Balancing Multiplayer Games across Player Skill Levels using Deep Reinforcement Learning

Conor Stephens and Chris Exton
University of Limerick, Ireland

Keywords: Artificial Intelligence, Reinforcement Learning, Games Design, Deep Learning.

Abstract: The balance or perceived fairness of Level & Character design within multiplayer games depends on the skill level of the players within the game, skills or abilities that have high contributions but require low skill, feel unfair for less skill players and can become the dominant strategy and playstyle if left unchecked. Player skill influences the viable tactics for different map designs, with some strategies only possible for the best players. Level designers hope to create various maps within the game world that are suited to different strategies, giving players interesting choices when deciding what to do next. This paper proposes using deep learning to measure the connection between player skills and balanced level design. This tool can be added to Unity game engine allowing designers to see the impact of their changes on the level's design on win-rate probability for different skilled teams. The tool is comprised of a neural network which takes as input the level layout as a stacked 2D one hot encoded array alongside the player parameters, skill rating chosen characters; the neural network output is the win rate probability between 0-1 for team 1. Data for this neural network is generated using learning agents that are learning the game using self-play (Silver et al., 2017) and the level data that is used for training the neural network is generated using procedural content generation (PCG) techniques.

1 INTRODUCTION

Game balance in multiplayer games can be measured using player win rate or win probability. If a game's level design has a 50/50 win rate for both attacking and defending teams, the level can be said to be balanced. An example of this within a high profile game is Overwatch when game director Jeff Kaplan showed map win rates for attacking and defending teams as shown in the Appendix, Figure 4. Throughout the last decade, artificial intelligence has disrupted most industries, game design has also seen an influx of research interests, the most relevant examples are in using artificial intelligence to train learning agents to create and test games design for both QA (Quality assurance) and Design purposes (Gisslén et al., 2021).

Reinforcement learning agents have been used to collect data that can infer the probability of win-rate of players or teams in various environments. Previously these agents were trained before data collection, this research highlights how collecting data during the learning process brings two key benefits:

- Accelerated Data Collection when compared to using learning agents or traditional behaviour trees. Training for deep learning agents can require considerable time to emulate high performing players, the learning agents in this paper took 5 days to reach 50 million steps for a single policy Figure 1. Similarly, the engineering time taken for developing custom AI (Artificial Intelligence) behaviour is significant.
- Inclusion of Player Skill as an input parameter to the game balance tool which allows finer inference and understanding of the meta-game when using the tool in editor. Player skill level is an important metric to consider when designing levels and future characters.

Previous research by Daniel Karavolos showed how level design and character statistics can be used to evaluate the outcome of multiplayer games within procedural generated levels (Karavolos et al., 2019).

This paper extends previous research that focused on assessing level design in a 2 player game (Stephens and Exton, 2020) key additions to this research is using teams of 2 to create a more valid learning environment and generating data for training the model which includes each agent's skill rating to use during the training of the model and output of the tool, this allows a neural network to predict the outcome of a game given the skill rating of the players playing the game. The learning agents were built and trained within *Top Down Game Balance Project* https://github.com/Taikatou/top-down-shooter open source Unity project that showcases this

Stephens, C. and Exton, C.
Balancing Multiplayer Games across Player Skill Levels using Deep Reinforcement Learning.
DOI: 10.5220/0010914200003116
In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence (ICAART 2022) - Volume 3*, pages 827-833
ISBN: 978-989-758-547-0; ISSN: 2184-433X

technique where the agents are trained in an adversarial environment which provides an additional dataset corresponding to the current policy's skills in the game which can be used to fine tune our predictive networks to determine fairness for different skilled players.

Research companies such as *modl.ai* have sprung up to offer tools to accelerate the games development process using a combination of simulation and artificial intelligence, allowing them to offer services to games companies which include; "Glitch Finder" and "Player bots" which uses learning agents to both explore the game and to use player data to create smarter ai for companions and enemies within the game.

