Computing Science (CMPUT) 455

Search, Knowledge, and Simulations

James Wright

Department of Computing Science
University of Alberta
james.wright@ualberta.ca

Fall 2021

455 Today - Lecture 20

- Convolutional neural networks (CNN)
- Deep CNN
  - For image recognition
  - For move prediction in Go

Coursework

- Work on Assignment 4 (now available on website)
- Reading parts of first AlphaGo paper
- Watch tutorial and review slides, Rich Sutton, RL tutorial
- Quiz 11: Neural networks and deep learning. Double length.

Quiz 10 Review

- Quiz 10,
  Machine Learning Intro & Learning w/Simple Features
- 74 attempts. Average grade: 89%
- Lowest scores: Q1: 74%, Q8: 75%, Q10: 75%, Q17: 70%

Quiz 10 Review

Q1 Assume we have already learned a move generation policy. Then it is easy to convert this policy into a state evaluation function.

Quiz 10 Review

Q1 Assume we have already learned a move generation policy. Then it is easy to convert this policy into a state evaluation function.

- *False.*
- Not easy, as far as I know this is an unsolved problem
- The other direction is easy
- Get a move evaluation from a state evaluation by 1-ply search

Quiz 10 Review

Q1 Assume we have already learned a move generation policy. Then it is easy to convert this policy into a state evaluation function.

- *False.*
- Not easy, as far as I know this is an unsolved problem
- The other direction is easy
- Get a move evaluation from a state evaluation by 1-ply search

Q8 In linear regression we try to fit the training data as well as possible to a given line.

Quiz 10 Review

Q1 Assume we have already learned a move generation policy. Then it is easy to convert this policy into a state evaluation function.

- *False.*
- Not easy, as far as I know this is an unsolved problem
- The other direction is easy
- Get a move evaluation from a state evaluation by 1-ply search

Q8 In linear regression we try to fit the training data as well as possible to a given line.

- *False.* It's the other way around: we try to fit a line as well as possible to the given training data.

Quiz 10 Review

Q10 In a linear model, weights represent the strength of connection between features.

Quiz 10 Review

Q10 In a linear model, weights represent the strength of connection between features.

- *False.* Weights represent the importance of the features themselves, with an implicit assumption of independence.

Quiz 10 Review

Q10 In a linear model, weights represent the strength of connection between features.

- *False.* Weights represent the importance of the features themselves, with an implicit assumption of independence.

Q17 Given a Go position p. The goal of learning feature weights is to find the move from p which has the single feature with the highest value.

Quiz 10 Review

Q10 In a linear model, weights represent the strength of connection between features.

- *False.* Weights represent the importance of the features themselves, with an implicit assumption of independence.

Q17 Given a Go position p. The goal of learning feature weights is to find the move from p which has the single feature with the highest value.

- *False.* The goal is to learn weights such that the best move will have the heighest combination of feature values, e.g.,

$$\text{strength}(m) = \sum_i w_i f_i(m)$$

or

$$\text{strength}(m) = \prod_{i:f_i(m)=1} w_i.$$

Convolutional Neural Networks (CNN)

Convolutional Neural Networks

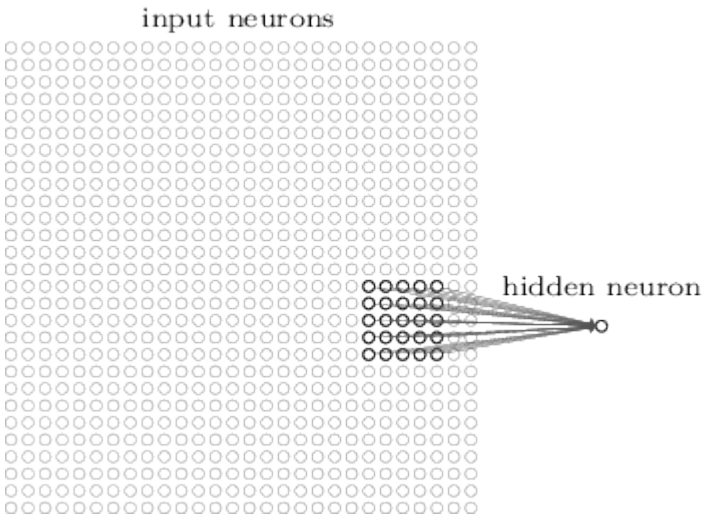


Image source: `http://neuralnetworksanddeeplearning.com/chap6.html`

- Hugely successful in image recognition tasks
- Later applied to move prediction in Go
- Main ideas:
  - start learning locally
  - Higher layers can learn successively more globally
- Each neuron depends only on a small, closely related group of neurons on the previous layer
  - “Local receptive field”

Local Receptive Field (or Filter)

- Typical sizes 5×5, 3×3
- Compare with 3×3 simple pattern features in Go
- First hidden layer: represent many simple *local features*
- Main difference to learning with simple features:
  - The network itself decides what the features are!

