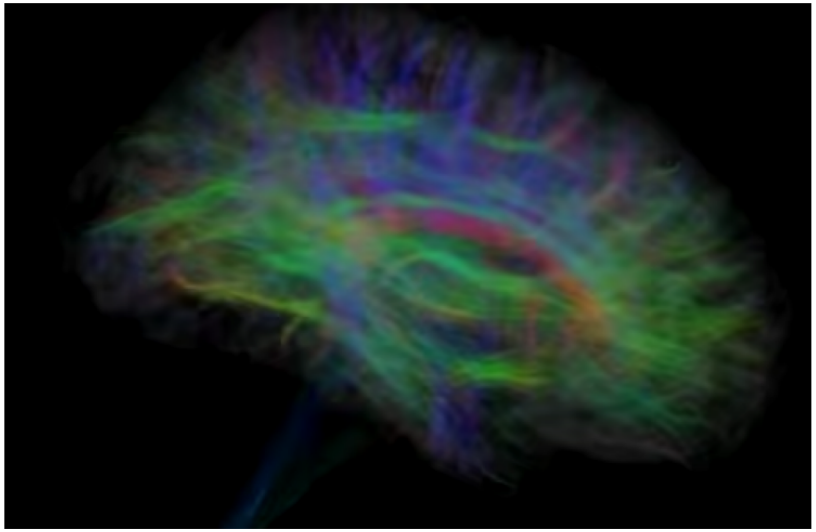


http://www.digicortex.net/node/17

Intro to Deep Learning

Jay Urbain, PhD
Professor, Department of Electrical Engineering and Computer Science
Milwaukee School of Engineering

Credits:

- Deep Learning, Goodfellow, et al.
- Deep Learning in Python, François Chalet
- TensorFlow Tutorials https://www.tensorflow.org/tutorials/
- Introduction to Neural Networks, Ben Gorman http://blog.kaggle.com/2017/11/27/introduction-to-neural-networks/
- MIT Introduction to Self-Driving Cars https://selfdrivingcars.mit.edu
- nVidia Deep Learning Institute

Deep Learning Summary

- **What is it:**
  Extract useful patterns from data.
- **How:**
  Neural network + optimization
- **How (Practical):**
  Python + TensorFlow & friends
- **Hard Part:**
  Good Questions + Good Data
- **Why now:**
  Data, hardware, community, tools, investment
- **Where do we stand?**
  Most big questions of intelligence have not been answered nor properly formulated

Exciting progress:

- Face recognition
- Image classification
- Speech recognition
- Text-to-speech generation
- Handwriting transcription
- Machine translation
- Medical diagnosis
- Cars: drivable area, lane keeping
- Digital assistants
- Ads, search, social recommendations
- Game playing with deep RL

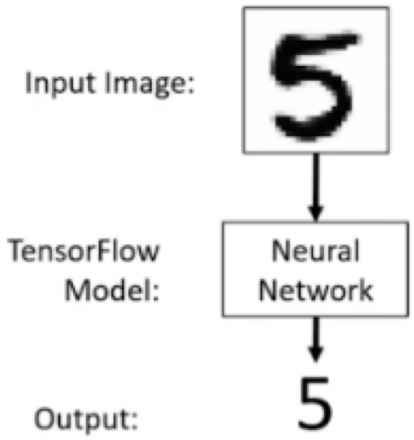
Abbreviated History of Deep Learning

- 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

Tools

- Mark 1 Perceptron – 1960
- Torch – 2002
- CUDA – 2007
- Theano – 2008
- Caffe – 2014
- DistBelief – 2011
- TensorFlow 0.1 – 2015
- PyTorch 0.1 – 2017
- TensorFlow 1.0 – 2017
- PyTorch 1.0 – 2017
- TensorFlow 2.0 – 2019

Tools tf.Keras



1
```python
# import tensorflow and keras (tf.keras not "vanilla" Keras)
import tensorflow as tf
from tensorflow import keras
```

2
```python
# get data
(train_images, train_labels), (test_images, test_labels) = \
keras.datasets.mnist.load_data()
```

