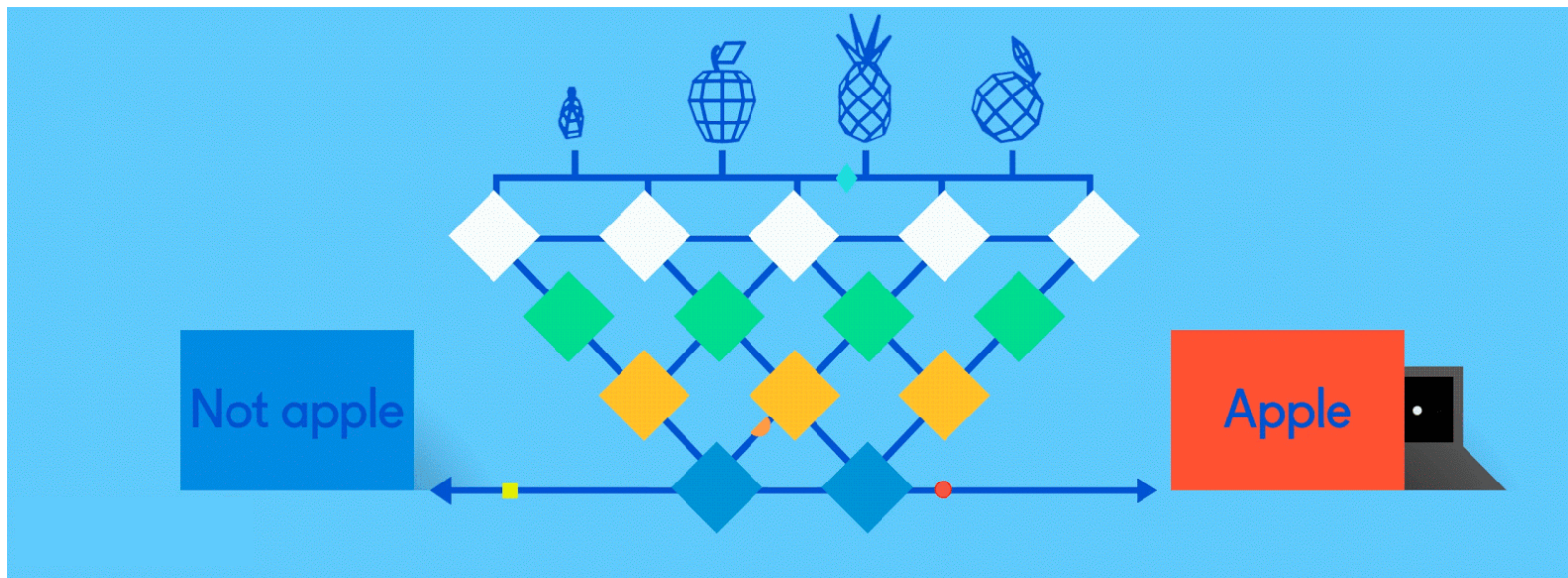
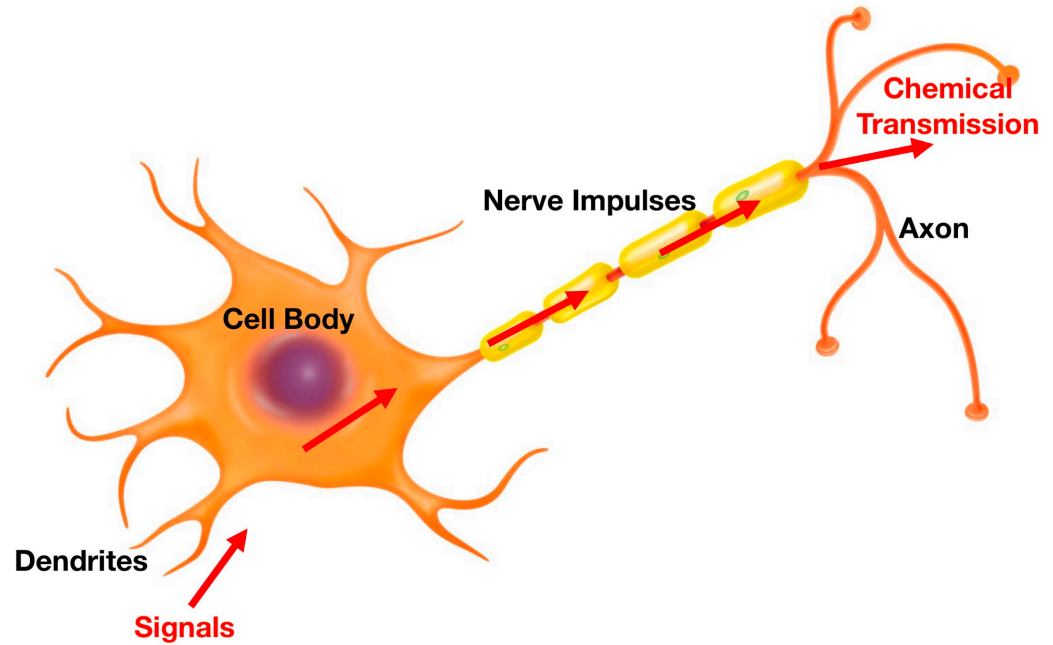


