INTELLIGENT SYSTEMS FOR
PATTERN RECOGNITION
A.Y. 2020/2021

MIDTERM 4

MASTERING THE GAME OF GO WITH DEEP NEURAL NETWORKS AND TREE SEARCH

ELIA PICCOLI

621332

HTTPS://GITHUB.COM/ELIAPICCOLI/ISPR-PROJECTS

AlphaGo

ALPHA GO – THE CHALLENGE

GO

- The game is 2500 years old and still counts over 20M players
- Usually played on 19x19, 13x13, 9x9 board
- Very easy rules but complex game

IDEA : Given a position of the board the model should be able to find the best possible move that will lead to victory

Can be solved by recursevely computing the optimal value in a search tree

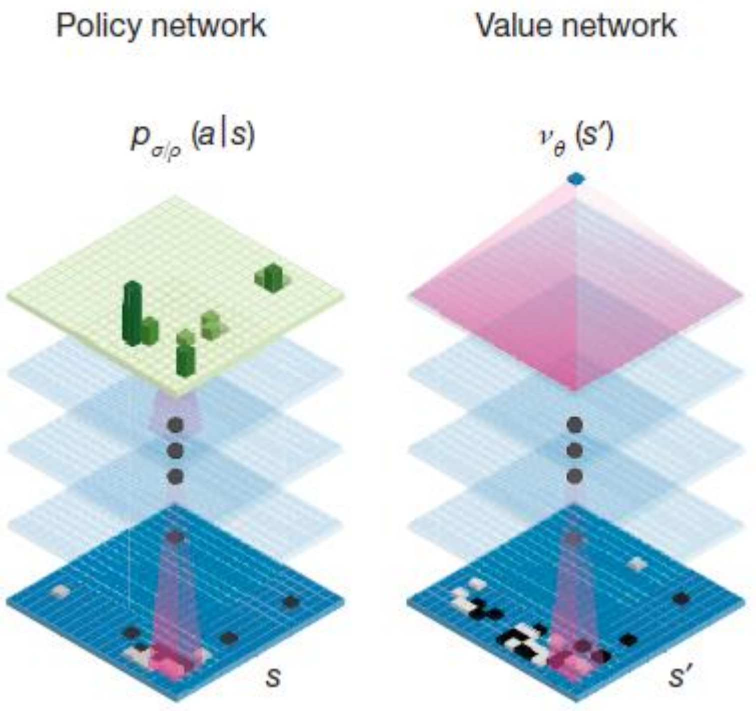
Unfeasible search space: b^d
($b \approx 250$, $d \approx 150$)

