AlphaGo

By Philip Dietrich and Laurenz Hemmen

Structure

1. The Game of Go
2. DeepMind’s AlphaGo
3. Data
   a. Representations of Boards
4. Softmax Network
   a. Architecture
   b. Results
5. Convolutional Neural Network
   a. Architecture
   b. Results
6. Comparing Models
7. Conclusion

Structure

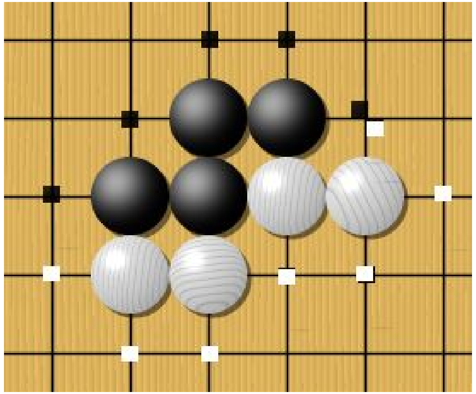
1. **The Game of Go**
2. DeepMind’s AlphaGo
3. Data
    a. Representations of Boards
4. Softmax Network
    a. Architecture
    b. Results
5. Convolutional Neural Network
    a. Architecture
    b. Results
6. Comparing Models
7. Conclusion

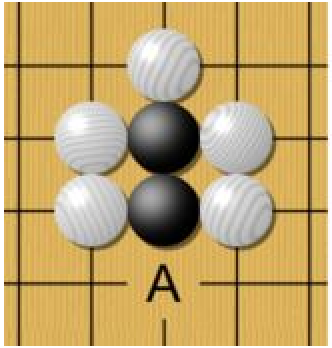
The Game of Go

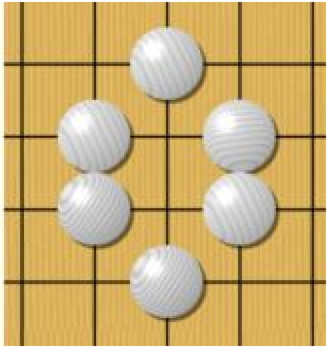
- Literally: ‘encircling game’
- Two players
- 19x19 fields
- Goal: surround more territory than opponent
- Simple rules, though extremely difficult

The Rules of Go

- Players place stones on vacant points (alternately)
- Adjacent stones of one color form a group
- Liberty of a group: number of vacant adjacent points (group shares liberties)
- If a group has no liberty (gets captured) it is removed from the board



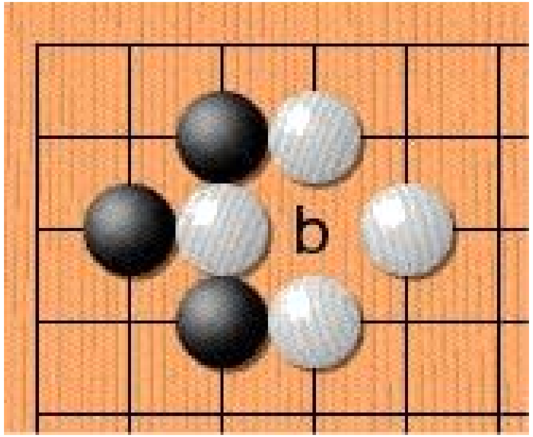




Images from: https://en.wikibooks.org/wiki/Go/Lesson_2:_Basic_Rules_and_Foundational_concepts

The Rules of Go

- Previous board positions are not allowed to be repeated (Ko-Rule)
- score = number of stones on board + intersections surrounded by stones
- Players can skip
- Game ends if both players skip/ one player gives up



Structure

1. The Game of Go
2. **DeepMind’s AlphaGo**
3. Data
   a. Representations of Boards
4. Softmax Network
   a. Architecture
   b. Results
5. Convolutional Neural Network
   a. Architecture
   b. Results
6. Comparing Models
7. Conclusion

DeepMind’s AlphaGo - Go vs. Chess

	Go	**Chess**
approx. #(possible moves)	250	35
#(moves in a game)	150	80
Estimated number of different games	10^{761}	10^{120}

