LukyLuke
the smart AI poker bot

1st Svein-Egil Seppola Haugan
Student at University of Tromsø
Tromsø, Norway
sha288@uit.no

2nd Benjamin Lunde
Student at University of Tromsø
Tromsø, Norway
blu009@uit.no

***Abstract*—A lot of breakthroughs in Artificial Intelligence has been in entertainments and games, with has been adapted to real and useful situation. One big struggle in AI is handling uncertainty and in-perfect information. Poker is a grate place to explore these features and the philology of how humans handle it.**

***Index Terms*—AI, deep-learning, poker,**

I. Introduction

Poker is a game of gambling, and has been an important part of human history and technology advances. There has been a lot of mathematical advances gained for people seeking fortune in gambling, like Pierre de Fermat, Blaise Pascal and Gerolamo Cardano. [1]

Lately these fortune seekers has been computer scientist, like Facebooks DeepStack [2] and Microsoft's Claudico [**?**]. The easiest and most popular Poker variant is called Texas hold' them, and specify Heads-up no-limit Texas hold'em are used for computer bots, because of simplicity and because all bets are of a fixed size resulting in just under 10^{14} decision points [**?**]

Texas hold them works by first getting dealt two cards, known as hole cards, are dealt face down to each player, and then five community cards are dealt face up in three stages. The stages consist of a series of three cards ("the flop"), later an additional single card ("the turn" or "fourth street"), and a final card ("the river" or "fifth street"). Each player seeks the best five card poker hand from any combination of the seven cards of the five community cards and their two hole cards. Players have betting options to check, call, raise, or fold. Rounds of betting take place before the flop is dealt and after each subsequent deal. The player who has the best hand and has not folded by the end of all betting rounds wins all of the money bet for the hand, known as the pot." [3]

A. Background and Related work

Although the main part of poker is down to strategy and calculating odds, psychology and the human element is a part of the game that is the hardest for the computer to emulate.

The tow big parts of psychology comes down to reading your opponent and also how the other player perceives you and your play style. Knowing how the bot plays can be an important part of the psychology, because playing constant can be an downfall. This is because it makes the bot easy to read, and lets the player predict how it's going to play, making it easy to exploit.

When playing against an human player we the bot needs to know how a person played and adapt to it. There are four main player mentalities that are popular at the table. Tight-passive Loose-passive, Tight-aggressive (TAG), Loose-aggressive (LAG). Although this classification might be crude, it's a way to abreast the problem.

These categories can be identified based on two spectrum's. One is how aggressive the player is at betting, and the other how conservative and strict the player is. When playing versus an aggressive player, the money is going to fluctuate, meaning that playing thigh and waiting for optimal plays to bet hard is smart. A passive player on the other hand can be easily drained by upping the bets and draining them when they are holding back. Loose players means playing for value-bets, and less bluffing and the opposite for to tight players. [**?**]

The difficulty in creating an imperfect game is that we can can't determine the optimal strategy for the subsection of the game alone. This means that we have to take into consideration parts of the game that are unreachable still plays an role in the outcome of the current section of the game.

They way related bot has solved this is to try to achieve a game state which is called a Nash equlibrium. The definition of a Nash equilibrium is: "if each player has chosen a strategy, and no player can benefit by changing strategies while the other players keep theirs unchanged, then the current set of strategy choices and their corresponding payoffs constitutes a Nash equilibrium." [**?**]

An example in the game of tic tac toe, the first player to reach three in a row of it is given symbol wins, which is played in an 3 by 3 grid. In therms of a Nash equilibrium, if each player has an optimal strategy, the game would always draw.

In poker we can try to approximate a Nash equilibrium, and if the other player tries to play optimally. To try an approach this prior work like Liberatius has approached this using Counterfactual Regret Minimization (CFR) [**?**]

CFT converges to an equilibrium by iteratively traversing the game tree. In order to deal with extremely large games, abstraction is typically applied before running CFR. The abstracted game is solved with tabular CFR, and its solution is mapped back to the full game. [**?**]

This can be done by first, coming up with a counterfactual values representing different game outcomes. Second, they apply a regret minimization approach to see which strategy leads to the best outcome.

