Vizdoom

Ethan Poe, Nolan Winsman, and Timothy Fields

***Abstract*— VizDoom is a program that allows the input of pixel data from the 1993 video game Doom. The revolutionary aspect of VizDoom is the fact that it relies on Pixel Data. Artificial Intelligence through pixel data in games has been done before with Atari games. The convolitional neural network we worked with for VizDoom is the one prepackaged with the windows precompiled install of VizDoom on Github. Our research and work was to implement rigorous data collection in order to understand how efficient the current convolutional neural network is at learning. Once a thorough understanding of how efficient the previous architecture was, We began implementation of a new convolutional neural network architecture. This new architecture obtained marginal to substantial improvements to the Doom agents capacity to learn human like behavior in gameplay.**

I. Introduction

VizDoom is an AI research platform created for the 1993 video game Doom. It comes with a a deep convolutional neural network that teaches an Agent to play different scenarios in Doom. Scenarios such as rooms filled with acid and health kits. Another scenario is where there the agent must move left and right to track and defeat a cacodemon. The convolutional neural network is decent for many scenarios but we wanted to make it far better. There were many interesting aspects of working on this project, many were foreseen and there were plenty that weren't. One of the things that we had not predicted was the massive time commitment that this project required. However, that being said, we were able to collect a massive amount of data and consistent data at that.

In this article we hope to fully elaborate upon the many interesting and intricate topics and challenges we faced during our time spent working on it.

II. Related Work

When we began pondering where to start on the project we immediately knew that we would need a sound structural base due to the sheer amount of information and knowledge that this project demands. That was when we discovered "VizDoom: A Doom-based AI research platform for visual reinforcement learning". [1]

Once we had a solid grasp of what we could/needed to achieve, we decided that we would need to look at some differing sources that could either give more information about our current pool of knowledge in VizDoom or something that we would be able to compare to in terms of a professional looking article. That is when we were fortunate enough to stumbled upon the article "Playing Atari With Deep Reinforcement Learning" [2]. While this was not another VizDoom article the incredible amount of raw information that was actually applicable to our project was stunning. Not only did we obtain references for structure within the paper we were able to apply many concepts both in application and mathematics.

With our understanding of structure and conceptual math beginning to broaden we knew that the next step was to begin to seek a deeper understanding of dueling networks as to continue to make progress in our research. By using the article titled "Dueling Network Architectures for Deep Reinforcement Learning" [3] we began to understand the concepts of deep reinforcement technologies and their connection to Vizdoom.

Now with all of our fundamental understandings of the concepts presented to us we moved onto trying to discover other sources that would be able to help either give us inspiration or potnetial goals to look toward. One of the sources that we discovered was a video running VizDoom experimenting with an assortment of learning techniques. "VizDoom Environment. AI Agent Is Playing Doom Using Reinforcement Learning Algorithms"[4], used algorithms like Deep Q-Learning, Deep Convolutional Q-Learning, and A3C. With this in mind and the video showing clear learning progress this gave us something to work towards as somewhat of an end goal.

One of the last aspects of creating an article that we would deem worthy of producing was refining and polishing it. Through the need to construct an article that was not only coherent but also intriguing and productive enough to be considered worth while we decided to look at other articles that we believed had achieved this concept in one way or another. Looking at "PLaying FPS Games With Environment-Aware Hierarchical Reinforcement Learning"[5], we found many note worthy aspects of the article that would help us to raise the level of our own article.

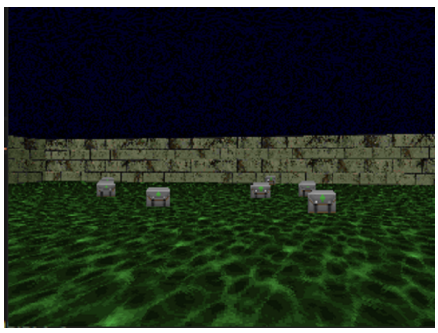
For a final resource we had decided to give some credit back to the relatively good amount of information that we were able to gather from Perusall's "Reinforcement Learning: An Introduction" by Richard Sutton and Andrew Barto [6]. This on top of the many other sources that we were able to find that helped incorporate many interesting concepts into our article, were incredibly useful. Without those that gave a strong base, those that helped our basic understanding and those that helped refine our work we would have gotten no where. It was through these, that we were able to expand upon the concept of VizDoom.