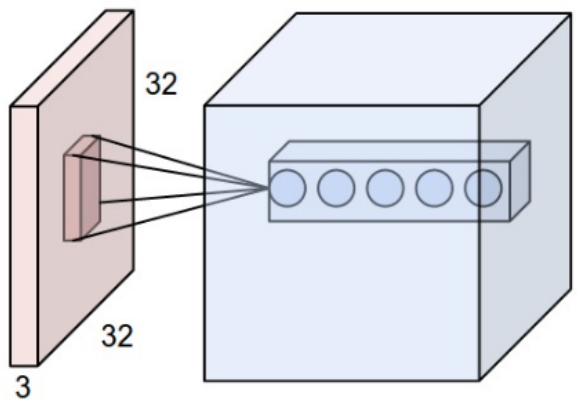


Image source: `http://cs231n.github.io/convolutional-networks`

Weight Sharing

- For both image recognition and in Go:
- Want to learn a universally useful set of local features
- Idea: share weights across *all neurons*
- Each neuron on the same level computes the same function...
- ...but with different input variables
  (different receptive fields)
- Same idea in simple feature learning
  - Learned one weight for one 3×3 pattern
  - Same weight used everywhere on the board

Convolution Kernel

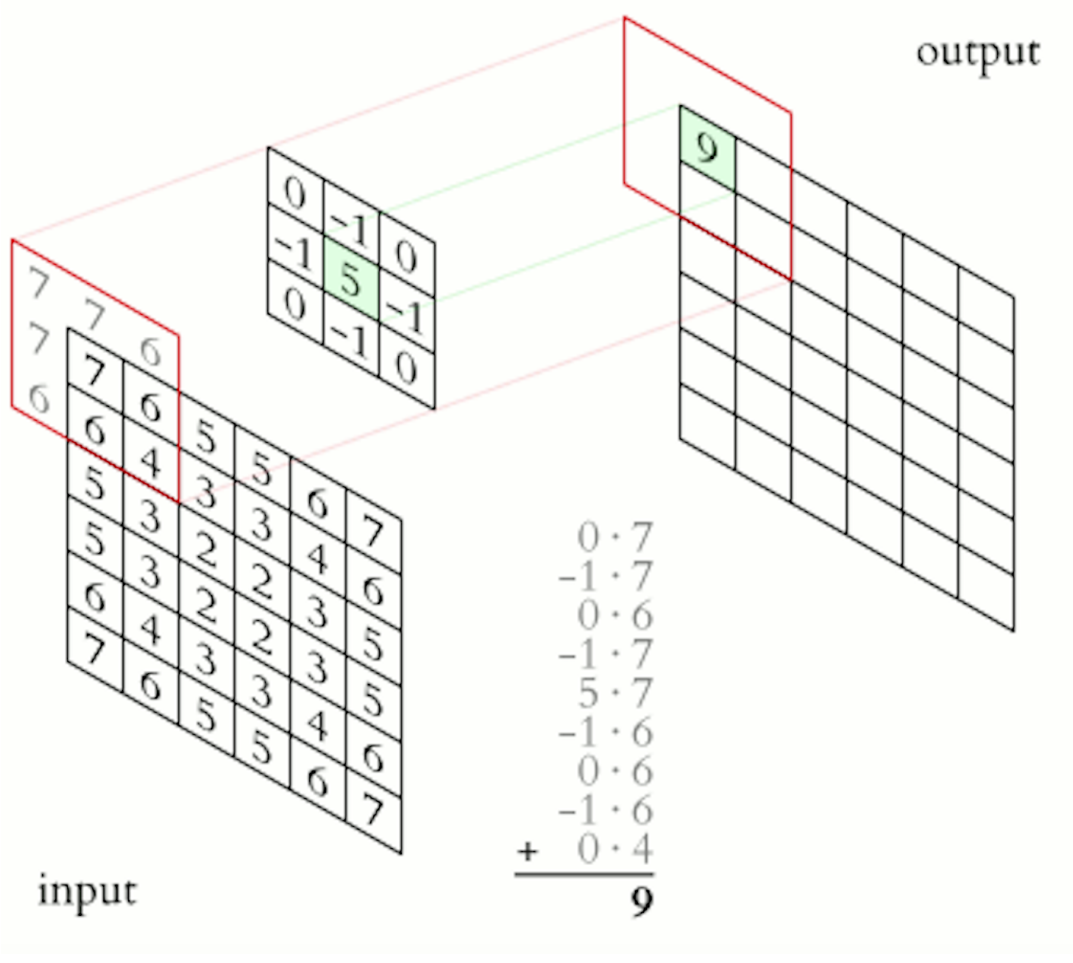


Image source:

https://commons.wikimedia.org/wiki/File:3D_Convolution_Animation.gif

Convolution with Three Kernels

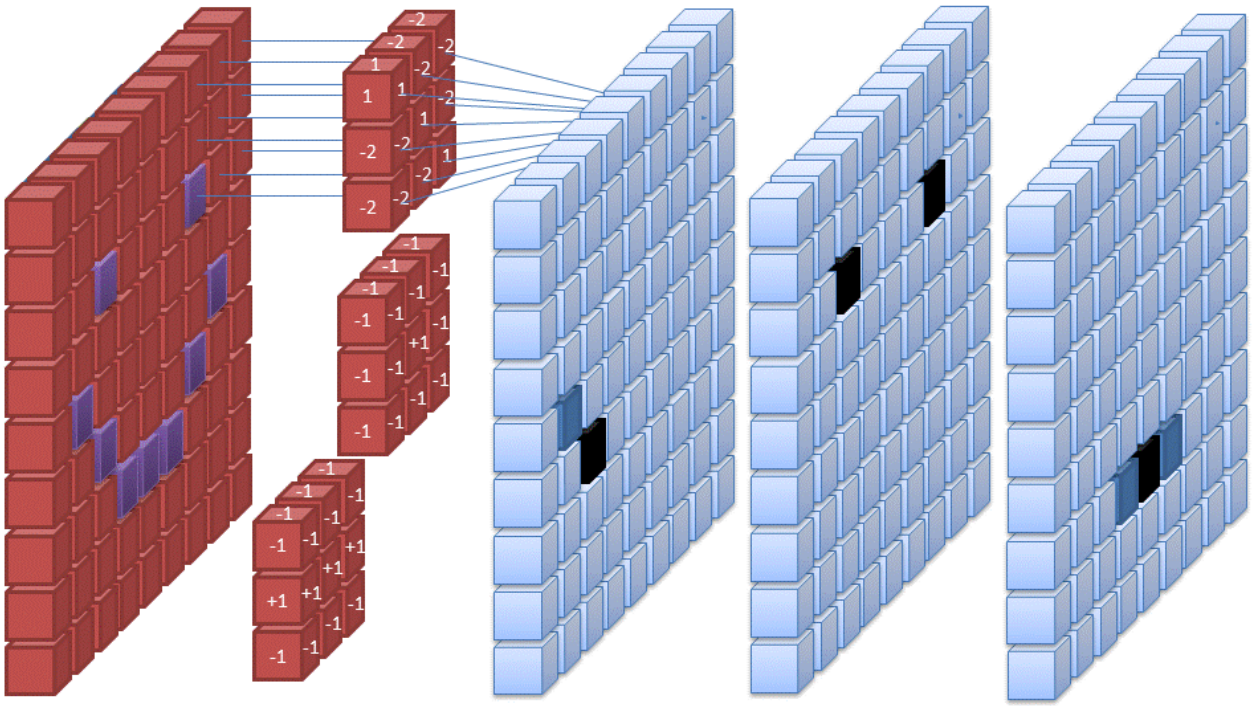


Image source: `https://commons.wikimedia.org/wiki/File:3_filters_in_a_Convolutional_Neural_Network.gif`

Feature Maps

- Consequence of weight sharing:
- Each layer can only learn a single feature
- To learn more, we use several copies of the layer
- Each copy is called a *feature map*
- Example in picture: 20 different features for image recognition task learned in first hidden layer

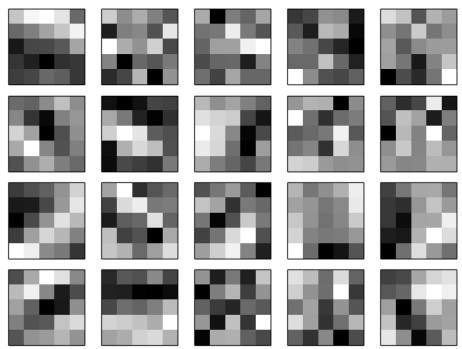


Image source: `http://neuralnetworksanddeeplearning.com/chap6.html`

Deep Convolutional NN Concepts

- Repeat many layers of local filters, processing pipeline
- Higher-level features built from simpler local features
- Pooling: reduce from e.g. 2×2 region to single neuron by averaging or maximum
- Padding: add artificial outside elements to prevent shrinking
- Often: fully connected layers at the end of pipeline

