Deep Learning and the Prisoner's Dilemma: A Strategic Evaluation

Jaden Cohen & Brandon Holland

Received July 18, 2024
Accepted October 02, 2024
Electronic access October 31, 2024

This paper presents a high-level exploration of the behavior of Deep Learning models in complex environments through the lens of the Repeated Prisoner's Dilemma problem. The strategic acumen of multiple models including binary classification models, convolutional neural networks and recurrent neural networks are evaluated based on their performance in a Repeated Prisoner's Dilemma tournament. By evaluating this performance, it is found that the Deep Learning models studied lack the strategic abilities necessary to have success in dynamic situations. With the ubiquity of Deep Learning models in today's world, this paper serves as a warning against the use of such models in sufficiently complex situations – like the often convoluted and certainly dynamic real world.

Keywords: Artificial Intelligence (AI), Deep Learning (DL), Game Theory

Introduction

In today's world, DL models are taking increasingly prominent roles across dozens of fields. There are even indications that DL techniques could soon play a role in big government decisions [1]. Their prevalence, though, raises a question: how qualified are these models to assume such a prominent role in our world?

First, though, it is worth elaborating on what is meant by "DL models". This paper provides a cautionary tale describing the dangers of misinterpretation of "simple" DL – ranging from neural networks to full RNN's – as "black boxes" which can solve any problem. Reinforcement learning models for example would defeat the purpose of the paper in their subversion of this point and, with this in mind, this paper only considers only the "simple" DL, defined and addressed below.

For DL models to assume their current role, they must be able to understand their environments. The question of these models' strategic abilities is, therefore, an extremely important one and constitutes the root study of this paper: what can we learn about the general, strategic potential of DL based models in dynamic, "real world" scenarios by having them compete in a simulated dynamic environment (the Repeated Prisoner's Dilemma) and, by extension, how qualified are they to fill their prevalent role in our world?

Literature Review

There have been multiple studies published on topics similar to that of this paper in the past. Though this paper takes no inspiration from these studies, their work is summarized – in an extreme lack of detail – below.

The first and most important citation to be made is that of Axelrod's tournament. Professor Axelrod of University of Michigan ran a study in which strategies competed to play in an integrated Prisoner's Dilemma [2]. There has been extensive research on the results of this study, but most of them do not relate to this study in the sense that they do not have anything to do with DL; only those which do will follow.

Firstly, Tuomas W. Sandholm and Robert H. Crites, both of the University of Massachusetts at Amherst Computer Science Department, analyzed the usage of archaic RNN Q-learning models in the context of the Prisoner's Dilemma [3]. Their study, though, does study reinforcement learning – again, not the point of this paper – and does not consider the modern context of DL's popularity and usage. Given that their paper was written in 1996, it is simply outdated and does not use modern RNN techniques or consider more philosophically the repercussions of the study in today's world.

Secondly, Stanford Master's Student Keven (Kedao) Wang wrote a paper building off of Sandholm's and Crites's study, using new RNN models (the same LSTM used in this paper) [4]. This study found limited results surrounding the actual usability of these models and considered only very small tournaments—3 agents at most. Seeing as there was little to no data with large enough tournaments to really simulate a real world – let alone dynamic – environment, this study fails to capture the intricacy of this paper's environments. All of this, as well, does not account for the usage of reinforcement learning, another differentiating factor between this and his paper.

Finally, Shashi Mittal, writing with the Department of Computer Science and Engineering at the Indian Institute of Technology, Kanpur, considered the use of genetic algorithms and found some success [5]. This, while tangentially related to our

© The National High School Journal of Science 2024

paper, does not even use the same brand of machine learning and, while he did find success, it is in single round matches against one model—not nearly as dynamic or applicable as this paper's findings.

There are multiple other papers which analyze a similar concept, but they each fall into similar categories to those mentioned prior. Their citations, however, are still listed[6–10].

With this in mind, the main differentiators between prior study and this paper are these:

1. This paper considers massively more complex and dynamic scenarios far more applicable to the problem being solved.

2. In ignoring reinforcement learning, we have, while weakening the general argument that DL struggles as a whole, broken the argument down to be more realistic yet similarly applicable.

