Personal and Ubiquitous Computing
https://doi.org/10.1007/s00779-024-01801-z

ORIGINAL PAPER

Using a NEAT approach with curriculums for dynamic content generation in video games

Daniel Hind[1] · **Carlo Harvey**[1]

Received: 2 May 2023 / Accepted: 20 March 2024

Abstract
This paper presents a novel exploration of the use of an evolving neural network approach to generate dynamic content for video games, specifically for a tower defence game. The objective is to employ the NeuroEvolution of Augmenting Topologies (NEAT) technique to train a NEAT neural network as a wave manager to generate enemy waves that challenge the player's defences. The approach is extended to incorporate NEAT-generated curriculums for tower deployments to gradually increase the difficulty for the generated enemy waves, allowing the neural network to learn incrementally. The approach dynamically adapts to changes in the player's skill level, providing a more personalised and engaging gaming experience. The quality of the machine-generated waves is evaluated through a blind A/B test with the Games Experience Questionnaire (GEQ), and results are compared with manually designed human waves. The study finds no discernible difference in the reported player experience between AI and human-designed waves. The approach can significantly reduce the time and resources required to design game content while maintaining the quality of the player experience. The approach has the potential to be applied to a range of video game genres and within the design and development process, providing a more personalised and engaging gaming experience for players.

Keywords Dynamic content generation · Game experience · NEAT

1 Introduction

This paper aims to explore the application of artificial intelligence (AI) techniques within video games to investigate whether dynamic content can be generated similar to that of a human game designer. Additionally, we aim to evaluate the impact of AI-generated content on player engagement in comparison to manually created content. To accomplish this, a machine learning-driven AI wave manager was developed for a tower defence game using NeuroEvolution of Augmenting Topologies (NEAT) and then compared with human-designed waves to determine its viability as a content generation system.

Developing video game content is often an expensive and time-consuming process. Crafting each encounter that a player may experience and manually tuning difficulty levels for a range of skill levels can take up a large part of a game designer's work time. Even then, players may still reach the end of the finite content and crave more. Hence, the goal of this project is to investigate whether AI can ease this strain on development by dynamically generating engaging content for players to experience. By doing so, this project aims to reduce the amount of human work hours required to create content while simultaneously enhancing the value of the end product for players and game studios alike.

The use of AI-generated content has the potential to be more dynamic and engaging than finite manually designed content. This project adapts the output into a tower defence (TD) paradigm, a popular game genre that utilises similar AI enemy waves versus player structure. Olesen et al.'s [1] research on Dynamic Difficulty Adjustment (DDA) using rtNEAT within a real-time strategy game shows that games with easily observable game states and simple goals work

Daniel Hind and Carlo Harvey contributed equally to this work

✉ Carlo Harvey
carlo.harvey@bcu.ac.uk

Daniel Hind
daniel.hind@mail.bcu.ac.uk

[1] DMTLab, School of Computing and Digital Technology, Faculty of Computing, Engineering and the Built Environment, Birmingham City University, Curzon Street, Birmingham B4 7XG, West Midlands, UK

Published online: 11 April 2024

best for this approach, and a TD game fits this criterion perfectly, making it an ideal candidate for a similar ML DDA technique.

Neuroevolution, an approach to AI inspired by biological nervous systems, employs evolutionary algorithms to evolve complex artificial neural networks capable of intelligent behaviour. A notable extension of this approach, NEAT, combines topology and parameter evolution, incorporating features such as complexification, historical markings to avoid competing conventions, speciation, and fitness sharing. NEAT's performance has been enhanced over the years with advancements like HyperNEAT and CoDeepNEAT [2].

The project aims to explore whether an evolving neural network can be effectively trained and applied to a tower defence game to generate enemy waves that challenge the player's strategy with the goal of increasing player engagement. By achieving this goal, this project can have implications for the video game industry, offering a more efficient and cost-effective way to generate content that is engaging, dynamic, and endlessly re-playable for players.

The contributions of this project can be summarised as follows:

- Implementation of a wave director using a NEAT NN that can observe the current game state and make informed decisions about the composition of the next wave of enemies
- Development of a NEAT NN-based curriculum generator for tiered learning environments to enhance the ability of a wave director to learn generation of effective waves
- Evaluation of whether player engagement is increased by the addition of the wave manager through A/B testing
- Analysis of whether AI-generated content is comparable to human-designed content, to determine the effectiveness of dynamic AI content generation in minimising human designer time
- Usability evaluation of the wave director tool by professional game designers
- Discussion and suggestions of practical use cases for the project

2 Related work

The development of artificial neural networks (ANNs) has been tailored for specific tasks, such as image classification (Krizhevsky et al. [3]), computer vision tasks (Guo et al. [4]), and speech recognition tasks (Hochreiter and Schmidhuber [5]). A unique aspect of ANNs is that their topologies can be evolved alongside their weights and biases, allowing for rapid evolution. In recent years, the development of neuro-evolution and machine learning has also led to the incorporation of ANNs into video games, such as dynamic difficulty adjustment and automated game design. For instance, the cgNEAT [6] was developed for automated content generation in real-time video games, while the rtNEAT [1] was used to adjust the difficulty of a real-time strategy game. Other genetic techniques have been applied in the design of waves for video games [7]. The most recent work in this area is a report by Risi and Togelius [8].

In NEAT, each neuron in the network has an associated activation function that determines how it processes its inputs and generates an output. The choice of activation function can have an impact on the network's learning ability and performance. While NEAT networks typically use fixed activation functions, it is possible to introduce adaptive activation functions within the NEAT framework [9]. Adaptive activation functions can modify their behaviour dynamically based on the network's inputs, outputs, or other factors. These adaptive activation functions can learn and adjust their parameters during the training process, allowing the network to adapt to different data patterns and improve its performance.

The prevailing method for training deep neural networks (DNNs) in various domains has been gradient descent (GD) and its variants. However, gradient-based optimisation has limitations, such as requiring differentiability between the neural network and the loss function. Problems that cannot be directly modeled or solved without alterations, such as formal logic and hard attention [10], necessitate alternative approaches that traverse the search space efficiently without gradients. Neuroevolution, as an evolutionary optimisation algorithm focused on DNNs, presents a promising alternative to stochastic gradient descent. While NEAT may not be directly suitable for data classification, the introduction of two methods, *divide and conquer* and *backpropagation*, addresses this issue in a new algorithm called L-NEAT (Learning NEAT). NEAT is used for searching the ideal topology, and gradient descent is subsequently employed to train the proposed networks. For a full exploration of the latest state-of-the-art in current research trends for NEAT, please see [2, 10].

