Architecture Design for Deep Learning

Sargur N. Srihari
srihari@cedar.buffalo.edu

Topics

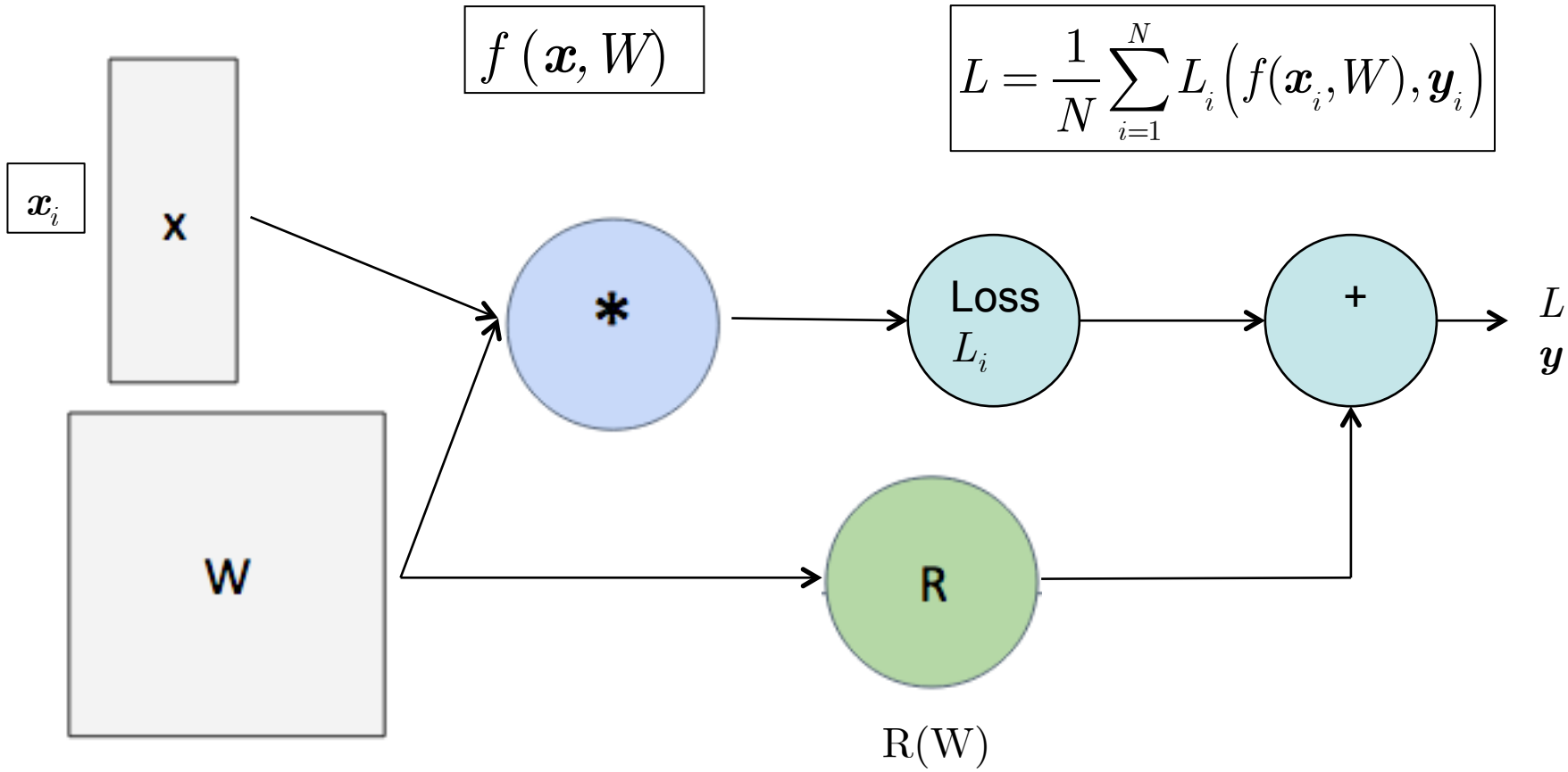
- Overview

1. Example: Learning XOR
2. Gradient-Based Learning
3. Hidden Units
4. Architecture Design
5. Backpropagation and Other Differentiation
6. Historical Notes

Topics in Architecture Design

1. Basic design of a neural network
2. Architecture Terminology
3. Chart of 27 neural network designs (generic)
4. Specific deep learning architectures
5. Equations for Layers
6. Theoretical underpinnings
   - Universal Approximation Theorem
   - No Free Lunch Theorem
7. Advantages of deeper networks

Basic design of a neural network



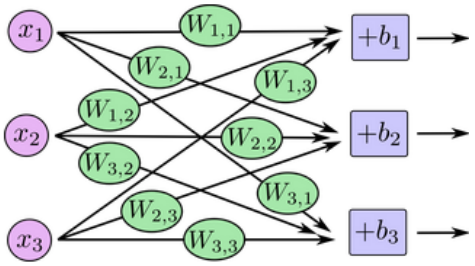
Design with two layers

- Most networks are organized into groups of units are called layers
  - Layers are arranged in a chain structure
- Each layer is a function of layer that preceded it
  - First layer is given by $\boldsymbol{h}^{(1)} = g^{(1)}(\boldsymbol{W}^{(1)\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}^{(1)})$
  - Second layer is $\boldsymbol{h}^{(2)} = g^{(2)}(\boldsymbol{W}^{(2)\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}^{(2)})$, etc.
    - Example

$$\boldsymbol{x} = [x_1, x_2, x_3]^{\mathrm{T}}$$

$$W_1^{(1)} = \left[W_{11} W_{12} W_{13}\right]^T, W_2^{(1)} = \left[W_{21} W_{22} W_{23}\right]^T, W_3^{(1)} = \left[W_{31} W_{32} W_{33}\right]^T$$