Monte Carlo Tree Search

- Used by strong amateur level Go AIs (Fuego, Pachi, Zen, ...)
- Basic Idea:
    - randomly simulate many games (to the end)
    - remember how often node has been visited, how often has that led to a win
    - direct random selection to favor nodes that led previously to wins
- Compared to Deep Blue no knowledge of game necessary

AlphaGo

- SL (rollout) policy network
  - Networks trained on human expert positions to predict next human move
  - One fast softmax network, one big convolutional network (**later more!**)
- RL policy network
  - Playing against different versions of itself
- Value network
  - Predicting probability of a Win
- Tree Search
  - Combining outputs of networks to evaluate value of position
  - Rollout network used to traverse game tree

Structure

1. The Game of Go
2. DeepMind’s AlphaGo
3. **Data**
    a. Representations of Boards
4. Softmax Network
    a. Architecture
    b. Results
5. Convolutional Neural Network
    a. Architecture
    b. Results
6. Comparing Models
7. Conclusion

Dataset

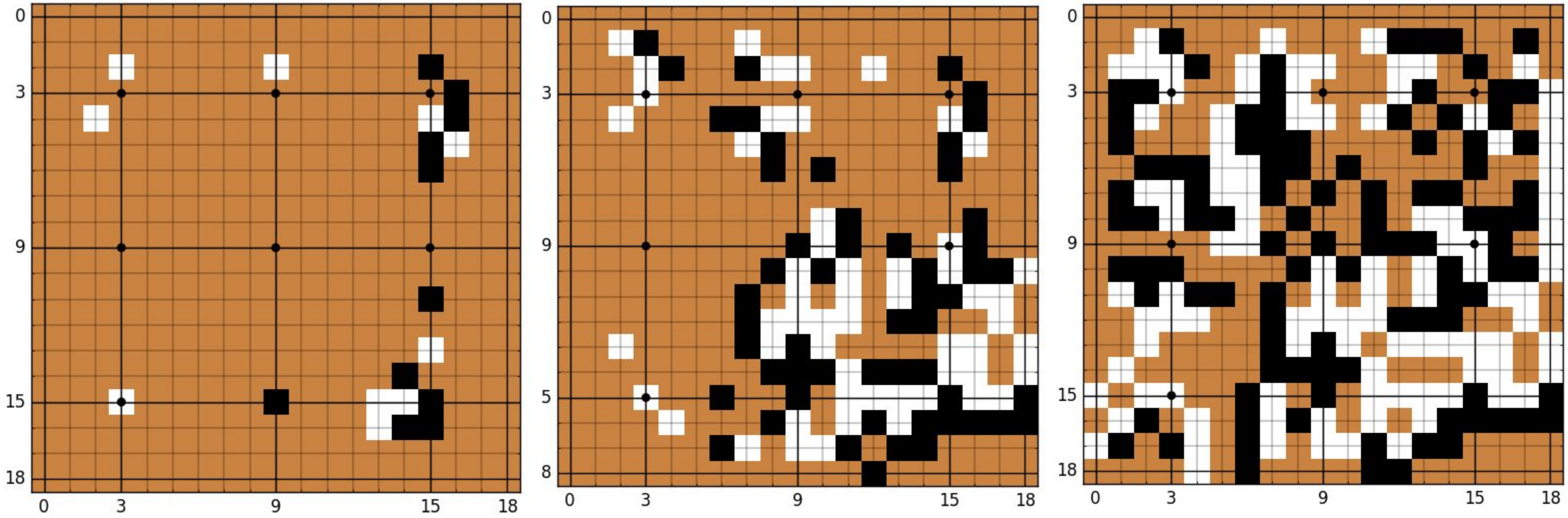
- Human Go games from KGS Go Server from games between 2001 and 2007
- Roughly 2.000.000 positions for training (95%) and validation (5%)
- 8000 position for final testing

Representation of data: as feature planes

- Encoding Information on feature planes
  - Positions of own/opponent’s stones
  - Liberties of stones
  - if move would be illegal due to ko rule
  - Special patterns
  - Distance to last move, ...
- One plane encodes one of the information for every field on the board

Representation of data: as image

- Own stone: 1, opponent stone 0, vacant field 0.5



Structure

1. The Game of Go
2. DeepMind's AlphaGo
3. Data
    a. Representations of Boards
4. **Softmax Network**
    a. Architecture
    b. Results
5. Convolutional Neural Network
    a. Architecture
    b. Results
6. Comparing Models
7. Conclusion

