Fall 12-12-2016

Application of Neural Networks for Intelligent Video Game Character Artificial Intelligences

Simon D. Mikulcik
Eastern Kentucky University, simon_mikulcik@mymail.eku.edu

EASTERN KENTUCKY UNIVERSITY

Application of Neural Networks for Intelligent Video Game Character Artificial Intelligences

Honors Thesis

Submitted

in Partial Fulfillment

of the

Requirements of HON 420

Fall 2016

By

Simon Mikulcik

Dr. George Landon

Associate Professor, Computer Science

Application of Neural Networks for Intelligent Video Game Character Artificial Intelligences

Simon Mikulcik

Dr. George Landon, Associate Professor, Computer Science

Much of today's gaming culture pushes for increased realism and believability. While these movements have led to much more realistic graphics, we also need to keep in mind the behavior of artificial intelligence characters also in the game. Neural Networks are complicated data structures that have shown potential to learn and interpret complex behavior. This research analyzes the application of neural network as primary controllers for video game characters' artificial intelligence. The paper considers the existing artificial intelligence techniques, and the existing uses of neural networks. It then describes a project in which a video game was created to serve as a case study in analyzing neural network controlled game characters. The paper discusses the design of the game where the player communicates with an artificial intelligence astronaut. The way that the player phrases the messages he sends to the astronaut helps determine whether the astronaut survives. The paper then analyzes the effectiveness of the astronaut character in exhibiting intelligent behavior. It then discusses potential future work in better demonstrating effective neural network controlled artificial intelligences.

Keywords: artificial intelligence, video game, neural network, behavior, learning, feed-forward neural network, backpropagation, spaceship, astronaut, honors thesis

Table of Contents

List of Figures

Acknowledgements

Firstly, I would like to express my gratitude for my mentor Dr. George Landon for encouraging me to develop video games. Although I have learned many impractical skills making games, I have also learned how to work as a team, how to produce a product, and most importantly, how to make computer programs for their own sake of having fun. I would also like to thank the Honors Program Director Dr. David Coleman and the rest of the Honors faculty and staff at EKU for their support, guidance, and opportunities to share my work with others. Finally, I would like to thank my gracious friends and family for alpha and beta testing my game.

Introduction

In video games, the demand for realism is very high. Gamers want to play games that are highly interactive, and believable. Thus, they want the characters in the games to be more believable. When developing a video game character, the more intelligent they seem, the better gamers will receive them. However, much of the research in computer science has yet to be implemented in the game industry even though machine learning and neural networks have been employed in other fields to provide more human-like learning and behavior. Our research analyzes the application of neural networks for use in video game character artificial intelligence.

In the creation of interactive video games, game developers have employed various techniques to provide an interesting, immersive experience. However, for a game to be very immersive, it must have some form of realism or believability. Developers have had success by scripting complicated behaviors for characters that 'appear' to be intelligent, but upon playing the game enough, their behavior becomes predictable and the illusion of realism fades (Schwab, 2009, p. 76). Recently though, neural network based systems have been able to learn and adapt to react to the actions of the player in a manner similar to that of a real human opponent. Neural networks play a crucial part in the development of interactive video game artificial intelligences. I expect to see that neural network based AIs will create a more immersive experience for the player than would a traditional AI technique.

Related Work

Background Information

Prior to neural networks, developers employed several development techniques to create interactive artificial intelligence agents. One technique to use is the finite state machine. Finite state machines are one of the earliest techniques for describing behavior, but they are still widely in use today (Bourg, 2004, p. 165). Suppose we develop a finite state machine for a shopkeeper in a store. The shopkeeper may be idle, conversing with a customer, or going to fetch an item for a customer. We would have several transitions between these states. For instance, if idle, when a customer approaches the shopkeeper, he begins conversing, and once the customer requests an item, he fetches the item. The game developer would then code in various transitions like these between the set of the states.

Using finite state machines is very useful for programming systems when the desired behavior can be broken down into a set of states and transitions. However, as the finite state machine is deterministic, the behavior of a finite state machine controlled AI may become predictable upon multiple encounters with the same agent.   Also, the behavior of the agent is set and cannot change for new conditions.

Another tool that developers may use for AIs is the rule based system. A rule based system has a predefined set of rules that dictate the proper action for an agent to follow (Bourg, p. 212). Consider, for instance, an enemy in a combat game. Suppose we decide that the enemy will be idle most of the time except when the player is within eye-sight and has a weapon equipped. Thus, we have defined a set of rules that dictate the enemy character. This technique is simple as it is straightforward to implement; however, it often becomes predictable if the player encounters a similar AI multiple times. Also, like finite state machines, rule based systems do not perform well for accommodating to new situations as they cannot adjust its rules on its own.

To add a bit of randomness to the behavior of game characters, developers may employ a genetic algorithm to help make the agents less predictable. In a genetic algorithm, the developer mimics the process by which natural selection determines which genes to pass on (Schwab, p. 515). In this, the developer defines a set of parameters (also called a genome) such as accuracy, balance, agility, height, weapon skill, and then he generates an initial population of randomly modulated parameters. Once he has a population of these genomes, he evaluates the usefulness, or fitness, for each potential character and recombines the features of the genomes with the higher fitness rating. This technique allows for the characters to be less predictable since they each have their own parameters. Also, since genetic algorithms recombine genomes to tweak the parameters, these systems may 'learn' to adjust to the environment to perform better over time. However, this technique requires a large amount of processing power to determine which parameters to set for the characters.