First Network layer



Network layer output

$$\begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{pmatrix}$$

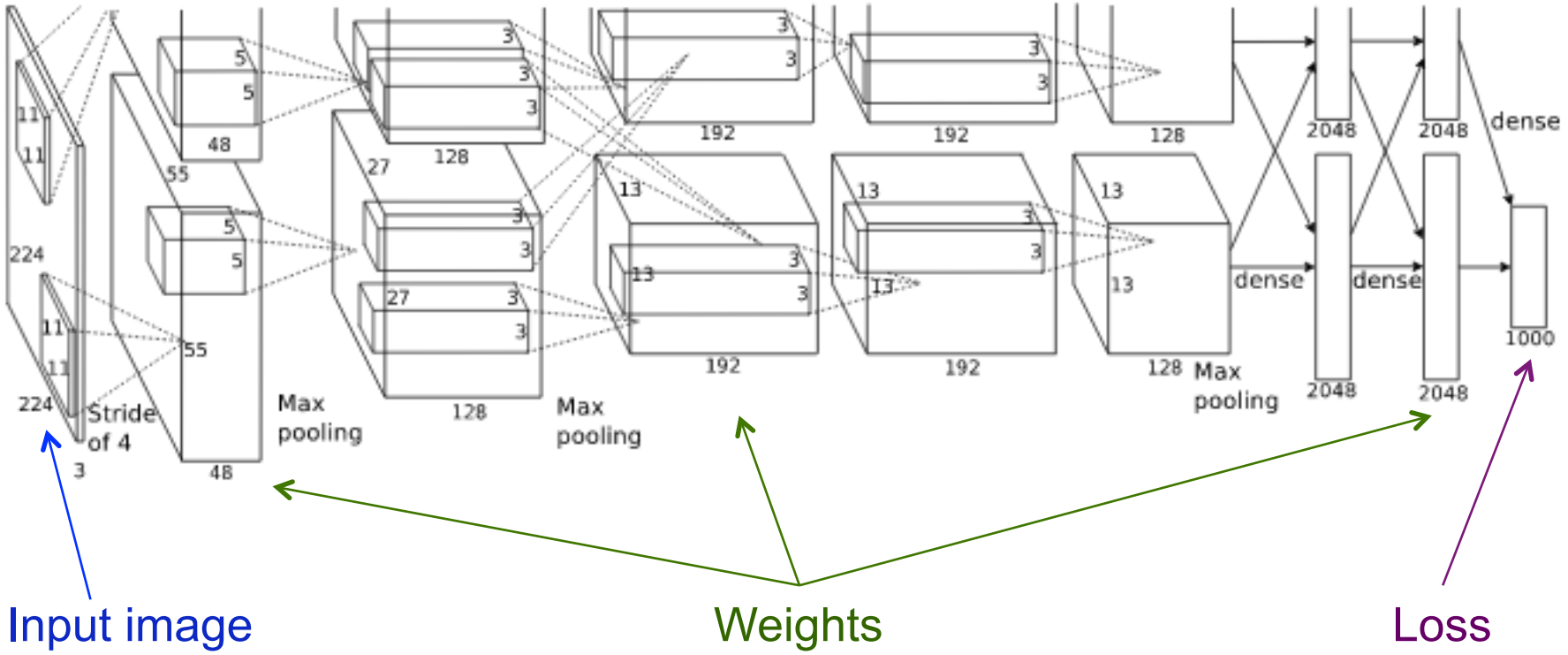
In matrix multiplication notation

$$\left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

Architecture Terminology

- The word *architecture* refers to the overall structure of the network:
  - How many units should it have?
  - How the units should be connected to each other?
- Most neural networks are organized into groups of units called *layers*
  - Most neural network architectures arrange these layers in a chain structure
  - With each layer being a function of the layer that preceded it

A network with eight layers



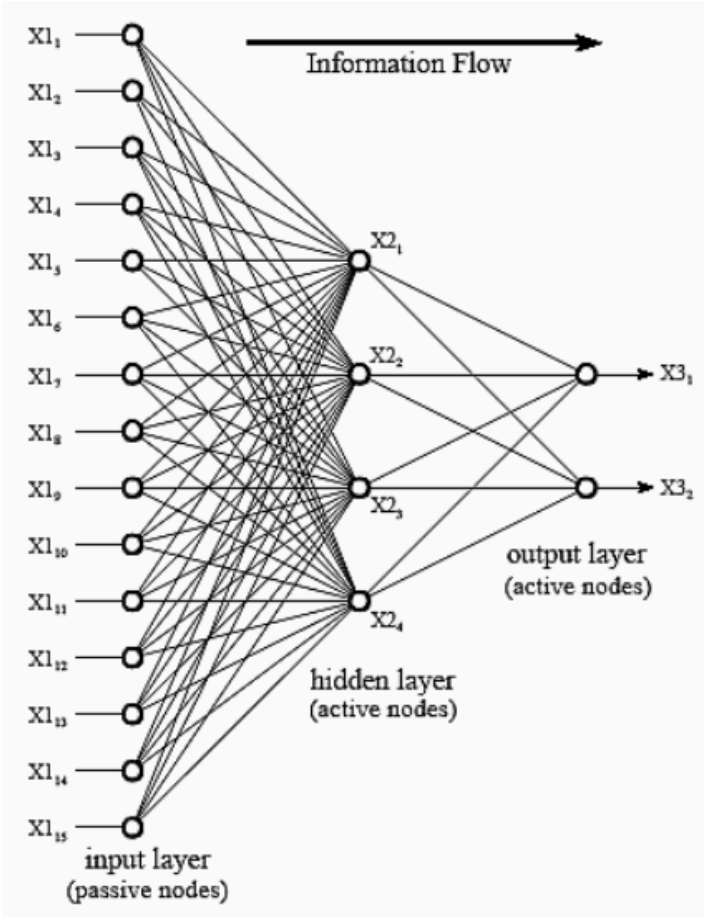
Alexnet: For ImageNet (1.2m high-res images) with 1000 classes.

First five layers: convolutional and remaining three fully-connected.