Neural Networks for Machine Learning introduction



Biological neural activity



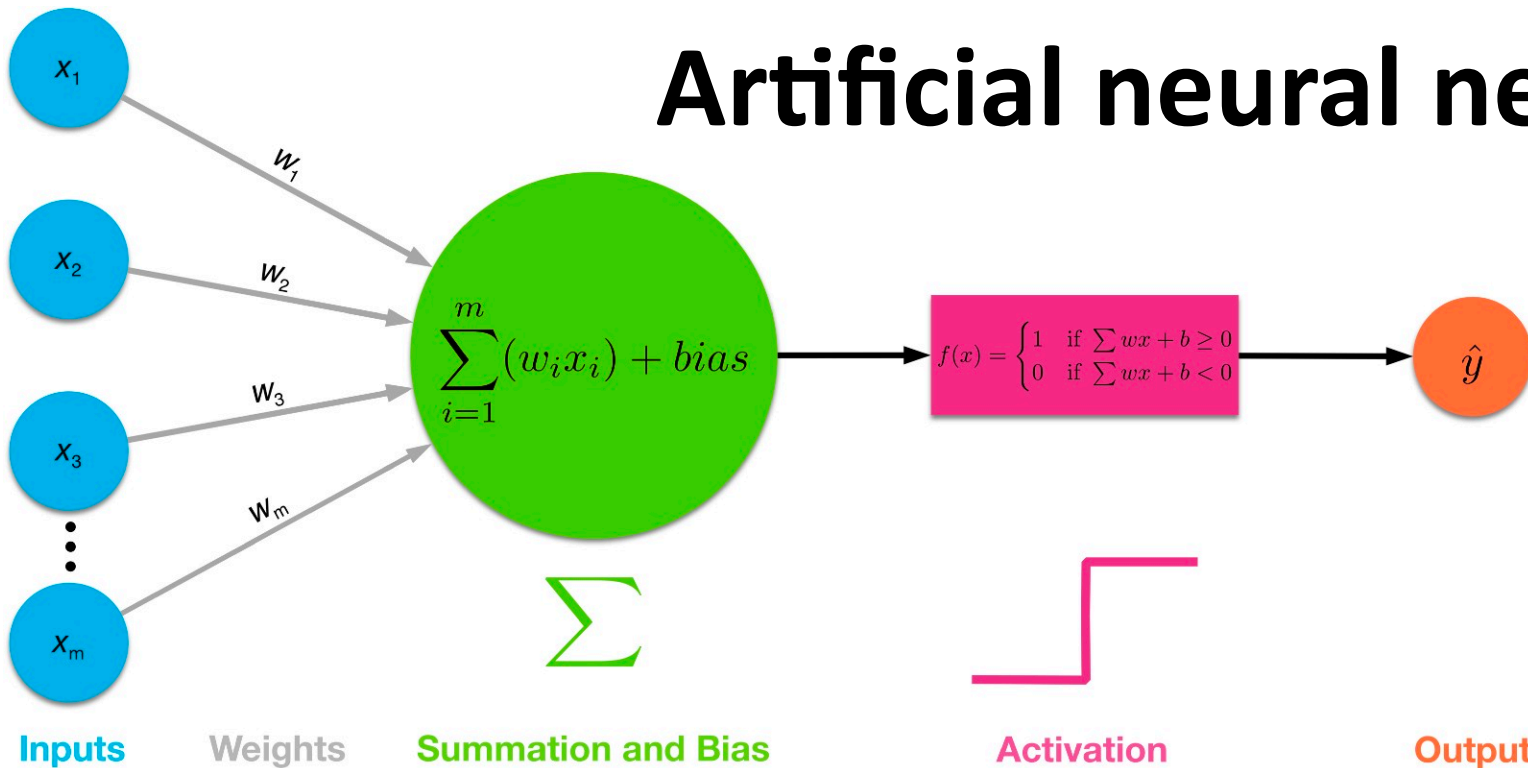
Neurons have body, axon and many dendrites

- In one of two states: firing and rest
- They fire if total incoming stimulus $>$ threshold