3
```python
# setup model
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])

model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

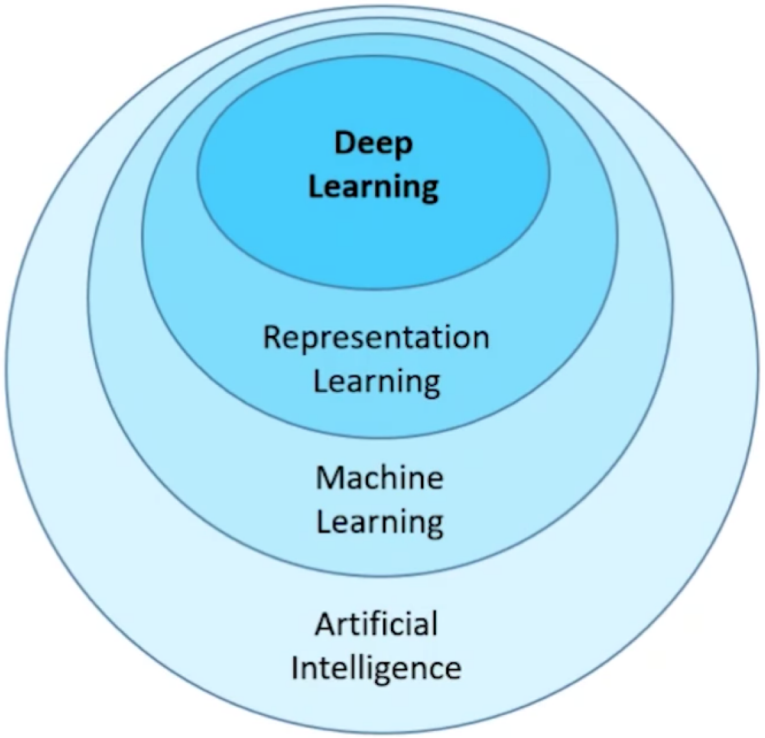
4
```python
# train model
model.fit(train_images, train_labels, epochs=5)
```

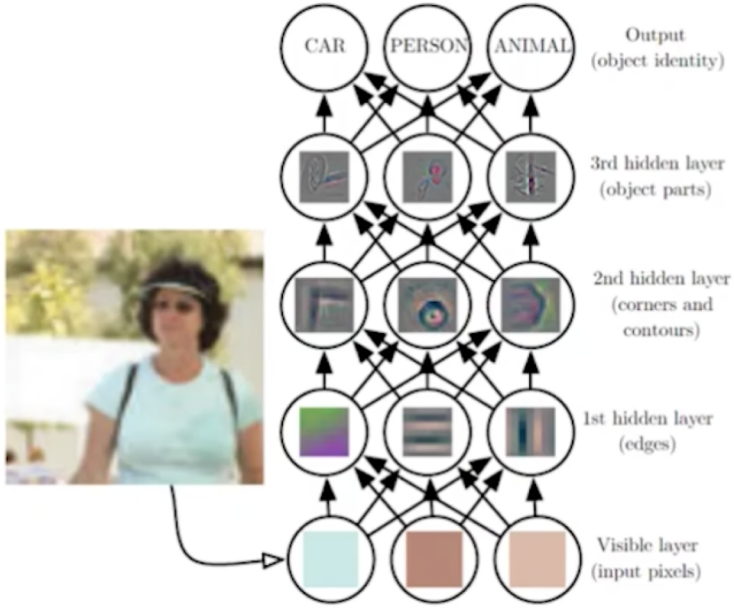
5
```python
# evaluate
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('test accuracy:', test_acc)
```

6
```python
# make predictions
predictions = model.predict(test_images)
```

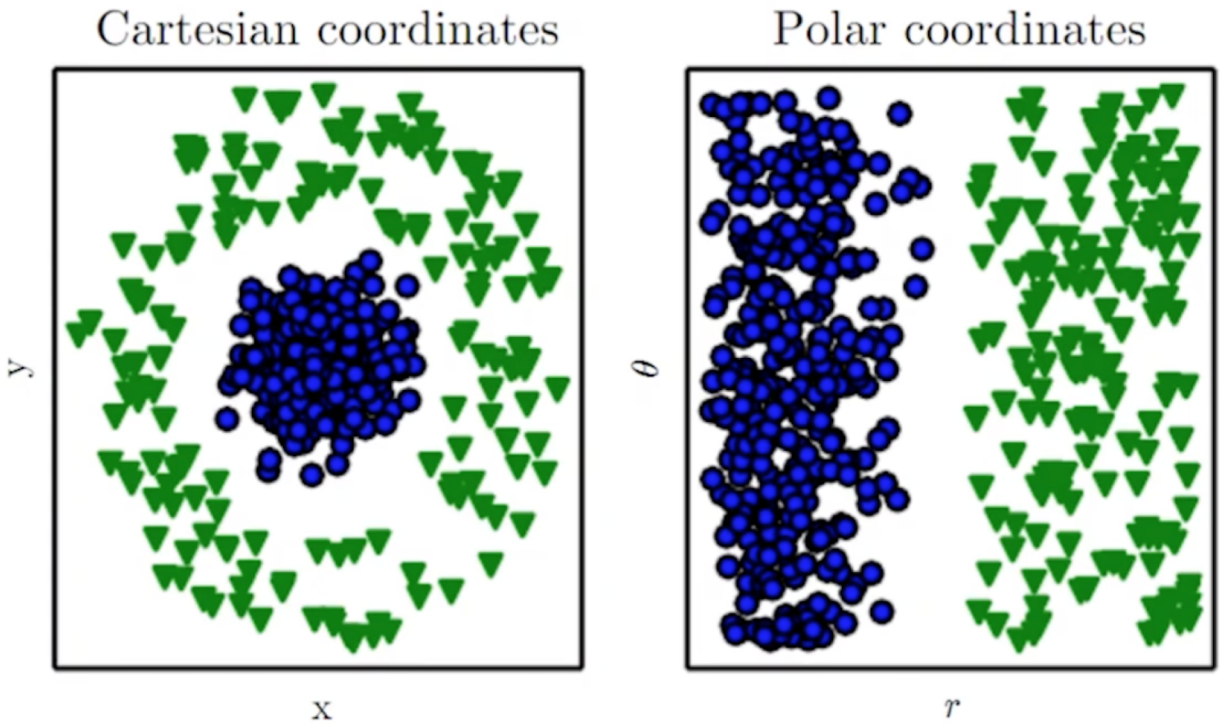
Deep Learning is **Representation Learning**

(aka Feature Learning)



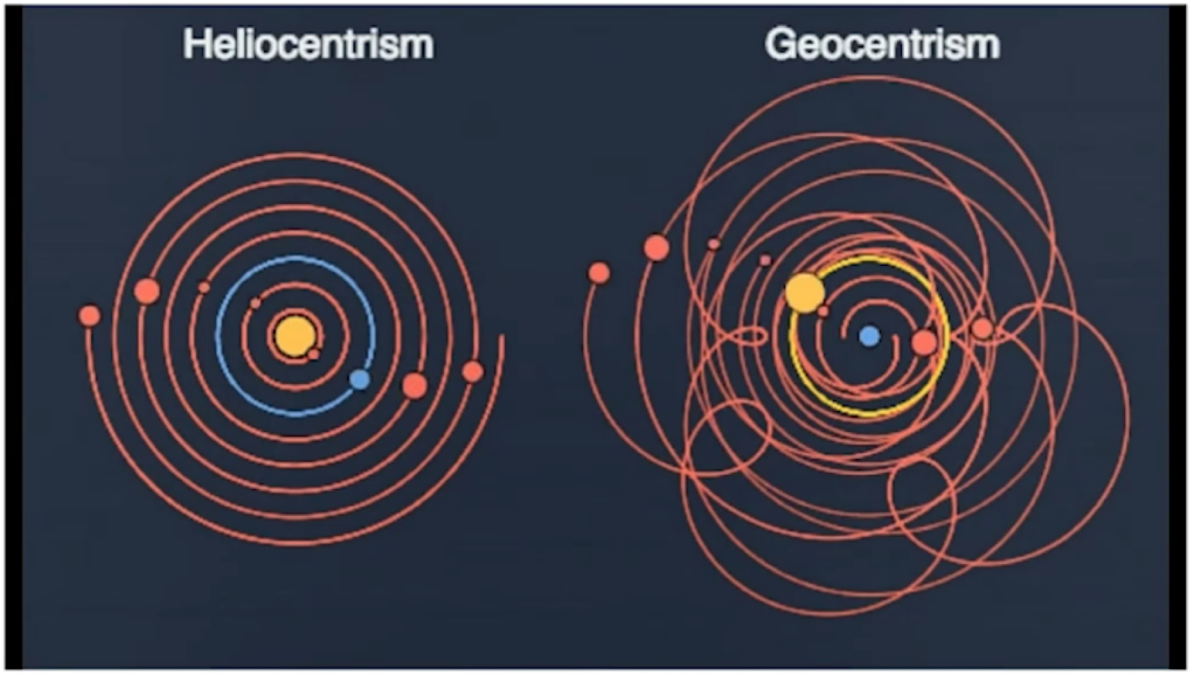


Representation Matters



Task: Draw a line to separate the **green triangles** and **blue circles**.

Representation Matters



Sun-Centered Model
(Formalized by Copernicus in 16th century)

Earth-Centered Model

"History of science is the history of compression progress."
- Jürgen Schmidhuber

Deep learning =
Learning representations/features

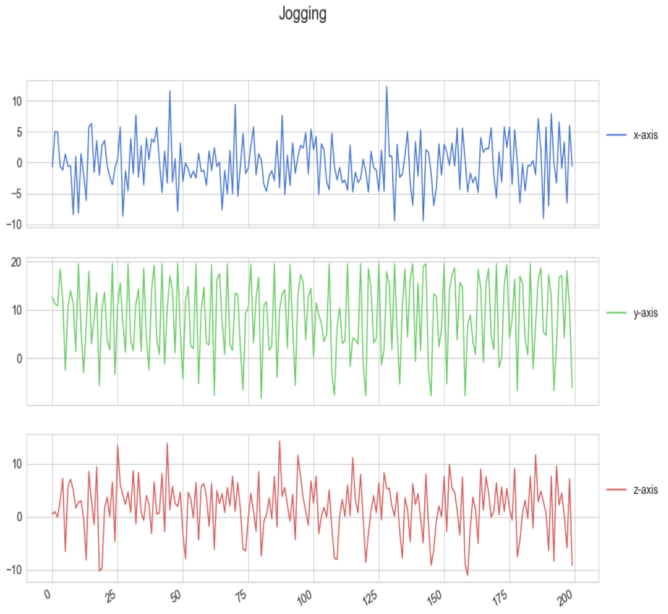
– The traditional model of pattern recognition (since the late 50's)
  – Fixed/engineered features (or fixed kernel) + trainable classifier

– End-to-end learning / Feature learning / Deep learning
  – Trainable features (or kernel) + trainable classifier

Sequence classification: activity recognition

Temporal sensor analysis and activity tracking

	user	activity	timestamp	x-axis	y-axis	z-axis