Output of last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels

Main Architectural Considerations

1. Choice of depth of network
2. Choice of width of each layer

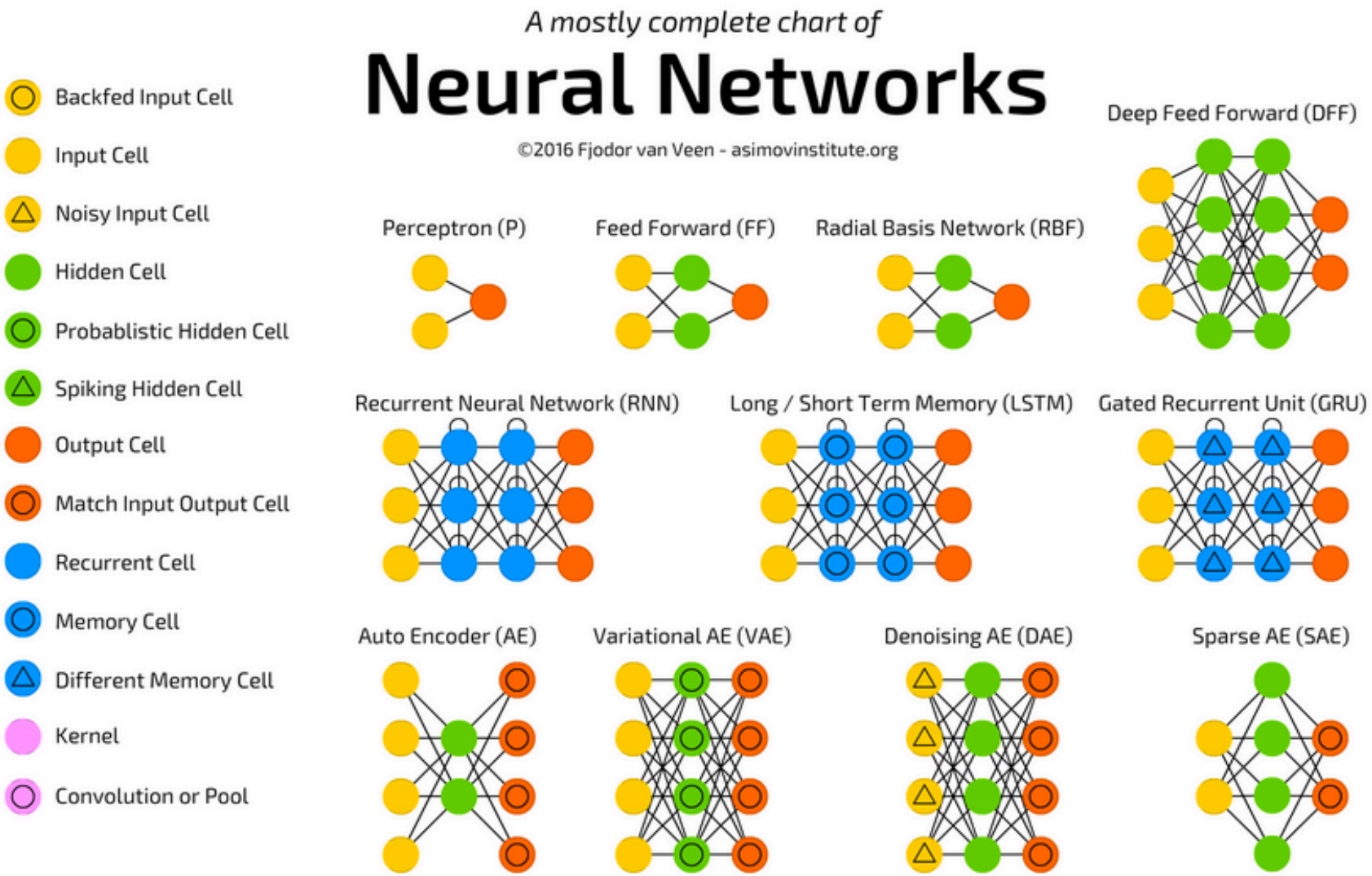


Network with even one hidden layer is sufficient to fit training set

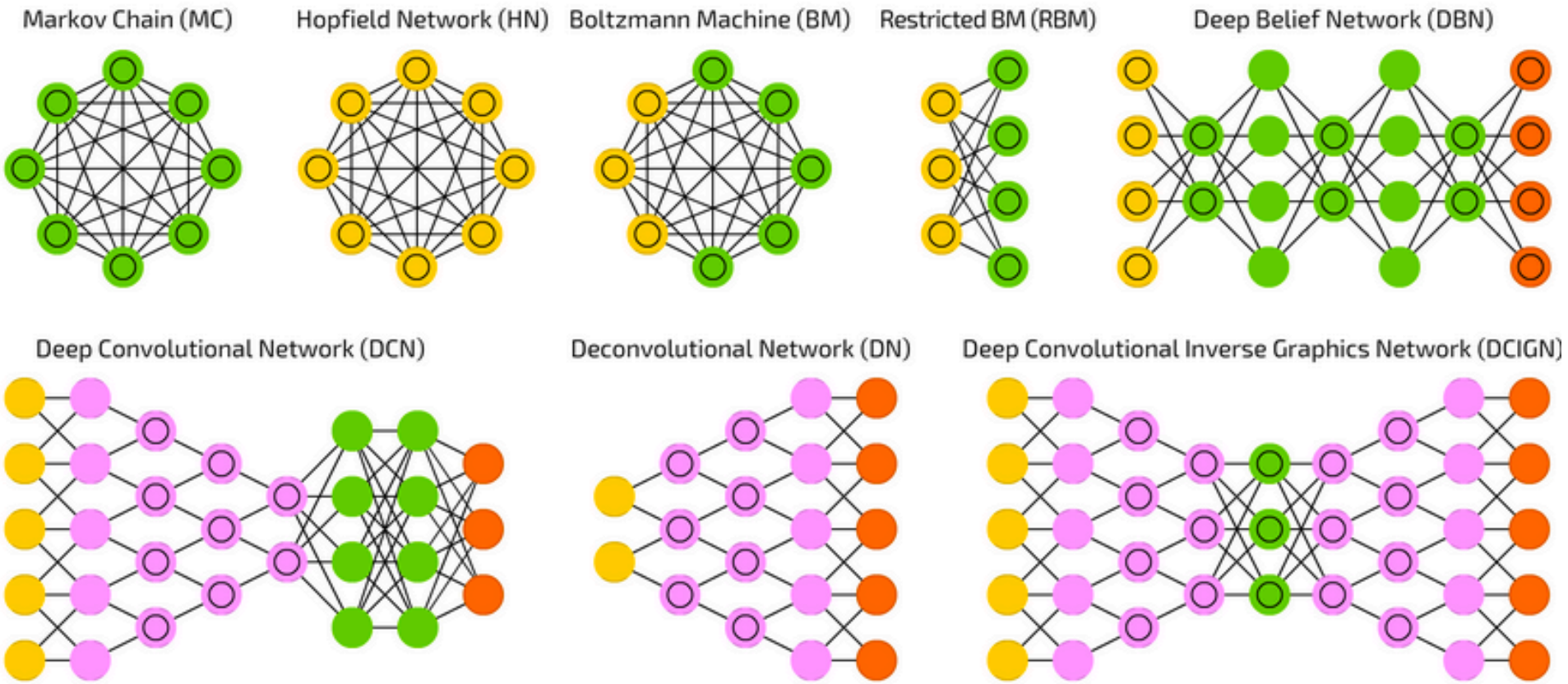
Advantage of Deeper Networks

- Deeper networks have
  - Far fewer units in each layer
  - Far fewer parameters
  - Often generalize well to the test set
  - But are often more difficult to optimize
- Ideal network architecture must be found via experimentation guided by validation set error