The Prisoner's Dilemma

The Prisoner's Dilemma is a game studied extensively in game theory. Two prisoners, Bob and Alex, are presented with a decision:

Each prisoner can either testify against the other (defect) or remain silent (cooperate). Should they both cooperate, they each get a 3 year sentence reduction, the police having no evidence with which to convict. However, if Bob defects, he receives a 5 year sentence reduction while Alex receives no reduction. Should both defect, though, they each only receive a 1 year reduction[11]. It's important to note that any values can be used, as long as they satisfy the conditions:: defecting against cooperation ¿ mutual cooperation ¿ mutual defection ¿ cooperating against defection. 5, 3, 1 and 0 were chosen because the original tournament used them. These situations are illustrated in Figure 1, below.

		Alex: Cooperate	Alex: Defect
Bob	Cooperate	3, 3	0, 5
	Defect	5, 0	1, 1

Fig. 1 Prisoner's Dilemma outcomes (Bob, Alex)[11].

This game is considered a dilemma because, while the best overall outcome occurs when both players cooperate, the choice with the greatest net benefit for an agent in that round is to defect. This becomes clear when one considers the game from only one player's perspective. Bob, as shown in Figure 1, knows Alex will either cooperate or defect. Should Alex cooperate, Bob nets more points to defect; the same goes for the defection. With this, both players seemingly benefit most from electing to only defect. In this dilemma, trust is impossible and both players will, if acting as purely logical agents, always defect. This, of course, leads to a worse outcome for both players than if they had simply cooperated.

The dilemma has a surprising pertinence to the dynamic and shifting world of humans, and even in its base form, although this resemblance in strength when multiple scenarios are placed in sequence, leading to the repeated prisoner's dilemma

Repeated Prisoner's Dilemma

The Repeated Prisoner's Dilemma builds on this scenario by playing the game multiple times in succession. In this extended version, players remember their opponent's previous moves and adjust their strategies accordingly, thus introducing the potential for strategy development and rapport building.

In the classic version, the lack of repeated interactions means that defection is the rational choice. With no future retaliation possible, it is better to take advantage of the opponent. In contrast, the Repeated Prisoner's Dilemma allows for the possibility of punishment or reward in future rounds depending on the actions taken in current and past rounds. This setup mimics real-life social interactions where people repeatedly encounter the same individuals, which can lead to stable cooperation or long-term rivalry based on the history of their interactions. The iteration of the Prisoner's Dilemma therefore introduces the idea of trust and cooperation building, both integral to the simulation of a more human environment.

In essence, while the classic Prisoner's Dilemma provides a snapshot of how individuals act under a single, isolated set of circumstances, the Repeated Prisoner's Dilemma offers a broader view of how strategies and relationships evolve over time. Study of such an environment can reveal complexities of human – and AI, in our case – decision making[12]. The study of these complexities and their relationship with the actual dynamic decision making capabilities of AI models are the main premise of this paper.

A: **Axelrod's Tournament**

The most famous study of the Repeated Prisoner's Dilemma is Professor Axelrod's tournament. Robert Axelrod, a political science professor at the University of Michigan, ran a Repeated Prisoner's Dilemma tournament in 1984[2]. This tournament was a computer-simulated competition in which participants submitted various strategies to play the Repeated Prisoner's Dilemma against each other. Each strategy was essentially a set of rudimentary rules

© The National High School Journal of Science 2024

dictating whether to cooperate or defect based on a variety of factors. The tournament iterated through numerous rounds allowing detailed analysis of what strategies worked and which did not. In his first tournament, Axelrod saw intriguing results. The winning strategy, Tit-for-Tat, was extremely simple. It starts by cooperating on the first move and, on all subsequent moves, simply mirroring the opponent's last move. Axelrod's conclusions [13] as to the success of strategies were:

1. Niceness: Never be the first to defect.
2. Provocability: Get mad quickly at defectors and retaliate.
3. Forgiveness: Do not hold a grudge once you have vented your anger.
4. Clarity: Act in ways that are straightforward for others to understand.

Axelrod's tournament findings were many, but they are largely outside of the scope of this paper. It is suggested that the reader read the prior source for more information.

B: **Why use the Prisoner's Dilemma?**
The Prisoner's Dilemma is an admittedly simple look at dynamic, multi-agent environments. While any of the hundreds could have been chosen, the Prisoner's Dilemma was selected for its simplicity and the fact that it does not cater to DL models. Its simplicity makes the construction of models and tournaments less intricate, therefore limiting possible confounding variables within the models. This also means that the tournaments need less computational power, allowing for more diverse simulations to be run with higher round counts and more agents. Also, while other multi-agent environments are built for the use of DL models, the Prisoner's Dilemma doesn't cater in that fashion. Thus, results from the Prisoner's Dilemma are more generalizable to real world situations where models might not have such an idyllic environment.

