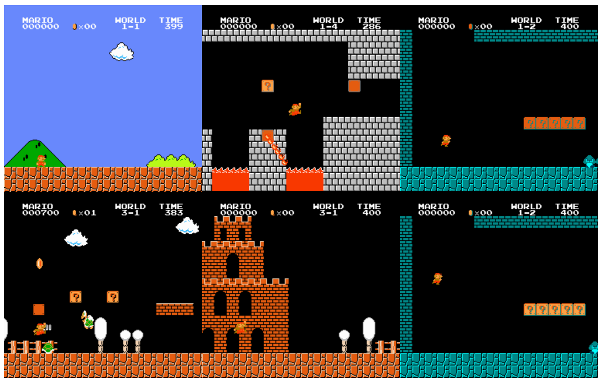
- participant 0

Precision	0.6710526315789473
Recall	0.6035502958579881
f1 score	0.6355140186915887

- participant 1

Precision	0.7131147540983607
Recall	0.7073170731707317
f1 score	0.7102040816326531

- participant 2



Discussion and Conclusion

1. We didn't distinguish status down events among game frames so far, in that we cannot conclude game frame information changes when status down events happen. Instead, we observe that there are two features and identifiers that status down has: Mario would repeatedly disappear and reappear in the continuous frames, or Mario's size would become smaller when status down events happen. However, the former solution will fail because our input data are sampled frames, and we don't have proper labels setting for the second solution.Hence, we didn't do this practice.
2. It was a mistake for us to set the inappropriate label of our first approach. Originally, we labeled frames to be 1 if there is an event in the ground truth file. But we should mark frames to be 1 if there is an event in the ground truth file plus minus 25.Hence, we will implement this adjustment in our future work.
3. Just like the BVP or EDA data in our decision-tree training approach, we consider that the result of such a method would be less precise in event detection. On top of that, we get the conclusion of some possible solutions, "*Using physiological signals and facial emotion recognition are not enough for event detection.*" Hence, we change our direction to game frame image detection.
4. We use Google Colaboratory to implement our program together, and we have encountered a very strange problem, which is that we can only use 97077 images. We thought that is because Google Drive has a limitation for each folder. So we have to produce images into two different folders, and finally get the whole result.