III. Methods

A large portion of this experiment involved adding code that tracks the data. The experimental procedure goes into more detail as to how this was done. There were two new

(a) Rocket Basic: Strafe to the Cacodemon and shoot it with a rocket

(b) Deadly Corridor: Survive to the end of the hall by shooting the enemies



(c) Health Gathering: Find health packs to survive the acid floor

Python files created to track data and aid the experimentation. There is load in data which is a python file that only contains the default data class. This class holds data for numerous variables such as total epochs for how many epochs are in a training session. Standard Convolutional neural network variables such as learning rate, discount factor and batch size. This class is also where many boolean variables are stored. The most important booleans are skip training and skip evaluation. To start a training session skip training is set to False and skip evaluation is set to True. For starting an evaluation the booleans should be the opposite of the values when training a scenario. Unfortunately in the current version of the program, it is not set to be able to start a scenario training session, then automatically evaluate that training once the training is complete.

Once we had gathered enough data to establish a baseline we then implemented a different reinforcement learning architecture. We used a variation of the Duel Q-network architecture with additional convolutional layers compared to the baseline architecture. We wanted to explore if these additional layers would help improve the accuracy due to the complex nature of the game. Additionally, we implemented batch normalization to try and offset some of the side effects of increasing the depth of the neural network. By standardizing the output of the previous layer we were able to ensure that the input to the next layer would have a significantly more predictable distribution. This normalization process helped to reduce the number of large variations and outliers in the data we collected.

The other Python file, graph-data.py is discussed more in section 5. Most of what it does is load in the evaluation results for multiple networks, take the average for those results, and plot the data using matplotlib. Numpy is used to calculate the average of the list that contains all the training data. It is important that the data is loaded into the lists correctly.

IV. Application Domain

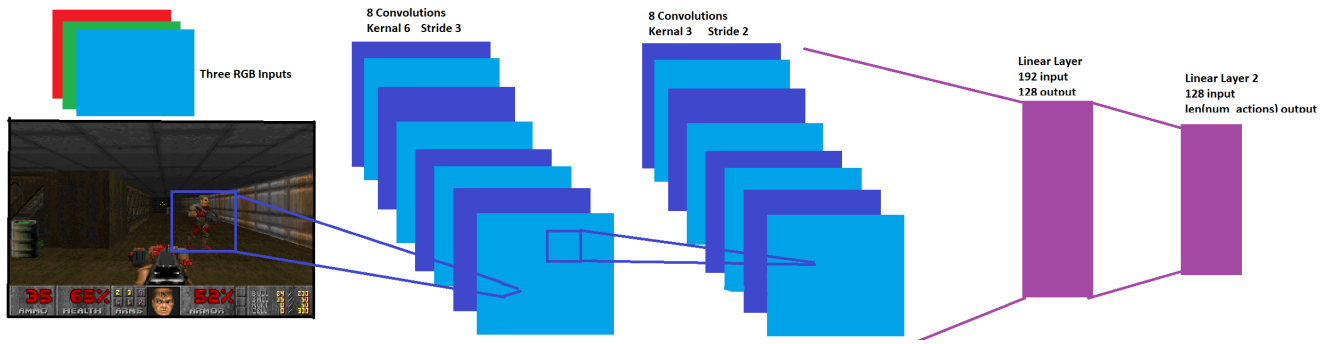
We chose to use doom as our domain because it offered a unique set of challenges while still remaining within a reasonable scope of what is achievable in a semester. The VizDoom code base is already well established and has very good documentation, and Doom is a much simpler FPS game than more modern titles. Some of the interesting features of doom include its pseudo 3D environment that allows the agent to move in three dimensions while also requiring it to aim in the 2D plane. We believe that it is possible to add additional features to this existing convolutional neural network in order to better identify enemy agents and thus improve the average performance of the agents in VizDoom.

When we started this project we used multiple different scenarios for training the agents in order to gather as much data as possible on the behavior in different environments. We ended up settling on three scenarios to focus on when evaluating the performance of the different reinforcement learning architectures.

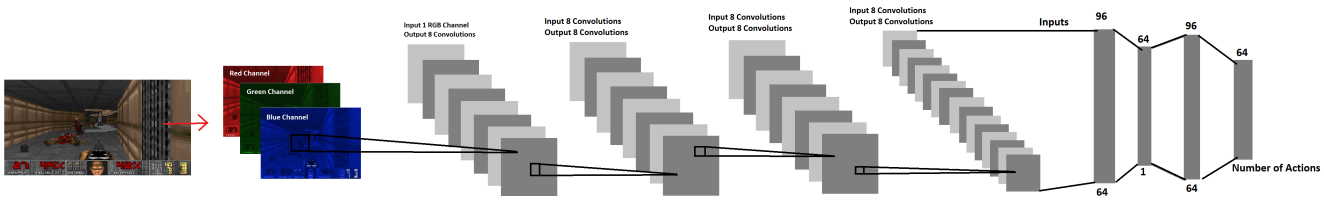
The first of these scenarios was Rocket Basic Figure 1a. We chose this scenario because it provided a solid baseline for us to be able to verify that the agent was evolving and improving. In this scenario the agent is trapped in a square room with a stationary enemy “demon” at a randomly generated position on the opposite wall. This scenario requires the agent to move left or right to reach a position where they can hit the demon with a rocket. In this scenario the agents would start off by moving and shooting randomly, but as the epochs continued the agents were able to identify the position of the demon and quickly move to a position to get an accurate shot.

