Emotional Mario: A Game Analytics Challenge

Team 24 Multimedia Final Project (108000165 許育恩, 107062273 廖宏淇, 107021206 羅詔文)
Yuk Ian Khor

***Abstract*—In this report, we compare and present a model that aims to classify game events in the game *Super Mario Bros.* We used several methods in the process including both machine learning and image analysis methods, and will explain the most successful method in detail, which is image analysis on number displays.**

I. Introduction

This project is based on the *Toadstool* dataset aims for training emotional intelligent machines playing *Super Mario Bros*. However, the aim of the project is to detect the significant game events, thus not all datas is required to be used. In normal cases, machine learning models are very efficient at classifying different labels, however the biometric data presented in the *Toadstool* dataset is too indistinguishable from each other, and thus we prepared another model using game frame data to detect events without using biometric data. Since we know what game the participants are playing, the game rules can be easily found online and be used to our advantage in event detections.

II. Data Analysis & Preprocessing

A. Data types

The *Toadstool* dataset includes 10 sets of data from 10 participants playing *Super Mario Bros.* Each set of data include (1) the participant's biometric data, (2) game frame data which is each frame of the game during the playthrough, (3) recording of the face of the participant throughout the playthrough, and (4) a questionnaire that was filled out by the participant before the game session. The biometric data further includes 6 types of data: (1) *ACC.csv* - accelerometer results measuring the movement of the wearer, (2) *EDA.csv* - the electrical conductivity of the skin, (3) *BVP.csv* - data from the photoplethysmography sensor measuring BVP at 64Hz, (4) *IBI.csv* - the interbeat intervals, (5) *HR.csv* - the average heart rate in the spans of 10 seconds, (6) *TEMP.csv* - the temperature of the player.

B. Data analysis

The dataset is very imbalanced. The ratio of labeled to not labeled frame is around 1:800, varying from playthrough to playthrough. This is a problem we need to solve in the machine learning approaches because not labeled frames will dominate the classification process.

After viewing the dataset, we decided that we will use the biometric data and game frame data in the classification process. We found that the facial expression in the recordings of the participants' faces varies from person to person, from time to time, which is not very stable data to extract features from, and thus we will not use them in the project. Since the biometric data is tabular and relatively hard to process by mathematical algorithms, we will use them in training the machine learning model; as for the game frame data is very stable in terms of data feature extraction, we will use the extracted feature in a sequence of arithmetic algorithm to determine the game events. Each game frame is a RGB image of size 256 pixels times 240 pixels. The score, stage, and time information is displayed at pixel [24:31, 32:55], [24:31, 152:175], [24:31, 208:231] respectively, with each number occupying 7x7 pixels.We call them score display, stage display, and time display.

C. Data preprocessing

The events are labeled as integers like the following with format {label: event}:- {0: new_stage, 1: flag_reached, 2: status_up, 3: status_down, 4: life_lost, 5: not_event}

1) For biometric data, we first integrate the data into a single .csv file, with [participant number, event number, frame number, EDA, BVP, HR, TEMP] datas as the columns. Since we believed biometric data changes differently among different persons. Therefore, for each participant, the data value is rescaled linearly in between 0 and 1 before integration.

2) For game frame data, we extract the 7x7 pixels for each frame, convert them into grayscale and save them as a numpy array. Then, we standardize the pixels in the array, making the pixel value minimum as 0 and maximum as 1. Finally, we identify the number the array represents based on a library of number pixels we build. This library contains pixel arrays we identified for each number from 0 to 9, the identified arrays are mainly extracted from all the frames of participant 0's gameplay. In the identifying process, We check if the number arrays from the testing frame equals to any of the number arrays in the library. If it equals, the number the library array represents is assigned to the array. In the end, we can find out the values of the frame displays in integers. However, for most frames in the playthrough, the values do not change, to reduce the amount of data to process, we only record the frames with the values or pattern changed. To ensure the number could be identified correctly, it requires the library to be large enough. Fig. 1 shows some identified number arrays representing the number 0, Fig. 2 shows some not accepted number arrays.

