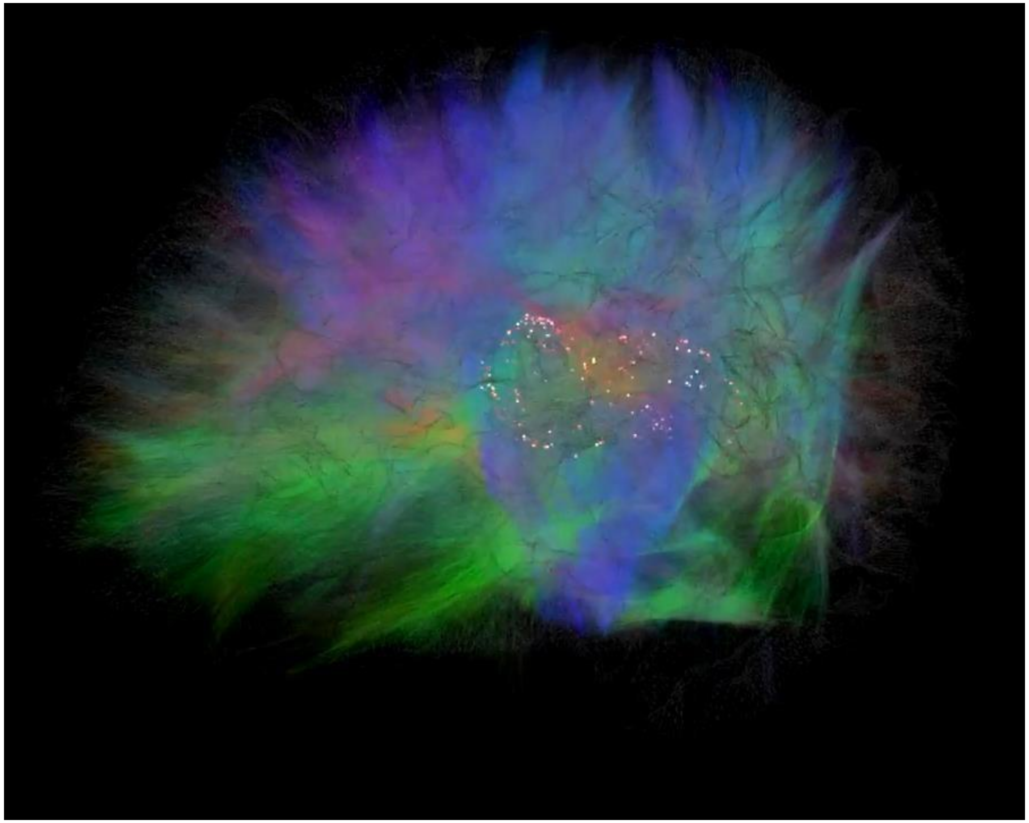
Deep Learning 101

Biological and Artificial Neural Networks



Human Brain

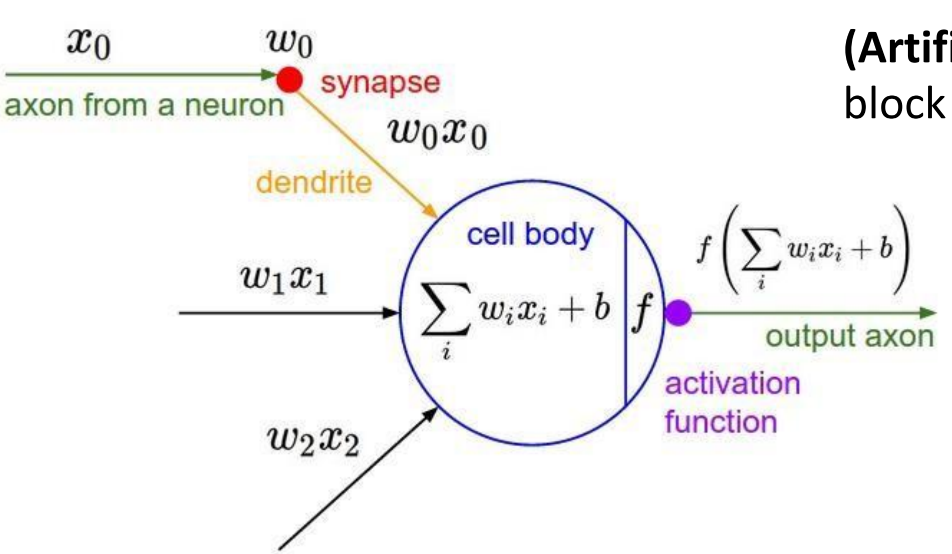
- Thalamocortical system:
  3 million neurons
  476 million synapses
- Full brain:
  100 billion neurons
  1,000 trillion synapses

Artificial Neural Network

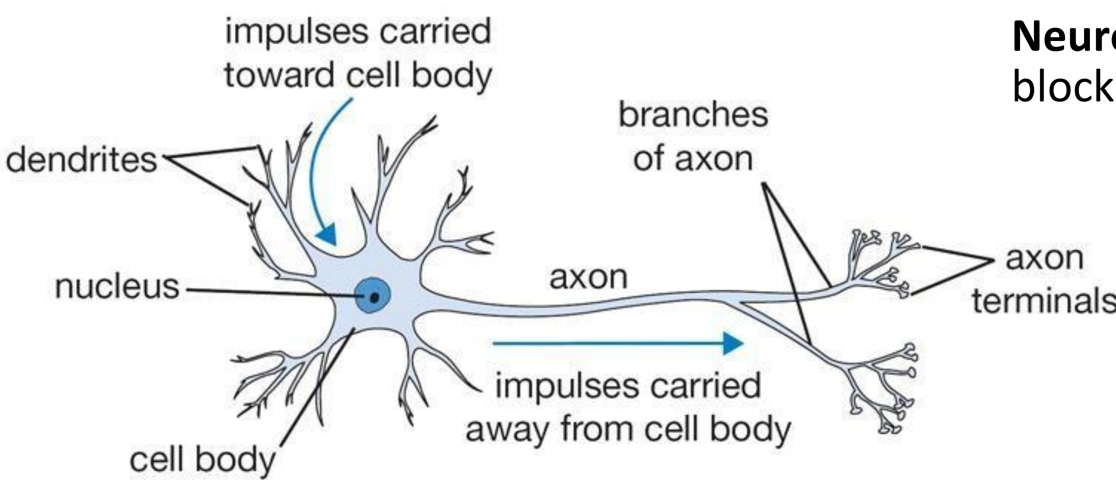
- ResNet-152:
  60 million synapses

Human brains have ~10,000,000 times synapses
than artificial neural networks.

Neuron: Biological Inspiration for Computation

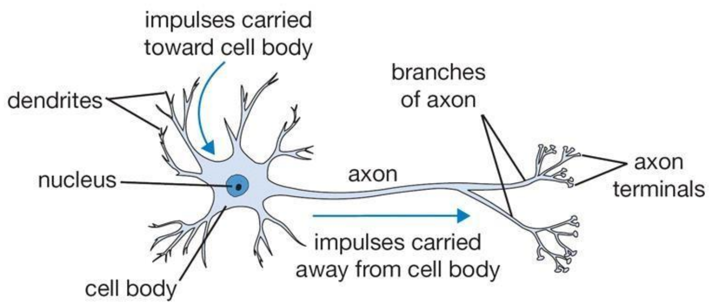


(Artificial) Neuron: computational building block for the "neural network"

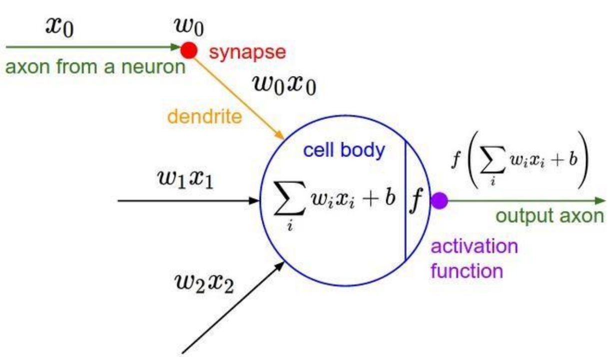


Neuron: computational building block for the brain

Neuron: Biological Inspiration for Computation



- **Neuron:** computational building block for the brain



- **(Artificial) Neuron:** computational building block for the "neural network"

Key Difference:

- **Parameters:** Human brains have ~10,000,000 times synapses than artificial neural networks.
- **Topology:** Human brains have no "layers". **Async:** The human brain works asynchronously, ANNs work synchronously.
- **Learning algorithm**: ANNs use gradient descent for learning. We don't know what human brains use
- **Power consumption:** Biological neural networks use very little power compared to artificial networks
- **Stages:** Biological networks usually never stop learning. ANNs first train then test.

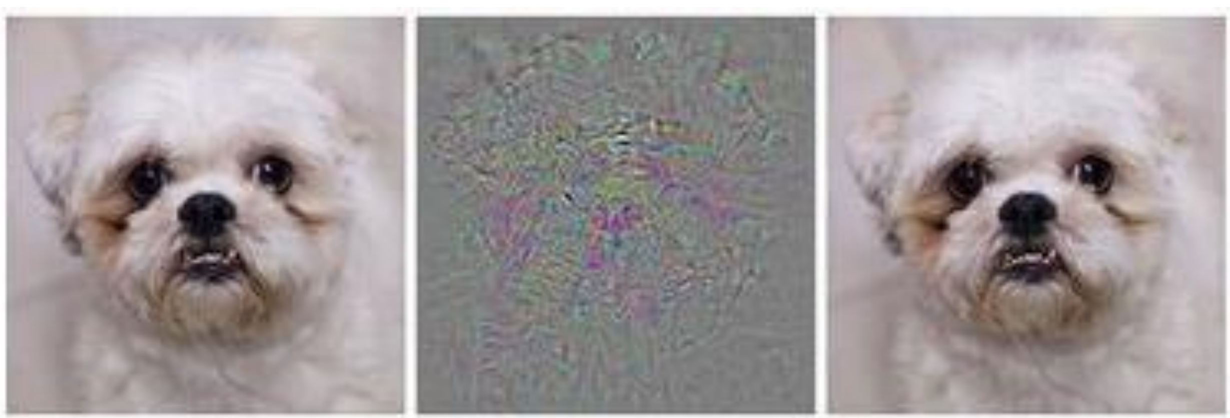
Deep Learning:

Our intuition about what's "hard" is flawed (in complicated ways)

Visual perception: 540,000,000 years of data

Bipedal movement: 230,000,000 years of data

Abstract thought: 100,000 years of data



Prediction: **Dog** + Distortion Prediction: **Ostrich**

"Encoded in the large, highly evolve sensory and motor portions of the human brain is a **billion years of experience** about the nature of the world and how to survive in it.... Abstract thought, though, is a new trick, perhaps less than **100 thousand years** old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it."
- *Hans Moravec, Mind Children (1988)*

History of Deep Learning Ideas and Milestones*

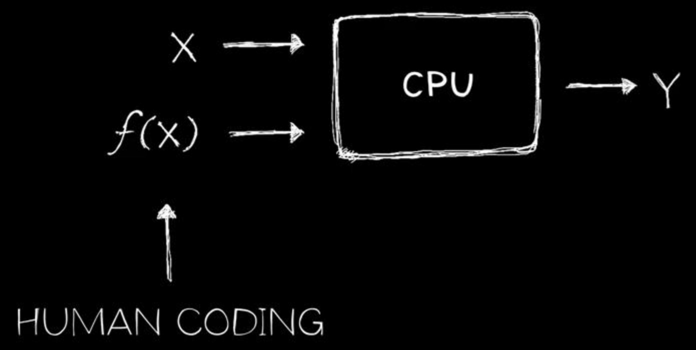
Perspective:

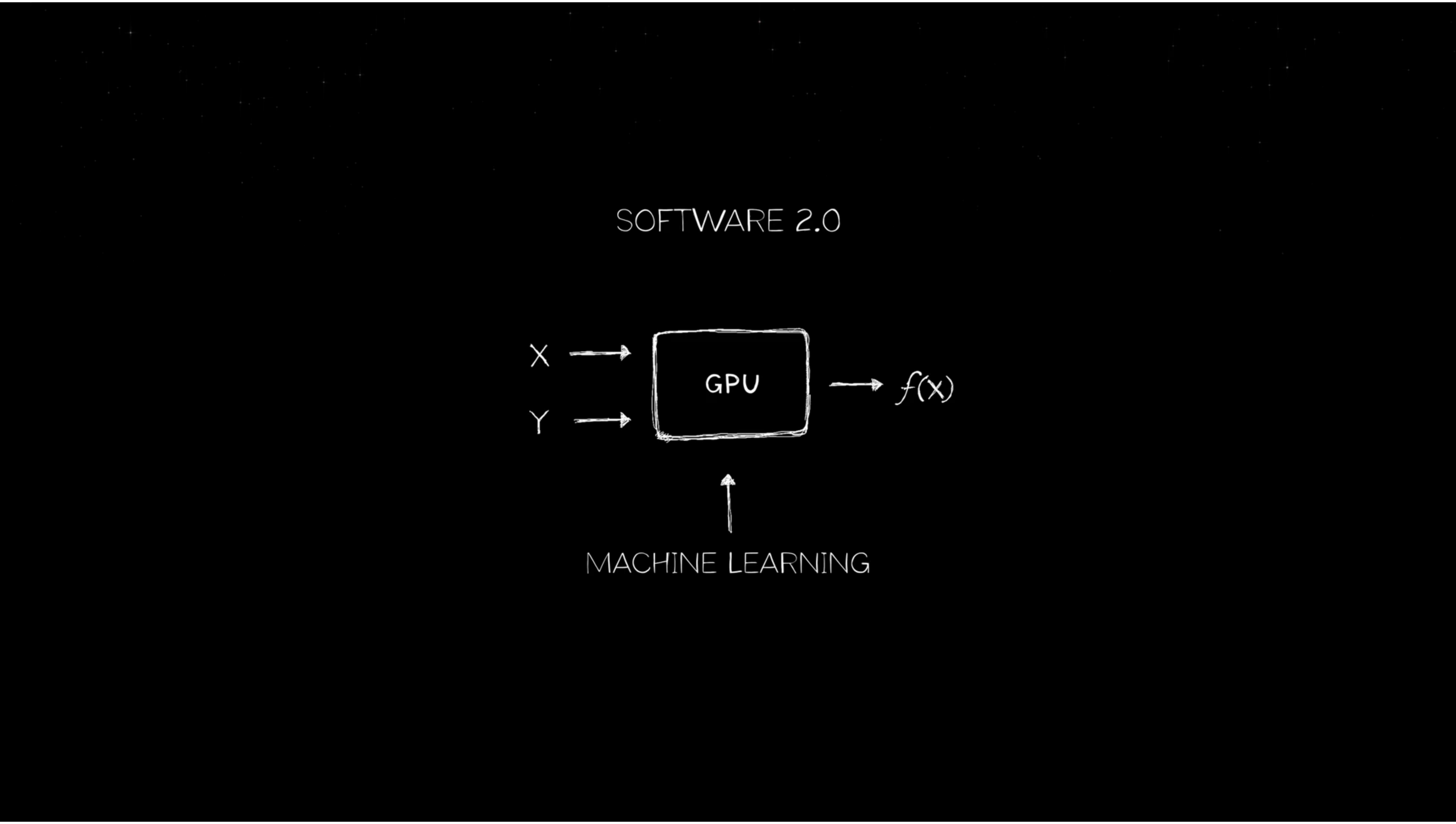
- Universe created
  13.8 billion years ago
- Earth created
  4.54 billion years ago
- Modern humans
  300,000 years ago
- Civilization
  12,000 years ago
- Written record
  5,000 years ago

