Processing Poker, Playstyle, and Personality with Machine Learning

[name redacted for privacy][1]
[1]*The University of Melbourne, Australia*

INTRODUCTION

For all of technology's steady but inevitable advancements, none other than one murmuring of an idea (a by-product of discussions of possible future technological implementations) has struck more fear into the heart of the working man; the idea that the lion's share of humanity's executive functions could, should, and necessarily would come to be replaced by machine. In isolation, this fear is paradoxical in nature, implying a phobia for what has been relentlessly sought since the invention of the wheel; a maximisation of the ratio between living conditions and hours worked. But with the context of capitalistic greed, burgeoning under- and unemployment, as well as the increase in wealth inequality and the proportion of the population in casual/part-time positions, this collective hysteria contra automation is shown for what it truly is: an expectation of the outcomes of opening Pandora's box. The contents of which is of course automation, albeit more specifically to a white collar perspective, artificial general intelligence ("AGI"), which despite the endless hype by opportunistic mass media outlets as being an ultra-intelligent sentient algorithm, is still many a developer's tantalising green light in the horizon – a distant, presently intangible concept at best.

A difficult and labour-intensive journey stretches out before the prize of AGI, with notable examples such as the resources that were poured into currently best-of-class neural networks ("NN") such as DeepMind and its successors; which are certainly impressive in their areas of focus (viz. Go, StarCraft2, protein folding, …) but cannot be used to compute intelligence as a whole, and require prohibitively demanding training resources. With NNs' key limitation of restrictive scope in mind, the possibility of breaching this constraint arises, and the most economical and elegant approach to this lies in NNs' key requirements: training resources. Is the duality of complex training algorithms and enormous training sets an inseparable part of AGI's arduous journey, or can these requirements be minimised? Would it be possible to train a NN to perform a complex task from training data (which is abundant) alone, without intricate training code? If not, and meticulously instructing the NN on how to process training data and act is implicitly necessary, then the path to AGI stretches at least as many steps further to the horizon as executive processes the mind can imagine needing to be programmed into a completed AGI.

"Texas hold 'em" poker is a card game enjoyed throughout the world, and the focus of multi-million dollar championships such as the World Series of Poker. Simply put, the game starts by a "dealer" dealing two "hole" cards to each player, who then proceed to bet based on a variety of factors (although mainly how favourably they perceive their own cards to be), before the dealer deals three "flop" cards. This is followed by subsequent rounds of betting with more card dealing interspersed, with the player with the "best" hand at the end (decided by a set of complicated rules) winning the sum of all previous bets. Metagame, psychology, game theory, personal play styles, and a variety of other factors massively complicate the game, but despite this, numerous poker "bots" such as Cepheus (University of Alberta) have been developed, and play allegedly "essentially perfect" games.

However, Cepheus has poker's rules programmed into it, which, if necessary, would lead one through induction to be able to argue that it would therefore be necessary for AGI to be taught the rules of thought itself, an obviously unfeasible task. Thus, it remains to investigate if such rules and training instructions can be stripped from complex-task NN development and consider what the implications of the resulting findings may be.

RESULTS

The training data to be used was acquired from the Computer Poker Competition's latest archives, a recorded poker competition between poker bots.