1.1 Skill Curves - Ceilings & Floors

Interesting and viable choices are key to creating an engaging multiplayer experience, to ensure players have a broad range of choices even if they are facing stronger opponents, some gameplay options are designed to have a higher impact on the required skill ratio allowing weaker players to assist their friends and change the outcome of any game. To capture, measure, and document these design considerations, an analytical tool such as skill curves and spreadsheets can be used to portray the difference to players and the potential limitations of each game mechanic. Most of these tools are an artistic portrayal of a character effectiveness given the players skill with that character. One popular example to compare is *Genji* in Overwatch who has a broad variety of mechanics and skills that are necessary for the player to use him to the best of their ability. These skills include:

Hit-scan accuracy for Genji's deflect [1], game play knowledge such as cooldown awareness allowing players to time Genji's Dash reset and ability combinations with other players on the team, projectile-based accuracy for his Shurikens and communication skills which allows the player to combine Genji's ultimate ability with other ultimate abilities on the players team. In comparison, characters such as *Lucio* and *Mercy* can provide value to the team simply by existing Mercy, requiring little to no technical skill, with the differentiation between different players being how well they know what is going to happen in the game which is known as "Game-Sense" (Skyline, 2017).

[1]Hit scan accuracy relies on instantaneous rays which intersect with the games geometry to determine what the player hit (Wikipedia, 2021)

1.2 Matchmaking

A key pillar supporting this research is the importance and data connected to skill rating in games, online multiplayer games have been using skill rating metrics to match players together for a considerable time (Herbrich et al., 2007a),

This is due to the game's designers wanting players to play against similarly skilled opponents, this gives two key use cases for this research, the first is as a design tool, the second option is to use the predictions made from the tool during the matchmaking process allowing the game to balance itself even if players have wide variances in their level of skill.

Skill is a difficult attribute in games to quantify, as there are different aspects to skill in different situations and games. To make it easier, skill rating systems have traditionally quantified skill by the probability that one player/team will defeat another. The first popularised example of a skill rating algorithm is the Elo rating system in chess. The Elo rating system uses a Gaussian distribution to quantify the player's skill, the mean of the distribution represents the player's skill, and the standard deviation is a constant defined by the game representing how big the increments of a player's skill rating can be. For example, the probability of player one beating player two is the aggregate of both of their Gaussian Distributions.

Skill rating in games is an important component of large-scale competitions and leader boards in competitive games. The most well-known type of Skill Rating in games is called the Elo rating system (Elo, 1978). Elo has shaped the terms we use to talk not just about how the rank is calculated after each game but what we call distinct spaces within the skill rating spectrum of any zero-sum game. Elo is used for chess and other single-player competitive games, however, it has experience issues when used directly in multiplayer games (Rank has to be recalculated if the team changed). This brought about the True Skill 1 and 2 (Herbrich et al., 2007b) (Minka et al., 2018). These skill rating algorithms allow developers and publishers to create fairer and more enjoyable games for their players by understanding the probability that any Player A will beat Player B with high statistical accuracy.

1.3 Player Skill

Most games try to give all players the same chance to win or succeed, this is known as *Player Balance*. Player Balance does not usually consider the skill of players and usually only applies to the beginning of games (Nystrom, 2014). Imbalance has a variety of

sources within multiplayer games, these include the cool-down times, the size of the collision boxes, play styles that don't have sufficient strategic counters, and class imbalances within the character classes.

Mechanics are designed in games to have diminishing returns in comparison to a players skill level, this brings about a variety of benefits, firstly less skilled players can achieve great impact with friends or other players by using a character with a high Skill Floor. This allows them to enjoy the mechanics of the game without feeling overwhelmed by a better opponent, The most notable example is the "Noob Tube" in *Call of Duty: Modern Warfare 2* (Credits, 2012). The Noob Tube is a mechanic with a rather high output in terms of impact within a multiplayer game compared to the skill level required to use it. Alternatively players that play with higher technical abilities would make different playing decisions, which should result in them playing Characters or strategies with greater skill requirements but with higher risks. A good rule of thumb for game's designers is that the relationship between players' skill & a players impact in game should have diminishing returns as players improve at the game.

2 RELATED WORKS

Recent work in this area and the most relevant towards this research was conducted by (Liapis et al., 2019) which explored deep learning techniques to evaluate procedural generated content and this work is spread over 4 key papers. This work leverages deep learning to predict the outcome of games from generated datasets. The first paper titled *Using a Surrogate Model of Gameplay for Automated Level Design* refers to this neural network as a "Surrogate Model of Gameplay" (Karavolos et al., 2018), this terminology is carried out throughout the 4 papers. (Liapis et al., 2019) creates a dataset for solving several supervised learning problems by simulating games within a First-Person-Shooter by having agents play against each other by using behaviour trees. Behaviour trees define agent behaviour based on the state of the environment. The impact of this design choice is that the agent's policy is consistent throughout the data collection phase of this research, secondly, the behaviour of the simulated players is biased towards how the AI would play the game and not how the players would play the game. Players see what is on the screen, have limitations with their controls, and play to win rather than following a rigid behaviour.