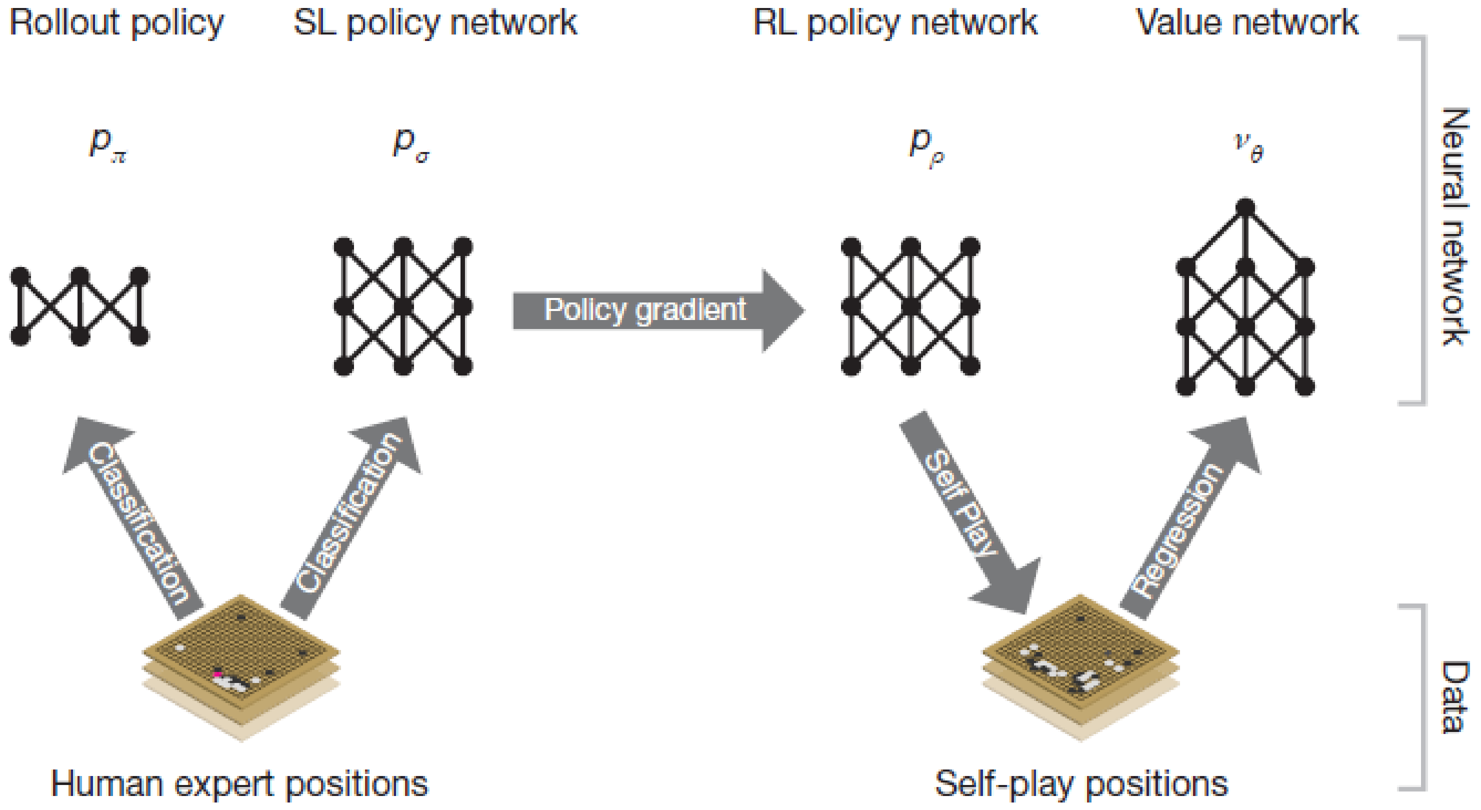
Narrow the depth of the trees using a value functon $v(s) \approx v^*(s)$

ALPHA GO – ARCHITECTURE

GO: abstraction is the key to win

CNN: abstraction is its *forte*





ALPHA GO – SL POLICY NETWORK

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0

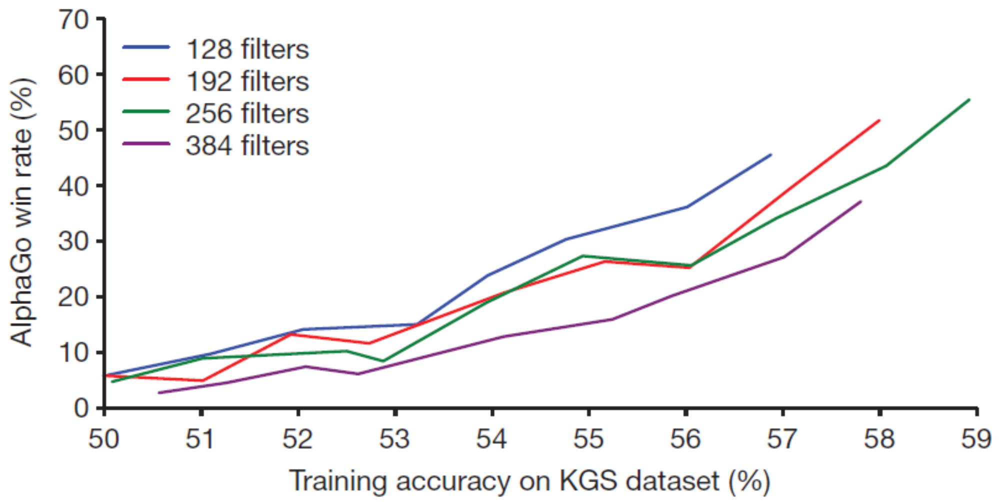
Training Data

Trained to classify positions according to expert moves played in the KGS data set. This data set contains 29.4 million positions from 160,000 games played by KGS 6 to 9 *dan* human players.

