




Artificially Intelligent Game Framework Based on Facial Expression Recognition

Itisha Patidar, Karan Sanjay Modh, and Chiranjoy Chattopadhyay^(✉) 

Indian Institute of Technology Jodhpur, Jodhpur, India
{patidar.1,modh.1,chiranjoy}@iitj.ac.in

Abstract. During gameplay, a player experiences emotional turmoil. In most of the cases, these emotions directly reflect the outcome of the game. Adapting game features based on players' emotions necessitates a way to detect the current emotional state. Researchers in the area of "video game user research" has studied biometric data as a way to address the diverse characteristics of players, their individual preferences, gameplay expertise, and experiences. Identification of the player's current state is fundamental for designing a game, which interacts with the player adaptively. In this paper, we present an artificially intelligent game framework with smart features based on automatic facial expression recognition and adaptive game features based on the gamer's emotion. The gamer's emotions are recognized at run-time during gameplay using Deep Convolutional Neural Networks (CNN), and the game is adapted accordingly to the emotional condition. Once identified, these features directly modify critical parameters of the underlying game engine to make the game more exciting and challenging.

Keywords: Facial expression recognition · Deep learning · CNN · Game engine · User experience · Game interface

1 Introduction

Artificial Intelligence (AI) has seen tremendous progress in recent years. It is a prosperous research field containing a cumulative number of vital research areas, as well as an essential technology for a growing number of application areas. AI in games was started with board games, and then slowly graduated into video games, where researchers are making a significant contribution in terms of developing new algorithms for the game engine, rendering, human-computer interaction, etc. As a result, video games, in their present form, have become intelligent. Analyzing game statistics and learning the critical performance parameters have been the area of interest to many AI researchers. In the literature there has been efforts to combine biometry based feedback for making the game engaging. For facial expression recognition works like [8, 10, 14, 15, 17] have demonstrated significant contribution. Also there were efforts to create a dataset [6] solely based on the expressions of the player while playing the game. Apart from emotional

states, efforts were made to capture the gamer's psychological and physiological states in the area of user experience research based on biometry and also game-user research through biometry [9,11]. In [1], a facial expression based game personalizing technique was proposed. Bio-feedback in games is another area where very recently researchers have started looking into and made significant contributions [4,7,13,16].

In this paper, we provide an artificially intelligent game framework that adapts to the users emotional status by analyzing the facial expression during game play. We hypothesize that a user will be happy on winning points bonus in the game. On the other hand, the user will be in a sad mood if the performance in the game is not well. Under this assumption, to make the game more engaging and to keep the gamer motivated, by introducing two key features. They are: (1) if the user is happy (i.e., the facial expression reflects happiness), then the game automatically adapts to itself and makes it more challenging, (2) for sad facial expression, we introduce reward for the gamer and thereby makes it engaging. One thing to be noted here is that we do not change the level of the game, which is by default is more challenging. We change the difficulty of the game in the same level by updating the critical parameters and features of game by interacting directly with the game engine.

The paper is organized in the following way: Section 2 presents the proposed framework in details. In Sect. 3, we present the details of the experimental studies. Finally Sect. 4 concludes the paper.

2 Proposed Framework

Figure 1 depicts the flow diagram of the proposed framework. There are three critical components in the framework, namely (i) Player's Emotion Recognition, (ii) Game Engine, (iii) Adaptation Rules. In the following subsections, we present the details of the proposed artificially intelligent game framework.

2.1 Player's Emotion Recognition

We leveraged the *Mememoji* [3] model to extract the facial emotions of the user in real time, and adapted it as per the requirements. Figure 2 depicts the overall facial expression recognition framework. OpenCV was used for face detection in the image. Detection of objects in an image using Haar feature-based cascade classifiers has proven to be an effective method. In this work, we have used the Viola-Jones Face Detection technique [12] to detect and crop the face of the gamer from the input video stream. Next, to ensure that all the face images are of the same dimension, the cropped regions are converted into 48×48 sized grayscale images. The resultant matrix is linearized into an array of dimension $1 \times 48 \times 48$ and fed into the input layer. The input vector goes through the Convolution2D layer. The filters in the convolutional layer is a small matrix of size 3×3 , which are convolved with the original image and yields a feature map.