In recent studies, researchers have explored the use of automatic curriculum generation to build complex behaviours for artificial agents in video games [11]. However, the manual development of curricula is time-consuming, which has led to the exploration of automatic curriculum generation. It has been shown that considering goal validity, feasibility, and coverage is vital for constructing useful curricula. These curricula have proven to be effective in sparsely rewarding environments where an agent is required to achieve a single goal from a set of goals that varies between episodes, which demonstrates the potential of automatic task curricula towards learning complex goals.

Automated game design (AGD) is a field that uses AI to generate video game content or entire games from scratch. This technology can help reduce the time and resources needed for game development while also enabling the creation of endless procedural playable content that increases the value of a game. AGD involves the use of AI to create game content or entire games. The technology has been used to combine existing game systems to create new games using machine learning [12]. It has also been used to generate two-dimensional level layouts inspired by classic games [13]. One of the recent developments in AGD is the implementation of wave function collapse techniques in procedural content generation to produce three-dimensional meshes as game content [14]. This technique has been bolstered in its utility with design-level constraints embracing knowledge of tile-based connective structures of the used imagery [15]. AGD is essential as it reduces hands-on designer time and the resources required for a given project.

Dynamic difficulty adjustment (DDA) is another key theme in the incorporation of ANNs in video games, demonstrated by Li et al. [16] and Ebrahimi and Akbarzadeh-T [17]. It is the ability of video games to change their difficulty level dynamically in real time to match the player's skill level. This technique has been used in several games and has proven to be successful. Csikszentmihalyi's [18] concept of "flow" suggests that players enjoy games most when they strike a balance between their skill and the challenge presented. The traditional approach of pre-set difficulties, such as easy, medium, and hard, can be too restrictive and not nuanced enough to account for a wide range of skill levels. In the tower defence game genre, where levels are predictable and can be solved once a viable strategy is found, this can lead to players feeling disengaged and approaches exist to improve engagement throughout the course of a game [19]. Sutoyo et al. [20] attempted to tackle this problem by creating a traditional DDA system for a wave-based tower defence game that adjusts the difficulty based on in-game metrics to ensure that players of all skill levels are always presented with a suitable challenge.

Sutoyo et. al. [20] proposed adjusting difficulty via three in-game metrics: player health, enemy health, and skill points. However, these generic metrics may not provide a comprehensive picture of a player's actual competence. For instance, player and enemy health are simply side effects of the player's defence structure, making it more valuable to evaluate the tower placement itself. In order to achieve this, a machine learning method can be used to directly evaluate tower placement and observe tower synergies, which can provide a more direct estimate of actual player strategy. This approach can facilitate more fine-tuned DDA through wave generation and management.

AGD and DDA are two exciting areas of research in video game design. AGD can help reduce the time and resources required for game development while also enabling the creation of endless procedural playable content. DDA, on the other hand, can improve the gaming experience by dynamically adjusting the game difficulty based on the player's skill level. Both technologies have the potential to enhance the quality of video games and keep players engaged for longer periods.

3 Methodology

In this study, the overarching objective is to develop an innovative wave manager driven by a NEAT network. The proposed wave manager will be trained to assess the player's defensive strategies in a tower defence game and make informed decisions on which enemies to spawn for each wave. The system will be equipped with a predetermined number of points that will be allocated based on both traditional difficulty increments and the current wave count. This method will ensure that the wave manager remains fair and generates a balanced level of incremental difficulty for each player, bespoke to them.

To ensure that the wave manager remains fair, we use a combination of conventional difficulty increments and the current wave count. These allocated points will be used to control the number of enemies that are spawned, their types, and their strengths. We believe that this methodology will allow us to generate optimal enemy waves that challenge the player while ensuring that they are still beatable.

To facilitate the visualisation of this process, Fig. 1 provides an overview of the inputs and outputs of the proposed wave manager.

3.1 Game development and project setup

In this project, the Unity game engine (version 2020.2.1f1) was utilised to develop the game. The selection of this game engine was a logical choice due to its robust set of tools, vast community support, and availability of templates. To further expedite development, SongGameDev's Tower Defense Toolkit 4 (TDTK-4) was chosen as the foundation for the game. This toolkit provides an all-in-one set of assets and tools to swiftly construct a comprehensive tower defence game.

For the implementation of the NEAT ANN, UnitySharpNeat was integrated into the project. This package provides a user-friendly interface for training and testing ANNs, with a wide range of available features tailored for Unity game development. While most of the necessary features required to start training an ANN are already provided within the package, some project-specific setup was required to ensure seamless integration.

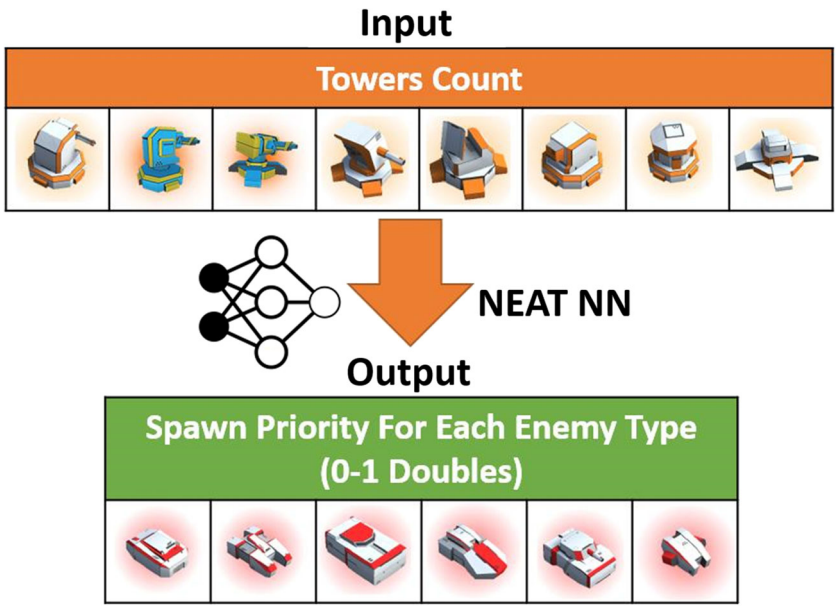

Fig. 1 Illustration of the inputs and outputs for the neural network for the tower defence game

3.1.1 Units and towers

During the iterative process of development and rigorous testing, it was discovered that the network generated waves that favoured certain strategies, which was caused by an imbalance in the features of individual units and towers in the game design. This led to an adjustment in the wave unit spawn costs and stats, which aimed to make them feel fair and provide a balanced challenge to the player.

By tweaking the cost and stats, some units were made less powerful, while others were made more challenging when used against player tower strategies. The changes made are summarised in Tables 1 and 2 respectively, with the ratios of units chosen presented in Fig. 7 for a clearer representation of the modifications made.

3.1.2 NEAT management and fitness

To facilitate the interaction between the NEAT agents and the game environment, we designed and implemented an AIWaveManager class, which not only provides the required inputs to the agents but also handles their outputs, as well as collecting and processing feedback data on the outcome of each wave to calculate the network's fitness.