	(1)	(2)	(3)	(4)	(5)	(6)	(7) *	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	Winrate
(1) Intermission		287.00	70.99	139.20	137.45	153.60	750.00	58.48	216.32	392.52	22.02	511.89	750.00	750.00	750.00	356.39
(2) PokerBot5	-287.00		-210.04	-308.60	-242.19	-290.66	602.35	750.00	550.16	750.00	750.00	263.22	750.00	750.00	750.00	326.95
(3) Feste	-70.99	210.04		9.10	65.05	27.68	750.00	-2.52	212.89	346.21	291.80	475.76	750.00	750.00	750.00	326.07
(4) RobotShark_iro	-139.20	308.60	-9.10		-43.98		750.00	40.61	78.33		-110.81	465.61	750.00	750.00	750.00	299.17
(5) HITSZ	-137.45	242.19	-65.05	43.98		61.55	750.00	-99.09	155.53	375.75	-170.12	504.81	750.00	750.00	750.00	279.44
(6) RobotShark_tbr	-153.60	290.66	-27.68		-61.55		750.00	45.45	79.40	241.75	-115.98		750.00	750.00	750.00	274.87
(7) ASHE	-750.00	-602.35	-750.00	-750.00	-750.00	-750.00		750.00	750.00	395.29	750.00	406.52	750.00	750.00	750.00	67.82
(8) Slumbot	-58.48	-750.00	2.52	-40.61	99.09	-45.45	-750.00		219.56	302.74	-14.31	499.14	750.00	-164.79	750.00	57.10
(9) PokerCNN	-216.32	-550.16	-212.89	-78.33	-155.53	-79.40	-750.00	-219.56		174.57	-25.37	309.23	750.00	324.94	750.00	1.51
(10) Hugh_tbr	-392.52	-750.00	-346.21		-375.75	-241.75	-395.29	-302.74	-174.57		-233.76		750.00	750.00	682.84	-85.81
(11) Rembrant6	-22.02	-750.00	-291.80	110.81	170.12	115.98	-750.00	14.31	25.37	233.76		458.15	-741.90	-750.00	750.00	-101.94
(12) Hugh_iro	-511.89	-263.22	-475.76	-465.61	-504.81		-406.52	-499.14	-309.23		-458.15		-403.70	623.61	750.00	-243.70
(13) PPPIMC	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	741.90	403.70		750.00	750.00	-346.74
(14) SimpleRule	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	164.79	-324.94	-750.00	750.00	-623.61	-750.00		-750.00	-538.13
(15) ElDonatoro	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-750.00	-682.84	-750.00	-750.00	-750.00	750.00		-638.06

Figure 1 – Cross table of win rates for each poker bot within the computer poker competition, with a perfect win rate holding a value of 750.00. Intermission's matches against ASHE are highlighted with an asterisk.

Of all bots in this competition, bot "Intermission" had the highest win rate, and was chosen as an optimal training set, and its games against bot "ASHE" were chosen, as that is the next most able bot with which it had a perfect win rate (750.00)(Fig.1, *). This data set was compiled into the training data set ("t-set") to be used, which was then parsed pythonically. The test data set to be used was an excerpt of the t-set, and the NN's performance was measured by comparing its outputs to the t-set (which is for all intents and purposes represents a practically perfect set of plays).

Firstly, three separate NNs with different activation functions (Swish, Sigmoid, and ReLU) were trained on the t-set at varying hyperparameters and 1000 epochs. This was aimed at approximately determining which hyperparameters and activation functions should be further analysed. Their optimal performance values and the epoch values at which they were achieved were recorded.[1]

Table 1 – Swish activation function's performance values with differing hyperparameters. The first value in each cell is the highest performance value recorded throughout 1000 epochs, and the number following it is the epoch at which that performance value was recorded.

		Hidden Nodes		
		10	**100**	**1000**
Learning Rate	**1**	67.27, 1000	67.04, 673	65.27, 1000
	0.1	74.42, 566	80.25, 698	91.30, 606
	0.01	28.82, 1000	28.04, 1000	34.26, 1000

[1] Unless specified, all performance values ignore "check" (represented as 0s) plays in the test set and bets that are larger than the test set's bets, as any value above a 0 (i.e. a bet) would produce the same result as a check, and is better than a fold, that is, it leads to an increased chance of potentially winning one game turn, and any bet larger than required would simply increase the money at play, and therefore increase the profitability, as this is a "perfect" data set. Whether it is better to check if given the option than be forced to either bet or fold depends on the players' skill levels, but in a high throughput match such as this, where earning a small win many times is just as well as earning one large win once (as a human would prefer), it is therefore valid to make this alteration to the way the NN is tested. Regardless, this leads to a more stringent test for the NN than if checks and extreme bets were considered, with a 5 – 30 percentage point decrease in the NN's performance scores being seen in this method of testing. In other words, in a more relaxed environment, where the NN is not deprived of "easy passes" on its tests (i.e. practical applications), it performs noticeably better than the values in tables 1, 2, and 3.