Softmax Network: Features and Architecture

Features are all binary for each of the 19x19 squares:

- Colors of stones
- Liberties of stones
- Forbidden fields due to Ko rule
- Neighbors to last move
- Certain 3x3 patterns around candidate moves

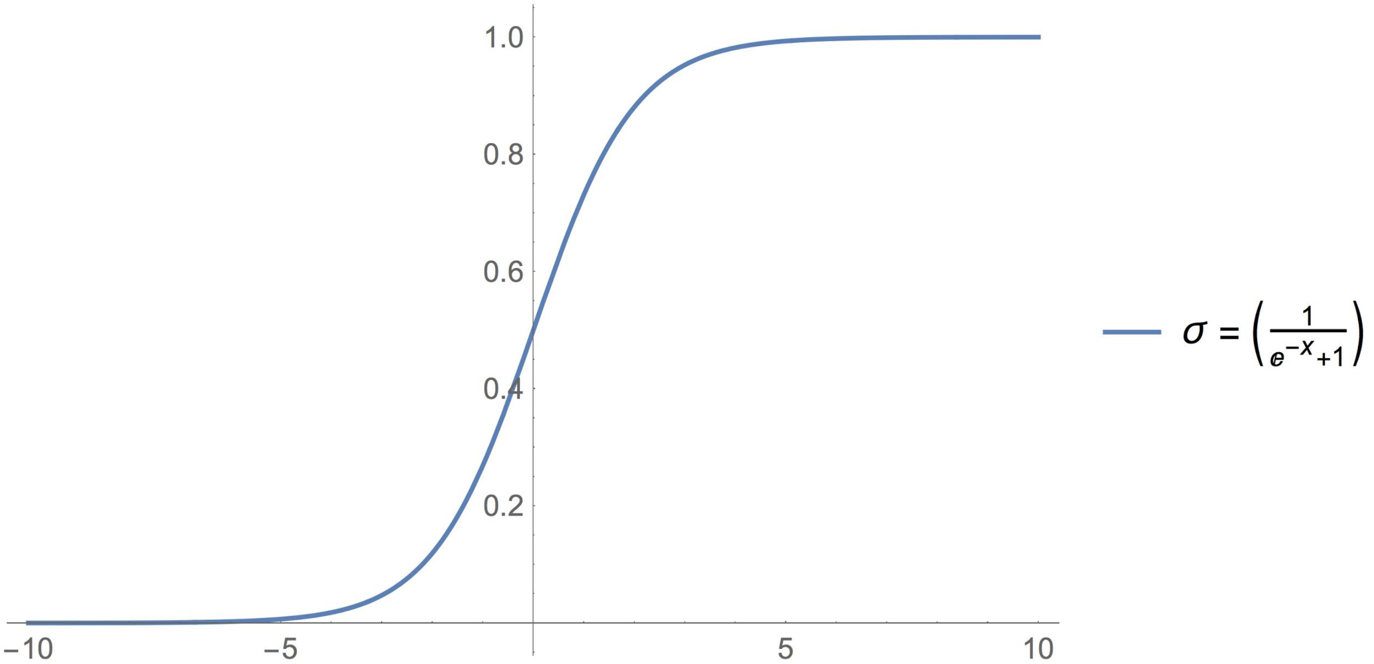
⇒ Input size: 19 x 19 x (#feature planes)