Implementation

To successfully simulate a Repeated Prisoner's Dilemma situation for DL models, a pre-existing environment must exist in which all models can train and act. Bearing that in mind, Axelrod's first tournament was used as the base environment for the models.

To create said base environment, documentation of Axelrod's tournament [14] was referenced. It is worth noting that the original tournament was recreated to the best of our ability, but some documentation from the original tournament was lost. With the exception of necessary liberties taken so as to fill those gaps, the tournament exactly replicates the original with no changes to cater to the models' needs.

A: **Classical Strategies**
Short descriptions of some of the classical strategies are listed below. However, it is suggested to refer to the aforementioned documentation for more in depth information on all of the strategies.

- **Tit-for-Tat:** An extremely simple strategy which starts by cooperating and copies its opponent's last move on all subsequent rounds.
- **Random:** Randomly selects between defection and cooperation.
- **Grofman:** Cooperates approximately 28% (2/7) of the time if the players' moves differed. Otherwise, always cooperate.
- **Joss:** Play Tit-for-Tat, but 90% of cooperations from the opponent are interpreted as defects.
- **Grudger:** Always cooperate until your opponent defects. Then always defect.

B: **DL Models**
In this study, three DL models were added to the tournament for analysis. Each was trained with no early stopping and 100 epochs – at batch sizes of 64.

1. A feedforward neural net (FNN)
2. A convolutional neural net (CNN)
3. A recurrent neural net (RNN)

Each model represents a major classification of neural networks [15]. All models were implemented using tensorflow keras. The first model is a binary classification model, a simple fully connected neural network, with two hidden layers (each of size 16), maximizing the success of the model within reasonable complexity. It takes in various inputs and outputs a value between one and zero representing the predicted chance its opponents will cooperate. This model was trained on a variety of input data: the last three moves of each player, the points for each, the percent cooperation for each, the round number, and a chi-squared test meant to detect random choices. These choices encode the data in a manner such that the model has:

1. A very strong understanding of the recent moves of both players, thereby understanding its local environment
2. Some understanding of its own location and the general choices of its opponent throughout the whole match, allowing for long term planning but avoiding overfitting

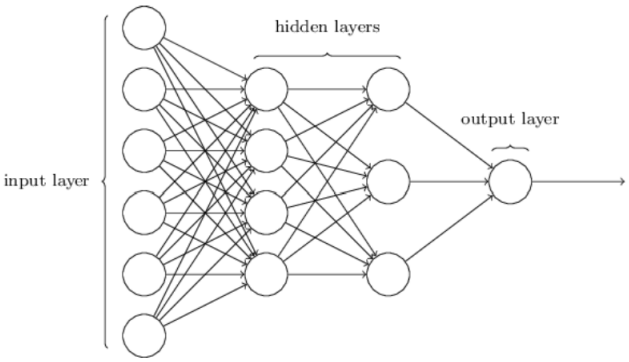

Fig. 2 Fully connected binary classification FNN[16] (general model, not the one implemented in the study)

For more information on the specifics of the input data and training, see the actual code of the models[17]. The second model was a convolutional neural network, implemented with 1D convolution meant to process sequential data (Figure 3). This model was built to take a sequence of

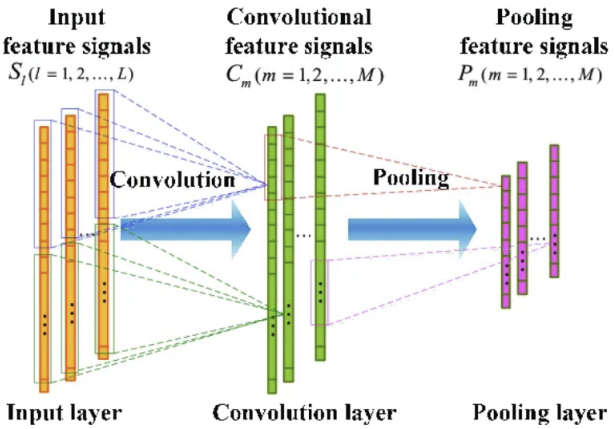

Fig. 3 1-Dimensional CNN[18].