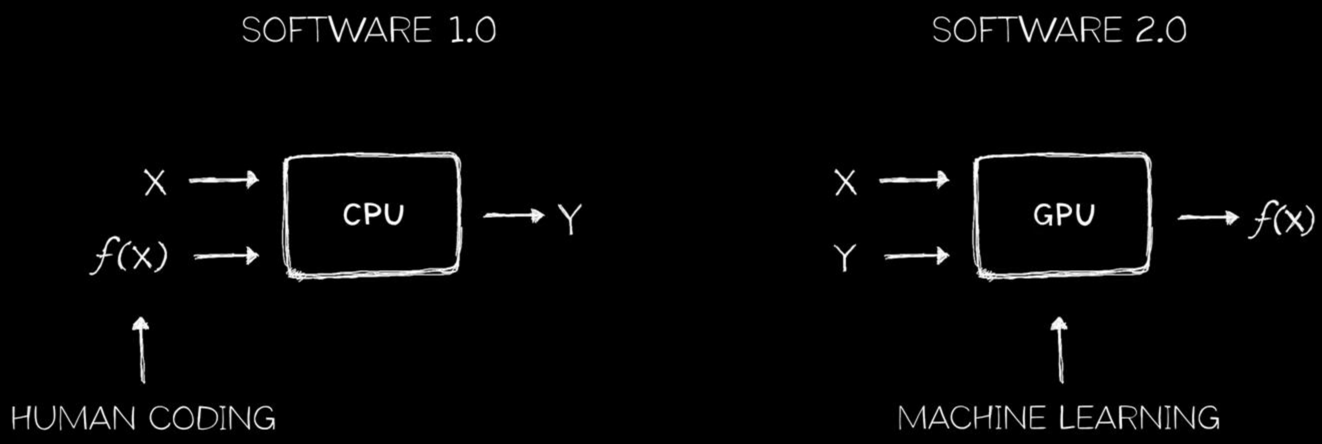
- 1943: Neural networks
- 1957: Perceptron
- 1974-86: Backpropagation, RBM, RNN
- 1989-98: CNN, MNIST, LSTM, Bidirectional RNN
- 2006: “Deep Learning”, DBN
- 2009: ImageNet
- 2012: AlexNet, Dropout
- 2014: GANs
- 2014: DeepFace
- 2016: AlphaGo
- 2017: AlphaZero, Capsule Networks
- 2018: BERT

* Dates are for perspective and not as definitive historical record of invention or credit

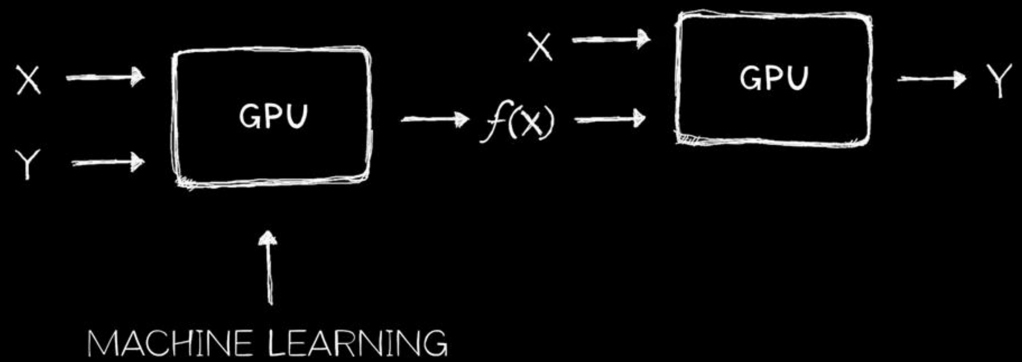
SOFTWARE 1.0




SOFTWARE 2.0
X
Y
GPU
f(X)
MACHINE LEARNING

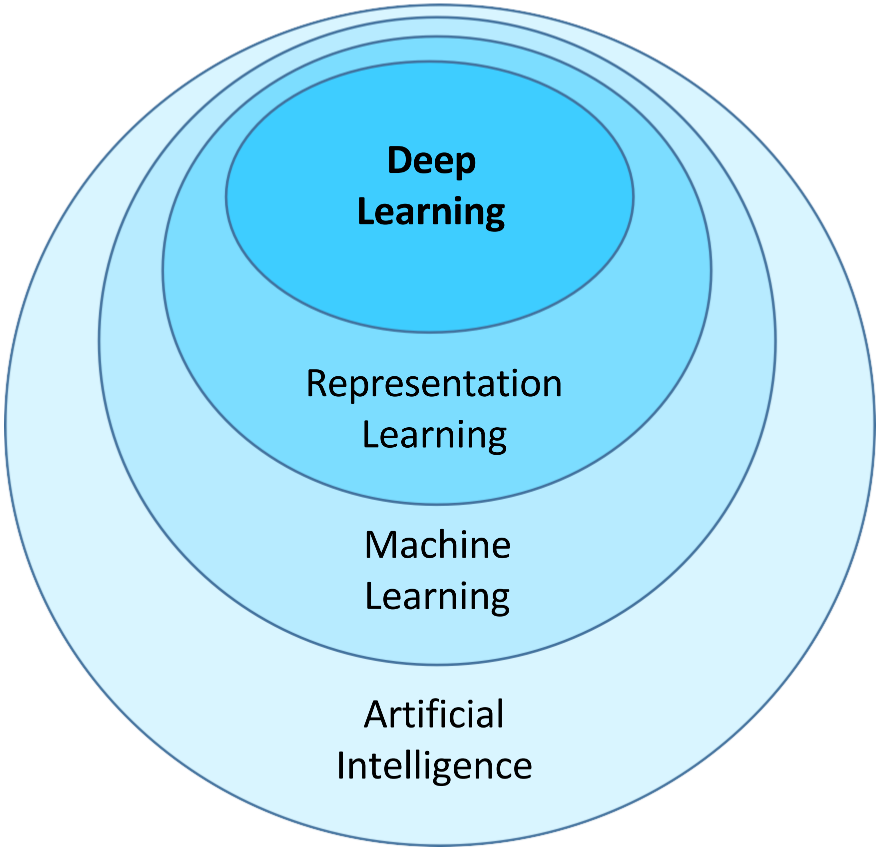

SOFTWARE 1.0
X
f(X)
CPU
Y
HUMAN CODING
SOFTWARE 2.0
X
Y
GPU
f(X)
MACHINE LEARNING

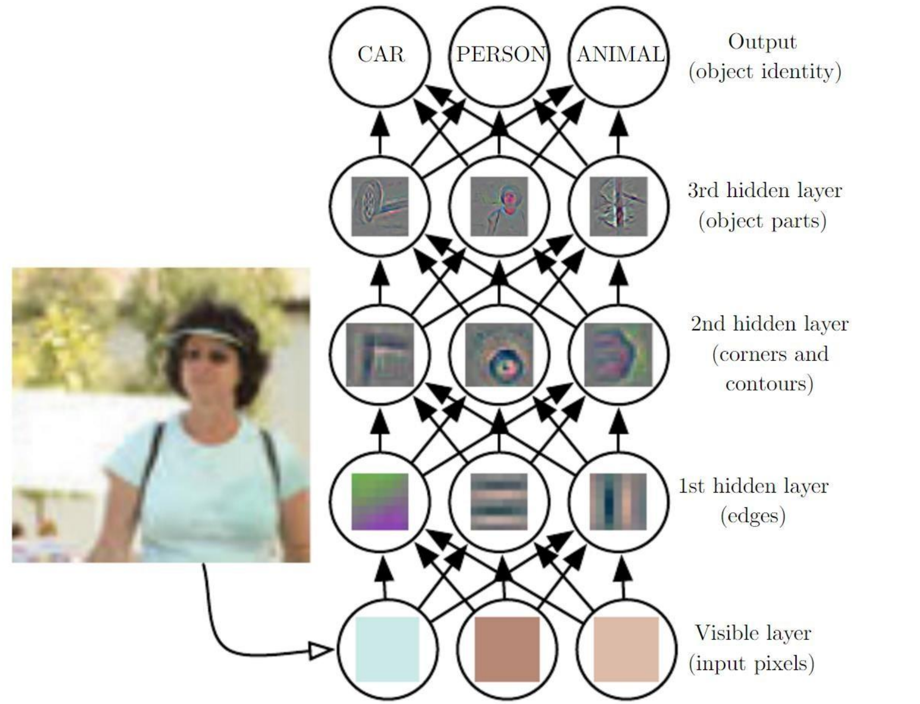
SOFTWARE 2.0



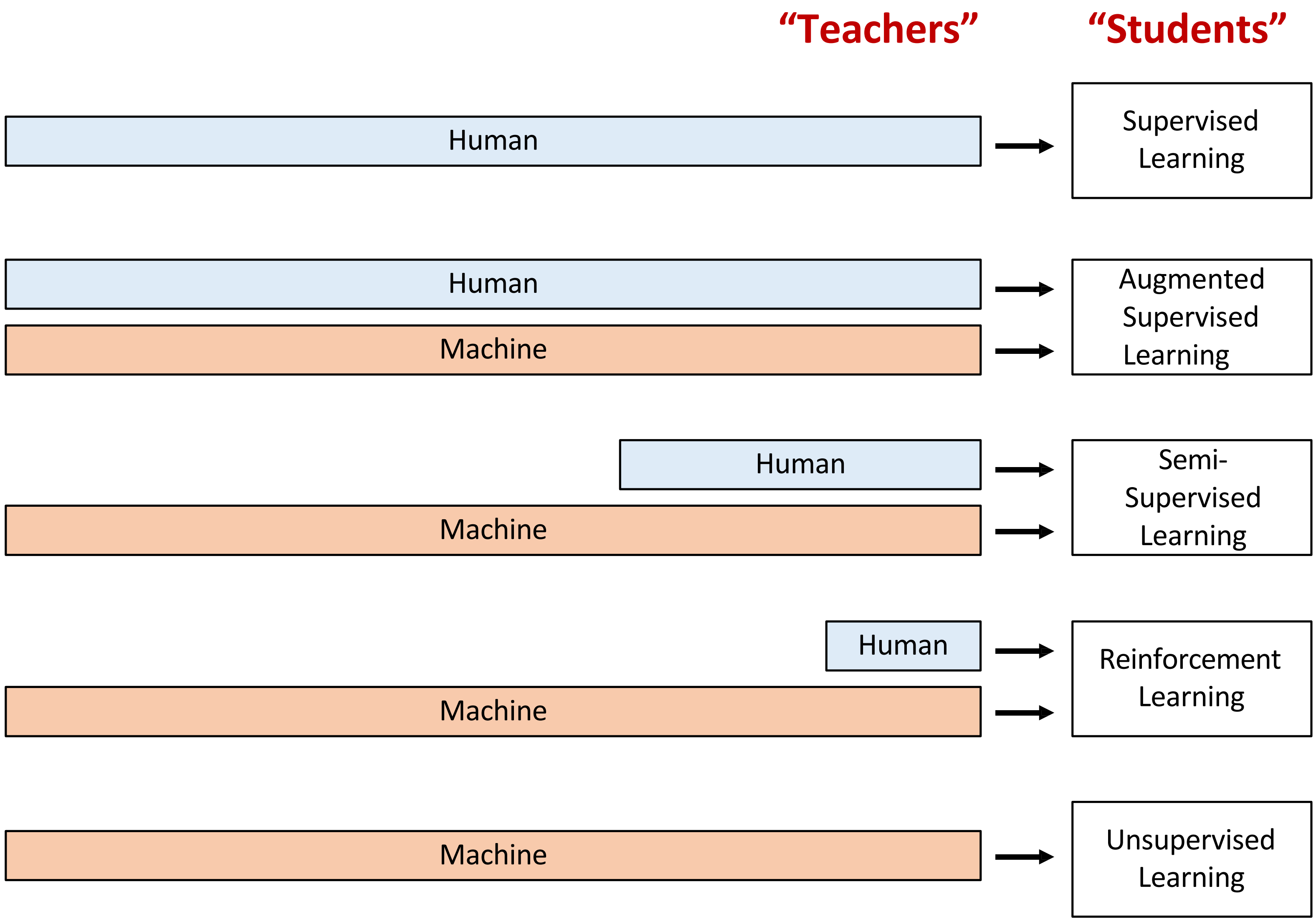
Deep Learning is **Representation Learning**

(aka Feature Learning)





Deep Learning from Human and Machine



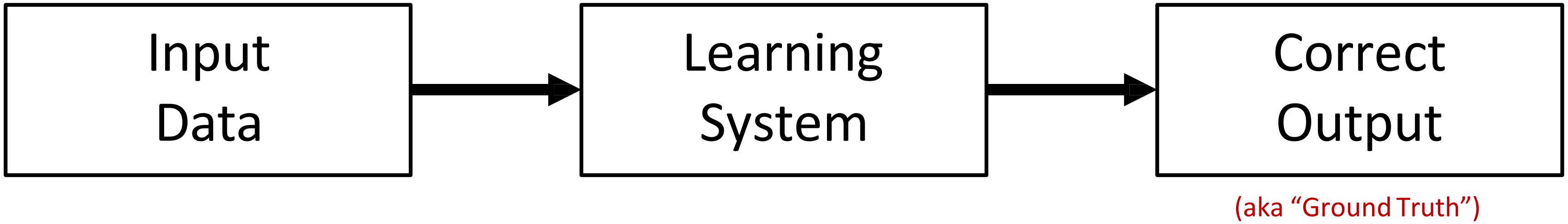
The Challenge of Deep Learning:

Efficient Teaching + Efficient Learning

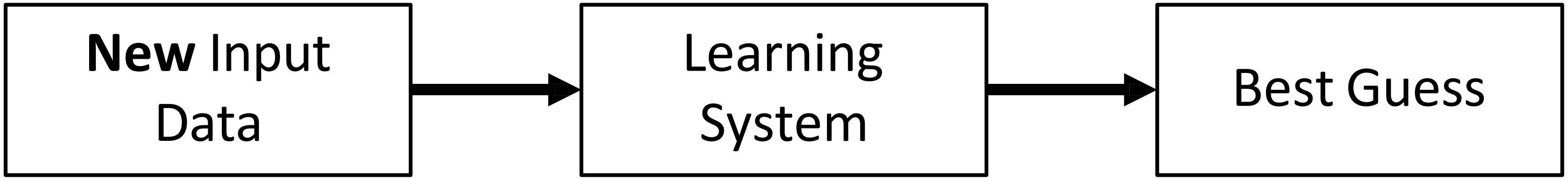
- Humans can learn from very few examples
- Machines (in most cases) need thousands/millions of examples

Deep Learning: Training and Testing

Training Stage:

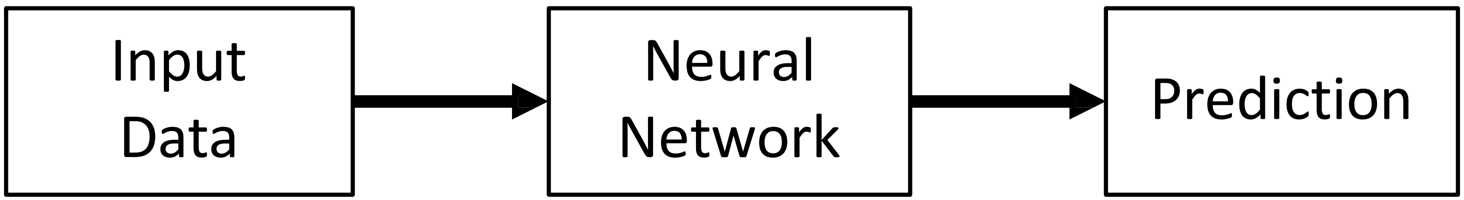


Testing Stage:

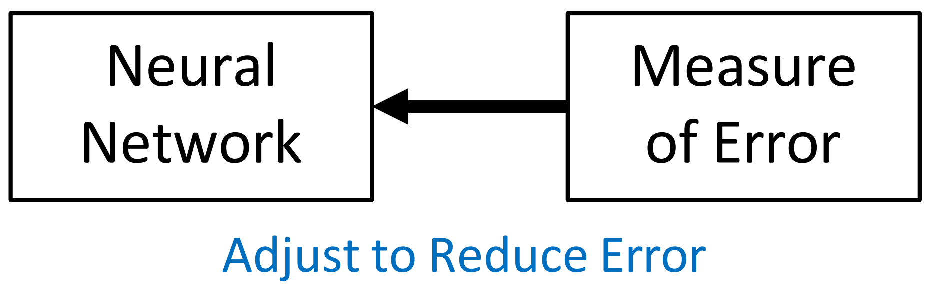


How Neural Networks Learn: Backpropagation

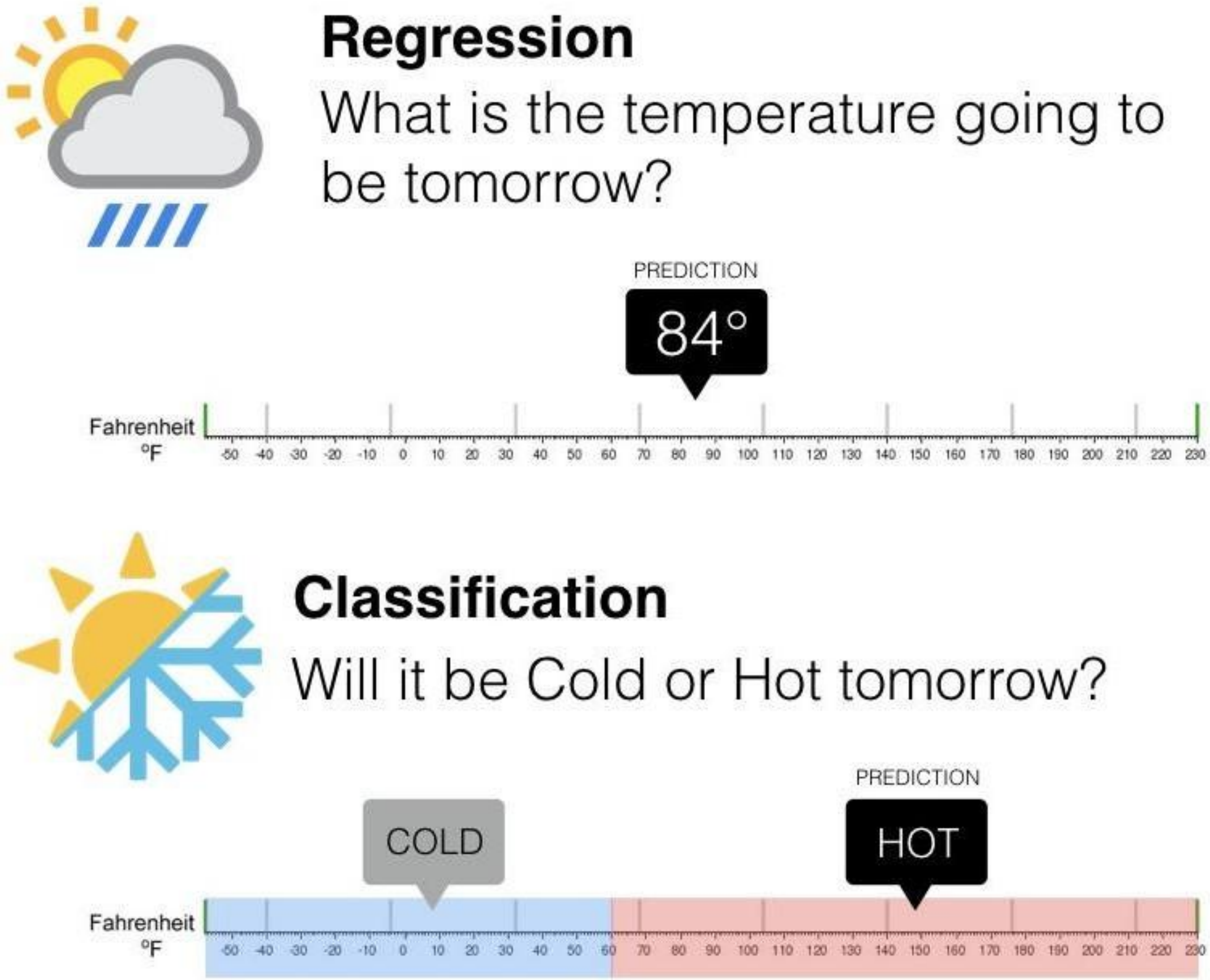
Forward Pass:



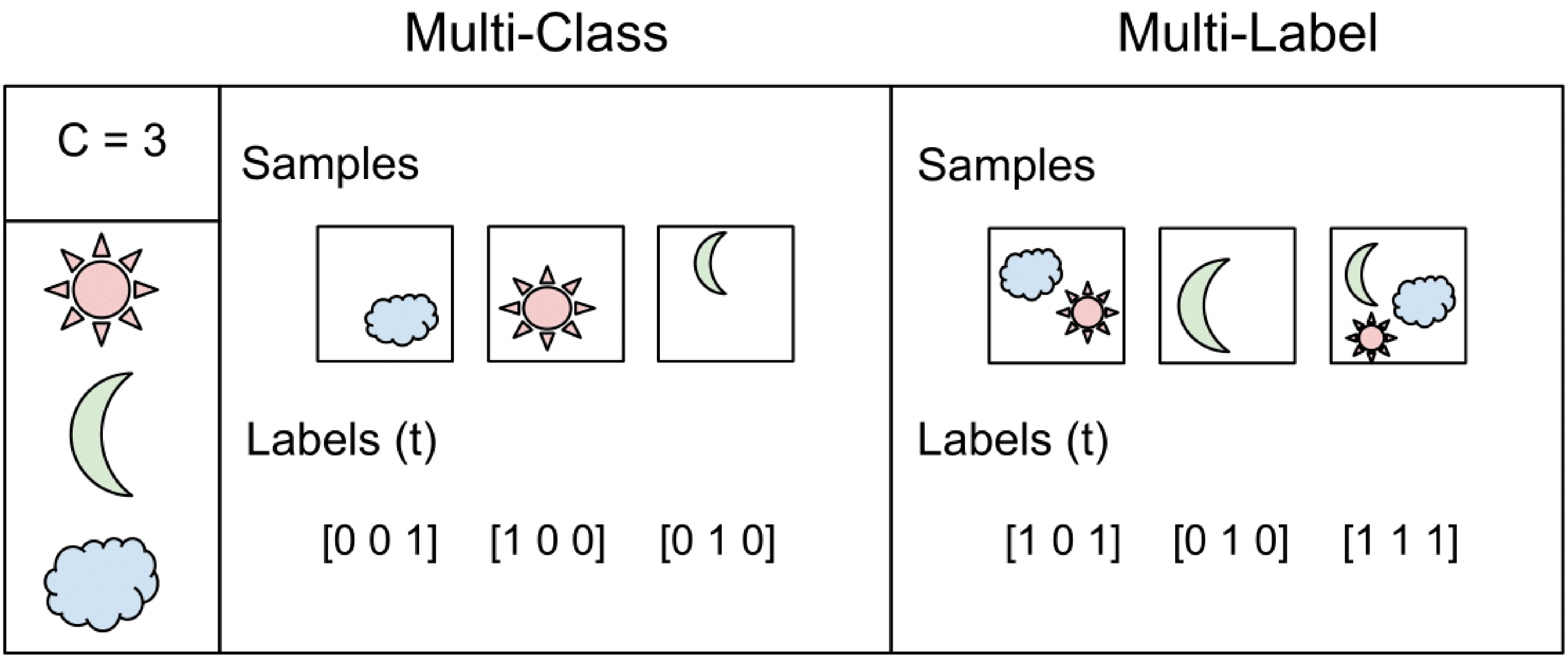
Backward Pass (aka Backpropagation):



Regression vs Classification



Multi-Class vs Multi-Label

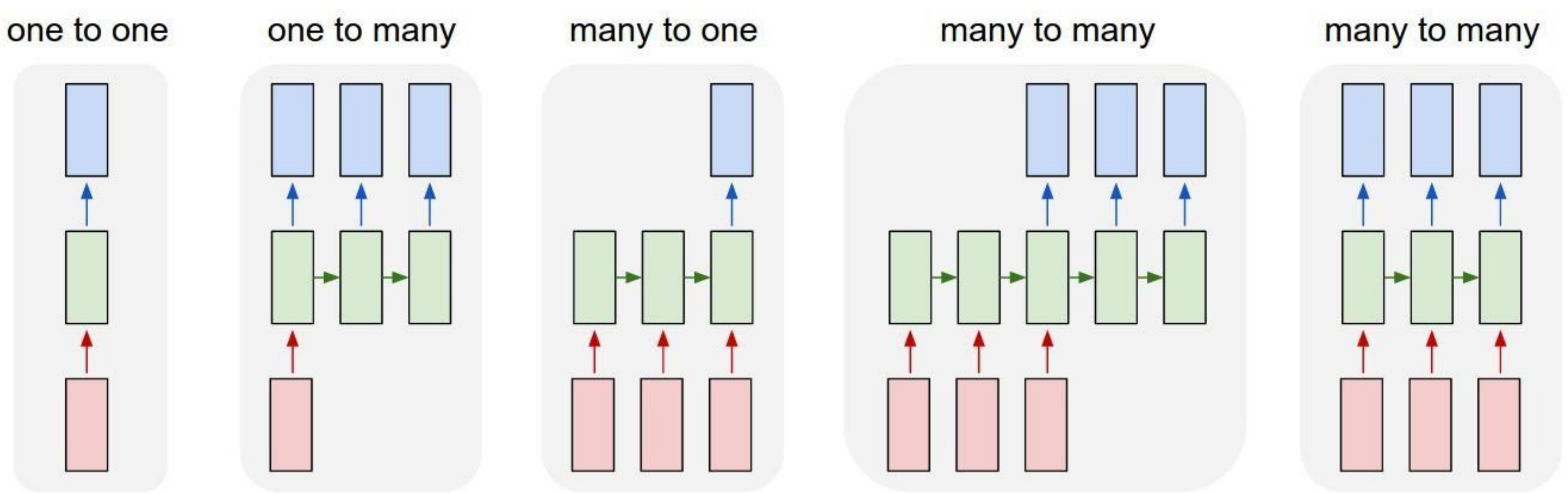


What can we do with Deep Learning?

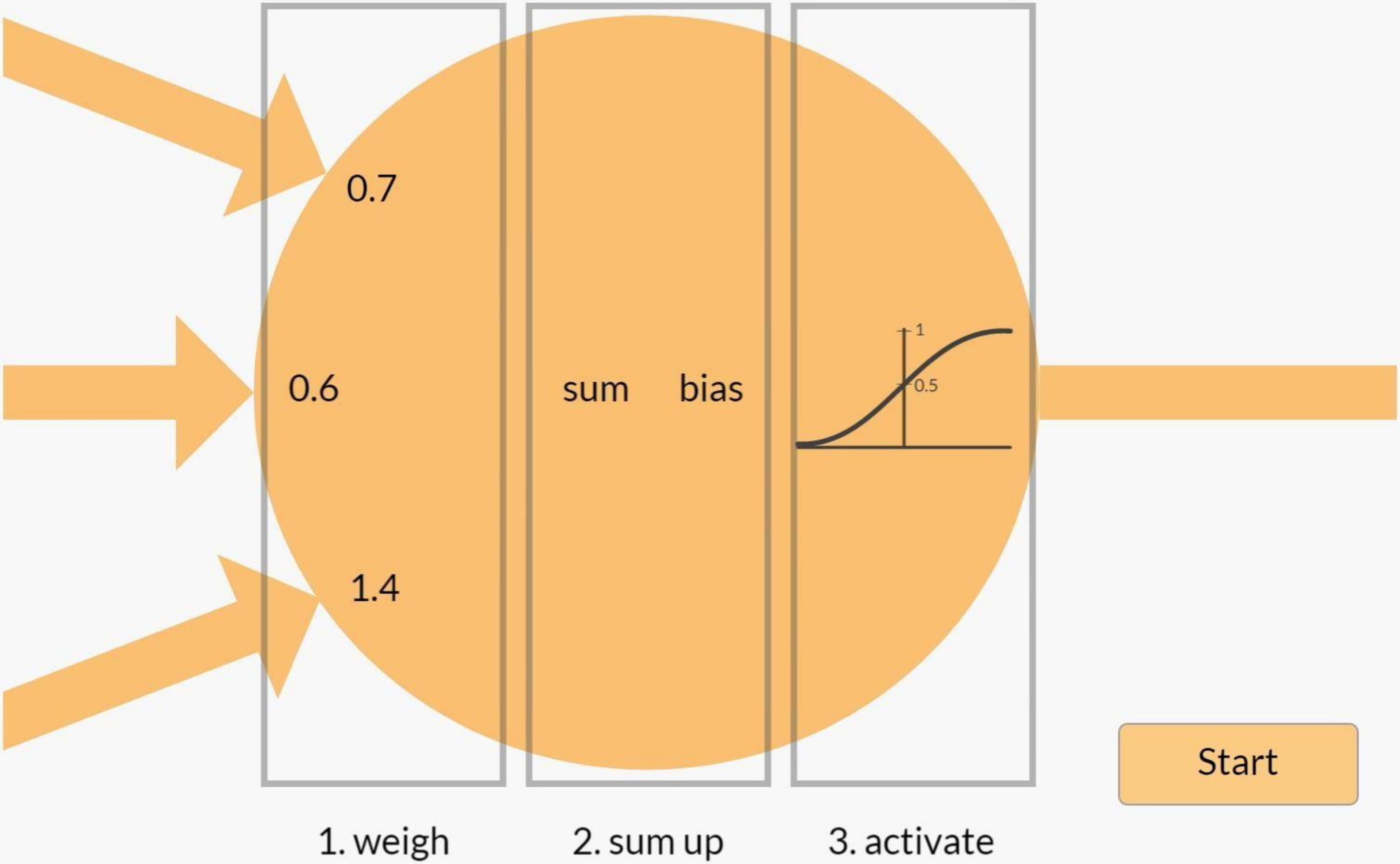
Input Data → Learning System → Correct Output

- Number
- Vector of numbers
- Sequence of numbers
- Sequence of vectors of numbers

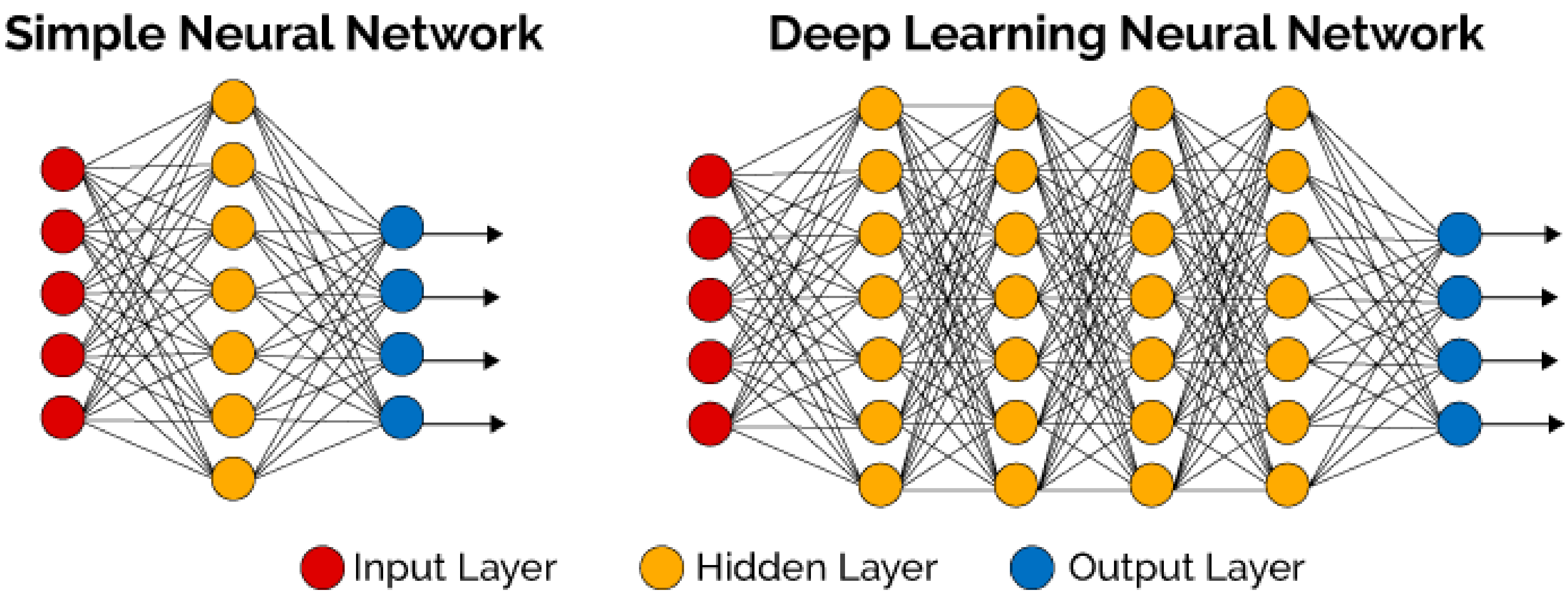
- Number
- Vector of numbers
- Sequence of numbers
- Sequence of vectors of numbers