The fitness function used for evaluating the NEAT agents' performance is carefully crafted to optimise the gameplay experience and ensure that the waves generated by the network are challenging yet fair. The algorithm for computing fitness, presented in Algorithm 1, takes into account several factors, including the number of enemy wave units that manage to reach the end of the tower course, the average distance

Table 1 Showing the wave unit stats that were used in game development and for training the AI

Icon	Name	SC	HP	S	Sp	Ar	Special ability
	Transport	1	17	0	1.5		
	Speeder	3	15	10	2.5	Physical	
	Support	5	50	30	1.5		50% resistance to nearby units
	Carrier	7	40	40	1	Electric	Spawns a drone when killed
	Tank	25	650	0	0.5		
	Drone	3	1	10	2		Flying: can't be slowed

SC spawn cost, *HP* health, *S* shield, *Sp* speed, *Ar* armour

Table 2 Illustrating final tower stats that were used to train the NEAT network

Icon	Name	C	Dmg	CD	Rng	Crt	DmgT	Special ability
	Machine gun	10	1–2	0.75	2	25% 1.25×	Phys	
	Zapper	15	3	1.25	2.5	10% 2×	Elec	10% to stun on hit
	Laser	20	2–3	1	3	10% 2×	Elec	Damage over time 6 damage/3 s
	Cannon	25	8-10	2	3.5	0%	Phys	Area of effect
	Slow	25			2.25			Slows enemies in range 40% slow speed
	Support	30			3			Fire rate increase 25% attack speed

C cost, *Dmg* damage, *C D* cooldown, *Rng* range, *Crt* critical modifiers (% chance and × multiplier), *DmgT* damage type (*Phys* physical, *Elec* electric)

traveled by the units, and the number of units requested to spawn versus the available spawn points.

We believe that this fitness function strikes a good balance between encouraging the network to generate waves that pose a real threat to the player while still making sure that the game remains winnable. By rewarding the network for each point of damage dealt to the player's health pool, we incentivise the wave generator to create waves that are tough enough to inflict damage by traversing to the end of the tower course without dying, but not so overwhelming as to be impossible for the towers to overcome. Similarly, by penalising the network for spawning more units than there are spawn points, we prevent the wave generator from creating waves that are inherently unfair and impossible to beat.

We expect that this approach will lead to the creation of a wave manager that can dynamically adjust the difficulty of the game in real time, providing a challenging yet enjoyable experience for players of all skill levels.

The SpawnManager class in the TDTK-4 framework only allowed for waves to be defined at the start of the game, which limited the flexibility of wave generation. In this project, the SpawnManager class was modified to allow for the generation of new waves at runtime from the AIWaveManager class. The AIWaveManager system was used to infer units to spawn and pass this list to the SpawnManager class for runtime generation. Whenever a new wave was created, the AIWaveManager would signal to the NEAT supervisor that a new wave had started, clear any fitness trackers from the previous wave, and then request each NEAT agent to generate a wave for its associated tower defence path. The project employed ten NEAT agents (as can be seen in Fig. 2), each represented by a NeatGameWaveGeneratorUnit class that handled all the inputs, outputs, and fitness for a given agent.

When the AIWaveManager requested a wave to be generated, it supplied each NEAT agent with an associated path. The agent would then query the game state for relevant inputs, activate itself, and evaluate the neural network to generate an output array representing the wave to spawn. The agent's function was to return a double array [0-1] that represented

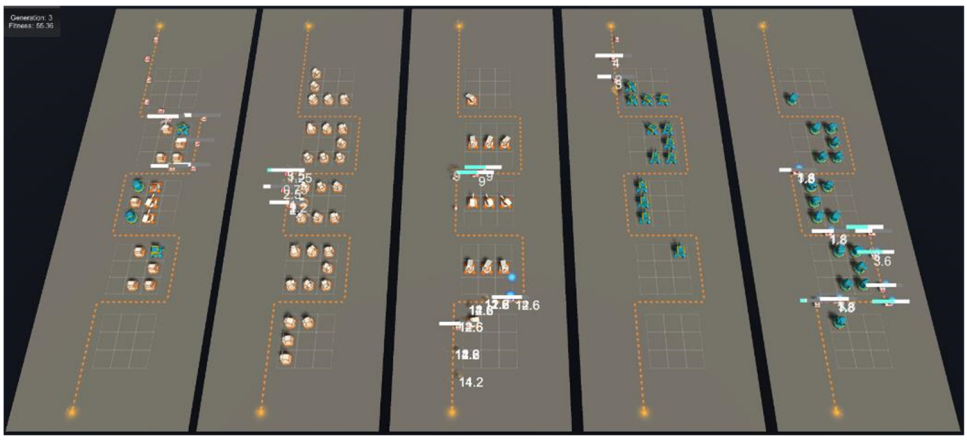

Fig. 2 Illustration of five NEAT agents, concurrent training within Unity of the NEAT network used for wave generation

Algorithm 1 GetFitness implementation details for the AIWaveManager class, relying on the method GetAveragePathCompletion.

```
Input: P, U                                  ▷ Path of defence and List of units
Output: R                                    ▷ Average Path Completion
  function GetAveragePathCompletionP,U:
      fTotComp ⟵ 0
      foreach U ∈ Units do
        fTotComp ⟵ fTotComp + P.U.PathCompletion
      end for
      iNumSpawned ⟵ U.Len()
      if iNumSpawned = 0 || fTotComp = 0 then
        R ⟵ 0
      end if
      R ⟵ fTotComp / iNumSpawned
      R ⟵ Clamp(R, 0, 1)
      return R
  end function

Input: W                                     ▷ Wave applied to path
Output: F                                    ▷ Fitness Function
  functionGetFitnessW:
      fMaxFitness ⟵ 100
      fAvgComp ⟵ GetAveragePathCompletion(P, U)
      F ⟵ fMaxFitness ∗ fAvgComp
      F ⟵ Clamp(F, 0, fMaxFitness)
      return F
  end function
```

the priority for which enemy units to spawn at the start of a given wave.

3.2 Training

The AI wave manager's input and output were designed to optimise performance in the tower defence game. The inputs for the network consisted of the number of each tower type in the level, and the outputs were the spawn priority for each enemy type. During training, each of ten lanes was initially given a unique tower layout and used ten trials so that each NEAT agent performed against each tower layout before having fitness evaluated. The tower layouts were human-designed to allow the network to learn the strengths and weaknesses of each tower type and to also test it against different combinations of towers, effectively human-designed curricula. The final fitness was determined by the average path completion percentage of all enemies spawned in that wave. The tower layouts used in training can be seen in Fig. 2.

