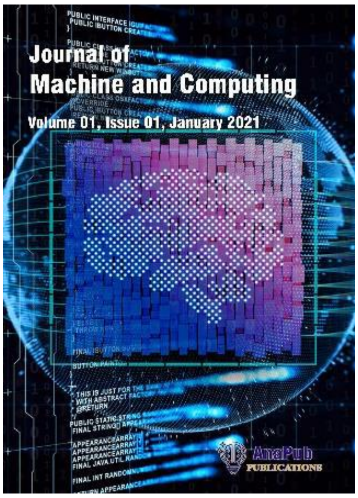
Journal Pre-proof



A Framework for Designing Behaviour Tree Artificial Intelligent Game based on Dynamic Human Emotions

Sreenarayanan N M and Partheeban N

DOI: 10.53759/7669/jmc202505059
Reference: JMC202505059
Journal: Journal of Machine and Computing.

Received 14 April 2024
Revised form 26 November 2024
Accepted 20 January 2025

Please cite this article as: Sreenarayanan N M and Partheeban N, "A Framework for Designing Behaviour Tree Artificial Intelligent Game based on Dynamic Human Emotions", Journal of Machine and Computing. (2025). Doi: https:// doi.org/10.53759/7669/jmc202505059

AnaPub
PUBLICATIONS

A Framework for Designing Behaviour Tree Artificial Intelligent Game based on Dynamic Human Emotions

[1]Sreenarayanan N M, [2]Partheeban N
[1]Research Scholar, [2]Professor
[1,2]School of Computing Science and Engineering
[1,2]Galgotias University, Greater Noida, Uttar Pradesh, India
[1]sree.narayanan1@gmail.com, [2]n.partheeban@galgotiasuniversity.edu.in

Abstract

An adaptive game design includes human emotions as a key factor for selecting the next level of harness in the game design. In most of the situation, facial expression could not identified exactly and this may leads to an uncertainty for selecting a next level in the game flow. An efficient game design requires an efficient behaviour tree construction model based on the emotions of a player. This paper presented an artificial intelligent based game design using behaviour tree model by including an efficient emotion detection or classification system using a deep learning model ResNet 50. The proposed technique classifies the emotion of a player based five different category and the player current state of mind will be calculated based on this emotion score. The Behavior Tree has been constructed from the foundation based on the hardness value calculated for each sub-BT. The performance evaluation for the emotion classification archives close to 92% and this accuracy will lead to construct an efficient BT for any game. We have evaluated the emotion detection system with other related Deep learning models

Keyword: Behavior Tree, Game Engine, BT Editor, ResNet 50, AI Emotion Detection

1. Introduction

A Behaviour Tree (BT) is graphically represented as tree for the plan of execution for a game, such shown in Figure 1. The top-bottom approach is a way to represent the information hierarchically and it is extensively used in the field of Computer Science [5]. Figure 1 is a general BT tree representation and shows the overall stage plan and accepted answer for the particular stage. We can develop any kind of game as a BT tree based on conceptual and plan view.

Behavior Tree usage has emerged as a prevalent tool for exhibiting artificial intelligence (AI) in games design is Behaviour Tree Halo 2 [6] game [7][8]. The Figure 1 is an example for decision making support system for Non Player Character (NPC) [9]. This is one of the effective method for handling known situations and situation becomes as an event [10][11]. Generally, the behaviour of an entity is defined by using four methods other than BT:

1. A normal programming language has been used for writing procedures [12]
2. The behavior between states and transitions are described by using Finite State Machines [12]
3. The plan for an entity has been create by using an AI Software called a planner and this will be known as Hierarchical task networks. This is equivalent to a behavior tree.
4. Scripts is a scripting or behaviour language for representing the behaviour and this will be in a high level language [13][14]

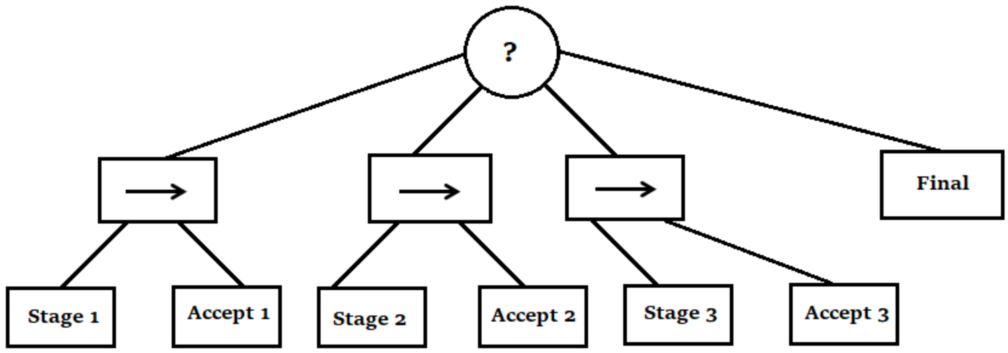


Figure 1: Behaviour Tree Construction

Traditional game loop has been designed with three separate and inter-related phases of getting input and process it, enhance the game world, and attain output for the corresponding input. The basic game loop has been designed in a high level design look like,

While game is running Mode

1. Getting and Process *inputs*
2. *Update* or Enhance Game Environment
3. Produce *Outputs*

End of Game Loop

The above discussed phases are need more depth activities for making changes in the world game design. The inputs are gathered from different types of devices and treated it for changing the game environment accordingly. In general, a game engine repeatedly executes the game loop and these steps include sub-engines in running mode [4]. The artificial intelligent sub-engine runs the code associated with current game entity. This will be a Non Player Character (NPC) role play from time to time.

