

# Neural Networks - Deep Learning

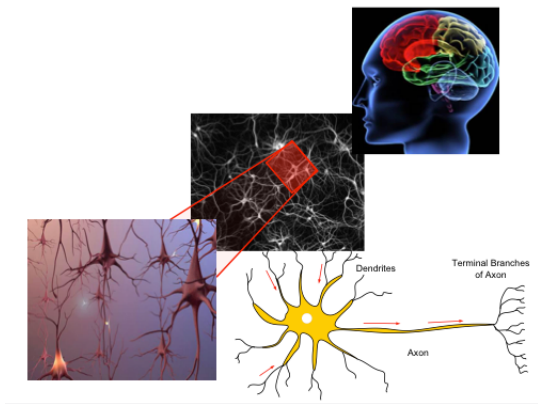
Artificial Intelligence @ Allegheny College

Janyl Jumadinova

October 13-18, 2021

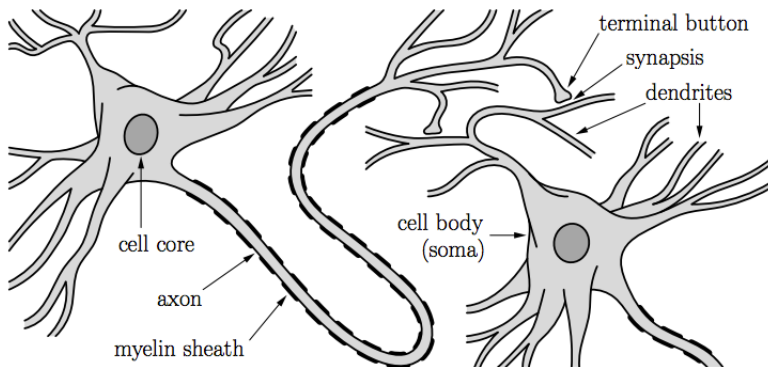
Credit: Google Workshop

# Neural Networks



# Neural Networks

## Structure of a prototypical biological neuron



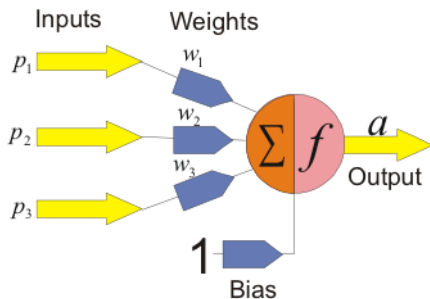
# Neural Networks

Neural computing requires a number of **neurons**, to be connected together into a **neural network**.

Neurons are arranged in layers.

Two main **hyperparameters** that control the architecture or topology of the network: 1) the number of layers, and 2) the number of nodes in each hidden layer.

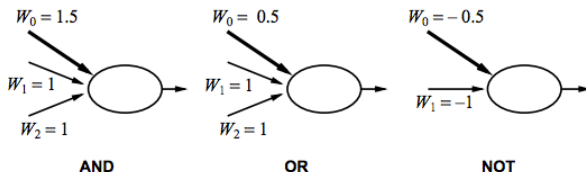
# Neural Networks



$$a = f(p_1 w_1 + p_2 w_2 + p_3 w_3 + b) = f\left(\sum p_i w_i + b\right)$$

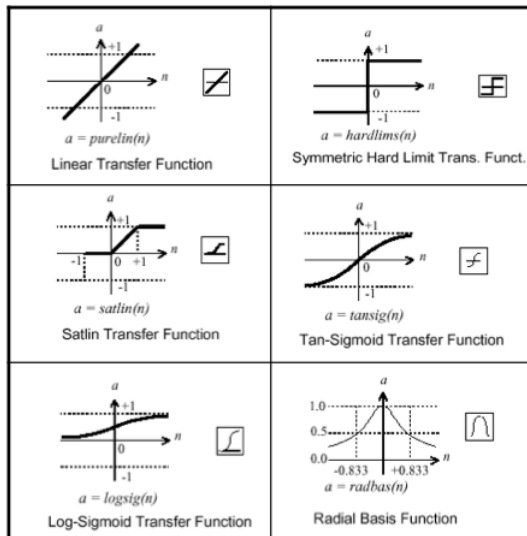
# Activation Functions

- The activation function is generally non-linear.
- Linear functions are limited because the output is simply proportional to the input.