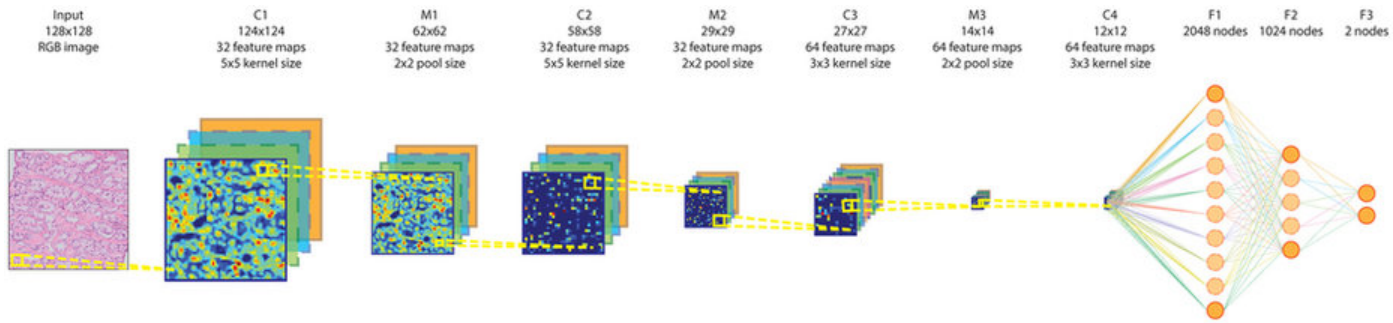


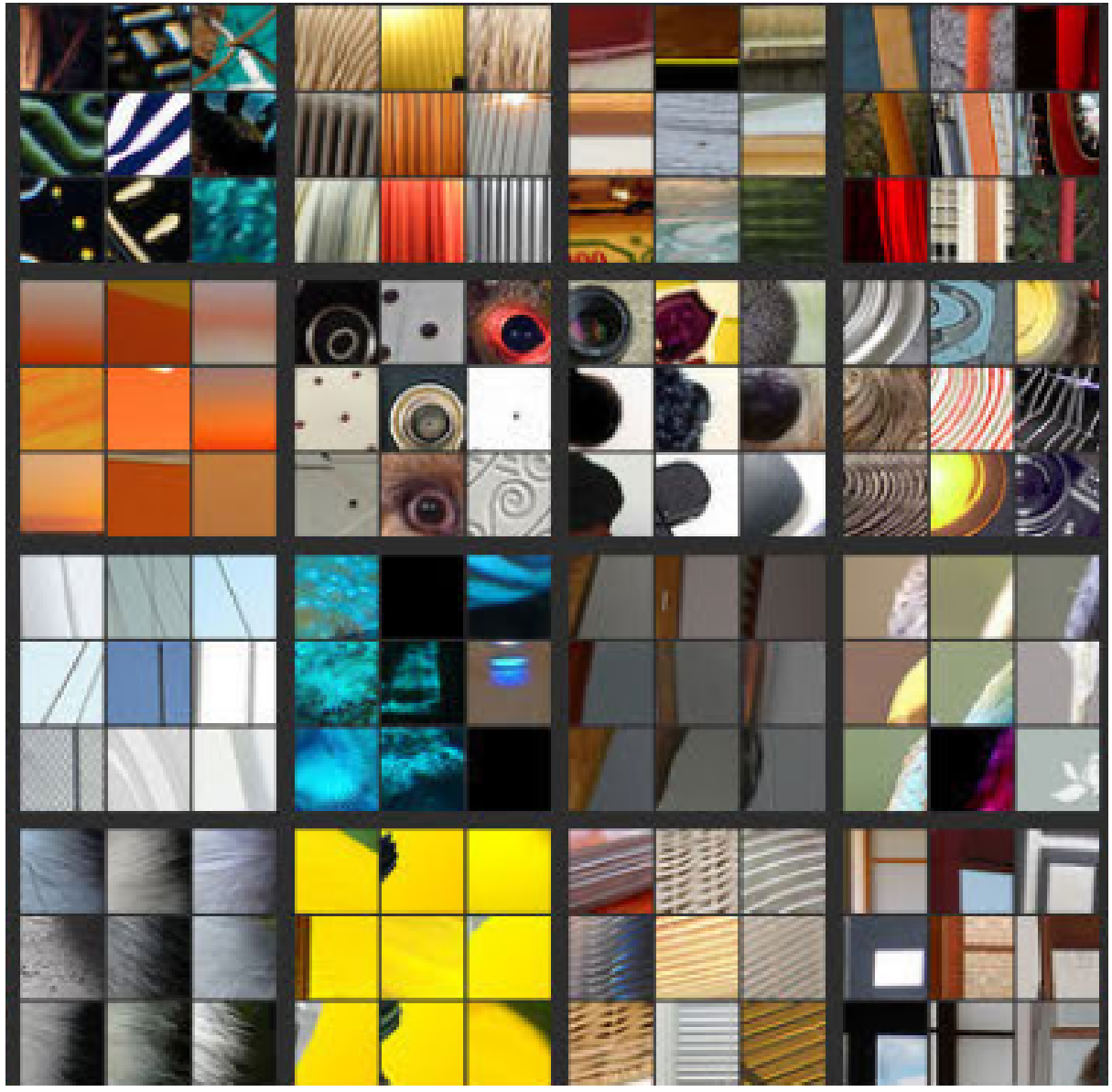
Image source: `http://www.nature.com/articles/srep26286/figures/1`

What Do the Learned Features Mean?

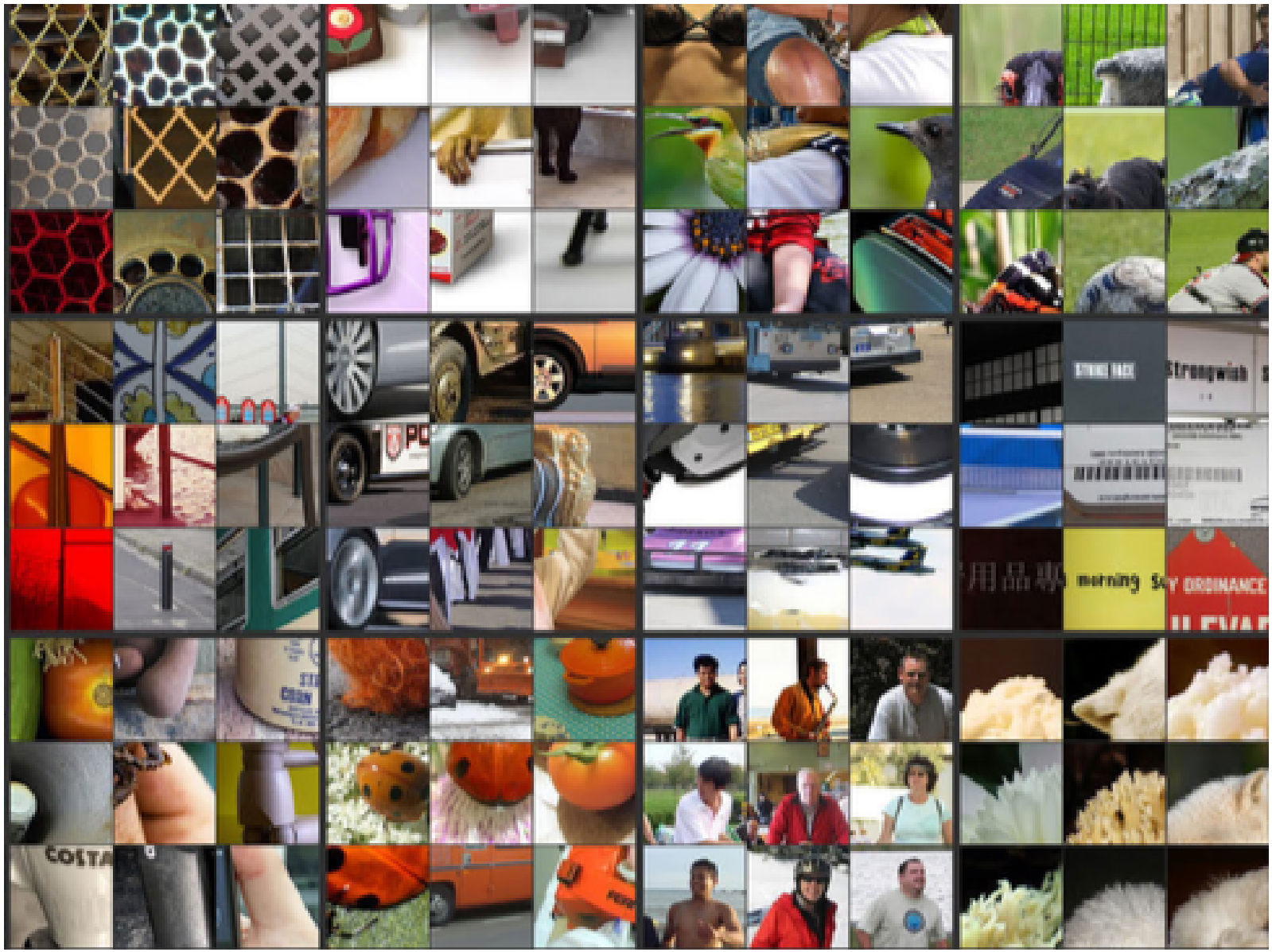
- Short answer: usually, we don't know
- But they work!!!
- Long answer: "it's an active area of research"
- Some successes with interpreting features from image recognition tasks
- Images on next few slides:
  Zeiler and Fergus (2013),
  Visualizing and Understanding Convolutional Networks,
  `https://arxiv.org/pdf/1311.2901.pdf`