0	33	Jogging	49105962326000	-0.694638	12.680544	0.503953
1	33	Jogging	49106062271000	5.012288	11.264028	0.953424
2	33	Jogging	49106112167000	4.903325	10.882658	-0.081722
3	33	Jogging	49106222305000	-0.612916	18.496431	3.023717
4	33	Jogging	49106332290000	-1.184970	12.108489	7.205164



Traditional models: activity recognition

http://www.cis.fordham.edu/wisdm/public_files/sensorKDD-2010.pdf

Average[3]: Average acceleration (for each axis)

Standard Deviation[3]: Standard deviation (for each axis)

Average Absolute Difference[3]: Average absolute difference between the value of each of the 200 readings within the ED and the mean value over those 200 values (for each axis)

Average Resultant Acceleration[1]: Average of the square roots of the sum of the values of each axis squared $\sqrt{(x_i^2 + y_i^2 + z_i^2)}$ over the ED

Time Between Peaks[3]: Time in milliseconds between peaks in the sinusoidal waves associated with most activities (for each axis)

Binned Distribution[30]: We determine the range of values for each axis (maximum – minimum), divide this range into 10 equal sized bins, and then record what fraction of the 200 values fell within each of the bins.

Table 2: Accuracies of Activity Recognition

	% of Records Correctly Predicted			
	J48	Logistic Regression	Multilayer Perceptron	Straw Man
Walking	89.9	**93.6**	91.7	37.2
Jogging	96.5	98.0	**98.3**	29.2
Upstairs	59.3	27.5	**61.5**	12.2
Downstairs	**55.5**	12.3	44.3	10.0
Sitting	**95.7**	92.2	95.0	6.4