3 RESEARCH QUESTIONS

The main focus of this paper is to assess the effect of player skills on a game's level design. The aims of this research are as follows:

- How can reinforcement learning generate gameplay data for multiplayer games for various skilled players?
- Can we infer the win rate of different game levels for different skilled players using supervised learning?

Asymmetric gameplay is a key consideration when designing multiplayer levels to ensure both teams have equal opportunities to complete their objectives, *Capture the Flag* Vs. *Attack Vs. Defence* are examples of symmetric and asymetric gameplay options. This paper aims to answer the research questions for symmetric gameplay in a death match style game within an asymetric map, this decision limits the affect of level design to the geometry of the level (e.g. allowing agents to cover from shots) and the positioning of the players spawn location and items.

4 METHODS

This research uses a variety of techniques to generate game-play data used to train a neural network capable of predicting the "Fairness" of a level given the level's design and the skill of the players comprising the team. These techniques include using reinforcement learning agents and Procedural Content Generating (PCG) techniques to generate both levels and gameplay data simulating both Level designers and players, allowing this tool to be used both in the early stages of game development and can prevent miss used content creation and play testing on unbalanced content, avoiding costly and time consuming processes.

4.1 Reinforcement Learning Agents

To simulate players playing this top-down game, we used deep reinforcement learning agents built using Unity's excellent ml-agents framework (Juliani et al., 2018). Agents are trained with a variety of sensors to allow them to understand the world around them.

This work collects game data during the training process. In other examples of this technique pre-trained agents were used to ensure the skill of the agent's policy is consistent during the data collection process and to ensure the results collected from the following simulations are consistent, this research

takes an alternative approach by having the learning agents train during the collection process as it allowed us to have the skill rating of the agent's policy as an input to the neural network.

4.1.1 Action Space

Each agent takes an action every 3 frames and can decide a movement action and a gun action the movement actions are as follows: *None, Left, Right, Up, Down*. The gun actions are: *None, Rotate Left, Rotate Right, Shoot*.

4.1.2 Sensors

To allow the agents understanding of the world, they are equipped with sensors that capture game data and feed it into the neural network representing the agents policy. The policy's network is comprised of two hidden layers each with 512 neurons. The output of the neural network are the controls for the different playable characters. To structure the learning after each game, the winning team gets a group reward of 1 after each loss a reward of -1 and 0 for a draw.

Grid Sensor. A custom sensor that shows a representation of the world as a stacked 2D matrix of the game worlds layout, the learning agent uses a one hot encoding for each game object including:

- Team Bullets
- Enemy Bullets
- Walls
- Grass
- Team Characters
- Enemy Character

Ray Cast Sensor. A Ray Cast Sensors shoots invisible rays out from the agent's position, each ray allows the agent to detect what is at the end of the ray and how far away they are from it. Each agent has a ray cast sensor that shoots out a ray every 7.5 degrees in a circle, providing the agent a broad knowledge and finer grain understanding of the world when compared to the grid sensor, however this sensor cannot see through the walls and does not have the necessary observations to accurately substitute for the information provided to a human player.

Game Play Sensor. The last sensor is the game play sensor which captures any data that would be presented within the UI of the game, including their health, gun rotation if their gun can shoot, and the time left in the game.

4.1.3 Training

Each agent is trained using MA-POCA (MultiAgent POsthumous Credit Assignment) (Cohen et al., 2021) allowing multiple agents to play collaboratively as a team, similarly to games such as *League of Legends* and *DOTA 2*. MA-POCA solves some interesting problems found within competitive video games, previously early termination of an agent within a learning environment could lead an agent that contributed a lot to the teams success a reward of 0 which is refered to as the *Posthumous Credit Assignment* Problem. MA-Poca prevents the sample complexity that is caused by using sampling states (a previous solution to this problem) and uses attention instead of a fully connected layer with absorbing states.

Creating a balanced dataset for all skill ratings is key to successful training, self-play (Silver et al., 2017) allows a consistent iterative improvement of the agent's policy, and provides the necessary Elo values for the dataset. Self play integrates into the learning environment by having the current policy play against older policies. The learning environment with self-play allows the current policy to improve iteratively with a positive reward signal showing improvement after each iteration of the agent's policy. This prevents the policy from not having any rewards or learning due to playing against the same strength of the policy. Self-play creates an auto-curriculum effect allowing the learning environment to become more difficult over time. Figure 1 shows the training process for the learning agents, showing the stable increase of the agent's policy skill rating over the training process.