Synapse: thin gap between axon of one neuron and dendrite of another

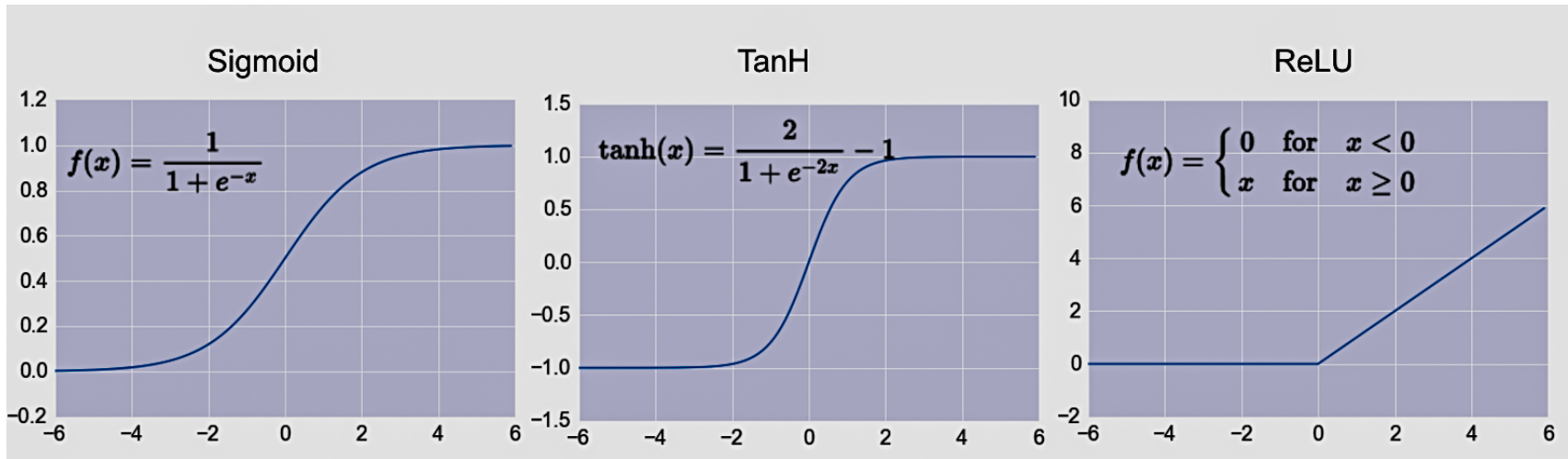
- Signal exchange

Artificial neural network



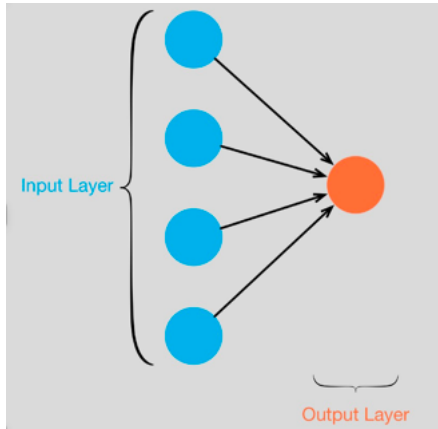
- Set of **nodes** with inputs and outputs
Node performs computation via its **activation function**
- **Weighted connections** between nodes
- Connectivity gives network architecture
- NN computations depend on connections & weights

Common Activation Functions



Choice of activation function depends on problem and available computational power

Single Layer Perceptron

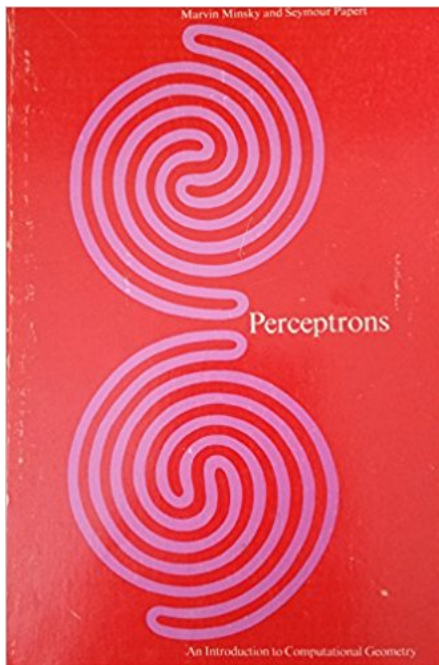


NEW NAVY DEVICE LEARNS BY DOING; Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

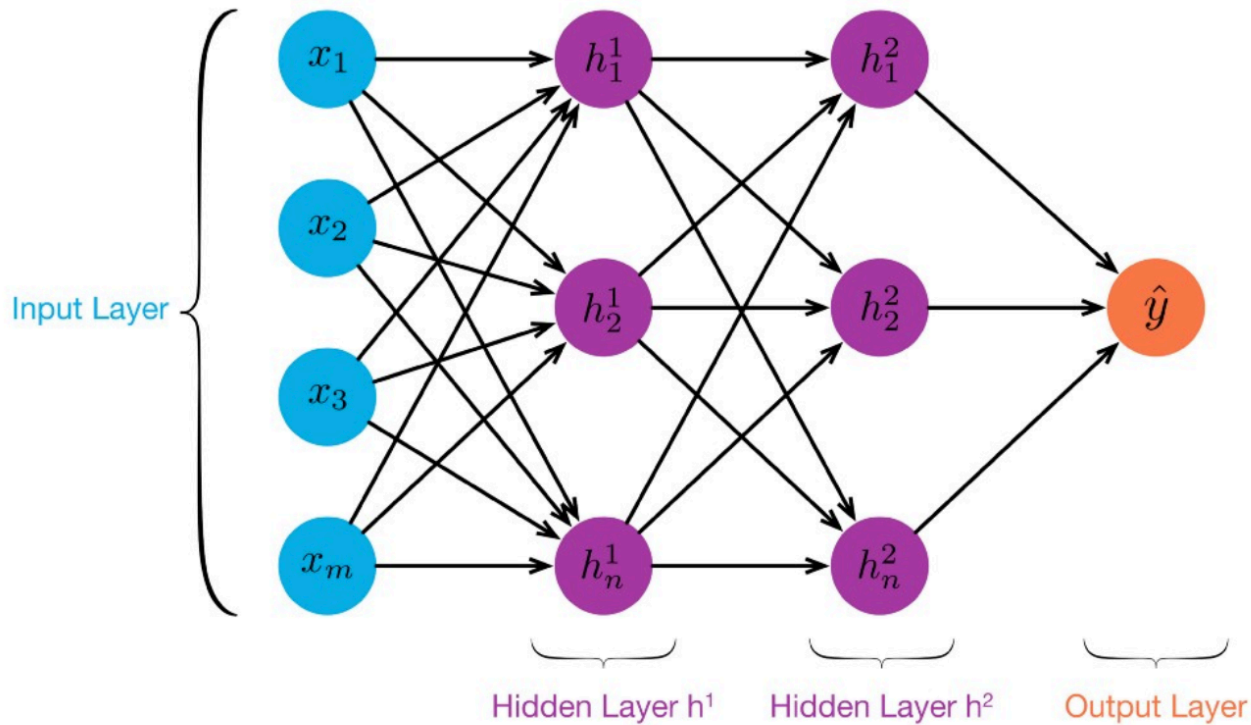
SPECIAL TO THE NEW YORK TIMES JULY 8, 1958



WASHINGTON, July 7 (UPI) -- The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.