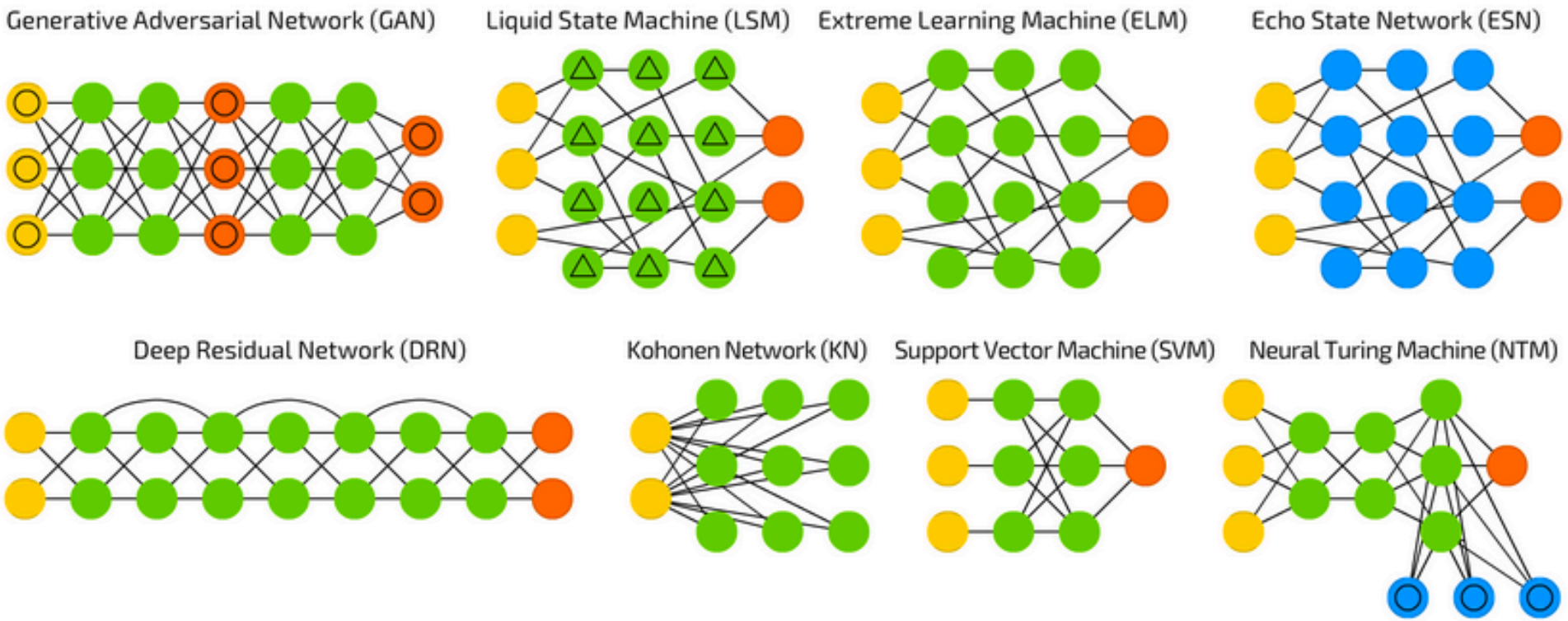
Generic Neural Architectures (1-11)



Generic Neural Architectures (12-19)

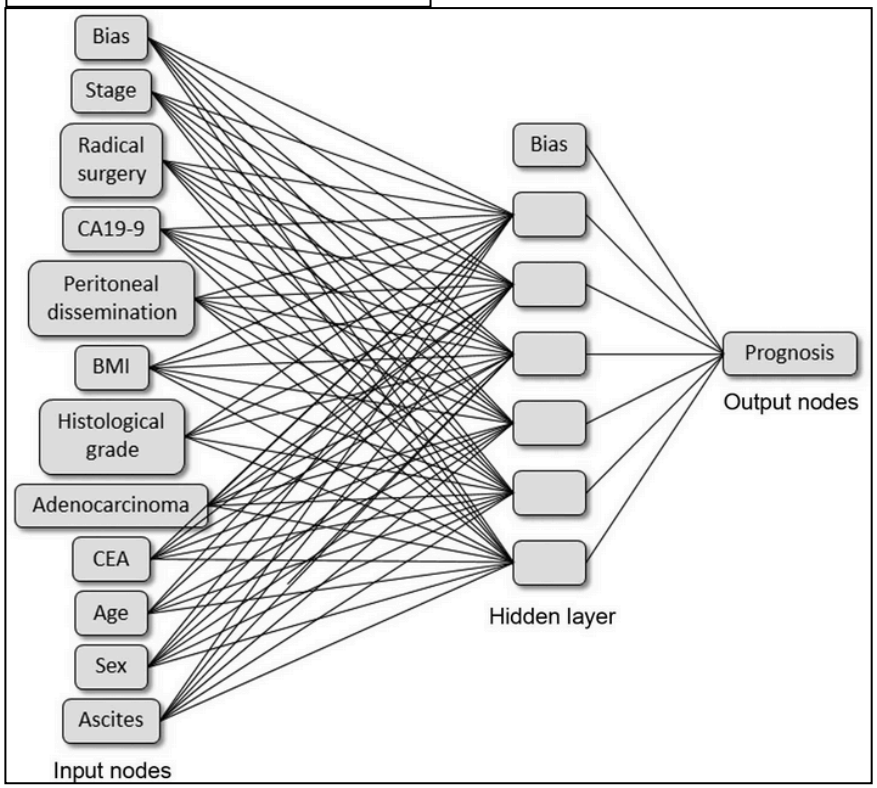


Generic Neural Architectures (20-27)