The Behaviour Tree construction is an important part for the designing an game from step by step. The working principle of BT is processed from root not and proceeds according to the pre-order traversal [illegible]. For example in figure 1, first root node is executed and moves to the left most children from the root node. Each for the child node is processed from left to right and this will be executed in the same fashion. The figure 1each of the shapes represents one node and two nodes are connected in a common line. The parent node will be executed first and lower level child node will be executed next based on the direction given by the parent node. The Behavior Trees are designed to specify the behaviors of game movement based on their maintainability, scalability, reusability, and extensibility [15] [16] [17] [18][19][20]

1.1. Organization of Paper

The remaining section of paper organized as follows, section 2 discussed about the related work on emotion detection system using facial expression analysis and Behavior tree construction methods using behavior tree editors. The detailed methodology and BT construction details are given in the section 3. The section 4 provides an explanation about the proposed technique for constructing the game behavior tree based on the pler emotional factors. The section 5 discussed about the result and discussion part for the proposed technique based on emotion detection or classification using facial expression dataset. The performance evaluation provided for the proposed technique in the following sections. The section 6 concluded with future direction for the BT tree construction methods based emotional factors

2. Related Work

The related work section discussed about the two important aspects, Emotion detection from facial expressions and game design based the construction of Behavior tree

Giuseppe provided a comparison of detection accuracy of deep learning approaches including with facial innovations performs well in face verification task. Ebrahim et al illustrated the significant facial signals to perceive attentional states by assessing a dataset of participants with 15 numbers by including their challenges and engagement levels. Sasanko et al. [11] provides a detailed study and analysis of some existing algorithms for detecting emotions from facial expressions. Few methods for feature extraction are designed for identifying facial expressions from like videos and standard images. Zhanna et al [12] showed the accuracy of visual mood inference algorithm for producing preview of a facial image in Google Mobile Vision API. Zhanna et al. [13] discussed about the relationship between the smartphone applications and user emotions using bidirectional causal relationship method. They have made a study with participants of 30 with 502,851 examples from application usage in smartphone in a tandem with consistent emotional data from facial expressions. Kangning et al [14] selected and investigated five prominent commercial based emotion detection or recognition system and evaluated the performance of these systems. They have provided some recommendations for the developers who to design an facial emotion classification technology for their applications.

Clark et al. [15] provided a systematic review of FACS working principle of human emotion detection for consumer product based stimuli.

Krumhuber et al. [16] presented a survey on aims to testing result in a cross-corpus for dynamically changing facial expressions. They have used 14 databases with featured facial expressions of simple emotions.

The Behavior Tree are crated and maintained by using software package BT editor. This section discussed about the various BT construction software like Behavior Designer [17], Behave [18], Behavior3 [19], Brainiac Designer [20], and Unreal Engine Behavior Trees.

All software packages are managing a collection of BTs; from this we can open for editing or making changed in multiple BTs. We can use drag-and-drop facility for making changes in all the BT editors for constructing new BTs. Most of the BT editors are GUI based and this will provide a excellent view of making any update over the existing BT. The modules can be freely moving everywhere in the screen. The auto arrangement of components are available in [18] and [20] into an artistically pleasing format. We can add comments to any type of individual component by using Behavior Designer about specific sub-behaviors.

The designers are allowed to extend the basic architecture by including new components through the methods. The external events to make some interrupt for processing of a BT is available.

Behavior and Brainiac Designer are independent from the planned game environment. The Behavior Designer, Behave, and Unreal Engine BTs are knotted to definite developing environments. The latter has been corrects the features exact to Behavior Trees. For an example for BT editor, we define Behavior3, which is an open-source visual BT editor. The following facilities are provided by the Behavior3 action flow, condition flow, sequence flow, and selector based on priority elements and other designing related items.

Recent studies on artificial emotions in video games highlight advancements in AI-driven emotional interactions. One approach introduces an appraisal-based chain-of-emotion architecture using large language models to enhance NPC believability. Research on game-based learning emphasizes the role of affective computing in optimizing cognitive abilities by recognizing and responding to emotions. Emotional AI is being explored to personalize gaming experiences, with frameworks focusing on empathetic AI to support player resilience. Additionally, new facial emotion recognition models for VR gaming overcome the challenge of head-mounted display obstructions. These advancements underscore the growing role of artificial emotions in enhancing immersion, interactivity, and mental well-being in gaming.

3. Methodology

This section discussed about the basic concept of "behavior tree" constructions with detailed study about the tree design algorithms

Behavior Trees are a powerful system for decision making used in the game AI of the control system for non-player characters and agents. They find a wide area of application nowadays in modern game development due to their modularity, scalability, and ease in debugging.