During training on tower layouts created by humans, it was observed that NEAT could be utilised not only for generating waves to traverse the tower course but also for generating automated curricula of tiered difficulty for the waves to encounter. This approach is similar to those used by Racaniere et al. [11] and Du et al. [21]. The curricula approach effectively presents increasingly difficult environments to the wave generator during its training, with the curricula creating more challenging obstacles for the spawned waves as the wave generator's fitness improves. This approach is believed to be generally applicable to other domains, such as the development of waves of pedestrians and traffic flows for automatic vehicle training [22].

3.2.1 Wave generator training

Figure 4 presents the final neural network architecture developed by NEAT for generating waves of enemies in a tower defence game. As depicted in Fig. 1, the input to the network consists of the current number of each tower type, with upgraded towers counted twice due to their increased impact. The output is a [0-1] double for each of the 6 enemy types, representing the spawn priority of each enemy type. These priorities are then used to distribute available spawn points across enemy types, proportional to their given priority. Any remaining points are redistributed to the next highest priority enemy type until all points are spent.

After approximately 5000 generations, shown in Fig. 3, the network consistently defeated the training lanes with a fitness level between 95 and 100. To challenge the network further, the tower layouts were made significantly harder (by human design) and the network was trained for another 500 generations, resulting in relatively consistent fitness levels. The final selected champion network had a fitness of 56.3 against the harder tower layout and was found to be sufficiently challenging and dynamic in internal playtesting (Fig. 4).

3.2.2 Tower generator training

Using the same AIWaveManager, once the wave generator was trained, we investigated whether the system could be used to generate curricula of towers to encounter. Towers in this paradigm can also be considered waves, and in the development of the wave generator, were human-designed. This paradigm can be considered a flipping of Fig. 1, whereby the input is the wave to be sent down each lane and the output of the NEAT wave generator is the spawn priority for each tower type. The NEAT fitness function is not modified and is evaluated again on the progress of average path completion shown in Algorithm 1. As knowledge of the wave is required to infer a set of tower spawn priorities, in this investigation, the equivalent generation of the prior trained wave generator was used to produce the waves. Additionally, tower placement respected the first available slot on the lane, in contrast to the centric distribution for the wave generator training observable in Fig. 2. This results in the low fitness observed in early generations shown in Fig. 3. This shows applications of the technique towards automated curriculum; however, the

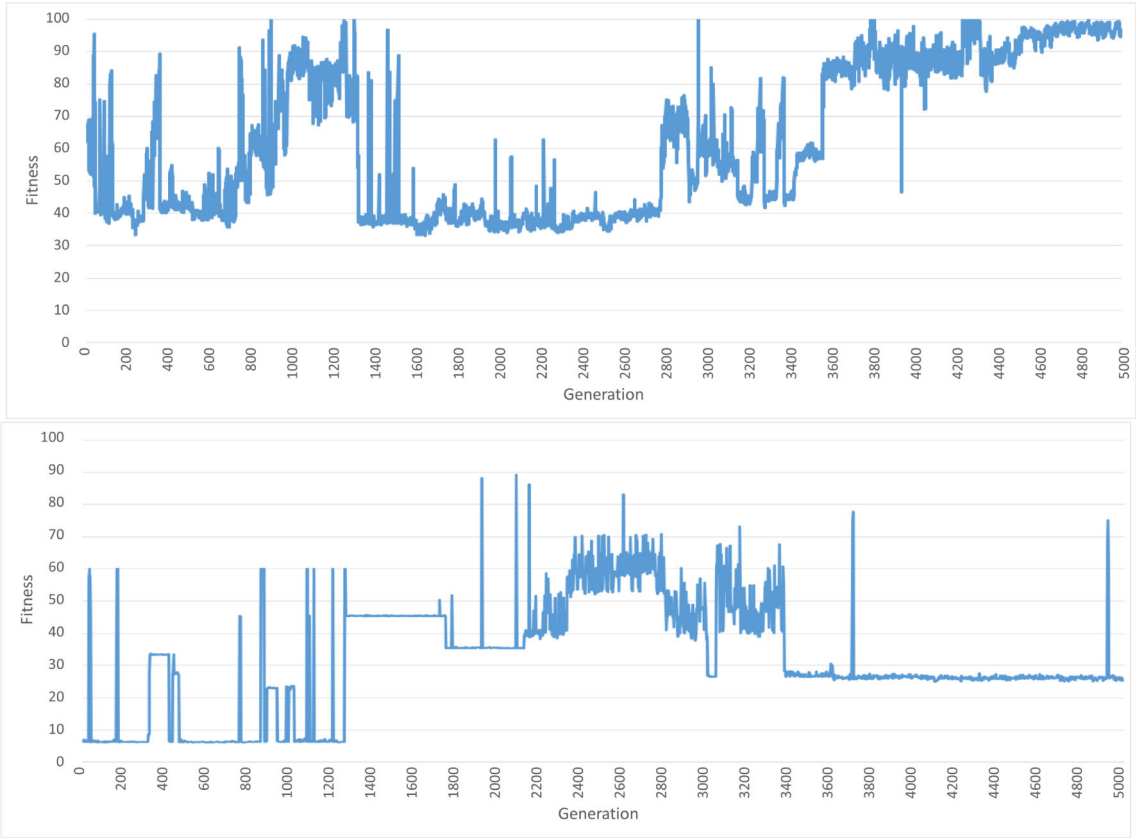

Fig. 3 Top: Fitness of the Wave Generator over 5000 generations showing convergence to a network capable of designing challenging waves. Bottom: Fitness of Tower Generator over equivalent generations to the Wave Generator

tower layout chosen by the curricula after generation 3400 is too strong to see waves progress, on average, past roughly 28% of the lane. This is largely due to the curriculum developing an optimal front-weighted tower deployment on the lane.

3.3 Evaluation

To evaluate the success of this project, we use two key metrics: similarity of content generation and player engagement.