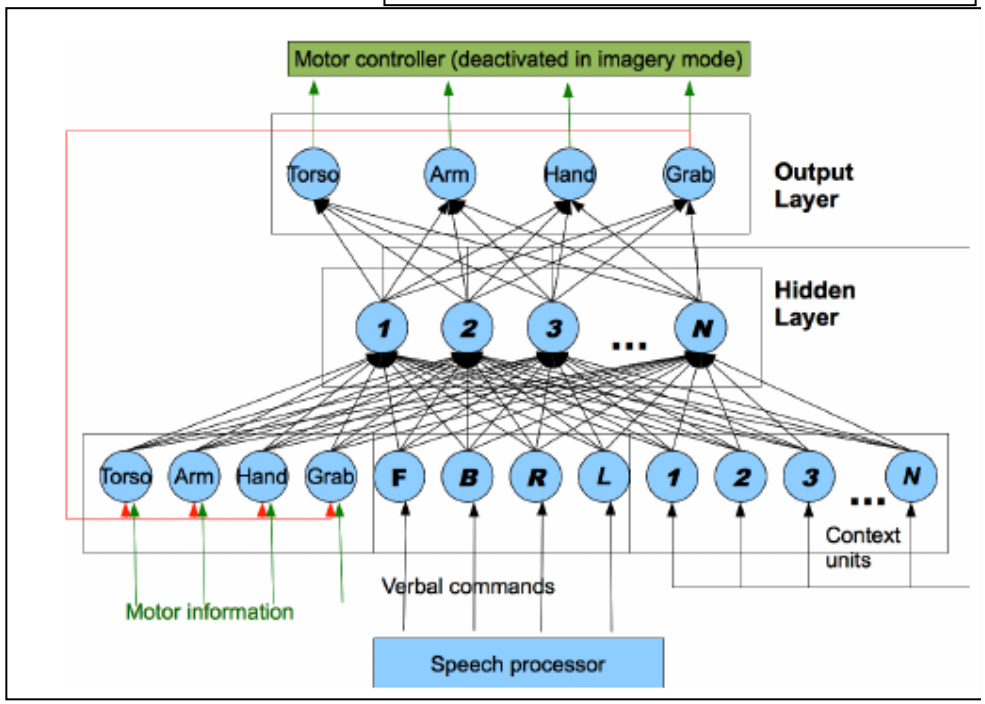


Specific Application Architectures

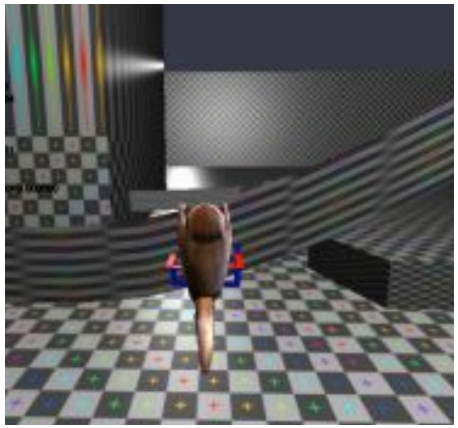
Cancer Prognosis

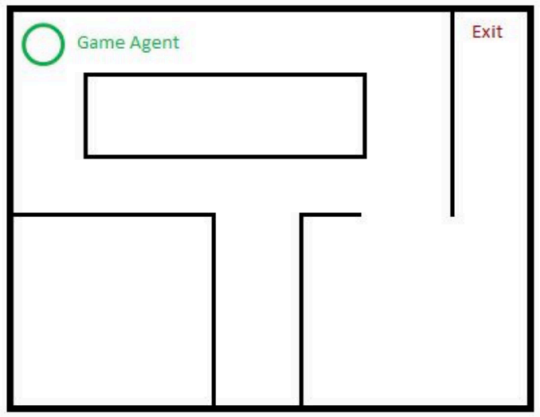


Architecture to study how images in the mind can influence movements and motor skills (RNN)