Neuron: Forward Pass



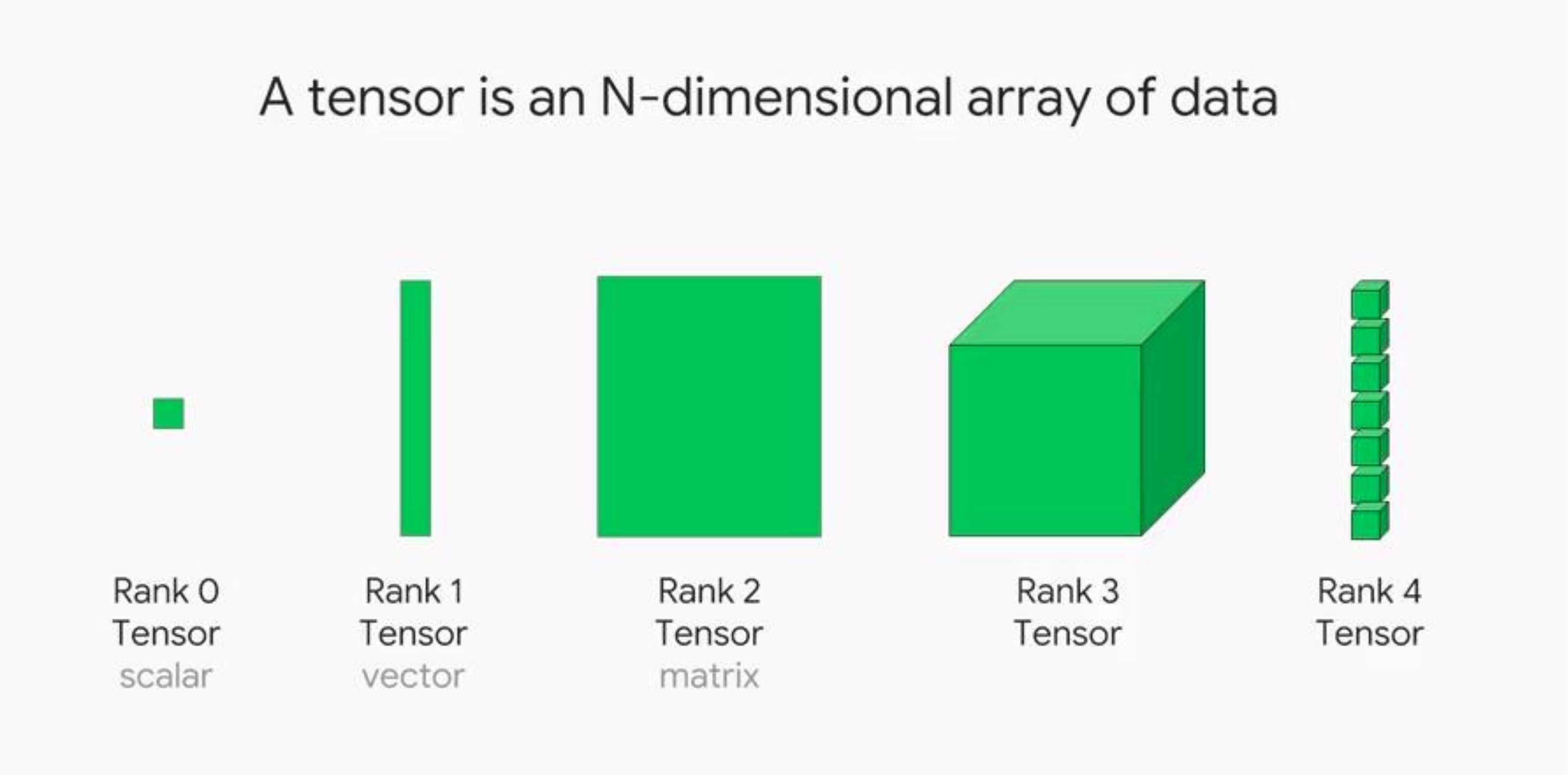
Combing Neurons in Hidden Layers:
The “Emergent” Power to Approximate



Universality: For any arbitrary function f(x), there exists a neural network that closely approximate it for any input x

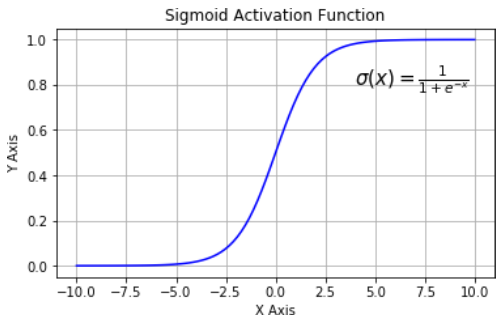
Tensorflow:
Bringing artificial neurons to life

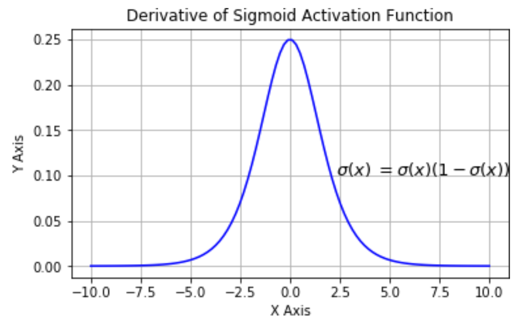
Tensor: Arrays that can be of any dimension (rank) and shape
Tensorflow: Framework for manipulating tensors



Key Concepts:

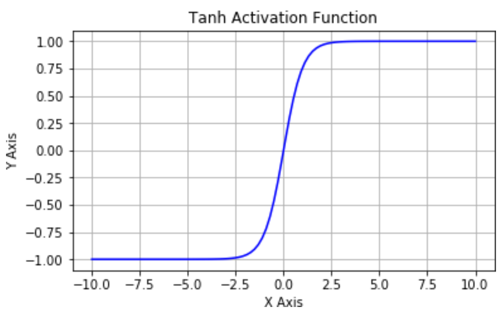
Activation Functions

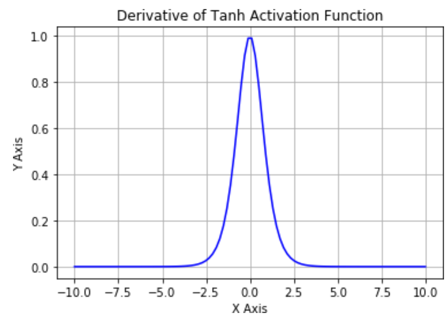




Sigmoid

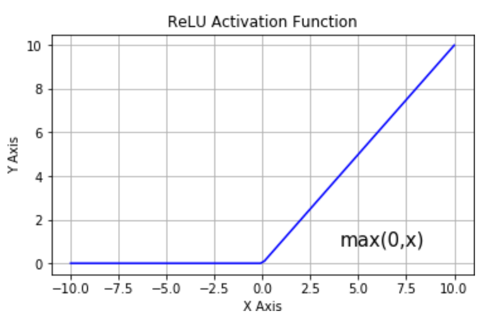
- Vanishing gradients
- Not zero centered

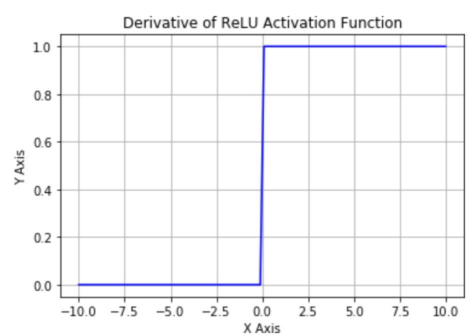




Tanh

- Vanishing gradients

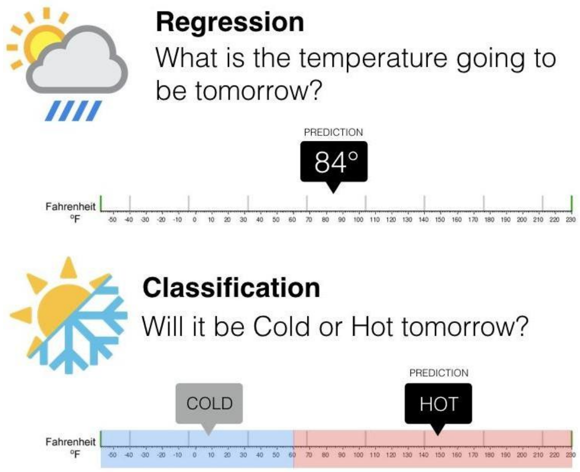




ReLU

- Not zero centered

Loss Functions



- Loss function quantifies gap between prediction and ground truth
- For regression:
  - Mean Squared Error (MSE)
- For classification:
  - Cross Entropy Loss

Mean Squared Error

Prediction

$$MSE = \frac{1}{N}\sum(t_i - s_i)^2$$

Ground Truth

Cross Entropy Loss

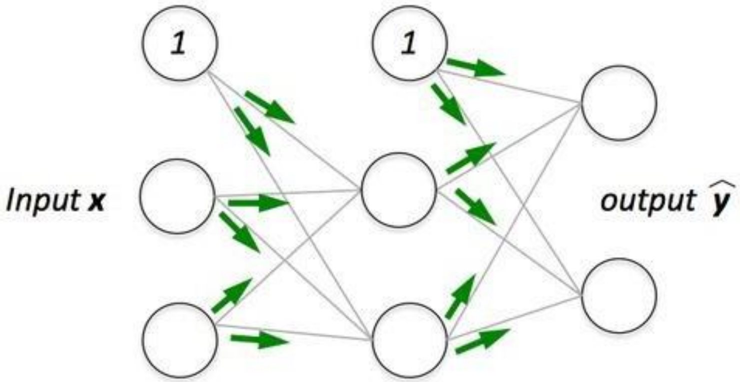
Classes

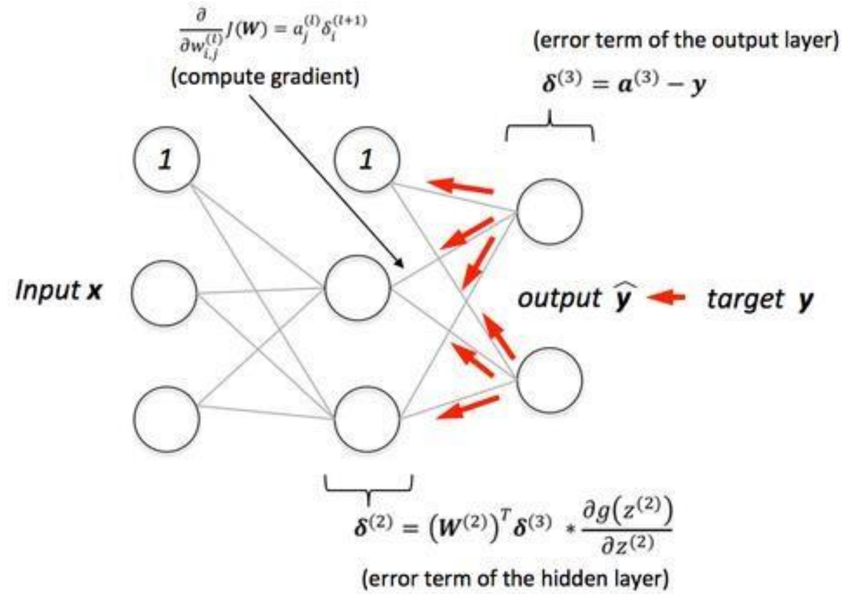
Prediction

$$CE = -\sum_{i}^{C} t_i log(s_i)$$

Ground Truth {0,1}

Backpropagation





Task: Update the **weights** and **biases** to decrease **loss function**

Subtasks:

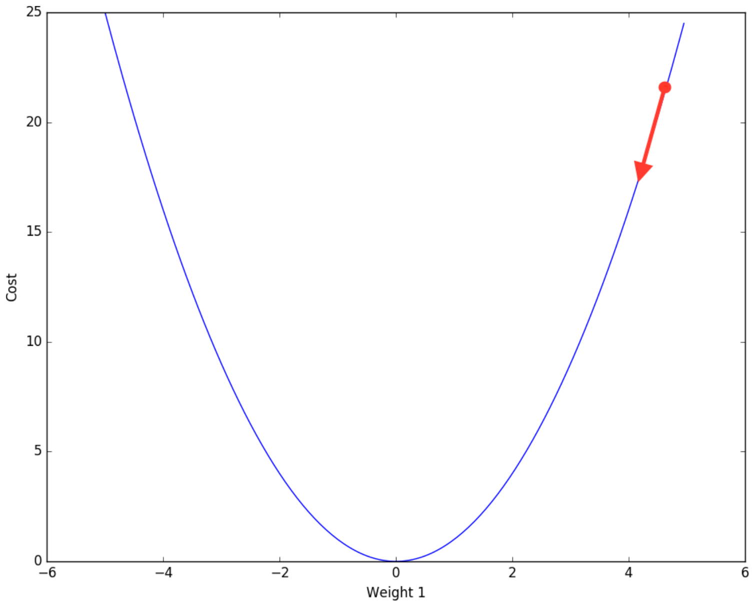
1. Forward pass to compute network output and "error"
2. Backward pass to compute gradients
3. A fraction of the weight's gradient is subtracted from the weight.