4.2 Level Generation

Each level is generated using procedural generated techniques, the reason for this is to allow us to make content to train the neural network that is the basis of our game balance tool without lengthy development time. These techniques have been used before, but it uses a combination of the drunken walk algorithm with distance spawning for items such as spawn points and health packs. Drunken walk is the process of selecting walkable terrain by randomly moving throughout the world, each level has between 6-8 walkers that have a 5% chance of dying after every movement of the procedural technique. When the ground is created, the world is wrapped by walls to ensure the players have a limited space and give the world cover and sight lines required for high skilled play with projectile weapons. Each level has 4 spawn points for both teams, the two on the leftmost side of

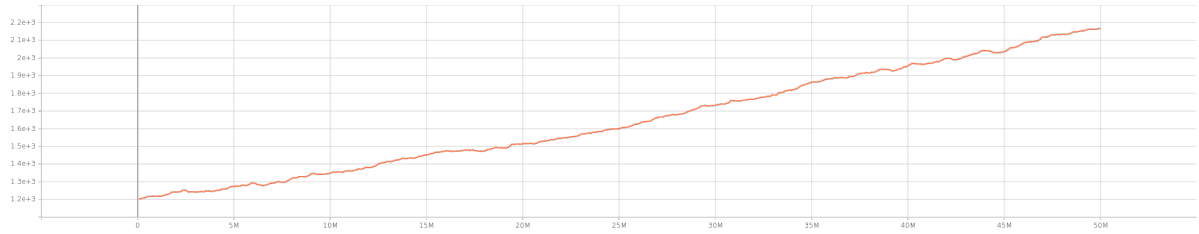

Figure 1: Agent Training Process.

the game's map are for Team 1, the two on the right are for Team 2. Up to 3 health packs are placed within the level they should be placed on the floor and at least 4 tiles away from a spawn point. Examples of these levels are shown in Figure 2

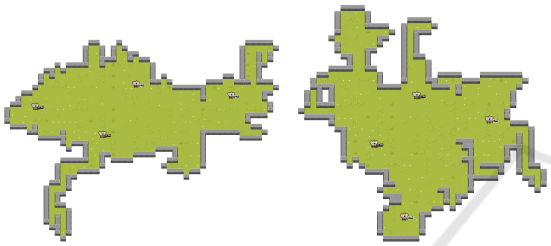

Figure 2: Procedurally Generated Levels.

4.3 Data Collection

During the learning process, we output the data after each completed game, this data is identified by a unique id for each game including the following data points which are saved as a CSV (Comma Separated Value) file:

- Winning Team (1/2, 0 if draw)
- Time Taken to Win
- Agent Elo rating[2]
- Level ID

Levels were saved as a one-hot-encoded CSV file with the Level ID in the file name, this data is used as a 2s input to our neural network, allowing us to use convolutional neural networks to achieve better performance during both training and inference. Each level design is used 100 times to create a win rate for inference. This calculation was done over 10,000 different levels: 10,000 simulations is a factor of 10 over our previous work, this value gave a broader variety of skill ratings for this data, allowing the agent to differentiate between different levels of skilled agents within each level.

[2]Capturing the Elo skill rating was possible using a back channel to move the data from the python trainer to the Unity instance

4.4 Neural Network Design

The tool to measure the game's balance was built using Python Keras and the model is imported into Unity and uses Barracuda for inference. The neural network is given the Agent's skill-rating as a normalized integer representing the values from 1200-2100, a 2D one-hot encoded matrix to represent the game world, and a one-hot encoding of the players character. The output of the neural network is the probability of Team 1 winning the game given the input parameters of the world.

The aim of the neural network is to generate a graph that outputs the probability of player 1 winning, given the skill rating, to achieve, there were two different methods the first would be for the model to output multiple data points for a single map, this was seen as having a harder training process however the model would require less computation during inference. This approach would be more suitable within a production ready version of the tool, the implementation within this research is to have the skill as an input and using the model multiple times with different inputs to generate 10 separate probabilities that comprise the required curve.

4.4.1 Network Layout

The network structure is designed for multiple inputs or mixed data. In machine learning, mixed data refers to the concept of using multiple types of independent data. In our context, this is the two continuous values for both teams playing the game when playing the game and the image data for the level design. Figure 7 shows the structure of the model; *input_17: InputLayer* is the input of our mixed data into an otherwise conventional convolutional neural network. This output of this neural network is a single neuron with a sigmoid activation function (Narayan, 1997) with an output of 0-1 to create the logistic regression needed for predicting the "Fairness" of the level.