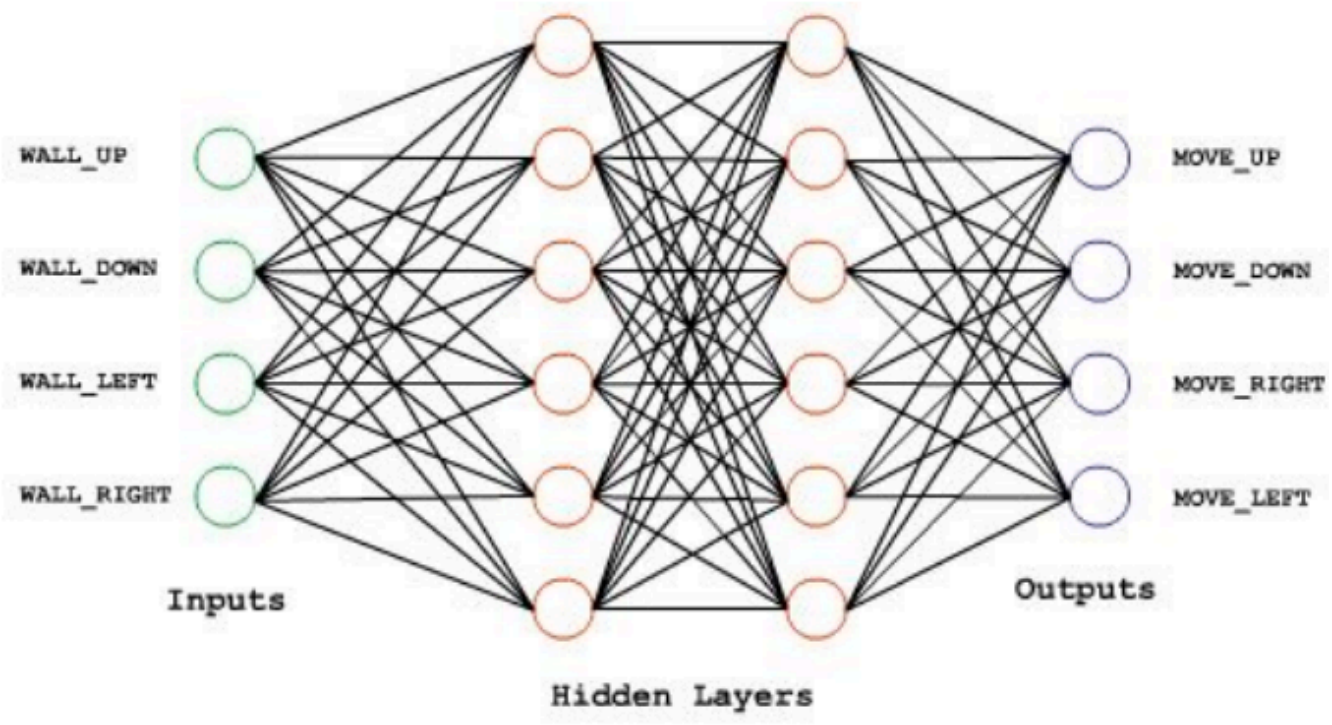


An architecture for Game Design

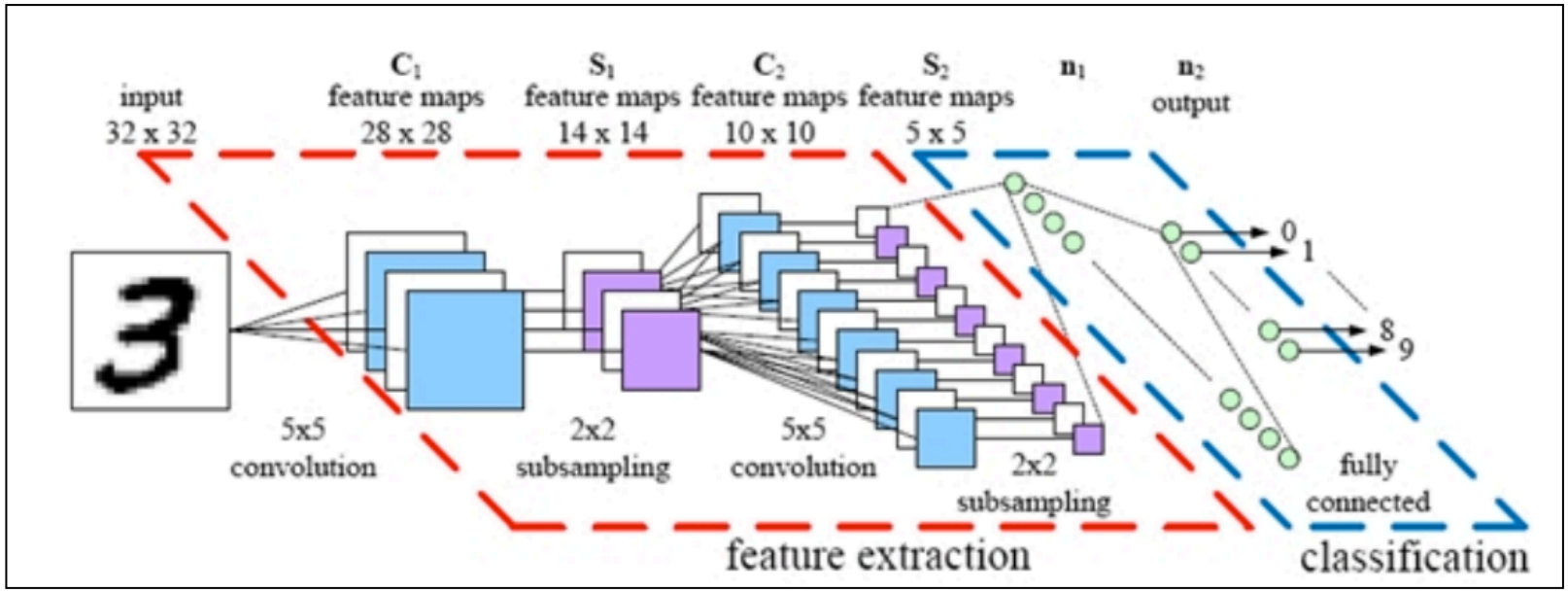


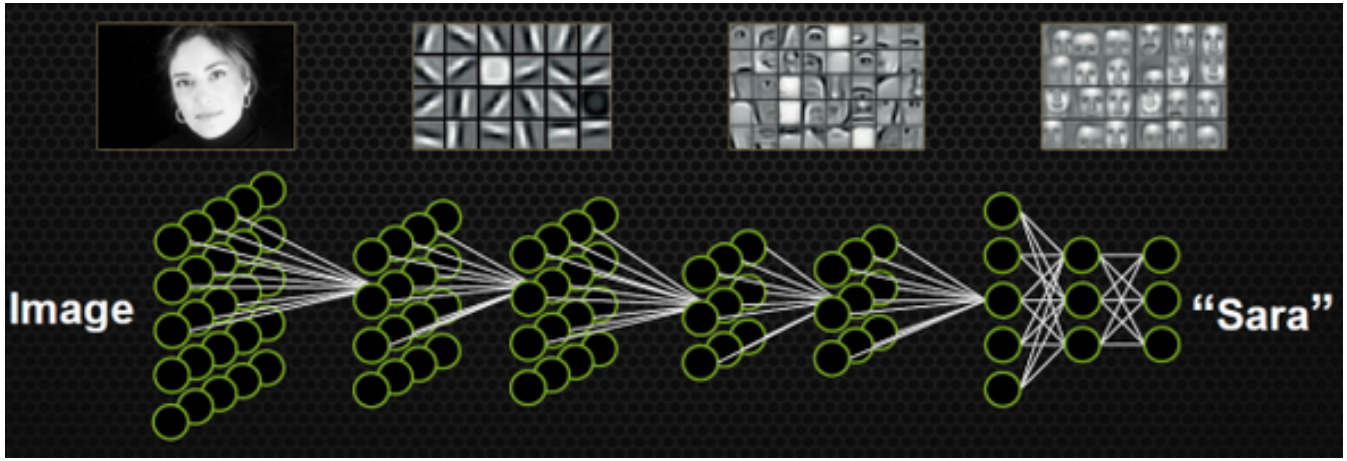


Maze Game Recording Output File							
Inputs: Wall Directions				Outputs: Game Agent Direction			
UP	DOWN	LEFT	RIGHT	UP	DOWN	LEFT	RIGHT
1	0	0	0	0	1	0	0
1	0	1	0	0	1	0	1
0	1	1	0	1	0	0	1
0	0	1	0	0	0	0	1