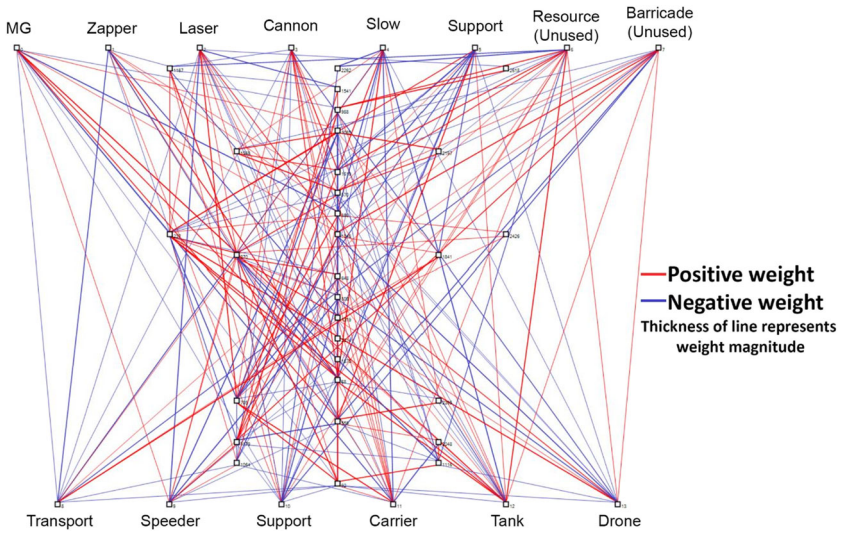

Fig. 4 Graph showing the topology for the trained NEAT network used for wave management. Inputs at the top and outputs at the bottom

Table 3 The total scores from the GEQ A/B testing survey with 11 participants for each group (AI and human waves)

	GEQ total scores						
	Co	SaII	Fl	T/A	Ch	NA	PA
AI	149	152	107	18	72	23	179
Human	120	115	86	33	86	32	167
p-value	0.19	0.06	0.21	0.19	0.41	0.34	0.38

This is complemented with a t-test to determine significance
Co competence, $SaII$ sensory and imaginative immersion, Fl flow, T/A tension/annoyance, Ch challenge, NA negative affect, PA positive affect

Similarity of content generation is assessed by comparing the NEAT-generated waves to human-generated waves. Player engagement is evaluated by comparing the engagement levels of players who played against the AI wave manager to those who played against human-designed waves. In addition, we evaluate content generation capability via expert assessment, engaging game designers to evaluate this system's usability in the context of the potential as a tool for wave design.

To evaluate player engagement, A/B testing was employed. According to Winkler [23], a minimum of ten participants are required in each test group. Half of the participants played against the AI wave manager, while the other half played against human-designed waves. The A/B testing methodology involves testing a known "control" implementation against an experimental "challenger" software. After playing a set of levels for 25 min or until they completed all three levels, participants were surveyed using the Game Experience Questionnaire (GEQ) developed by Ijsselsteijn et al. [24]. Due to the COVID-19 pandemic and associated new ethical policies, the testing was conducted remotely, and participants were given a link to download the game, a participant information leaflet, and a web consent form.

The GEQ Core Module was used to measure participants' feelings and engagement levels throughout the play session. The areas of the survey that were observed closely were those related to flow, tension/annoyance, and challenge, as these areas were expected to be most influenced by the AI wave manager.

4 Results and discussion

The survey was conducted with a total of 22 participants, with 11 participants assigned to each trial (A or B). It is important to note that all participants met the inclusion criteria of having prior experience playing at least one tower defence game, which ensured that they had some understanding of the genre's fundamental gameplay. Additionally, all participants were able to complete the first level of the game during the allocated timeframe. This criterion was important as it ensured that participants were able to experience enough of the game to provide meaningful feedback.

The final results of the survey are presented in Table 3 with question means and breakdown within Fig. 5. This table displays the total scores achieved for both the AI and human-generated waves across all seven aspects of the Game Experience Questionnaire (GEQ). The survey results show that the AI performed better than the human waves in positive aspects such as competence, sensory and imaginative immersion, flow, and positive affect. Additionally, the AI waves scored lower in the negative aspects of tension/annoyance and negative affect.

Surprisingly, the AI scored slightly lower than the human-designed waves in the challenge aspect, indicating that the AI wave manager may not be as challenging as it was intended to be. This result is unexpected since the entire concept behind the AI wave manager's implementation was to train it to adapt

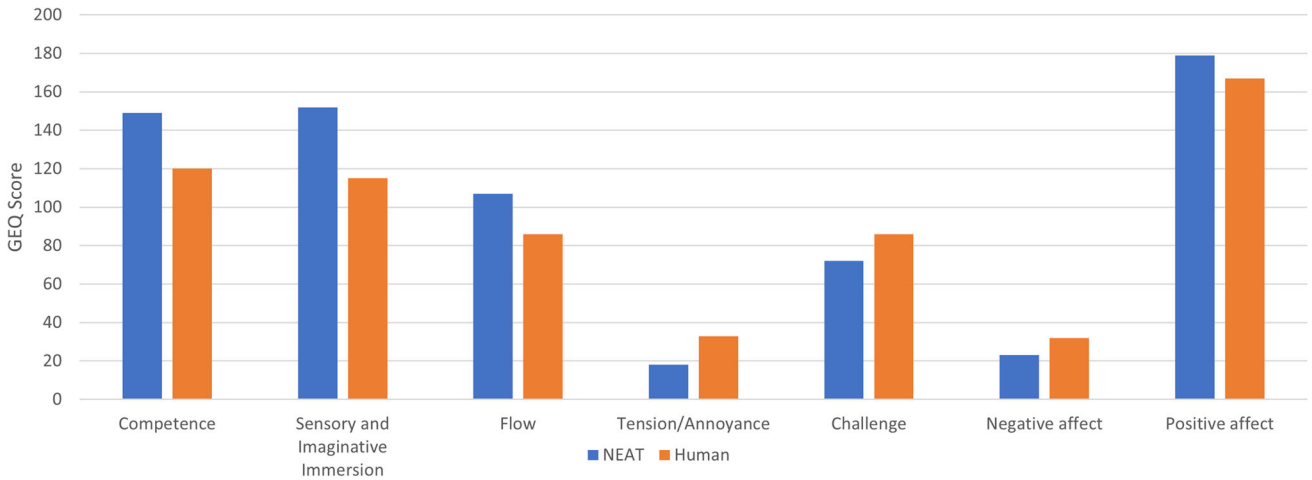

Fig. 5 Game experience score summary for the 22 A/B trials

to player strategies and provide a challenging wave, which should theoretically increase the level of engagement through the principles of flow. However, the challenge scores were lower than expected, and this may explain why the related tension/annoyance scores were also lower than‘ anticipated.

One of the main findings of the survey is that the AI waves were generally more engaging than the human-designed waves, scoring 107 in the flow category, compared to the human waves which only scored 86. Another key category was tension/annoyance, where the AI scored a total of 18 compared to the 33 of the human waves. This suggests that players were generally less frustrated when playing against the AI, which likely contributes to the higher flow score too. There was a concern that the AI would score higher in this category since it would be less predictable and intentionally counter the player's strategy, which some players could find frustrating.

Despite the apparent advantages of the AI wave manager, a deeper look at the data in Table 3 reveals that the results are not statistically significant when comparing the two groups. This leads to the acceptance of a null hypothesis, indicating that means are equal under testing and that there is no significant difference between the two groups. No group can be determined to be better or worse than the other with this sample in terms of the dependent variable of Game Experience Factors. In this sample, the NEAT-guided wave management system was observed to perform the function of a human wave designer while delivering a similar (not statistically significantly different) game experience to a player audience.