Table 2 – Sigmoid activation function's performance values with differing hyperparameters. The first value in each cell is the highest performance value recorded throughout 1000 epochs, and the number following it is the epoch at which that performance value was recorded.

		Hidden Nodes		
		10	**100**	**1000**
Learning Rate	**1**	60.54, 1	27.22, 1	5.42, 356
	0.1	69.37, 13	58.65, 3	27.24, 1
	0.01	69.11, 140	73.43, 13	58.73, 3

Table 3 – ReLU activation function's performance values with differing hyperparameters. The first value in each cell is the highest performance value recorded throughout 1000 epochs, and the number following it is the epoch at which that performance value was recorded.

		Hidden Nodes		
		10	**100**	**1000**
Learning Rate	**1**	4.17, 2	2.03, 16	0.60, 6
	0.1	15.72, 3	4.55, 49	2.32, 679
	0.01	17.96, 37	14.15, 13	4.84, 116

It was clear that on average, ReLU consistently underperformed, while Swish outperformed all other activation functions, with Sigmoid closely following. Swish was further analysed, and its optimal hyperparameters at 606 epochs (the coordinate for its local maximum within these limits) could be determined by training it for this many epochs with differential hyperparameters and mapping the output to a surface plot, with the highest performance and the hyperparameters at which it occurred, being recorded. This was done as a proof of concept at 100 epochs (an economical compromise where it previously regularly performed well)(Fig. 2a, 2b), but ultimately, simulating this at 606 epochs would be beyond the scope of this report and, in any case, is unfeasible with the current hardware at hand; more than 5 hours were elapsed while attempting to do so. The surface mapping that resulted (Fig. 2a, 2b) implied that the NN would benefit somewhat from an increase in hidden nodes, at least at this epoch.

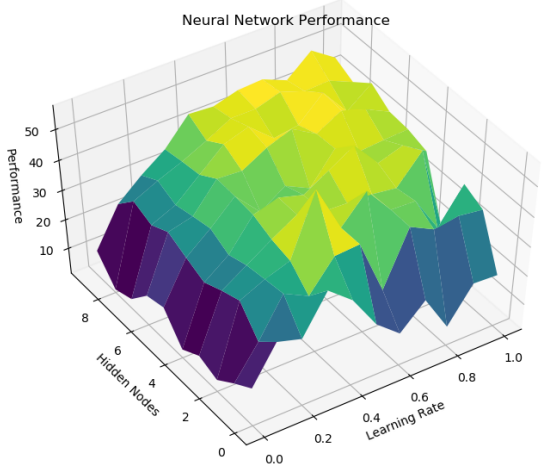

Figure 2a – Swish activation functions performance at 100 epochs, with differing hidden nodes and learning rate.

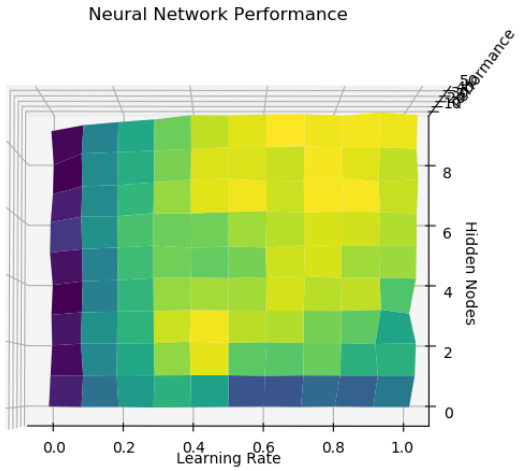

Figure 2b – Swish activation functions performance at 100 epochs, with differing hidden nodes and learning rate.

Ultimately, the probable optimal hyperparameters acquired in tables 1, 2, and 3 were used to train the NN with all three activation functions, and it was tested on the testing set thrice, yielding appreciable test performance scores (table 4).

Table 4 – Activation function perfomance values for three separate repeats of training and testing at the optimal hyperparameters determined in tables 1, 2, and 3, for each function.