Standing	**93.3**	87.0	91.9	5.0
Overall	85.1	78.1	**91.7**	37.2

	user	activity	timestamp	x-axis	y-axis	z-axis
0	33	Jogging	49105962326000	-0.694638	12.680544	0.503953
1	33	Jogging	49106062271000	5.012288	11.264028	0.953424
2	33	Jogging	49106112167000	4.903325	10.882658	-0.081722
3	33	Jogging	49106222305000	-0.612916	18.496431	3.023717
4	33	Jogging	49106332290000	-1.184970	12.108489	7.205164

Traditional models: activity recognition

Note: Improved features using traditional machine learning models (SVM, RF, LR) can yield an accuracy in the ~92% range (Quihao Jin, Jay Urbain):

- 3-sec mean
- 5-sec mean
- Median filtering
- Acceleration axis deltas
- FFT – Fast Fourier Transform

DL Experiment – WISDM Activity Dataset

Temporal sensor analysis and activi

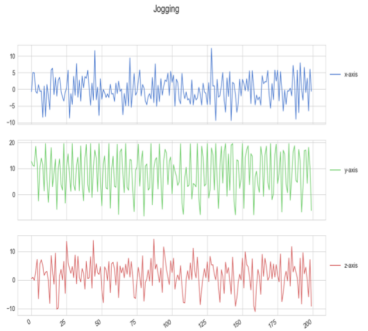
	user	activity	timestamp	x-axis	y-axis	z-axis
0	33	Jogging	49105962326000	-0.694638	12.680544	0.503953
1	33	Jogging	49106062271000	5.012288	11.264028	0.953424
2	33	Jogging	49106112167000	4.903325	10.882658	-0.081722
3	33	Jogging	49106222305000	-0.612916	18.496431	3.023717
4	33	Jogging	49106332290000	-1.184970	12.108489	7.205164

Layer (type)	Output Shape	Param #
conv1d_9 (Conv1D)	(None, 100, 64)	1024
max_pooling1d_9 (MaxPooling1	(None, 50, 64)	0
dropout_9 (Dropout)	(None, 50, 64)	0
conv1d_10 (Conv1D)	(None, 50, 128)	41088
max_pooling1d_10 (MaxPooling	(None, 25, 128)	0
dropout_10 (Dropout)	(None, 25, 128)	0
gru_5 (GRU)	(None, 25, 50)	26850