4.1 Engagement

An objective of this project was to determine whether an evolving neural network could be employed to enhance player engagement in video games. The results of the blind A/B test GEQ survey did not show any statistically significant improvement in player flow. However, the AI waves did score higher than the human designs for flow, indicating that the AI was subjectively evaluated to be comparable to the human designs in this sample.

The plan to increase engagement involved using the ANN to ensure that the player is constantly being challenged, which in turn increases engagement due to the theory of flow. However, the survey results demonstrated that players generally found the AI waves to be less challenging than the human waves, implying that the AI did not achieve its intended goal. There are several potential factors that may have contributed to the AI being less challenging than the human waves. For instance, the ANN may not have been trained for a sufficient period, the method of training may not have been representative of actual gameplay, or the fitness function used may not have been adequate, thus training bad habits.

Another possible reason could be the way the AI structures its waves. The AI can spawn any enemy type on any wave, which means that the AI wave manager may do a better job of naturally introducing each enemy type to the player earlier in the game compared to the human waves, which do not introduce some of the harder units until later waves. This may result in players playing against the AI waves developing better strategies earlier in the game as any weaknesses in their defences are exposed earlier in the game, where the punishment for letting their defences get overwhelmed is far less significant and less likely to lead to a game over, unlike later in the game when the human-designed waves introduce certain challenging enemies like the carrier unit. While this survey did not include any metrics to help confirm this theory, a future study could track the damage taken by the player over the course of the game to determine if this theory is correct.

Despite the fact that the AI waves were perceived as being less challenging than the human waves, the AI still scored higher in flow. This may indicate that the human waves were too challenging, as they also had a higher score for tension/annoyance, so lowering the challenge could have increased flow if the high level of challenge was the limiting factor at the time. While the AI and human waves conform to the same spawn points limitation in an attempt to keep them balanced, there are other factors that may affect the difficulty of the human waves. Unlike the human waves, the AI waves are structured with a constant 1-s gap between each enemy being spawned, and enemies are spawned from highest cost to lowest cost, meaning that the structure and timing are predictable and consistent across all AI waves. The human waves, on the other hand, are intentionally designed to take advantage of the natural synergies between certain enemy types, such as combining support units with closely grouped collections of transport units. Although these combinations are possible to generate with the current AI implementation, they are not curated by design intervention, so it is possible that these specifically designed combinations and unique wave timings are contributing to the increased challenge of the human waves.

The results of this study suggest that an AI approach like this has the potential to improve engagement in video games, but further research is needed to draw a more definitive conclusion. There are numerous factors at play when it comes to increasing engagement, and the results can be unpredictable. Therefore, each game that uses this approach would likely require a bespoke and fully tested system. This may prove to be too costly and risky for some game developers. However, if a game were to be designed with this method in mind from

the start, it is possible that it could benefit greatly from this approach.

4.2 Content generation

A second objective of the project was to investigate the feasibility of using an artificial neural network (ANN) to generate valuable game content, and it has successfully achieved that goal. Although the survey results were inconclusive, the AI-generated waves were not statistically separable from the human-generated waves, implying that the AI-generated content was subjectively rated comparably to the human content, and therefore has some value in automated game design processes.

One of the key benefits of an AI content generation system, such as the one developed for this project, is that it allows for endless content generation and replayability. The AI generates waves in real time as the player is playing, so there is no limit to the amount of content that can be generated, resulting in theoretically unlimited playtime. This could increase revenue streams through replayability, which is often an important factor for prospective players. Another significant advantage is that time and money can be saved during development by having the AI handle all of the wave generation, while human designers focus on other aspects of the project. Designing unique waves for each level and difficulty can be a time-consuming process, and the ANN developed in this research does not require specific training for each level, enabling rapid iteration and early playtesting of levels without needing to curate waves. The system allows designers to tweak parameters such as the allowed spawn points and individual cost for each unit, and it is easy to add additional tweakable parameters, such as blocking certain unit types from spawning until specified waves or changing the wave timings.

With this in mind, an additional consultation phase took place with 19 professional game designers, drawn from convenience sampling connections at game studios local to Silicon Spa in the West Midlands of the UK. These expert participants were presented with the game demonstration, accompanied by an explanation that the waves were generated by AI and asked questions derived from the System Usability Scale:

1. I think that I would like to use this tool frequently in my design work.
2. I would find this design tool unnecessarily complex.
3. I think the tool would be easy to use.
4. I think that I would need the support of a technical person to be able to use this.
5. I think the various functions in this tool could be well integrated.
6. I think there may be too much inconsistency in this tool.
7. I would imagine that most people would learn to use this as a design tool very quickly.
8. I would find the tool very cumbersome to use.
9. I would feel very confident using this tool in design work.
10. I would need to learn a lot of things before I could get going with this system.

This was a consultation on the possibility of tool adoption into the design workflow and was not an evaluation on a fully functioning system, and mockups and prototypes are commonly evaluated in this way with SUS [25]. The average SUS score reported was 75.8, which is considered to be "Good" and participant-level data for each question is presented in Fig. 6.

However, there are some potential drawbacks to this approach. The initial training of the network took a significant amount of time and could prove to be a bottleneck in the game development process, as the rest of the game mechanics need to be in place before the network can begin learning. More complex games with additional features and mechanics would likely require more training time and bespoke automated training configurations, particularly if the player can impact the wave while it is in progress. Updating games with balance changes or new content post-release is a common practice throughout the industry, but making any changes to the game will require the network to be retrained. For small balance changes, this may not be too costly, but larger changes, particularly with new towers or enemies, would require the entire network to be retrained rather than fine-tuned.

An alternative approach to having the AI generate waves from scratch would be to have it choose from pre-set subwaves designed by humans. For example, a subwave of support and transport units with their wave timings set, or a subwave of alternating physical and electrical resistant enemies. This could provide some of the benefits of human-designed content, such as customised wave timings, designed enemy synergies, and more unit variation, while still allowing for potentially endless content generation. However, this would require additional human design time, but the higher quality content may make it worthwhile. Another potential improvement could be to allow designers to set which enemy types are allowed to spawn on which waves, enabling a more naturally designed difficulty curve as new units could be introduced one at a time, which is typical for the genre.

While the specific system developed for this research may not be suitable for use in a real product, a similar purpose-built system could be viable for use within video games to generate endless content. The research demonstrates that AI-generated content can be subjectively rated to be comparable to human-designed content, providing value to developers who want to save designer's time.