McCulloch and Pitts: every Boolean function can be implemented

# Activation Functions



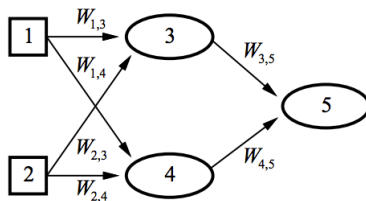
# Network structures

Two phases in each iteration:

- ① Calculating the predicted output  $\mathbf{y}$ , known as feed-forward
- ② Updating the weights and biases, known as backpropagation



# Feed-forward example



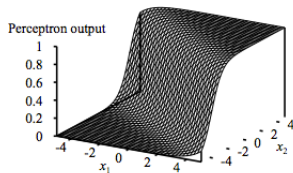
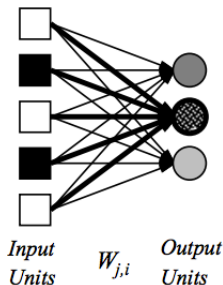
Feed-forward network = a parameterized family of nonlinear functions:

$$\begin{aligned} a_5 &= g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4) \\ &= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2)) \end{aligned}$$

Feed-forward networks:

- Single-layer perceptrons
- Multi-layer perceptrons

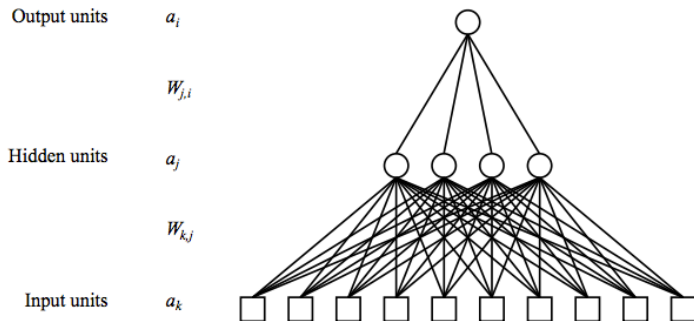
# Single-layer Perceptrons



Output units all operate separately – no shared weights.

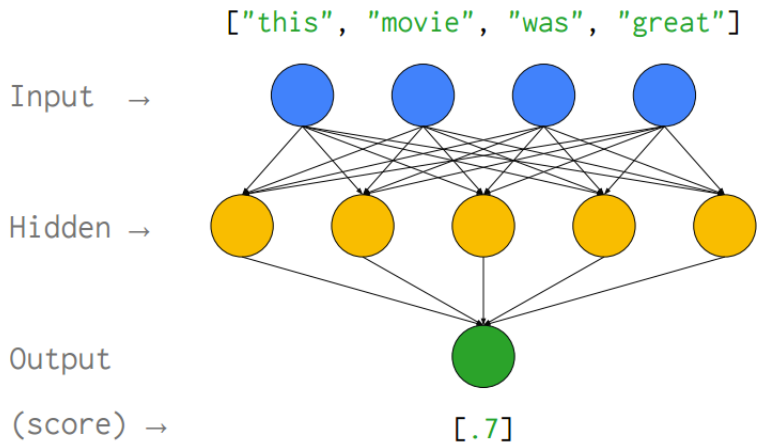
Adjusting weights moves the location, orientation, and steepness of cliff.

# Multi-layer Perceptrons

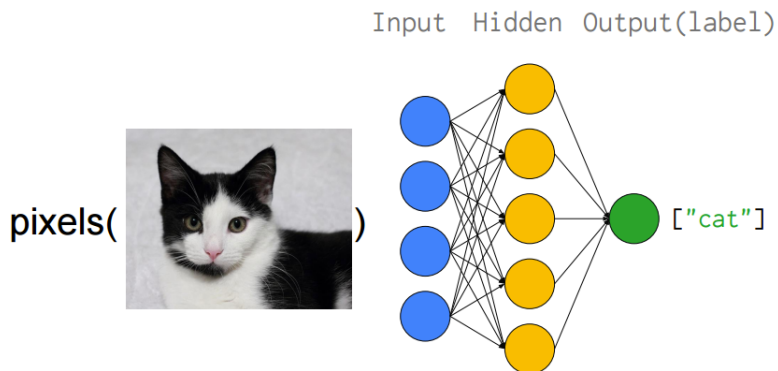


- Layers are usually fully connected.
- Numbers of hidden units typically chosen by hand.