Activation function		Run		
		1	2	3
	Swish	84.20	82.78	80.49
	Sigmoid	73.34	73.44	73.46
	ReLU	23.10	10.96	33.16

Table 4 was extracted from the plots generated by the outputs of all three activation functions (Fig. 3), which are representative of the NNs' play styles and bets respective to those expected by the test set.

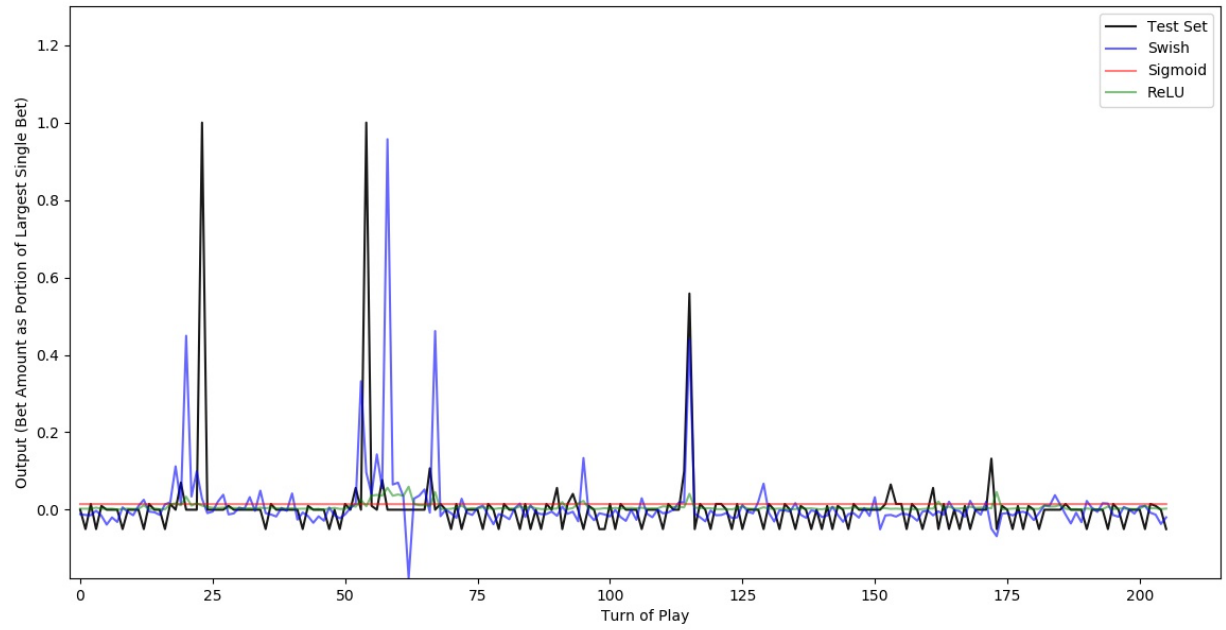

Figure 3 - Output of all three activation functions at optimal hyperparameters when tested for each value of the test data set at each turn of play of the test set. Values of 0 represent checks, above 0 bets, and below 0 folds.

More consideration was given to the Sigmoid activation function which has so far been overshadowed by the Swish activation function, but has had promising results. It underwent a differential hyperparameter surface performance mapping at 140 epochs, its optimal number of epochs, between 0 and 10 hidden nodes, a hidden node value with the best average performance of table 2 (Fig. 4). Unfortunately, it did not result in any better performance than that previously recorded and only showed that its optimal hyperparameters were those already determined in table 2.

