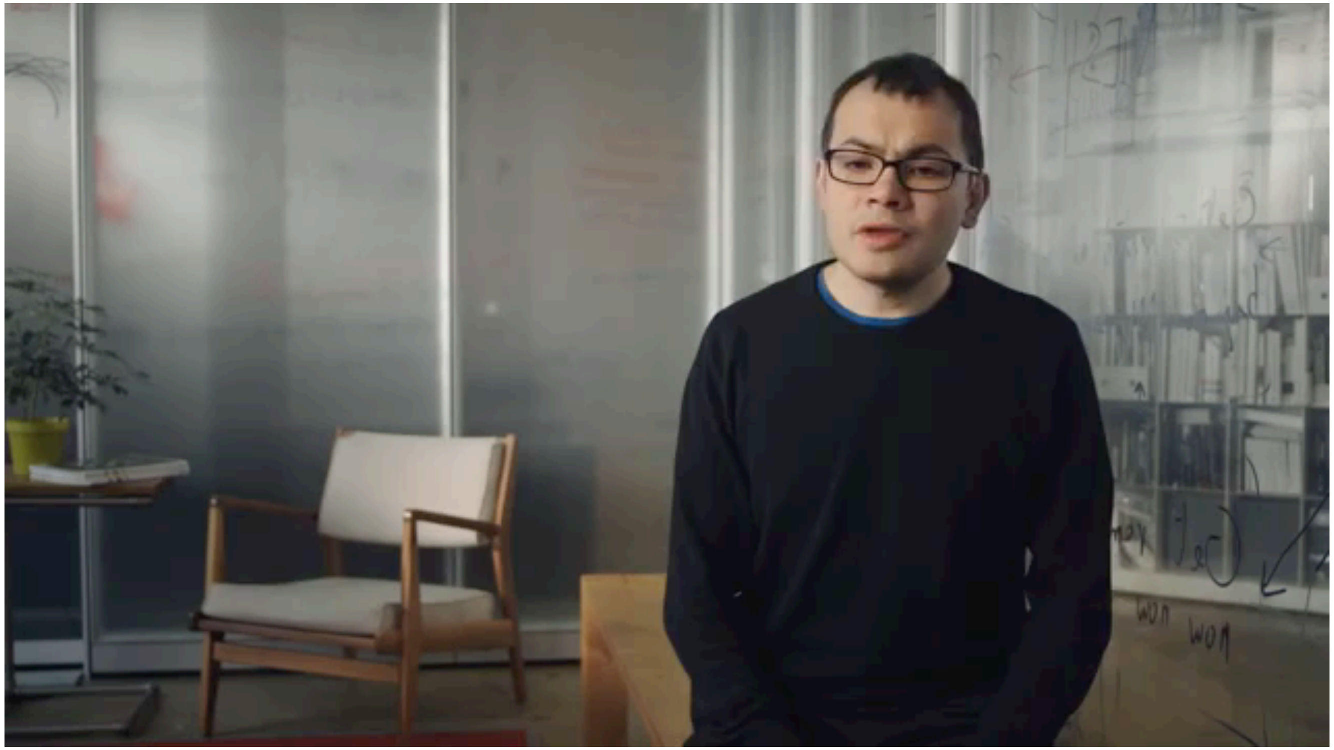
What makes AlphaGo, go?

Raghavendra Singh

Alpha-Go

* Alpha-Go (Google Deepmind) played against top human player in March 2016 and beat him 4 games to 1
* Demis Hassabis of Deepmind

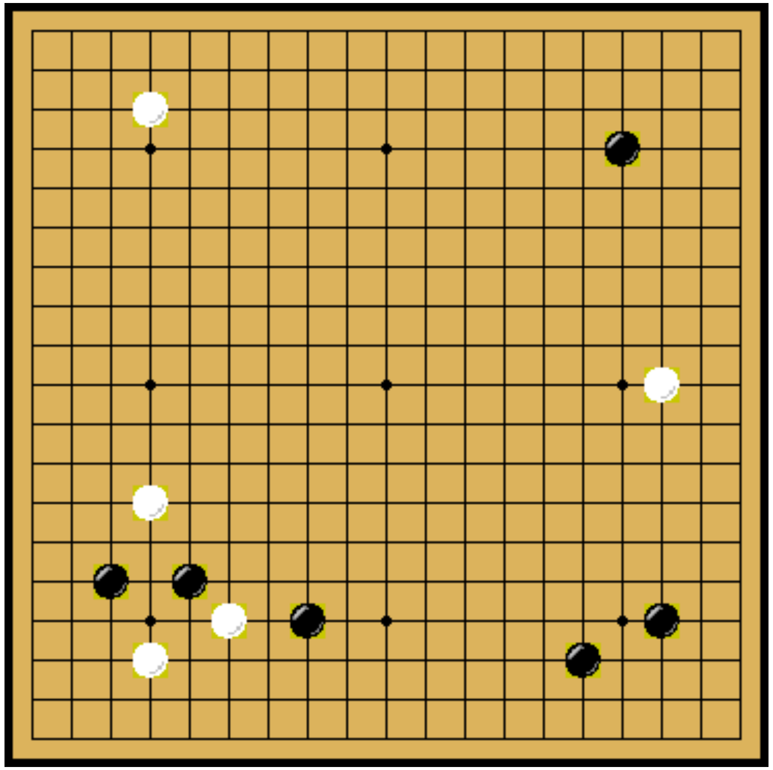


Go

Ancient Chinese board games with two players

Place Black and White stones on intersection of 19x19

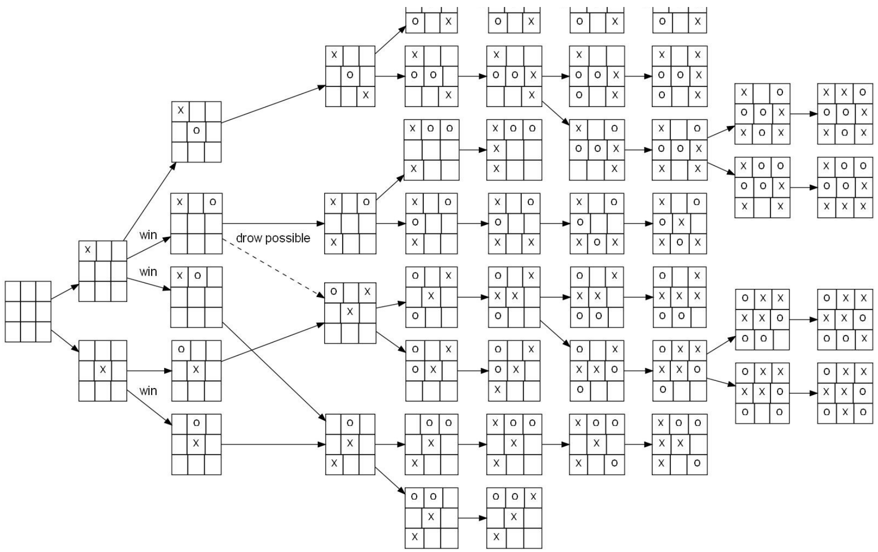
Capture a larger total area of the board with one's stones than the opponent by the end of the game



Surrounding an opposing stone by occupying all orthogonally-adjacent intersections will "capture" stone and lead to its removal

Players place stones alternately until they reach a point at which neither player wishes to make another move

Game tree



* Minmax algorithm on complete tree
    * from current node traverse down to leaf nodes and pick action
    * minmize the worst case scenario
* Knowing the complete game tree allows the program to pick the best possible move at a given game state.
    * Exhaustive search is infeasible
* Board games like Chess or Go solved by recursively computing an optimal value function in a search tree
    * game states (position) as nodes
    * possible actions as edges