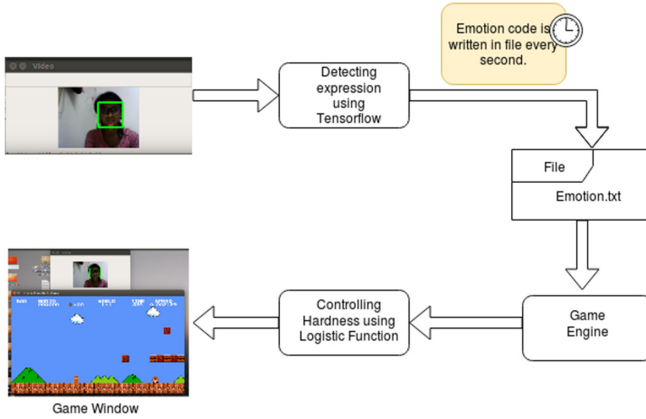


Fig. 1. Schematic representation of the proposed game framework.

The next processing stage is Pooling (in our case, Max Pooling is used) to reduce the dimension and results in computational efficiency. In this work, a (2, 2) pooling windows is used that scans through the feature map and keeps the maximum pixel value. The next processing stage is the dense layer or fully connected layers. It takes a large number of input features and transforms features through layers connected with trainable weights. These weights are trained by forwarding propagation of training data then backward propagation of its errors. The final segment in the network is the output layer, where the Softmax activation function is used. This output presents itself as a probability for each emotion class. To finalize the configuration of the network, we have conducted several experiments, and empirically determined the number of each type processing layers. In the end, the final neural network architecture that yielded the best performance has 9 convolution layers, and after three consecutive convolution layer, there is one max-pooling layer.

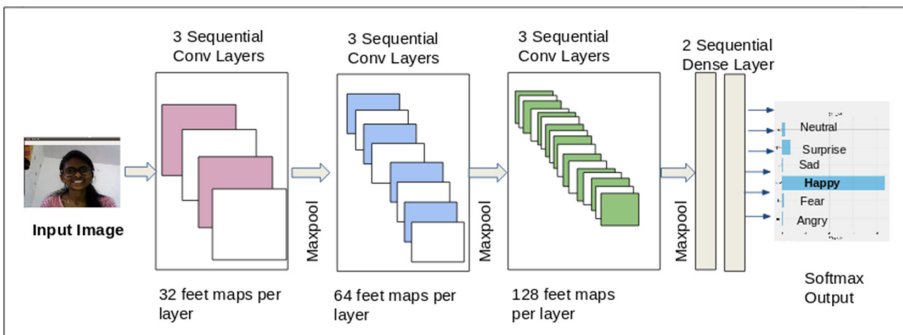


Fig. 2. Architecture of the convolutional neural network.

2.2 Game Engine

We consider the classical video game “SUPER MARIO BROS” and have made further enhancements to its game engine, and thereby making it adaptive to the individual player concerning experienced challenge applying PCG (Procedural Content Generation) [5]. The goal of our approach is to generate online game spaces such that the spaces optimize player to challenge for the individual player, by modeling the parameters like speed attributes, number of enemies, changing the abilities of the player, etc.

The working structure of the game is a huge Matrix, in which a map is loaded first based on the level which the user is playing. The map consists of blocks/bricks placed at individual squares of matrices, and also all enemies are loaded at the beginning and are called based on their position as a square of the matrix. A check on the state of the player is kept at every square of the matrix through which it passes, which responds whether the player is alive or dead. So we analyzed this structure and implemented the dynamic change of the game on each square of the matrix (where it checks for player state) and applied an algorithm to alter the attributes with the help of ‘mathematical modeling’ (discussed in Sect. 2.3) whenever the physiological states (emotions) of the players’ changes. A key contribution of this work is that game personalization techniques can leverage novel computer vision-based techniques to infer player experiences automatically based on facial expression analysis unobtrusively.