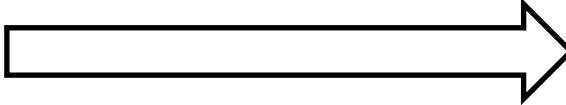
- Full 1958 NYT article above [here](#)
- Rosenblatt: it can **learn** to compute functions by learning weights on inputs from examples
- Not all functions ☹️, cf. [Perceptrons](#)

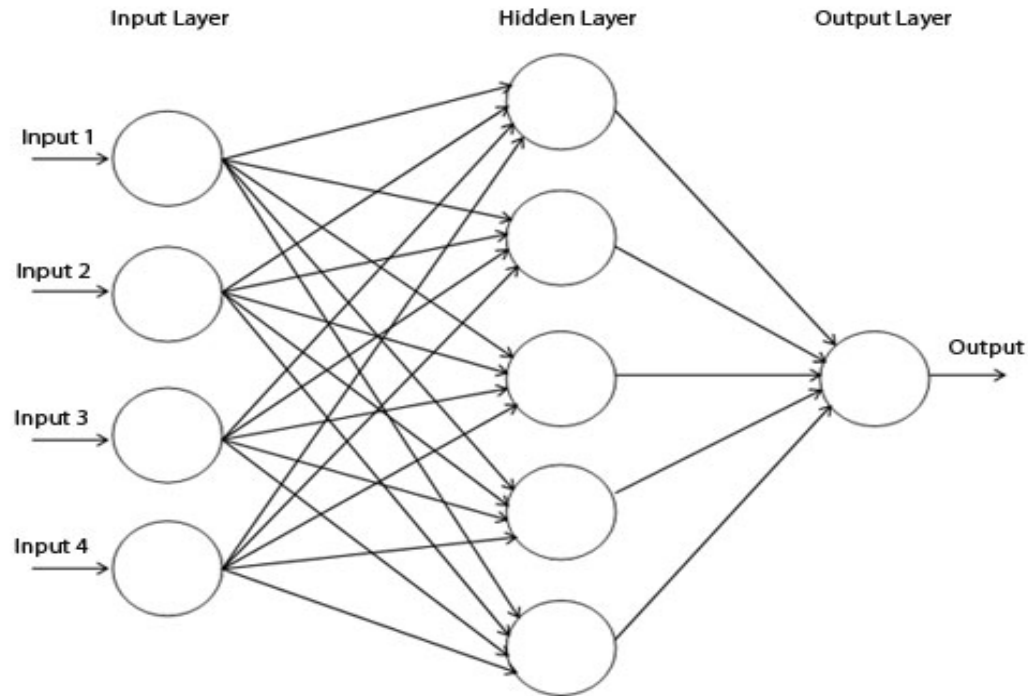


MLP: Multilayer Perceptron

- ≥ 1 “hidden layers” between inputs & output
- Can compute non-linear functions
- Training: adjust weights slightly to reduce error between output \mathbf{y} and target value \mathbf{t} ; repeat
- Introduced in 1980s, still used today

Backpropagation

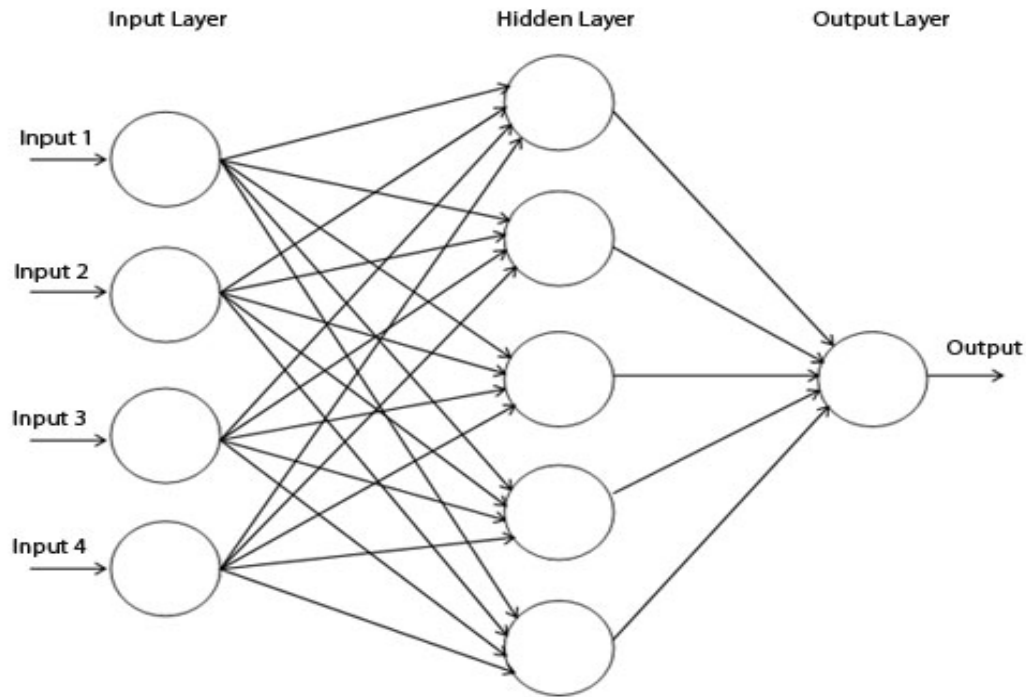
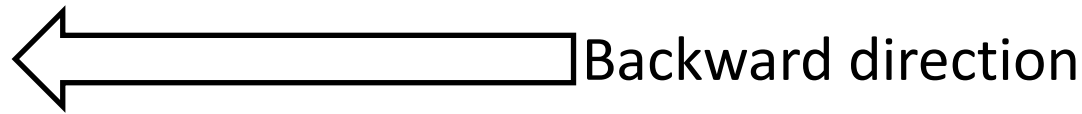
Forward direction 



Calculate network and error

Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986).
[Learning representations by back-propagating errors](#). *Nature*. 323 (6088): 533–536.