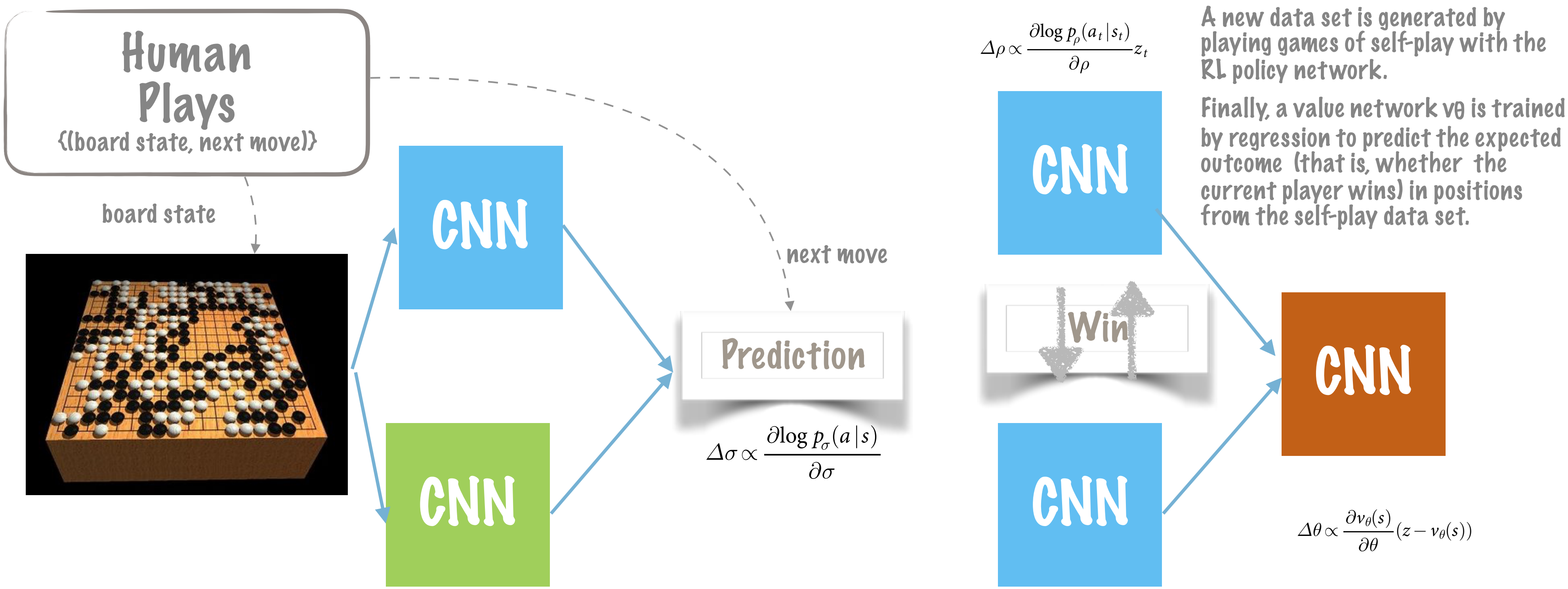
Deep Blue

* Search the game tree as far as possible
    * use an evaluation function to evaluate the quality of the nodes at that depth
        * is a function used by game-playing programs to estimate the value or goodness of a position in the minimax and related algorithms.
            * (1)The relative values of queen, rook, bishop, knight and pawn are about 9, 5, 3, 3, 1, respectively. Thus other things being equal (!) if we add the numbers of pieces for the two sides with these coefficients, the side with the largest total has the better position. (2)Rooks should be placed on open files. This is part of a more general principle that the side with the greater mobility, other things equal, has the better game. (3)Backward, isolated and doubled pawns are weak. (4)An exposed king is a weakness (until the end game)....
            * $f(P) = 200(K-K') + 9(Q-Q') + 5(R-R') + 3(B-B'+N-N') + (P-P') - 0.5(D-D'+S-S'+I-I') + \ldots$
            * calculates f(P) for the various positions obtained from the present position by each possible move that can be made
                * replace the subtree below that node with a single value summarizing this subtree
* Quality of an evaluation function and raw computer power to search to greater depths
    * Deep Blue relied largely on brute force, plus some well-designed heuristics including using known chess players

AI-Go

* Go is more difficult with its much larger tree
  * it is also more difficult to design evaluation functions for Go than for chess
* Monte Carlo Tree Search (MCTS) alternative approach to searching the game tree
  * run many game simulations
    * store some values: how often each node is visited, how often this has led to a win
  * Use numbers to guide the later simulations in selecting actions
* Domain knowledge is added to MCTS to make strong GO AIs(Fuego, Pachi..)
* AlphaGo along with MCTS makes extensive use of machine learning to avoid using hand-crafted rules
  * Deep architectures can learn and represent evaluation function

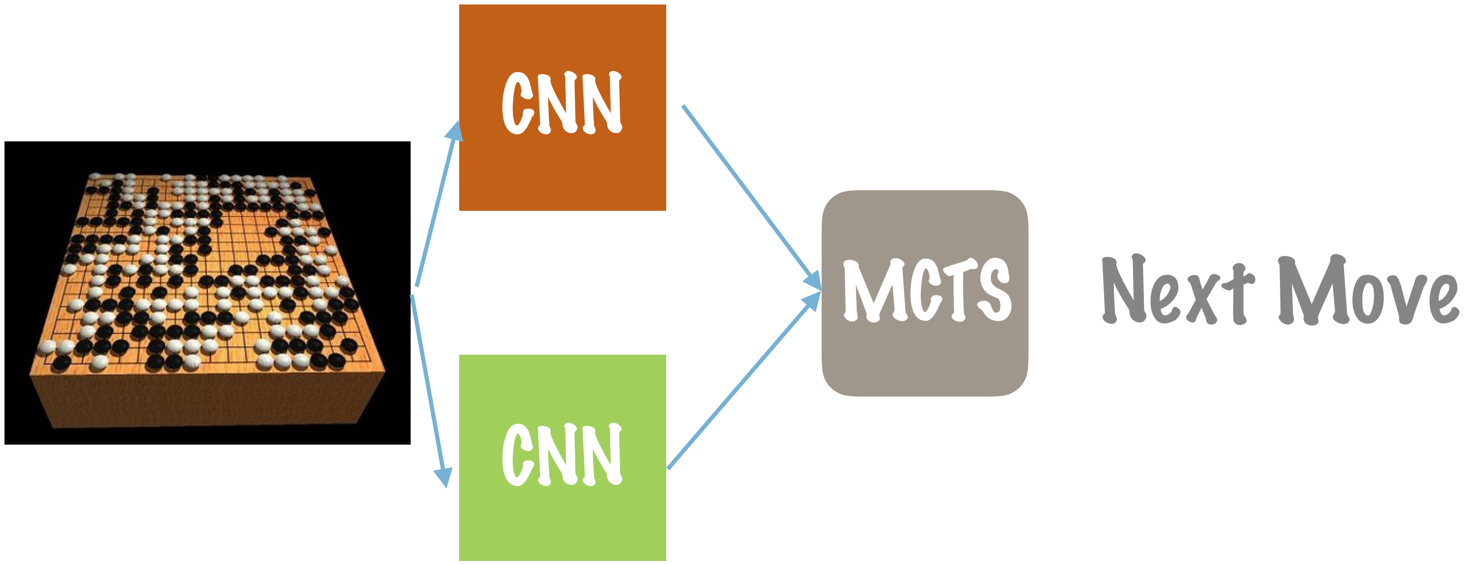
AlphaGo training system



* Policy networks provide guidance regarding which action to choose, given the current state of the game.
    * supervised learning trained on 30 million positions from games played by human exports
    * output is a probability value for each possible legal move, vector of dimension of board