A Behavior Tree is a hierarchical structure that determines what an AI entity will do according to certain preconditions and logic. It is comprised of several node types, which define the decision-making process.

Behavior Trees come from the robotics world but are widely applied in game development, especially when games are developed using AI like Halo, Assassin's Creed, and Unreal Engine.

3.1. Structure of a Behavior Tree

A Behavior Tree is a collection of nodes in a tree-like structure. The execution of the tree starts from the root node and continues through the branches to make decisions.

Types of Nodes in a Behavior Tree

- Root Node
  - The starting point of the tree.
  - It controls the execution of child nodes.
- Composite Nodes (Control Flow Nodes)
  - These nodes control the flow of execution by managing multiple child nodes.
  - Types of composite nodes:
    - Sequence (→) – Runs child nodes sequentially until one fails.
    - Selector (?) – Runs child nodes sequentially until one succeeds.
    - Parallel (||) – Runs all child nodes concurrently.
- Decorator Nodes
  - Changes the behavior of a single child node.
  - Common decorators:
    - Invert – Reverses success/failure output.
    - Repeat – Executes the node set number of times.
    - Cooldown – Ensures the node is not run too many times in quick succession.

- Leaf Nodes (Action & Condition Nodes)
  - Action Nodes – Perform actions such as moving, attacking, or interacting.
  - Condition Nodes – Evaluate specific conditions, say "Is the player in view?"

3.2. How Behavior Trees Work in Games

1. The root node begins execution
2. Processes composite nodes such as Sequence or Selector for determining the next action.
3. Condition nodes evaluate game states, like "Is enemy nearby?"
4. Action nodes execute based on decisions
5. Continue execution till a condition fails or an action completes

One instance of behaviour tree in NPC Enemy AI

Imagine an enemy NPC that patrols an area and chases the player when spotted.

```
Root
|
├── Selector (?)
|   ├── Sequence (→)
|   |   ├── Condition: "Can see player?"
|   |   ├── Action: "Chase Player"
|   |   ├── Action: "Attack Player"
|   |
|   ├── Sequence (→)
|   |   ├── Action: "Patrol Area"
|       ├── Action: "Idle"
```

Figure 2: Behaviour Tree Nodes

- The enemy chases and attacks the player if it views him.
- It patrols the place if the player is not in view.

3.3. Benefits of Behaviour Trees

- Modularity – It can use multiple reusable components to simplify AI logic.
- Scalability – Easy to scalable to implement complex AI behaviors.
- Debugging – This structure is very clear and helps in finding the issues.
- Flexibility – Multiple trees can be used together to build advanced AI behavior.

Behavior Trees is one of the powerful AI techniques applied in game development nowadays in creating intelligent, dynamic, and modular NPC behaviors. They help provide an organized, scalable, and easy-to-debug framework, thus enhancing the game AI by making characters more lifelike and responsive.

Sequence: Movement

For Each Node from Behavior Tree **do**
$StatusChild(Node_i) \leftarrow Tick(Child(Node_i))$
$\boldsymbol{Check\ condition}\ (StatusChild(Node_i) = \text{“}Running\text{”})$
  $return$ "Running"
$\boldsymbol{Else\ condition}\ (StatusChild(Node_i) = \text{“}Failure\text{”})$
  $return$ "$Failure$"
$\boldsymbol{End\ For}$
$return$ "$Success$"

Fallback: Additional

For Each Node from Behavior Tree **do**
$StatusChild(Node_i) \leftarrow Tick(Child(Node_i))$
$\boldsymbol{Check\ condition}\ (StatusChild(Node_i) = Running)\ \boldsymbol{Then}$
  "Running"
$\boldsymbol{Else}\ (StatusChild(Node_i) = Success)$
  $return$ "$Success$"
$\boldsymbol{End\ For}$
$return$ "$Failure$"

Parallel: Execution

N denoted total Nodes in BT
M denoted subset of Nodes in BT
For Each Node from Behavior Tree **do**
$StatusChild(Node_i) \leftarrow Tick(Child(Node_i))$
$\boldsymbol{If}\ (\forall M\ StatusChild(Node_i) = Success)\ \boldsymbol{Then}$
  "$Success$"
$\boldsymbol{Else\ if}\ (\forall(N-M)\ StatusChild(Node_i) = Failure)\ \boldsymbol{Then}$
  $return$ "$Failure$"
$\boldsymbol{End\ For}$
$return$ "$Running$"

4. Proposed Emotion based Behavior Tree Construction

This section proposed an efficient technique for constructing Behavior Tree using emotional factor. The emotional factor has been calculated from the facial emotion of a player. This section contains two sub-divisions of (1) Facial Emotion detection using Deep learning based classifier and (2) Behavior Tree construction based on hardness score. This proposed method works well for the Non Player Character with dynamic node changes in the Behavior Tree

4.1. Facial Emotion based Behacior Tree Construction

This paper proposed a framework for constructing a dynamic behavior tree for Non Player Character and this will helpful for the user to create an active participation in the game. The proposed method has two sub divisions, Initial Stage Behavior Tree Construction Phase and Dynamically changing States in Behavior Tree based on the Emotions from the player.