⇒ Output size: 19 x 19 = 361

⇒Number of parameters: >1.000.000 for 8 feature planes

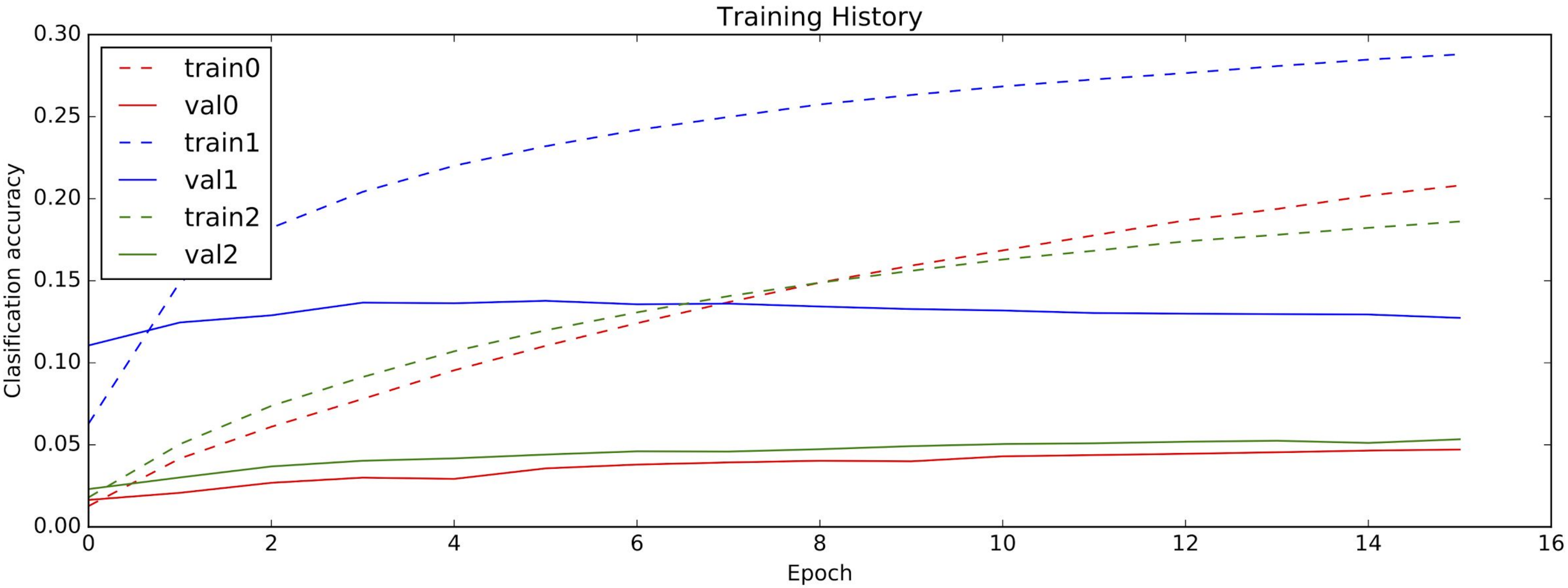
Softmax: Activation function

Reminder:

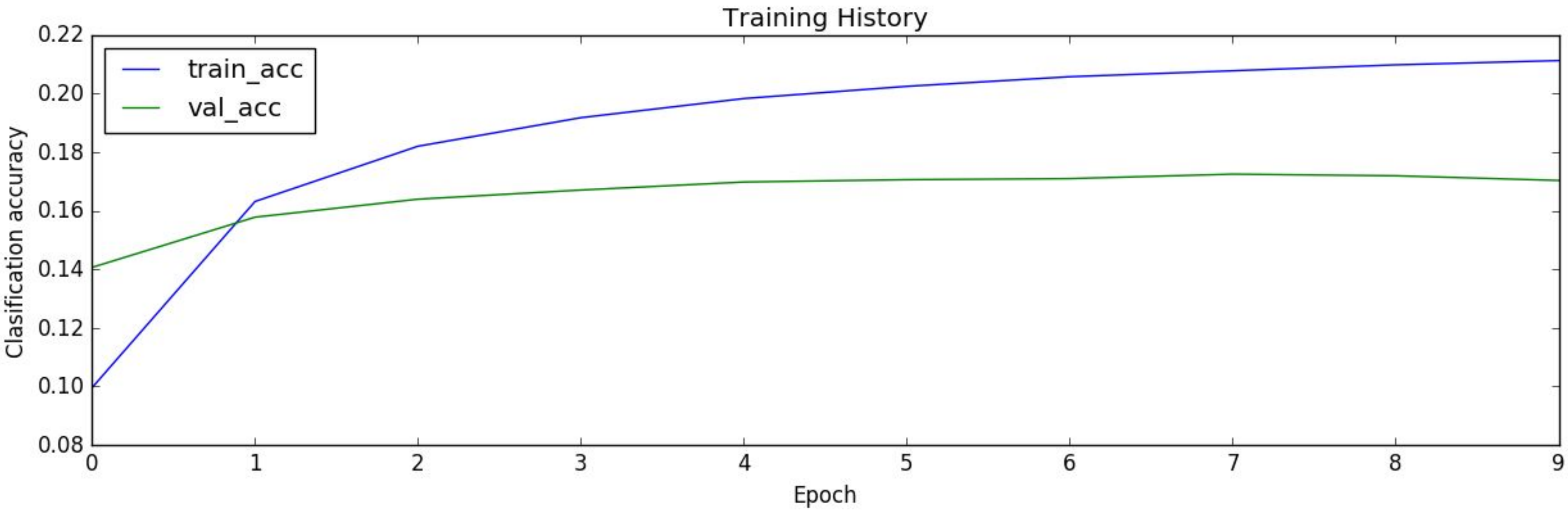


For small and huge values, gradient nearly vanishes during backprop ⇒ no/small parameter update

Results: Softmax networks



Results: Softmax networks



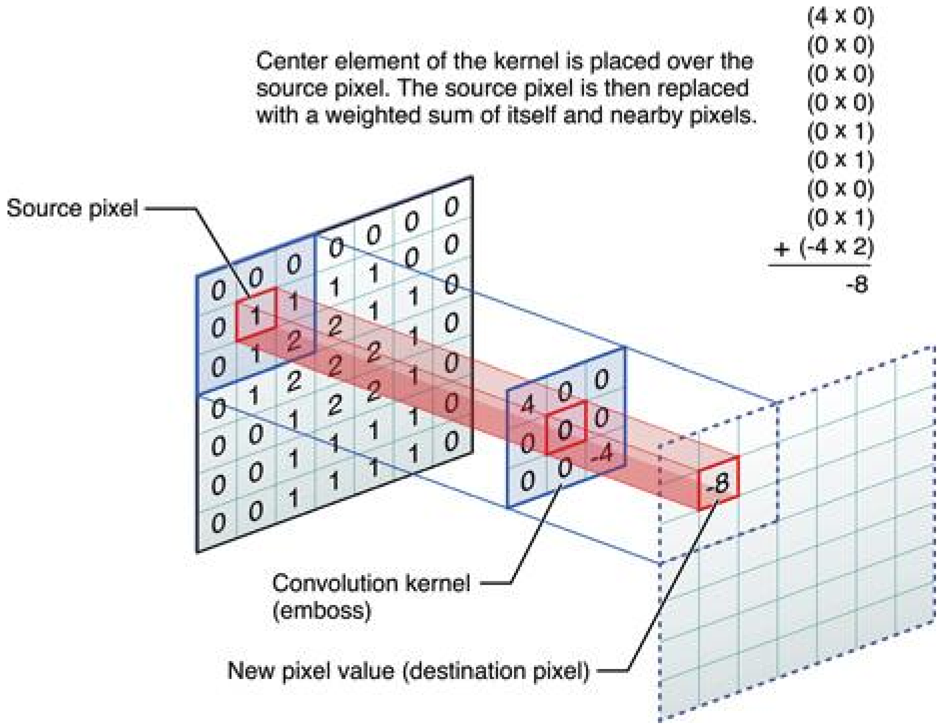
Softmax: Different features compared

	Test Accuracy [%]	time/prediction [ms]
Col+Lib, ko	6.7	0.11
Col, Lib	7.1	0.10
Col, Lib, Neighbors	17.1	0.13
Neighbors	6.7	0.11
AlphaGo's Softmax	24.2	-
Baseline: Guessing	>0.3 (=100/361)	-

Structure

1. The Game of Go
2. DeepMind's AlphaGo
3. Data
   a. Representations of Boards
4. Softmax Network
   a. Architecture
   b. Results
5. **Convolutional Neural Network**
   a. Architecture
   b. Results
6. Comparing Models
7. Conclusion

