Advancing the area of machine learning using video games as a catalyst

Justin Carter
Game Engineering
Newcastle University
United Kingdom
carterjustin67@gmail.com

***ABSTRACT* – This paper will explore the current applications of machine learning within the video games industry by investigating areas of machine learning in relation to video games. Rather than Artificial Intelligence (AI) intended as part of the game itself, focus will be given to AI designed to compete against the top echelons of a competitive game's professional scene. Current implementations include OpenAI Five and DeepMind's AlphaStar. The overall aim of this paper is to deduce whether and to what extent the field of machine learning can be advanced through use in the games industry.**

***Keywords* – Machine learning, artificial intelligence, neural network, deep learning, reinforcement learning, supervised learning, unsupervised learning, convolutional, recurrent, feed-forward, LSTM neuroevolution, OpenAI Five, DeepMind, AlphaStar**

I. INTRODUCTION

Video games are a unique area that reflect the chaos and unpredictability of the real world, unlike board games like Go or Chess which have a set number of rules with players taking actions in turns. As such, video games are an ideal environment to experiment and research machine learning in hopes of rapidly improving the field and developing new applications that are even useful in real life [4]

This section will aim to identify and explain machine learning, as well as many of the areas within it. The aim is to allow the reader to gain a solid foundation in the area of machine learning related to video games so that we can understand the topics covered later on. Thus the complex mathematics behind the subject will not be covered.

A. Machine Learning

Machine learning is much like how it sounds, the ability for AI to be trained and become better at performing certain tasks over time. How this is done will be explained over the next few pages.

1) Traditional AI vs machine learning

The main difference that separates AI that employ machine learning from traditional AI is the way they go about executing tasks given to them by humans. Traditional AI will provide an output of answers when given inputs of rules and data [1]. Using the example of a calculator, the data would be numbers taken from the user and what operations to perform on those numbers, the rules would be a program defining how the operations are executed, and the answers would simply be the results of performing the operation on the two numbers.

On the other hand, AI that follow the rules of machine learning will instead form the rules themselves, using inputs of data and expected answers, as can be seen in figure 1. Rather than a human programming the rules used to produce answers, a machine learning AI will be trained over time and produce the rules itself, as can be seen in figure 1 [1].

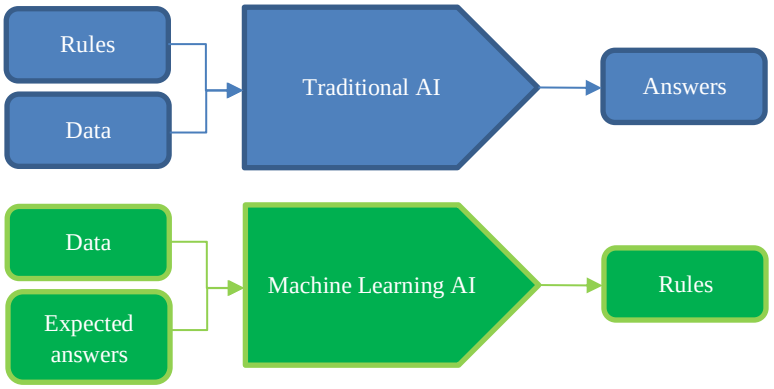

Fig. 1. Diagrams comparing the inputs and outputs of traditional AI with machine learning AI [1].

For an example related to video games; given a current position, destination and some obstacles, the input data would be videos of human players reaching the destination while avoiding obstacles and the expected result would be for the agent to travel to the destination as quickly as possible. Through training over time, the AI will work out which routes are faster and form a set of rules. An impressive trait that machine learning AI possess is that after being sufficiently trained, they can then be given new data and use the rules they produced to provide brand new answers [1].

Although figure 1 is a useful way to identify how machine learning differs from traditional AI, it is also missing a key aspect. This is the extra 'feedback' input that is used to deduce how close the AI is to achieving the expected output from its current output [1]. By comparing against this feedback, the AI can determine whether it is close enough to the desired outcome. For example, building upon the example above, the feedback could be the

amount of time it should take the agent to reach the destination having avoided all obstacles.

2) Representations and Hypotheses

For the AI to successfully produce desired results from input data, it is necessary to initially transform these data sets into a format more suitable for the task at hand, formatted data sets that are known as representations. By initially forming a representation that is more suitable to the task being performed, the AI will have a much easier time producing accurate results [1]. Using our current ongoing example of an agent reaching its destination as fast as possible. If we say that the pathways are a much lighter colour than the obstacles, and if the input data is the colour pixels taken from a picture of the screen, then a useful representation would be to encode the image into a grayscale format (as opposed to RGB), thus easily separating the traversable area from obstacles.

Rather than the AI deciding how to convert data into more useful representations by itself, it will search through a bank of stored methods, called a hypothesis space, a well-known example of which is a neural network, which will be covered throughout this report [1]. Within the hypothesis space, multiple hypotheses are contained. Each hypothesis describes how to transform the input data into a representation, and the job of the AI is to search through the hypothesis space, and identify the best possible hypothesis for the task at hand, while using the feedback to gradually improve the selection process [9]. This entire process of training to select better hypotheses is where the learning part of machine learning comes from [1][9].

B. Deep Learning

Deep learning is an area within machine learning that takes the use of representations one step further. Its approach is to have several 'layers' of representations, with each layer becoming more and more accurate in its representation of the input data in regard to the task at hand. As mentioned above, a neural network is one of the most heard of hypothesis spaces and is also the main model used in deep learning.