In the first phase, we have designed or constructed a BT with minimum level of hardness. The tree will be constructed with basic set of level and creates an interest in playing the game. The Initial stage of BT construction is done by using algorithm of 1, 2, and 3. Here, we have used an important scoring factor to mention the hardness level for individual game movement. The hardness score has been calculated by using equation ( ). The calculation of hardness score for a single level is depends on the following participating key elements in the game design, Minimum resource required for completing next level, level strategy, and average winning possibility for the particular level.

$$HardnessScore^{BT_i} = \sum \left(Min_{ResourceRequired} + \Pi\left(Level_{Strategy}\right) + Avg.WP^{BT_i} \right) \rightarrow (1)$$

Here $Min_{ResourceRequired}$ minimum resource required for completing the next level, $\Pi\left(Level_{Strategy}\right)$ indicates the level strategy, and $Avg.WP^{BT_i}$ mention about the winning possibility of next level.

The initial BT is designed with normal level of hardness and the remaining levels are designed with possible combination of hardness. The levels may be connected as a reference component to the current BT. Each level is designed with different set of strategy and hardness. These levels are indicated as BT_i and $HardnessScore^{BT_i}$indicates the hardness value for the particular BT_i.

4.2. Behavior Tree Construction based on the player emotions

This section proposed a method for constructing a BT based on the emotions of the player P_i. The emotion of a player has been identified by using an existing Facial Action Coding System (FACS) [3]. The FACS is a widely used model for detecting facial expressions. The movement of face parts are converted as action units to describe the parts of facial expressions. In the evaluation section, we have used an open-source OpenFace [4] software to extract AUs from each face image frame from the camera record during the game playing mode. We capture the face expression as an image from the video and apply minimum level of analysis for identifying the emotion of the player. The current state of the user mood is calculated as follows,

$$PlayerState_{P_i} = \left(EF_{P_i} \times P_i^{CST}\right) \rightarrow (2)$$

$$EF_{P_i} \leftarrow FACS(P_i)_{\{Sad,Angry,happy,Netural,DNT\}} \rightarrow (3)$$

Here, EF_{P_i} is indicated as an emotional factor captured from the FACS and P_i^{CST} is indicated as available set of resources for the player P_i

The following figure 3 explain the working principle of the proposed technique,

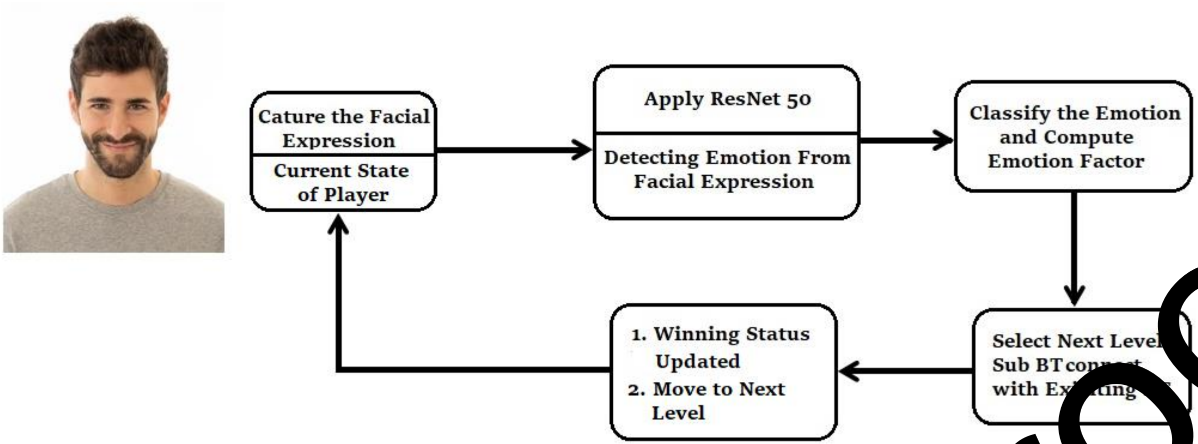


Figure 3: Proposed Behavior Tree Design Architecture based on Emotion Detection

The following figure 3 explain the simple game design with hardness score of Soft, Medium, and Hard

Algorithm 4 Proposed BT construction using Emotional Factor

1. Construct initial BT with n number of nodes $Node_i, 1 \leq i \leq N$
2. Construct sub-set of Behaviour Tree $BT_i, 1 \leq i \leq N$
3. Compute Hardness Score for each BT_i using equation (1)
4. $While(!EOG)$
5. *Begin*
6. Compute Emotional Factor EF_{P_i} for the player P_i by using the ***equation (3)***
7. Compute current state $PlayerState_{P_i}$ of the player P_i by using the ***equation (2)***
8. Based on the result value of the $PlayerState_{P_i}$ next level of the BT
9. Add Sub-BT using *Reference* Component
10. *End of While*
11. *End of Algorithm*