2.3 Mathematical Modelling

Mathematical functions are used for personalizing the various parameters (listed in Table 1) of game depending on classifications of the user’s facial expression – to the end of tailoring the affective game experience to the individual user. To dynamically change the game attributes we require a function which can regulate personalization such that the game remains playable and players’ could remain engaged. To regulate the challenge level a *hardness parameter* (x) is used which at the start of game is set to zero. Player’s emotions are then tracked using facial expression recognition model through out the entire duration of the game session, which are further used for sampling the hardness parameter. This is then applied on a mathematical function to give personalized game features depending on emotions.

Algorithm 1 is currently used for personalizing game features. The FER model gives percentage of distinct emotions, namely (0) angry, (1) fear, (2) happy, (3) sad, (4) surprise, and (5) neutral in an array, depending on the progress of the player through the Mario game. The index of expression with maximum percentage is taken as the expression code β . These expression code are then used for sampling hardness parameter x , which further optimizes game. The Algorithm 1 is called for every frame. For every 10^{th} frame we take the decision and update the game, as shown in line no. 6 to 12. The sampling parameter shown in line no. 9 is taken as 2.5 for logistic function and 0 for linear function

Algorithm 1. Optimising game using Linear Function

```

1: procedure PERSONALIZE( $E$ ) ▷  $E$ : Emotions array
2:    $x \leftarrow 0$ 
3:    $\alpha \leftarrow \max(E)$  ▷ Maximum probability of an emotion
4:    $\beta \leftarrow \text{index}(\alpha, E)$  ▷  $\beta$ : Expression code
5:    $\mathcal{E}.\text{append}(\beta)$  ▷  $\mathcal{E}$ : Expression array
6:   if  $\mathcal{E}.\text{Length} = 10$  then
7:      $\beta \leftarrow \text{mode}(\mathcal{E})$ 
8:   if  $\beta = 0 \vee \beta = 1 \vee \beta = 3$  then
9:      $x \leftarrow x - \gamma$  ▷  $\gamma$ : Sampling parameter
10:  if  $\beta = 2 \vee \beta = 4$  then
11:     $x \leftarrow x + \gamma$ 
12:   $\text{map} \leftarrow \text{OPTIMIZEGAME}(x)$ 

```

Table 1. Various game parameters and their update rule.

List of Parameter		
Parameter	On positive valence	On negative valence
Speed on enemies	Increased	Decreased
Number of enemies	Increased	Decreased
Frequency of power Ups	Decreased	Increased

Linear Function. In this approach, linear functions were used to map classifications of the human player’s facial expressions to appropriate in-game challenge levels. It was employed for optimising the challenge levels in the game such that the human interactions yield increased hardness level for positive valence (i.e., happiness, surprise), while decreasing hardness level for negative valence (i.e., fear, sad, anger) and no change for neutrality. Algorithm 2 optimizes the game based on the hardness parameter calculated from users affective states. This approach failed when emotional activity giving positive valence increases rapidly. The minions in game attains such a high speed that their trajectories do not even lie within the game frame. Also the number of minions increases very rapidly which flood the game frame making diversions which are uncontrollable.

Algorithm 2. Optimising game using Linear Function

```

1: procedure OPTIMIZEGAME( $x$ ) ▷  $x$ : Hardness parameter
2:   if  $x > 0$  then
3:      $\gamma \leftarrow 1.5 \times x$  ▷  $\gamma$ : Challenge Level
4:   if  $x < 0$  then
5:      $\gamma \leftarrow 0.8 \times x$ 
6:   UPDATEGAME( $\gamma$ ) ▷ Update game parameters

```

Determining speed of Game based on values of X using Logistic Function.