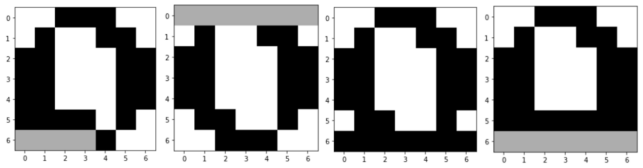

Fig.1. Identified number array of number 0

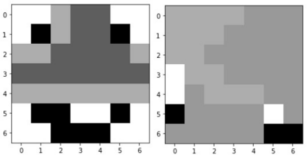

Fig.2. Identified number array of number 0

III. Machine learning Approaches

We will approach this challenge using 4 models, (A) *Linear Regression*, (B) Support Vector Machine aka. *SVM*, (C) *Random Forest*, and (D) Neural Network. The following will just be a brief description of what we did because the models were later replaced by the image analysis approaches.

A. Linear Regression

The first model that comes from our idea is directly Linear Regression, which is suitable for the data which is linear. Although we thought that it may not work since the Linear Regression model is for continuous data, we still try to get some insight and hope there could be some relations between the biometric data and the event number. Regrettably, the biometric data is too similar to classify the event number. They all predict the same event, which must not be true.

B. Support Vector Machine (SVM)

With the calculated distance between the boundary and class data, SVM attempts to find boundaries in the hyperplane to better split class data. We applied our integrated dataset on the SVM. Fig.3. shows the confusion matrix of the model testing on one set of participant's biometric data, we can see that the model predicts all data as label 5, "not_event". The reason is that our dataset is highly imbalanced. For simply one participant, there are over 120,000 pieces of data labeled "not_event" while others in total are less than 200. This shows that the model is not successful at detecting events. Therefore, we turned into another classification model, random forest, with the hope of solving the data imbalance problem.

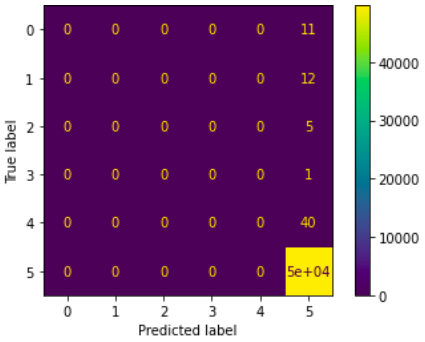

Fig. 3. Confusion matrix of the SVM model's test result

C. Random Forest Classifier

A random forest model consists of several decision trees. The model utilizes ensembling learning, and is established based on the performance of each decision tree to avoid overfitting.

We first did the hyperparameters tuning to optimize the performance. Adjusted parameters included "num_trees", "max_features", "min_samples_split", "min_samples_leaf", "class_weight", "bootstrap", "criterion", and "oob_score". Macro-F1 was adopted to evaluate the cross-validated model. From Fig. 4.,it is obvious that most of the frames were still classified as label 5, "not_event". However, there are some improvements compared to the SVM model. Some frames can be classified as other events although the accuracy is rather low. The model has accuracy with only 0.1 and macro-F1 score with 0.03 which is nearly the same as random classification performance, 0.02.

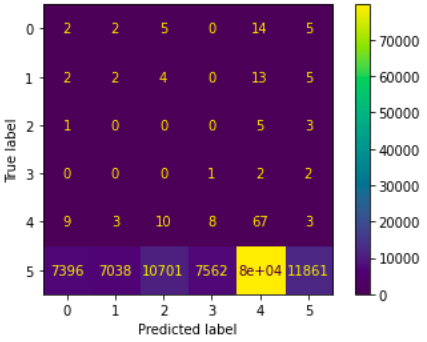

Fig. 4. Confusion matrix of the random forest classifier's test result

D. Neural Network

With a neural network, the model will train itself with the training dataset we provided to the classifier. The structure of the model is as the following table:

TABLE I. NEURAL NETWORK MODEL SUMMARY

Model Summary		
Layer type	***Output Shape***	***Param***
Dense Layer	(None, 2862)	11448
Dense Layer	(None, 256)	732928
Dropout Layer	(None, 256)	0
Dense Layer	(None, 256)	65792
Dropout Layer	(None, 256)	0
Dense Layer	(None, 6)	1542

The input length of the first layer or the model is 4, as we have four data types as training data, namely EDA, BVP, TEMP, and HR. Total trainable parameters are 811710 weights. To combat the imbalanced data problem, we upsample the labeled training data to make the training data balanced and improve the accuracy. To prevent overfitting, we will stop the training early if the loss of the testing set is decreasing but the loss of the validation set is increasing for a few epoches, *dropout* layers are also implemented to further reduce overfitting. Also, the training set is also weighted when fitting the model.