all the moves that have occurred in the round and use them to predict the opponent's next move. This is as opposed to the input of the FNN, which is fed extra data such as the chi-square test, percent cooperation and other values which are all omitted from the CNN. At its core, it uses convolution, a process used to extract useful features of the data; and pooling, a process of creating a more compact and accurate representation of the data; to compact data to be fed into a binary classification model[19]. Thanks to the aforementioned processes and, therefore, the architecture of our CNN as a whole, the feature extraction is far more automatic in this model. Taking advantage of that, the CNN was trained on data that included all moves performed before the current one, allowing it to create its own compact representation. This is opposed to the FNN, which requires that the designer choose features to input as data, robbing the model of its ability to learn its own representations and heavily influencing the model's functionality. It should be noted that, in order to make that data of uniform size, it was pre-padded with 1's. This ensures the sequences are of equal length and therefore easily processed. It does, however, skew the average choice of the CNN towards cooperating. This skews the data significantly and means that the CNN's choices are not entirely its own. However, seeing as the CNN fared best with a padding of 1's (as opposed to 0's, for example), it is understood that the shortcomings of the model are still its own and not a product of the padding. The final model implemented was a recurrent, long short-term memory (LSTM) neural network. In its training, it was given the same data as the convolutional network, namely a sequence of the last moves, up to 200, padded with 1s. LSTM networks

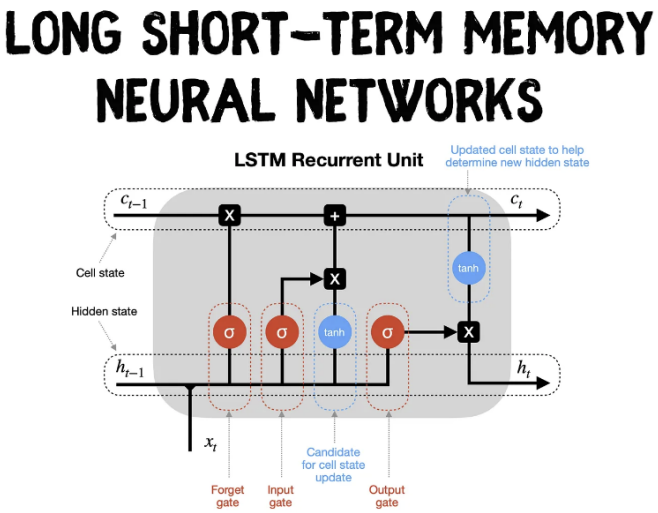

Fig. 4 Long-short term memory RNN[20].

are designed to retain information for extended periods. Each LSTM cell is a complex unit with a memory-carrying component, the cell state, which conveys information down the sequence. Three gates manage the regulation of information flow within the cell: the forget gate, which decides what to discard; the input gate, which updates the memory with new data; and the output gate, which determines the current output based on the memory state. These gates employ sigmoid functions to make binary decisions and tanh functions to scale values, ensuring the LSTM cell selectively retains or forgets information[21].

C: Overall Layout

In the tournament, every strategy faces off against each

© The National High School Journal of Science 2024

other in classic round robin style. Each strategy, each round makes a choice based on their architecture and they are awarded points based on the recorded outcomes. A complete tournament consisted of every model playing every other model twice in games that were 200 rounds (choices) long. It was discovered that, since strategies play each other to determine their overall score, the removal of one strategy can have massive ripples throughout the tournament as a whole. To account for this, many different tournaments were run with different combinations of models, for a total of 14 combinations:

1. All Strategies
2. DL Only
3. No DL
4. Everything but Random
5. Cooperation Focused (TitForTat, Grudger, Davis)
6. Competition Focused (Tullock, Shubik, Grofman)
7. Forgiving Strategies (TitForTat, Grudger, Joss, Davis)
8. Adaptive vs. Static (TitForTat, Joss, Grudger, RNN, Random, AlwaysCooperate, AlwaysDefect)
9. All Strategies with AD and AC (All strategies with AlwaysDefect and AlwaysCooperate added)
10. Complexity vs Simplicity (TitForTat, Stein&Rappoport, RNN, CNN, AlwaysCooperate, AlwaysDefect)
11. Early vs Late Game (Joss, Tullock, TitForTat, Grudger, AlwaysDefect)
12. No DL with AD and AC (All non-DL strategies with AlwaysDefect and AlwaysCooperate added)
13. Forgiveness Factor (Grudger, Q-LearningHard, Joss, AlwaysCooperate)
14. High Risk vs Low Risk (Random, Tullock, RNN, TitForTat, Grudger, AlwaysDefect, AlwaysCooperate).