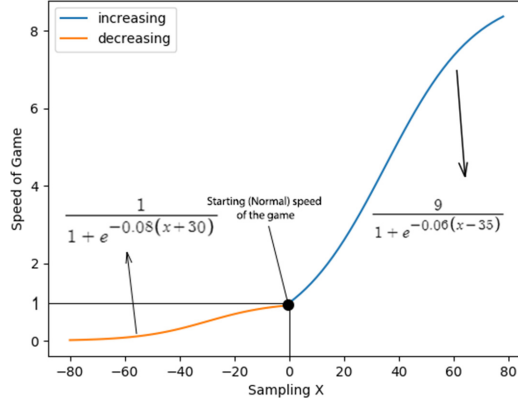


Fig. 3. Logistic functions for game personalization.

Logistic Function. The excessively high challenge levels imposes an unpleasant experience on game players. In order to avoid player abandonment resulting from an inappropriately high challenge level, a **logistic function** was formulated. This helped to remove the flaws of linear function by getting saturated after certain values of hardness parameter. Additionally, it also provided a cushion, in the form of graceful increase/decrease. We not only take into consideration player assessments made during actual play of the game, but also the playable conditions of the game as we observed during observational period many game players express high emotional activity, which could lead to extreme increase of speed or flood the game with enemies. To avoid such scenarios logistic function is useful by streamlining and controlling the flow from point of entry to the end of game. The logistic functions formulated for the algorithm as:

$$f(x) = \frac{1}{1 + e^{-0.08(x + 30)}} \tag{1}$$

The above function (Eq. 1) decreases the challenge levels in the face of user anger, while the following (Eq. 2) increases the challenge levels in the face of

Algorithm 3. Optimising game using Logistic Function

- 1: **procedure** OPTIMIZEGAME(x) ▷ x : Hardness parameter
 - 2: **if** $x < 0$ **then**
 - 3: $\gamma \leftarrow \frac{1}{1 + e^{-0.08(x+30)}}$ ▷ γ : Challenge Level
 - 4: **if** $x \geq 0$ **then**
 - 5: $\gamma \leftarrow \frac{9}{1 + e^{-0.06(x-35)}}$
 - 6: UPDATEGAME(γ) ▷ Update game parameters
-

Table 2. Comparative analysis of logistic and linear functions.

Using logistic function		Using linear function	
Increasing	Decreasing	Increasing	Decreasing
2.12098	0.90025	1.5	0.8
2.27666	0.880797	2.25	0.64
2.44998	0.858149	3.375	0.512
2.64183	0.832018	5.0625	0.4096
2.85283	0.802184	7.59375	0.32768
3.08328	0.768525	11.3906	0.262144
3.33303	0.731059	17.0859	0.209715
3.60145	0.689974	25.6289	0.167772
3.88739	0.645656	38.4434	0.134218
4.18909	0.598688	57.665	0.107374

user neutrality or happiness.

$$g(x) = \frac{9}{1 + e^{-0.06(x - 35)}} \quad (2)$$

Thereby, the online personalizing method operates as expected. Algorithm 3 optimizes game based on the hardness parameter calculated from users affective states. Graphs for these functions are shown in the Fig. 3. Table 2 shows how challenge level changes using both the functions, hence one can observe the need to discard the idea of linear function. The *updateGame* function call at line number 6 in Algorithm 2 and 3 updates the critical parameters of the game.

3 Experiments

Experiments were conducted on a computer having Ubuntu 18.04 with Intel i5 processor, and 8 GB memory. The entire programming was done on OpenCV, Keras and TensorFlow for face expression recognition, while SDL2 and CMake packages were used for the Mario game engine [2].