Although the area of deep learning covers any model that implements multiple layers of representations, models that feature only a few layers are often referred to as shallow learning, whereas deep learning models will feature in the tens or hundreds [1]. As opposed to traditional and shallow machine learning, deep learning has a clear-cut advantage due to their vast depth in layers. Models that are only a few layers deep will have great difficulty forming representations that are ideal for the task at hand, especially when said task is complex. To solve this problem, a human would have to delve in and create better layers manually, a process called feature engineering. On the other hand, in deep learning, feature engineering is not necessary as all the features can be learned at the same time, rather than having to specify the good ones [1]. This is exactly what makes deep learning the best possible option to use in the field of video games, as in a game, there are hundreds if not thousands of possible actions a player can take at a given point in time, thus the need for more specialised representations is necessary.

C. Neural Networks

A neural network consists of groups of nodes, with each node representing a layer. Each node takes inputs from other nodes, while also sending outputs to other nodes, with some of these inputs and outputs being from outside the network. Nodes connect to each other through 'links', each of which is associated with a 'weight' which is the main method that the network learns [7]. The links between nodes are decided by the human researcher/programmer who will usually have enough experience to deduce the best structure [12].

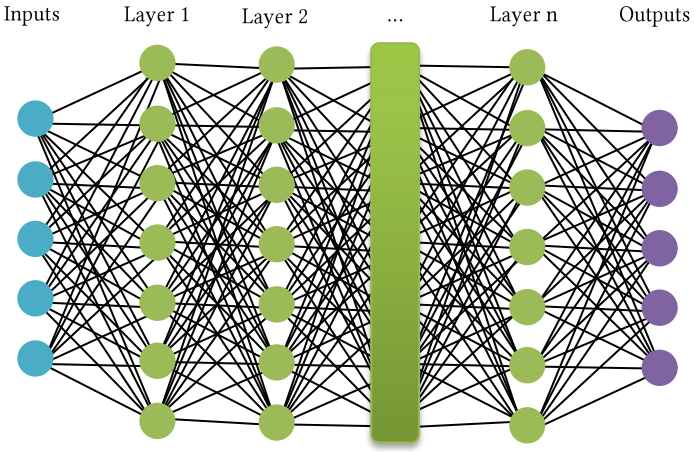

Fig. 2. Depiction of a deep learning neural network

The weight of each link explains the transformation that will be performed on the input data. The network will begin with random weights; thus, the transformations being performed will also be random, resulting in inevitably inaccurate outputs. However, as the network is trained over time, the weights of each node will be updated time and time again to slowly increase the accuracy of the output. Through using the expected output, as explained earlier. After each iteration of the network, if we compare the output against the expected output and calculate the difference between them, we can then adjust the weights slightly in the hopes of increasing the networks accuracy in the next iteration. This process is performed by an algorithm called Backpropagation and is the main source of "learning" in neural networks. Upon completing an iteration of the network and having calculated the difference between actual and expected outcome, this "feedback" is sent backwards through the network by the Backpropagation algorithm to update the weights of each node [1].

Each node in a network also has something called an activation level, with each one being calculated by an activation function. In short, the activation function is the controller of the output of a node. Once a node has taken input, and formed an output, it will send that output to the activation function. The activation function is then in charge of deciding what to do with the output, which could include deciding what nodes in the next layer to send

the output to, or, transforming the output into a better format [11].

Different types of neural networks are suited for different tasks, but an ideal neural network can be formed using a bespoke combination. The following explains the most common types of neural network used in video games.

1) Feed-forward

The nodes of a feed-forward network follow the rule that they may only output to a node in the next deepest layer. The option of links in the direction of the same or a shallower layer are ruled out. In addition, links cannot skip layers. This structure ensures that no cycles occur in the network, with a cycle being a node inputting back into a node it has taken input from. Of course, the disadvantage to such a model is that the network will have no knowledge of previous game states, which is almost mandatory for complex video games [7].

2) Convolutional

Convolutional Neural Networks are specialised in the transformations of one, two and three-dimensional image data, but mainly two-dimensional (2d games). The name convolutional stems from its main operation; convolution. This is the process of comparing the input data (pixels) against a subset of this data (filter) in the hopes of successfully finding a match. By comparing the filter against every part of the input image, it will be possible to identify a specific characteristic, such as an enemy or a projectile [10].

3) Recurrent

Similar to feedforward networks, but in this case the output can be fed back as an input. This allows the network to become aware of what the output was for previous captures of the state of a game, and thus will aid greatly in forming a more accurate output/prediction. For example, if when comparing the current game state to a previous state it is found that an enemy is moving closer to the agent, the agent can make better predictions about that enemies intentions, e.g. the likelihood of them attacking is higher so maybe we should retreat. Or, if playing a MOBA such as Dota 2 and an enemy has only just left our range of vision, the current game state will tell the network that no enemies are nearby, however the previous game state can prove otherwise [6]. Although recurrent networks certainly have advantage over feed-forward in terms having "memory", the cycles that are present also cause them to take much longer to produce stable outputs due to being less orderly. However, considering its disadvantages, the pros of recurrent learning are simply too good to pass up, especially for video game AI, where the game is in a constant state of chaos and unpredictability [7].

Recurrent learning is not perfect, and in fact has a flaw which threatens to undermine its effectiveness entirely; the fact that the agent's "memory" of previous iterations will gradually be forgotten. This problem is known as the vanishing gradient problem and is a flaw in the Backpropagation algorithm. The solution to this problem is known as Long Short-Term Memory (LSTM) and essentially allows the information at any point in the iteration of a network to be saved and accessed by a later iteration at the same place [1]. LSTM is used widely when developing deep learning agents for complex video games.