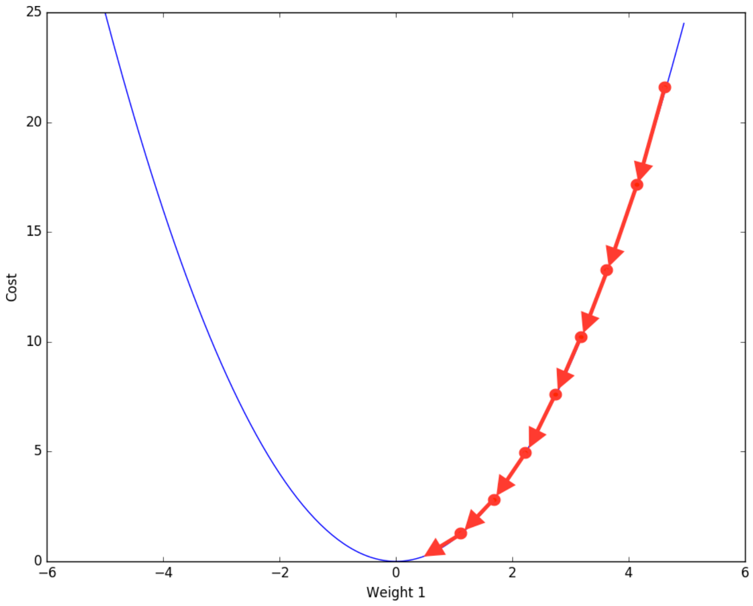
Learning Rate

Numerical Method: Automatic Differentiation

Learning is an Optimization Problem

Task: Update the **weights** and **biases** to decrease **loss function**

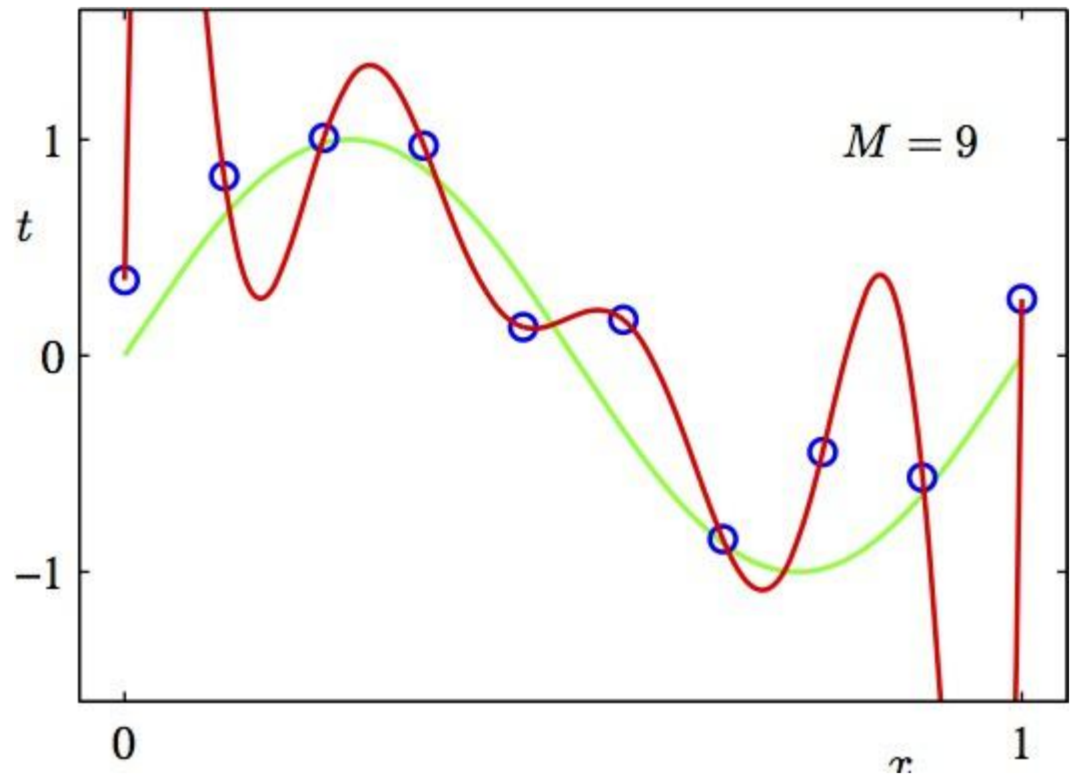




SGD: Stochastic Gradient Descent

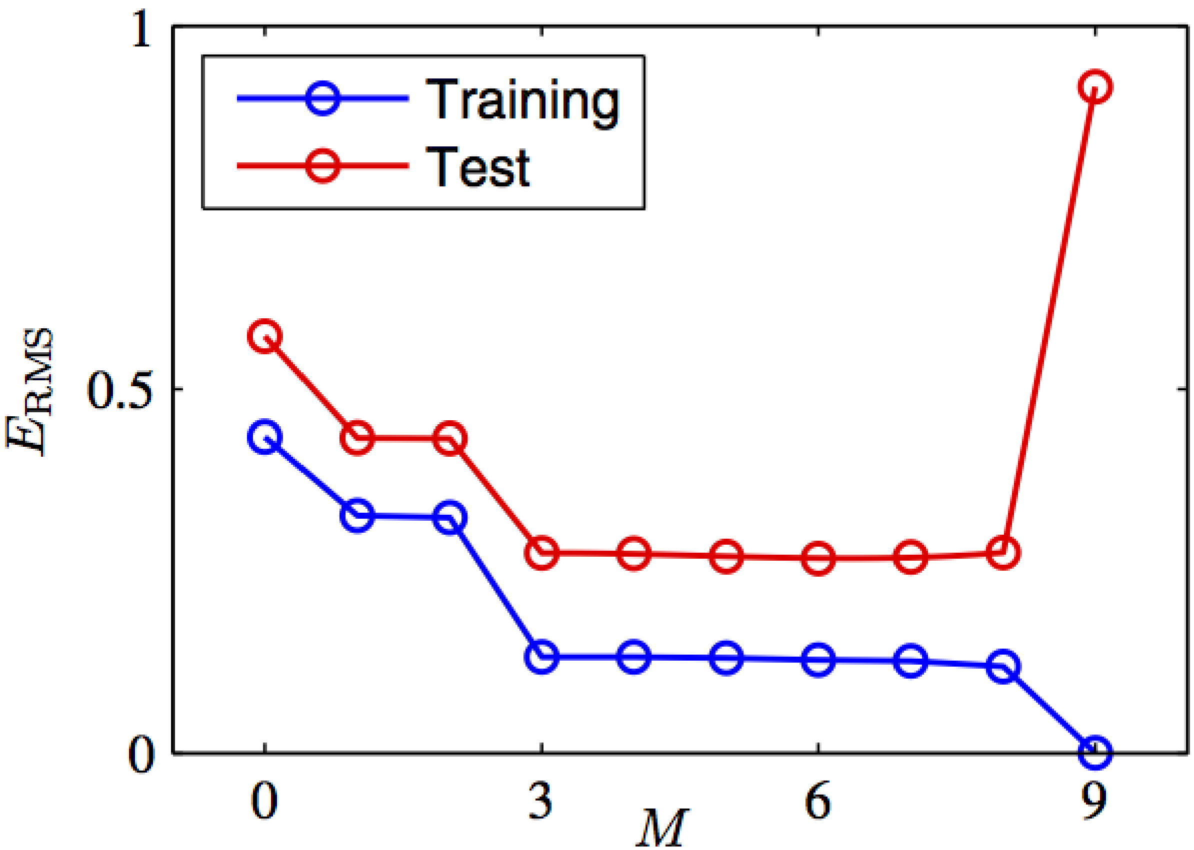
Overfitting and Regularization

- Help the network **generalize** to data it hasn't seen.
- Big problem for **small datasets**.
- Overfitting example (a sine curve vs 9-degree polynomial):

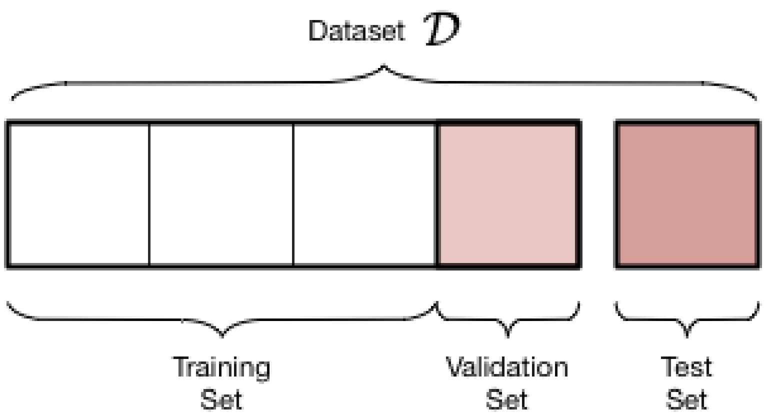


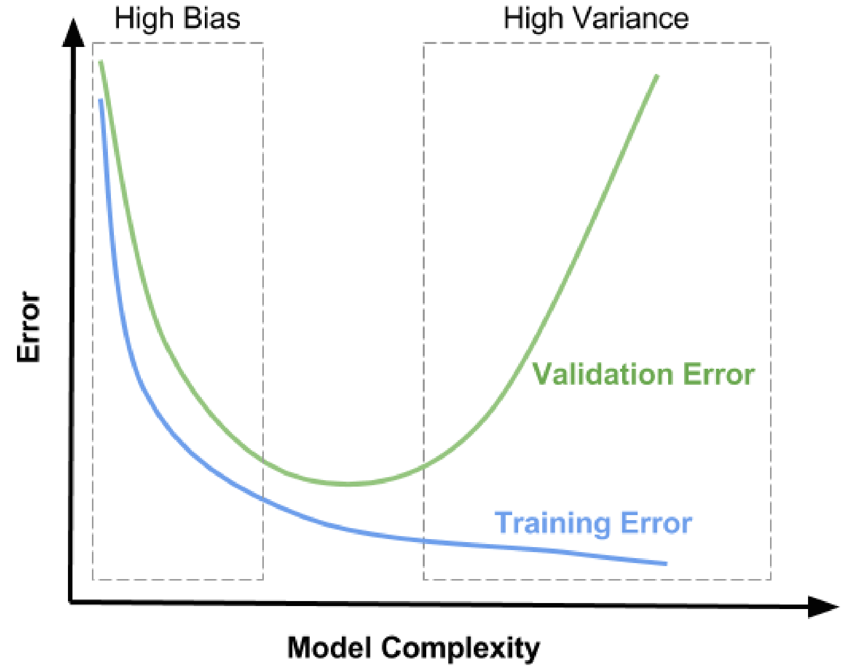
Overfitting and Regularization

- Overfitting: The error decreases in the training set but increases in the test set.



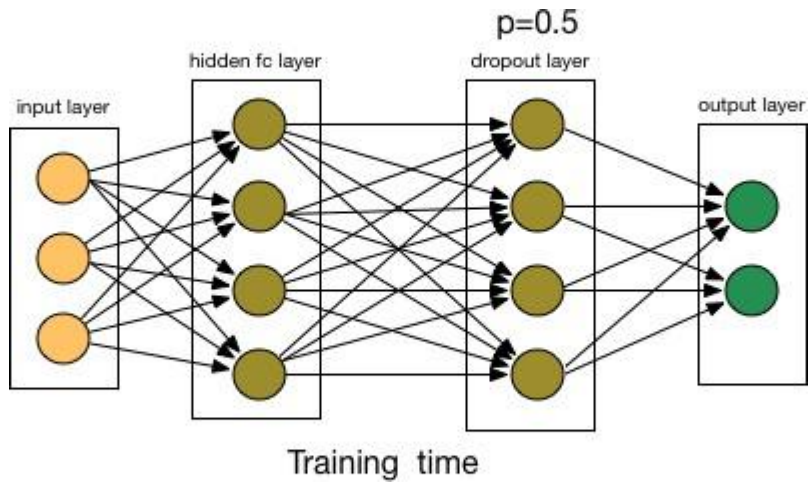
Regularization: Early Stoppage





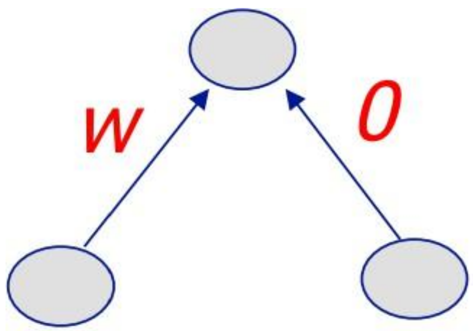
- Create “validation” set (subset of the training set).
  - Validation set is assumed to be a representative of the testing set.
- **Early stoppage:** Stop training (or at least save a checkpoint) when performance on the validation set decreases

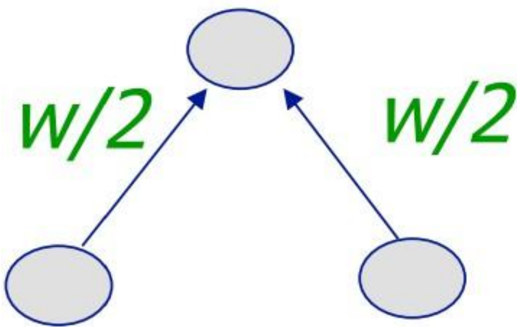
Dropout



- **Dropout:** Randomly remove some nodes in the network (along with incoming and outgoing edges)
- Notes:
  - Usually *p >= 0.5* (*p* is probability of keeping node)
  - Input layers *p* should be much higher (and use noise instead of dropout)
  - Most deep learning frameworks come with a dropout layer

Regularization: Weight Penalty *(aka Weight Decay)*





- **L2 Penalty:** Penalize squared weights. Result:
  - Keeps weight small unless error derivative is very large.
  - Prevent from fitting sampling error.
  - Smoother model (output changes slower as the input change).
  - If network has two similar inputs, it prefers to put half the weight on each rather than all the weight on one.
- **L1 Penalty:** Penalize absolute weights. Result:
  - Allow for a few weights to remain large.

Normalization

- Network Input Normalization
  - *Example:* Pixel to [0, 1] or [-1, 1] or according to mean and std.

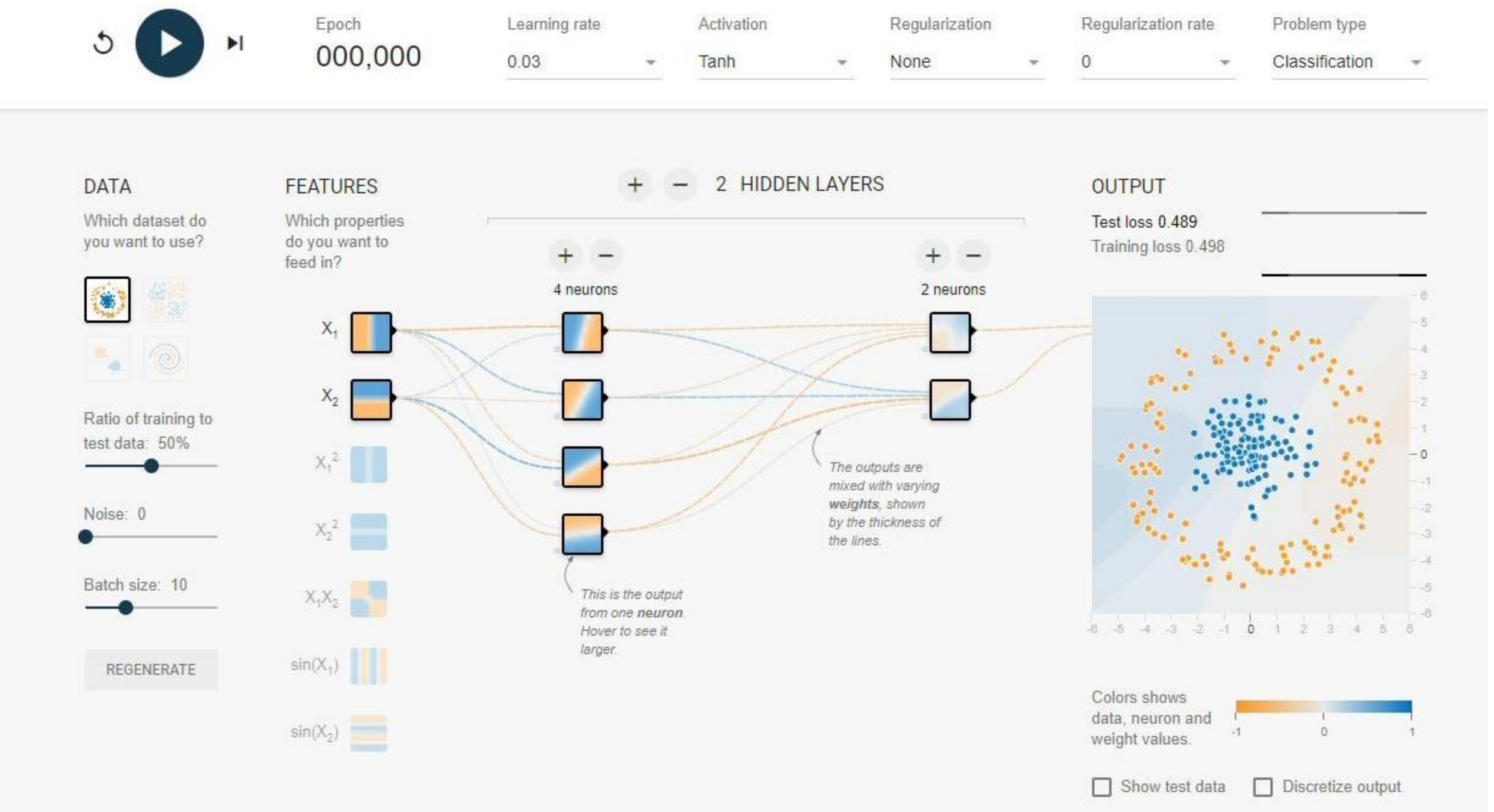
- Batch Normalization (BatchNorm, BN)
  - Normalize hidden layer inputs to mini-batch mean & variance
  - Reduces impact of earlier layers on later layers

- Batch Renormalization (BatchRenorm, BR)
  - Fixes difference b/w training and inference by keeping a moving average asymptotically approaching a global normalization.