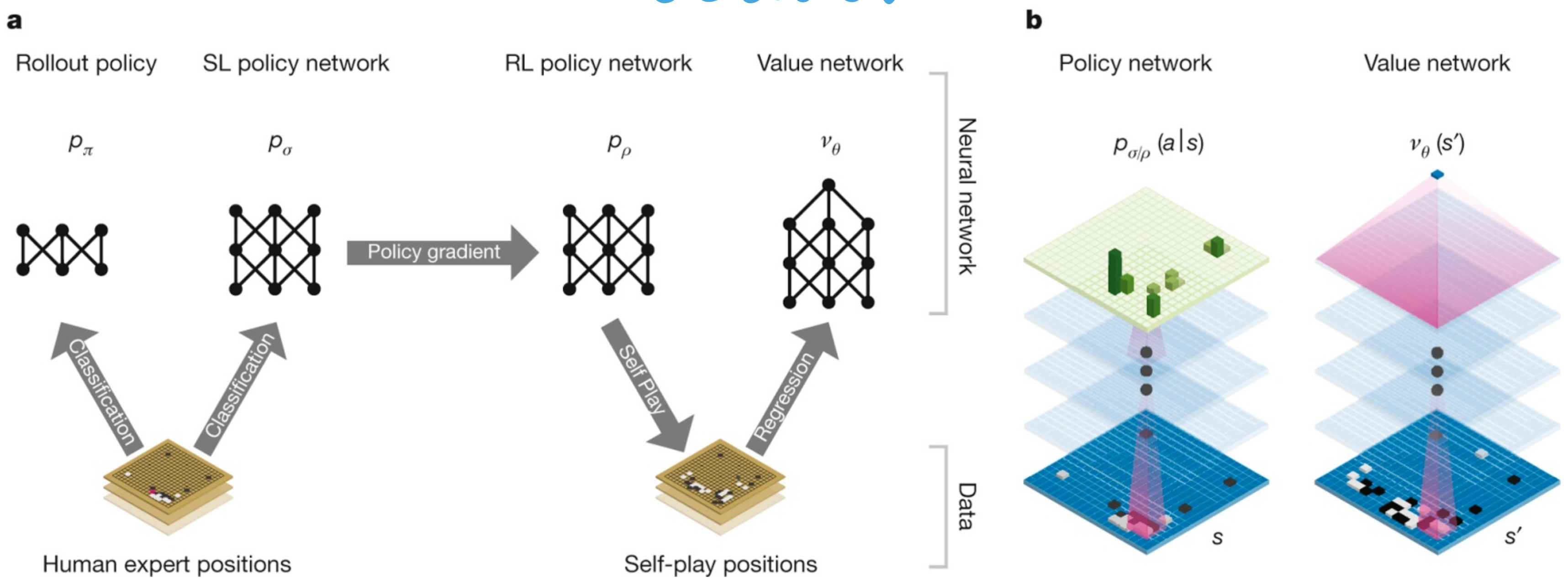
* A value network adjusts the policy towards the correct goal of winning games, rather than maximizing prediction accuracy
    * reinforcement learning play against each other, using the outcome of these games as a training signal
    * output is a single number, representing the probability of a win.

AlphaGo playing system



* MCTS and convolutional neural networks (CNN) that guide the tree search
    * evaluating positions using a value network, and sampling actions using a policy network
* CNN similar to the evaluation function in Deep Blue, except that they are learned and not designed.
* Without search, just networks, Alpha-Go beat search based Go programs
    * intuition is very important in the game of Go. It also shows that it is possible to play well without relying on very long reflections

Mastering the game of Go with deep neural networks and tree search

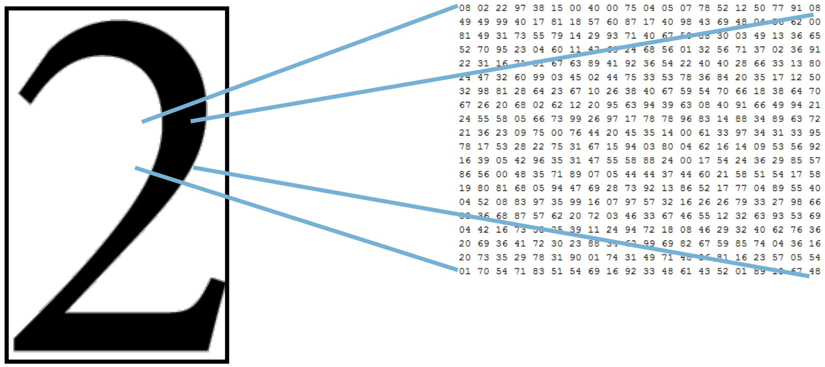


D Silver *et al.* *Nature* **529,** 484–489 (2016) doi:10.1038/nature16961

nature

Computer "Vision"

Image

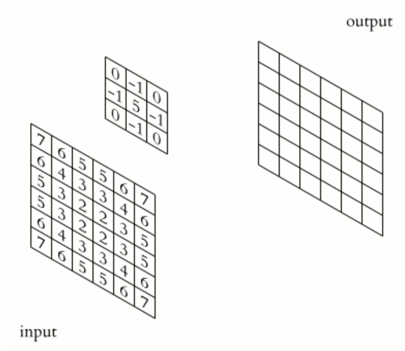


Computer sees a matrix of numbers

Classifying an image is to label the image

2

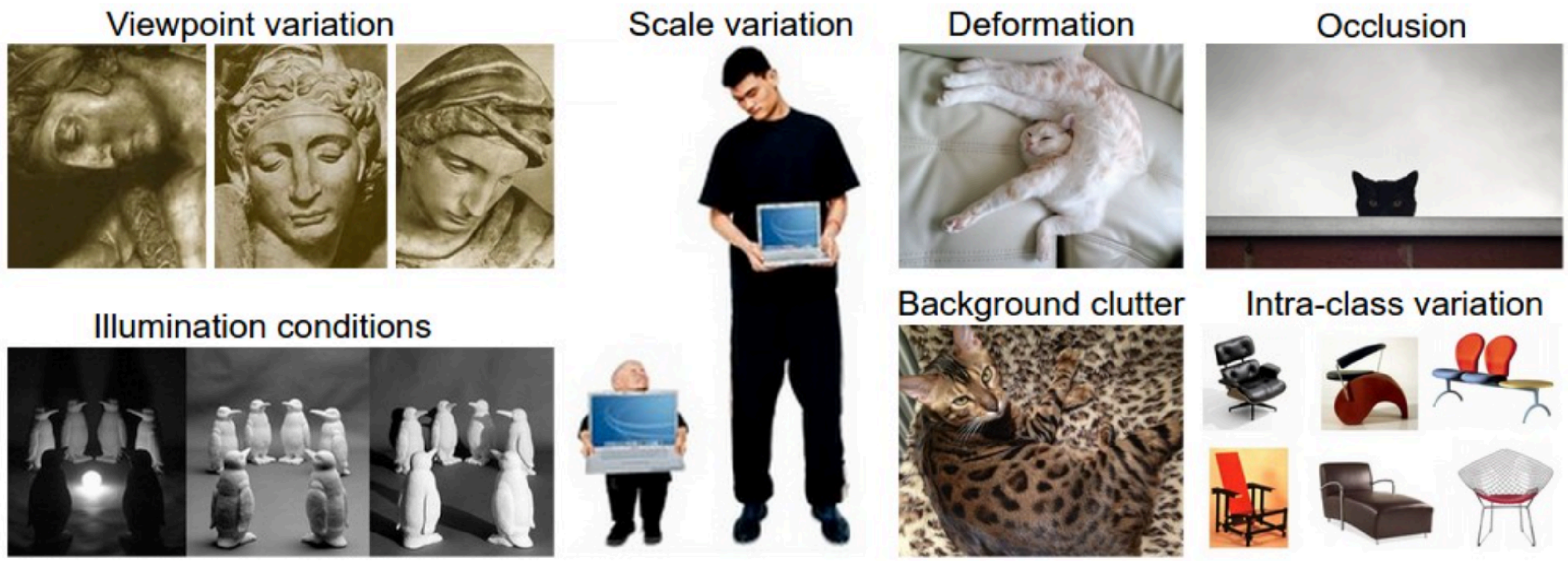
Processing an image is to mathematically operate on it