D. Learning

Now that the models that form the foundations of a deep learning AI have been explained, we just need to investigate the methods used to iterate through these models, and actually train the AI to improve over time. As the performance of a network is mainly determined by the weights if its links, approaches to learning all seek to improve these weights, albeit through different methods [12].

1) Supervised Learning

Supervised learning algorithms are a common method and are very similar to how the basics of machine learning works, as explained above. The agent will receive a certain input and try to form an output as accurately as it can. The actual output will then be compared against the expected output and the difference between the two will be used to update the model, improving its performance in the next iteration. When the agent has been sufficiently trained, and produces accurate outputs every iteration, it can be given inputs for which the expected output is unknown [6][7].

2) Unsupervised Learning

In contrast to supervised learning, unsupervised learning does not have any expected output that it can compare the actual output with. Due to this, the agent will have no way to train itself as the there is no way for it to determine how accurate its outputs are. Rather than improving an agent's accuracy of output, the purpose of unsupervised learning is to find patterns in the data so that similar data can be grouped together and thus increase the overall efficiency of the agent [6][7].

3) Reinforcement Learning

Reinforcement learning is similar to supervised learning in the fact that the agent can slowly train to produce more accurate outputs, however the method is slightly different. Reinforcement learning does not have any expected output that it can compare to its actual output, it instead receives an evaluation of its output, known as a reinforcement. By measuring the reinforcement after an action has been performed, the agent can deduce whether this action was good or bad, and then make changes accordingly [7]. This approach is especially useful in video games, as reinforcements are essentially already built in, through score, lives, currency etc. For example, upon performing an action, if the agent receives a large influx of score, the action can be considered as good and should likely be performed again when in a similar situation. In contrast, if the agent lost a life as a result, that action can be considered as bad and should not be performed again. Of course, not all games have such a system that can be used to provide the agent with reinforcements. Such an example would be an open world game, as there is often no clear objective for the player to complete, and thus no reinforcement to provide the agent [6].

4) Hierarchical Reinforcement Learning

Hierarchical reinforcement learning breaks down a normal reinforcement learning task into many smaller sub-tasks. Not all tasks are on the same level, some may be parents whereas others might be children, forming a hierarchy of tasks instead of one huge one. Breaking down tasks like this helps to reduce computational complexity while also allowing for subtasks to become reusable [19].

5) Neuroevolution

Neuroevolution is also method in training a neural network up to the stage it can be considered as complete, but its approach is very different [8]. Rather than having one network slowly teach itself how to improve through trial and error, Neuroevolution takes a Darwinist approach and forms a trained neural network through natural selection. The method of producing a fully trained neural network would likely begin with a pool of tens, if not hundreds of neural networks with random weights. Each network is given the same task, and then given feedback based on how their actual output compares to the expected output, this feedback is known as its fitness. Using this fitness, a sub-set of the original pool will be selected, with the rest being discarded. This sub-set will act as the "parents" of a new generation of networks by having their weights altered to improve their performance. This process will repeat itself until we are left with some highly trained networks, as can be seen in figure 3 [12].

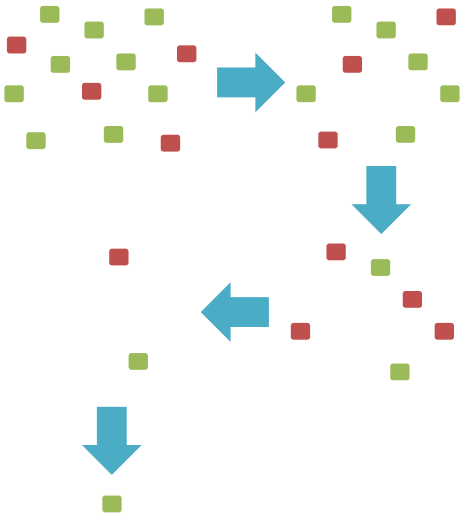

Fig. 3. Example of producing a trained network through Neuroevolution

Initially, we have a large group of networks, as seen in the top left. From these networks, the ones that perform the weakest (red) will be weeded out. The remnants will have their weights updated before the process begins again. As can be seen, we are left with a single, highly trained network.

As with normal methods of teaching networks, the base approach of neuroevolution does not allow the overall structure of nodes and links to change. However, an area within neuroevolution allows for exactly this behaviour; topology and weight evolving ANNs (TWEANNs). Not only does this approach allow the weights to be altered through successive generations of networks, the links between nodes can also be altered, even brand new nodes and links can be added. This approach allows for yet another layer of learning on top of altering weights, although the approach has its share of issues, one of the most prominent being the difficulty of combining two networks to produce a stronger one. The reason being, that although both parent networks could have outputs of very similar quality, they would likely be very different in terms of the layout of their nodes and links, as well as the values of their weights. Thus, deducing which network is best would be very challenging, a problem known as the competing conventions problem. A solution to this problem emerged, which eliminates many of TWEANNs issues; Neuroevolution of Augmenting Topologies (NEAT) [12].

In nature, genes contain information which explains which gene matches up with the same gene from a different animal, even if those genes are located in two completely different layouts. NEAT follows this example by employing Historical Markings. Each link in the network has one of these markings, which is just a number which can be used to identify that individual link. Each marking is unique and chronological through the use of a "global innovation number". Each time a new link between two nodes is created, the global innovation number is incremented, and used as the historical marking. Links in all networks are also kept track of, and in the event that a brand-new link has already been seen elsewhere, it is given the same innovation number [13].