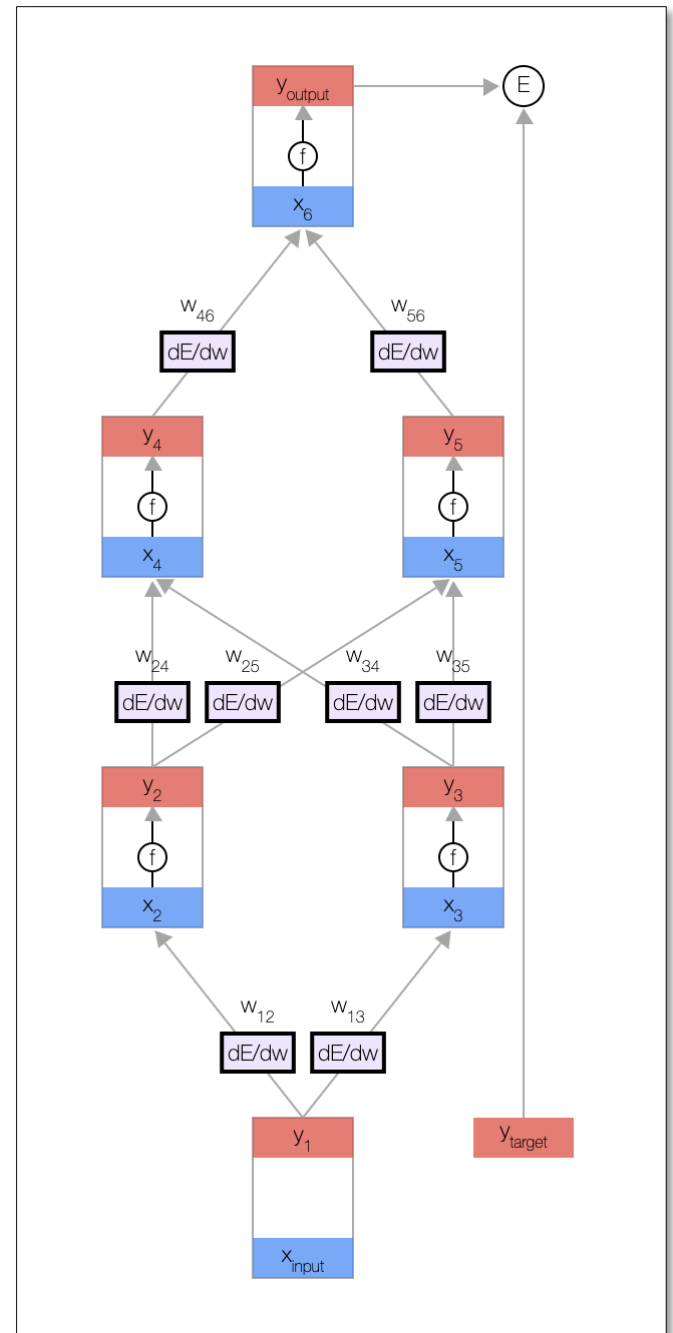
Backpropagation



Backpropagate: from output to input, recursively
compute $\frac{\partial E}{\partial w_{ij}} = \nabla_w E$ and adjust weights

Backprop Explained

- A simple animation shows how backprop works
- A MLP with one input, two hidden layers and one output
- Click on image or go to <http://bit.ly/471bpd> to view in your browser