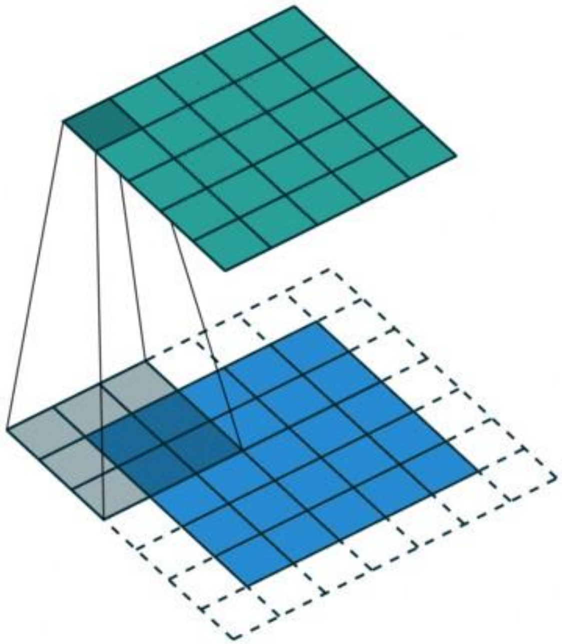
- Other options:
  - Layer normalization (LN) – conceived for RNNs
  - Instance normalization (IN) – conceived for Style Transfer
  - Group normalization (GN) – conceived for CNNs

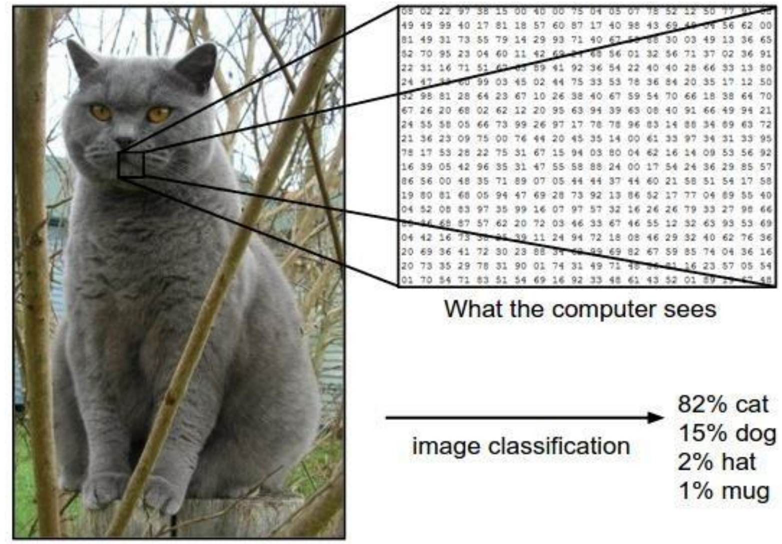
Neural Network Playground

http://playground.tensorflow.org

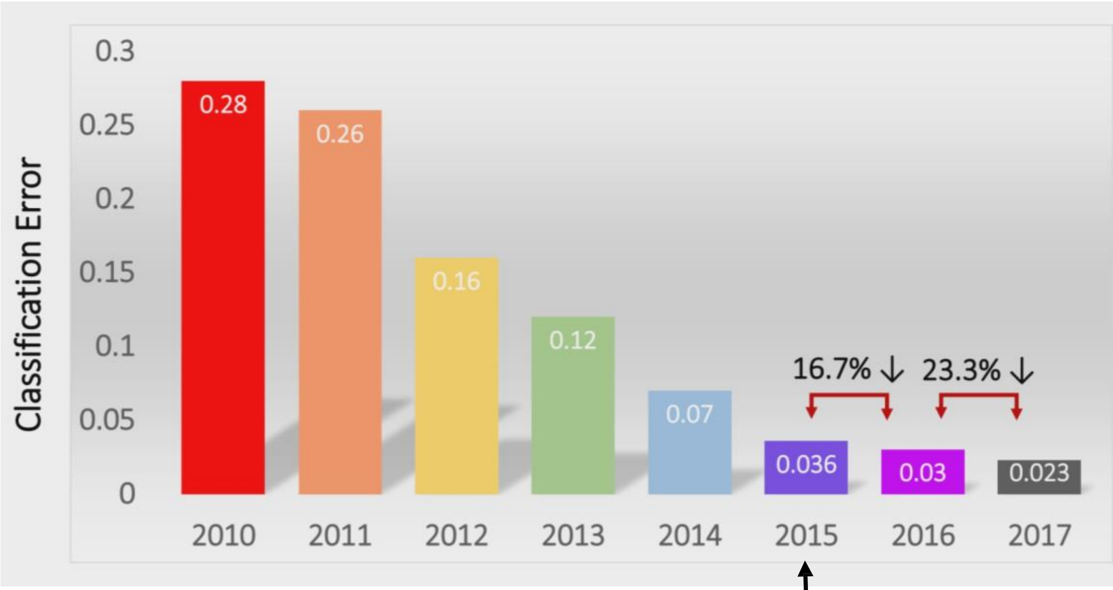


Convolutional Neural Networks: Image Classification





- Convolutional filters: take advantage of spatial invariance

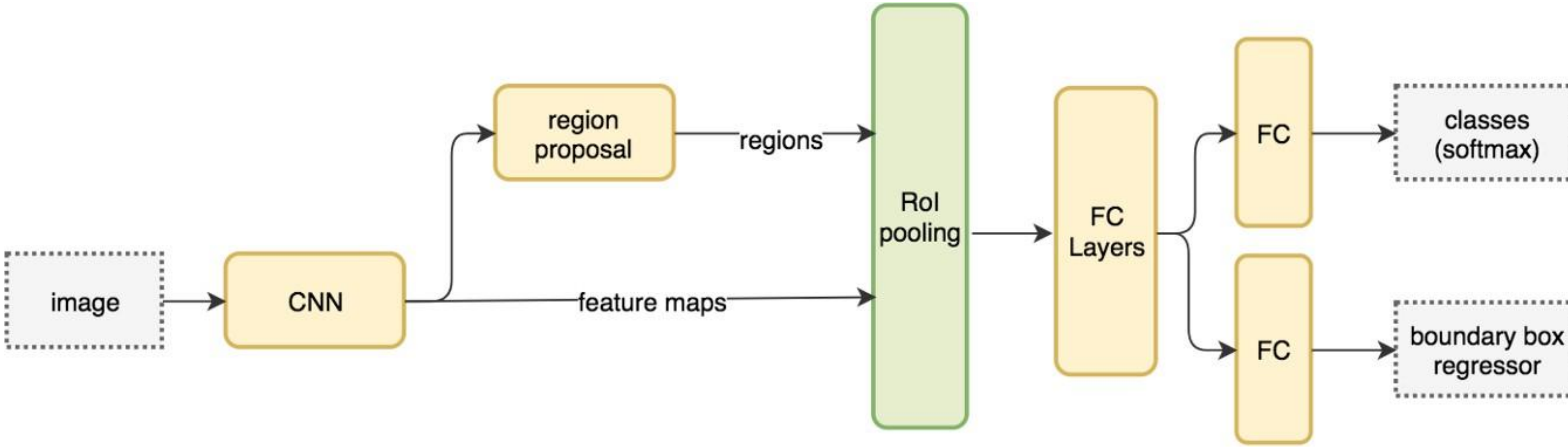


Human error (5.1%) surpassed in 2015

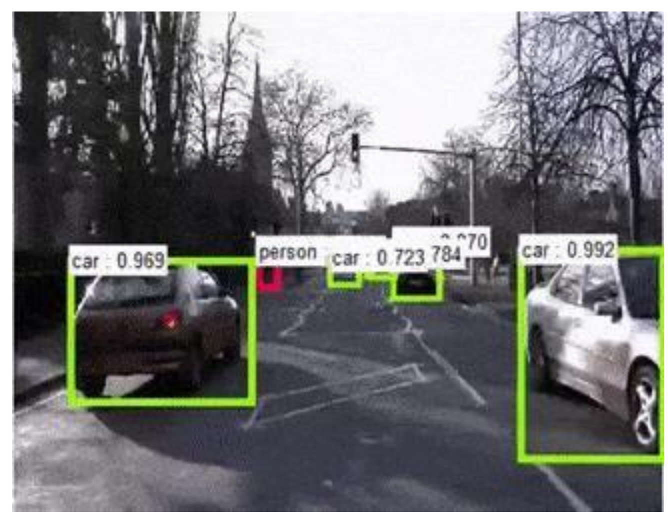
- **AlexNet (2012): First CNN (15.4%)**
  - 8 layers
  - 61 million parameters
- **ZFNet (2013): 15.4% to 11.2%**
  - 8 layers
  - More filters. Denser stride.
- **VGGNet (2014): 11.2% to 7.3%**
  - Beautifully uniform:
    3x3 conv, stride 1, pad 1, 2x2 max pool
  - 16 layers
  - 138 million parameters
- **GoogLeNet (2014): 11.2% to 6.7%**
  - Inception modules
  - 22 layers
  - 5 million parameters
    (throw away fully connected layers)
- **ResNet (2015): 6.7% to 3.57%**
  - More layers = better performance
  - 152 layers
- **CUImage (2016): 3.57% to 2.99%**
  - Ensemble of 6 models
- **SENet (2017): 2.99% to 2.251%**
  - Squeeze and excitation block: network is allowed to adaptively adjust the weighting of each feature map in the convolutional block.

Object Detection / Localization

Region-Based Methods | Shown: Faster R-CNN

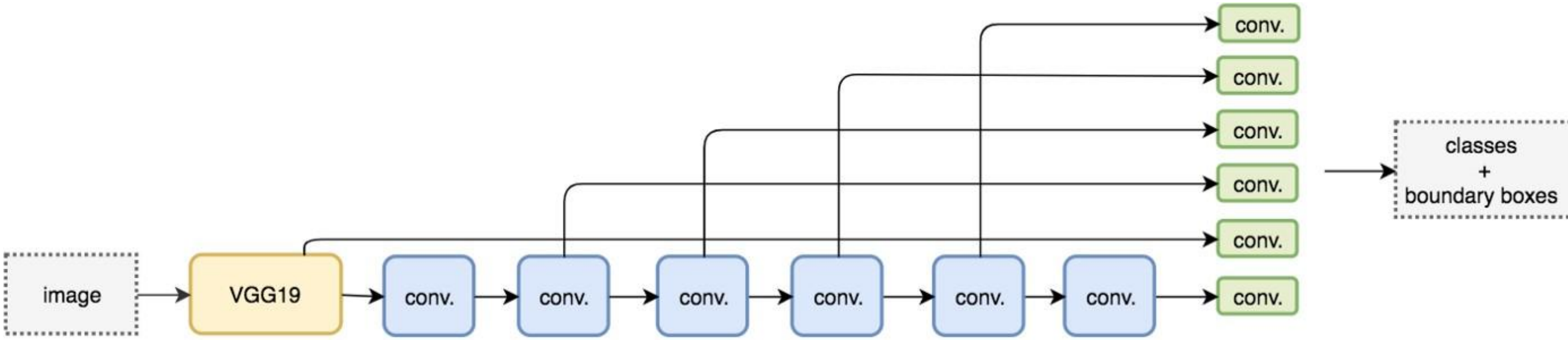


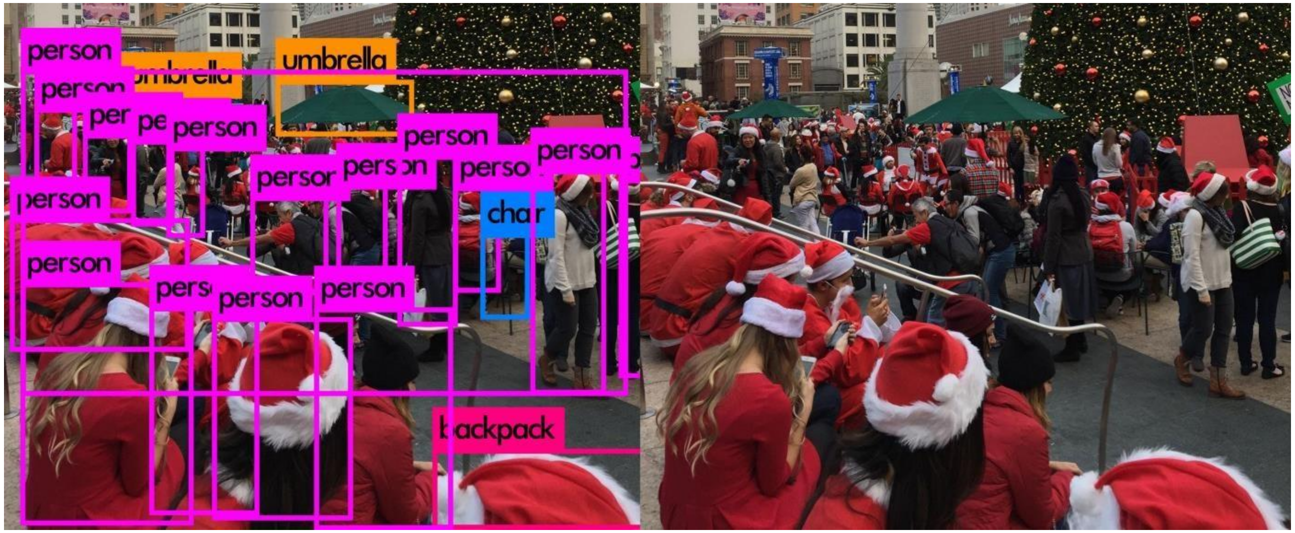

```
ROIs = region_proposal(image)
for ROI in ROIs
    patch = get_patch(image, ROI)
    results = detector(patch)
```