When combing two networks, it is now easy to decide which links between nodes should be chosen for the new one. For links that are the same, the new link is chosen at random, from either of the parents. Links that are present in one parent but not the other are known either as disjoint or excess. Disjoint links have innovation numbers which are within the range of the other network's innovation numbers. On the other hand, excess links are outside of that range. In forming the child network from the two parents, disjoint and excess links are chosen only from the parent with better performance. If in the case that both parents have equal performance, links are chosen at random [13].

II. RELATED WORK

Now that a suitable foundation of deep learning has been established, this next section will identify some of the current applications of deep learning agents in the games industry.

A. OpenAI Five

Through the use of LSTM recurrent neural networks trained through reinforcement learning, resulting in the equivalent of 180 years' worth of games daily, the team at OpenAI have produced great achievements in the field of video game playing AI. To be more specific, OpenAI Five consists of five highly trained neural networks, which together play the video game Dota 2 as a team. Dota 2 is one of the most prevalent games within the MOBA genre and has a thriving esports scene. In addition, the game is exceedingly complex, with a multitude of characters (heroes) the

player can choose from for a single match, each of which having various abilities, there truly is enormous amount of possible combinations that a team could consist of. Since the possible actions that can be executed are different in each individual hero, OpenAI Five uses separate LSTMs for each hero. Upon doing so, it can become familiar with heroes on a level which rivals the top human players. For example, many heroes will have various abilities that will complement each other when used in different timings. Not only does OpenAI Five know how to successfully execute multiple abilities in quick succession to produce the best possible outcome, it can execute these abilities in perfect timing. Upon selecting its next action, OpenAI Five is able to delay by up to 4 frames. With the game running at 30 frames per second, this allows for actions to often be executed at the perfect moment, an ability hard to find in human players, if at all [4].

OpenAI stress that their approach has been as general as possible, meaning that they have hopes for their research to have applications outside of video games [4].

1) Playing Dota 2

Dota 2 is not a simple game. As explained above, there are many heroes to choose from, with each hero possessing many completely unique abilities. During a match there are a huge amount of possible actions the agent can take at a single moment, including moving to a location or attacking an enemy with an ability. Per hero, there is a total of roughly 170,000 actions that can be taken, of course, many of these possible actions cannot be executed, due to restrictions of the game, e.g. ability cooldowns or being restrained by the ability of an enemy. However, on average, there are still roughly 8000 – 80,000 valid actions the agent can take, whereas in comparison, for a board game like chess, there is merely 35 [4][17]. Additional comparisons between Dota 2 and board games can be seen in figure 4.

Statistic	Dota 2	Go	Chess
Valid actions	1,000	250	35
Moves per match	20,000	150	40
Number of values to represent game state	20,000	400	70

Fig. 4. Comparing Dota 2 with Go and Chess

Just from this comparison shown in figure 4, it can be seen just how much more complex a video game is compared to a board game. In addition, board games like Go and Chess are completely static in their respective rule sets. In contrast, video games like Dota 2 are updated on a regular basis, not only introducing brand new heroes, but also altering the values and characteristics of existing items to help balance the game, keeping it fair. Some such slight changes may render previous strategies as useless.

Of course, the fact that the game is constantly updated with brand new features has a significant impact on the agent. When a brand new feature appears, the agent will have no previous experience with it, this lead to the team at OpenAI developing a system to automatically alter the properties of the agent's network to account for these new additions to the game. Rather than allowing the agent to blindly attempt to adapt to these changes, OpenAI solve the problem through a system they call Network Surgery. The cost of training networks grows over time, and so when necessary to begin again in a new network for whatever reason, starting over completely from scratch would be a huge waste of time and resources. In the area of Machine Learning, this problem is usually circumvented by the weights of the network manually being transferred over, and although still uses a lot of resources, is the lesser of two evils. This is where Network Surgery comes in. The team at OpenAI developed a method for automatically selecting appropriate trained weights to be transferred over to a new network with 98.68% of weights being transferred automatically, saving a tremendous amount of time and resources [15]. Through network surgery, a new model can be produced from the current model, with the new model being compatible with the new features after an update to the game. Training then proceeds with the new features in place [17].

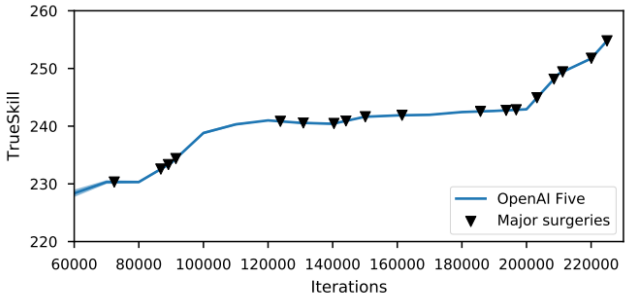

Fig. 5. Progress of OpenAI Five including all major surgeries [15]

The overall success of OpenAI Five is a statement towards the usefulness of Network Surgeries, especially when considering the frequent updates, a video game like Dota 2 experiences. Without the use of surgeries, the progress of OpenAI Five would surely be much slower. Below, figure 5 shows the overall progression of OpenAI Five, along with every major surgery and with the agents "TrueSkill"; a rating of its proficiency in the game.