Similar to genetic algorithms, neural networks mimic natural processes in order to add a level of unpredictability and realism to the artificial intelligences. In the human brain, there are billions of neurons that receives signals from surrounding neurons and transmit signals to other neurons (Schwab, p. 556). We call the connection between these neurons synapses. In computer science, this has been modeled by having a collection of nodes each of which sends signals to other nodes in order to describe complicated behavior among many inputs. This networks of nodes is then called a neural network.

Neural networks have been shown to describe many complicated interactions very well. The problem with this technique is that it is often difficult to determine how the network shall be constructed in order for it to exhibit the desired behavior. Similarly, as humans we learn information in order to ‘determine’ the connections in our ‘neural network’. Accordingly, there are processes for determining how artificial neural networks should be constructed. Thus, by using artificial neural networks, we are able to code a system that can exhibit complex behavior as well as process new information by learning. This quality is of particular interest to us as we may use this technique to develop game characters that may learn and interact more realistically. However, one problem with this technique is that using such a system is much more complicated than a rule based system or a finite state machine and requires, like genetic algorithms, much more processing power to determine its conclusions.

One common type of neural network is the feed-forward, backpropagation neural network. In this kind of network, the nodes are grouped in layers where the first layer is the input layer, the last layer is the output layer, and all intermediate layers are called hidden layers (Schwab, 558). Each node in a layer may only be connected to nodes in the previous layer, and all outputs of the nodes in a layer may only be received by nodes in the next layer. Thus, we have a directed ordered network with no cycles, and so we have the name feed-forward. Also, for our network, we will assume that each synapse between two nodes has a weight value that scales the effect of that particular input connection. So, to determine the state of the network, we set the number of layers, number of nodes in each layer, and the weights of each synapse.

In order for developers to train a network, one technique that they may use is backpropagation. In backpropagation, they provide the network with pairs of inputs and desired outputs. Thus, as the developer 'supervises' the training of the network, this technique is classified as supervised learning. In game development, for instance in a fighting game, the inputs may be the actions of our opponent and the outputs may be the actions of the system. They run the network with the input and compare how close its output is to the desired output to compute an error value. Then, they adjust the weights in the network in order to minimize the error value for each node (Schwab, p. 582). However, as the network is very complicated it is expensive to determine how much to adjust each synapse weight individually. So, they use the backpropagation optimization of computing the error per synapse starting from the output layer and moving backwards to the input layer. Hence, this technique has the name backpropagation. Upon many repetitions of the process, they are able to effectively train the network to produce the desired output.

Another way of training a network is to use reinforcement learning. Instead of using backpropagation with sets of input/output pairs, suppose the developer specifies inputs and only tells the system whether the output is desirable or not. In reinforcement learning, when the network performs a desirable action and it receives positive feedback, it adjusts its parameters to maximize the positive feedback (Taylor, 2014, p. 46). It is important to note that with reinforcement learning, the system is not presented with specific desired output. So, the system never actually knows if its output is ‘correct’, rather that it may only know that it was more correct than it was before. Since reinforcement learning does not necessarily require optimal output in order to train the network, reinforcement learning looks attractive to the game developer who may not know the best way to play the game.

Notable Studies Involving Neural Networks

In the computer science community, there have been many artificial neural network applications in fields ranging from sports and weight training to human biometric identifications. In 2013, the *Journal of Sports Science and Medicine* published a research article by Hristo Novatchkov where he analyzed the application of artificial neural networks to weight training. Novatchkov recorded input from weight training machines and used a neural network to analyze the effectiveness of the exercise as well as any potential safety hazards from misusing the training equipment. Novatchkov found that his neural network was able to provide useful feedback for the users and has found the technique to show promise for future exercise analysis technologies (p. 36). Novatchkov’s tool was able to aggregate large amounts of input data to analyze the effectiveness of the actions of a human using the system. In video game development, we may hope to expect positive results for similar systems that gather human input data. For instance, an enemy AI may analyze the motions of the human player to analyze game performance and accommodate strategy accordingly.

On the other hand, artificial neural networks may also be used in driving analysis applications. In 2015, Gys Albertus Mathinus Meiring reviewed several algorithms to analyze driving style. Meiring wanted to find appropriate machine learning techniques to best determine safe driving style. For instance, Meiring found that while finite state machines may be useful in modeling driver decision making, artificial neural networks are effective in detecting drowsiness, and predicting steering behavior (p. 30669). Meiring analyzed many more algorithms than what we have discussed thus far. However, Meiring did show that in real-world applications, it is often preferable to use the artificial intelligence technique that best fits the problem. Likewise in video game programming, we must be sure to use the appropriate technique for the problem.

Apart from analysis software, artificial neural networks have been used for classification and identification purposes. In 2013, Fadi Sibai researched the application of feed-forward neural networks in the identification of a person by their ear shape. Just as the human fingerprint is an identifying biometric, so is the shape of the human ear (p. 1265). Sibai was able to apply his neural network to images of a person's ear and identify them with 95% accuracy (p. 1272). Thus, Sibai was able to use a neural network for the classification. Neural networks are often effective for classifying items into categories based on a set of input parameters. For game programming, we can expect to obtain similar results for classification problems. For instance, in a survival, adventure game we may use such an algorithm for determining what the AI character should do next, be it gathering food, money, or weapons.

Apart from specific techniques used in developing a game AI, we also need to consider the effectiveness of the game character in being intelligent. Intelligence is a broad term with many definitions. For our analysis, we used Robert Sternberg's Triarchic Theory of Intelligence. From this definition, intelligence was the ability to be successful in one's environment (Sternberg).

For Sternberg, Analytical Intelligence is the ability to solve a problem through logic and reasoning independent of past experiences (Neill). In a Feed-Forward Neural Network, the decisions made previously have no impact on future decisions. However, a Feed-Forward Neural Network may still perform complex decisions based on all of the input parameters. Thus, for the problems that it is trained to solve, it is able to use reasoning and logic to form a solution.