Convolutional network

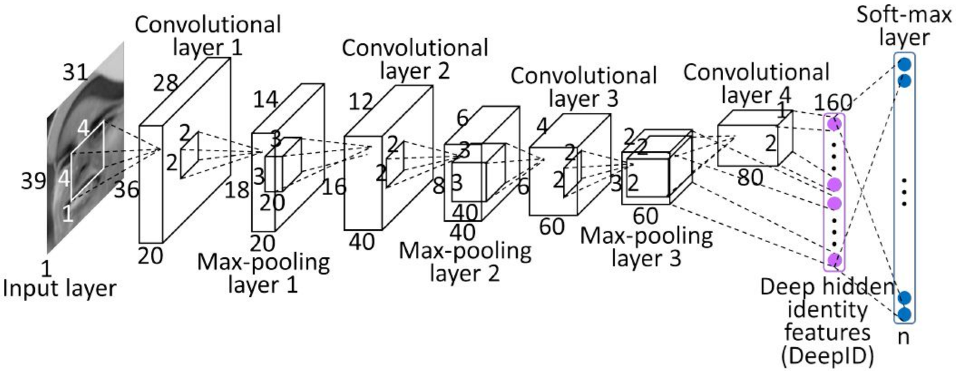


Taken from:

https://developer.apple.com/library/content/documentation/Performance/Conceptual/vImage/ConvolutionOperations/ConvolutionOperations.html

Convolutional network

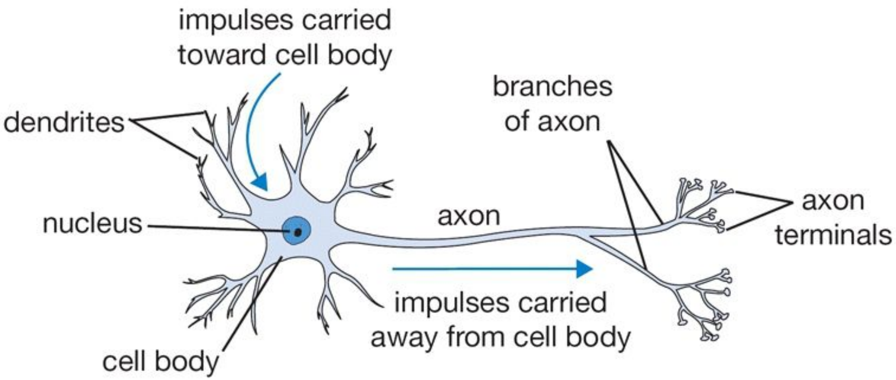
- Moves filter over image pixels
- Certain shapes give stronger output
- Normally several filters applied to one image
- Hyperparameters: number and size of filters
- Parameter determined by backpropagation: weights of filters

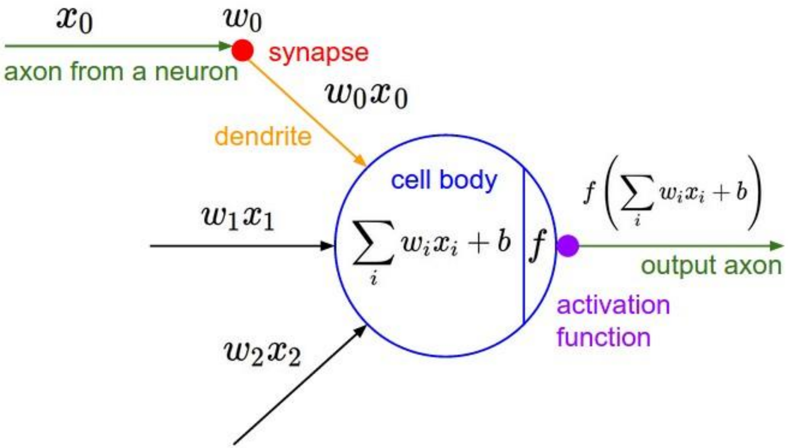


Taken from:
http://blog.csdn.net/stdcoutzyx/article/details/42091205

Convolutional network

- So far: fully convolutional, in AlphaGo: additional nonlinearities
- Biological view: (taken from http://cs231n.github.io/neural-networks-1/#bio)