In the Deadly Corridor scenario Figure 1b the agent gets a choice between two potential methods of reaching the end of the level. These choices are highly representative of the human “fight or flight” response. In this scenario the agent spans in a small semicircular room with a demon to the left and right and a corridor extending behind the two demons. This layout is effectively repeated two more times before reaching the exit. In this scenario it is technically possible for the agent to run between the two demons without dying but the margin for error is very small. Taking this approach allows the agent to ignore trying to shoot the demons and focus on evolving better movement strategies, however any error will likely result in death. The other approach is to shoot one or both of the demons before moving on to the next section of the corridor. This approach significantly reduces the importance of movement but is still potentially risky as it takes time to aim at the first demon which allows the other demon to potentially kill the agent.

We also chose to include the Health Gathering Figure 1c as it provided a very different set of challenges compared



(a) Original Architecture: Graphic of the Originl Architecture



(b) DuelQNet Architecture: Graphic of the New Architecture

Fig. 2: Architectures

to some of the more typical scenarios. In this scenario the agent is trapped inside of a large square room and the floor constantly inflicts damage. In order for the agent to stay alive it must move around the room to pick up health packs as they spawn on the ground. This provided a novel challenge for the agent as the use of weapons was not relevant and the focus was instead on moving quickly and decisively in order to pick up the health packs.

V. Experimental Procedure

To track the results a network is saved every $\lfloor epochs/4 \rfloor$ epochs and the first and last epoch. With most training sessions being 20 epochs this results in a network saved at epoch 1, 5, 10, 15, 20. After training for a particular scenario is complete it is viable for evaluation. Evaluations work by loading in each network for the saved results and running the network 50 times or 50 iterations of gameplay. All of these scores from the gameplay iterations are saved into a plain text file. Testing the performance of the two architectures with a certain scenario consits of a training cycle and an evaluation cycle. A training cyling is 5 training sessions. To clarify, at 20 epochs the training cycle is training from epoch 1 to 20, 5 times. After all the training and evaluating for a scenario, all the data is loaded into our graph-data.py which is a Python file that uses matplotlib to graph the data. This data will be shown in the results section.

VI. Results

Figure 2 shows the promise of the DuelQNet architecture. The graphs on the top row display 10 lines, 5 blue which represent the original architecture and 5 green which represent the new DuelQNet architecture. The scenario graphs on the second row, below the graphs just discussed, contain the same data from the above graphs. These graphs just take the average of the five lines to display the perfomance more concisely.

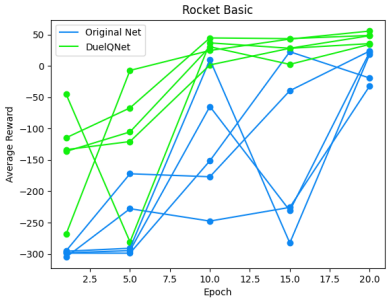
The Rocket basic scenario Figure 3a does marginally to substantially better with the DuelQNet architecture. The DuelQNet learns faster than the original architecture. Towards the ends of the training they are performing quite similarly. The Rocket Basic is one of the easiest scenarios. It is simple for a bot to learn to strafe left and right and track a Cacodemon.

The DuelQNet architecture does considerably better in every the Rocket Basic scenario and the Health Gathering scenario. It does marginally better in the Deadly Corridor scenario Figure 3b. This is just a difficult scenario to achieve optimality in. Artem Tkachev was able to achieve near optimality in this scenario so it is possible, just challenging [4].

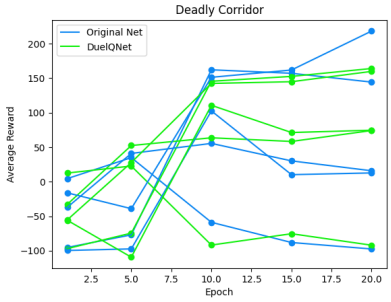
The Health Gathering scenario Figure 3c in particular did extraordinarily well. Part of this is due to the outlier with one of the networks getting an average score of almost 1000. Most of the training lines due dip at certain points.