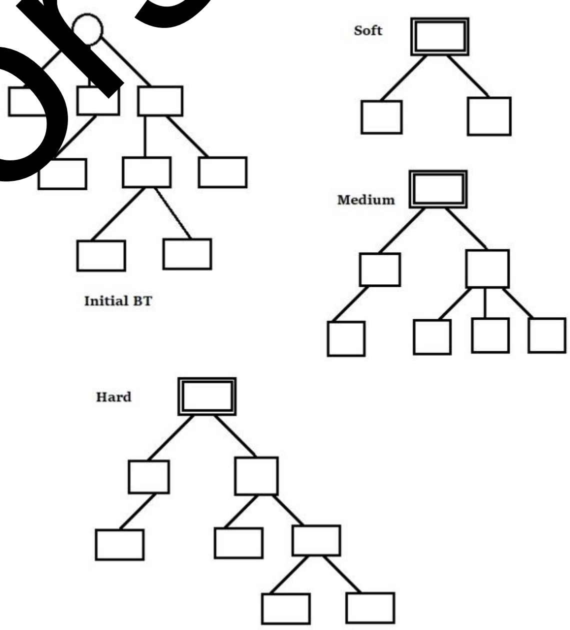


Figure 4: Construction of Behavior Tree for the proposed method

5. Result and Discussion

Computer vision applications are playing important for many real-time usages and Residual Network (ResNet) is an efficient model based on deep learning approach used for object identification and classification. ResNet model has been designed by using Convolutional Neural Network (CNN) architecture with different set of convolutional layers. The other types of CNN architectures are not suitable for increasing large number of layers. Due to this restriction in other CNN architecture, we can archive only limited performance. If we increase the number of layers in CNN models then this will face problem of "vanishing gradient". The ResNet classifier provides an innovative solution to the vanishing gradient problem. We have used ResNet 50 architecture for training the facial expression from the dataset.

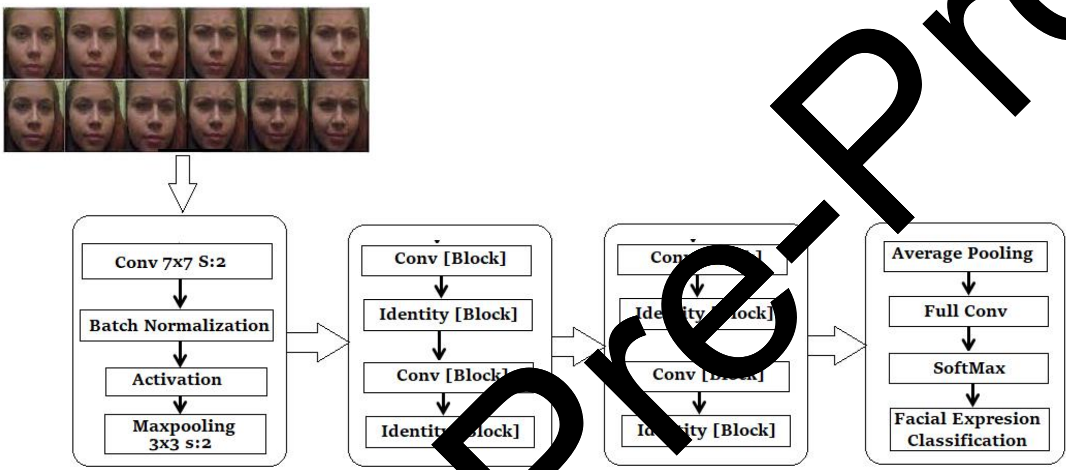


Figure 5: ResNet 50 Architecture for the Facial Emotion Classification

The 135-class of Emotional Expression dataset has been designed by Chen et al. [1] and this dataset is abbreviated as Emo135. This dataset contains 135 emotion categories and 6,96,168 facial images in total. Each emotion categories designed with from 994 to 12,794 facial images which are labelled with terms of emotions. In this dataset, we have added 258 new images of Didn't Identified emotions.

The total data is split into three files in the json format, namely Dataset for Training, Dataset for Validation and Dataset for Testing. The Dataset for Training contains 5,56,803 facial images; the Validation Dataset contains 69,560 facial images; the Testing Dataset contains 69,805 facial images. The split is corresponding to the relevant work [1]. We have used this dataset for the training phase and testing phase of facial emotion expressions. We have calculates the Emotional Factor value for the player by using the equation (3).

5.1. Statistical Analysis for Emotion Classification

The analysis of facial emotion expression is shown in table for the facial Dataset [1]. According to the emotion detection the ResNet 50 archives 91.7% of accuracy. We have made this training and testing process for the emotions of Sadness, Happy, Angry, Normal and Didn't Identified (DNI). The table 1 and figure 5 illustrated for these five types of emotions with different number of execution various from 5, 10, 15, 20, 25, 30 and 35.

Number of Time Executed	Sadness	Happy	Angry	Normal	DNI
5	85	88	81	84	91
10	88	86	84	89	93
15	92	90	88	91	89
20	94	93	92	94	87
25	95	95	95	95	96
30	96	94	96	95	96
35	95	96	94	96	97
Average	92.14	91.71	90	92	92.714

Table 1: Testing Accuracy for the Facial Emotion Detection

The following table 2 provides the total accuracy for all the emotion classification using ResNet50 by including other related CNN models of ResNet 34, Deep CNN, and AlexNet.