flatten_5 (Flatten)	(None, 1250)	0
dense_8 (Dense)	(None, 6)	7506

Total params: 76,468

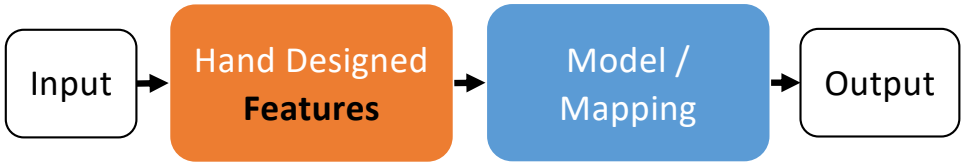
Model	Accuracy
1) 1-layer LSTM	94.2
2) 2-layer LSTM	95.5
3) 1-layer Convnet	94.5
5) 2-layer Convnet	98.7
5) 2-layer Convnet + 1-layer LSTM	98.5

Test accuracy: 0.985

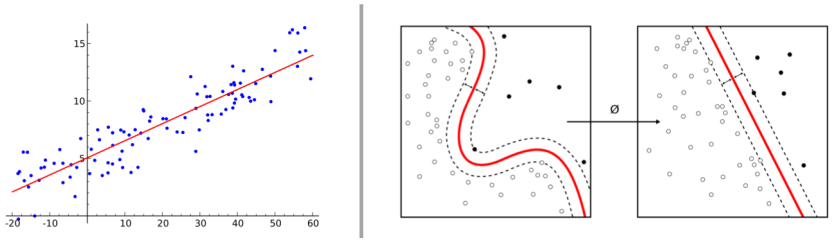


Difference in Workflow

Classic Machine Learning [1990 : now]

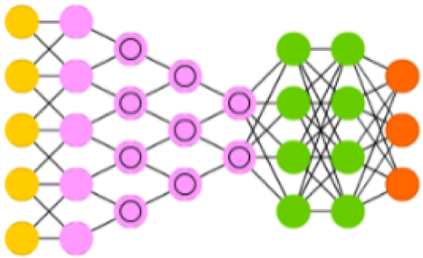


Examples [Regression and SVMs]



Deep/End-to-End Learning [2012 : now]

Example [Conv Net]

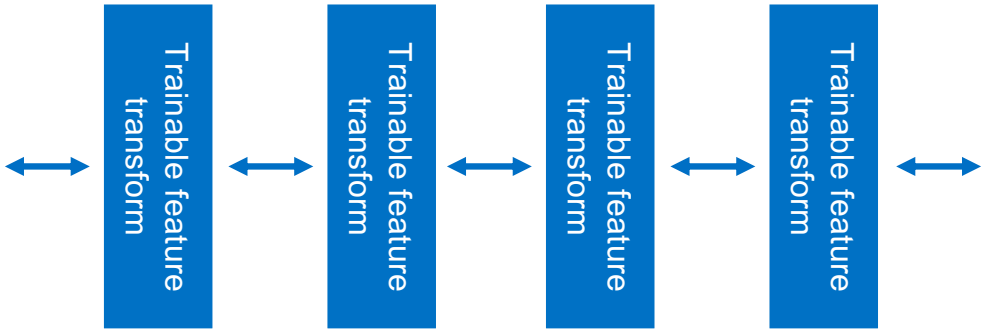


Machine learning workflow shifts from engineering features for “shallow” models
to architecting deep learning models with the ability to learn hierarchical representations of features

Trainable feature hierarchy

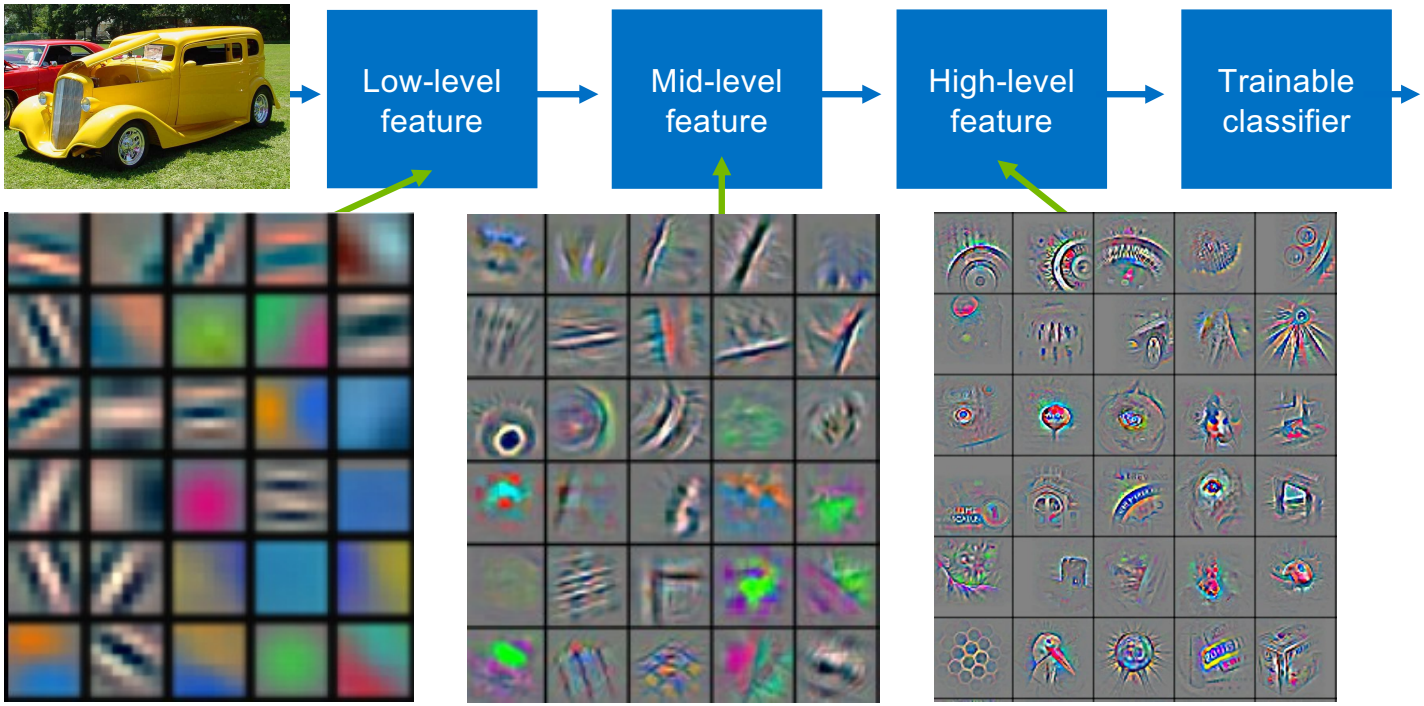
Hierarchy of representations with increasing level of abstraction. Each stage is a kind of trainable feature transform

- Image recognition
  - Pixel → edge → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story/semantic understanding
- Speech
  - Sample → spectral band → sound → … → phone → phoneme → word

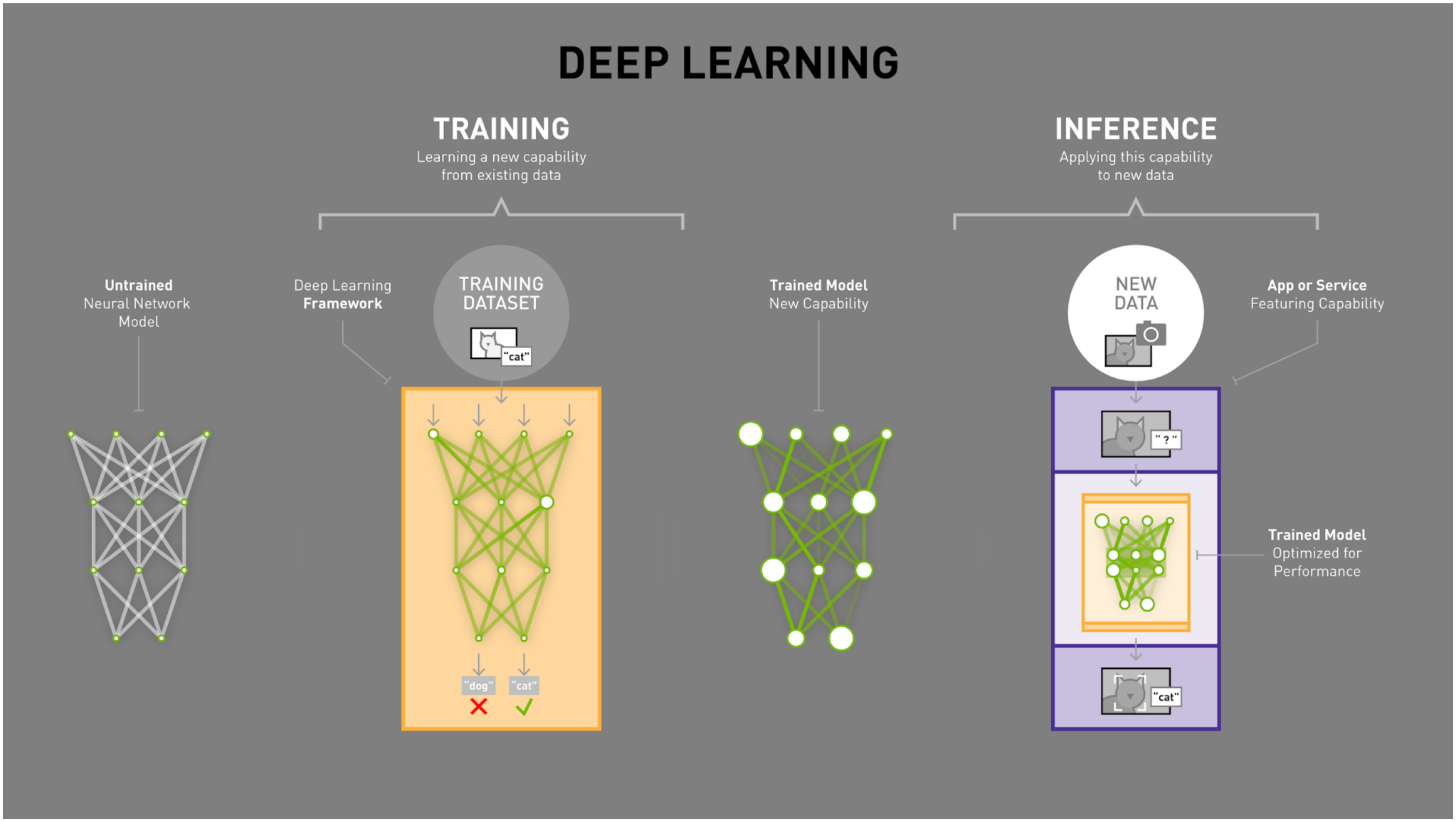


Deep learning = learning hierarchical representations

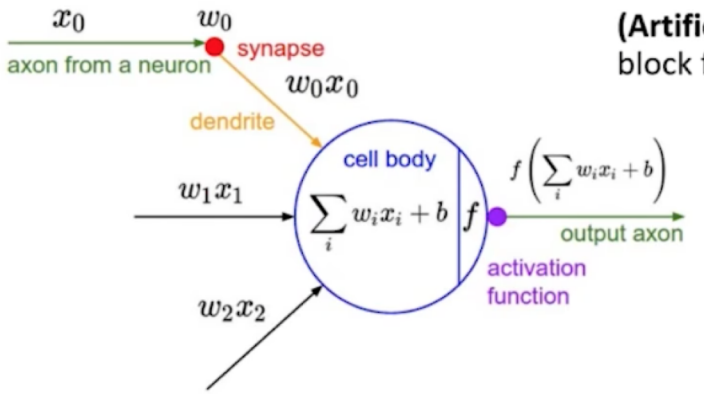
It's deep if it has more than one stage of non-linear feature transformation



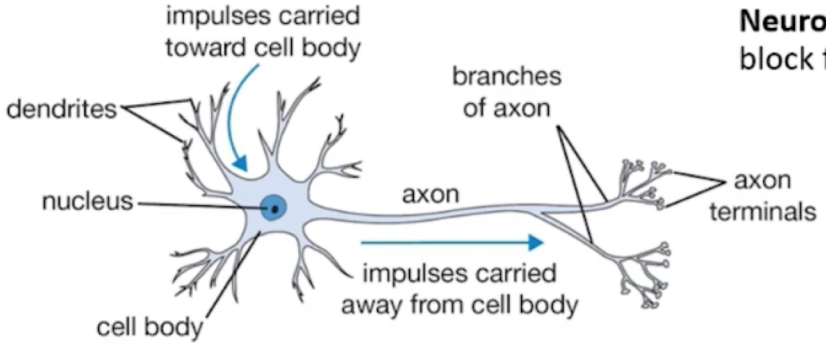