A feature not seen in many games is that of the "fog of war" which limits the player's field of vision. Although there are things that can provide the player with some vision, such as minions and towers, many areas of the game map will have no vision, and thus even if there is an enemy headed in their direction, the player will have no way of knowing. The OpenAI 5 agent accounts for this by making assumptions based on partial information, much in the same way a human would [4]. For example, if a low health fleeing enemy escapes from the agent into the fog of war, the agent will have to make an informed decision as to the direction the enemy went. Not only does the agent make small predictions like this, it also plans its strategies in the long-term, in order to win the game [16].

In order to produce an agent capable of successfully planning for long-term situations, most previous agents would employ Hierarchical Reinforcement Learning. OpenAI Five however were trained to have such a high knowledge of the game and its features that it could adapt to situations it had never experienced before, allowing for future planning 30-90 seconds ahead of time [16].

2) OpenAI's Approach

As stated earlier, OpenAI used reinforcement learning to train their agent, and thus no expected outputs were provided, the agent would have started with random weights, and given reinforcements to refine those weights. The reinforcements that OpenAI provide the agent would also be used by a human player to determine how well they are doing in a single match. These reinforcements include both positive and negative types, so it is relatively easy to provide the agent feedback on their decisions [4]. In order to provide the team of agents with a high chance of winning the game, the team at OpenAI prioritised the reinforcements that could do so. For example, things like destroying towers and winning team fights would have higher weights than singular player kills and farming minions. This allows the agents to focus on more important aspects that are more likely to help out in the long term [16].

The training of OpenAI Five begins with a neural network composed with random weights and trains against itself. In the first few games, the AI will walk around the map without purpose. After a few hours, they will start to learn basic concepts necessary to play the game successfully, such as killing 'creeps' in one of the lanes to receive money. After several days of experience the agent will have learned to steal resources from the territory of their opponents as well as other more advanced strategies. A fully trained agent will even be able to perform well in team fights (when both teams will group together to fight each other, usually over highly valuable resources or to defend/attack vital point such as towers) [4].

In order to allow a single agent to work together with its 4 teammates, OpenAI use a parameter called "team spirit" which each individual agent possesses. Team spirit has a value from 0 to 1 which, in short, indicates how much attention the agent should be paying to aiding its team, rather than itself. Similar to the weights of each link in the network, team spirit is also trained over time [4].

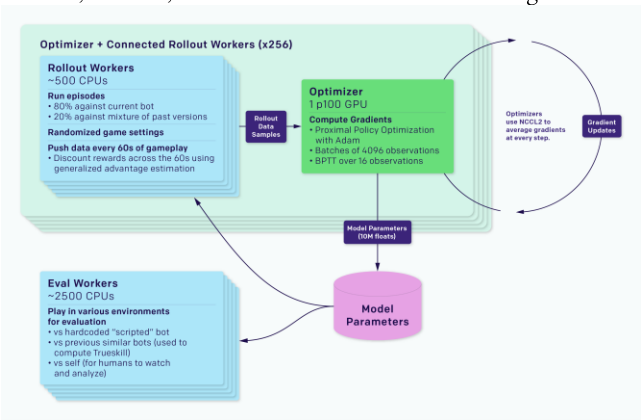

Fig. 6. OpenAI's general-purpose reinforcement learning training system – Rapid [4]

To train their networks, OpenAI use a "general-purpose reinforcement learning training system" called Rapid. As can be seen in figure 6, Rapid uses two different types of 'workers'. The first are rollout workers, who run an agent playing the game, and having its weights updated by the optimiser. The Evaluation workers test the trained agent against other AI, including scripted bots, previous less trained versions of itself, and current version of itself.

3) Observations

After testing the performance of OpenAI Five against multiple teams, including even semi-pro and pro teams, OpenAI were able to make several observations of their agents. Throughout playtests, OpenAI Five would repeatedly sacrifice its safe lane in order to gain full control of the enemy's safe lane (the safe lane for each side being the lane with its outer tower farthest away from the base). In gaining full control of the enemy's safe lane, the AI team can force fights to occur on their safe lane, at which the opposing team will have the disadvantage. At one of the playtests, a professional player Blitz had said that he had only learned this strategy after being told by another professional team, and after 8 years of experience:

> "The teamwork aspect of the bot was just overwhelming. It feels like five selfless players that know a good general strategy" – Blitz [4]

Not only does a professional with almost a decade of experience compliment the agent's strategy, he also compliments their teamwork. From this we can deduce that OpenAI's approach with the team spirit parameter was justified [4].

Blitz's judgment of OpenAI Five's teamwork was also justified, as the agents would start making aggressive plays to gain momentum before their human opponents. Such plays include ambushing opponents when they overextend their lanes (push too close to the enemy's tower, leaving a lot of room for ambushes from behind) and also taking the enemy's towers, allowing them to push their lanes further and gain territory. OpenAI Five even employed new strategies not seen in the current meta (known best strategies), showing that the AI has even gone beyond the level of its human counterparts [4].

Following numerous play tests against multiple pro teams, gaining back to back wins in each and having upped the training capabilities to allow an average of 250 years of in-game experience daily (from 180), OpenAI Five went on to challenge the Dota 2 world champions, and won back to back. This event marked the first time an AI was able to beat the world champions in any game. This is also a huge occasion in the whole field of AI, as an AI was able to overcome the abilities of the best humans of a certain field [2][3]. Like previous events, OpenAI were able to make a few more observations, the most

prominent of which, was OpenAI Five's ability to play on the same team as humans, and successfully work together, with the agents even going as far as to sacrifice themselves to save their human teammates. OpenAI Five was originally trained to work with four other instances of itself as teammates, and in order to play alongside humans, the team at OpenAI had assumed that extra training would be required [2]. The ability to play alongside humans is mostly made possible by OpenAI's strides in long-term planning, and highlights its potential for use in other fields [16].