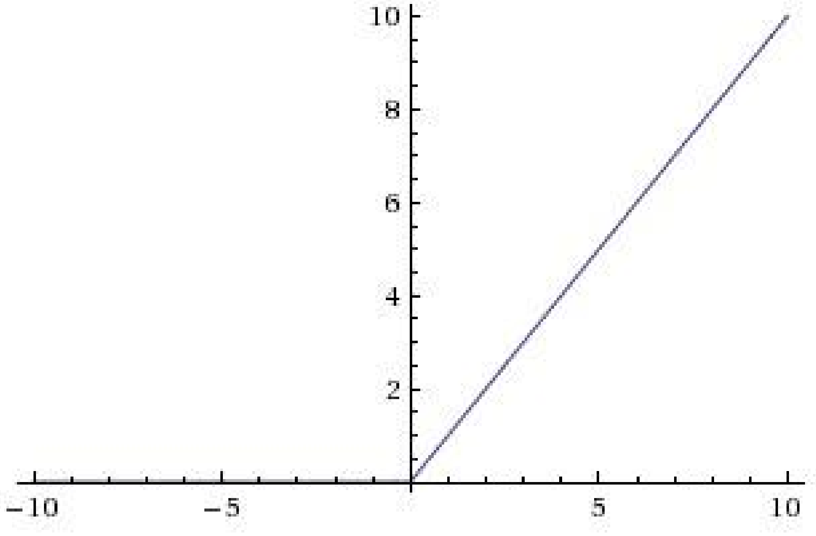




Convolutional network

Activation function used here: Rectified Linear Unit (ReLU)

- Faster to compute than tanh, sigmoid
- No saturation in positive spectrum
- Threshold at zero



taken from: http://cs231n.github.io/neural-networks-1/#bio

Convolutional Networks

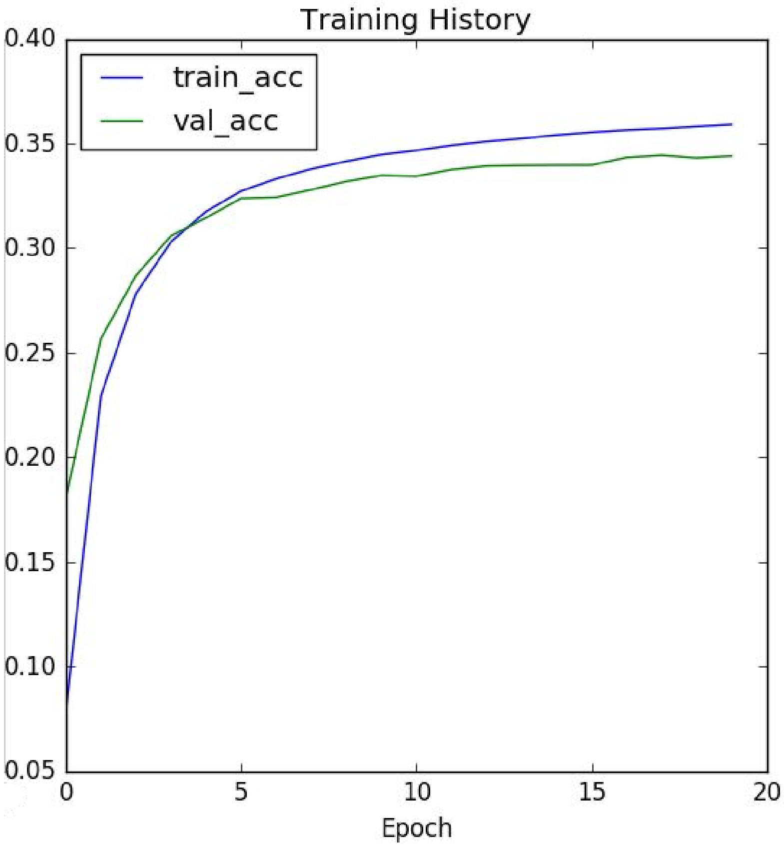
Network		Accuracy one board layer [%]	Accuracy 3 board layers [%]	Accuracy 7 board Layers [%]	Accuracy Col + Lib, Ko [%]
Convolutional - ReLu - Softmax	4 filters	4.0	10.3	-	6.7
	32 filters	3.8	14.2	26.4	7.2
Convolutional - Softmax	4 filters	1.3	-	-	7.8
	32 filters	1.5	-	-	8.5
AlphaGo	192 filters/layer	-	55.7	-	57.0

Convolutional Networks: More Conv Layers

Approach from DeepMind Paper:

- Convolutional Layer with k 5x5 filters
- Convolutional Layer with k 3x3 filters
- Convolutional Layer with 1 1x1 filter
- Softmax-Layer

- AlphaGo: k = 196, we: k = 32
- Number of parameters ≈145.000
- ⇒ no overfitting!
- 7 input layers: 34.4 % accuracy
- 3 input layers: 17.9 % accuracy