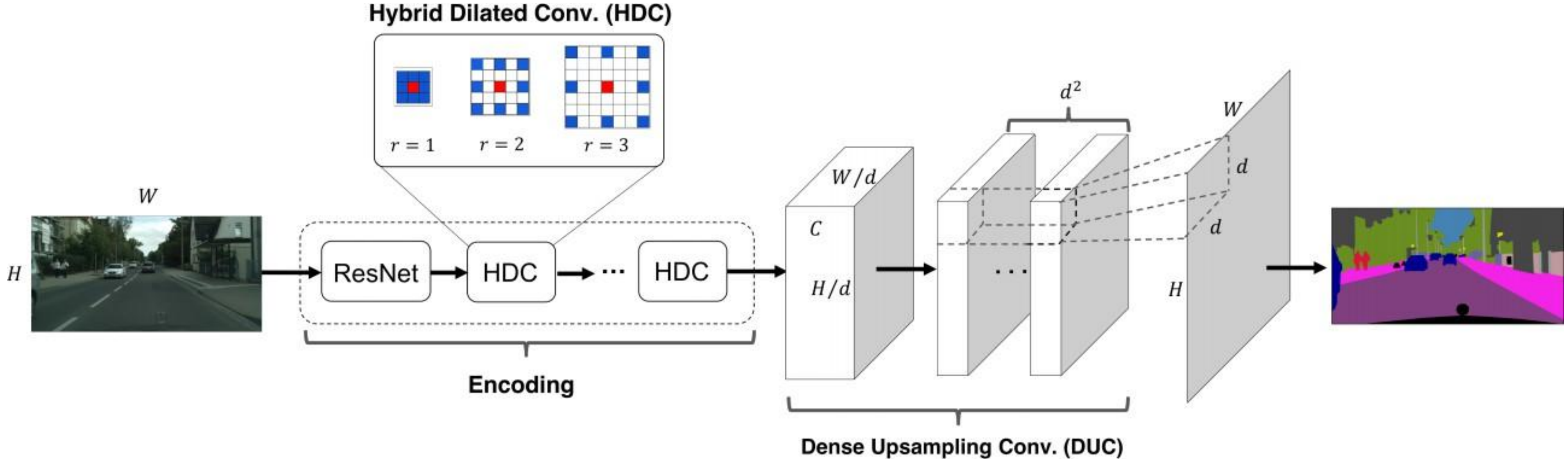
Object Detection / Localization

Single-Shot Methods | Shown: SSD

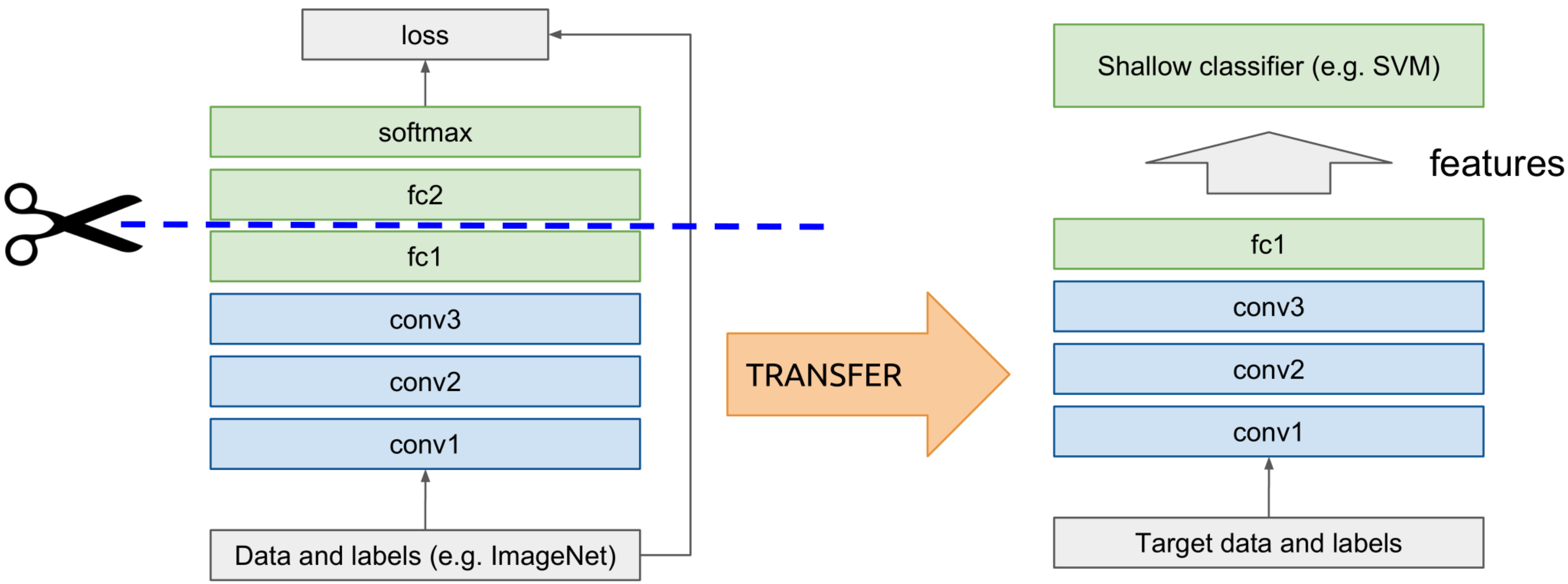




Semantic Segmentation

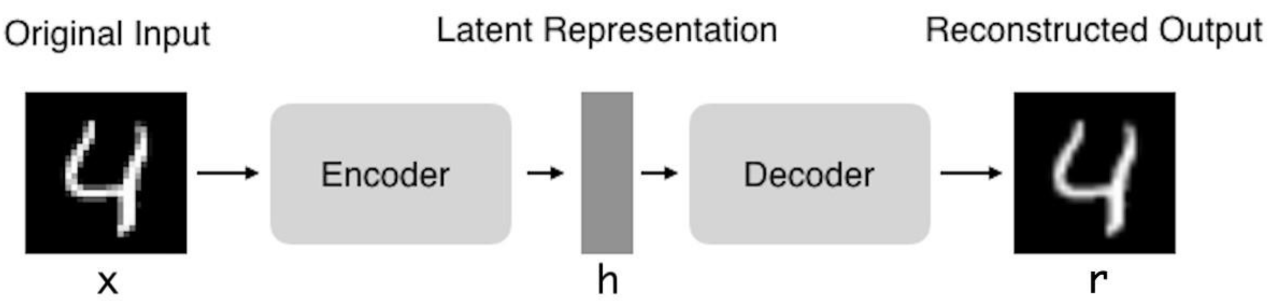


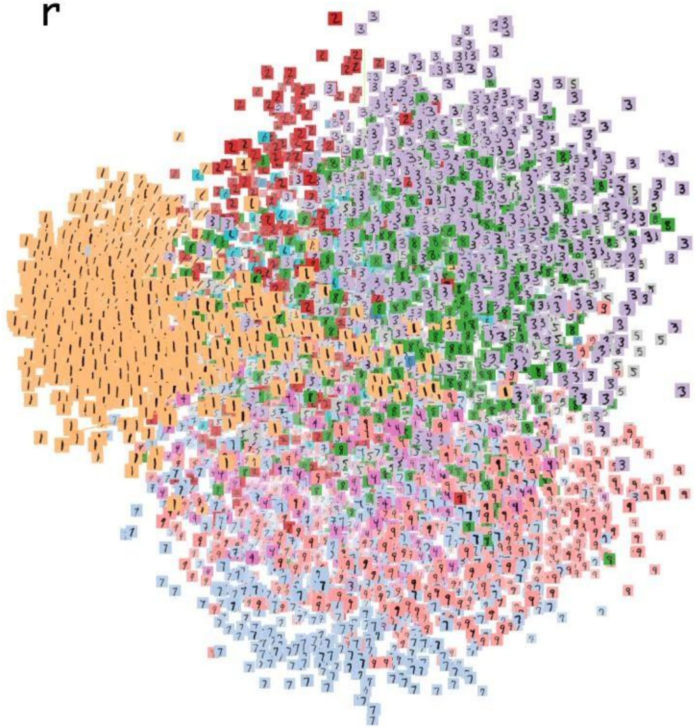
Transfer Learning



- Fine-tune a pre-trained model
- Effective in many applications: computer vision, audio, speech, natural language processing

Autoencoders

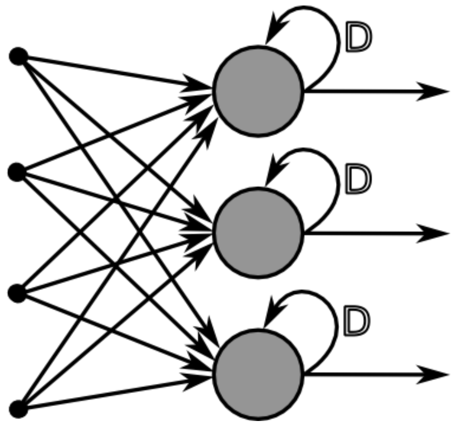


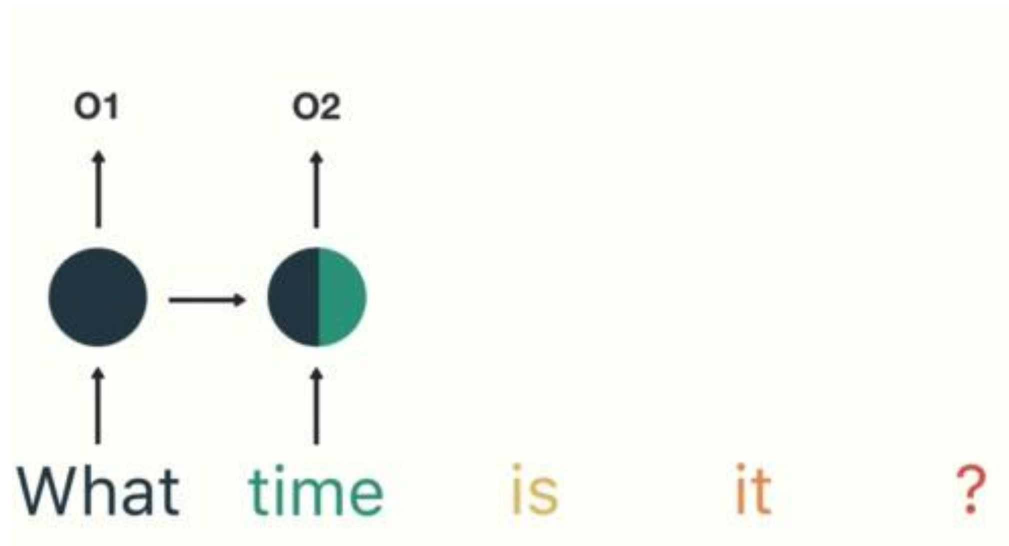


- Unsupervised learning
- Gives embedding
  - Typically better embeddings come from discriminative task

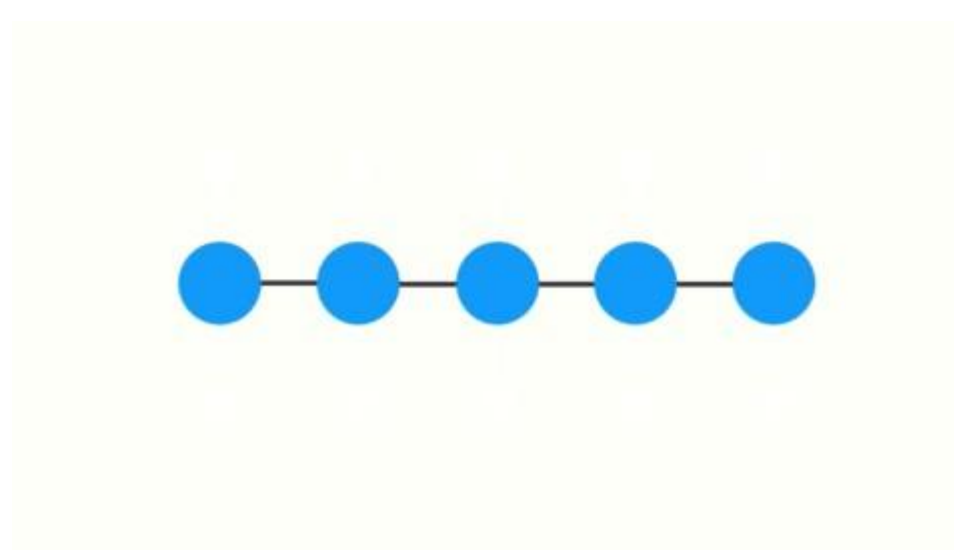
http://projector.tensorflow.org/

Recurrent Neural Networks

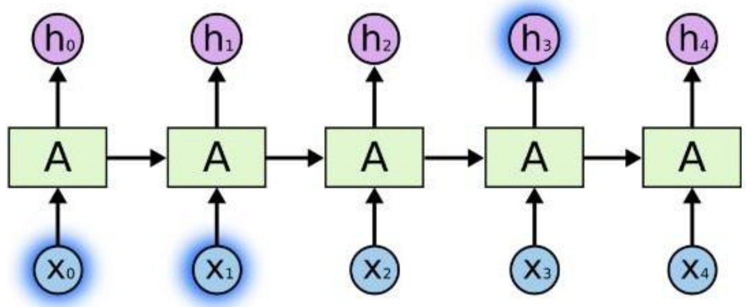




- Applications
  - Sequence Data
  - Text
  - Speech
  - Audio
  - Video
  - Generation

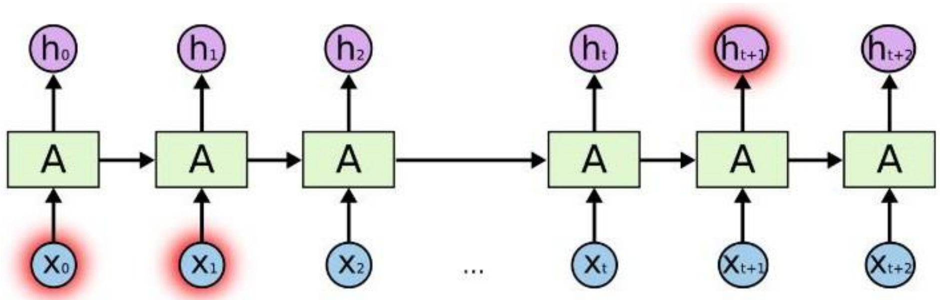


Long-Term Dependency



Context

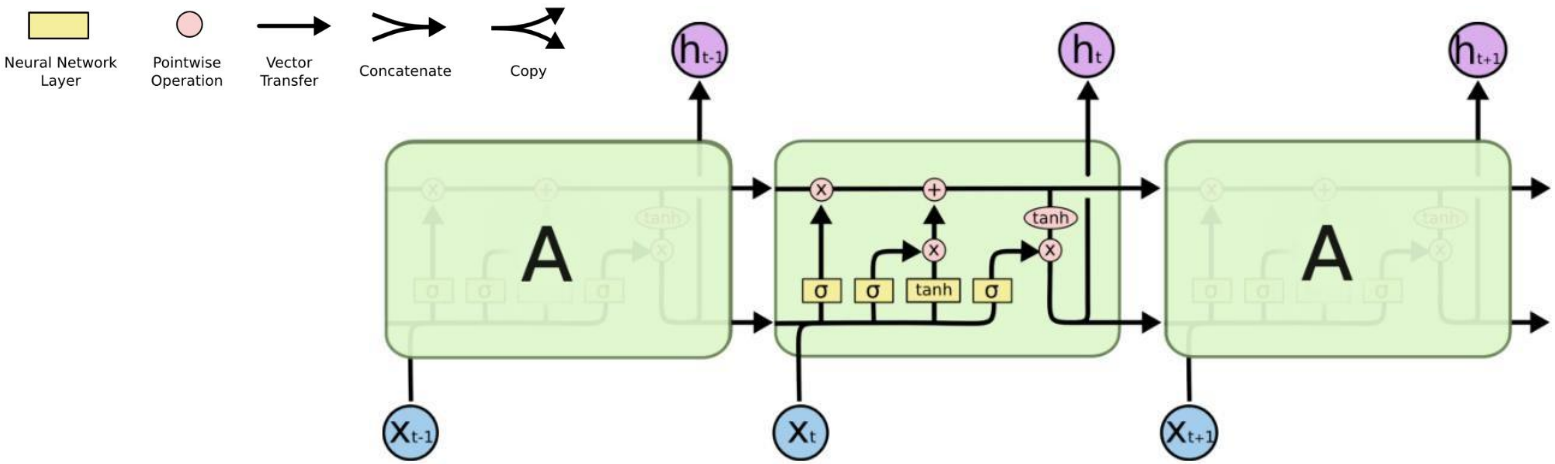
- Short-term dependence:
  Bob is eating an **apple.**
- Long-term dependence:
  Bob likes **apples**. He is hungry and decided to have a snack. So now he is eating an **apple.**



In theory, vanilla RNNs can handle arbitrarily long-term dependence.

In practice, it's difficult.