Overall, OpenAI as well as other AI enthusiasts are impressed by OpenAI Five's performance, especially its ability to work with human teammates without any additional training. Due to OpenAI keeping their training methods very general, their accomplishments foreshadow the true potential of deep learning AI as a whole, without being restricted to video games.

B. DeepMind's AlphaStar

Similar to OpenAI, the team at DeepMind decided to tackle the challenge of teaching a deep learning AI to play an extremely complex video game. However, this time, instead of Dota 2, the game at hand is Star Craft 2.

1) Playing StarCraft 2

StarCraft 2 is a complex real-time strategy (RTS) game where two players face off against each other. Although there are much less players compared to a game of Dota 2, StarCraft 2 is no less complex. During a match, the aim of players is to gather resources and construct buildings using a set of worker units and use these buildings to train even more advanced units who can in turn craft more advanced buildings, with the idea of eventually having enough strong units to destroy the other player's base. Although this may sound simple at first, high level game play involves the micro-management of their units, buildings and economy while thinking about their overall strategy to eventually win the game in the long-term. This is a game where even the smallest mistake would allow the opponent to gain the upper hand [5].

StarCraft 2 is a video game that has been around for about ten years, and if the original game is counted, there are a huge number of people that have played and are still playing the game at a very high level. As such, there will have been countless strategies deduced in how to play the game to eventually win [5].

Similar to Dota 2, StarCraft 2 has the "fog of war" feature, only allowing players to see the areas nearby their buildings and units. Due to this it is difficult to find out what the enemies is doing, and what exact strategy they could be employing to win the game. As such it is necessary for the AI to predict many things [5]. Although the two games have similarities, it should be noted that StarCraft 2 is much more complex, at least in the sense of possible actions. As explained earlier, OpenAI Five has roughly 8000 – 80,000 valid actions it can take at any point in time, many more compared to board games such as chess or go. In contrast, AlphaStar's valid actions number up to 10^{26}, a truly huge amount [14][17].

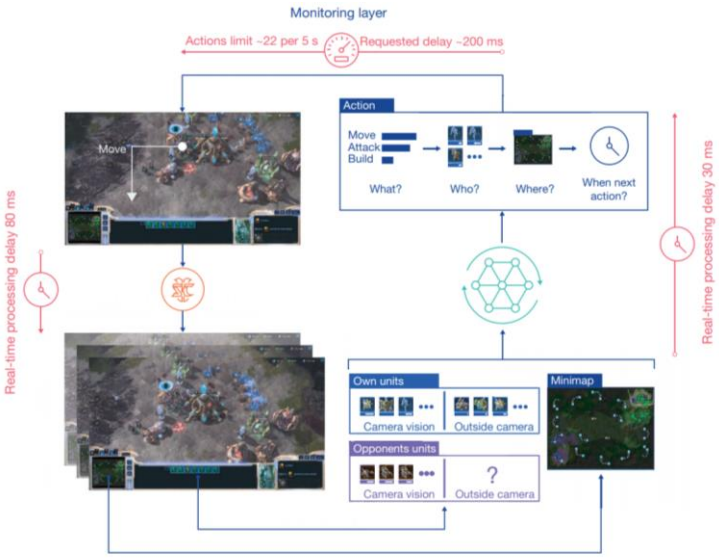

Fig. 7. How AlphaStar plays StarCraft 2 [18]

AlphaStar plays the game by gathering information from the game view, including friendly and opponent units, both inside and outside the camera view. From this information, AlphaStar can queue an action to commit, consisting of what to do, who to do it to, where to do it and when to do it, as can be seen in figure 7 [18].

After putting an action together, the monitoring layer decides whether or not these actions can actually be executed. By limiting the amount of actions that can be performed at any one time, AlphaStar can be kept within the bounds of what a human is capable of, making competitive play fairer [18].

2) DeepMind's Approach

Similar to OpenAI Five, DeepMind's approach with AlphaStar was to use a deep recurrent neural network with LSTM. However, in contrast, DeepMind employed two stages of training. The first stage was to use supervised learning to train the network by providing real anonymous matches played by humans. From these examples, AlphaStar was able to learn the basic strategies of the game in order to win matches and was even able to beat the strongest AI that the game itself had to offer [5].

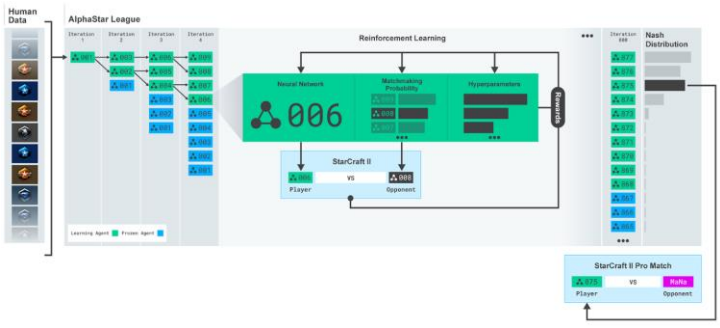

Fig. 8. DeepMind's league of agents used to train AlphaStar [5]

DeepMind's second approach to training is arguably the most interesting way to train a deep learning agent we've covered so far. Using the supervised learning taught network as a foundation, DeepMind formed a league of agents, with each agent being the child of a previous agent in the league, and with the original network taught by supervised learning as their ancestor, a system called multi-agent learning [18][5].
As can be seen in figure 8, agents continuously play matches against each other, while being trained using reinforcement learning. Each iteration of the league represents a stage with a set number of agents, and when progressing to the next stage, a selection of these agents will branch off, forming children who will then be trained during the next iteration. This approach can be said to combine the ideas of Neuroevolution with single network reinforcement learning [5].