Creative Intelligence focuses on the ability to apply past experiences to solve new problems (Neill). When considering a Feed-Forward Neural Network, the computation of a single input does not use any past decisions. However, if one performs a learning operation such as Reinforcement Learning after several inputs, it may evaluate its own performance and improve its calculations. Then, with the adjusted weights, it can make better decisions for new problems by using the experiences of previous problems.

Apart from Analytical and Creative Intelligences, Sternberg describes Practical Intelligence as the ability environment (Neill). With the training to achieve goals and show useful behaviors for one’s of ANNs, one either sets the goal as a set of desired outputs as in Supervised Learning, or as a reward function where the goal is to have the most fitness as in reinforcement learning. By incorporating the goal-oriented learning step into the intelligence of the game character, we provide the practical intelligence for the AI to live intelligently in its environment.

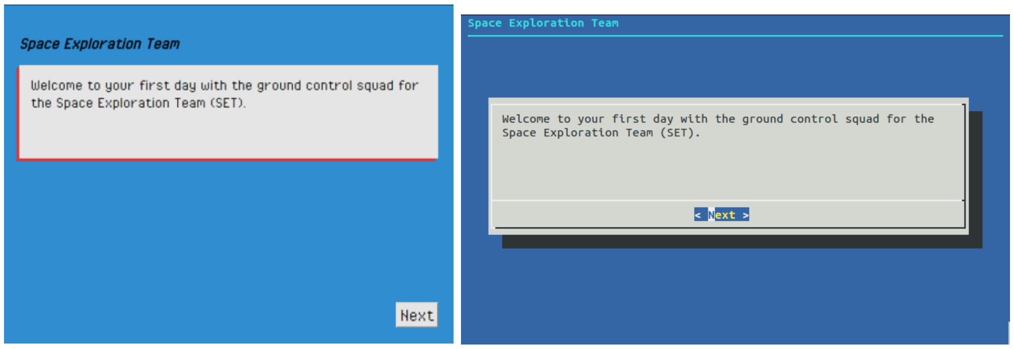
Project Design

To analyze the effectiveness of neural network controlled AIs, I developed a video game demonstration that features one. This game had two main actors: the player (Jeff), and the AI (Anton). The setting of the game is at a Ground Control Station for a space exploration firm. Jeff's serves as a communications operator for the firm. Anton is an astronaut for a foreign nation's space program. Anton ship begins to fail and Anton reaches out to help. Then, when Jeff hears of Anton Distress, he helps Anton regain stability and return safely to Earth.

The game has three phases: the tutorial, the introduction to Anton, and the rescuing of Anton. The introduction phase begins as Jeff's first day on the Job. Often with job employment, the new hire will be required to complete an online training course. So, the player clicks through the training course to learn how to play the game. In this world, Jeff uses a chat application to send and receive messages to communicate with astronauts. Also, as an operator, Jeff may access a remote dashboard to monitor vitals and ship states from his terminal.

User Interface Design

To bring emphasis more on the algorithm behind the AI, I tried to keep the user interface simple, but effective. Since Jeff supposedly works with a Ground Control Space program, I imagined that the software used in running these systems would be older, simple graphics like that of a terminal application. One common tool in developing simple terminal graphics for applications in Dialog. Dialog is commonly used to send simple graphical and textual messages to users and to receive user input. So, I decided to use a theme that resembled the theme of Dialog. See Figure 1 for a comparison between my game, and the interface produced from a one line script using Dialog.


Space Exploration Team

Welcome to your first day with the ground control squad for the Space Exploration Team (SET).