CNN Architectures

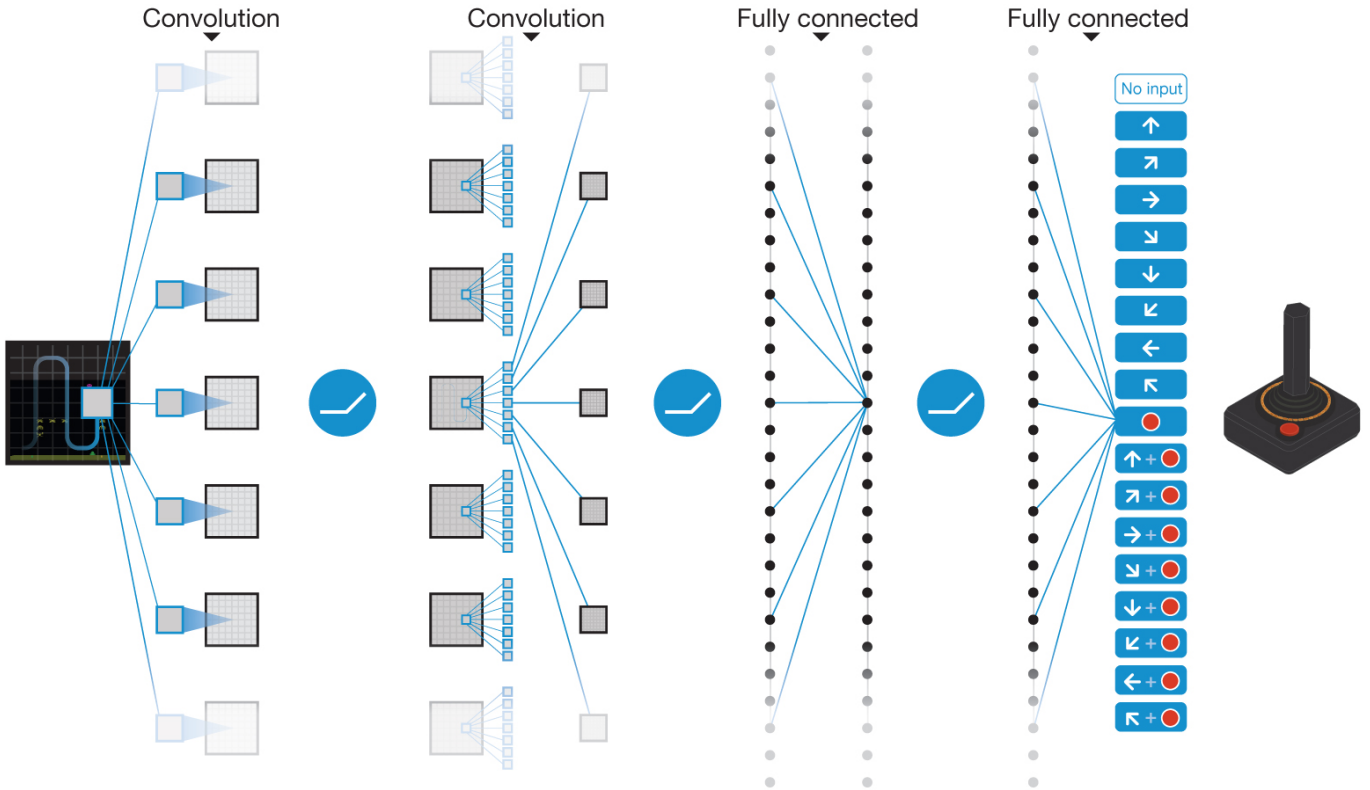




More complex features captured In deeper layers

Architecture Blending Deep Learning and Reinforcement Learning

- Human Level Control Through Deep Reinforcement Learning



Theoretical underpinnings

- Mathematical theory of Artificial Neural Networks
  - Linear versus Nonlinear Models
  - Universal Approximation Theorem
- No Free Lunch Theorem
- Size of network

Linear vs Nonlinear Models

- A linear model with features-to-output via matrix multiplication only represent linear functions
  - They are easy to train
    - Because loss functions result in convex optimization
- Unfortunately often we want to learn nonlinear functions
  - Not necessary to define a family of nonlinear functions
  - Feedforward networks with hidden layers provide a universal approximation framework

Universal Approximation Theorem

- A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n, under mild assumptions on the activation function
  - Simple neural networks can *represent* a wide variety of interesting functions when given appropriate parameters
  - However, it does not touch upon the algorithmic learnability of those parameters.

Formal UA Theorem

- Let $\phi(\cdot)$ be continuous (*activation function*)
  - Non-constant, bounded, monotonic increasing function
- I_m is the unit hypercube $[0,1]^m$ (m *inputs, values in* $[0,1]$)
- Space of continuous functions on I_m is $C(I_m)$
  - Then, given any function $f \in C(I_m)$ and $\varepsilon > 0$, there exists an integer N (*no. of outputs*)
  - real constants $v_i, b_i \in R$ (*output weights, input bias*)
  - real vectors $w_i \in R^m$, $i = 1, \cdots, N$ (*input weights*)
  - such that we may define:

$$F(\boldsymbol{x}) = \sum_{i=1,..N} v_i \, \phi\left(w_i^{\mathrm{T}} \boldsymbol{x} + b_i\right)$$

as an approximation of f where f is independent of ϕ; i.e.,

$$|F(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon \quad \text{for all } \boldsymbol{x} \in I_m$$

i.e., functions of the form $F(\boldsymbol{x})$ are dense in $C(I_m)$

Implication of Theorem

- A feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation function (such as logistic sigmoid) can approximate:
  - Any Borel measurable function from one finite-dimensional space to another
  - With any desired non-zero amount of error
  - Provided the network is given enough hidden units
- The derivatives of the network can also approximate derivatives of function well

Applicability of Theorem

- Any continuous function on a closed and bounded subset of R^n is Borel measurable
  - Therefore approximated by a neural network
- Discrete case:
  - A neural network may also approximate any function mapping from any finite dimensional discrete space to another
- Original theorems stated for activations that saturate for very negative/positive arguments
  - Also proved for wider class including ReLU

Theorem and Training

- Whatever function we are trying to learn, a large MLP will be able to represent it
- However we are not guaranteed that the training algorithm will learn this function
  1. Optimizing algorithms may not find the parameters
  2. May choose wrong function due to over-fitting
- No Free Lunch: There is no universal procedure for examining a training set of samples and choosing a function that will generalize to points not in training set

Feed-forward & No Free Lunch

- Feed-forward networks provide a universal system for representing functions
  - Given a function, there is a feed-forward network that approximates the function
- There is no universal procedure for examining a training set of specific examples and choosing a function that will generalize to points not in training set

On Size of Network

- Universal Approximation Theorem
  - Says there is a network large enough to achieve any degree of accuracy
  - but does not say how large the network will be
- Bounds on size of the single-layer network exist for a broad class of functions

  - But worst case is exponential no. of hidden units
    - No. of binary functions on vectors $\boldsymbol{v} \in \{0,1\}^n$ is 2^{2^n}
      - e.g. there are 16 functions of 2 variables
    - Selecting one such function requires 2^n bits which will require $O(2^n)$ degrees of freedom