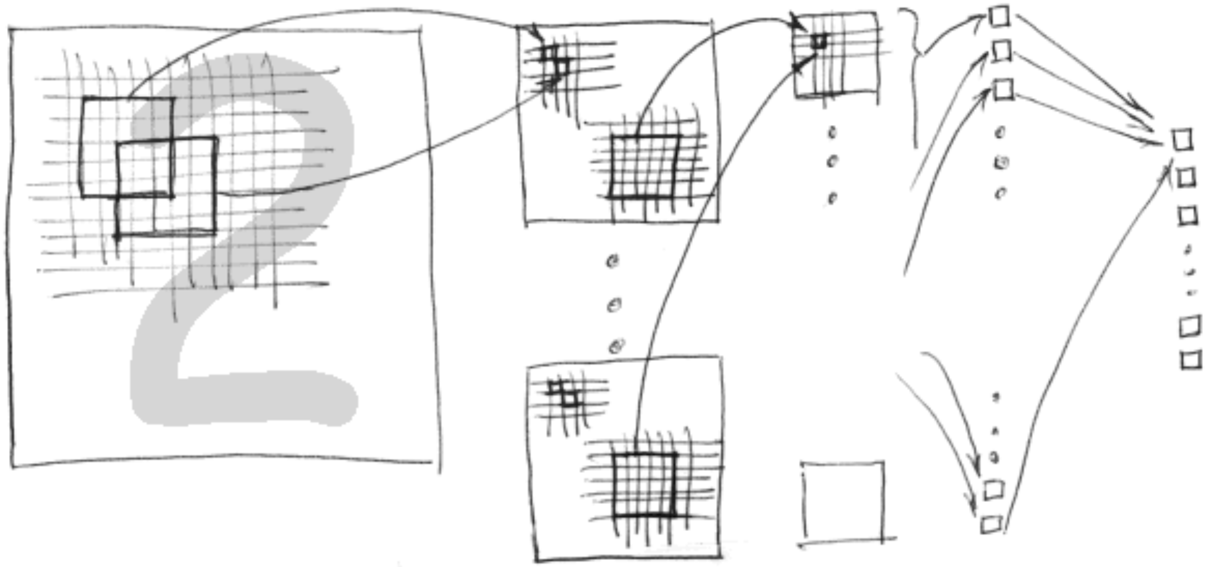
Understanding an image is to describe the image

numeral 2
in black on white background

Why is it hard?



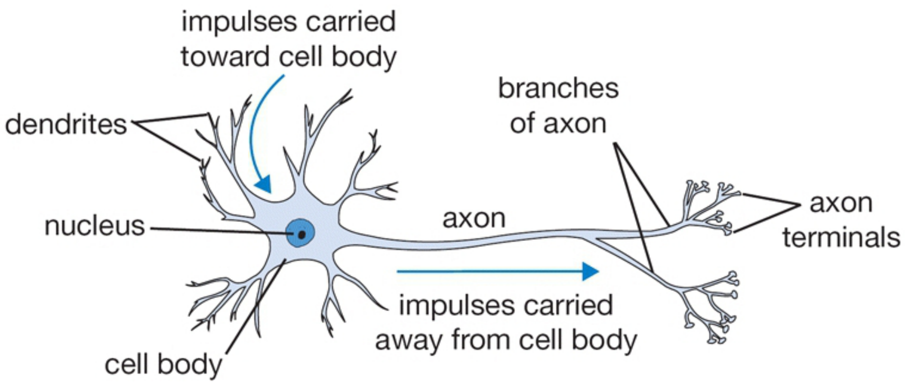
Convolutional Neural Networks

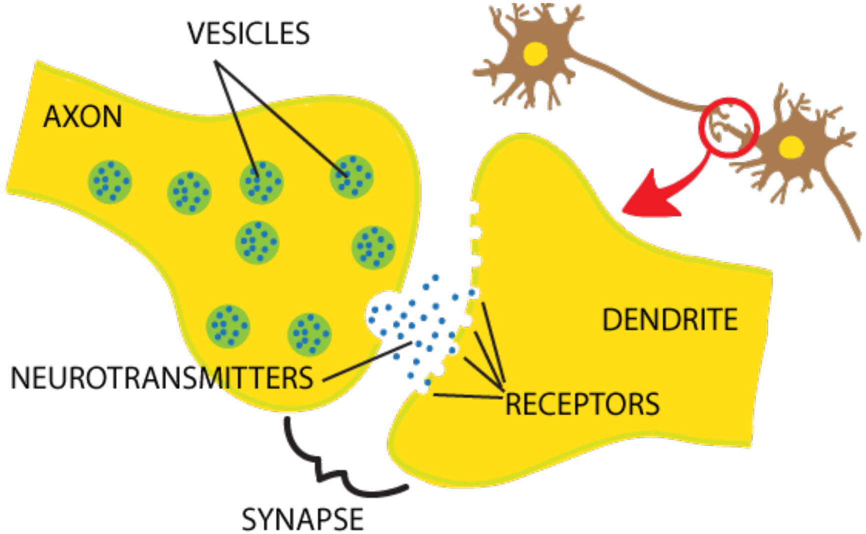


* Layers of operations
* Each operation is local —
* Uses common learned parameters —
* Each operation by a artificial neuron

Convolution

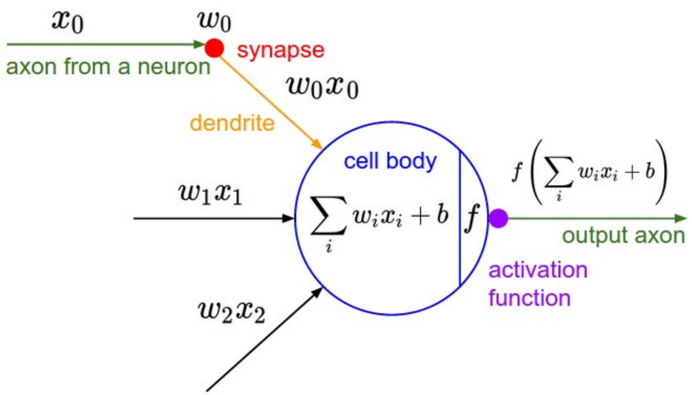
Biological Neurons

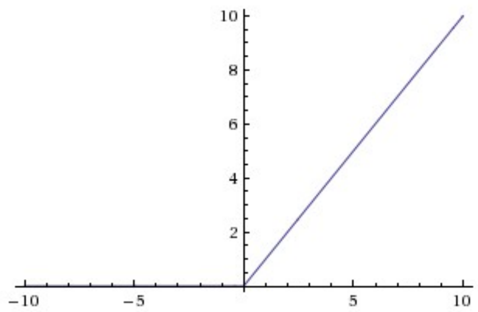




* Integrate signals on dendrites
* If result greater than a threshold send out an impulse along axon
  * Leak potential if no incoming signal
* Impulse carries time stamp
* Neurotransmitters in synapse modulate signal strength
  * Neurotransmitter properties change as brain "learns"