After entering a new iteration, the training of previous agents is frozen, preventing them from learning new and better strategies while discarding old ones. Meaning that the strategies of iteration 1 will be less complex than iteration 2, who's strategies will also be less complex than iteration 3 and so on. This ensures that the newest generation will constantly be training against all kinds of strategies, ranging from the most old and basic to the newest and most complicated [5]. This approach solves the problem of the agent training to such a high level that it forgets how to deal with the original, simpler strategies [14].

DeepMind's league lasted for a total of 14 days, with each agent having gained up to 200 years of in-game experience through constant matches against each other. Of course, since the model is akin to Neuroevolution, it can be said that the experience of every agent contributed greatly to the final, fully trained version.

Through this unique method of training, DeepMind noticed a huge variety of strategies being developed amongst the agents, some very basic, but also some very complex. As the league progressed, many of the more basic strategies were scrapped, as they were too simple and easy to counter. Agents instead began to develop more and more complex strategies [5].

If agents were allowed to train aimlessly, their progression would likely stagnate, and it'd be likely for many agents to develop the same strategies. DeepMind's solution was to provide each agent with a unique objective. For example, one agent may be trying to beat a specific agent, whereas another might be trying to beat a range of other agents. This causes agents to train to beat specific agents, which would involve forming certain strategies which counter their opponent's strategies. This allows the training of each agent to be more targeted, and in the long run, provides more variety in the strategies developed [5].

DeepMind tested two different methods of providing AlphaStar information of the game state, the first being to receive raw data of the current positions and states of visible units. DeepMind later developed an input system in which the agent controls the camera in the way a human would, and only can receive information about units within the view of the camera. The prior has a clear advantage in the fact that it has no need to manually move the camera around the map when executing different kinds of tasks, which on average would happen 30 times a minute. As expected, the agent using raw input performed better, but the camera conrolling agent was not too far behind [5].

3) Observations

Like OpenAI Five, after training, AlphaStar was pitted against some of the most skilled and experienced players in StarCraft 2's professional scene. Initially, AlphaStar went up against TLO, a renowned professional player with roughly ten years of experience, and unexpectedly, AlphaStar ended up winning 5-0. TLO praised the agent's strategies greatly [5].

> "AlphaStar takes well-known strategies and turns them on their head. The agent demonstrated strategies I hadn't thought of before, which means there may still be new ways of playing the game that we haven't fully explored yet" – TLO [5]

TLO went further to say that AlphaStar has near perfect timing in attacking and retreating, yet the agent doesn't feel omnipotent. TLO believes that the tier AlphaStar has reached is attainable by humans [14].

Shortly after, AlphaStar went on to play against another professional player; MaNa, who again lost to the agent 5-0 [5].

> "I was impressed to see AlphaStar pull off advanced moves and different strategies across almost every game, using a very human style of gameplay I wouldn't have expected" – MaNa [5]

Similar to TLO, MaNa praises AlphaStar's unique strategies and even goes on to call its style of play very human, a great achievement in the broader area of AI let alone deep learning in video games [5].

The team at DeepMind have the same objective as OpenAI; to not simply limit their achievements in deep learning to video games, but to also contribute to the wider pursuit of truly intelligent AI which can be used in real world scenarios. Due to AlphaStar having the ability to make tens of thousands of intelligent decisions per game, with each of them contributing to an overall strategy, DeepMind predicts that future advancements will be fuelled by their achievements with AlphaStar.

III. EVALUATION

OpenAI Five and DeepMind's AlphaStar are some of the most prominent developments of machine learning in the games industry, and their accomplishments in games as complex as Dota 2 and StarCraft 2 speak for themselves. Although many deep learning agents have been developed for board games such as Go

and Chess, these examples are not fit for determining the potential for deep learning agents in day-to-day life. These games have fixed structures and rulesets, with the players taking turns to make their moves. StarCraft 2 and Dota 2 on the other are a much closer model to real life; they feature chaotic and unpredictable gameplay with all players acting at the same time and regular updates adding brand new heroes (in the case of Dota 2) and balance changes causing the need for agents to constantly adapt. Due to the nature of these games, the number of valid moves an agent can execute at any point in time is much higher than that of Go or Chess, and as such, are more suited to compare to a real world scenario.

The methodologies that OpenAI and Deepmind have come up with such as network surgery [15] and long-term planning [16] are not things that are limited to use within video games. These ideas most certainly can be applied for use in real world systems and their potential has been proved already.

The advantages that network surgery provides can be carried over directly to broader uses of machine learning. The method is generalised, and the environment is very similar for most cases, because as stated before, a video game mirrors the chaos and unpredictability of the real world. As can be understood, with real world situations changing constantly, network surgery would increase the efficiency of training an agent exponentially [15].

> "We believe the presented techniques can generalize to other large scale RL agents that operate in a semantically rich environment, extracting knowledge about plans in situations where hundreds to thousands of actions are required to accomplish a task" [16]

The quote above, taken from a paper published by OpenAI themselves clearly states the potential that long-term planning presents to the wider area of machine learning. Additionally, the methods that OpenAI themselves used can be generalised for agents in similar environments. OpenAI go further to suggest that the long-term planning strides they have made can be furthered to allow collaboration between AI and humans even outside of video games. As mentioned earlier, at one of the OpenAI playtests, humans successfully played alongside OpenAI Five agents, further proving potential for future applications [16].