Number of Time Executed	Overall Accuracy			
	ResNet 50	ResNet 34	DCNN	AlexNet
5	85.8	78	[illegible]	83
10	88	82	85	84
15	90	87	[illegible]	88.3
20	92	[illegible]	88.3	90
25	95.2	[illegible]	90.3	91.7
30	95.4	91.3	92	93.2
35	95.6	93.2	94.5	94.6
Average	91.71	[illegible]	88.7	89.3

Table 2: Overall Testing Accuracy for Facial Emotion Detection

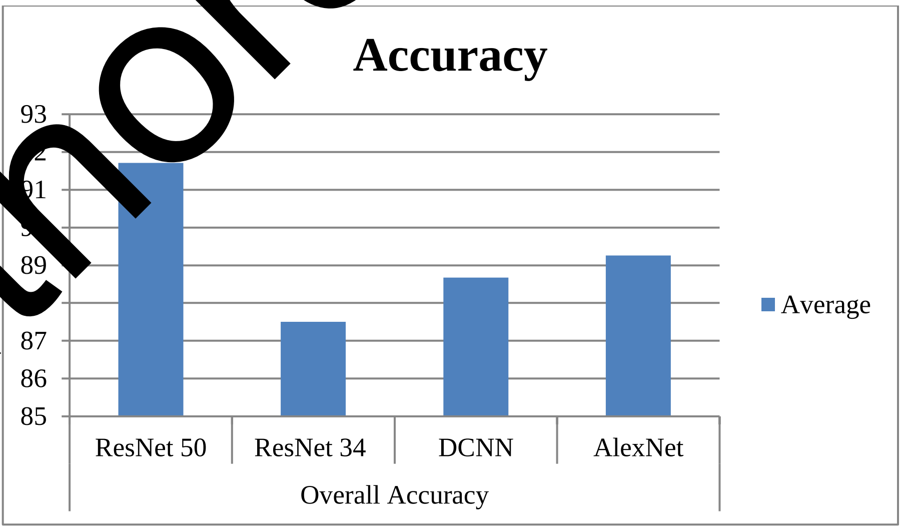


Figure 6: Overall Accuracy for Emotion Detection

Here, compare different artificial emotional based parameters which anticipate more in the performance of game experience. Considering major four parametrs like Facial Recognition, Voice Tone, Behavioral Cues and Physiological Datas. Each parameter can contribute into the game experience.

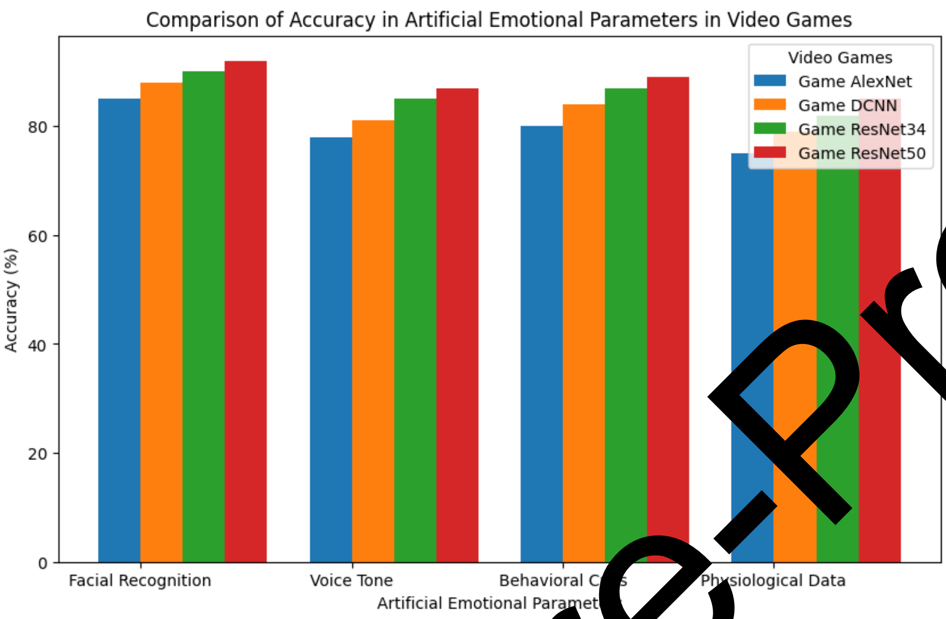


Figure 7: Emotional Parameter acuuracy

Another important parameter is state-space complexity cost. state-space complexity cost can affect the overall performance of the algorithm. If number of iterations are getting increased, then overall state-space complexity cost also increasing. Here ResNet50 algorithm works in efficient manner and giving average state-space complexity cost is lesser than other methods.