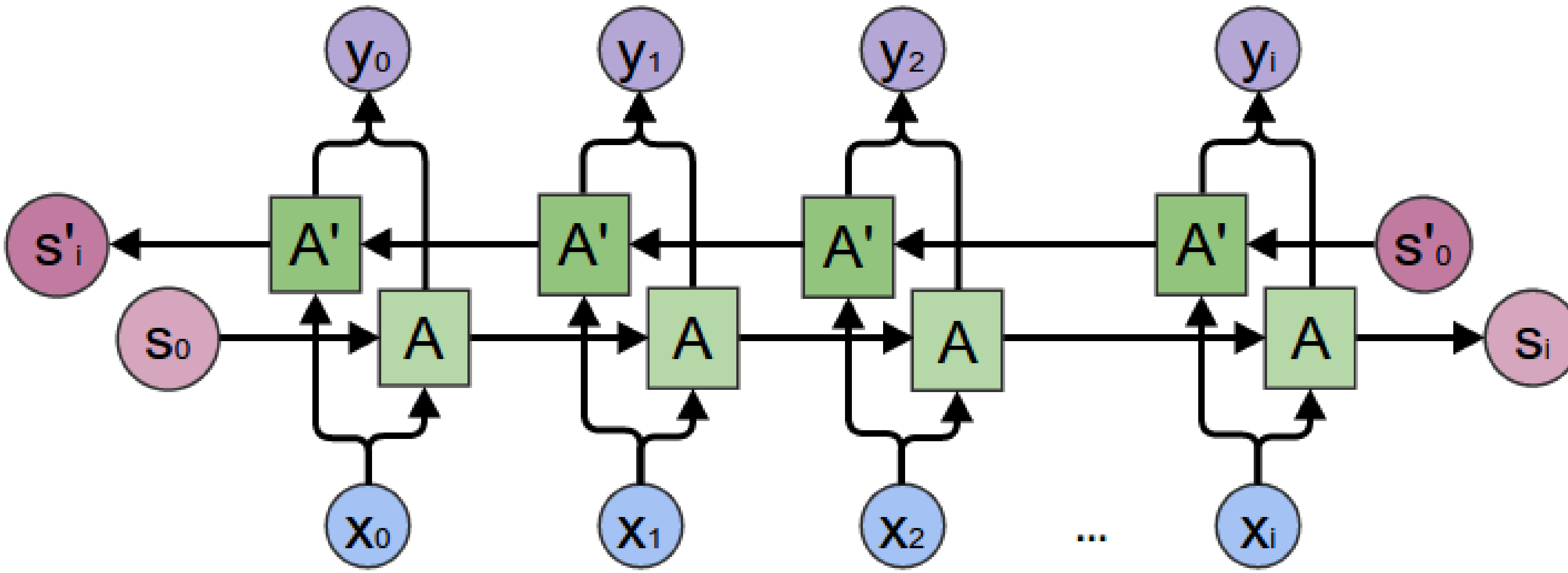
Long Short-Term Memory (LSTM) Networks:
Pick What to Forget and What To Remember



Conveyer belt for **previous state** and **new data**:

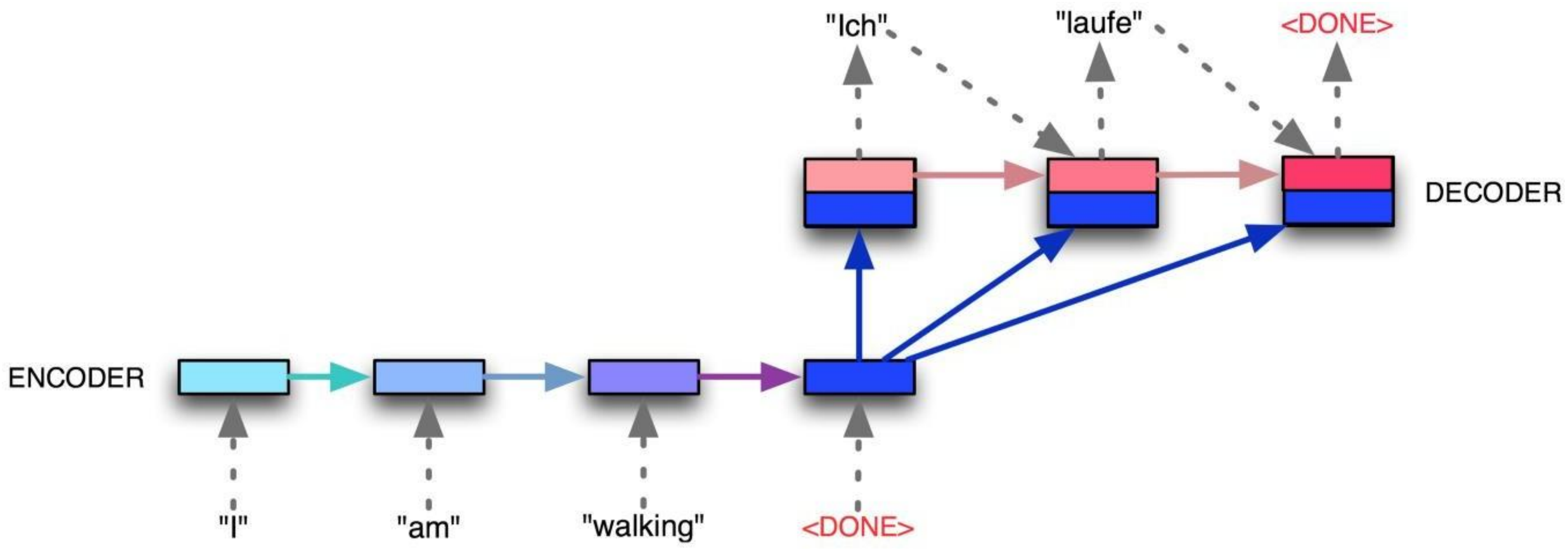
1. Decide what to forget (state)
2. Decide what to remember (state)
3. Decide what to output (if anything)

Bidirectional LSTM



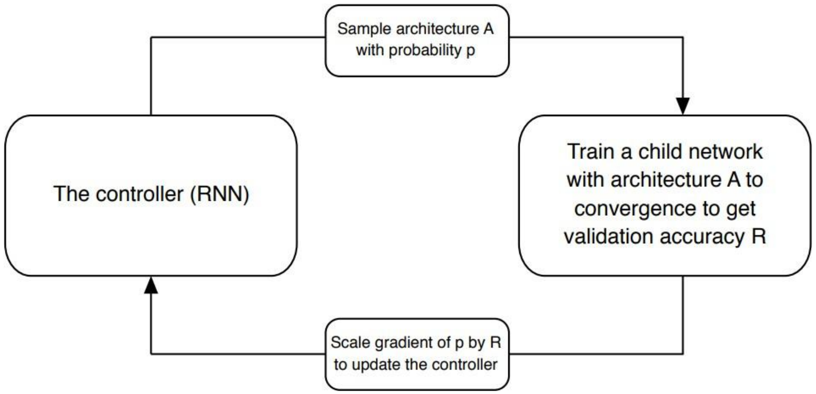
- Learn representations from both previous time steps and future time steps

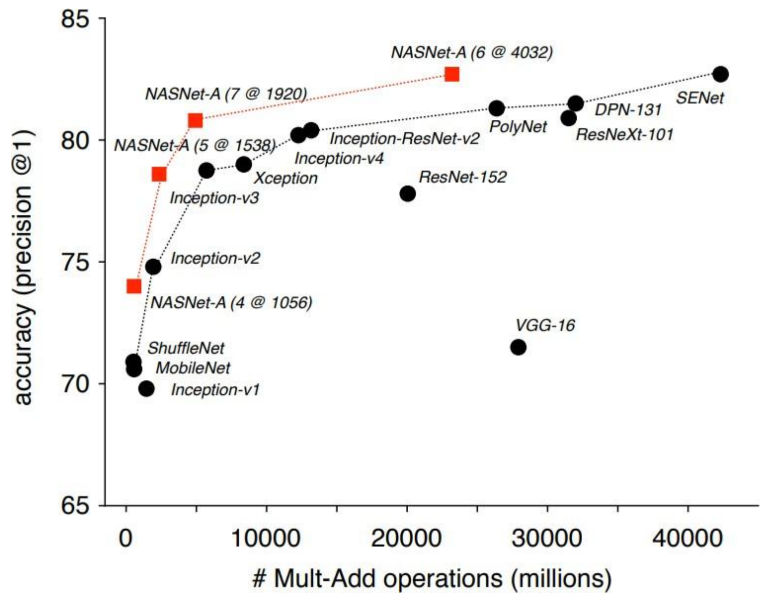
Encoder-Decoder Architecture

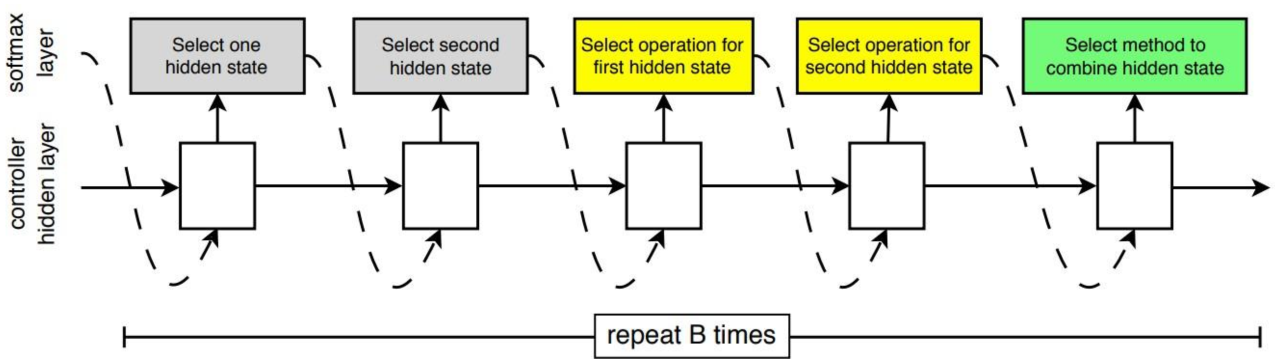


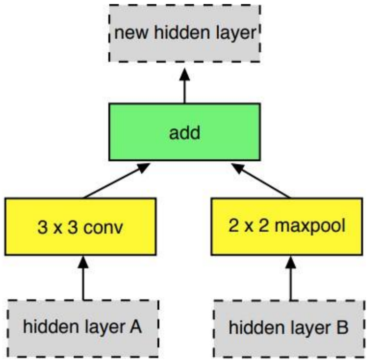
Encoder RNN encodes input sequence into a fixed size vector, and then is passed repeatedly to decoder RNN.

AutoML and Neural Architecture Search (NASNet)

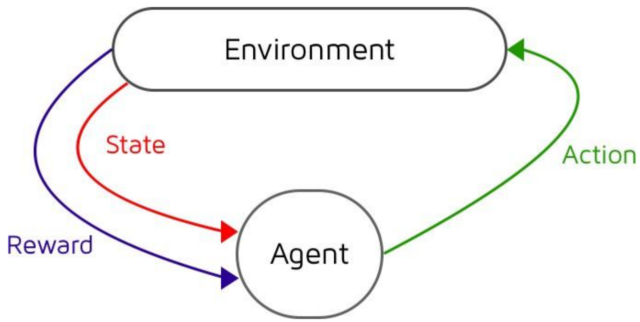


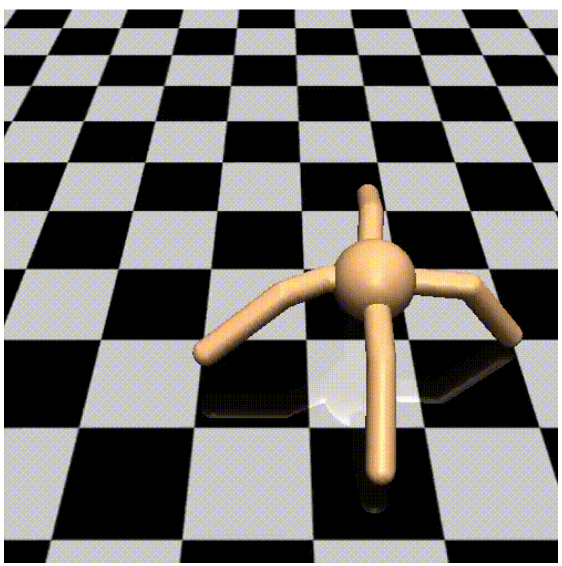






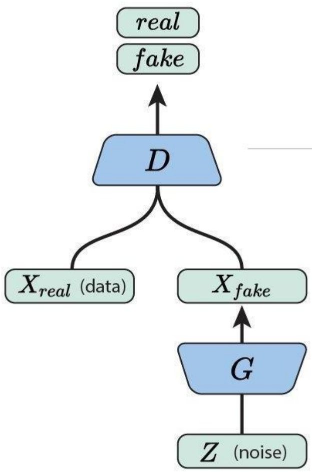
Deep Reinforcement Learning





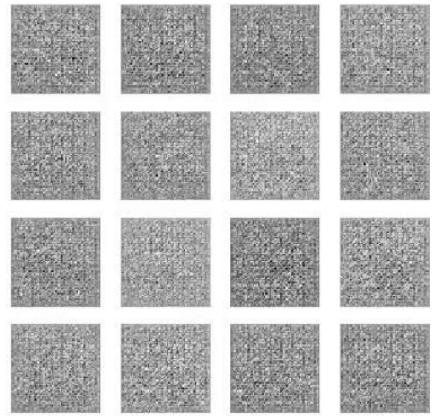
Generative Adversarial Network (GANs)

Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.



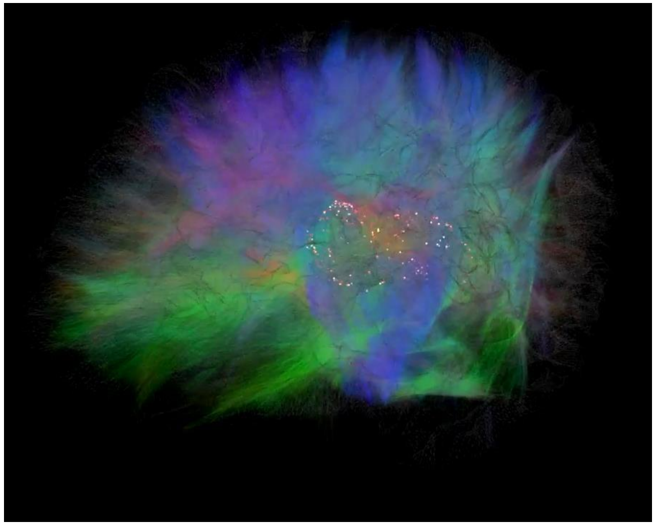
The **discriminator** tries to distinguish genuine data from forgeries created by the generator.

The **generator** turns random noise into immitations of the data, in an attempt to fool the discriminator.

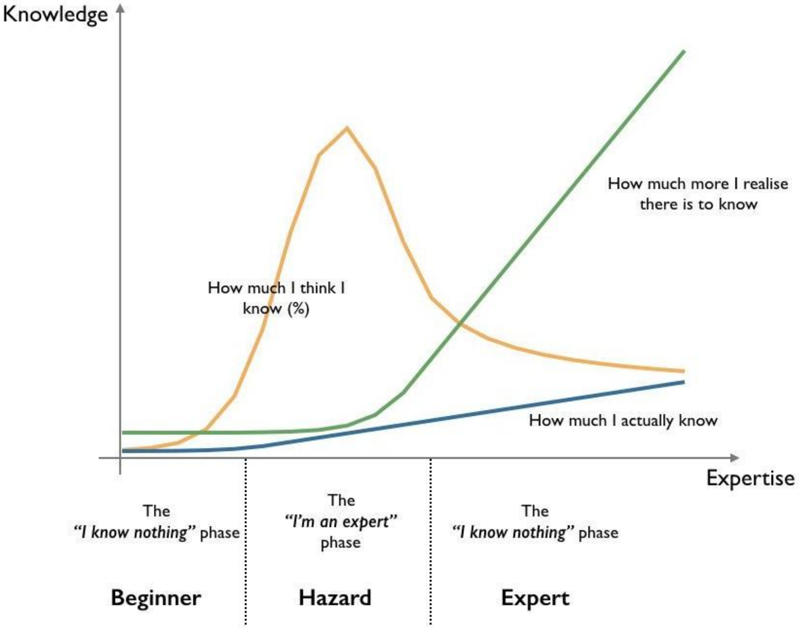


Progressive GAN
10/2017
1024 x 1024

Toward Artificial General Intelligence



- Transfer Learning
- Hyperparameter Optimization
- Architecture Search
- Meta Learning



Thank You

Website:

aoxo.pages.dev

- Videos and slides will be posted online
- Code will be posted on GitHub