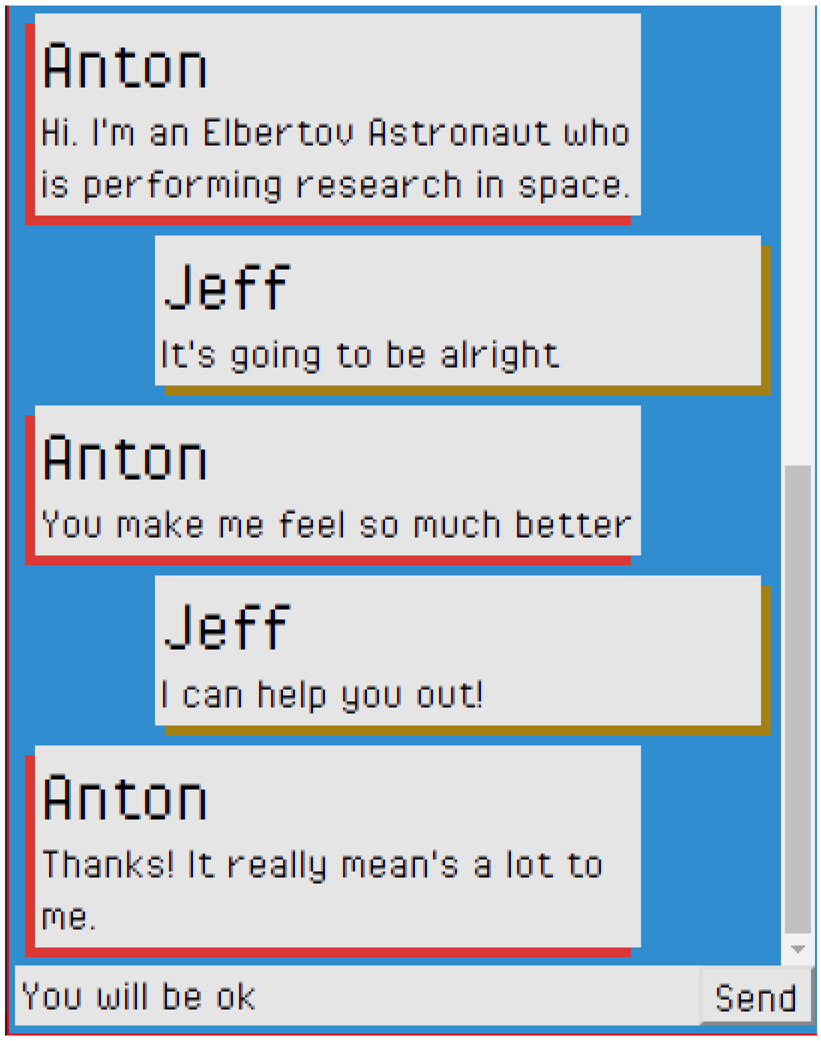
Space Exploration Team

Welcome to your first day with the ground control squad for the Space Exploration Team (SET).

Figure 1: Dubiety Tutorial Phase, left, and Dialog, right

Figure 1:

In the next phase, Jeff chats with Anton. To keep the interface simple, the only controls are for the player to type text into the input box, and to press the send button. These simple controls make it easy for the player to learn how to play the game. Also, the style is very similar to that of the Dialog application. See Figure 2 for an image of the Chat Interface.


Anton
Hi. I'm an Elbertov Astronaut who is performing research in space.

Jeff
It's going to be alright

Anton
You make me feel so much better

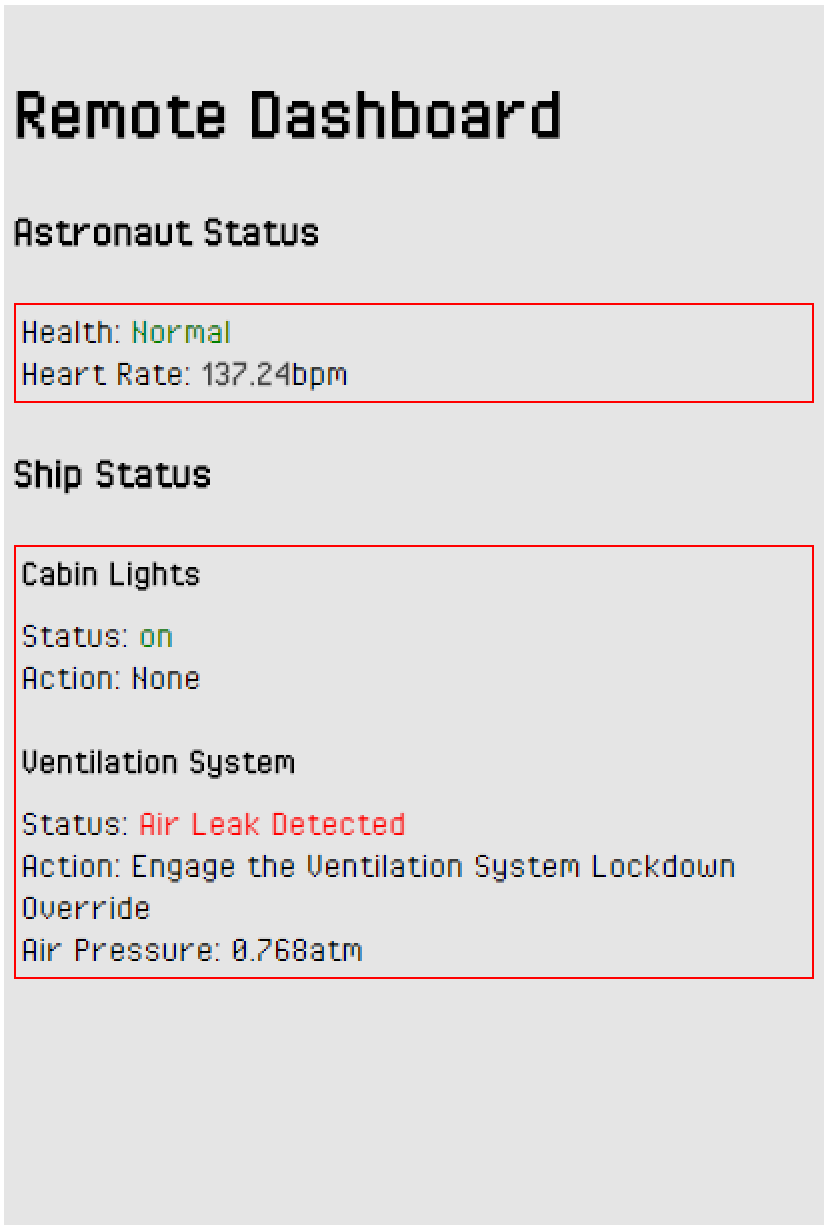
Jeff
I can help you out!

Anton
Thanks! It really mean's a lot to me.

You will be ok

Figure 2: The Chat Interface with text box and button

Finally, for the third phase of the game, I used text based widgets to display information from the remote dashboard. I imagined this dashboard as having many blinking lights, and sensor outputs. I designed the ship to have various subsystems each of which reported their status to the operator. If a subsystem had a critical status, it's status would blink red. Otherwise, it would remain green. These color cues help indicate to the operator, Jeff, how the ship is doing. Also, for vitals, I displayed the heart rate of the astronaut. But I also, made the numbers fade at the heart rate to bring to life the seriousness of the astronaut's health state. Finally, I kept the same theme of having lowresolution, text heavy displays as before. See Figure 3 for a screenshot of the remote dashboard.