Neural Network architecture

- The input is a 19 × 19 × 48 image stack consisting of 48 feature planes
- First hidden layer pads to 23 x 23 images and then apply first convolutional layer with 192 filter of size 5 x 5 followed by rectified non-linearities
- Layer 2 to 12 pads to 21 x 21 images and then apply 192 filters of size 3x3 followed by rectified non-linearities
- Layer 13 apply 1 filter of size 1 x 1 followed by a softmax

- 3 weeks of training



ALPHA GO – FAST POLICY NETWORK

Training Data
Trained from 8 million positions from human games on the Tygem server to maximize log likelihood by stochastic gradient descent.

Neural Network architecture

- Use a small set of features as input considering ‘response’ patterns and ‘non-response’ patterns
- Simple Softmax layer

- The aim of this model is not accuracy - only 24% - but speed require just 2 μs to select an action (vs 3ms SL policy network)
- Will be exploited during MCTS to rapidly compute rollouts

Feature	# of patterns	Description
Response	1	Whether move matches one or more response pattern features
Save atari	1	Move saves stone(s) from capture
Neighbour	8	Move is 8-connected to previous move
Nakade	8192	Move matches a *nakade* pattern at captured stone
Response pattern	32207	Move matches 12-point diamond pattern near previous move
Non-response pattern	69338	Move matches 3×3 pattern around move

ALPHA GO – RL POLICY NETWORK

Training – Self Play

Each iteration consisted of a minibatch of *n* games played between the current policy network p_ρ and an opponent p_{ρ^-} randomly sampled from a pool of opponents (network at previous iteration)

Neural Network architecture

- The input is a 19 × 19 × 48 image stack consisting of 48 feature planes
- First hidden layer pads to 23 x 23 images and then apply first convolutional layer with 192 filter of size 5 x 5 followed by rectified non-linearities
- Layer 2 to 12 pads to 21 x 21 images and then apply 192 filters of size 3x3 followed by rectified non-linearities
- Layer 13 apply 1 filter of size 1 x 1 followed by a softmax

- The network is initialized with the same weights of the SL policy network
- One day of training

- The aim of this model is to fine tune the SL policy network towards a winning policy

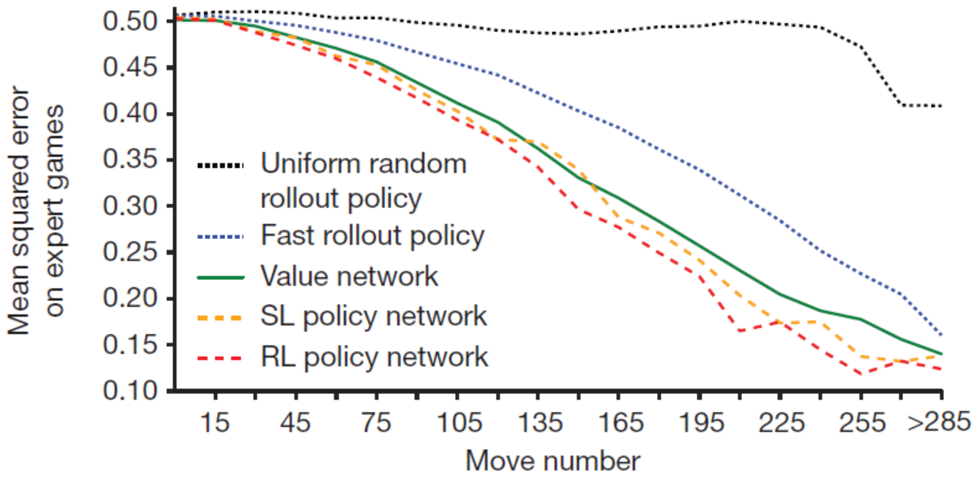
- It is trained using a sparse reward function that return *1* for victory and *-1* for defeat

- 80% winrate vs SL policy network
- 85% winrate vs Pachi

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0

ALPHA GO – VALUE NETWORK

Training Data
To avoid overfitting due to very correlated data (successive moves) is created during the phase of self-play considering one random position and the outcome of that game (s_{U+1}, z_{U+1})