In short description of the motivation behind these selections, certain combinations acted as controls; "DL Only" and "No DL" acted as positive and negative controls, respectively. Other controls included "All Strategies" and "Everything but Random." The rest of the combinations were chosen to highlight some specific facet of the game. In that way, each combination should have some differing effect on the performance of the strategies/models. In doing this, one can see the breadth of possibilities for the model's successes and failures, thereby eliminating the possibility that one model's success was entirely based on the specific environment that was chosen.

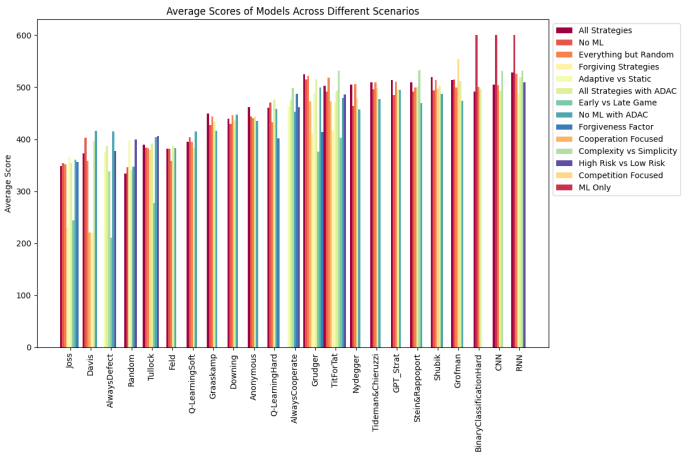

Fig. 5 Graph of all data.

Results

Figure 6 shows the average score of all models over twenty tournaments on the Y-axis, the model names on the X-axis and the specific scenarios color coded as per the legend. At first glance, it seems that the DL models are dominating. Closer observation, however, shows that their lead is derived in large part from a single matchup, namely "DL Only."

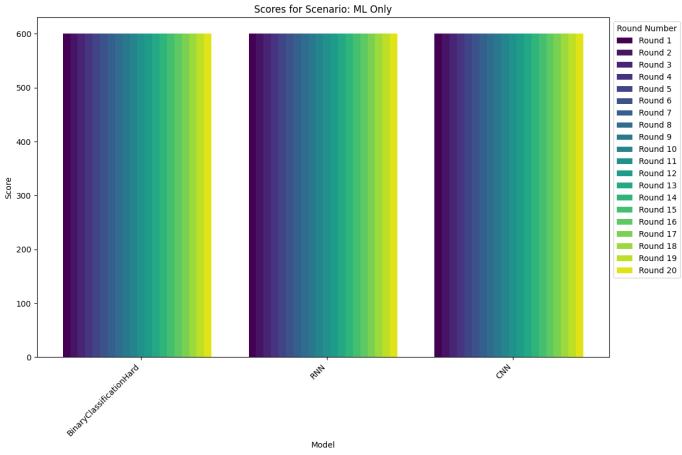

Fig. 6 Graph of DL only scenario.

DL models achieved perfect scores when playing each other, revealing that their favored strategy in a static scenario was cooperation, and in this a controlled environment, they executed it perfectly. In this respect, the model showed some evidence of strategic actions.

This paper does not refute DL's success in static scenarios like this one; it is well established that, in multi-agent environments with many DL models, the agents tend to strategize successfully. However, while they strategized successfully in this static case,

this changes as dynamic variables are added.

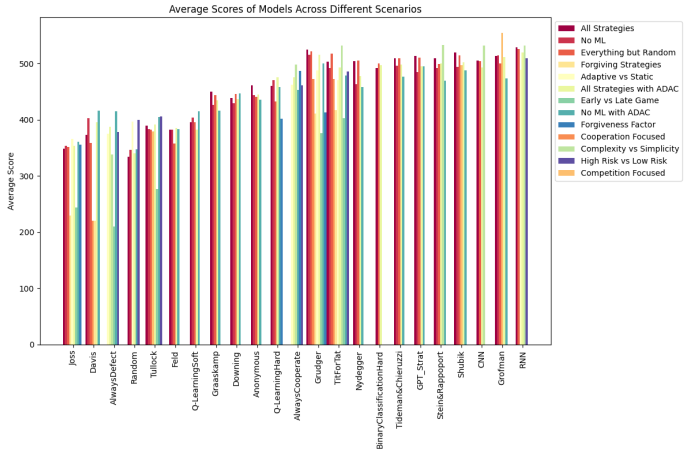

Fig. 7 Graph of all scenarios except DL only.