The first value is created using Real Time Planing/citeClaudio. This works by re-calculating for that sub-tree every time for the state, only re-calculating for the remaining sub-tree of the current game. By doing continual re-solving, reconstructing a strategy by re-solving every time we need to act, never using the strategy beyond our next action. We can try to reach an optimal play pattern.

The second value can be reached using the local best technique /citeClaudio. Local best-response technique produces an approximate lower-bound on how much a strategy can lose. This is done by choosing their next action probabilistic based on which strategies are best responses to the tree as a whole.

Usually a third value is used to reduce excitability. Like the example for tic tac toe, an approach to use one sub optimal play, can if you know the other player continue playing optimally be exploited to gain an win.

Usually this is done by average the most recent strategy with all past strategies, but has been shown to be suboptimal versus smart players. [**?**]

Deep-stack works around this by creating an intuition using deep learning on an Artificial neural network. Deep learning is a class of machine learning algorithms that uses multiple layers to progressively extract higher level features from the raw input. For example, in image processing, lower layers may identify edges, while higher layers may identify the concepts relevant to a human such as digits or letters or faces. [**?**]

The Artificial neural network (ANN) will be created using an large data-set from UCI [**?**]. ANN are computing systems vaguely inspired by the biological neural networks that constitute animal brains.[1] Such systems ”learn” to perform tasks by considering examples, generally without being programmed with task-specific rules. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as ”cat” or ”no cat” and using the results to identify cats in other images. They do this without any prior knowledge of cats, for example, that they have fur, tails, whiskers and cat-like faces. Instead, they automatically generate identifying characteristics from the examples that they process. [**?**]

B. Problems and aim

What we hope to improve is the human element of the poker bots. By combining different approaches we hope to create a more engaging and fun bot to play against.

By approaching a Nash equilibrium we hope to create a bot that is competitive and difficult to play against. To measure we want to do a measurement used to rank professional players called milli-big-blinds per game (mbb/g). One milli-big-blind is 1/1000 of one big blind. A player that always folds will lose 750 mbb/g (by losing 1000 mbb as the big blind and 500 as the small blind. [**?**]

We also want people to feel like they are playing versus an player that acts like a human, and dose not feel like it is cheating. To do measure this we hope to have players try versus the bot, and rank their satisfaction versus the bot.

This bot should also be adaptable to other real world applications that hopefully can improve other real life application is the world, where analyzing in-perfect information is key, like in business negotiation.

Potential issues when creating a model is that it can overfit and underfit. Overfitting is when it does not generalize well on new data that it has not seen before. Normnally one has evalution data that one tests against the model. If the accuracy is far from what the model gave out while training means that an overfit has occured. While an underfit would be when it cannot see patterns from the data that it is given and thusly give unprecise predictions [**?**].

II. Contribution

Here you describe the contribution of your work, with interesting parts of your actual implementation.

Present following: the goal(s), and deliverables, impacts and results of the project. This should describe the importance of the anticipated results in terms of the potential impact. The potential impact can be in the short or longer term. The chapter should also specify the planned measures for exploitation, communication and dissemination of the project results.

Describe the target audiences and stakeholders/users of the project results and outputs (in or beyond the scientific community) and the ethical issues (what ethical problems can arise) and the sustainability aspects of the project.

The goal of this project was to create a prototype for an poker AI that was able to play poker, so by creating a model and training it would one be able to make it predict what the likelihood of it having a winning combination. Though the methods for creating the bot can be used with similar problems, where one has to make decisions on something that one cannot for certain predict the outcome for.