Learned Features, Hidden Layer 1

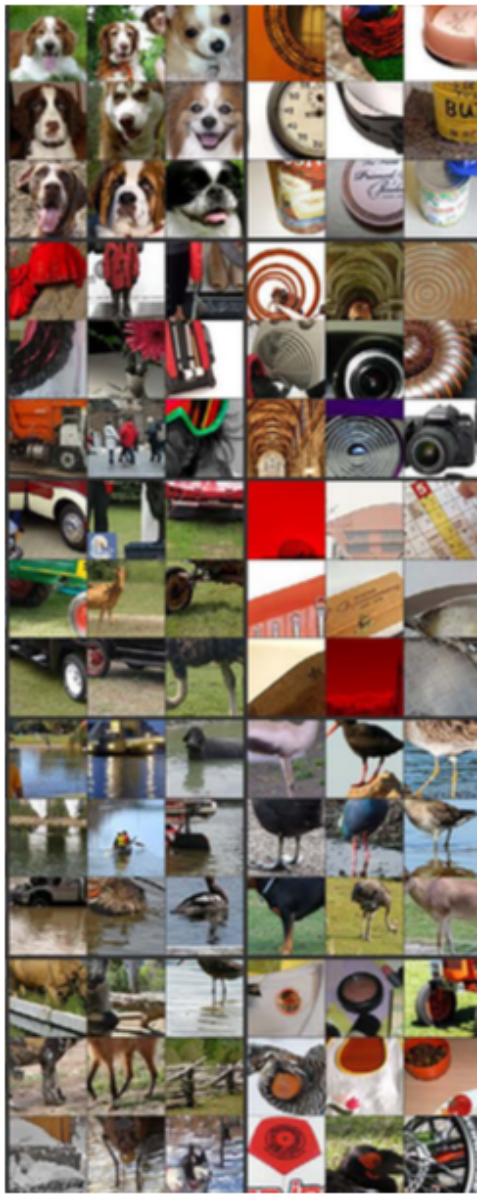
Learned Features, Hidden Layer 2

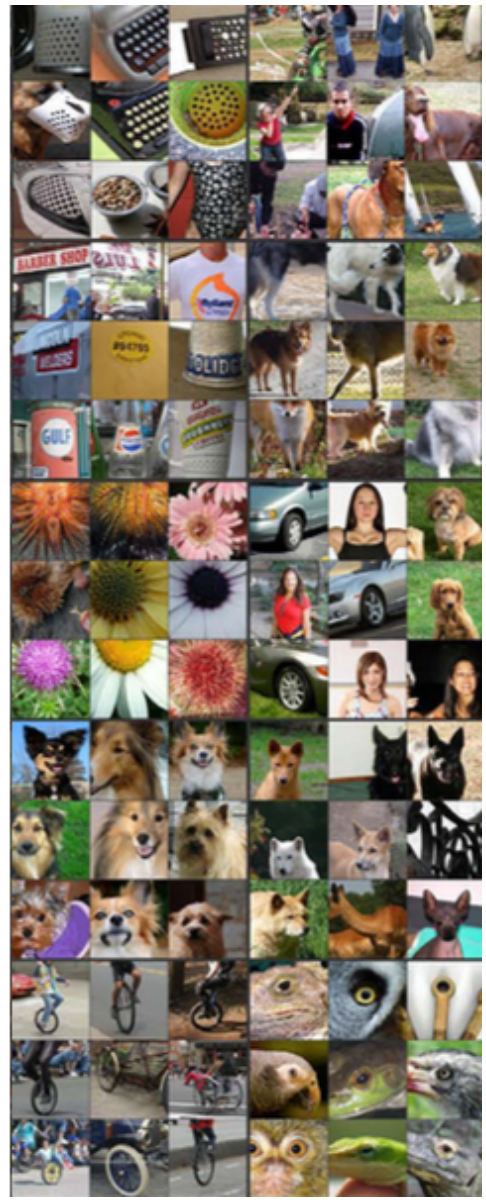


Learned Features, Hidden Layer 3



Learned Features, Hidden Layer 4 and 5





Summary of CNN - General

- Studied NN architectures
- Fully connected or sparse
- Convolutional NN (CNN) learn hierarchy starting with local features
- Deep CNN were breakthrough for image understanding

Deep Convolutional Neural Networks for Go

Deep Convolutional Networks for Go

- Idea: image recognition is similar to recognizing high-level Go concepts
- Go is a very visual game
- Two papers in 2015 within weeks of each other
- Massive improvement in move prediction for Go
  From about 40% to 57%
- Both trained on full 19×19 board
- Clark and Storkey, Training Deep Convolutional Neural Networks to Play Go.
- Maddison, Huang, Sutskever, Silver. Evaluation in Go using deep convolutional neural networks. ICLR 2015.