In figure 8, with the DL Only scenarios removed, it can be seen that the models' total standing dropped sharply. They are still ranked highly, with RNN ranked first and CNN ranked third, but it's not the effortless dominance that the original graph would seem to imply.

Additionally, when noise was added (a 5% chance of a strategy's choice switching), this trend was magnified exponentially.

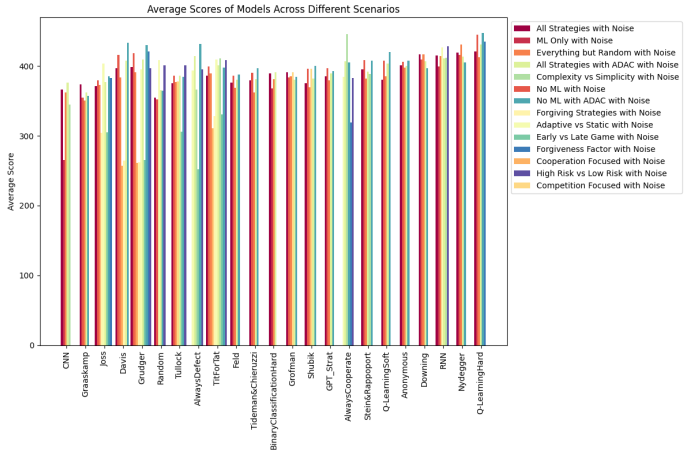

Fig. 8 Graph of all data with 5% noise.

A mere 5% percent chance of noise, a small fraction of the chaos embodied in the real world, was enough to completely turn the tables. RNN was lowered to third, Binary Classification to the middle of the pack, and CNN, which previously held third place, got last. Additionally, looking at the "DL Only" scenario further strengthens this conclusion.

Instead of the neat rows seen in the scenario without noise, the models seem unable to cope with the added complexity, and as a result, show a sharp decrease in performance.

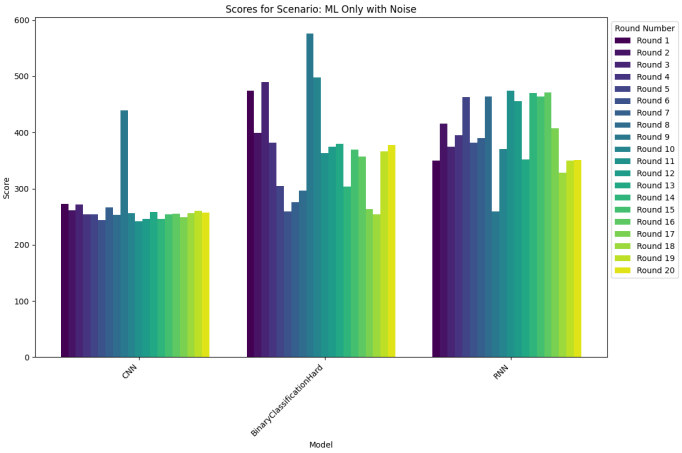

Fig. 9 DL models only with noise.

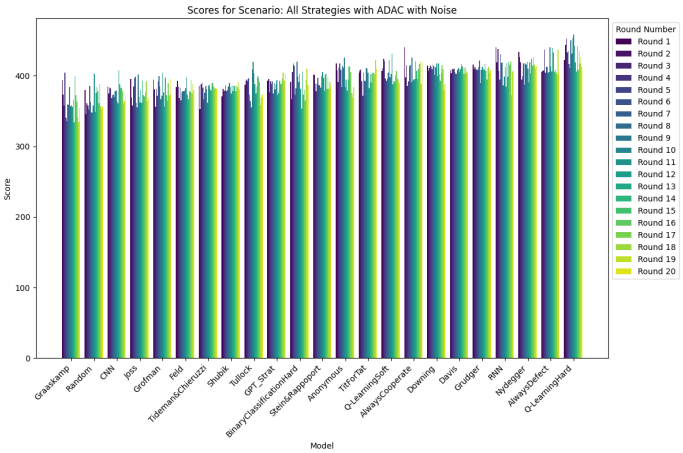

Fig. 10 All strategies scenario over 20 rounds.