There was not a specific ”stakeholder” the finalized prototype was meant for, instead it was meant to see how a potential poker bot could be implemented, find its limits, and what takes to create a precise bot. Though the bot could be used in real world cases where one could e.g. use it to predict on online poker. Though an ethical issue with that is that it can be considered as cheating. If the poker bot does not make precise predictions then it can loose money for the player, so a larger data set should be used to ensure that the neurons are weighed correctly for all possible cases that might arise in a poker game to make as precise predictions as possible.

III. Implementation

Present the implementation of the system. Use a prototype software development method, such as presented below, in Figure 1. The method could be any but the task is to describing the implementation and showing a visual image of the phases of your implementation.

Common phases are:

- Requirements collecting
- Design the requirements
- Architecture of the system
- Development of the system
- Testing and Quality assurance
- Delivery with User manual

1) Requirements collecting: As we set out to create a poker bot we first needed to check out whether anyone had created anything like this, and if it was the case then what those creators had in mind when they created their own poker AI.

Since the poker game that we try to implement is Texas Hold 'Em, we needed to check what the rules of the game were and then try to implement them. Due to the many rules of the game so have we created a more clean version of the game where the bot predicts whether the cards one has are good and then one decides whether to bet or not.

2) Design the requirements: The requirements had to be set by the rules of the game itself – texas hold 'em. Though it was early seen that the rules would need more players, thus adding more complexity to the game. Instead was it more focused on getting the deep learning part into the game, since if the game can predict the chances of winning based on the cards it has then other rules should not be as complex to add in, though it would need more human players or bots for quality assurance.

3) Architecture of the system: The visual part (front end) part of the application was implemented using pygame, while the computations (back-end) were done with tensorflow.

4) Development of the system: To implement the "Luky-Luke" AI was a dataset found online [1]. After The data set was found could one train a neural network to make predicitons. To do this was tensorflow used, tensorflow is a library for machine learning created by Google, and it made it possible to train the model that will be used to make predicitons based on which cards one has.

The game itself was implemented using python with the library pygame. By going online and finding pictures of cards and a poker table could we "simulate" a poker table and display the players cards as well as the "dealers" cards that are on the table. After that one added in

5) Testing and Quality Assurance: Testing this application can be quite hard. We've done the normal testing and training with tensorflow on a dataset, and then try to apply it in our game. It cannot either say for sure whether a hand is going to win, especially since we are using deep learning and give the bot an opportunity to bluff. Though one knows for certain that certain kinds of "hands" are more likely to statistically be better off than others, which the bot has been trained with to some extent since the labels in the dataset tells whether the different kinds of hands won or lost.

A. Design

Present a design of your solution. Describe the contents with models that are used in the degree project.

Preferable use a kind of map.

[1]Poker Hand Data Set https://archive.ics.uci.edu/ml/datasets/Poker+Hand

- Cognitive maps
- Inference network
- Decision tree
- Charts

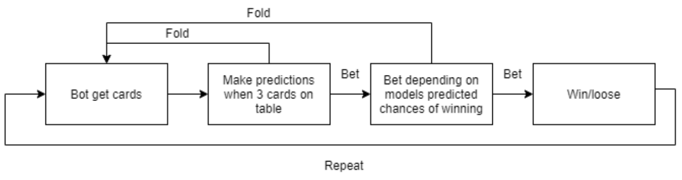
Fold
Fold
Bot get cards
Make predictions when 3 cards on table
Bet
Bet depending on models predicted chances of winning
Bet
Win/loose
Repeat

Fig. 1. Diagram of the flow of the bot

As one can see in figure 1 so goes the bot through 4 different steps when deciding what it should do in the different situations. It can either fold when it has bad combination and does not think it can win, or it can bet when it thinks it has a winning chance. This will continuously repeat until there either are no more players or if the bot is out of money due to lacking performance.

B. Architecture

Present an architecture of your system. Describe the contents with models that explain what parts / modules are used in the project.