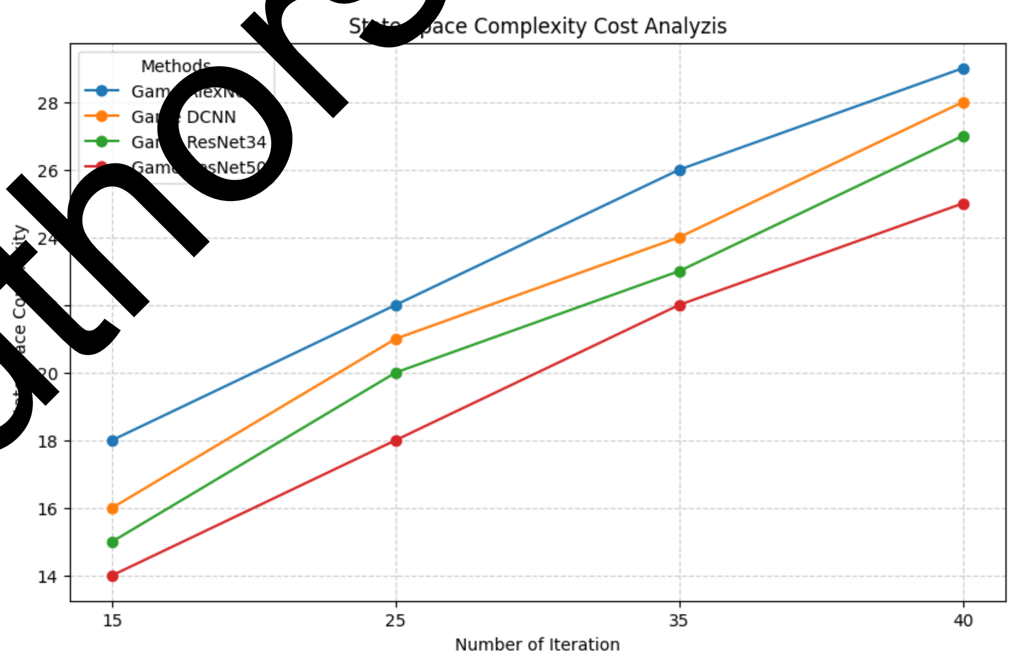


Figure 8: State-space Complexity Cost

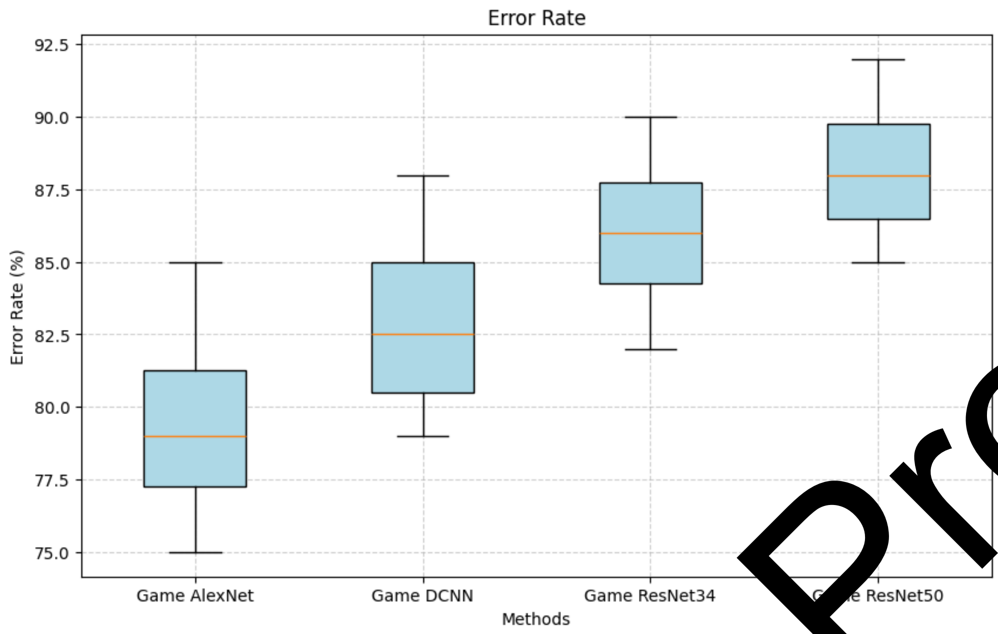


Figure 9: Error Rate

5.2. Performance Analysis

In the result and discussion section, we have discussed about the emotion prediction from the facial expressions and this has been developed based on ResNet50 deep learning model. According to the discussion about the Behavior Tree construction, we have used BT Editor and these editors are GUI enabled free open source software. We have not made any performance evaluation for the BT construction. We have made an comparison based on accuracy for emotion detection from the facial expression using ResNet 50. The table 1 and 2 provides an average accuracy rate for the individual emotion and overall average accuracy for the different number of execution results. According to the accuracy rate for the five types of emotion is around 92% and this will be a key factor for identifying the emotion about the player. The Table 2 illustrates the accuracy for the Deep learning based Convolutional Neural Network Models. All the models are working well for detecting the emotions through facial expression with respect to the number of time training the models.

6. Conclusion and Future Work

In this paper, we have presented an artificial intelligent based behaviour tree model by using an efficient emotion detection or classification system using a deep learning model ResNet 50. The proposed technique classifies the emotion of a player based five different category and the player current state of mind will be calculated based on this emotion score. The Behavior tree has been constructed from the foundation based on the hardness value calculated for each sub-BT. The performance evaluation for the emotion classification archives close to 92% and this accuracy will lead to construct an efficient BT for any game. We have evaluated the emotion detection system with other related Deep learning models.