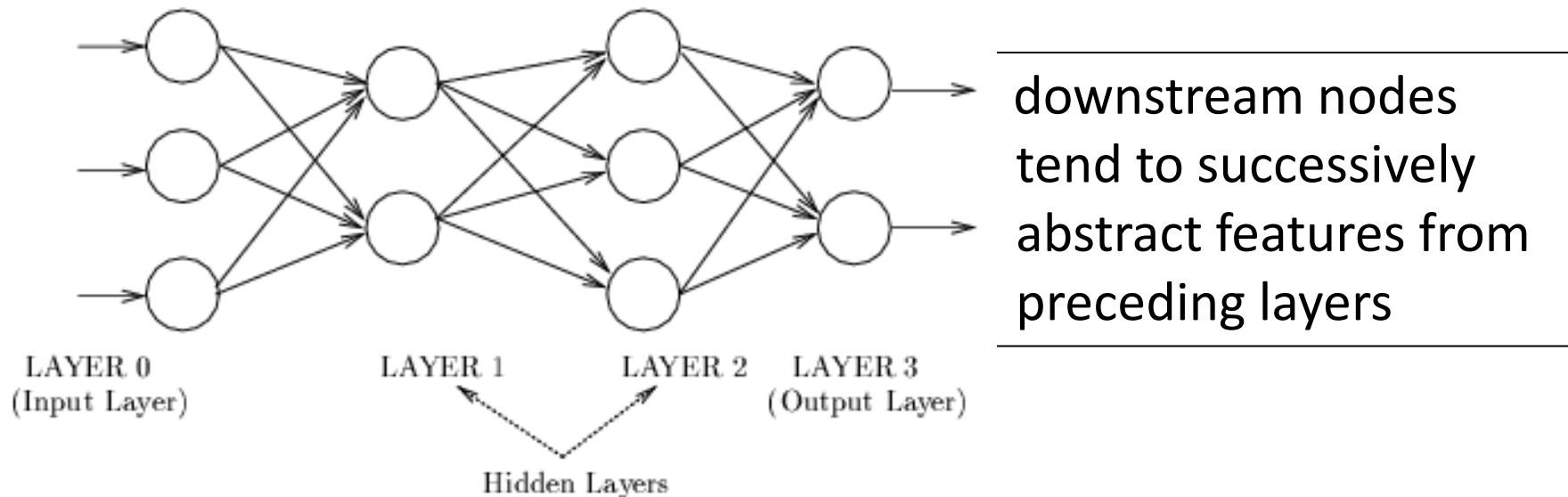
Neural Network Architectures

Current focus on large networks with different “architectures” suited for different kinds of tasks

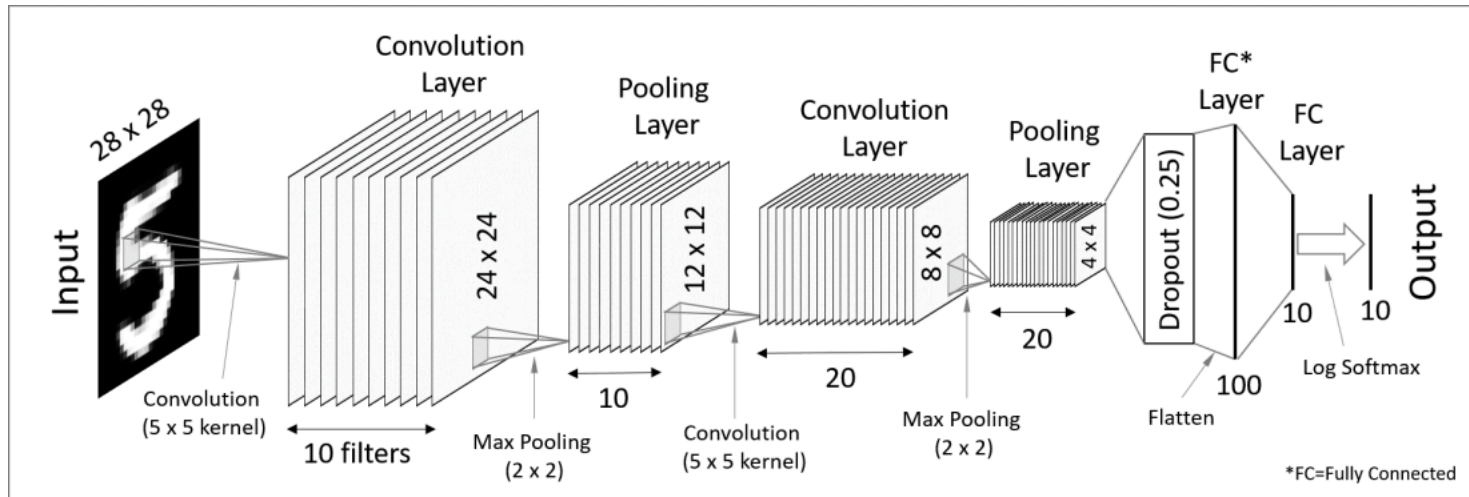
- Feedforward Neural Network
- CNN: Convolutional Neural Network
- RNN: Recurrent Neural Network
- LSTM: Long Short Term Memory
- GAN: Generative Adversarial Network

Feedforward Neural Network

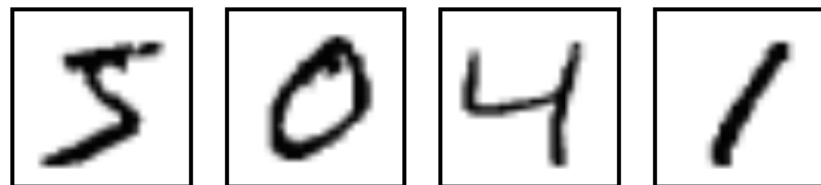
- Connections allowed from a node in layer i only to nodes in layer $i+1$
i.e., no cycles or loops
- Simple, widely used architecture.