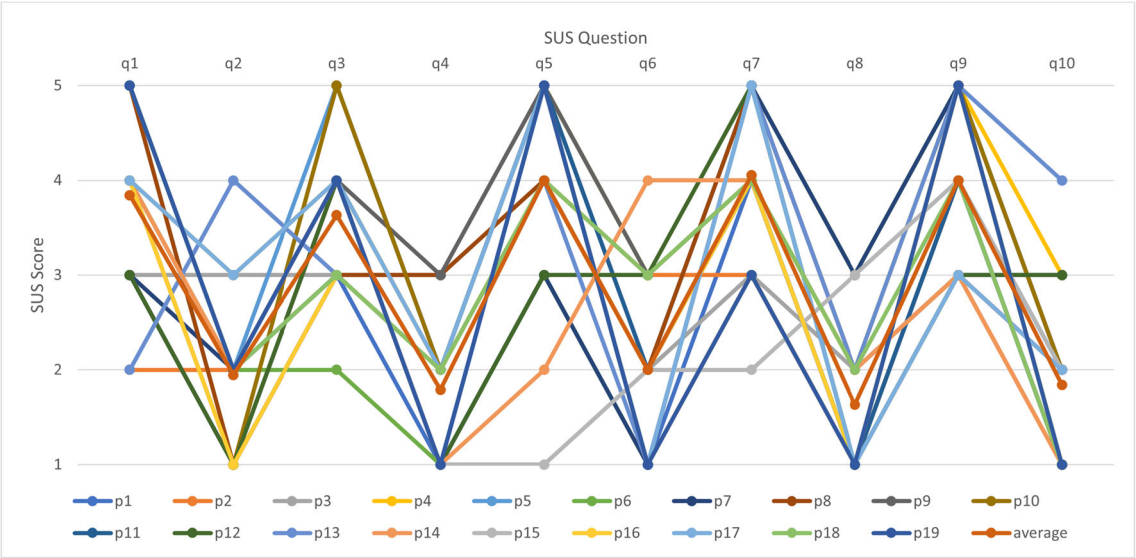

Fig. 6 System Usability Scores from 19 game designers evaluating the possibility of this technique as an assistive tool in the design process. The SUS scores in general cannot be considered in isolation; however, when looking across the trends, it is clear that distributions across every participant follow the common positive and negative responses

4.3 Future use cases

The presented methodology for automated content generation using artificial neural networks has a range of potential applications in video game development beyond the tower defence game explored in this project. One promising avenue for this methodology is its potential to aid game designers in balancing a game. During the training process of the tower defence game, it was observed that the AI-generated content exhibited a preference for certain enemy types over others. This observation helped to fine-tune the enemy stats and ensure that the game units were more balanced.

As shown in Fig. 7, which depicts the proportions of each enemy type spawned in the final network, data such as this is very useful for game designers who are looking to create a balanced game experience. The proposed approach can enable game designers to make balance changes to a game, train the AI model, and harvest data from the resulting champion network to observe how the balance changes affected the game compared to the results before the changes. By running these simulated waves constantly, this approach can also be used as a form of automated testing, and additional analytics can be recorded to provide more insight into the health of the software.

This methodology offers numerous advantages over traditional methods of game balancing. First, it allows for automated and continuous game balancing, enabling designers to make changes and test them in real time. Second, it saves time and resources that would otherwise be spent on manual testing, as the AI generates a large amount of data that can be used to analyse the impact of balance changes. Third, it provides objective data on the effectiveness of balance changes, which can help to avoid subjective bias in the design process.

However, some challenges may arise when applying this methodology to other game genres. For instance, more complex games with a greater number of game mechanics and features may require longer training times and more bespoke configurations for training the model. Additionally, for games where the player can impact the wave while it is in progress, balancing the game may be more challenging. Moreover, as the network needs to be retrained whenever

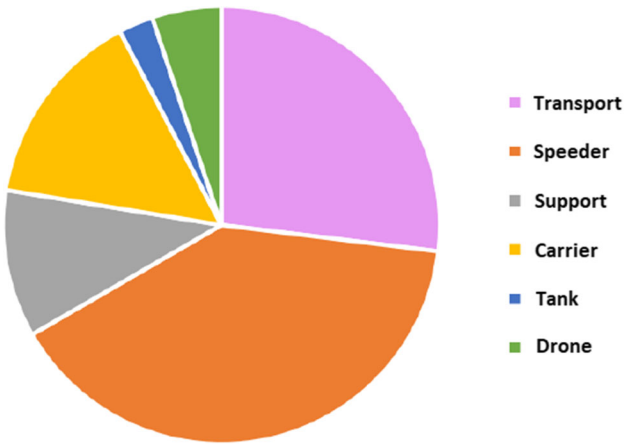

Fig. 7 Proportion of each enemy spawned by the final champion network when being run through the ten test lanes

significant balance changes are made, updating games with new content or balance changes may be a more costly and time-consuming process.

This methodology has the potential to be applied beyond tower defence games and could easily be mapped to other prominent game genres in which the player competes against an AI opponent, such as real-time strategy games, base-building games, or horde-style games. For instance, in Left 4 Dead 2 (2009), a similar system is used to observe the current game state when procedurally generating enemy waves, but these encounters are predetermined and limited in their nature. However, a machine learning approach could allow for this content to be generated infinitely, resulting in enemy waves that are far less predictable and repetitive.

Furthermore, AI-generated content like this is not necessarily limited to the wave generation demonstrated in this project. Given the promising results of this experiment, it is possible that a similar machine learning approach of allowing a network to observe the current game state and make informed decisions could be applied to other areas of game development. For example, an artificial neural network could be trained to generate rooms and scenarios in a dungeon-crawler-style game, or it could generate dynamic quests in a role-playing game. The potential applications of this approach are near limitless, providing game designers with a powerful tool to create novel and engaging content.

Moreover, this approach could be a great way to incorporate dynamic difficulty adjustment (DDA) within games, as the AI can observe the game state to see how well the player is doing and adjust difficulty accordingly. This could provide a more tailored and enjoyable experience for players, while also improving the overall game design. By using this method, game designers can continuously test and fine-tune their games, resulting in more balanced and engaging gameplay.

5 Conclusion

This work explored the use of an evolving neural network approach to generate dynamic content for video games, focusing on a tower defence game. The study demonstrates that the NEAT technique can effectively train a neural network as a wave manager to generate challenging enemy waves. The AI-generated content was found to be comparable to human-designed content in terms of observed means in player experience, engagement, and subjective ratings. The approach has the potential to reduce the time and resources required for designing game content while maintaining the quality of the player experience. It offers possibilities for creating personalised and engaging gaming experiences in various genres and can aid in game balance and automated testing. The research suggests the potential of AI-driven content generation systems as valuable tools for game development.

The findings suggest that this approach led to an increase in overall player engagement; however, it also resulted in a decrease in the game's perceived difficulty. This observation is contrary to the assumption that a link between challenge and engagement exists, as posited by the principles of flow. Therefore, additional research is required to further evaluate the applicability of this approach.