3.1 Methodology

We analyzed our system on the age group of 15–20 where participants interacted with the game under controlled experimental conditions like proper lighting, face exposed to the web cam for the entire duration, game starting at neutral hardness level. Our hypothesis is that the game would change its hardness level in correspondence to the emotional state of the user. We observed that the maximum

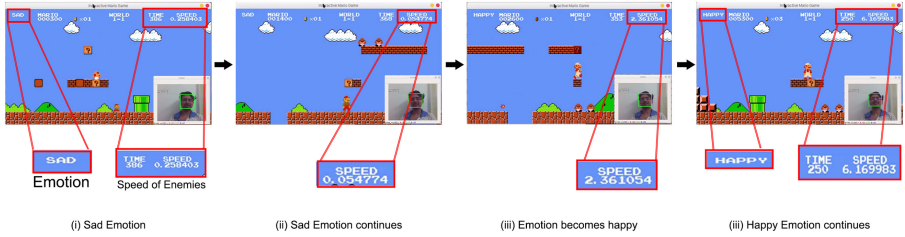


Fig. 4. Schematic representation of variation of speed based on user’s emotion. (i) As the user is Sad, the speed is less than normal i.e. 1. (ii) User continues to remain sad and hence the speed even decreases more. (iii) User becomes happy and the speed increases above the normal level i.e. 1. (iv) User remains happy and speed keeps increasing.

facial expressions were changed with 15–18 age group while with participants of higher ages the maximum observed emotion was sad and game decreased gradually. We observe the general trend where the algorithm decreases the hardness levels in the face of user anger, sad or fear and increases the hardness levels in the user’s emotion of surprise or happiness, hence validating our approach. Thereby, the mathematical model along with FER algorithm operates as expected. Also the framework maintains the levels in neutral face.

3.2 Qualitative Results

The mostly observed pattern showed that hardness is initially decreased because of “sad” levels. However, later in the game, high “neutral” and “happy” levels cause a increase in the hardness level. When lastly the “happy” emotion disappears, the hardness level becomes stable as well. Furthermore, we observe that the mathematical model appears stable in the face of classification noise, i.e. when the human player suddenly expresses a “mix” of emotions; denoting, in practise, that the player is talking or moving too much. As expected, the associated hardness level remains stable in the face of this noise from the facial expression classifier. This phenomenon is also in unison with our hypothesis.

Figure 4 shows the series of frames depicting the speed of enemies and emotion of the player. The facial expression is shown as inset to the frame. It can be observed from Fig. 4 (i) and (ii) that when the emotion continues to be sad for sometime, the speed parameter (that controls the speed of the enemies) is reduced to make the game easy, and thereby allowing the gamer to avoid the enemies and collect the bonus points. On the other hand, in Fig. 4 (iii) and (iv) depicts the situation when the gamer is happy. Here, the enemy speed is increased to make the game more challenging.

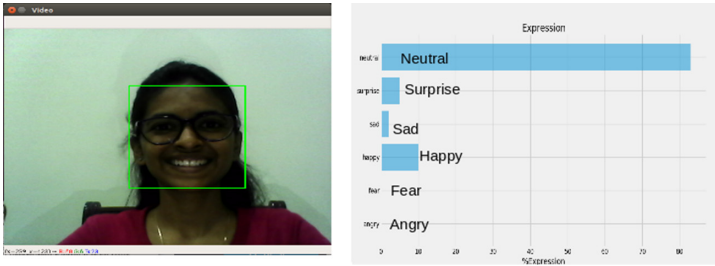


Fig. 5. Expression being wrongly recognized.

3.3 Failure Cases

As per the current model, the failure cases includes the following cases: (i) expression being recognized wrongly and (ii) expression not being recognized at all. For (i), expression recognized is not the correct expression. For example, the actual expression is ‘happy’, however, it being recognized as ‘neutral’. In such a scenario, our model would work according to the expression recognized, which is not expected (ideally). The solution to such scenario is design of a better by improving the efficiency of FER module. For he later scenario, there will be no effect on the game since no expression is detected. Case (ii) occurs when the player is (a) not in-front of the camera, (b) at a distance of more that 60 cm from the camera, or (c) in dark room/low lighting area. In terms of game, our Linear model failed since it increased or decreased the game speed to such an extent such that it is not playable, hence owing to this failure we came up with Logistic Model to control the hardness of the game. Moreover, if the game is exposed to multiple faces at the same time with different expressions, then the model recognizes multiple expressions at the same time and hence the result is not favorable. This issue can be resolved using single-face FER detection model. Figure 5 shows the failure case when the expression ‘happy’ is wrongly recognised as ‘Neutral’.