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]


DEEP LEARNING
TRAINING
Learning a new capability from existing data
INFERENCE
Applying this capability to new data
Untrained Neural Network Model
Deep Learning Framework
TRAINING DATASET
"cat"
"dog"
"cat"
Trained Model New Capability
NEW DATA
App or Service Featuring Capability
" ? "
"cat"
Trained Model Optimized for Performance

Neuron: Biological Inspiration for Computation

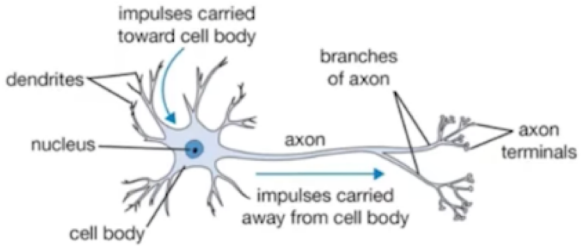


(Artificial) Neuron: computational building block for the "neural network"

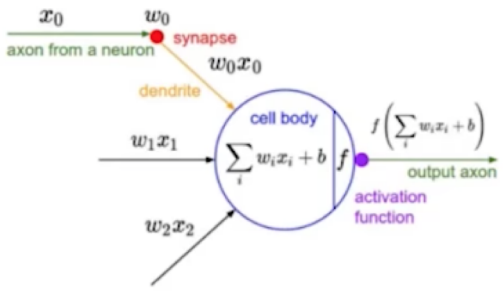


Neuron: computational building block for the brain

Neuron: Biological Inspiration for Computation



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



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

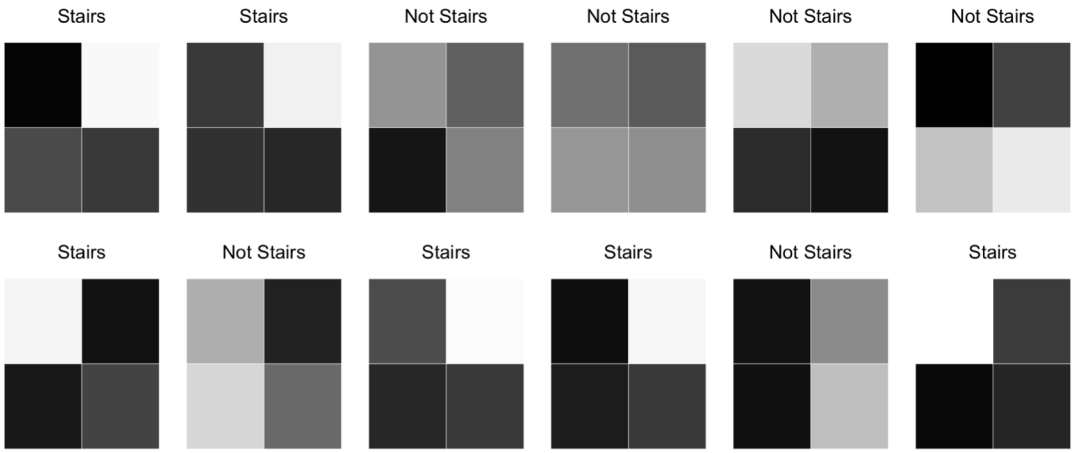
Key Difference:

- **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, Goodfellow, et al.

Non-linear problem: predict Stairs/Not Stairs

- Collection of grayscale images, each a 2×2 grid of pixels where each pixel has an intensity value between 0 (white) and 255 (black).
- The goal is to build a model that identifies images with a “stairs” pattern.



https://gormanalysis.com/introduction-to-neural-networks/

Single Layer Perceptron

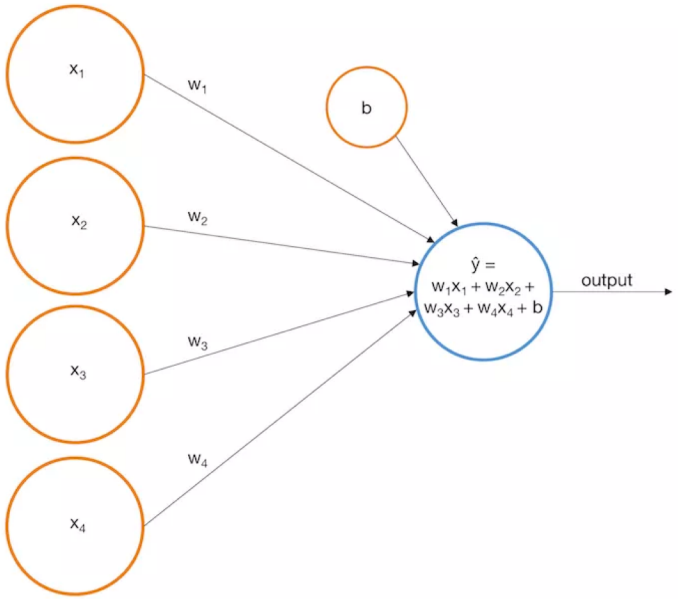
$$f(x) = \begin{cases} 1 & \text{if } w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 > threshold \\ 0 & \text{otherwise} \end{cases}$$