However, with various tries, adding/subtracting different layers, trying out different layer types, eg. LSTM & Activation layers, and using different activation functions and metrics, the result still comes out unsuccessful. Fig.5. is the confusion matrix of the model, you can see that the model cannot identify the events successfully. All labels are misclassified as event 4. As a result, we move on to the image analysis methods.

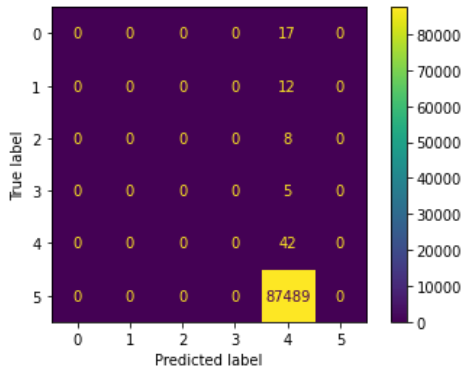

Fig. 5. Confusion matrix of the neural network's test result

IV. Game Frame Analysis Approaches

Following the machine learning approaches, the game frame analysis approach will go into a completely different direction. In this approach, we focus on identifying the game data, namely current score, current stage, and current time, and using the datas and the game rules to detect game events and classify them. Whenever the numbers are not indistinguishable, the frame will not be processed, since the numbers can be distinguished in 1-5 frames after the indistinguishable frame, which is tolerable because the numbers are rarely indistinguishable. The following section will explain how each of the events is detected and classified.

The *new_stage* and *flag_reached* events focus on the change on the stage numbers; the *status_up* event focuses on the changes on score numbers; the *status_down* event focuses on the time numbers; and finally, the *life_lost* event relies on all the three number displays.

A. New Stage Event

On the top of every game frame, there is a section displaying "WORLD {*number*}-{*number*}", as shown in Fig.6. This is the stage display section consisting of two numbers and a dash line. We observe that when the number in the stage section changes, there is a "new stage" event, and only when this event happens, the numbers will change. The stage number will not change in other frames that's not *new_stage* event. Additionally, the time display will be reset back to 400 or 300 depending on the stage our Mario is in during the event.

Fig. 6. Stage display

So whenever the stage number changes, we flag the frame as *new_stage* event. From the validation result, we found that this event is classified with relatively high accuracy. This is also partly due to the numbers not being blocked or distorted when Mario warps back to the start of the stage during a *new_stage* event.

B. Flag Reached Event

By looking at the ground truth data, we found that the *flag_reached* event normally happens before the *new_stage* event, so we can utilize the predicted *new_stage* event as a pivot and say that *flag_reached* events happen 1 frame before *new_stage* events.

C. Status Up Event

The game constantly adds scores throughout the gameplay as displayed in the score display, with different amounts of scores added in different events. The score display is shown in Fig.7.

Fig. 7. Score display

Focusing on the *status_up* event, we noticed that the score will increase by a thousand (1000) points every time the event occurs. Also, the score only increments in steps of 100 points, meaning we can disregard the last two 0's completely as they remain unchanged. Based on this observation, we can conclude that whenever the score increases by a thousand points, *a status_up* event occurs.

D. Status Down Event

In the top right corner of every game frame, there is a time display, as shown in Fig.8, which is the time limit for Mario to achieve his task in one game stage. Normally, the time display counts down one for every 20 or 21 frames. However, it is observed that when the "status down" event happens, Mario will shrink, and during the transformation, the time display stays at the same number longer than 21 frames.

Fig. 8. Time display

So, to detect the *status_down* event, we only need to measure each of the length of time ticks, if one particular number stays for more than the threshold 20 frames, it'll be identified as a status_down event.