Interestingly enough, in this noisy environment, Q-learning takes first place—therefore incidentally providing evidence that the use of reinforcement learning, in other words, the right tool for the job, can have great success. However, Q-learning was provided with more information than other models (an edge case which need not be discussed seeing as Q-learning is not the focus of this paper) so in reality, AlwaysDefect is crowned the rightful champion in this tournament. This, once again, highlights the limitation of DL in dynamic scenarios.

Analysis

These findings quite clearly underscore the limitations of DL when applied to real-world problems, but with this data, a new question arises: why do these models fare so poorly in dynamic scenarios?

The answer is, very simply, that this sort of action is not

© The National High School Journal of Science 2024

what DL models are built to do. Take the predictive success of the models, for example. All DL models were evaluated on their ability to predict their opponents moves after having been trained; in each case, they scored above 90% accuracy. Obviously, though, that performance was not reflected in the actual tournaments. The reason for this is that, despite successfully predicting the opponent's move, that is a far cry from making the correct decision yourself.

This idea is represented in the difference between hard and soft action policies. The DL models only predict the next move of their opponent; the choice of what to do with that information had to be implemented separately from the models. After a bit of experimentation, it was found that the best policy was to simply mimic the predicted choice, ensuring either mutual cooperation or mutual defection.

Additionally, it was found that taking the output of the model (the percent chance its opponent would cooperate) and turning it into a prediction was harder than expected. The first iteration used a soft action policy, a strategy that cooperated with a probability equal to the predicted probability of cooperation. For example, if the predicted probability of cooperation was 20%, then 20% of the time the model would cooperate and 80% of the time it would defect. After some experimentation, it was found that a hard policy worked best (any probability over 50% is cooperation, anything less is defect) because any mistaken prediction, which soft policies are more prone to due to the nature of random chance, will lead to a loss of trust, more defections, and fewer points.

The very fact that human hands must select an action policy reveals a fundamental weakness in DL; ideally, such models should be able to make their own decision. By making the choice for them, the DL models are robbed of flexibility and nuance, crucial parts of successful participation in the human world.

This concept and its application to static vs. dynamic scenarios is a fundamental weakness of DL models. In dynamic scenarios, the environment shifts even with the addition of the model to the environment. With even this miniscule change in environment, the model becomes less effective and each other change only compounds the issue.

This manifested itself as the tournament actually shifted when models were incorporated, with other strategies making decisions based on the models' actions, data which by its very nature cannot be included in the training data.

It's a futile chase, where every round of training will change its actions, and every one of its actions will change its environment, mandating more training.

The graph above is a perfect example. It's from an alternate CNN model, one that was not only trained before the tournament but between rounds based on the data from the previous round. The x-axis represents the epochs it was trained over, and the y-axis represents the average accuracy for that specific epoch.

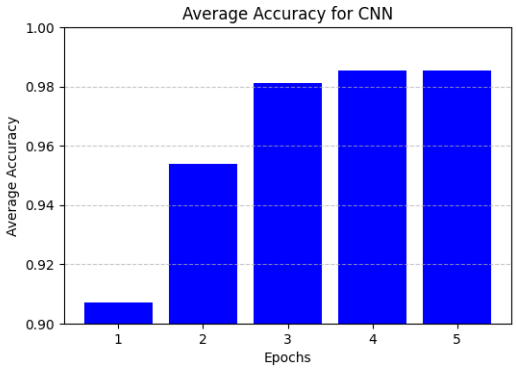

Fig. 11 Average accuracy per epoch in an adaptive CNN implementation.

Just by looking at the data, it becomes apparent that the model, despite achieving near-perfect accuracy after training, is once again made to start at a lower accuracy upon revising data from the previous round. This reflects the idea the model is constantly trying to catch up, with every improvement causing a cascade that just leads it right back to where it started.

It highlights how, while DL is good in static scenarios, when it encounters dynamic ones that shift upon its own entry, it struggles.

Unfortunately, all real, human scenarios are dynamic, and until this lack of flexibility is resolved, AI lacks the ability to participate in a truly human manner or, for that matter, successfully in any human environment.