Remote Dashboard

Astronaut Status

Health: Normal
Heart Rate: 137.24bpm

Ship Status

Cabin Lights

Status: on
Action: None

Ventilation System

Status: Air Leak Detected
Action: Engage the Ventilation System Lockdown Override
Air Pressure: 0.768atm

Figure 3: Remote Dashboard Screenshot with Subsystems and Vitals

I strived to keep the user interface and graphics simple, but effective. By keeping the number of user controls low, I make it easy for the user to understand how to play the game so that they may focus on the interesting behavior of the artificial intelligence. Also, by using a simple old style for the graphics, I help define the setting for the player.

Artificial Intelligence Design

When Jeff hears of Anton’s distress call, Jeff reaches out to Anton and gets introduced to him. To become acquainted with Anton, Jeff sends textual messages to him. Anton then processes the messaged that Jeff sends and replies with his own message. Anton’s message processing system incorporates several different language processing systems to provide intelligent feedback. Ultimately though, Anton adjusts how he feels (spirits) and how much he trusts Jeff based on what Jeff (the player) says. When Anton trusts Jeff enough, he provides Jeff with access to his remote dashboard so that Jeff can help Anton with his problems on the ship. Since neural networks are great at processing many inputs to determine complicated features, I decided to model the Anton’s trust level and spirits with a neural network. In other words, how the player interacts with Anton will affect how Anton feels and his trust in Jeff.

I used a feed forward, back propagation neural network to control Anton’s state. It is setup with nine input nodes and two output nodes with five hidden nodes as in Figure 4. The network considers the sentiment and tone of whatever the player sends to Anton as well as Anton’s current spirits level and trust level. When a player sends a message, its text is evaluated on its overall sentiment (good/bad) and its tone (anger/disgust/fear/joy/sadness/confidence). These input nodes feed into a feed forward neural network along with Anton’s current state (his spirits and trust level). The network then determines Anton’s new spirits and trust level.

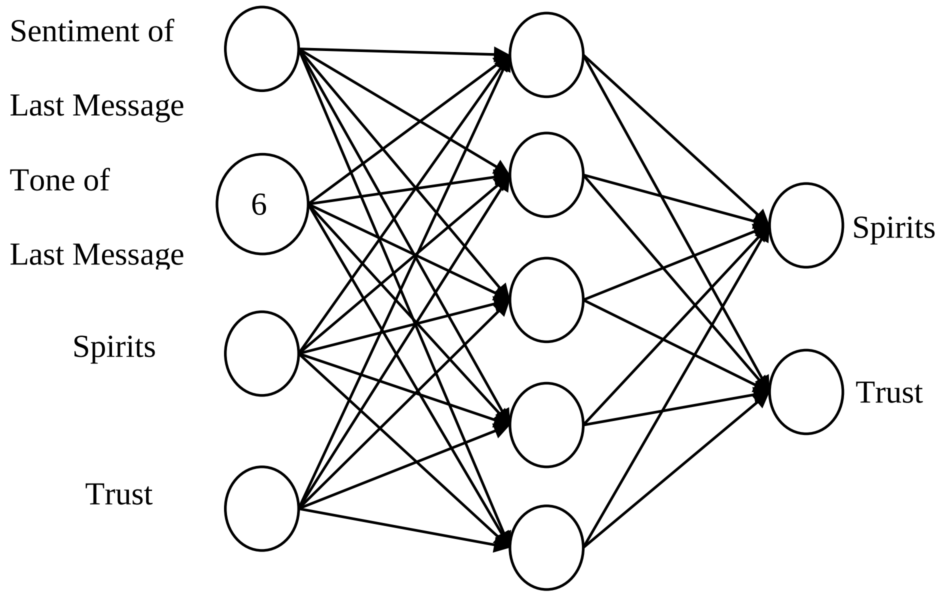


Figure 4: Neural Network Topology

Project Analysis

The game demonstrated an effective neural network controlled AI. The behavior of Anton was based upon the output of the three layered, feed forward back-propagation neural network. The AI exhibited less predictable behavior since the complexities of the neural network made Anton’s rules harder to follow. Also, since the neural network was trained to react to human language input similarly as a human would, Anton provided an interesting personality whose ‘mood’ changes as the conversation progresses.

Intelligence Analysis

We considered the intelligent qualities that Anton exhibited. We used Robert Sternberg’s Triarchic Theory of Intelligence. From this definition, intelligence was the ability to be successful in one’s environment (Sternberg, 1985). He identified three factors that describe intelligence: Analytical Intelligence, Creative Intelligence, and

Practical Intelligence (Sternberg, p. xi). We considered these three factors in our comparison.

For Sternberg, Analytical Intelligence is the ability to solve a problem through logic and reasoning independent of past experiences (Neill, 2004). In a Feed-Forward Neural Network, the decisions made previously have no impact on future decisions. However, a Feed-Forward Neural Network may still perform complex decisions based on all the input parameters. Thus, Anton may use his neural network to react appropriately to Jeff when he sends messages with new types of tone and sentiment.

Creative Intelligence focuses on the ability to apply past experiences to solve new problems (Neill). When considering the basic Feed-Forward Neural Network, the computation of a single input does not use any past decisions. However, Anton's neural network has inputs which were the outputs of the previous step. Thus, the trust level for the next value is dependent on the previous trust level. This way, Anton may provide context depended behavior as he solves the problem of intelligently communicating with Jeff. Thus, Anton exhibits a limited form of creative intelligence.

Apart from Analytical and Creative Intelligences, Sternberg describes Practical Intelligence as the ability to survive and thrive in an environment (Neill). Anton's environment is his ship, and his health. For Anton to survive, he has programmed that he must get help from people who he can trust. Anton starts by reaching out for help, and then, for his self-preservation, he must verify that he can trust Jeff. By striving for survival, Anton is goal oriented. Thus, Anton exhibits some aspects of the Practical Intelligence that Sternberg describes.