5 RESULTS

Agent training and data generation is slow, but the steady learning shown in Figure 1 rate and the high entropy in Figure 6 created a broad and valid dataset for this problem domain. This consistent improvement of the learning agent's skill rating avoids class imbalance during both training and testing of the model.

95% of the sample maps from the testing data split model achieves the 10%+- win rate probability that is currently considered acceptable in multiplayer games for matchmaking. This is acceptable, however, the variance of the model is too high for production use.

As shown in Figure 5 the graph generation in Unity is relatively straight forward with the graph updating in almost real time when an artist changes the level's tile-map. The UI shown in Figure 3 showcases how the tool looks when the graph is clicked it shows Figure 5. Each graph is made up of 10 data points, each 0.1 increment on the graphs *X-axis* is the equivalent of 40 Elo skill points. The Y-axis is the probability of Team 1 winning the game which is in the range of 0-1.

6 DISCUSSION

Automated tools are becoming an increasingly common place in games. Companies have moved from automated build systems that can create daily builds using Jenkins or Team City to procedural art tools for creating large open world games such as Houdini and more recently to automated QA testing using learning agents, as games expand in scope the development and testing process for them becomes more arduous, *Red Dead Redemption 2* is an excellent example of this. Other areas of expansion for the games industry are wider worlds and procedural art tools, a common term is a 4k world which stands for 4 kilometres by 4 kilometres. Automation allows easier creation and iteration of content and is a key focus for game companies to prevent burn out that has been a massive issue within the industry.

There is a broad range of future applications for this tool kit, ranging from designing content to testing exploits within the game's mechanics. One suitable use case within Player Vs. Enemy (PvE) games is to evaluate how powerful different combinations are either made by hand or using (PCG) techniques for players of different skill levels. Another use case is for testing new gameplay elements and rulesets, this tool can evaluate key gameplay metrics such as the session length and key weapon statistics such as average damage and max damage.

The authors think due to the broad applicability of game design tools built using supervised learning should hopefully see unique and new usecases within the games development process. One key consideration is the difficulty of integrating similar tools into game engines other than Unity. Unreal Engine and other proprietary game engines such as Lumberyard don't have easy to use inference tools at this current time preventing this approach from being used in a wide variety of games especially when we consider Unity's poor multiplayer support. While Unity is moving towards a more scalable consistent multiplayer architecture, Unreal Engine is moving towards integration of more AI within the game engine with projects such as *InteractML* and *Airsim* getting key support from Epic Games (Developer of Unreal Engine).

ACKNOWLEDGEMENTS

We would like to thank Lero: The Irish Software Research Centre for their continued help and support.

REFERENCES

Cohen, A., Teng, E., Berges, V., Dong, R., Henry, H., Mattar, M., Zook, A., and Ganguly, S. (2021). On the use and misuse of absorbing states in multi-agent reinforcement learning. *CoRR*, abs/2111.05992.

Credits, E. (2012). Balancing for skill - the link from optimal power to strategy. *https://www.youtube.com/watch?v=EitZRLt2G3w*. In comment section, accessed on 2021/11/06.

Elo, A. E. (1978). *The Rating of Chessplayers, Past and Present*. Arco Pub., New York.

Gisslén, L., Eakins, A., Gordillo, C., Bergdahl, J., and Tollmar, K. (2021). Adversarial reinforcement learning for procedural content generation. *CoRR*, abs/2103.04847.

Herbrich, R., Minka, T., and Graepel, T. (2007a). Trueskill^{tm}: A bayesian skill rating system. In Schölkopf, B., Platt, J. C., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19 (NIPS-06)*, pages 569–576. MIT Press.

Herbrich, R., Minka, T., and Graepel, T. (2007b). Trueskill(tm): A bayesian skill rating system. In *Advances in Neural Information Processing Systems 20*, pages 569–576. MIT Press.

Juliani, A., Berges, V., Vckay, E., Gao, Y., Henry, H., Mattar, M., and Lange, D. (2018). Unity: A general platform for intelligent agents. *CoRR*, abs/1809.02627.

Karavolos, D., Liapis, A., and Yannakakis, G. N. (2018). Using a surrogate model of gameplay for automated level design. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8.

Karavolos, D., Liapis, A., and Yannakakis, G. N. (2019). A multi-faceted surrogate model for search-based procedural content generation. *IEEE Transactions on Games*. accepted.