Conclusion

This research offers a glimpse into the potential and limitations of DL in strategic contexts. While DL can mimic and sometimes enhance human strategic thinking, its ability to innovate under unpredictable circumstances has proven unreliable. DL cannot be treated as mere "black box", as has been becoming increasingly common in contemporary society, its limitations must be understood. Simply put, DL models are capable of amazing things, but, without more advanced techniques, they are not quite capable of acting as the omnipotent "black box" they are often made out to be, especially not in the human world and its chaotic twists and turns.

Conflict of Interest

The authors declare no conflict(s) of interest.

Acknowledgements

Asilomar Microcomputer Workshop

References

1 W. Caballero and P. Jenkins, *On Large Language Models in National Security Applications*, https://arxiv.org/html/2407.03453v1, arxiv.org.

2 R. Axelrod, *The Evolution of Cooperation*, https://ee.stanford.edu/~hellman/Breakthrough/book/pdfs/axelrod.pdf.

3 T. Sandholm and R. Crites, *Multiagent reinforcement learning in the Iterated Prisoner's Dilemma*, https://doi.org/10.1016/0303-2647(95)01551-5.

4 K. Kedao and Wang, *Iterated Prisoners Dilemma with Reinforcement Learning*, https://web.stanford.edu/class/psych209/Readings/2017ProjectExamples/wangkeven_17581_1628229_psych209_paper.pdf, Online]. Available:.

5 S. Mittal, *Machine Learning in Iterated Prisoner's Dilemma using Evolutionary Algorithms*, https://www.egr.msu.edu/~kdeb/seminar/shashi.pdf, Online]. Available:.

6 A. Dolgopolov, *Reinforcement learning in a prisoner's dilemma*.

7 A. Agrawal and D. Jaiswal, *When Machine Learning Meets AI and Game Theory*, https://cs229.stanford.edu/proj2012/AgrawalJaiswal-WhenMachineLearningMeetsAIandGameTheory.pdf, Available:.

8 Q. Bertrand, J. Duque, E. Calvano and G. Gidel, *Q-learners Can Provably Collude in the Iterated Prisoner's Dilemma*, https://arxiv.org/abs/2312.08484, arXiv.org, 2023.

9 B. Lin, D. Bouneffouf and G. Cecchi, *Online Learning in Iterated Prisoner's Dilemma to Mimic Human Behavior*, https://arxiv.org/abs/2006.06580, arXiv:2006.06580 [cs, q-bio], Sep. 2020, Available:.

10 P. Barnett and J. Burden, *Oases of Cooperation: An Empirical Evaluation of Reinforcement Learning in the Iterated Prisoner's Dilemma*, https://ceur-ws.org/Vol-3087/paper_24.pdf, Online]. Available:.

11 S. Kuhn, *Prisoner's Dilemma (Stanford Encyclopedia of Philosophy*, https://plato.stanford.edu/entries/prisoner-dilemma/.

12 Y. Dao, *The Prisoner's Dilemma: Real Life Application of Game Theory*.

13 L. Tesfatsion, *Notes on Axelrod's Iterated Prisoner's Dilemma (IPD) Tournaments*, https://faculty.sites.iastate.edu/tesfatsi/archive/econ308/tesfatsion/axeltmts.pdf., Online]. Available:.

14 *Strategies index — Axelrod 0.0.1 documentation*, https://axelrod.readthedocs.io/en/fix-documentation/reference/all_strategies.html.

15 C. C. Aggarwal, *Neural Networks and DL : a Textbook*, Springer, Cham, 2nd edn.

16 N. Bakshi, *Model Reference Adaptive Control of Quadrotor UAVs: A Neural Network Perspective*, https://doi.org/10.5772/intechopen.71487., Online]. Available:.

17 J. Cohen and B. Holland, *Prisoner's Dilemma Tournament*, https://colab.research.google.com/drive/1UAB8P4VfzjgH5EK4Dz6qovmDF_gAlPWo?source=post_page-----419bcc86397e.

18 K. Yang, Z. Huang, X. Wang and X. Li, *A Blind Spectrum Sensing Method Based on Deep Learning*, https://doi.org/10.3390/s19102270., Online]. Available:.

19 S. Kiranyaz, *1D convolutional neural networks and applications: A survey*.

20 D. Wang, *Typical artificial intelligence algorithms and real-world applications related to handwritten number classifier*, https://doi.org/10.54254/2755-2721/62/20240398., Online]. Available:.

21 B. Lindemann, T. Müller, H. Vietz, N. Jazdi and M. Weyrich, *A survey on long short-term memory networks for time series prediction*.

© The National High School Journal of Science 2024