However, since the time might freeze for more than two times the threshold, around 50 frames, two *status_down* events might arise from the same actual in-game event. To combat this, we set a counter *lock_counter* to count the frame counts from the last *status_down* event. If the counter is still within a 100 frame (from the last *status_down*), the *status_down* event flag will not be raised.

E. Life Lost Event

The "life lost" event indicates that Mario has to restart his task. Therefore, we find out that there will be three incidents on the displays: (1)the score will be reset to 0, (2) the time limit will be reset to 400 or 300 depending on the

game stage our Mario is in, and (3) Mario will also warp to the start of the stage at the *life_lost* event, the stage will not change.

We set the indicator of the *life_lost* event as follows: whenever the score becomes 0 and the time is reset to the stage's time limit, then it is a *life_lost* event. But there are some exceptions with this indicator. On a *new_stage* event, the time and score also resets, this could lead to error in classification. So to solve this problem, we will reuse the counter from identifying *status_down* events. Whenever the *new_stage* event occurs, we will set the counter back to 50 frames, this prevents the *life_lost* event to be raised, and we will set the priority of the *new_stage* event higher than the *life_lost* event, partly due to the high f1-score performance of the *new_stage* event detection. With this priority set, whenever there is a frame labeled with these two events (*new_stage* & *life_lost*), *new_stage* will be set to that frame and the *life_lost* flag will be removed.

V. Detection & Classification Results

In this section we will present the results of the different methods we implemented.

A. Machine Learning Approaches Results

In Table 2, we state the f1-scores of the 4 models we tried.

TABLE II. ML Model Performances

Methods			
Linear Regression	***SVM***	***Random Forest***	***Neural Network***
0.10	0.17	0.17	0.170

You can see that all the models have bad performances. From the confusion matrices seen in the previous section, the labels produced by the classifiers all fall to a few certain events, the model thinks all the frames are a certain event but rarely the other.

B. Game Frame Approach Result

In Table 3, we state the performance of the game frame analysis model by each participant.

TABLE III. Game Frame Analysis Model Performance

Game Frame Analysis Model Performance				
Participant number	***Average Distance***	***Precision***	***Recall***	***F1-Score***
0	0.0494	0.931	0.959	0.945
1	0.125	0.916	0.976	0.945
3	0.744	0.860	0.936	0.897
5	0.553	0.888	0.945	0.916
6	0.482	0.881	0.953	0.916
8	0.492	0.939	0.989	0.964
9	0.625	0.813	0.944	0.874
Average	0.439	0.890	0.957	0.922

The performance of the game frame analysis model is highly accurate. It performs well at detecting the game events, but might falsely detect frames that are not events, as seen from the lower precision score and higher recall score.

VI. Discussions on the Results

A. Machine Learning Approaches

The machine learning methods apply biometric data, BVP, EDA, HR, and TEMP on several different models. However, we can see that their performances are not good. We believe that there are two reasons behind this. First, our dataset is highly imbalanced. The ratio of label 5 and other labels in all training dataset is about 860:1 which makes our models hard to predict other labels. Second, the biometric data alone might be simply not strong enough to split the events, emotions are still too complex for simple machine learning models; by simple PCA analysis, we can see that the dataset is all meshed together. Fig.9 shows one PCA plot on the top two principal components analyzing on EDA, BVP and HR data, with blue colored points as unlabeled frames and the other colors as labeled frames from 0-4 coded labels. There is no clear distinct separation between event and no event frame. To improve the performance of the machine learning approach, we can utilize the game frame data and also connect the emotions detected from the face cam video to the biometric data to further identify the player's emotion.

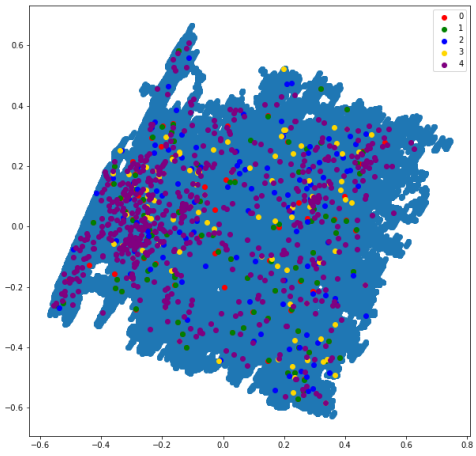

Fig. 9. Principal Component Analysis On EDA, BVP, HR data

B. Game Frame Analysis Approaches

We are pretty satisfied with the performance of the game frame analysis model, but there are still rooms to improve.