CNN: Convolutional Neural Network

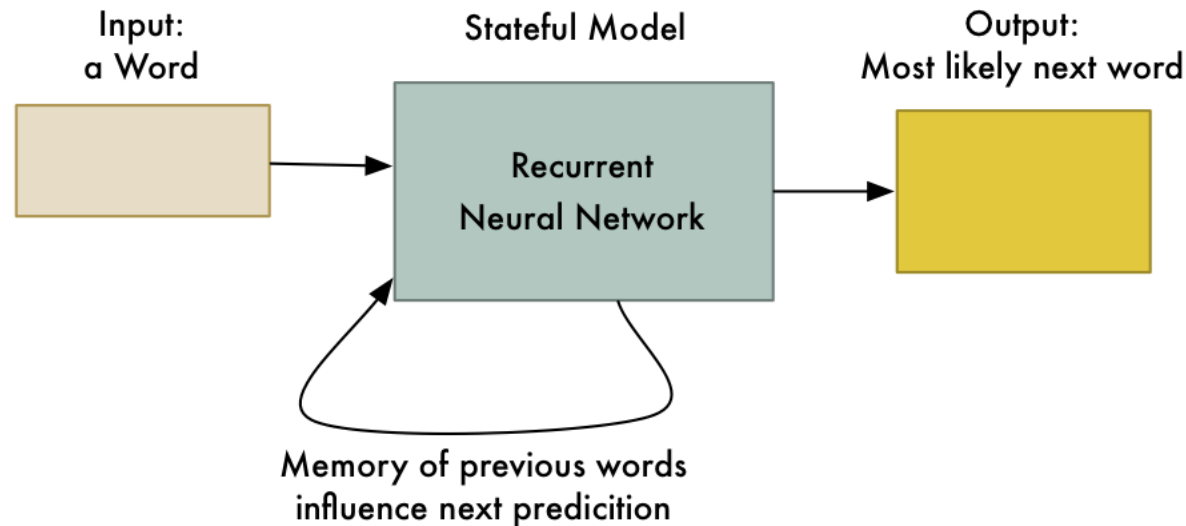


- Good for image processing: classification, object recognition, automobile lane tracking, etc.
- Classic demo: learn to recognize hand-written digits from [MNIST](#) data with 70K examples



RNN: Recurrent Neural Networks

- Good for learning over sequences of data, e.g., language understanding tasks
- LSTM (Long Short Term Memory) a popular architecture



Output so far:
Machine

gif from [Adam Geitgey](#)

Tinker With a Neural Network Right Here in Your Browser. Don't Worry, You Can't Break It. We Promise.



Epoch
000,000

Learning rate
0.03

Activation
ReLU

Regularization
None

Regularization rate
0

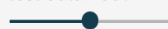
Problem type
Classification

DATA

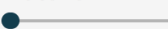
Which dataset do you want to use?



Ratio of training to test data: 50%



Noise: 0



Batch size: 10



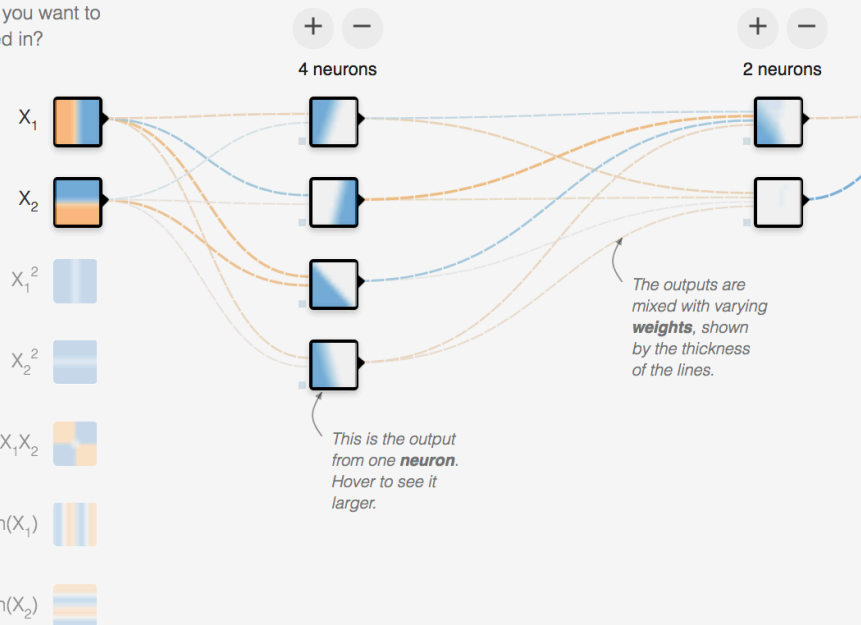
REGENERATE

FEATURES

Which properties do you want to feed in?

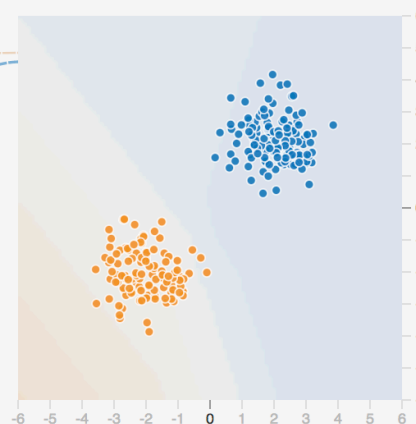
- X_1
- X_2
- X_1^2
- X_2^2
- $X_1 X_2$
- $\sin(X_1)$
- $\sin(X_2)$

2 HIDDEN LAYERS



OUTPUT

Test loss 0.435
Training loss 0.432



Colors shows data, neuron and weight values.