Clark and Storkey DCNN

Studied two different types of input

- Raw input, three *channels*
  - Channel = 19 $\times$ 19 bitmap (boolean array)
  - Black stones
  - White stones
  - Point forbidden by simple ko rule
- Input with liberty information, seven channels
  - Split black, white stones into three separate channels by liberty count
  - 1 liberty (in atari)
  - 2 liberties
  - 3 or more liberties

Architecture

- Many convolutional layers
- Many small convolutional filters
- Zero-padding to keep each layer at size 19×19
  - Add extra rows and columns with values of 0 around the edges after applying filters
- One fully connected layer as last layer before output
- ReLu activation function

ReLU Activation Function

- ReLU stands for rectified linear unit
- Very popular simple nonlinear function
- $f(x) = \max(0, x)$
- Derivative is also very simple:
- 0 for $x < 0$
- 1 for $x > 0$
- Often more efficient learning than sigmoid, other continuous activation functions

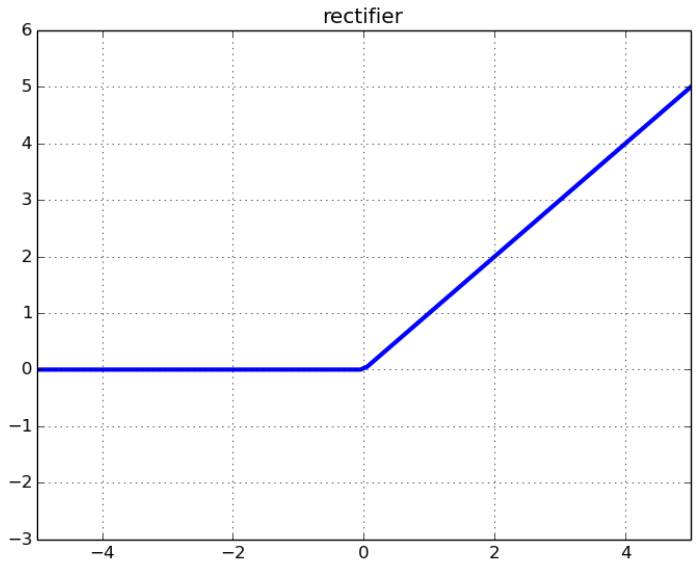


Image source:

`https://i.stack.imgur.com/8CGlM.png`

More Design Features

- Edge encoding
  - Extra channel to indicate edge of board
    - Avoid that the zero-padding looks like empty points
- Masked training
  - Do not backpropagate results for illegal moves
- Reflection preservation
  - Share weights to make sure output is same if we rotate or mirror the board

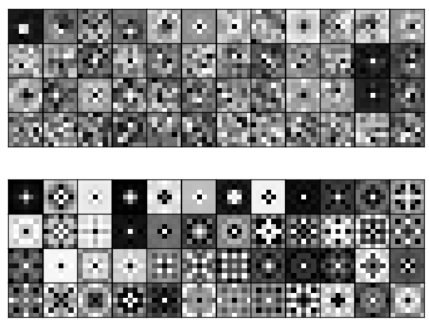


Image source: Clark and Storkey

Training Data

- Two data sets
- 81,000 games by professionals from Games of Go on Disk (GoGoD) collection
- 86,000 games by strong amateurs from KGS Go server
- About 16.5 million move-position pairs for each dataset
- Split into 88% training, 8% test, 4% validation data

Training Method

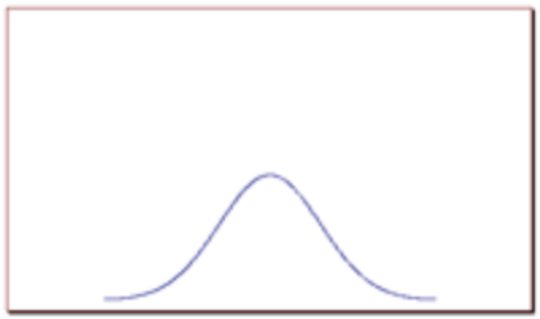


Image source: `http://images.tutorvista.com/cms/images/67/NORMAL-DISTRIBUTION.PNG`

- Basic gradient descent algorithm
- Initialize weights randomly, from *normal distribution* with mean zero, standard deviation 0.01
- 10 “epochs” of training over all data
- Measured three metrics:
  - Accuracy, Rank, Probability

Accuracy, Rank, Probability

- Accuracy
  - How often is the network's move equal to the master move?
- Rank
  - In the ordered list of moves as ranked by the net, where is the master move?
- Probability
  - What probability did the net assign to the master move?
- Best: 100% accuracy, rank 1, probability 100%
- As discussed before, this cannot be achieved in Go
  - Equally good moves
  - Different preferences in opening
  - Different ideas about the "safest" way to win, or best way to complicate game when losing

Results

	Accuracy	Rank	Probability
Test Data	41.06%	5.91	0.1117
Train Data	41.86%	5.78	0.1158

Table 2. Results for the 8 layer DCNN on the train and test set of the GoGoD dataset. Rank refers to the average rank the expert's move was given, Probability refers to the average probability assigned to the expert's move.