Artificial Neurons

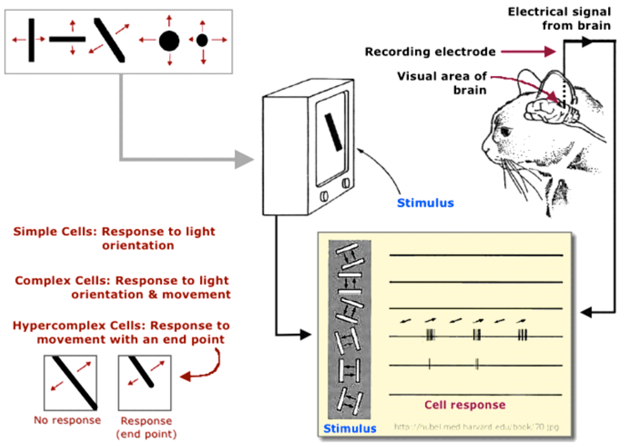




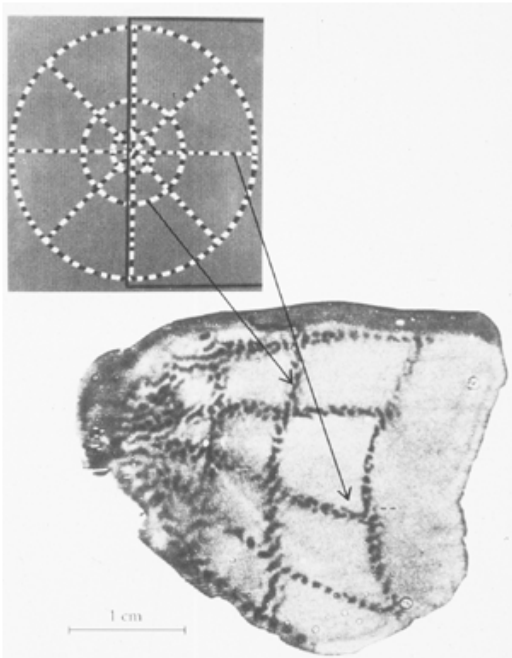
ReLU

* Sum weighted inputs
* Activation function is a non-linear function, e.g. ReLu
* No time stamp
* Synapse — weights that multiply input
  * Weights and biases are "learned" from data

Why Convolution



Hubel & Wiesel, 1959

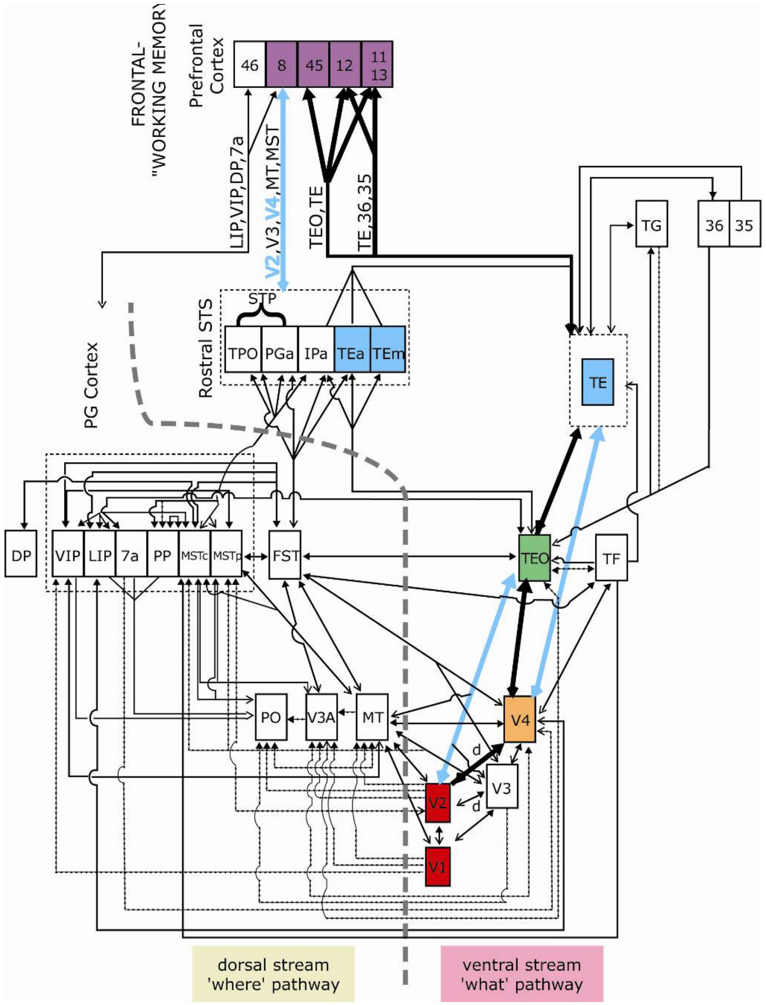


* Receptive fields of visual system are local
* Filters are oriented
* Topographic Mapping — nearby cells in retina processed by nearby regions in the brain

Hubel & Wiesel

* Won Nobel Prize for this work

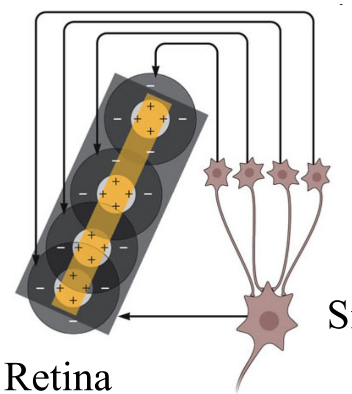
Why Layers

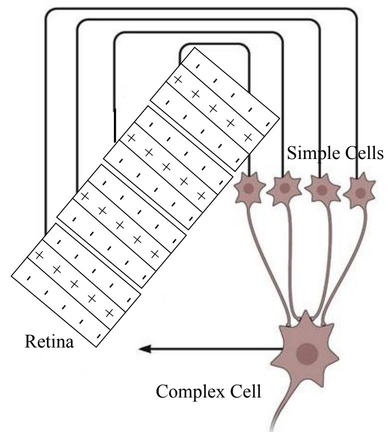


Modified from (Ungerleider & VanEssen)