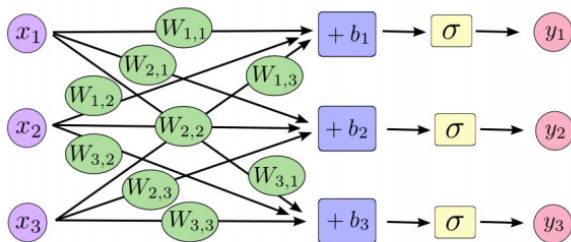
# Neural Networks



# Neural Networks



# Neural Networks: A fully connected NN layer



$$\begin{aligned} y_1 &= \sigma ( W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 ) \\ y_2 &= \sigma ( W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 ) \\ y_3 &= \sigma ( W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 ) \end{aligned}$$

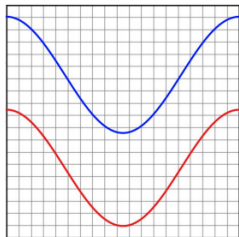
# Implementation as Matrix Multiplication

$$\begin{aligned} y_1 &= \sigma ( W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 ) \\ y_2 &= \sigma ( W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 ) \\ y_3 &= \sigma ( W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 ) \end{aligned}$$

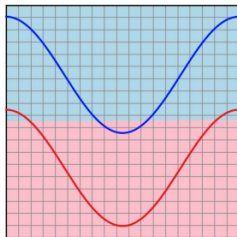
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \sigma \left( \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

# Non-Linear Data Distributions

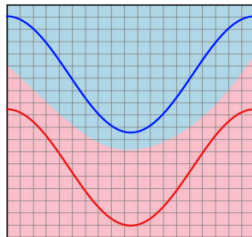
The Problem



Linear Model



Neural Network

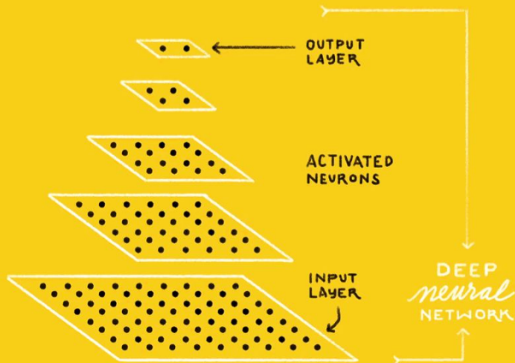




IS THIS A  
**CAT or DOG?**



**CAT DOG**



# Deep Learning

- Most current machine learning works well because of human-designed representations and input features.
- Machine learning becomes just optimizing weights to best make a final prediction.

# Deep Learning

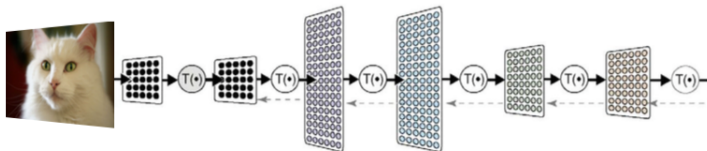
- Most current machine learning works well because of human-designed representations and input features.
- Machine learning becomes just optimizing weights to best make a final prediction.
- **Deep learning** algorithms attempt to learn multiple levels of representation of increasing complexity/abstraction.

# Deep Learning

- Each neuron implements a relatively simple mathematical function.
- $y = g(\overline{w} \cdot \overline{x} + b)$

# Deep Learning

- Each neuron implements a relatively simple mathematical function.
- $y = g(\overline{w} \cdot \overline{x} + b)$
- The composition of  $10^6 - 10^9$  such functions is powerful.



# Deep Learning

Book: <http://www.deeplearningbook.org/>

## Chapter 5

“A core idea in deep learning is that we assume that the data was generated by the composition of factors or features, potentially at multiple levels in a hierarchy.”

# Deep Learning

Results get better (to a degree) with:

- more data
- bigger models
- more computation

# Deep Learning

Results get better (to a degree) with:

- more data
- bigger models
- more computation

Better algorithms, new insights and improved methods help, too!