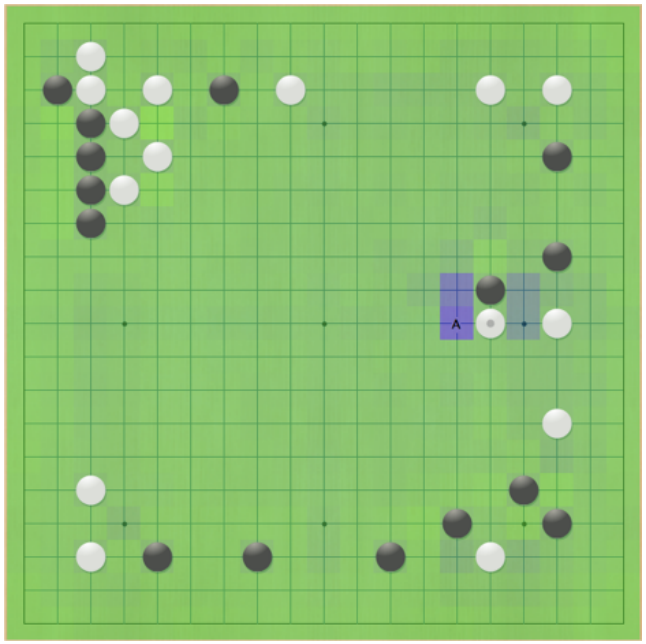
	Accuracy	Rank	Probability
Test Data	44.37%	5.21	0.1312
Train Data	45.24%	5.07	0.1367

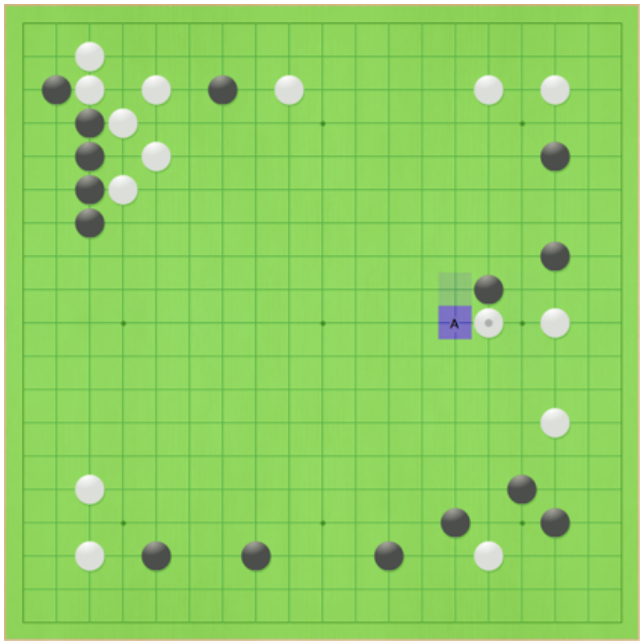
Table 3. Results for the 8 layer DCNN on the train and test set of the KGS dataset. Rank refers to the average rank the expert's move was given, Probability refers to the average probability assigned to the expert's move

Image source: Clark and Storkey

- About 4% improvement over previous best results
  - Using simple features and pairwise interactions
- Trained 4 days on single GPU
- Did *not* use the previous move as an input feature, which is very important with simple features
- Policy-only player, no search
- Wins most games vs GNU Go, some vs (old) Fuego

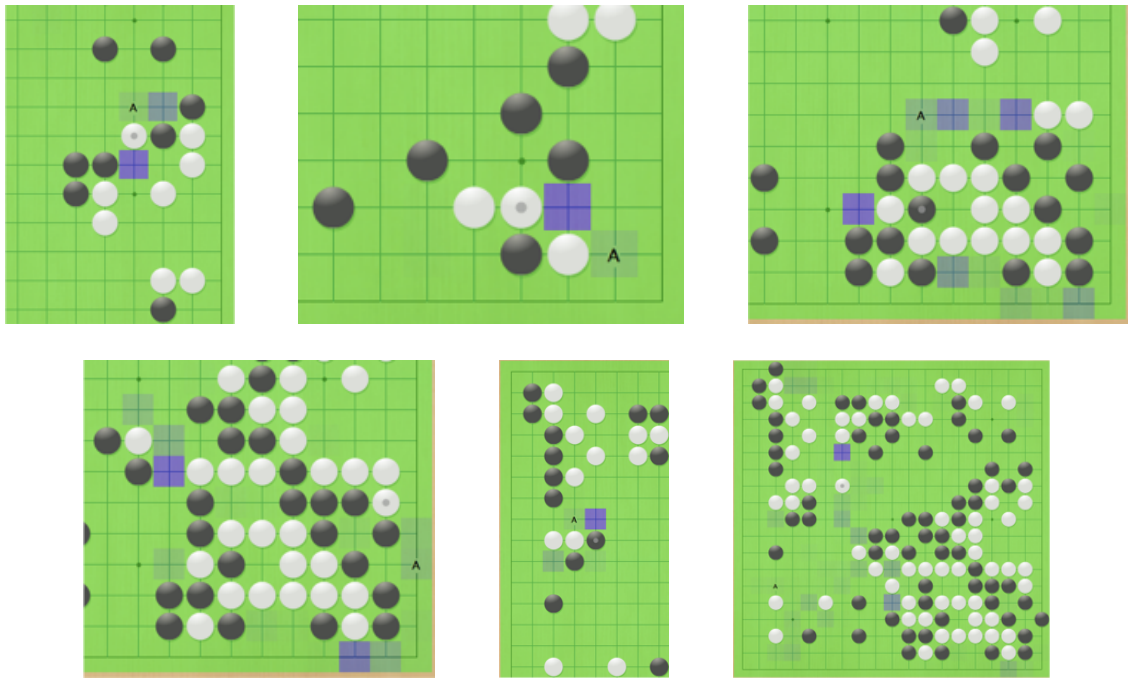
DCNN vs Simple Features





- Compare simple features with interactions (left) vs Clark and Storkey DCNN (right)
- Simple features: more generic knowledge, spread out
- DCNN: focused on very small number of moves