VII. Discussion

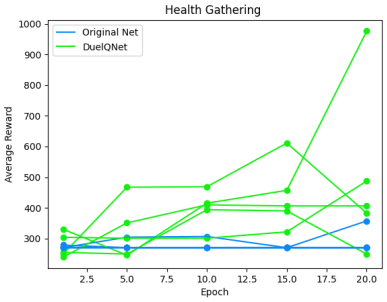
The promising results of the DuelQNet architecture displayed in the results section are very promising. While Deadly Corridor is only slightly more optimal with the DuelQNet architecture, the other scenarios tested perform substantially better with the implementation of the DuelQNet. The leading theory behind this is that the DuelQNet is better at motion and navigating the game space of Doom. It is competent at learning to detect objects like enemies and health kits. It is not competent enough for the scenario Deadly Corridor. The greatest challenge with this challenge is the enemy to the left with the shotgun. If this enemy to the left is not defeated quckly the Doom agent will be defeated



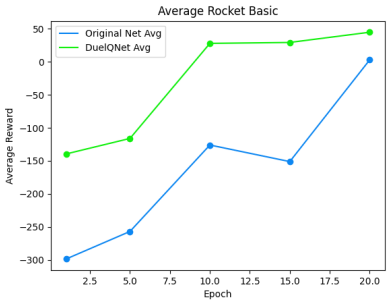
(a) Rocket Basic: Score of All Training Sessions



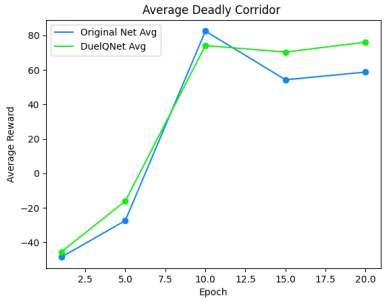
(b) Deadly Corridor: Score of All Training Sessions



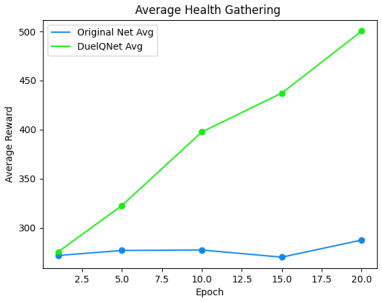
(c) Health Gathering: Score of All Training Sessions



(d) Rocket Basic: Average Score of All Training Sessions



(e) Deadly Corridor: Average Score of All Training Sessions



(f) Health Gathering: Average Score of All Training Sessions

Fig. 3: Scenario Results

with minimal opportunity for learning.

Another reason the DuelQNet Figure 2b performs better than the original architecture Figure 2a is due to several differences. Differences like more convolutional layers and a much deeper fourth convolutional layer. While we did not utilize the Dueling Q Network to its maximum potential, it was more than enough to improve performance in a variety of scenarios

There are certainly many aspects open to improvement in the future. One of the biggest things is the implementation of Nvidia's Cuda to run the training sessions on an Nvidia Graphics Card. The training sessions were slow running on a CPU. Experimentation would be much faster with Cuda implemented.

Another thing is maximizing the use of the DuelQNet. The architecture used in this project does not implement the splitting and re-converging of models which is inherent to Dueling Q Networks.

Lastly, while the project did implement a lot of automation, it lacks the ability to run a complete training cycle, saving the numerous networks that creates, and without user input, start the evaluation of the saved networks. Currently the user must start a training cycle and once that is complete, rename a few directories and variables to point to said directories. Once all of that is done an evaluation of the saved networks can commence.

VIII. Conclusion

VizDoom is a tremendous stepping stone in artificial intelligence based on pixel data input. It is a large step closer to human like simulations compared to Atari video games. Through the implementation of rigorous data collection to calculate how efficient a given convolutional neural network architecture is. The implementation of the DuelQNet greatly improved the performance in three distinct scenarios. By combining the new Q-net architecture and the batch normalization we were able to not only improve the results of the training across many different scenarios but also decrease the run-time of the simulations as a result of the agents learning faster from the more detailed and streamlined inputs.

References

[1] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "ViZDoom: A Doom-based AI research platform for visual reinforcement learning," September 2016. [Online]. Available: 10.1109/CIG.2016.7860433

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Reidmiller, "Playing Atari With Deep Reinforcement Learning," *Deepmind Technologies*, 2013. [Online]. Available: https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

[3] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," November 2015. [Online]. Available: https://arxiv.org/abs/1511.06581

[4] A. Tkachev, "VizDoom Environment. AI Agent Is Playing Doom Using Reinforcement Learning Algorithms," 2019. [Online]. Available: https://www.youtube.com/watch?v=OfyUqZ3tvU8

[5] J. Weng, S. Song, H. Su, D. Yan, H. Zou, and J. Zhu, "Playing FPS Games With Enviornment-Aware Hierarchical Reinforcement Learning," 2019. [Online]. Available: https://cdn.discordapp.com/attachments/835262038606479401/837069714370723840/0482.pdf

[6] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," 2014. [Online]. Available: https://app.perusall.com/courses/artificial-intelligence-121780934/suttonbartoiprlbook2nded

IX. Honor Code

We have acted with honesty and integrity in producing this work and are unaware of anyone who has not

-Nolan Winsman, Ethan Poe, Tim Fields