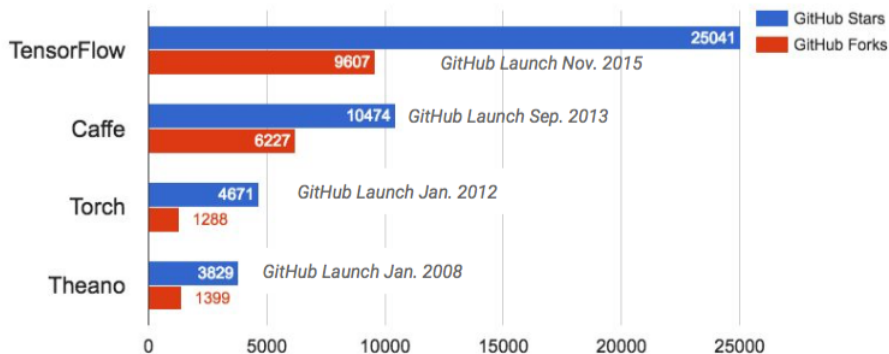


# TensorFlow



- Open source Machine Learning library
- Especially useful for **Deep Learning**
- For research **and** production
- **Apache 2.0** license
- [tensorflow.org](https://www.tensorflow.org)

# Adoption of Deep Learning Tools on GitHub



- *Epoch*: a training iteration (one pass through the dataset).
- *Batch*: Portion of the dataset (number of samples after dataset has been divided).
- *Regularization*: a set of techniques that helps learning models to converge  
(<http://www.godeep.ml/regularization-using-tensorflow/>).

# TensorFlow

A multidimensional array.



- Operates over **tensors**: n-dimensional arrays      A graph of operations.

# TensorFlow

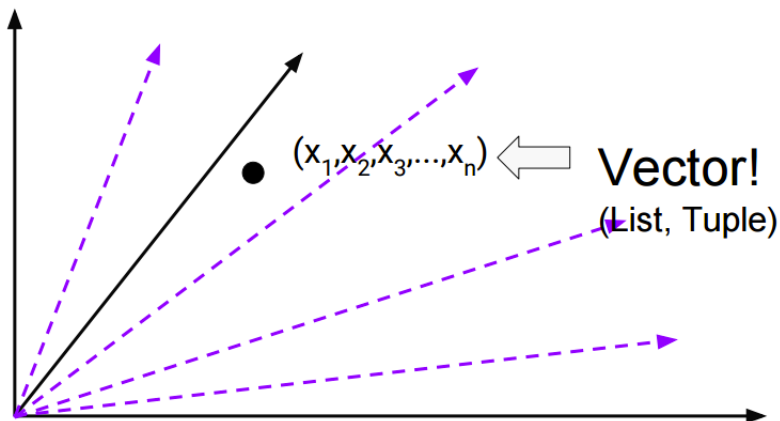
A multidimensional array.



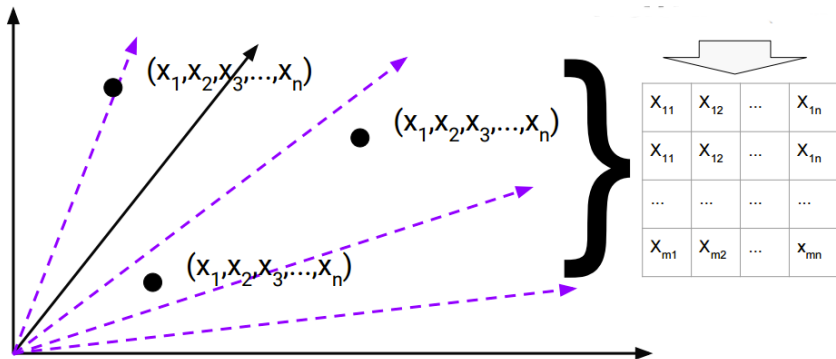
- Operates over **tensors**: n-dimensional arrays
- Using a **flow graph**: data flow computation framework

- 5.7 ← Scalar
- Number, Float, etc.