Let's re-express this as follows

$$\widehat{y} = \mathbf{w} \cdot \mathbf{x} + b$$

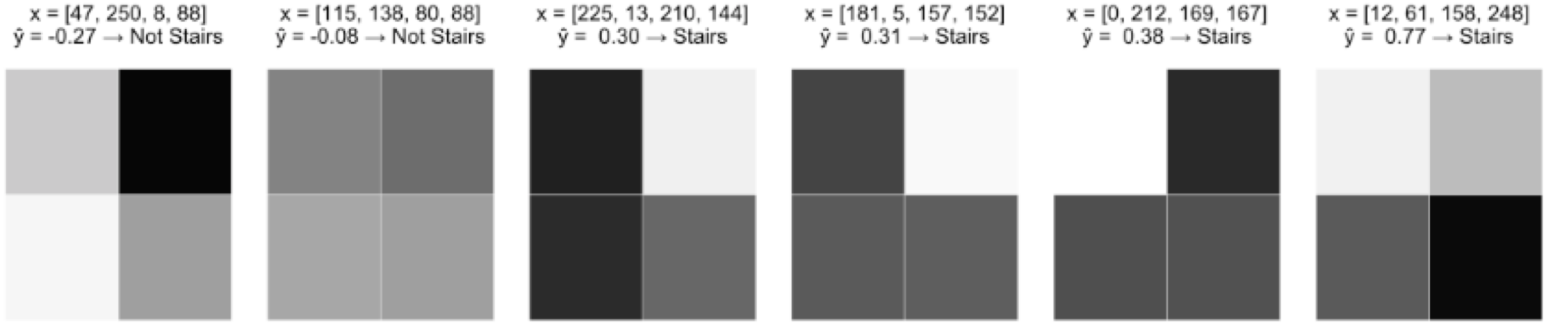
$$f(x) = \begin{cases} 1 & \text{if } \widehat{y} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Here $\widehat{y}$ is our *prediction score*.

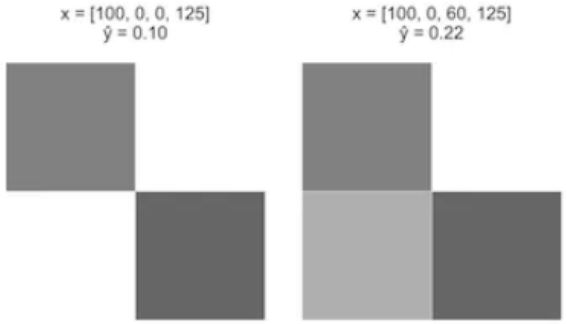


$$\widehat{y} = -0.0019x_1 + -0.0016x_2 + 0.0020x_3 + 0.0023x_4 + 0.0003$$

Single Layer Perceptron - problems



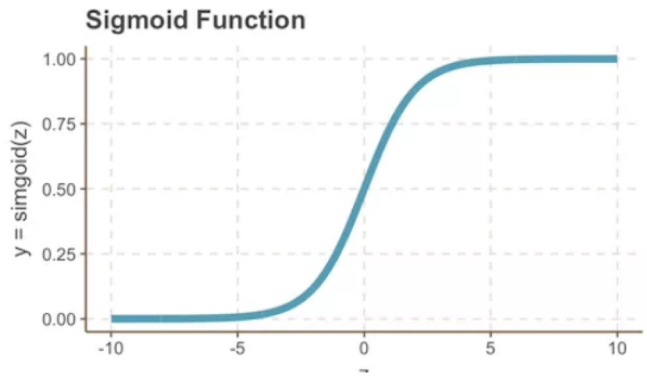
- The model outputs a real number whose value correlates with the concept of likelihood, but there's no basis to interpret the values as probabilities, they can be outside the range [0, 1].
- The model can't capture the non-linear relationship (interactions) between the variables and the target:



Start with an image, x = [100, 0, 0, 125]. Increase x3 from 0 to 60
Variables are independent.

Single Layer Perceptron with Sigmoid Activation: use non-linear activation function

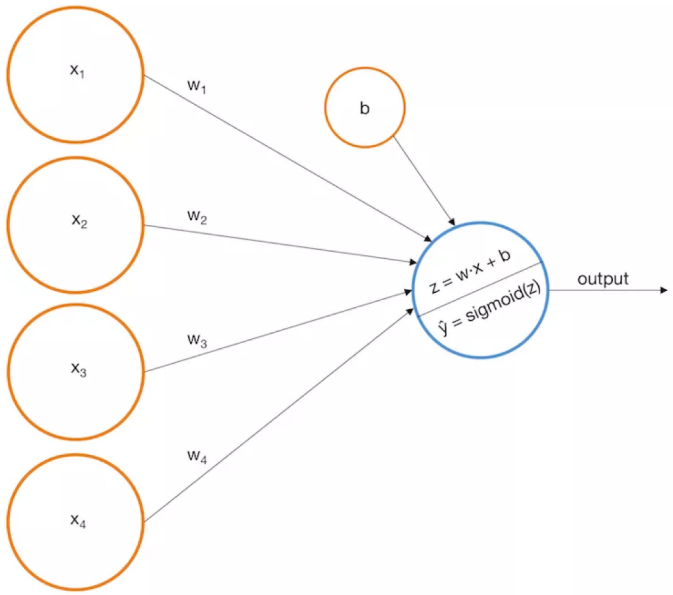
$$sigmoid(z) = \frac{1}{1+e^{-z}}$$



$$z = w \cdot x = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

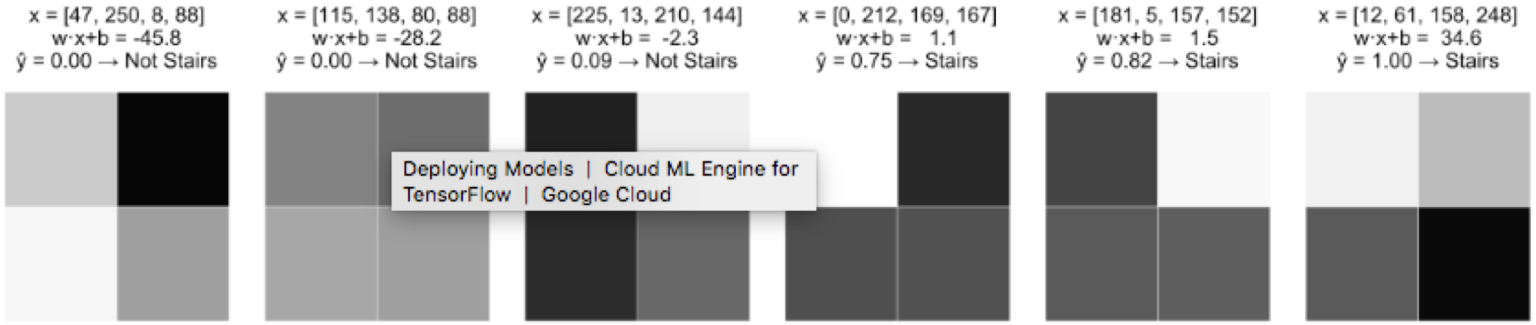
$$\hat{y} = sigmoid(z) = \frac{1}{1+e^{-z}}$$

$$f(x) = \begin{cases} 1 & \text{if } \hat{y} > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

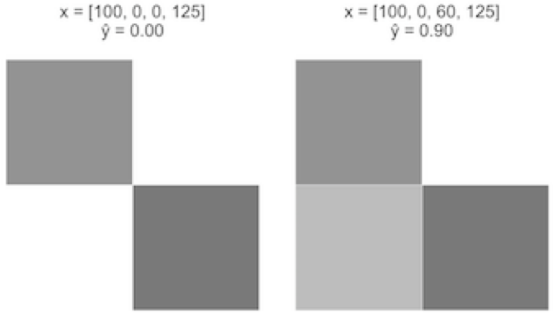


Logistic Regression

Single Layer Perceptron with Sigmoid Activation: sigmoid activation function