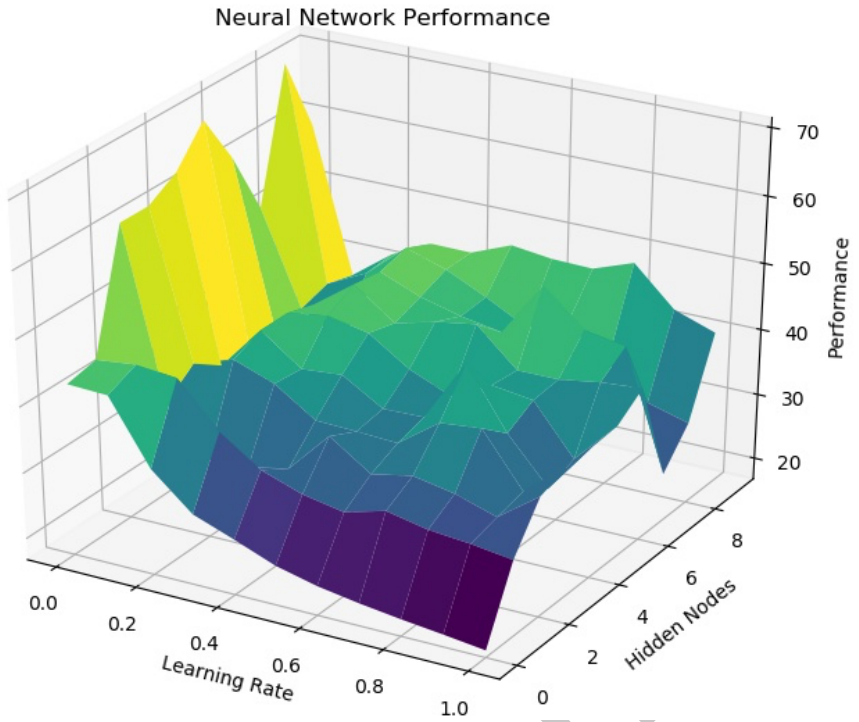

Figure 4 – Surface mapping of Sigmoid activation function's performance with differential hyperparameters at 140 epochs.

DISCUSSION

Complications

Numerous setbacks were experienced, the most immediate being the inherent nature of the data set, which was overwhelmingly large, totalling hundreds of thousands of lines of information fragmented across thousands of logs. Parsing was required both within python and within a Linux shell to produce coherent and usable data. Furthermore, the data, having been generated by bots, at times contains questionable plays, which, when considered in addition to the naturally limited number of inputs that poker provides (even after the fact that the opponent's hands are accessible and visible in this particular case), can lead to realistically "rubbish" information (spots of noise amongst the signal), even if that information is considered legitimate by non-human players. This is another trivially obvious complication to the data – it is entirely of computer-generated origin, due to freely-available human data being scarce and even more difficult to parse. This point does not invalidate the data, as computers can, and often do, triumph over human players in many games, but does suggest it may lack critically "human" aspects of thought.

If it is posited then that these human-like properties of play could arise from the program's execution alone, it is largely irrelevant, as it is impossible to ascertain so in any rigorous quantitative manner. However, after one considers the entire program on first principles alone, it is obvious that this is a ridiculous postulation, and that the output produced by the NN is nothing more than the clockwork mechanism of a mathematical algorithm. It is an algorithm limited primarily by the computational capability of the hardware on which it is executed, and secondarily by the nature of its code, which was programmed practically by hand and lacks the capabilities, streamlining, and processing efficiencies of well-established NN-specific libraries, with tens of hours of processing time required for the results.

Limitations

Certain deficiencies obligatorily befall upon the NN due to inadequacies in programming wherewithal, though these are of a fortunate manner as they emphasise the method's purity. The NN itself is incapable of considering the game as a whole – it is limited to considering each turn of play in isolation from all others – and due to parsing difficulties does not have access to important implicit data, such as the size of its current betting allowance ("chip stack"), the opponent's chip stack, the size of the communal betting pool ("pot"), and its position. Furthermore, it also does not have access to other extrinsic statistics that are important for calculated, high level play, such as win rates, average bet sizes, pot odds, and other predictive strategies. All limitations further the concept of extracting behaviour solely from data, void of any nuanced instructions.