By using Neural Networks as the main controller in Anton, we provide some form of each of Sternberg's three factors of Intelligence. The nature of ANNs being able to provide useful output based on a set of inputs, we can analytically solve problems.

Project Limitations

On the other hand, there were some interesting limitations of Anton's AI. Occasionally, the player sends enough 'nice' messages to Anton that his trust level and spirits are extremely high. So, whenever Anton receives a message of any kind, nice or mean, he reacts in a positive manner (See Figure 5). Similarly, if Jeff sends Anton many 'nice' messages, and then sends one 'mean' message, Anton will probably send a 'thank you' message since one 'mean' message will probably not get his spirits too low. Consequently, Anton may respond to a mean comment with a 'thank you' (See Figure 6).

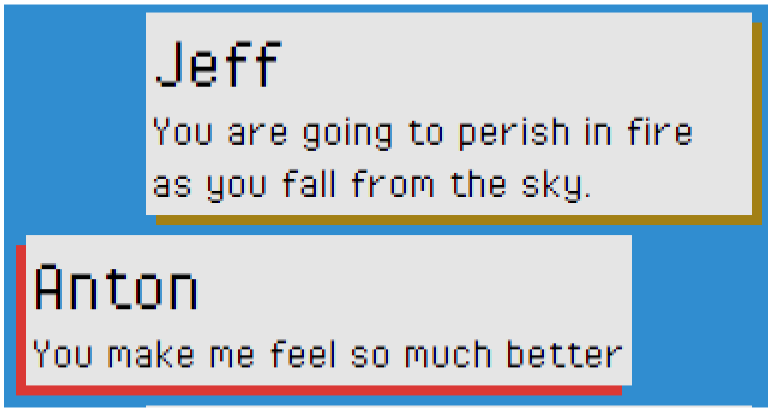

Jeff
You are going to perish in fire as you fall from the sky.
Anton
You make me feel so much better

Figure 5: Complement Loop

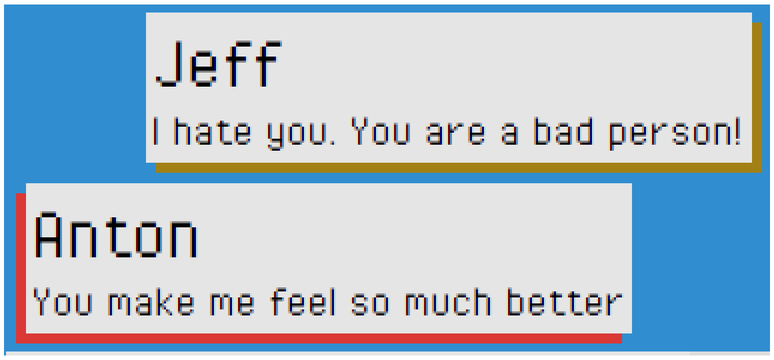
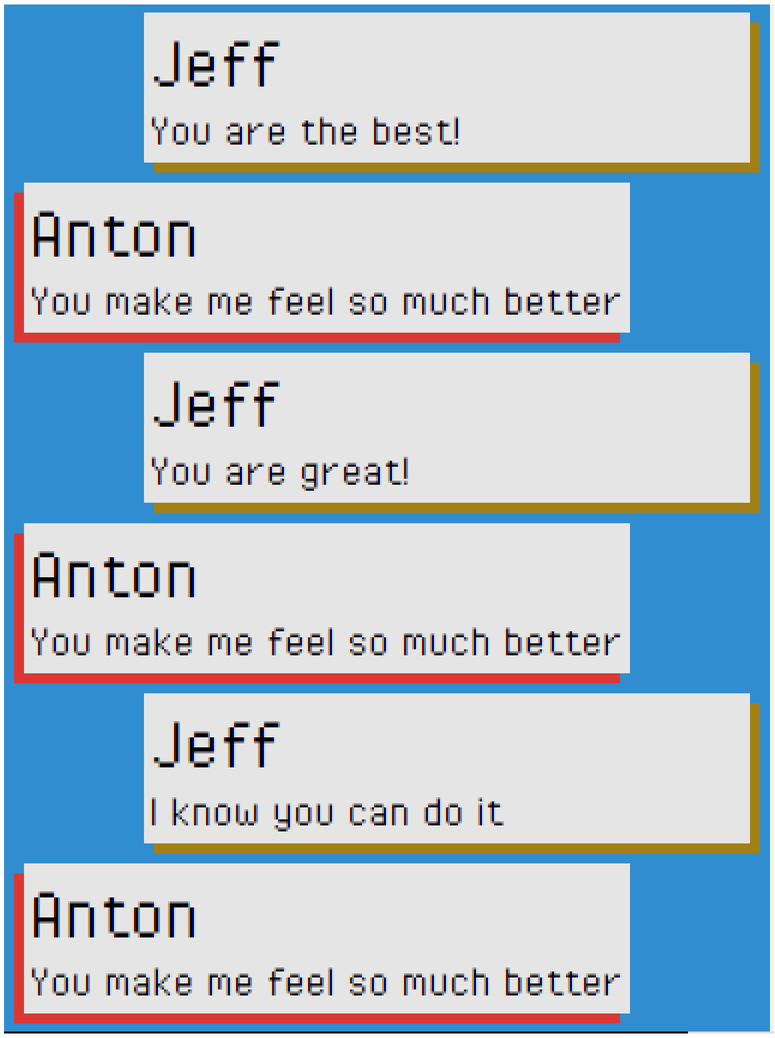

Jeff
I hate you. You are a bad person!
Anton
You make me feel so much better

Figure 6: Slow response to insults.