There is a known problem being whenever Mario bets another game NPC called Hammer Bro, the score also adds 1000 points. This event will be misclassified as a *status_up* event, which is wrong. We hope to implement machine vision to identify Hammer Bro and try to seperate the two events by removing the *status_up* flag when Mario bets Hammer Bro.

We later also found out that some *new_stage* events are not after the *flag_reached* event. This is because Mario is

able to change the stage through some way we call “special passage”. Some of these transitions are stated in Table 4. These are not the ordinary sequence of stages that mario goes through, and so the flag is not reached. We propose that the event of going through “special passage” can be identified and we can remove the output of that particular *flag_reached* event accordingly. The proposed method is by keeping track of the game stage Mario is in, and updating the stage during *new_stage* event, and we use a library to identify when the transition is through one of the “special passages”, since the “special passages” are only limited to a certain number.

TABLE IV. “SPECIAL PASSAGES” EXAMPLES

Special Passages			
Old stage	***New stage***	***Old stage***	***New stage***
1-2	1-3	2-4	3-2
1-3	2-2	3-1	3-2
1-4	3-1	5-2	5-3
2-1	2-3	5-1	4-2
2-3	2-4	6-4	7-1

Lastly, the number library can still be improved by completely changing it from a fixed library to a machine vision model and use it to identify what number the array represents. This way the range of different arrays representing the same number can be broadened.

VII. CONCLUSION

We tried machine learning approaches to analyze biometric data at the beginning, but as you can see from the results above, analyzing more straightforward data such as game data and numbers are easier than analyzing very complex data such as emotional and biometric data. Hence, we abandoned the machine learning methods to move on to the game frame analysis approaches. We hope to fix the current known problems with this model, further implement machine learning methods and utilize the biometric data onto the game frame analysis model to improve its performance even more in the future. Since Super Mario Bros. is a relatively simple game, we aim to achieve at least 0.97 f1-score or above. This concludes this report.

REFERENCES

[1] 為自己Coding, @CHWang on Matters News, “Machine Learning - Linear Regression迴歸模型 - 強大的Sklearn - 簡單線性迴歸模型、多項式迴歸模型、多元迴歸模型 - 完整實作教學”, Website: https://matters.news/@CHWang/92854-machine-learning-linear-regression%E8%BF%B4%E6%AD%B8%E6%A8%A1%E5%9E%8B-%E5%BC%B7%E5%A4%A7%E7%9A%84sklearn-%E7%B0%A1%E5%96%AE%E7%B7%9A%E6%80%A7%E8%BF%B4%E6%AD%B8%E6%A8%A1%E5%9E%8B-%E5%A4%9A%E9%A0%85%E5%BC%8F%E8%BF%B4%E6%AD%B8%E6%A8%A1%E5%9E%8B-%E5%A4%9A%E5%85%83%E8%BF%B4%E6%AD%B8%E6%A8%A1%E5%9E%8B-%E5%AE%8C%E6%95%B4%E5%AF%A6%E4%BD%9C%E6%95%99%E5%AD%B8-bafyreidbro25dokrljhrdiwfsyc2gg345yiym2uirxu4l3nc6zbxntdaqq

[2] Michael Galarnyk, Towards Data Science, “PCA using Python (scikit-learn)‘, Website: https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

[3] 野风同学, 知乎, “支持向量机（SVM）——原理篇”, Website: https://zhuanlan.zhihu.com/p/31886934

[4] Chung-Yi, Medium, “ML入門（十七）隨機森林(Random Forest)”, Website: https://medium.com/chung-yi/ml%E5%85%A5%E9%96%80-%E5%8D%81%E4%B8%83-%E9%9A%A8%E6%A9%9F%E6%A3%AE%E6%9E%97-random-forest-6afc24871857

[5] Jason Brownlee, Machine Learning Mastery, “Tour of Evaluation Metrics for Imbalanced Classification”, Website: https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/

[6] Bert Carremans, Towards Data Science, “Handling overfitting in deep learning models”, Website: https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e