4 Conclusion and Future Work

In this paper, we presented a framework for changing the properties of a game when played based on classifications of the user’s facial expression. A significant contribution of this paper is that the proposed technique unobtrusive in nature, and does not required the game player to be conscious about it. In this paper, we augmented the Mario game engine and manipulate its features through Facial Emotions recognition. For future work, we will investigate how to make the framework more robust by learning the game engine features, and trying to generalize the method of game personalization.

References

1. Blom, P.M., et al.: Towards personalised gaming via facial expression recognition. In: AAAI, pp. 30–36 (2014)
2. Jakowski, L.: umario. https://github.com/jakowskidev/uMario_Jakowski
3. Jostine, H.: Mememoji. <https://github.com/JostineHo/mememoji>
4. Leahy, A., Clayman, C., Mason, I., Lloyd, G., Epstein, O.: Computerised biofeedback games: a new method for teaching stress management and its use in irritable bowel syndrome. *J. R. Coll. Physicians Lond.* **32**(6), 552–556 (1998)
5. Lelis, L.H.S., Reis, W.M.P., Gal, Y.: Procedural generation of game maps with human-in-the-loop algorithms. *IEEE T-G* **10**(3), 271–280 (2018)
6. Li, W., Abtahi, F., Tsangouri, C., Zhu, Z.: Towards an “In-the-Wild” emotion dataset using a game-based framework. *arXiv e-prints* (2016)
7. Lobel, A., Gotsis, M., Reynolds, E., Annetta, M., Engels, R.C., Granic, I.: Designing and utilizing biofeedback games for emotion regulation: the case of nevermind. In: CHI EA, pp. 1945–1951 (2016)
8. Lopes, A.T., de Aguiar, E., Souza, A.F.D., Oliveira-Santos, T.: Facial expression recognition with convolutional neural networks: coping with few data and the training sample order. *PR* **61**, 610–628 (2017)
9. Mirza-Babaei, P., Nacke, L., Fitzpatrick, G., White, G., McAllister, G., Collins, N.: Biometric storyboards: visualising game user research data, May 2012
10. Mollahosseini, A., Chan, D., Mahoor, M.H.: Going deeper in facial expression recognition using deep neural networks. In: WACV, pp. 1–10 (2016)
11. babaei Pejman, M., Sebastian, L., Emma, F.: Understanding the contribution of biometrics to games user research. In: DiGRA International Conference: Think Design Play, January 2011
12. Viola, P., Jones, M.J.: Robust real-time face detection. *IJCV* **57**(2), 137–154 (2004)
13. Weerdmeester, J., van Rooij, M., Harris, O., Smit, N., Engels, R.C., Granic, I.: Exploring the role of self-efficacy in biofeedback video games. In: CHI PLAY, pp. 453–461 (2017)
14. Bai, Y., Guo, L., Jin, L., Huang, Q.: A novel feature extraction method using pyramid histogram of orientation gradients for smile recognition. In: IICIP (2009)
15. Yao, A., Shao, J., Ma, N., Chen, Y.: Capturing au-aware facial features and their latent relations for emotion recognition in the wild. In: ICMI (2015)
16. Zafar, M.A., Ahmed, B., Al-Rihawi, R., Gutierrez-Osuna, R.: Gaming away stress: using biofeedback games to learn paced breathing. *IEEE T-AC* **11**, 519–531 (2018)
17. Zhang, J., Huang, K., Yu, Y., Tan, T.: Boosted local structured hog-lbp for object localization. In: CVPR, pp. 1393–1400 (2011)