Another issue that happens was that sometimes Anton repeats the same phrase many times. If Anton's spirits and trust are high enough, he responds with a random message from a set of three pre-scripted 'thank you' responses (see Figure 7). Thus, the odds of repeating the same message three time in a row is one in ten. This behavior is very confusing to a player who expects the AI to use variety in speech as humans do.



Jeff
You are the best!

Anton
You make me feel so much better

Jeff
You are great!

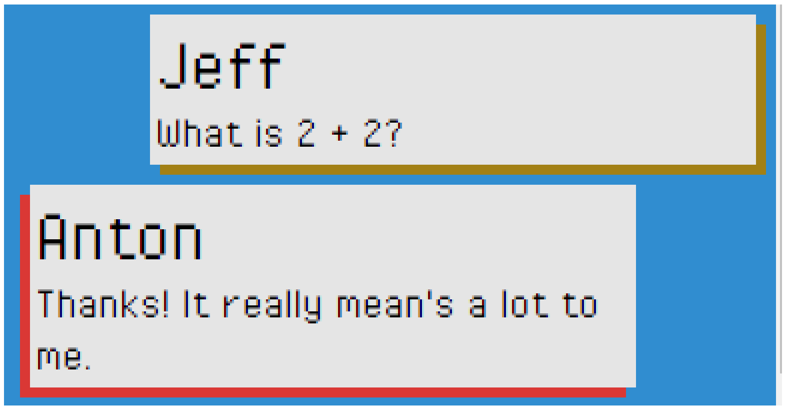
Anton
You make me feel so much better

Jeff
I know you can do it

Anton
You make me feel so much better

Figure 7: Repeated Phrases

While Anton's AI may do well at reacting to comments, he does not react very well to unknown questions. Anton's AI is programmed to provide responses based on *how* the message is phrased. The neural network does not consider the content of a question in determining a proper response. Often, a player will send Anton a question expecting a sensible answer, but Anton responds with a reaction as if the player sent a comment (See Figure 8).



Jeff
What is 2 + 2?

Anton
Thanks! It really mean's a lot to me.

Figure 8: Question Responses

Future Work

Along with addressing the above limitations, there are several other ways for this project to be extended. The purpose of this project was to create more interesting video game characters by using neural networks. Firstly, I should address the immediate limitations. So, one change that would help is to add more response phrases to Anton. This would help reduce the probability of having repeated messages. Also, to address the reaction to insults, I should tweak the training set for Anton so that he reacts more appropriately to insults and compliments. Being more sensitive to particular messages, Anton will be less likely to respond joyfully to an angry message. Also, I should improve Anton's ability to filter out directed questions. This would prevent situations where Anton provides an emotional state response, "It really means a lot to me", when a question has been provided, “What is 2 + 2?”. By addressing these limitations, I would make the game much more entertaining since Anton would appear more intelligent.

New AI Features

I may also add features to Anton’s AI. One idea to improve Anton’s AI is to use a recurrent neural network such as a long short-term memory (LSTM) unit. Recurrent neural networks have the property of storing temporal state information since the outputs of previous activations of the network are stored and used in the calculations of future activations. They have been shown to be effective in human activity recognition (Ordóñez, 2016, p. 22). We should consider using an LSTM layer in Anton’s neural network that may provide better characteristics in responding appropriately in a conversation.

Another feature to consider adding to Anton’s AI is a question engine. Being an video game character, Anton should not have infinite information about the game, nor the environment in which he is living. Anton probably wonders about certain aspects of the game, or the environment. It would be interesting if Anton exhibited a curiosity trait perhaps by asking the user questions. Currently, Anton responds to messages sent by Jeff, and occasionally sends Jeff a status report or an opinion. However, he does not ask any questions. Adding an engine that asks Jeff questions and collects the response data would increase the believability of Anton’s character as well as his perceived intelligence.

I would imagine that the question engine would have a certain probability of firing. When it does, Anton analyzes his environment, finds an aspect about his environment which Jeff should be able to know about with the remote dashboard, and sends a question for Jeff to answer. Then, Anton would listen for a response to the question, and possibly ask the question or give up on hearing a response. For example, if Anton notices that a flashing red light appears in the cabin, he might ask Jeff (who sees that a flashing red light means that the power system is failing), “What does the flashing red light mean?” Anton would then wait for a response of the form “… <subsystem> … is … <state> …”, or “…. <state> <subsystem>…”. This feature would make Anton more believable since it makes that Anton is seeking assistance rather than simply responding to requests.

Extended Character Interactions

While communicating with a single AI is interesting, having multiple AIs and multiple players would create even more complex behavior that may improve believability. One idea is to add a second astronaut on the ship so that Anton and his companion will be able to converse with one another and the player. Then, when we tell Anton messages that make him angry, his companion might also get angry. On the other hand, with multiple characters, they might have slightly different tuned neural networks such that they each exhibit unique ‘personalities’. This would pose a problem to the player since he must interact with each AI according to that AI’s personality. By adding more AIs to the game, the complexity of the interactions in the game increases exponentially.

Similarly, adding multiplayer support for the game could create some interesting, complex behavior. With the current implementation of Anton. The state of the AI character is living on a remote webserver while the player is communicating from a web client. So, suppose we add a second human player to the chat room. Anton would see two ‘strangers’ whose trust must be considered. Each AI, then, would have a list of people to whom he talks. Each person on that list would have their own trust metric to consider. Two players then, might play good cop/bad cop with Anton. Modelling this behavior could produce interesting behavior that might help push the bounds for neural network controlled AIs.