Strong and Weak Points of DCNN



- Many good moves, but not enough to play good Go
- About 10 bad moves per game, some real blunders
- Bad moves all from one randomly chosen game
- Need to combine it with search and simulations

Maddison et al Paper

- First Deepmind Go Paper
- Appeared soon after Clark and Storkey
- Very similar approach overall
- More input features, more training, deeper net
- Much stronger results

Input Feature Planes

- Total 36 input planes (Clark/Storkey had 7)

Feature	Planes	Description
Black / white / empty	3	Stone colour
Liberties	4	Number of liberties (empty adjacent points)
Liberties after move	6	Number of liberties after this move is played
Legality	1	Whether point is legal for current player
Turns since	5	How many turns since a move was played
Capture size	7	How many opponent stones would be captured
Ladder move	1	Whether a move at this point is a successful ladder capture
KGS rank	9	Rank of current player

Table 1: Features used as inputs to the CNN.

Architecture and Training

- 12 layers, ReLU activation function
- First layer has 5×5 filters, later layers 3×3
- Between 64 and 192 filters per layer
- Big network:
  - 2.3 million parameters
  - 630 million connections
  - 550,000 hidden units
- Asynchronous stochastic gradient descent
- 50 models trained in parallel with 1 GPU each
- Final training to create a single model,
  with reduced learning rate

Results (1)

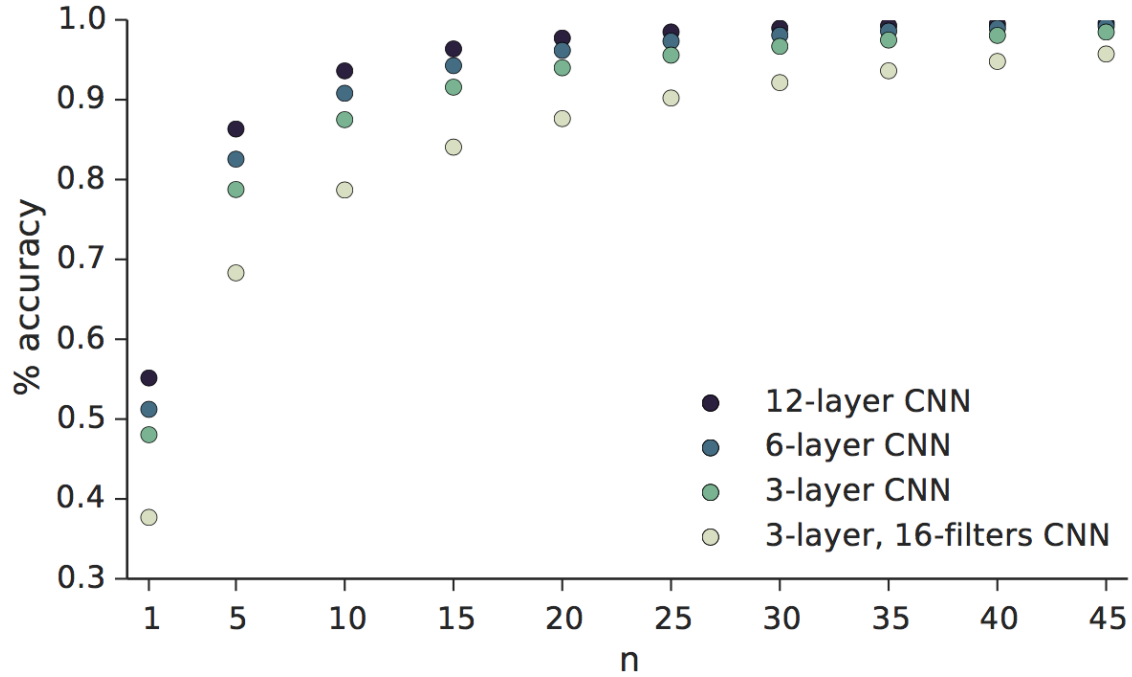


Figure 1: Probability that the expert's move is within the top-n predictions of the network. The 10 layer CNN was omitted for clarity, but it's performance is only slightly worse than 12 layer. Note y-axis begins at 0.30.

Image source: Maddison et al

Results (2)

Depth	Size	% Accuracy	% Wins vs. *GnuGo*	stderr
3 layer	16 filters	37.5	3.4	± 1.1
3 layer	128 filters	48.0	61.8	± 2.6
6 layer	128 filters	51.2	84.4	± 1.9
10 layer	128 filters	54.5	94.7	± 1.2
12 layer	128 filters	55.2	97.2	± 0.9
8 layer (Clark & Storkey, 2014)[4]	≤ 64 filters	44.4	86	± 2.5
Aya 2014		38.8	6	± 1.0
Human 6 dan		52 ±5.8	100	

Image source: Maddison et al

Results (3)

Opponent	Rollouts per move	Games won by CNN	stderr
GnuGo		97.2	± 0.9
MoGo	100,000	45.9	± 4.5
Pachi	100,000	11.0	± 2.1
Fuego	100,000	12.5	± 5.8
Pachi	10,000	47.4	± 3.7
Fuego	10,000	23.3	± 7.8

Image source: Maddison et al

Summary and Outlook

- Deep convolutional neural nets (DCNN) revolutionized image understanding
- They also work very well for move prediction in Go
- Best static (no search) method known
  - A very simple MCTS + DCNN in the Deepmind paper showed they could be combined in principle
- After these two papers, everyone started building DCNN for their Go programs
- Deepmind went quiet for a while
- Then the first AlphaGo paper appeared in January 2016