Summary/Implications of Theorem

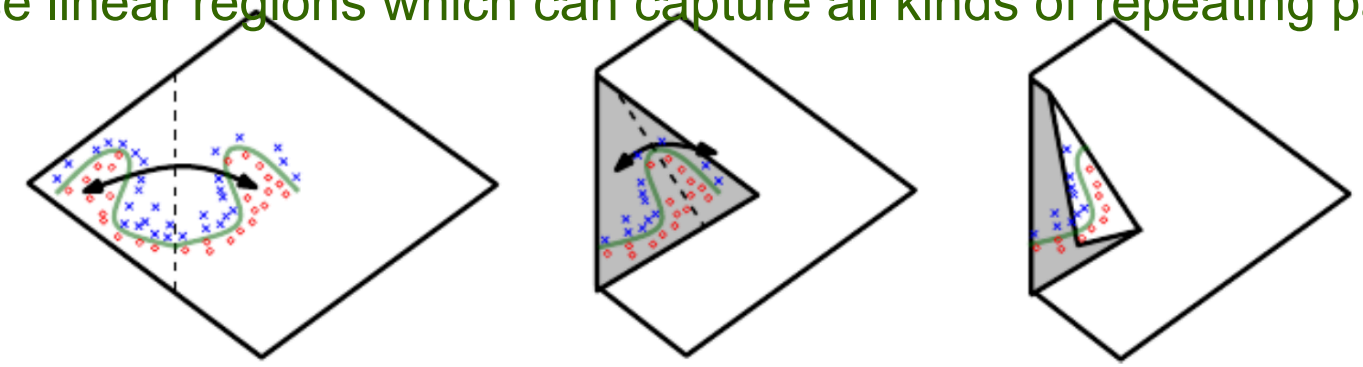
- A feedforward network with a single layer is sufficient to represent any function
- But the layer may be infeasibly large and may fail to generalize correctly
- Using deeper models can reduce no. of units required and reduce generalization error

Function Families and Depth

- Some families of functions can be represented efficiently if depth $>d$ but require much larger model if depth $<d$
- In some cases no. of hidden units required by shallow model is exponential in n
  - Functions representable with a deep rectifier net can require an exponential no. of hidden units with a shallow (one hidden layer) network
    - Piecewise linear networks (which can be obtained from rectifier nonlinearities or maxout units) can represent functions with a no. of regions that is exponential in d

Advantage of deeper networks

Absolute value rectification creates mirror images of function computed on top Of some hidden unit, wrt the input of that hidden unit.
Each hidden unit specifies where to fold the input space in order to create mirror responses.
By composing these folding operations we obtain an exponentially large no. of piecewise linear regions which can capture all kinds of repeating patterns



Has same output for every pair of mirror points in input. Mirror axis of symmetry Is given by weights and bias of unit. Function computed on top of unit (green decision surface) will be a mirror image of simpler pattern across axis of symmetry

Function can be obtained By folding the space around axis of symmetry

Another repeating Pattern can be folded on Top of the first (by another downstream unit) to obtain another symmetry (which is now repeated four times with two hidden layers)

Theorem on Depth

- The no. of linear regions carved out by a deep rectifier network with d inputs, depth l and n units per hidden layer is

$$O\left(\binom{n}{d}^{d(l-1)} n^d\right)$$

  – i.e., exponential in the depth l.

- In the case of maxout networks with k filters per unit, the no. of linear regions is

$$O\left(k^{(l-1)+d}\right)$$

- There is no guarantee that the kinds of functions we want to learn in AI share such a property

Statistical Justification for Depth

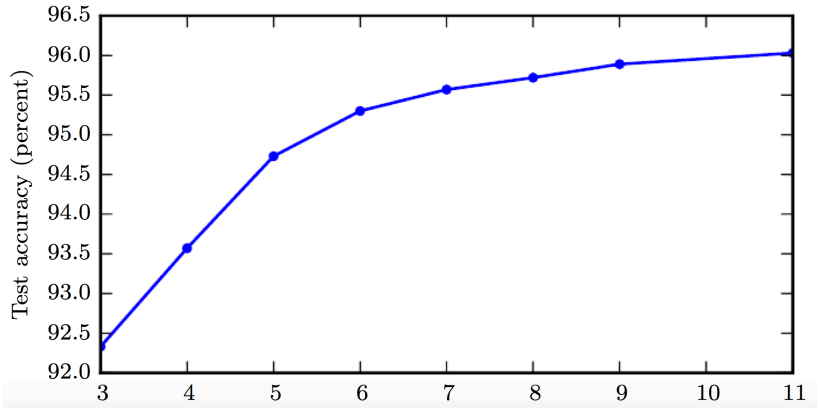
- We may want to choose a deep model for statistical reasons
- Any time we choose a ML algorithm we are implicitly stating a set of beliefs about what kind of functions that algorithm should learn
- Choosing a deep model encodes a belief that the function should be a composition several simpler functions

Intuition on Depth

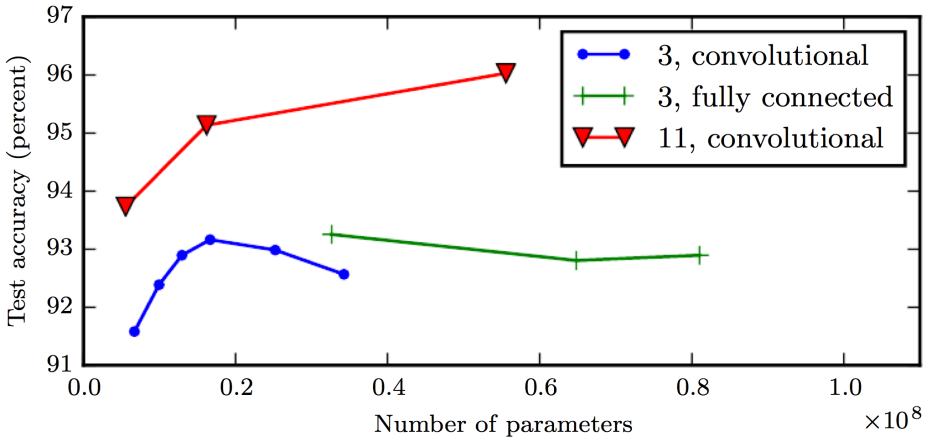
- We can interpret the use of a deep architecture as expressing a belief that the function we want to learn is a computer program consisting of m where each where each step makes use of the previous step's output
- The intermediate outputs are not necessarily factors of variation, but can be analogous to counters or pointers used for organizing processing
- Empirically greater depth results in better generalization

Empirical Results

- Deeper networks perform better



Test accuracy consistently increases with depth



Increasing parameters without increasing depth is not as effective

- Deep architectures indeed express a useful prior over the space of functions the model learns

Other architectural considerations

- Specialized architectures are discussed later
- Convolutional Networks
  - Used for computer vision
- Recurrent Neural Networks
  - Used for sequence processing
  - Have their own architectural considerations

Non-chain architecture

- Layers connected in a chain is common
- Skipping going from layer i to layer $i+2$ or higher
  - During learning, makes it easier for gradient to flow from output layers to layer nearer input

Connecting a pair of layers

- In the default neural network layer described by a linear transformation via a matrix W
- Every input unit connected to every output unit
- Specialized networks have fewer connections
  - Each unit in input layer is connected to only small subset of units in output layer
  - Reduce no. of parameters and computation for evaluation
  - E.g., CNNs use specialized patterns of sparse connections that are effective for computer vision