Liapis, A., Karavolos, D., Makantasis, K., Sfikas, K., and Yannakakis, G. (2019). Fusing level and ruleset features for multimodal learning of gameplay outcomes. pages 1–8.

Minka, T., Cleven, R., and Zaykov, Y. (2018). Trueskill 2: An improved bayesian skill rating system. Technical Report MSR-TR-2018-8, Microsoft.

Narayan, S. (1997). The generalized sigmoid activation function: Competitive supervised learning. *Information Sciences*, 99(1):69–82.

Nystrom, R. (2014). *Game Programming Patterns*. Genever Benning.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

Skyline (2017). Overwatch — the hardest hero? skill ceiling/floor discussion. *https://www.youtube.com/watch?v=AQ4BAG520LY*. In comment section, accessed on 2021/11/06.

Stephens, C. and Exton, D. C. (2020). Assessing multiplayer level design using deep learning techniques. In *Foundation of Digital Games*.

Wikipedia (2021). Hitscan — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Hitscan. [Online; accessed 28-December-2021].

APPENDIX

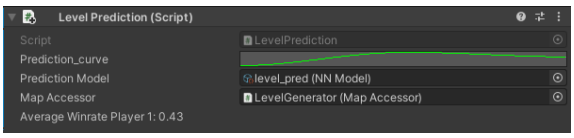

Figure 3: Unity Tool.

Figure 4: Overwatch Win Rate 2017 (Blizzard Forum).

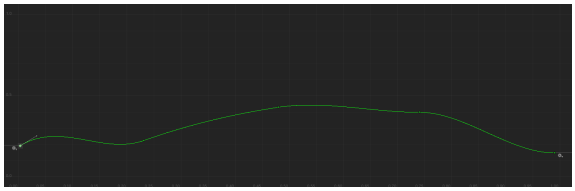

Figure 5: Win-rate Vs Player Skill Inferred.

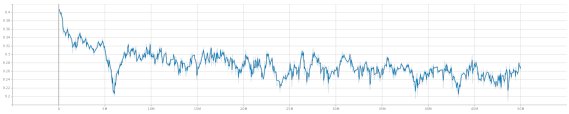

Figure 6: Policy Entropy.

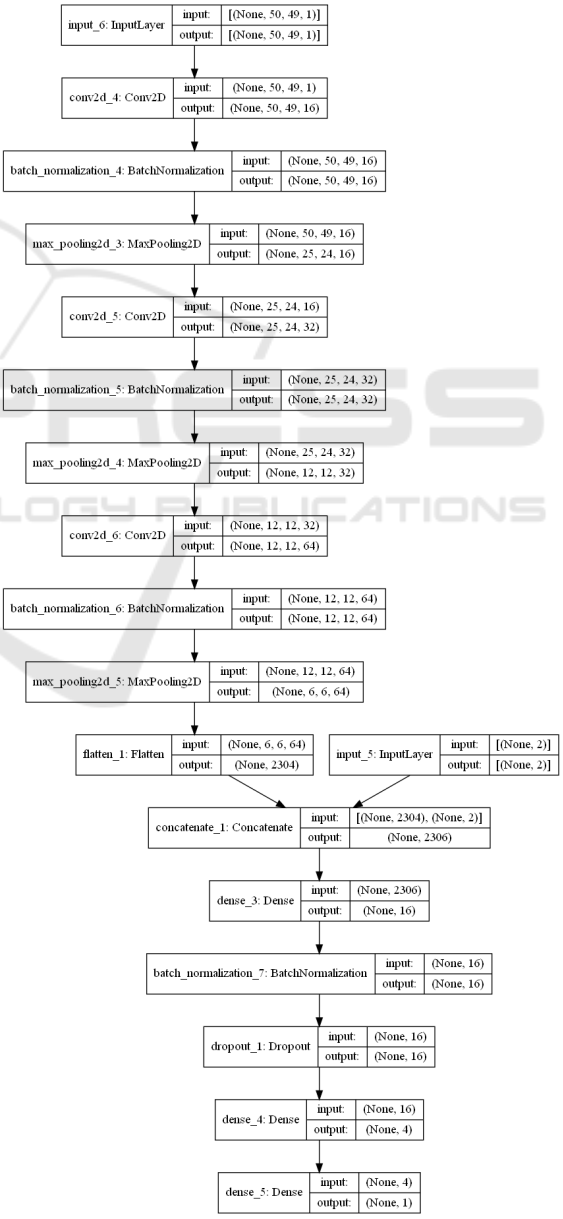

Figure 7: Level Design Neural Network Model.