* Visual system has layers of processing

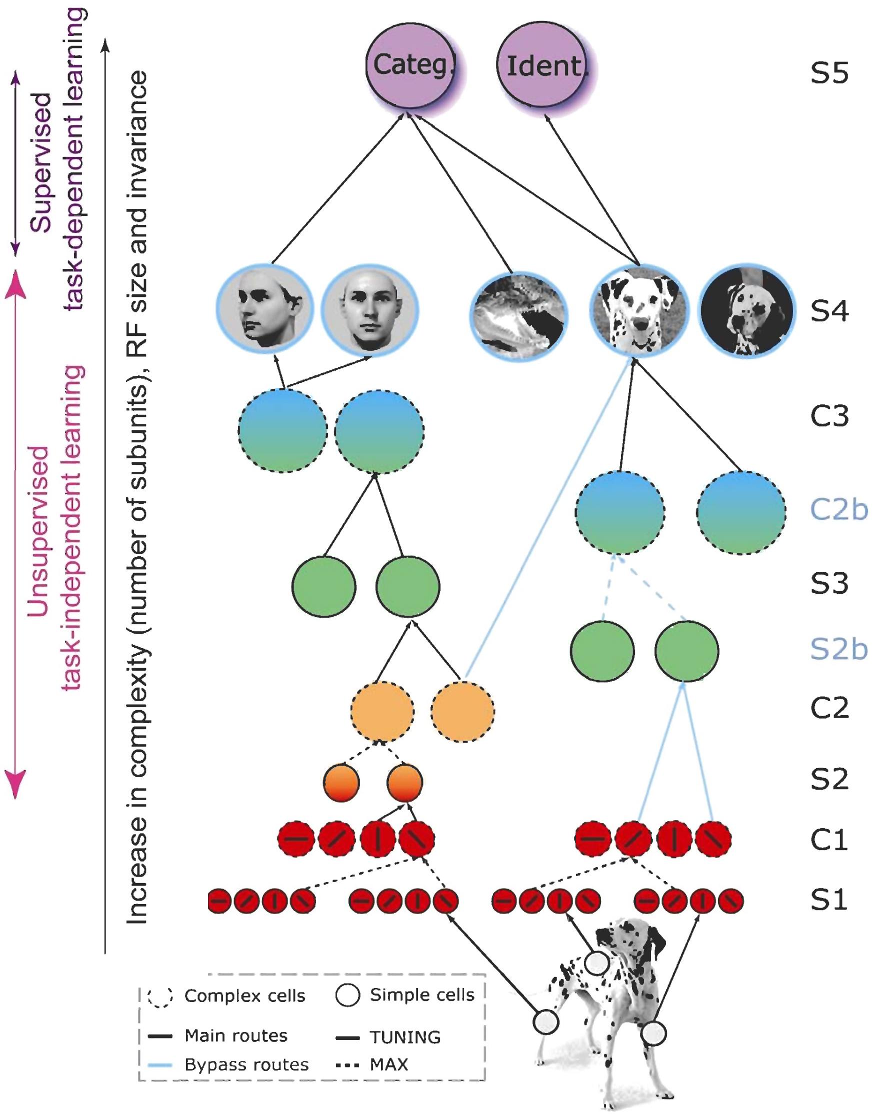
* Hierarchy of cells



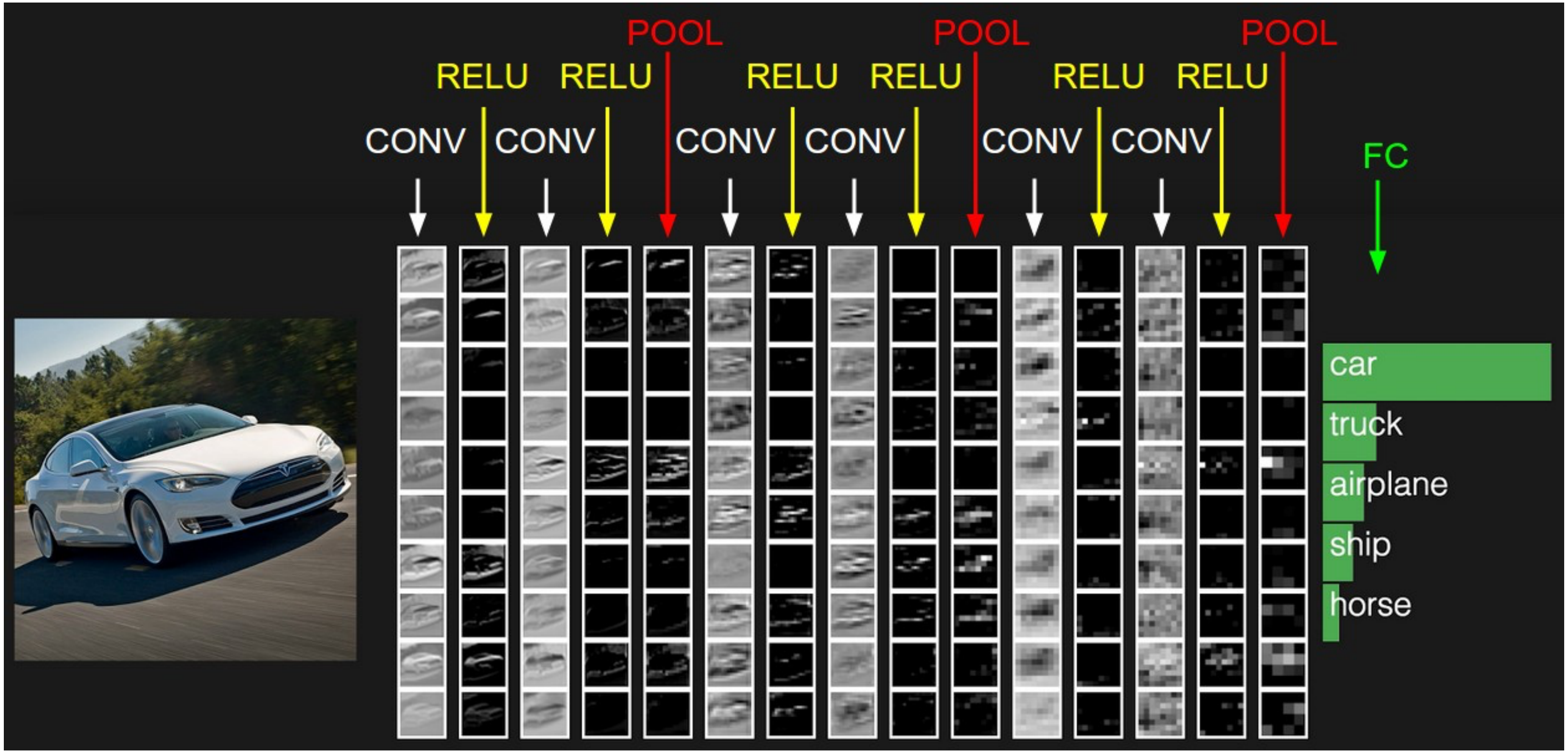


Visual Processing Model

Model layers	Corresponding brain area (tentative)	RF sizes	Number units	
classifier	PFC		$1.0\ 10^{0}$	
S4	AIT	$>4.4^{o}$	$1.5\ 10^{2}$	~ 5,000 subunits
C3	PIT - AIT	$>4.4^{o}$	$2.5\ 10^{3}$	
C2b	PIT	$>4.4^{o}$	$2.5\ 10^{3}$	
S3	PIT	1.2^{o}- 3.2^{o}	$7.4\ 10^{4}$	~ 100 subunits
S2b	V4 - PIT	0.9^{o}- 4.4^{o}	$1.0\ 10^{7}$	~ 100 subunits
C2	V4	1.1^{o}- 3.0^{o}	$2.8\ 10^{5}$	
S2	V2 - V4	0.6^{o}- 2.4^{o}	$1.0\ 10^{7}$	~ 10 subunits
C1	V1 - V2	0.4^{o}- 1.6^{o}	$1.2\ 10^{4}$	
S1	V1 - V2	0.2^{o}- 1.1^{o}	$1.6\ 10^{6}$	

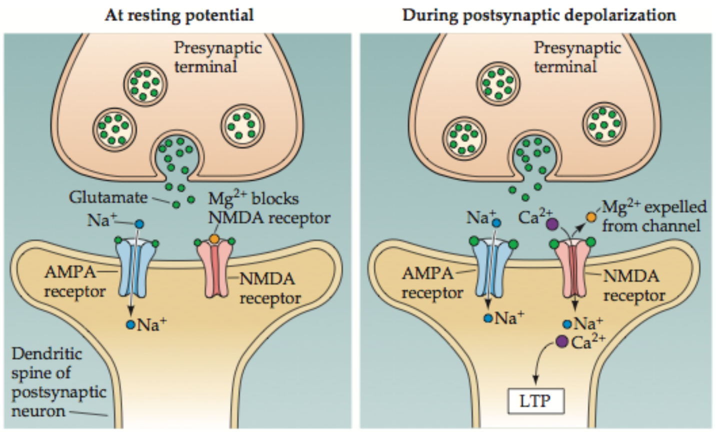


Convolutional NN



* Parameters of CONV layers are learned from data

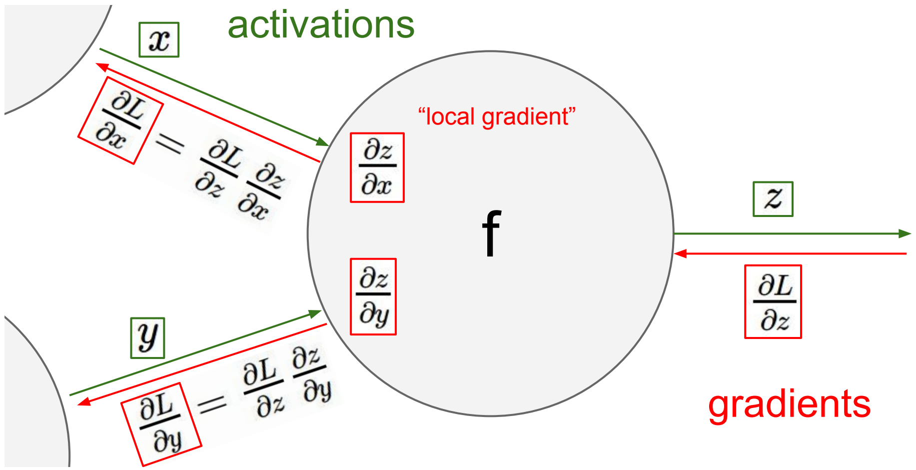
Learning



The NMDA receptor channel can open only during depolarization of the postsynaptic neuron from its normal resting level. Depolarization expels Mg^{2+} from the NMDA channel, allowing current to flow into the postsynaptic cell. This leads to Ca^{2+} entry, which in turn triggers LTP.