Convolutional Networks: More Conv Layers

- Even more layers:
  - 6 Convolutional layers (48 * 5x5, 32*5x5, 32* 5x5, 32*3x3, 8*3x3, 1*1x1)
  - ReLu activation function on every layer
  - One softmax layer in the end
- 214763 Parameters
- Used 1.000.000 training and validation samples
- Final test accuracy: 39.1% (training accuracy 43.0%)

⇒ best accuracy we could get

Convolutional networks

- Many more variations of #filters, sizes of filters,... tested
  - More filters [4, 8, 16, 32] result in higher accuracy, but more overfitting
  - Filter sizes [3x3, 5x5, 7x7] cause nearly no difference, 5x5 slightly better
  - More training data resulted in significantly higher accuracies
- Only small variations, results at similar accuracies
- General impression: the more layers used, ...
  - the longer it took to train the network
  - the easier it was to overfit the data to to high number of parameters ⇒ more data needed than we actually had (AlphaGo: 30 Mio, we: 2 Mio)
  - the longer the prediction time

Structure

1. The Game of Go
2. DeepMind’s AlphaGo
3. Data
    a. Representations of Boards
4. Softmax Network
    a. Architecture
    b. Results
5. Convolutional Neural Network
    a. Architecture
    b. Results
6. **Comparing Models**
7. Conclusion

Comparison: Softmax vs. ConvNet

	Convolutional Networks							Softmax		
	Conv - ReLU - SM			Conv - SM		CRCR CRS	$(CR)^6$ S			
	8p, 4f	8p, 32f	3p, 32f	8p, 4f	8p, 32f	7p, 32f	13p,48f	6 p	7 p	8 p
Runtime [ms]	0.24	0.24	0.24	0.13	0.21	0.35	**0.54**	0.06	**0.06**	0.08
Accuracy [%]	6.7	7.2	14.2	7.8	8.5	34.4	**39.1**	7.1	**17.1**	6.7

Structure

1. The Game of Go
2. DeepMind's AlphaGo
3. Data
    a. Representations of Boards
4. Softmax Network
    a. Architecture
    b. Results
5. Convolutional Neural Network
    a. Architecture
    b. Results
6. Comparing Models
7. **Conclusion**

Conclusion: Problems during Training

- Overfitting to training data due to high number of parameters in model compared to number of training positions
- memory/ processing limitations
- Very long training times (often: training time > 10 hours)

Conclusion: How well does it play?

- Doesn’t “know” the rules, can’t pass ⇒ can’t really play
- Only tries to predict next human move, not necessarily best move ⇒ AlphaGo uses additional RF Learning model playing against versions of itself
- Even if most moves are predicted correctly, false predictions can give opponent opportunity to easily win the game
- May create uncommon positions, isn’t trained on these

Conclusion: Possible further improvements

- Symmetry
  - Using mirrored board positions
  - Forcing weights to be symmetrical
- Add Go rules and already occupied fields
- Add additional layers (Dropout, more Convolutions, ..)
- Use more data (we: 2e6 vs. AlphaGo: 30e6 positions)
- Additional information / feature planes
  - Possible captures, stone saves
  - Matching certain kind of patterns
    - around last move
    - Around every position

Conclusion: Possible further improvements

Extended Data Table 4 | Input features for rollout and tree policy

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
Self-atari	1	Move allows stones to be captured
Last move distance	34	Manhattan distance to previous two moves
Non-response pattern	32207	Move matches 12-point diamond pattern centred around move

Features used by the rollout policy (first set) and tree policy (first and second set). Patterns are based on stone colour (black/white/empty) and liberties (1, 2, ≥3) at each intersection of the pattern.

Conclusion

- Our network wasn't nearly as good as AlphaGo's
- Convolutional networks gave better results
- Softmax networks were faster
- Best network architecture: some Convolutional layers (decreasing in size, #filters) , finally one softmax layer
    - Relatively low number of parameters (in the order 1e6) ⇒ data less overfitted
    - Better results than using fully connected layers

Questions?

Bibliography

[1] Silver, Huang, Maddison, Guez; Mastering the game of Go with deep neural networks and tree search

[2] Christopher Clark, Amos Storkey: Training Deep Convolutional Neural Networks to Play Go