Neural Network architecture
- The input is a 19 × 19 × 48 image stack consisting of 48 feature planes
- First hidden layer pads to 23 x 23 images and then apply first convolutional layer with 192 filter of size 5 x 5 followed by rectified non-linearities
- Layer 2 to 12 pads to 21 x 21 images and then apply 192 filters of size 3x3 followed by rectified non-linearities
- Layer 13 apply 1 filter of size 1 x 1 followed by a rectified non-linearities
- Layer 14 is a fully connected layer with 256 rectifier units
- Output is a fully connected layer with 1 tanh unit

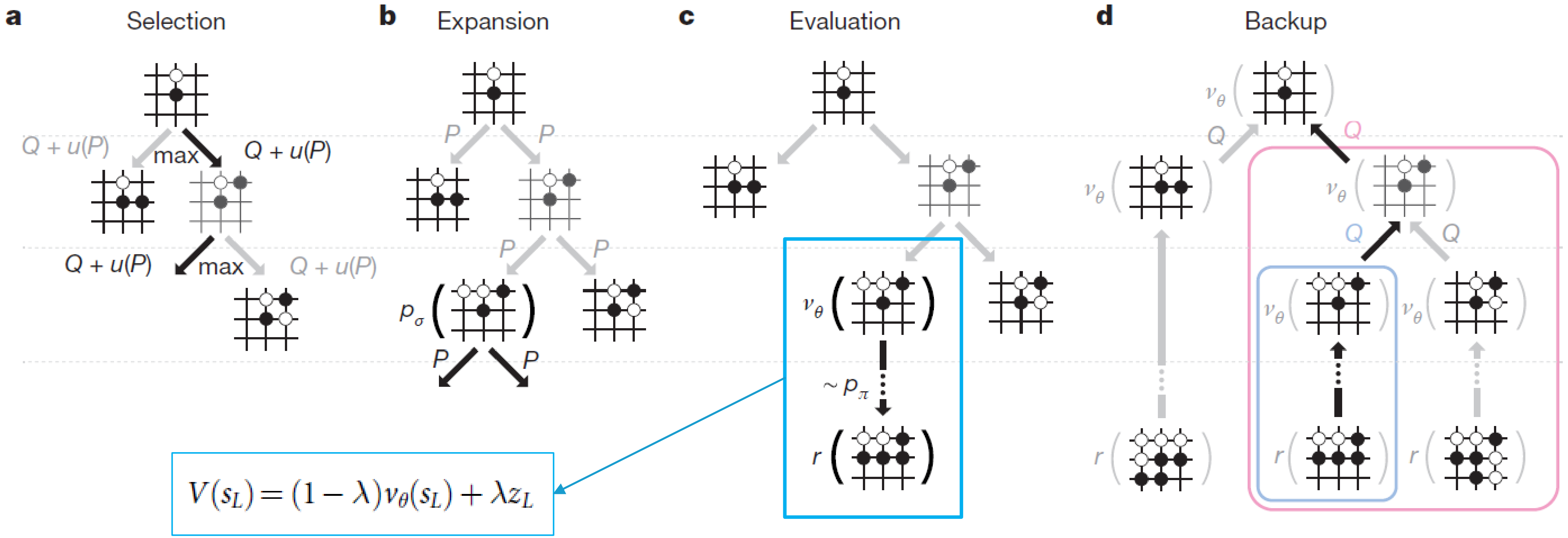
- One week of training

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

ALPHA GO – SEARCHING

- AlphaGo combines the policy and value networks in an MCTS algorithm

- Each edge (s, a) of the search tree stores an action value $Q(s, a)$, visit count $N(s, a)$, and prior probability $P(s, a)$.