- Hebbian principle
  - Cells that fire together wire together
- Change the neurotransmitters of the synaptic gap

- Supervised learning
  - **Change the weights** of each neuron based on a loss function
  - Gradient descent and chain rule (back-propogation) used to change the weights through layers

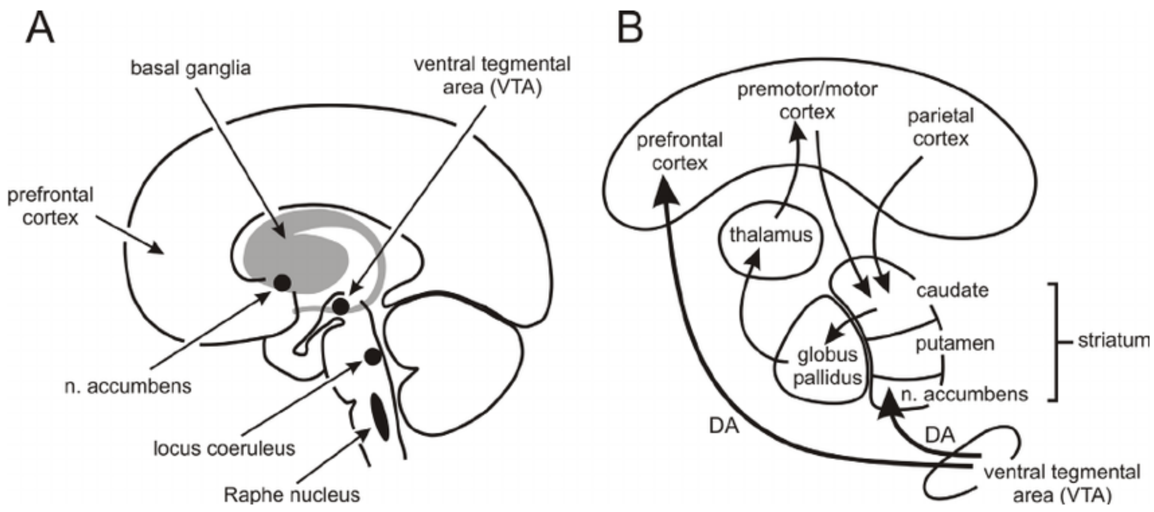


Reinforcement Learning

* Take actions in an environment so as to maximize some notion of cumulative reward.

* correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.

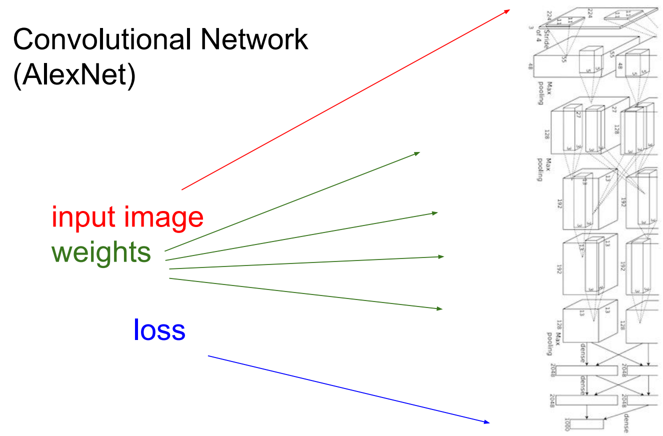
* finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

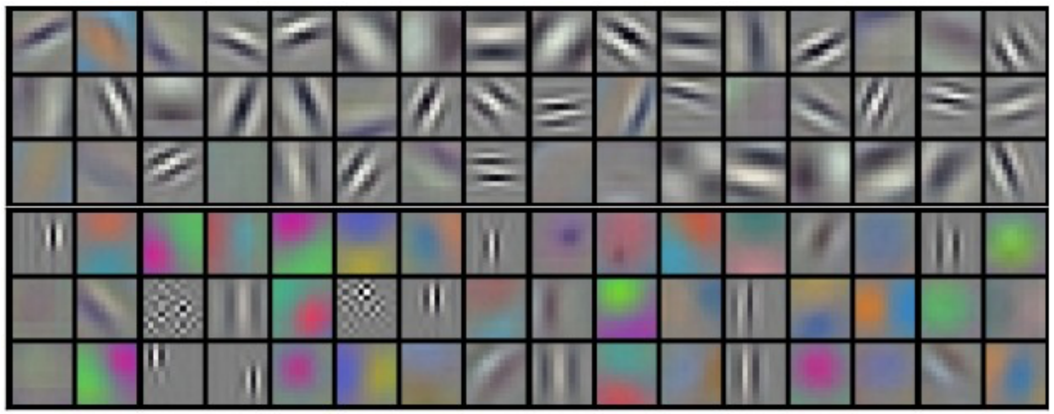


DA = Dopamine = actual reward -expected reward

Emergence of simple cell properties

* Batch of images each with its labels (car, truck, airplane...)
* Loss how well does predicted label match actual label
* Minimize loss
* Emergence of simple cell properties
    * Hubel and Weisel showed that cat responded to oriented bars
    * Learned visual fields of CNN are invariably oriented fiters





Finally the system perspective

* Google **trained** the neural networks on **30 million moves** from games played by **human experts**, until it could predict the human move 57 percent of the time
* Not clear how many self-plays they did
* A version of Alpha-Go used **1202 CPUs and 176 GPUs**
* Could not find details of Alpha-Go system
  * For people interested in Google Brain system look for videos by Jeff Dean

Graphical Processing Units

* GPUs have thousands of cores to process parallel workloads efficiently
* Modern deep neural networks are mostly trained by variants of stochastic gradient decent algorithms (SGD).
    * As SGDs contain high arithmetic density, GPUs are excellent for this type of algorithms.

Deep Learning systems

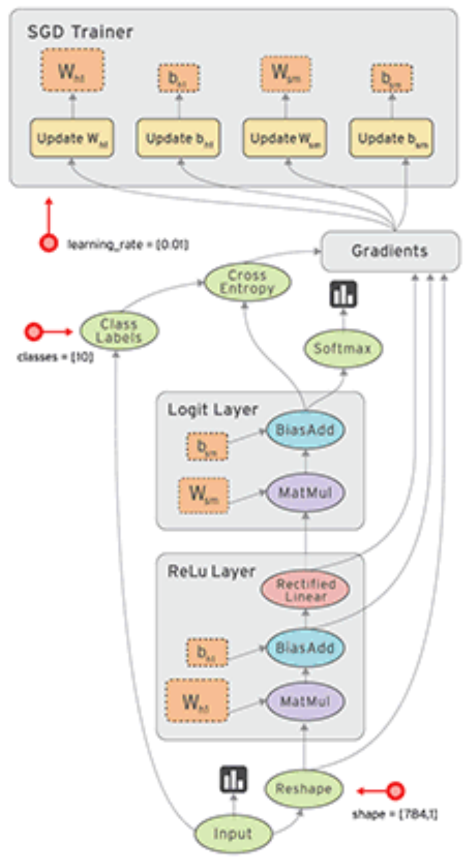
* Using multiple machines in a large cluster to increase the available computing power, ("scaling out")
* Leveraging GPUs, which can perform more arithmetic than typical CPUs ("scaling up").
* GPU clusters — Communication is the bottleneck
* Use of high- end networking infrastructure to remove the communications bottleneck between servers
  * incorporates Infiniband interconnects, which are dramatically faster (in terms of both bandwidth and latency) than typical Ethernet networks.