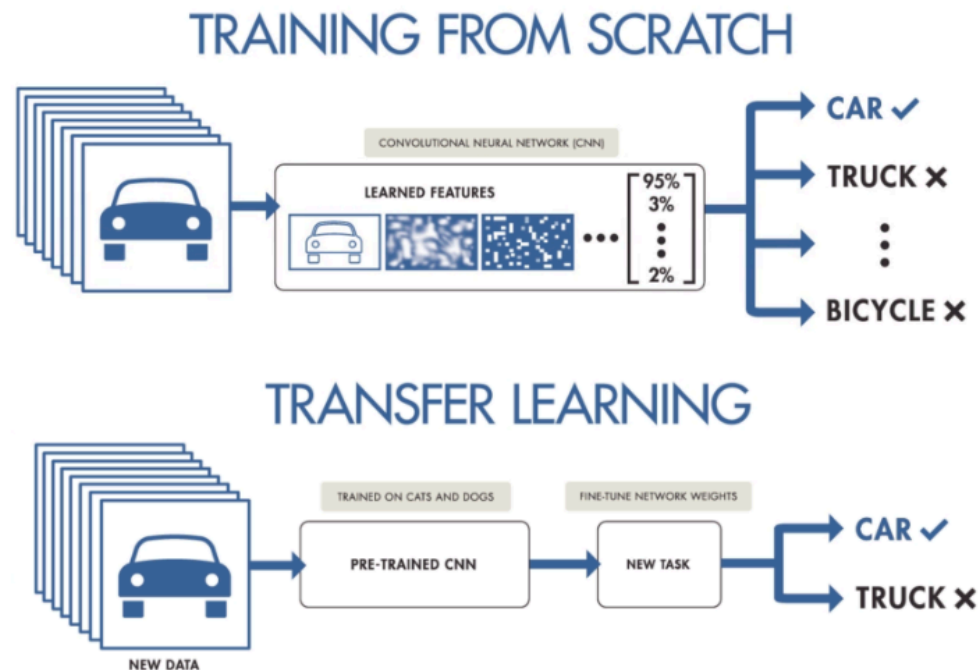
Show test data Discretize output

Deep Learning Frameworks

- Popular open source deep learning frameworks use Python at top-level; C++ in backend
 - [TensorFlow](#) (via Google)
 - [PyTorch](#) (via Facebook)
 - [MxNet](#) (Apache)
 - [Caffe](#) (Berkeley)
- [Keras](#): popular API works with the first two and provides good support at architecture level

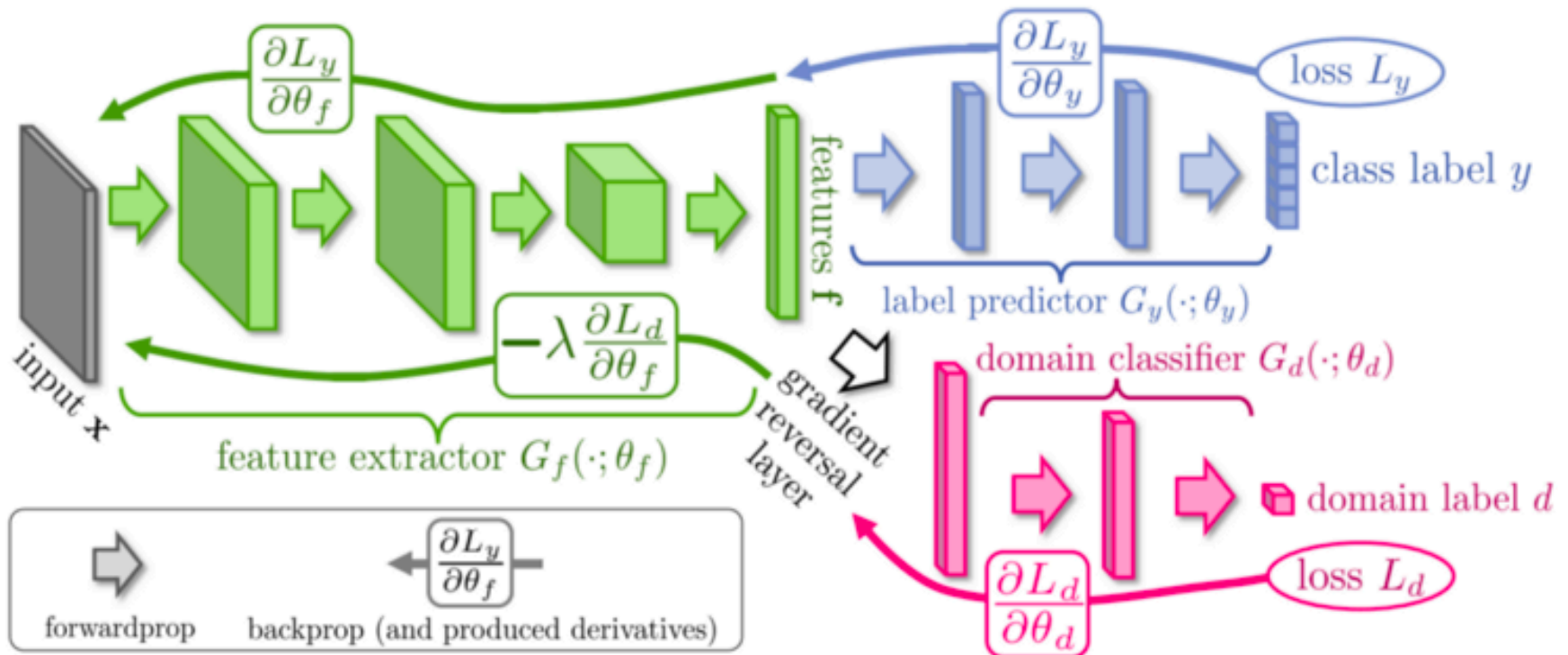
Good at Transfer Learning

- Neural networks effective for [transfer learning](#)
Using parts of a model trained on a task as an initial model to train on a different task
- Particularly effective for image recognition



Good at Transfer Learning

- For images, the initial stages of a model learn high-level visual features (lines, edges) from pixels
- Final stages predict task-specific labels



Conclusions

- Quick introduction to neural networks and deep learning
- Learn more by
 - taking UMBC's machine learning class
 - Explore Google's [Machine Learning Crash Course](#)
 - Try Miner/Kasch tutorial on [applied deep learning](#)
 - Work through examples
- and then trying your own project idea