Finally, the game could be improved by extending the plot and adding more subsystems to the ship. The current ship only has two subsystems: the lights, and the ventilation system. I could add more systems such as a power subsystem, a defense subsystem, a ship stability subsystem, and a life-support subsystem. Also, each of these would be interconnected with one another such that when the life-support system starts to fail, it might be because the power system was already failing. By building these interconnections of the subsystems on the ship, the second phase of the game becomes more challenging, and fun.

Conclusion

This project incorporates a neural network controlled AI that exhibits some limited qualities of intelligence. Although Anton does not always respond appropriately to Jeff’s messages, he does provide a non-deterministic conversation that make it interesting to interact with him. Anton is not simply a rule following AI, so upon a few repetitions, players have a hard time predicting how Anton will respond. Thus, we have shown a viable implementation of a neural network controlled AI that can exhibit higher level intelligence qualities. I hope to improve the game to be more interactive and interesting, but also, I hope to promote the use of neural networks as controllers for the decisions made by video game character AIs.

References

Bourg, D. M., Seemann, G. (2004). AI for Game Developers. Beijing: O'Reilly Media, Inc.

Clarke, A. M., Friedrich, J., Tartaglia, E. M., Marchesotti, S., Senn, W., & Herzog, M. H. (2015). Human and machine learning in non-markovian decision making. *Plos ONE*, *10*(4) , 1-15. doi:10.1371/journal.pone.0123105

Edwards, C. (2015). Growing pains for deep learning. *Communications of the ACM*, *58*(7), 14-16. doi:10.1145/2771283

Forbes, Nancy. (2004). *Imitation of life*. Cambridge, MA: MIT Press.

Gill, S. P. (2008). Cognition, communication and interaction: Transdisciplinary perspectives on interactive technology. London: Springer.

Hermundstad, A. M., Brown, K. S., Bassett, D. S., & Carlson, J. M. (2011). Learning, memory, and the role of neural network architecture. *Plos Computational Biology*, *7*(6), 1-12. doi:10.1371/journal.pcbi.1002063

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Retrieved From http://deeplearning.cs.cmu.edu/pdfs/Hochreiter97_lstm.pdf.

Langlois, M., & Sloan, R. H. (2010). Reinforcement learning via approximation of the q-function. *Journal of Experimental & Theoretical Artificial Intelligence*, *22*(3), 219-235. doi:10.1080/09528130903157377

Learned-Miller, E. G. (2014). Introduction to Supervised Learning. Retrieved From https://people.cs.umass.edu/~elm/Teaching/Docs/supervised2014a.pdf.

Lee, M. (2005). Reinforcement Learning. In Reinforcement Learning: An Introduction. London: MIT Press. Retrieved From https://webdocs.cs.ualberta.ca/~sutton/book/ebook/node7.html.

Li, C., Yang, L., & Lin, M. (2014). Parallel training of an improved neural network for text categorization. *International Journal Of Parallel Programming*, *42*(3), 505-523. doi:10.1007/s10766-013-0245-x

Li, X., Dou, Y., Niu, X., Xu, J., & Xiao, R. (2015). An efficient robust eye localization by learning the convolution distribution using eye template. *Computational Intelligence & Neuroscience*, *2015*, 1-10. doi:10.1155/2015/709072

Meiring, G. A., & Myburgh, H. C. (2015). A review of intelligent driving style analysis systems and related artificial intelligence algorithms. *Sensors (14248220)*, *15*(12), 30653-30682. doi:10.3390/s151229822

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., & ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529. doi:10.1038/nature14236

Neill, J. (2004). Sternberg's triarchic theory of intelligence. Retrieved From http://wilderdom.com/personality/L2-2SternbergTriarchicTheory.html.

Novatchkov, H., & Baca, A. (2013). Artificial intelligence in sports on the example of weight training. *Journal of Sports Science & Medicine*, *12*(1), 27-37.

Oniga, S., & József, S. (2015). Optimal recognition method of human activities using artificial neural networks. *Measurement Science Review*, *15*(6), 323-327. doi:10.1515/msr-2015-0044

Ordóñez, F. J., & Roggen, D. (2016). Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors (14248220)*, *16*(1), 1-25. doi:10.3390/s16010115

Orr, G., Müller K. (1998). *Neural networks: Tricks of the trade*. Berlin: Springer.

Rabunal, J., Dorado, J. (2006). *Artificial Neural Networks in Real-Life Applications.* Hershey, PA: Idea Group Publishing.

Sancho, P., Moreno-Ger, P., Fuentes-Fernández, R., & Fernández-Manjón, B. (2009). Adaptive Role Playing Games: An Immersive Approach for Problem Based Learning. *Journal Of Educational Technology & Society, 12*(4), 110-124.

Schwab, B. (2009). *AI game engine programming.* Boston, MA: Course PTR.

Sibai, F., Nuaimi, A., Maamari, A., & Kuwair, R. (2013). Ear recognition with feed-forward artificial neural networks. *Neural Computing & Applications, 23*(5), 1265-1273. doi:10.1007/s00521-012-1068-1

Sternberg, R. J. (1985). Preface. In Beyond IQ : A triarchic theory of human intelligence. Cambridge Cambridgeshire: Cambridge University Press, xi-i.

Taylor, M. E., Carboni, N., Fachantidis, A., Vlahavas, I., & Torrey, L. (2014). Reinforcement learning agents providing advice in complex video games. *Connection Science, 26*(1), 45-63. doi:10.1080/09540091.2014.885279

Yamamoto, K., Koakutsu, S., Okamoto, T., & Hirata, H. (2011). Fast backpropagation learning using optimization of learning rate for pulsed neural networks. *Electronics & Communications In Japan, 94*(7), 27-34. doi:10.1002/ecj.10249