ALPHA GO – EVALUATION

VS Programs

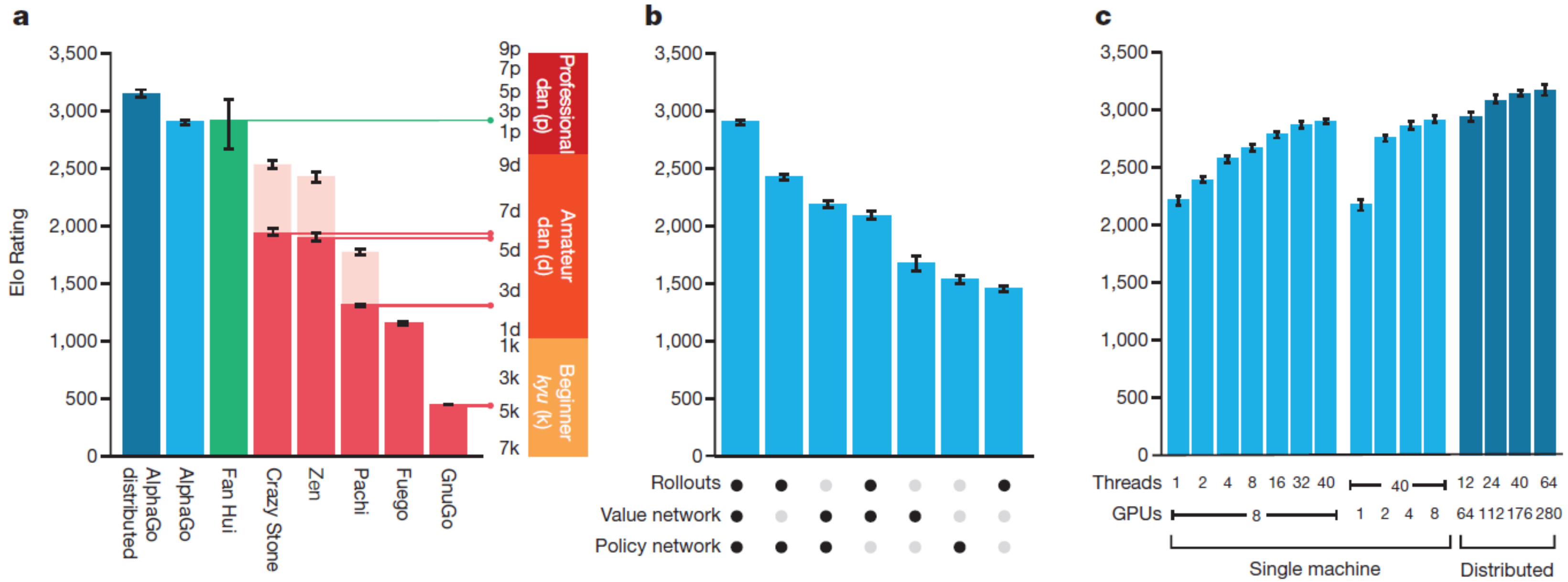
To evaluate AlphaGo, we ran an internal tournament among variants of AlphaGo and several other Go programs (CrazyStone, Zen, Pachi etc.)

- AlphaGo won 494 out 495 games
- AlphaGo won 77%, 86%, and 99% of handicap games against Crazy Stone, Zen and Pachi, respectively
- Distributed AlphaGo won 77% games versus single-machine AlphaGo

VS Humans

- 5–9 October 2015 AlphaGo and Fan Hui (2013/14/15 European Go champion) competed in a formal five-game match.
- 9-15 March 2016 AlphaGo and Lee Sedol (18-time Go world champion) competed in a formal five-game match.

- AlphaGo won 5-0 vs Fan Hui
- AlphaGo won 4-1 vs Lee Sedol



ALPHA GO – FINAL CONSIDERATIONS AND FUTURE WORK

- AlphaGo brought huge improvement with respect to the previous results obtained in GO
- Effective move selection and position evaluation functions for Go, based on deep neural networks that are trained by a novel combination of supervised and reinforcement learning.
- Differently from DeepBlue – handcrafted evaluation function - AlphaGo neural networks learns through supervised and reinforcement learning
- In 2017 DeepMind published AlphaGo Zero [1], a new version, that wins 100-0 vs Alpha Go starting from zero knowledge

I strongly suggest you to watch DeepMind's film about this huge achievement in AI world.
https://www.youtube.com/watch?v=WXuK6gekU1Y

[1] Silver, D., Schrittwieser, J., Simonyan, K. et al. Mastering the game of Go without human knowledge.