DL Parallelism

* In "**data parallelism**" mode, each GPU keeps a complete copy of the neural network parameters but computes a gradient using a different subset of the training data.
  * network parameters must fit on a single GPU—limiting us to, say, 250 million floating-point parameters (1 GB of storage).
  * computing a gradient for these parameters in just milliseconds per training image,
  * yet copying parameters or gradients to other machines will take at least 8 seconds over commodity Ethernet
* In "**model parallelism**" mode, each GPU is responsible for only a piece of the whole neural network,
  * reduces band- width requirements considerably
  * but also requires frequent synchronization (usually once for each forward- or backward-propagation step).
  * works well for GPUs in a single server (which share a high- speed bus)

DL Software

* The second major problem with building larger systems is a **software challenge: managing computation and communication amongst many GPUs** significantly complicates algorithm design.
* TensorFlow, Theano —**Symbolic Graph**
    * Nodes are assigned to computational devices and execute asynchronously and in parallel once all the tensors on their incoming edges becomes available.



Deep Image: Scaling up Image Recognition (Baidu)

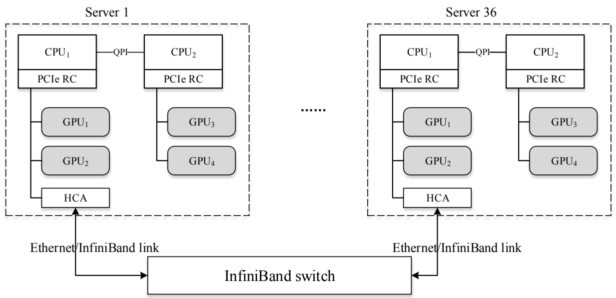
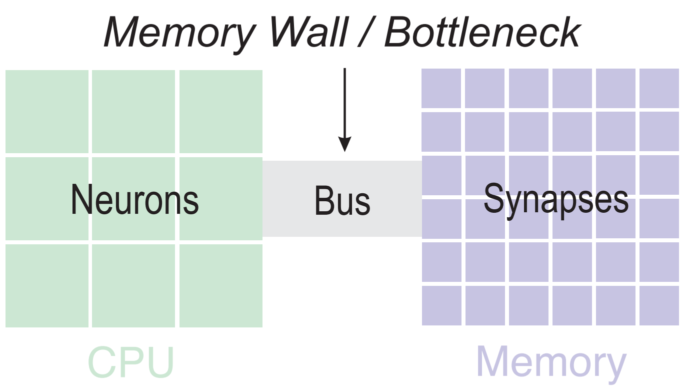
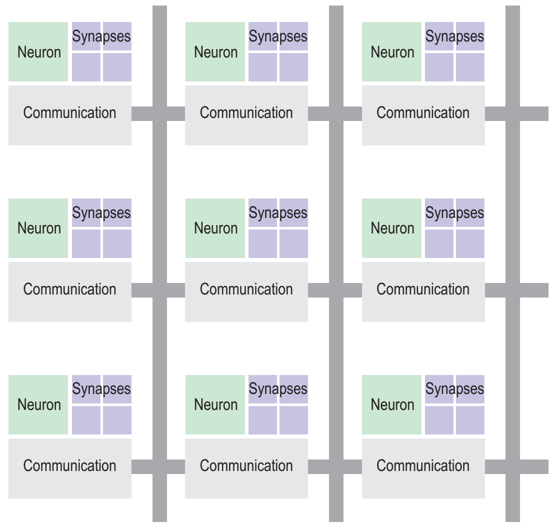


Figure 1: Hardware Architecture. Each server has four Nvidia K40m GPUs and one InfiniBand adapter.

* 2 six-core Intel Xeon
* 4 NVidia K40 GPU
* Infinband interconnect 56Gb/s
* RDMA - computers in a network to exchange data in main memory without involving the processor, cache, or operating system of either computer.

Neuro-morphic architecture





* Von Neumann

Separates memory and processor

Sequential, centralized processing

Ever increasing clock rates,

high active power

Huge passive power

Programmed system, hard-wired, fault-prone

Algorithms and analytics

* Neuro-morphic architecture

Integrates memory and processor

Parallel, distributed processing

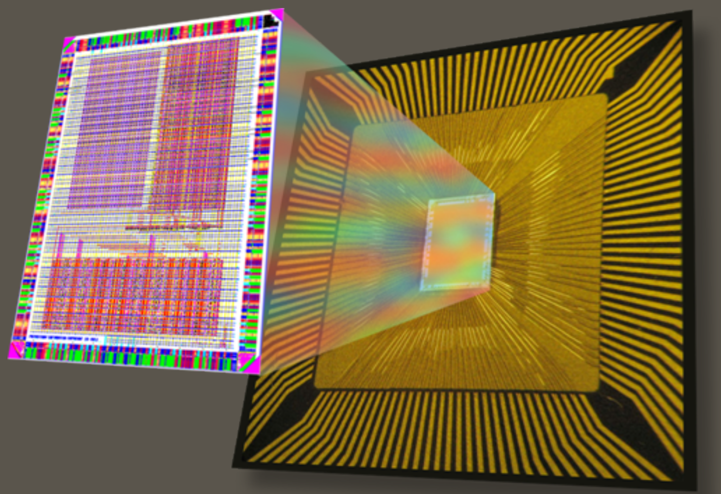
Event-driven, low active power

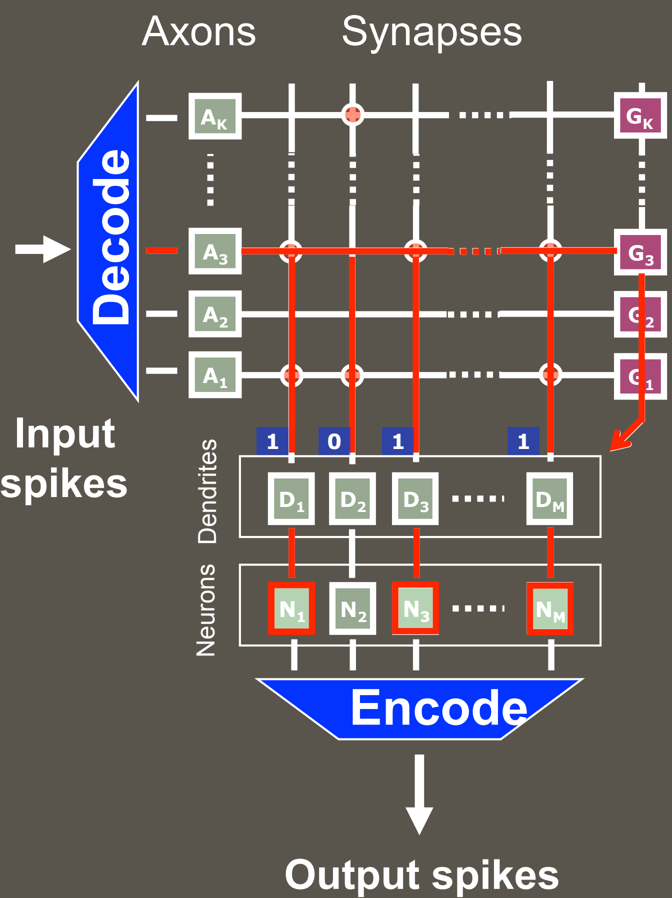
Does "nothing" better, low passive power

Learning system, reconfigurable, fault-tolerant

Substrate and pattern recognition

1





* 4.2 mm2 footprint
* 45 nm CMOS-SOI
* 256 neurons 262k programmable synapses
* Event-driven, 45pJ/spike

Summary

* Board games mix of computation and intuition
* AlphaGo serves intuition by CNN
    * serves computation by MCTS
* CNN effective representation of inputs
    * Biologically inspired
* GPU systems used to train CNN
    * can effectively parallelize dense maths operations such as gradient descent
* Communication bottleneck
    * Software has to be well designed
* Neuro-morphic architecture
    * Memory and Processor closely tied, no fetch involved, no bottleneck