To be able to create a Poker AI was several libraries needed. Everything from collecting the data, to train the models, among many other things. The libraries were used to remove the need to do everything manually and instead load some of that work onto the libraries.

- functools
- numpy
- tensorflow
- pandas
- sklearn
- os
- pygame

C. Results and Evaluation

Describe the results of the degree project. Evaluate what you have done. (What do the results mean, what are the implications of the results.)

As one can see on figure 2 so will the game display the cards on the table, and the player that is playing has his cards down in the bottom. As well as money that the current player has in the bottom right and the money on the table is on the bottom left.

Describe testing of your implementation, and results from your testing. The only way to test whether the bot behaves as it should is to see – among other things – whether or not it seizes on good card, if it is bluffing when it has bad cards, and these tests are done through playing multiple games against it. Though it is possible to make up a lot of different card combinations and see how the mode predicts the card combination.

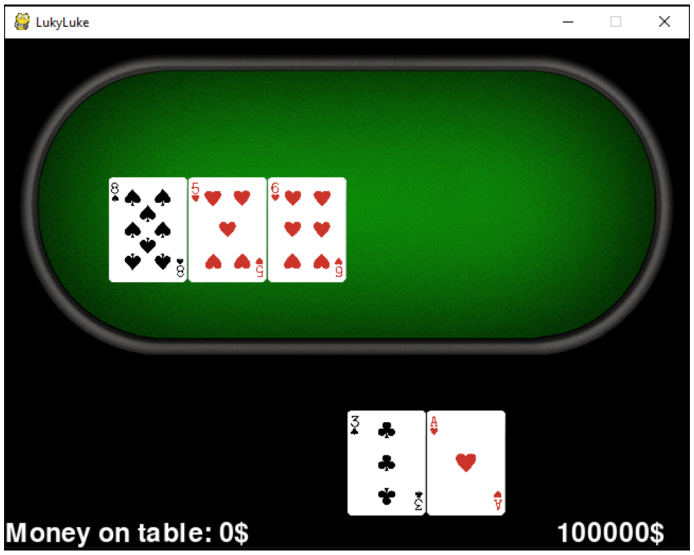

Fig. 2. Visuals of the game

When testing the bot it did bluff, as well as giving good predictions when it had good cards. Though it potentially could have been more precise if one had e.g. run fewer epochs, since overfitting can be a possible issue when training the model.

IV. Discussion

Describe the conclusions (reflect on the introduction, background and problem)

Discuss the positive effects and the drawbacks. The AI's precision when it comes to making predictions are hard to test. One of the reasons for making a poker bot with deep learning was to make it possible for it to bluff and mimic potential human behaviour. Thus it will not always play by what is statistically the most likely winning card combination and one cannot deterministically say what the expected behaviour should be.

Potential drawback is overfitting and underfitting. It's hard to say what the correct size for a data set is, whether it's enough data, and whether one runs too many epochs that the data becomes too attuned to the testing data that it cannot generalize on other data.

Describe the evaluation of the results of the degree project.

Describe valid future work. (Future work can alternatively be described in a separate chapter before the conclusion.) (Collaberation games?) The current implementation does not handle all the rules of Texas Hold 'Em, instead it mainly focuses on having an AI play against a real human and see whether the AI will make better predictions or "fool" the player.

V. Conclusion

In this paper we described the theory, design and implementation that went into making a Poker AI using deep learning. With pygame and tensorflow were we able to create a model that learned to imitate the playing style of real people using a data set.

[?]

References

[1] J. Pill. (2017) Pokertube. [Online]. Available: https://www.pokertube.com/article/the-insperable-history-of-math-and-gambling

[2] M. Moravc, M. Schmid, N. Burch, V. Lisy, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, "Deepstack: Expert-level artificial intelligence in heads-up no-limit poker," vol. 3, pp. 1–15, Mar. 2017.

[3] Comunity. Texas hold 'em. [Online]. Available: