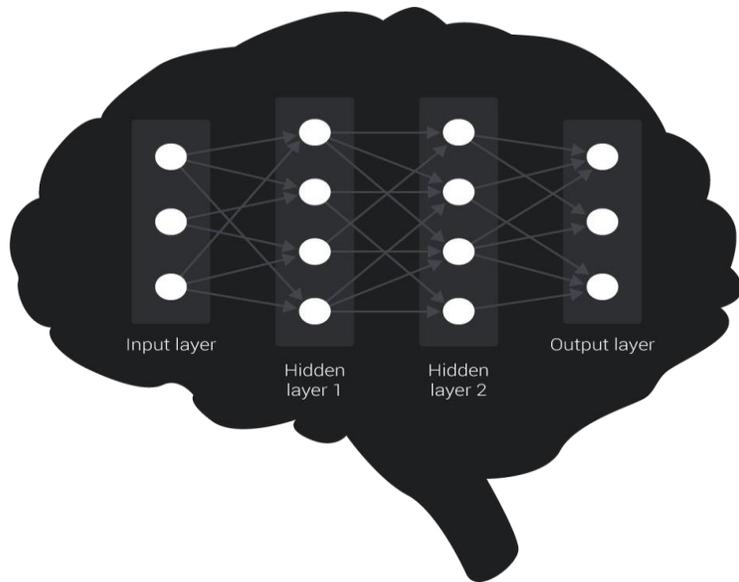


DeepStack: Expert-Level AI in Heads-Up No-Limit Poker



Surya Prakash Chembrolu



AI and Games

AlphaGo – Go

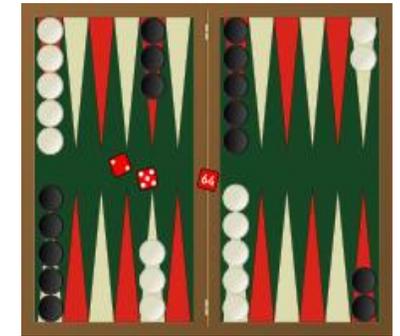
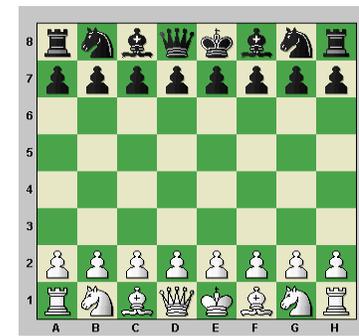
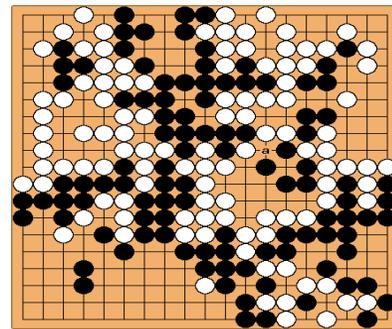
Watson – Jeopardy!

DeepBlue -Chess

Chinook -Checkers

TD-Gammon -Backgammon

Perfect Information Games



Cepheus - Heads up Limit Texas Hold'em

Deepstack - Heads up No-limit Poker

Imperfect Information Games



Heads Up **Limit** Hold'em (HULHE)

In a game played with a **fixed-limit** betting structure, a player chooses only whether to bet or not—the amount is fixed by rule in most situations. The fixed amount generally doubles at some point in the game and this double wager amount is referred to as a big bet.

- HULHE has around 10^{14} Decision points*.

Heads Up **No Limit** Hold'em(HUNL)

A game played with a **no-limit** betting structure allows each player to raise the bet by any amount up to and including their entire remaining stake at any time (subject to the table stakes rules and any other rules about raising). There is generally a minimum opening bet, and raises usually must be at least the amount of the previous raise.

- HUNL has about 10^{160} Decision points. Closer to Go which is around 10^{170} .



Some Poker Terminology

- **Pre-flop:** The first round; can refer to either the hole cards, or the betting round after these cards are distributed.
- **Flop:** The second round; can refer to either the 3 revealed public cards, or the betting round after these cards are revealed.
- **Turn:** The third round; can refer to either the 1 revealed public card, or the betting round after this card is revealed.
- **River:** The fourth and final round; can refer to either the 1 revealed public card, or the betting round after this card is revealed.
- **Showdown:** After the river, players who have not folded show their private cards to determine the player with the best hand.
- **Chip:** Marker representing value used for wagers; all wagers must be a whole numbers of chips.
- **Bet:** The first wager in a round; putting more chips into the pot.
- **Call:** Putting enough chips into the pot to match the current wager; ends the round.
- **Check:** Declining to wager any chips when not facing a bet.
- **Raise:** Increasing the size of a wager in a round, putting more chips into the pot than is required to call the current bet.
- **Fold:** Give up on the current game, forfeiting all wagers placed in the pot. Ends a player's participation in the game.
- **All-in:** A wager of the remainder of a player's stack. The opponent's only response can be call or fold.
- **Pot:** The collected chips from all wagers
- **Private cards:** Cards dealt face down, visible only to one player. Also called hole cards.
- **Public cards:** Cards dealt face up, visible to all players. Used in combination with private cards to create a hand. Also called community cards.

Hand Rankings



POKER HAND RANKINGS



ROYAL FLUSH



STRAIGHT



STRAIGHT FLUSH



THREE OF A KIND



FOUR OF A KIND



TWO PAIR



FULL HOUSE



ONE PAIR



FLUSH



HIGH CARD

Zero-Sum Games

- In a two-player zero-sum game, whatever one player wins, the other loses.

Example:

How about Rock, Paper, Scissors?

P1 \ P2	Rock	Paper	Scissors
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

- Rock, Paper, Scissors is a zero-sum game without perfect information.
- Head up No limit Poker is also a zero-sum game with information asymmetry because each player doesn't know the opponent's cards.

Nash Equilibrium

- Nash Equilibrium is a term used in game theory to describe an equilibrium where each player's strategy is optimal given the strategies of all other players.
- In the state of Nash Equilibrium no participant can gain by a changing the strategy as long as all the other participants remain unchanged.
- In two player Zero-Sum games, we also refer to Nash Equilibria as Game Theory Optimal (GTO).

Example: Prisoner's Dilemma

- The prisoner's dilemma has a single Nash equilibrium both players choosing to confess individually.

Prisoners' dilemma

		prisoner B	
		confess	remain silent
prisoner A	confess	5 years, 5 years	0 year, 20 years
	remain silent	20 years, 0 year	1 year, 1 year

Regret Matching

- Regret matching (RM) is an algorithm that seeks to minimize regret about its decisions at each step/move of a game. As the name suggests, it learns from past behaviors to inform future decisions by favoring the action it regretted not having taken previously.
- The regret of having taken a particular action in a particular situation is called a negative regret.
- Positive regret occurs when the player tracks actions that resulted in a positive outcome.

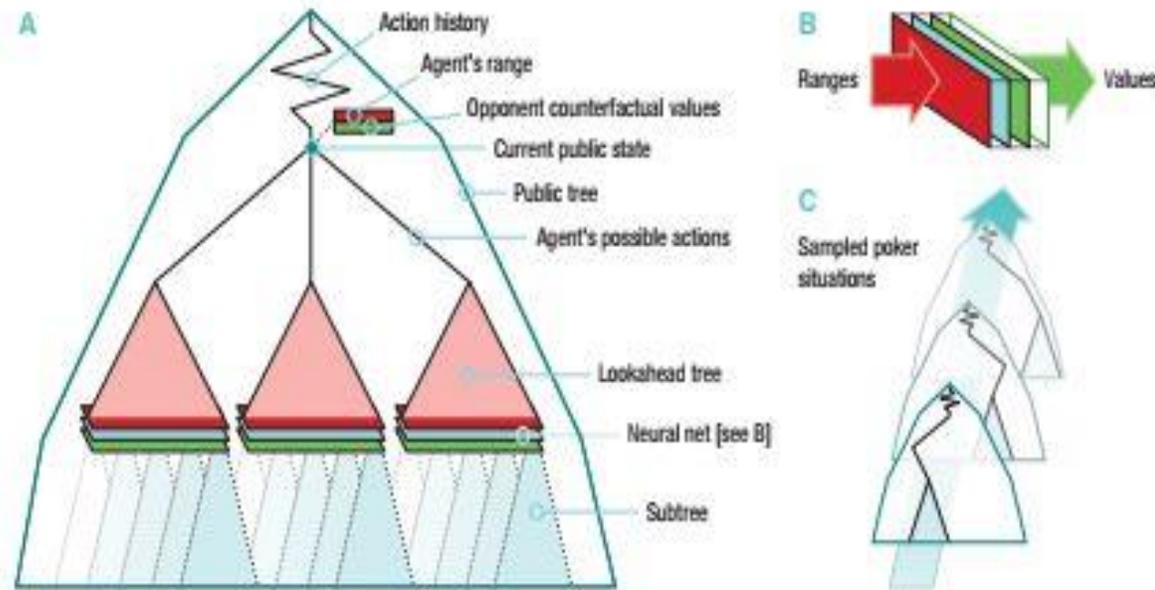
Counterfactual regret minimization(CFR)

- For two-player zero-sum games, the technique called Counterfactual Regret Minimization (CFR) computes strategies that are provably convergent to an ϵ -Nash equilibrium.
- Counterfactual regret minimization (CFR) minimizes the positive immediate counterfactual regret at each information set. Minimizing positive immediate counterfactual regret, minimizes the average overall regret.
- CFR+ uses a different regret matching scheme.
- Weights are updated after each iteration.

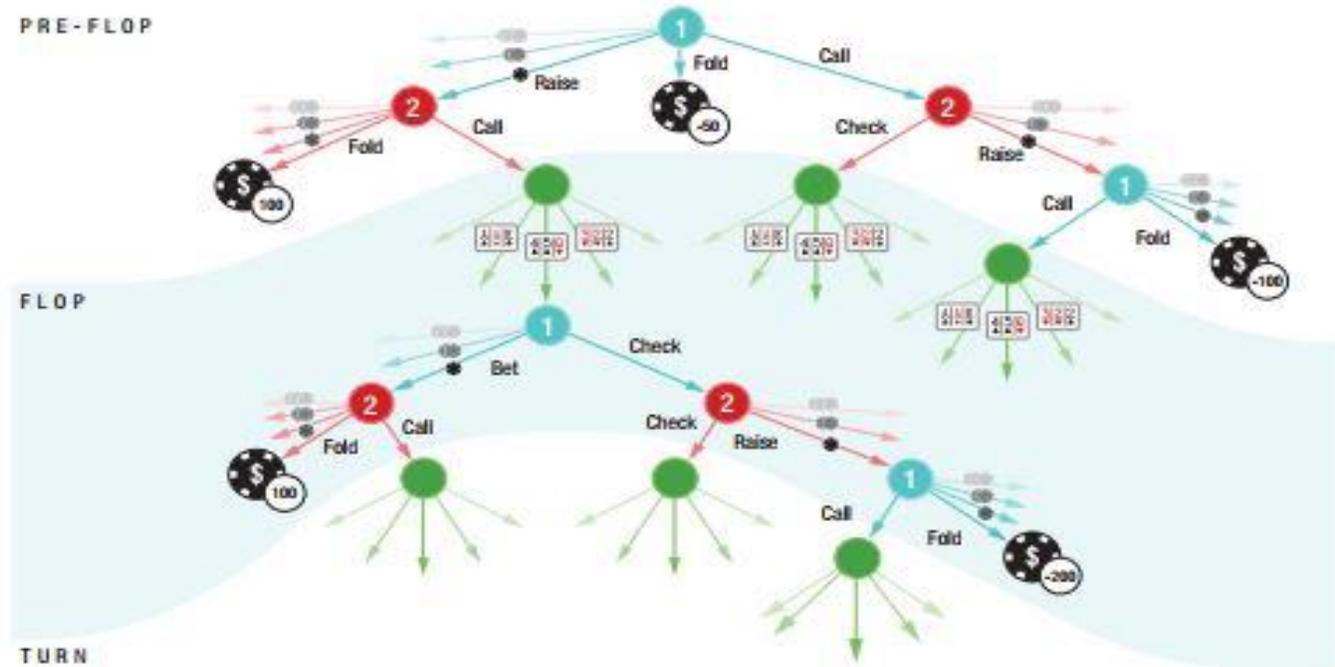
Cepheus and its Achievements

- Cepheus was the first computer program to essentially solve a game of imperfect information game that is heads-up limit Texas hold'em poker, a game with over 10^{14} information sets.
- Cepheus brought-up algorithmic advances to make the solving of HULHE possible by implementing a variant of CFR algorithm that is CFR+.
- CFR+ is Space-efficient compared to the prior and it speeds up solving too.
- CFR+ also particularly supports use of data with compression. The algorithm amortizes the compression/decompression overhead by doing a full and exact CFR update.
- Cepheus uses the approach to compress and store the values on disk, and distribute the CFR+ computation across a cluster of compute nodes.
- The architecture for Cepheus has one node processing the trunk, sending probabilities for subgames to worker nodes, and then waiting on subgame values from the worker nodes.
- CFR+ updates are run on worker nodes and these values are sent to the trunk node, and add the updated subgame to the buffer of processed subgames.

DeepStack Overview



- DeepStack reasons in the public tree always producing action probabilities for all cards it can hold in a public state. It maintains two vectors while it plays: its own range and its opponent's counterfactual values. As the game proceeds, its own range is updated via Bayes' rule using its computed action probabilities after it takes an action. Opponent counterfactual values are updated as discussed under "Continual re-solving".
- The evaluation function is represented with a neural network that takes the public state and ranges from the current iteration as input and outputs counterfactual values for both players.
- The neural network is trained prior to play by generating random poker situations (pot size, board cards, and ranges) and solving them to produce training examples.



A portion of the public tree in HUNL. Nodes represent public states, whereas edges represent actions: red and turquoise showing player betting actions, and green representing public cards revealed by chance. The game ends at terminal nodes, shown as a chip with an associated value. For terminal nodes where no player folded, the player whose private cards form a stronger poker hand receives the value of the state.

Continual Resolving

- Deepstack Computes a strategy based on current state of the game, not maintaining one for the whole game.
- Computing that strategy is achieved by continual re-solving.
- As with traditional re-solving, the re-solving step of the DeepStack algorithm solves an augmented game.
- The augmented game is designed to produce a strategy for the player such that the bounds for the opponent's counterfactual values are satisfied.
- To be able to re-solve at any public state, one should keep track of their own range and a suitable vector of opponent counterfactual values.
- These values must be an upper bound on the value the opponent can achieve with each hand in the current public state, while being no larger than the value the opponent could achieve had they deviated from reaching the public state (Minimum of maximum losses).
- In order to make continual re-solving practical, we need to limit the depth of the re-solved subtree.

Limited depth lookahead

- As in heuristic search for perfect information games limiting the depth of the subtree to reason about when re-solving is adopted in deepstack too.
- But, in imperfect information games we cannot simply replace a subtree with a heuristic or precomputed value.
As there is no information on players' ranges.
- In deepstack a modified version of CFR algorithm is used to resolve.
- The algorithm DeepStack uses to solve the augmented game is a hybrid of vanilla CFR and CFR+ ,which uses regret matching like CFR+, but does uniform weighting and simultaneous updates like vanilla CFR.
- Ranges change on each iteration of the CFR-solver.
- As mentioned before, when depth is limited for game with information asymmetry it is solvable.
- DeepStack overcomes this challenge by replacing subtrees beyond a certain depth with a learned counterfactual value function that approximates the resulting values if that public state were to be solved with the current iteration's ranges.
- The inputs to the value function are the ranges for both players, as well as the pot size and public cards, which are sufficient to specify the public state i.e, a description of a poker game: the probability distribution of being dealt individual private hands, the stakes of the game, and any public cards revealed.
- The outputs are a vector for each player containing the counterfactual values of holding each hand in that situation i.e, how valuable holding certain cards would be in such a game.
- With a depth limit of four actions(one hand), this approach reduces the size of the game for re-solving from 10^{160} decision points at the start of the game down to no more than 10^{17} decision points.

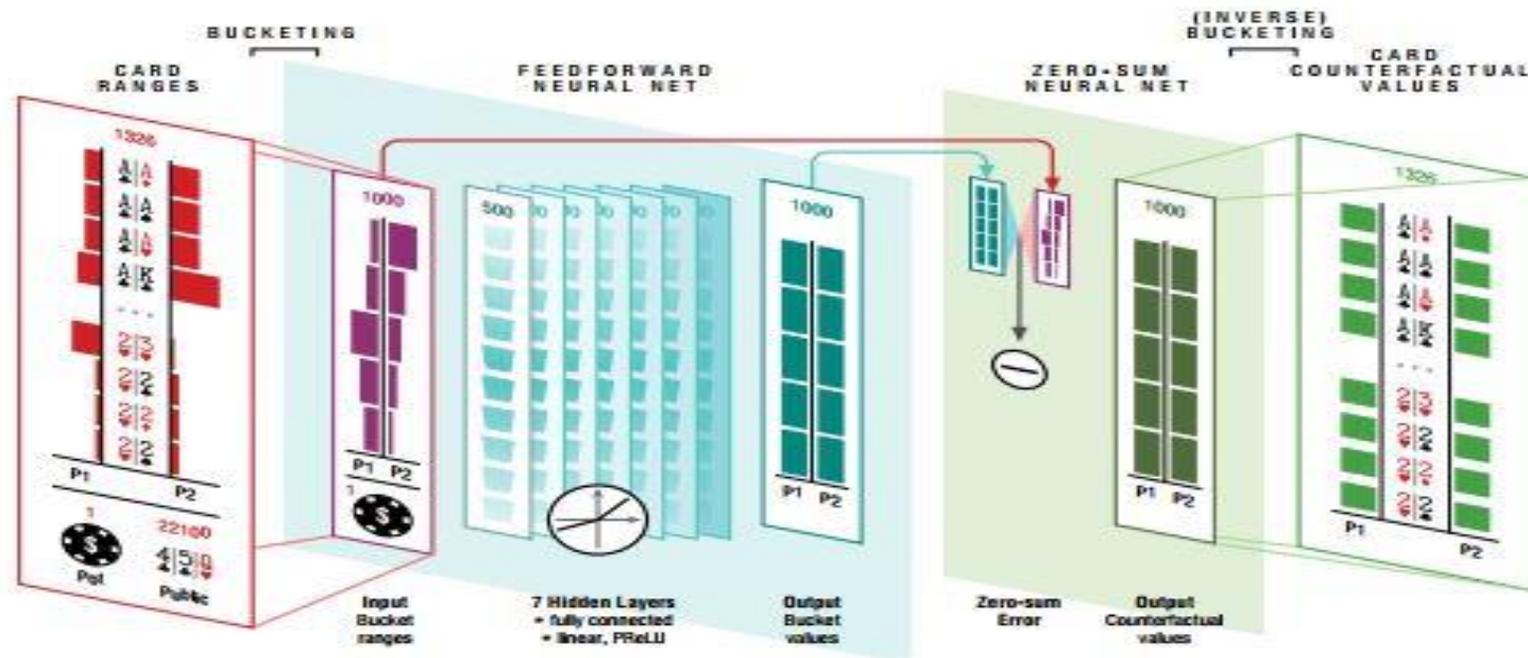
Sparse lookahead trees

- For reduction on number of actions deepstack constructs a sparse lookahead tree.
- DeepStack builds the lookahead tree using only the actions fold (if valid), call, 2 or 3 bet actions, and all-in.
- A major design goal for DeepStack's implementation was to typically play at least as fast as a human would using commodity hardware and a single GPU.
- The degree of lookahead tree sparsity and the number of re-solving iterations are the principle decisions that we tuned to achieve this goal.
- With sparse and depth-limited lookahead trees, the re-solved games have approximately 10^7 decision points.
- Which are solved under five seconds by a single GPU.



Architecture of Deep Counterfactual Value Network

- DeepStack uses a standard feedforward network with seven fully connected hidden layers each with 500 nodes and parametric rectified linear units for the output.
- The network's inputs are the pot size as a fraction of the players' total stacks and an encoding of the players' ranges as a function of the public cards.
- The output of the network are vectors of counterfactual values for each player and hand, interpreted as fractions of the pot size.
- The ranges are encoded by clustering hands into 1,000 buckets.
- There is an outer network that forces the counterfactual values to satisfy the zero-sum property.



Neural Network Training

- DeepStack uses two counterfactual value networks, one for the flop and one for the turn, as well as an auxiliary network that gives counterfactual values at the end of the pre-flop.
- The turn network was trained by solving 10 million randomly generated poker turn games. These turn games used randomly generated ranges, public cards, and a random pot size.
- The flop network was trained similarly with 1 million randomly generated flop games.
- The networks were trained using the Adam gradient descent optimization procedure with a Huber loss.
- All networks were trained using built-in Torch7 libraries.(Torch7 is a framework for machine learning based on Lua Programming language, Torch7 enables good GPU support.)
- Training used a mini-batch size of 1,000, and a learning rate 0.001.
- Later learning rate was decreased to 0.0001 after the first 200 epochs*.
- Networks were trained for approximately 350 epochs over two days on a single GPU.
- the epoch with the lowest validation loss was chosen.

* Epoch is one complete presentation of the *data set to be learned* to a learning machine.

Huber loss function

- A loss function or cost function is a function that maps an event or values of one or more variables onto a real number intuitively representing some "cost" associated with the event.

$$\text{huber}(\delta, r) = \begin{cases} \infty & \delta < 0 \\ \frac{1}{2}r^2 & 0 \leq \delta, |r| \leq \delta \\ \delta(|r| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

Parameters: The variable r refers to the residuals, that is the difference between the observed and predicted values.

Delta refers to quadratic vs. linear loss changepoint. Quadratic for small values of r and linear for large values of r .

Adam optimizer

- **Stochastic gradient descent** is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions.
- Stochastic gradient descent tries to find minima or maxima by iteration.