Productions

It was clear that on average, ReLU was simply unfit for this specific machine learning task, likely due to its activation function and derivative's propensity to create dying neurons. As the data set was scaled to fit all bets to values between 0 and 0.99…9, with most values either being 0 ("checks"), -0.05 ("folds"), or lying very close to 0 (small, reasonable bets), weight updates caused most to fall below 0, leading to completely dead NNs. On the other hand, Swish and Sigmoid were almost perfectly suited for this task, thanks to their activation functions and derivatives' shapes, which allow them to consider small negative values well, as well as the use of Xavier weight initialisation which mostly mitigated any issues with vanishing gradients for these functions. These activation functions produced very good performance values (Table 4), usually far better than simply guessing between checking, folding, or betting (1/3 chance; 33%), and being able to calculate an adequate bet amount when they did so (Fig. 3).

This is extremely promising for the concept of deriving meaning and action from simple data without the need for instruction programming by a human. It is interesting to consider the possibility that a refined NN could perform beyond its t-set, as the performance value derived in these instances is a crude hit-or-miss calculation (with "hits" such as bets being further graded based on how closely they came to the desired bet) which is perhaps not perfectly accurate of all possible poker performance. In other words, although the t-set already denotes a perfectly profitable game, in such cases where the t-set designates a check, it may be even more profitable (and thus of better performance) to instead bet, and so on, and it is entirely possible that a NN could perform to this higher standard, given the fact that it currently – with any of the hyperparameters tested – never entirely reaches 100% performance (Tables 1, 2, 3), meaning, it has not been overfitted and instead is playing a sort of style of its own in each activation function.

Remarkably, this is technically the case; ReLU, despite being composed entirely of dead neurons in most instances, can be interpreted as an underfitted model, resembling in its plays a very conservative, weak player that checks on every possible turn. Although a questionable strategy, this would not actually result in a loss every time; good fortune would ensure at least some profits sometimes, even if it is an inferior play style to the t-set. Conversely, the Sigmoid and Swish activation functions are reasonably fitted, and have their own, distinct playing styles, that resemble the t-set but behave differently – at times they bet where the t-set would not have, and vice versa (Fig. 4). It is almost resemblant of personal preference and higher order decision making, as is seen in human players, just with a lack of consciousness. This reasonable success places salient implications on the nature of biological NNs; if a complex executive function can be simulated with such unreasonable effectiveness simply through an algorithm, then perhaps the concept of consciousness is simply an algorithm too, or what emerges from a set of algorithms that allow for normal cognitive function.

Applications

The discoveries extracted from the NN have applications that go beyond simply playing poker. Although only a brutish approximation of winning poker technique, the grand realisation of it being possible to deduce correct behaviour for complex systems can lead to adaptations that allow applications in anything that can be described as a state (composed of quantifiable, related factors) resulting in an action.

The unimaginative, materialistic human mind instantly directs its focus to applications relating to the generation of wealth, through the share market, advertising, and the like, but this revelation of inherent patterns in mankind's unwavering stream of activity implies the existence of fundamental worldly truths, that is, underlying facts about man that could translate into breakthroughs in medicine, psychology, politics, and philosophy comparable to the advancements in mathematics and engineering that resulted from the development of calculus. Medical diagnosis, consequential personality tests that prescribe optimal careers, and even intelligent design of new innovations could theoretically be possible, all without the need for knowing the "rules" for each, just adequate data sets.

Extensions

However, as is clear from the resulting performance data (Table 4), the NN is far from perfect. Various options lay open for its subsequent improvement, and the fact that it was able to produce decent, meaningful (Fig. 3) results despite these improvement opportunities is yet another feather in its cap. Changes such as an exponential increase in the number of its hidden layers (deep NN) and nodes, and meticulous fine tuning of hyperparameters would undoubtedly yield greater results. The introduction of white noise into the data set, augmenting the t-set (through artificial manipulation of values), removing losing values, and using a human generated t-set could also be of benefit for future trials, as would increased processing power, introducing bias values, a more intricate NN kernel, and pitting two copies of the NN against each other in an adversarial learning arrangement.

CONCLUSION

Being able to derive a useful NN that produces meaningful output for a complex task such as playing poker, with only the need for a large data set, and without the need for programming instructions regarding the task, is extremely promising for future AGI programming and suggests that intelligence could simply be an incomprehensibly complex algorithmic output to a large set of inputs.