Although there were some differences in means between the human-designed and AI-generated waves, the results indicated no statistically significant difference between the two. This outcome implies that in this sample, the AI-generated waves were subjectively rated as similar to those designed by human game developers, which suggests the potential of this approach as a form of dynamic content generation.

Moreover, the present study revealed several potential uses for this technology, such as game balance and automated testing. The approach used in this research could serve as a valuable tool for game designers to fine-tune enemy stats and create a balanced experience. Additionally, the simulation and automated testing capabilities of the proposed methodology could provide valuable insights into the health of the design and interplay of agents in simulation software.

Acknowledgements The RTX A6000 used for this research was donated by the NVIDIA Corporation.

Author contribution Author 1 and Author 2 contributed equally to the conceptualisation and design of the study. Author 1 supervised the implementation of the NEAT technique and wave manager, and conducted the A/B testing with the Games Experience Questionnaire (GEQ). Author 2 supervised the development of the NEAT-generated curriculums for tower deployments and contributed to the data analysis. Author 2 provided guidance and oversight throughout the project as the senior author and student supervisor. All authors contributed to the writing and editing of the manuscript.

Availability of data and materials The data used in this paper are available upon reasonable request from the corresponding author. The Games Experience Questionnaire (GEQ) is a publicly available survey instrument [24].

Code availability The code used for the NeuroEvolution of Augmenting Topologies (NEAT) technique and wave manager implementation is not available due to licences of the code that was modified within this project (original code base cited in the manuscript).

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Olesen JK, Yannakakis GN, Hallam J (2008) Real-time challenge balance in an RTS game using rtNEAT. In: 2008 IEEE symposium on computational intelligence and games. pp 87–94. https://doi.org/10.1109/CIG.2008.5035625
2. Ibrahim MY, Sridhar R, Geetha TV, Deepika SS (2019) Advances in neuroevolution through augmenting topologies - a case study. In: 2019 11th International conference on advanced computing (ICoAC). pp 111–116. https://doi.org/10.1109/ICoAC48765.2019.246825
3. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Pereira F, Burges CJ, Bottou L, Weinberger KQ (eds) Advances in neural information processing systems, vol. 25. Curran Associates, Inc., ???. https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf
4. Guo Q, Yu Z, Wu Y, Liang D, Qin H, Yan J (2019) Dynamic recursive neural network. In: 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR). pp 5142–5151. https://doi.org/10.1109/CVPR.2019.00529
5. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735, https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf
6. Hastings EJ, Guha RK, Stanley KO (2009) Automatic content generation in the galactic arms race video game. IEEE Trans Comput Intell AI Games 1(4):245–263. https://doi.org/10.1109/TCIAIG.2009.2038365
7. Cho S-H, Kang S-J (2011) An automated wave generation technique in tower defense games based on a genetic algorithm. J Korea Game Soc, Korea Acad Soc Games. https://doi.org/10.7583/jkgs.2011.11.2.19
8. Risi S, Togelius J (2014) Neuroevolution in games: state of the art and open challenges. arXiv:1410.7326
9. Wang H, Smys S (2021) Overview of configuring adaptive activation functions for deep neural networks-a comparative study. Ubiquitous Comput Commun Technol (UCCT) 3(1):10–22
10. Baldominos A, Saez Y, Isasi P (2020) On the automated, evolutionary design of neural networks: past, present, and future. Neural Comput Appl 32(2):519–545. https://doi.org/10.1007/s00521-019-04160-6
11. Racaniere S, Lampinen AK, Santoro A, Reichert DP, Firoiu V, Lillicrap TP (2019) Automated curricula through setter-solver interactions. https://doi.org/10.48550/ARXIV.1909.12892
12. Guzdial M, Riedl M (2018) Automated game design via conceptual expansion. arXiv:1809.02232
13. Khalifa A, Bontrager P, Earle S, Togelius J (2020) PCGRL: procedural content generation via reinforcement learning. arXiv:2001.09212
14. Kim H, Lee S, Lee H, Hahn T, Kang S (2019) Automatic generation of game content using a graph-based wave function collapse algorithm. In: 2019 IEEE conference on games (CoG). pp 1–4. https://doi.org/10.1109/CIG.2019.8848019
15. Sandhu A, Chen Z, McCoy J (2019) Enhancing wave function collapse with design-level constraints. In: Proceedings of the 14th international conference on the foundations of digital games. FDG '19. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3337722.3337752
16. Li X, He S, Dong Y, Liu Q, Liu X, Fu Y, Shi Z, Huang W (2010) To create DDA by the approach of ANN from UCT-created data. In: 2010 International conference on computer application and system modeling (ICCASM 2010), vol. 8. pp 8–4758478. https://doi.org/10.1109/ICCASM.2010.5620008
17. Ebrahimi A, Akbarzadeh-T M-R (2014) Dynamic difficulty adjustment in games by using an interactive self-organizing architecture. In: 2014 Iranian conference on intelligent systems (ICIS). pp 1–6. https://doi.org/10.1109/IranianCIS.2014.6802557
18. Beck LA, (1992) Csikszentmihalyi, mihaly. (1990) Flow: the psychology of optimal experience. J Leisure Res 24(1):93–94. https://doi.org/10.1080/00222216.1992.11969876
19. Zohaib M (2018) Dynamic difficulty adjustment (DDA) in computer games: a review. Adv Human-Comput Interact 2018:5681652. https://doi.org/10.1155/2018/5681652
20. Sutoyo R, Winata D, Oliviani K, Supriyadi DM (2015) Dynamic difficulty adjustment in tower defence. Procedia Comput Sci 59:435–444. https://doi.org/10.1016/j.procs.2015.07.563. (**International Conference on Computer Science and Computational Intelligence (ICCSCI 2015)**)
21. Du Y, Abbeel P, Grover A (2022) It takes four to tango: multiagent self play for automatic curriculum generation. In: International conference on learning representations. https://openreview.net/forum?id=q4tZR1Y-UIs
22. Makri S, Charalambous P (2022) Navigating a road network using reinforcement learning. In: 2022 International conference on interactive media, smart systems and emerging technologies (IMET). pp 1–8. https://doi.org/10.1109/IMET54801.2022.9929765
23. Winkler S (2009) On the properties of subjective ratings in video quality experiments. In: 2009 International workshop on quality of multimedia experience. pp 139–144. https://doi.org/10.1109/QOMEX.2009.5246961
24. IJsselsteijn WA, de Kort YAW, Poels K (2013) The Game Experience Questionnaire. Technische Universiteit Eindhoven. https://pure.tue.nl/ws/files/21666907/Game_Experience_Questionnaire_English.pdf
25. Pinem AA, Yeskafauzan A, Handayani PW, Azzahro F, Hidayanto AN, Ayuningtyas D (2020) Designing a health referral mobile application for high-mobility end users in Indonesia. Heliyon 6(1):03174