Competitive video games also have huge player bases, with ranking system which can easily determine the skill level of each player. This is especially useful in this scenario as it is easy to compare the progress of an agent against its real human counterparts.

The teams behind OpenAI Five and AlphaStar also approached their networks and training methods in a very general sense, meaning that their work can be carried over into areas useful in real life scenarios.

IV. CONCLUSION

In conclusion, many AI systems with jobs to perform specific tasks are prone to making mistakes. Deep learning is a solution to not only preventing these mistakes, but also to take it a step further by completing tasks as efficiently and accurately as possible and even being able to successfully predict scenarios at a level far beyond human capabilities. The research and application of deep learning in complex video games can make strides where alternatives in board-games and robotics fail. Through developing methods to better train deep learning agents to play video games with a more general approach, we simultaneously learn new ways to train deep learning agents for practical use in the real-world, while refining the methods and techniques we already have.

V. FUTURE WORK

Suggested further work includes additional experimentation in deep learning within video games with generalised approaches. Experimenting with completely new methods and techniques will also be useful, as video games provide the perfect testing environment and will surely spur advances in the wider machine learning field.

Of course, taking the methods and techniques such as network surgery and long-term planning and using them in wider fields such as meteorology or speech recognition would not only prove their usefulness outside of video games, but show how video games can successfully be used as a training ground for developing the field of deep learning.

As shown by OpenAI, the potential for cooperation between deep learning agents and humans is great, with many possible applications in the real world. There are countless tasks which include subtasks which should be carried out by humans, whereas others should be automated. The ability for OpenAI Five to work together with humans was not even intentional, yet the result is astonishing. The prospective future for an agent built from the ground up with the goal of cooperating with humans is astonishing.

REFERENCES

[1] Francois Chollet, 2018. Deep learning with Python, (June, 2020), 3-11. worldcat: https://www.worldcat.org/title/deep-learning-with-python/oclc/1019988472.

[2] OpenAI, 2019. OpenAI Five Defeats Dota 2 World Champions, (June, 2020), openai: https://openai.com/blog/openai-five-defeats-dota-2-world-champions/.

[3] OpenAI, 2019. OpenAI Five, (June, 2020), openai: https://openai.com/projects/five/.

[4] OpenAI, 2018. OpenAI Five, (June, 2020), openai: https://openai.com/blog/openai-five/.

[5] The AlphaStar team, 2019. AlphaStar: Mastering the Real-Time Strategy Game StarCraft 2, (June, 2020), deepmind: https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii.

[6] Niels Justesen, Philip Bontrager, Julian Togelius, Sebastian Risi, 2017. Deep learning for video game playing, (June, 2020), 1-2. arxiv: https://arxiv.org/abs/1708.07902.

[7] Stuart J Russell, Ernest Davis, Peter Norvig, 2015. Artificial Intelligence : a modern approach, (June, 2020), 567-573. worldcat: https://www.worldcat.org/title/artificial-intelligence-a-modern-approach/oclc/928841872.

[8] Felipe Petroski Such, Vashisht Madhaven, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, Jeff Clune, 2017. Deep Neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, (June, 2020), 1-2. arxiv: https://arxiv.org/abs/1712.06567.

[9] Jason Brownlee, 2019. What is a hypothesis in machine learning, (June, 2020), machinelearningmastery: https://machinelearningmastery.com/what-is-a-hypothesis-in-machine-learning/.

[10] Jason Brownlee, 2019. How do convolutional layers work in deep learning neural networks, (June, 2020), machinelearningmastery: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/.

[11] missinglinkai. 7 types of neural network activation functions: how to choose, (June, 2020), missinglinkai: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/.

[12] Kenneth O. Stanley, 2017. Neuroevolution: A different kind of deep learning, (June, 2020). oreilly: https://www.oreilly.com/radar/neuroevolution-a-different-kind-of-deep-learning/.

[13] Kenneth O. Stanley, Risto Miikkulainen, 2006. Evolving neural networks through augmenting topologies, (June, 2020), 107-110. mitpressjournals: https://www.mitpressjournals.org/doi/pdf/10.1162/106365602320169811.

[14] The AlphaStar team, 2019. AlphaStar: Grandmaster levelin StarCraft 2 using multi-agent reinforcement learning, (June, 2020). deepmind: https://deepmind.com/blog/article/AlphaStar-Grandmaster-level-in-StarCraft-II-using-multi-agent-reinforcement-learning.

[15] Jonathan Raiman, Susan Zhang, Christy Dennison, 2020. Neural Network Surgery with Sets, (June, 2020). arxiv: https://arxiv.org/pdf/1912.06719.pdf.

[16] Jonathan Raiman, Susan Zhang, Filip Wolski, 2019. Long-Term Planning and Situational Awareness in OpenAI Five, (June, 2020). arxiv: https://arxiv.org/pdf/1912.06721.pdf.

[17] Christopher Berner, Greg Brockman, Brooke Chan, 2019. Dota 2 with Large Scale Deep Reinforcement Learning, (August, 2020). arxiv: https://arxiv.org/pdf/1912.06680.pdf

[18] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, 2019. Grandmaster level in StarCraft 2 using multi-agent reinforcement learning, (August, 2020). nature: https://www.nature.com/articles/s41586-019-1724-z

[19] Hengst B, 2011. Hierarchical Reinforcement Learning, (August, 2020). SpringerLink: https://doi.org/10.1007/978-0-387-30164-8_363