# TensorFlow



# TensorFlow



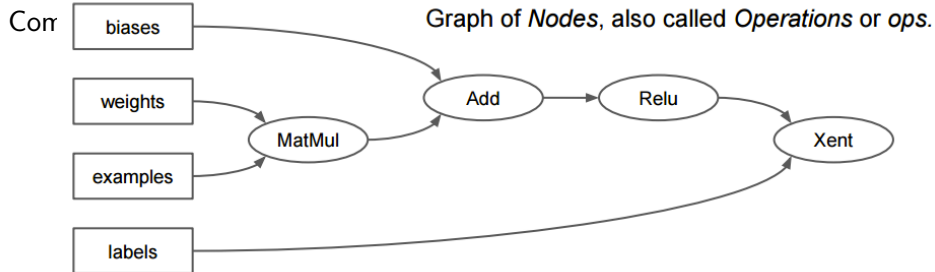


- Tensors have a **Shape** that is described with a vector

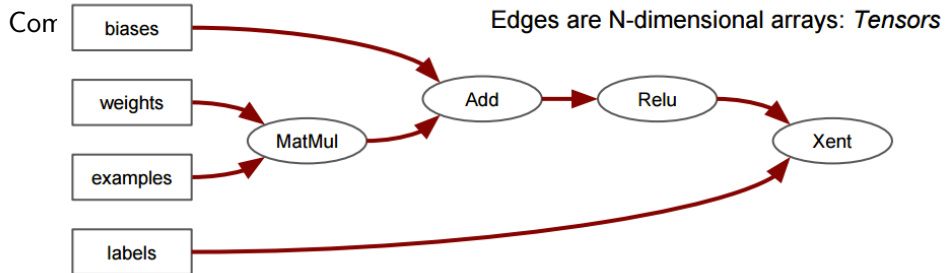
# TensorFlow

- Tensors have a **Shape** that is described with a vector
- [1000, 256, 256, 3]
- 10000 Images
- Each Image has 256 Rows
- Each Row has 256 Pixels
- Each Pixel has 3 values (RGB)

# TensorFlow



# TensorFlow



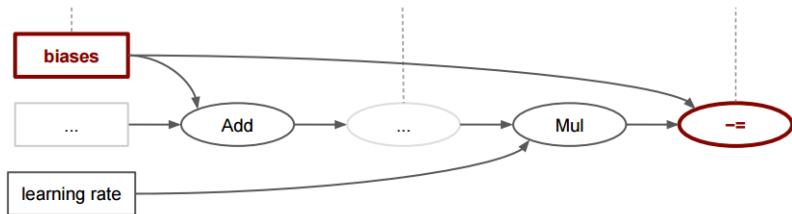
# TensorFlow

Computation is a dataflow graph with state

**'Biases' is a variable**

**Some ops compute gradients**

**--= updates biases**



# Core TensorFlow data structures and concepts

- **Graph:** A TensorFlow computation, represented as a dataflow graph:
  - collection of ops that may be executed together as a group.

# Core TensorFlow data structures and concepts

- **Graph:** A TensorFlow computation, represented as a dataflow graph:
  - collection of ops that may be executed together as a group.
- **Operation:** a graph node that performs computation on tensors

# Core TensorFlow data structures and concepts

- **Graph:** A TensorFlow computation, represented as a dataflow graph:
  - collection of ops that may be executed together as a group.
- **Operation:** a graph node that performs computation on tensors
- **Tensor:** a handle to one of the outputs of an Operation:
  - provides a means of computing the value in a TensorFlow Session.



- **Constants**

# TensorFlow

- **Constants**
- **Placeholders**: must be fed with data on execution.

- **Constants**
- **Placeholders**: must be fed with data on execution.
- **Variables**: a modifiable tensor that lives in TensorFlow's graph of interacting operations.

- **Constants**
- **Placeholders**: must be fed with data on execution.
- **Variables**: a modifiable tensor that lives in TensorFlow's graph of interacting operations.
- **Session**: encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

# TensorFlow

<i>Category</i>	<i>Examples</i>
Element-wise math ops	<b>Add</b> , Sub, <b>Mul</b> , Div, Exp, Log, Greater, Less...
Matrix ops	<b>Concat</b> , Slice, <b>Split</b> , Constant, Rank, <b>Shape</b> ...
Matrix ops	<b>MatMul</b> , MatrixInverse, MatrixDeterminant...
Stateful ops	<b>Variable</b> , Assign, AssignAdd...
NN building blocks	<b>SoftMax</b> , Sigmoid, <b>ReLU</b> , <b>Convolution2D</b> ...
Checkpointing ops	Save, Restore
Queue & synch ops	Enqueue, Dequeue, MutexAcquire...
Control flow ops	Merge, Switch, Enter, Leave...

`https://playground.tensorflow.org`