We have not discussed about situation of non-emotional classification model and this are need to be address in the future research articles. We have not focused on the real video stream capturing for classifying the facial emotions and this may be an open area for the researchers

Reference

[1] K. Chen, X. Yang, C. Fan, W. Zhang and Y. Ding, "Semantic-Rich Facial Emotional Expression Recognition," in IEEE Transactions on Affective Computing, vol. 13, no. 4, pp. 1906-1916, 1 Oct.-Dec. 2022, doi: 10.1109/TAFFC.2022.3201290

[2] Iske B., Ruckert U., “A methodology for behaviour design of autonomous systems,” Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180), 1, IEEE (2001), pp. 539-544

[3] Ekman, P., and Friesen, W. V., Facial Action Coding System, Consulting Psychologists Press, Palo Alto, CA (1978)

[4] Llopis, N. (2010). Game architecture. In S. Rabin (Ed.), Introduction to Game Development (pp. 235–270). Boston: Charles River Media

[5] Weiss, M. A. (2013). Data structures and algorithm analysis in C ?? (4th ed.). Upper Saddle River: Pearson

[6] Bungie Studios. (2004). Halo 2, video game, Xbox, Microsoft

[7] Isla, D. (2005). Handling complexity in the Halo 2 AI. Retrieved January 15, 2014, from http://www. gamasutra.com/view/feature/130663/ gdc_2005_proceeding_ handling_.php.

[8] Isla, D. (2008). Building a better battle: The Halo 3 AI objectives system. Retrieved February 8, 2017, from https://web.cs.wpi.edu/*rich/courses/imgd4000-d09/lectures/halo3.pdf

[9] Khoo, A. (2006). An Introduction to behavior-based systems for games. In S. Rabin (Ed.), AI Game programming wisdom (Vol. 3, pp. 351–364). Boston: Charles River Media

[10] Ebrahim Babaei, Namrata Srivastava, Joshua Newn, Qiushi Zhou, Tilman Dingler, and Eduardo Velloso. “Faces of Focus: A Study on the Facial Cues of Attentional States,” Association for Computing Machinery, New York, NY, USA, pp. 1–13, 2020, https://doi.org/10.1145/3313831.3376566

[11] Sasanko Sekhar Gantayat and Swathi Lenka, “Study of Algorithms and Methods on Emotion Detection from Facial Expressions: A Review from Past Research,” In Communication Software and Networks, Suresh Chandra Satapathy, Vikrant Bhateja, M. Ramakrishna Murty, Nguyen Gia Nhu, and Jayasri Kotti(Eds.). Springer Singapore, Singapore, 231–244, 2021

[12] Puttaswamy, B. S., and N. Thillaiarasu. "Fine DenseNet based human personality recognition using english hand writing of non-native speakers." Biomedical Signal Processing and Control 99 (2025): 106910.

[13] Zhanna Sarsenbayeva, Gabriele Marini, Niels van Berkel, Chu Luo, Weiwei Jiang, Kangning Yang, Greg Wadley, Tilman Dingler, Vassilis Kostakos, and Jorge Goncalves. 2020. Does Smartphone Use Drive Our Emotions or Vice Versa? A Causal Analysis. Association for Computing Machinery, New York, NY, USA, 1–15. https://doi.org/10.1145/3313831.3376163

[14] Kangning Yang, Chaofan Wang, Zhanna Sarsenbayeva, Benjamin Tag, Tilman Dingler, Greg Wadley, and Jorge Goncalves. 2021. Benchmarking commercial emotion detection systems using realistic distortions of facial image datasets. The Visual Computer (2021), 1447–1466. https://doi.org/10.1007/s00371-020-01881-x

[15] Ashwin Shenoy, M., and N. Thillaiarasu. "Enhancing temple surveillance through human activity recognition: A novel dataset and YOLOv4-ConvLSTM approach." Journal of Intelligent & Fuzzy Systems Preprint (2023): 1-16.

[16] Krumhuber, E. G., Ku¨ster, D., Namba, S., and Skora, L., ‘‘Human and machine validation of 14 databases of dynamic facial expressions,’’ Behav. Res. Methods, 53(2): 686–701 (2021) doi: 10.3758/s13428-020-01443-y

[17] Mosiman, J., & Watson, S. (2014, February 10). Behavior Designer—behavior trees for everyone | Unity Community. Retrieved June 17, 2016, from Unity3D Forums: http://forum.unity3d.com/threads/behavior-designer-behavior-trees-for-everyone.227497/.

[18] Johansen, E. (2016). The Behave project. Retrieved June 15, 2016, from AngryAnt: http://angryant.com/ behave/

[19] Pereira, R. (2015, June 24). GitHub—behavior3/behavior3editor. Retrieved June 14, 2016, from GitHub https://github.com/behavior3/behavior3editor

[20] Brainiac Designer. (2009, November 23). Retrieved June 11, 2016, from CodePlex: https://brainiac. codeplex.com/