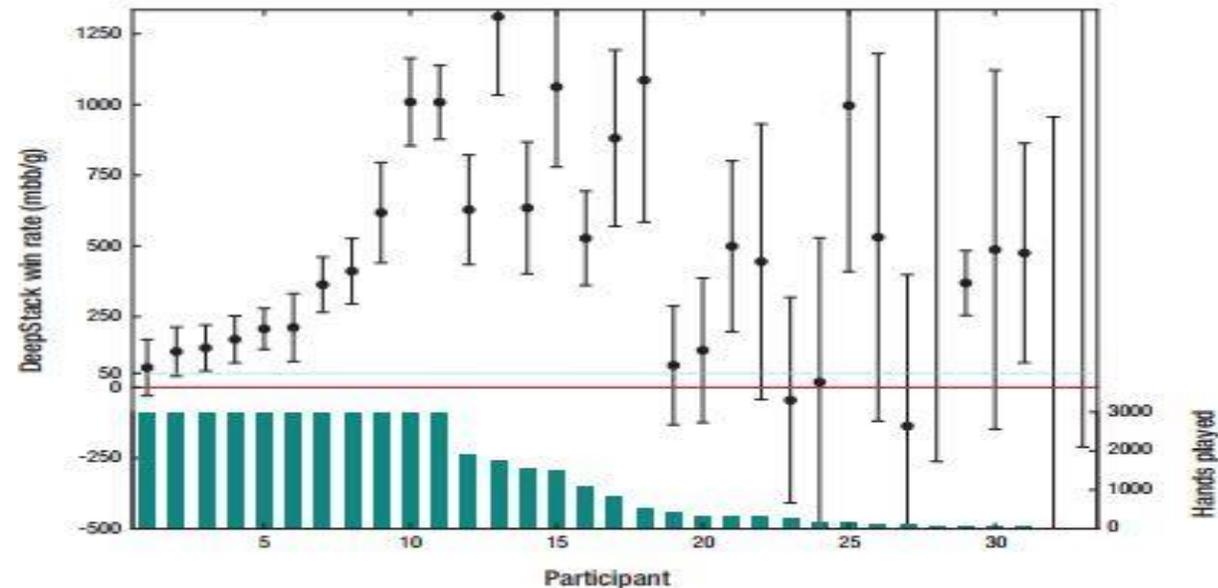
Adam's parameter update:

$$\begin{aligned}m_w^{(t+1)} &\leftarrow \beta_1 m_w^{(t)} + (1 - \beta_1) \nabla_w L^{(t)} \\v_w^{(t+1)} &\leftarrow \beta_2 v_w^{(t)} + (1 - \beta_2) (\nabla_w L^{(t)})^2 \\ \hat{m}_w &= \frac{m_w^{(t+1)}}{1 - \beta_1^t} \\ \hat{v}_w &= \frac{v_w^{(t+1)}}{1 - \beta_2^t} \\ w^{(t+1)} &\leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w + \epsilon}}\end{aligned}$$

Where, Epsilon is a small number used to prevent division by 0 and Beta1, Beta2 are the factors for gradients and second moments of gradients, respectively. Input to the optimizer are set of parameters 'w' and Loss Function 'L' and 't' is the current training iteration value.

Evaluating Deepstack

- In poker performance is measured in milli big blinds per game(mbb/g).
- Milli-big-blinds per game (mbb/g) is the average winning rate over a number of games, measured in thousandths of big blinds.
- Big blind is the forced bet made by the non-dealer before any cards are dealt.



- Deepstack performance is estimated using AIVAT as its value function estimates is perfectly suited for AIVAT.
- AIVAT is variance reduction technique for agent evaluation in imperfect information games.

Achievements of Deepstack

- DeepStack defeated professional poker players at HUNL with statistical significance, a game that is similarly sized to Go, but with the added complexity of imperfect information.
- DeepStack allows computation to be focused on specific situations that arise when making decisions and the use of automatically trained value functions.
- DeepStack brought up a new continual re-solving paradigm for resolving.
- DeepStack considers a reduced number of actions, allowing it to play at conventional human speeds.



Limitations of Deepstack

- Deepstack plays only heads-up variant that is only two players it is not extended to multiplayer variant.
- Deepstack should have a more sophisticated lookahead to restrict the opponent's exploitability in some cases.



References

- <https://www.deepstack.ai/>
- <https://arxiv.org/pdf/1701.01724.pdf>
- Cepheus Research paper link- <http://poker.cs.ualberta.ca/publications/2015-ijcai-cfrplus.pdf>
- Cepheus poker project- <http://poker.srv.ualberta.ca/>