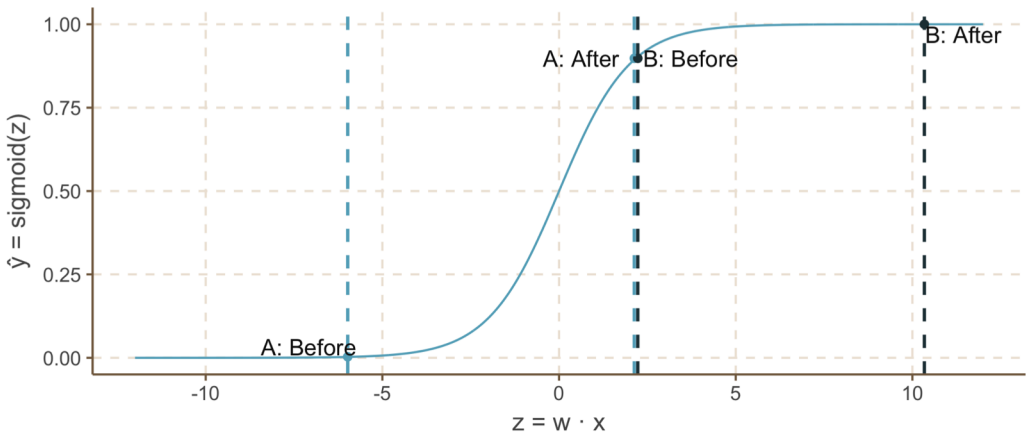
- Fixes two problems from perceptron

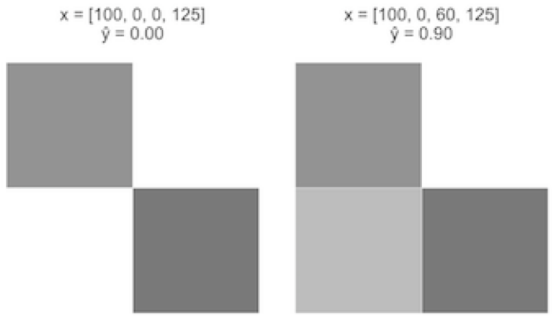


Start with an image, x = [100, 0, 0, 125]. Increase x3 from 0 to 60.

Sigmoid

- The sigmoid function causes the increase of 60 (right image) to “fire” (increase rapidly) as z=w·x increases, but the pace slows down as z continues to increase.
- This aligns with our intuition that the right image should reflect a greater increase in the likelihood of stairs versus the left image.





Still have problems

- yˆ has a monotonic relationship with each variable. What if we want to identify lightly shaded stairs?
- The model does not account for variable interaction.
  - Increasing or decreasing a pixel by x3 should potentially increase *or* decrease yˆ depending on the values of the other variables. The current model has no way of achieving this.

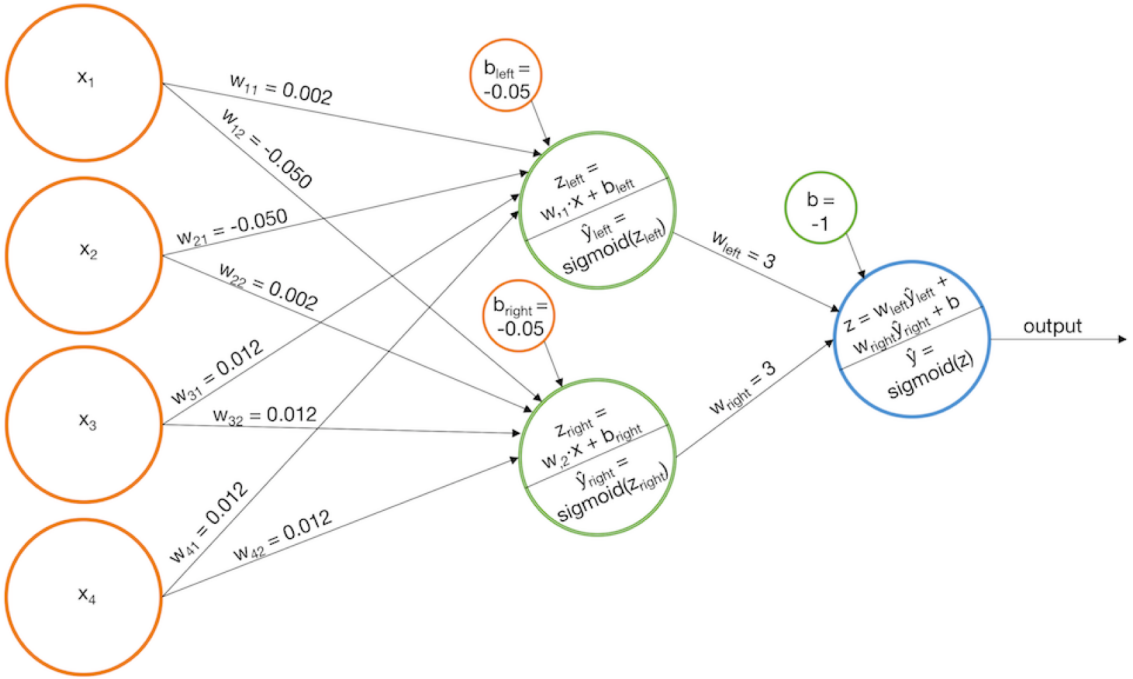
Multi-Layer Perceptron with Sigmoid activation function

- Can solve both of the above issues by adding an extra *layer* to the perceptron model.

Multi-Layer Perceptron with Sigmoid activation function

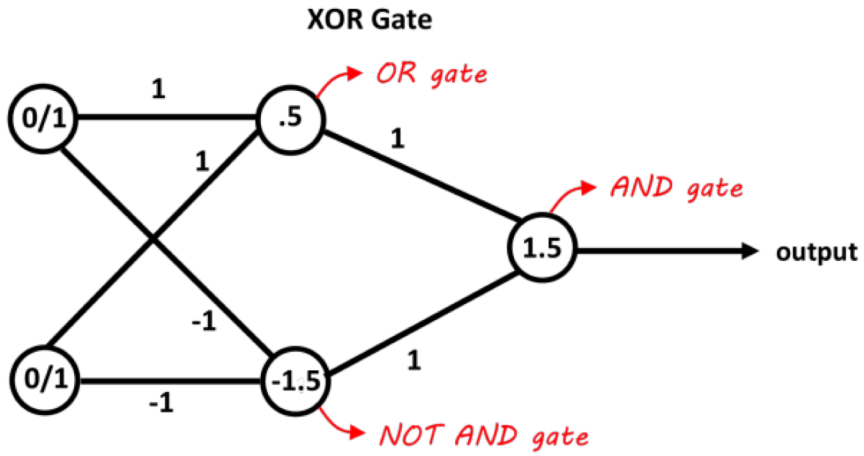
Example: Identify the stairs pattern

- Build a model that fires when “left stairs” are identified, yˆleft
- Build a model that fires when “right stairs” are identified, yˆright
- Add the score of the base models so that the final sigmoid function only fires if **both** yˆleft and yˆright are large



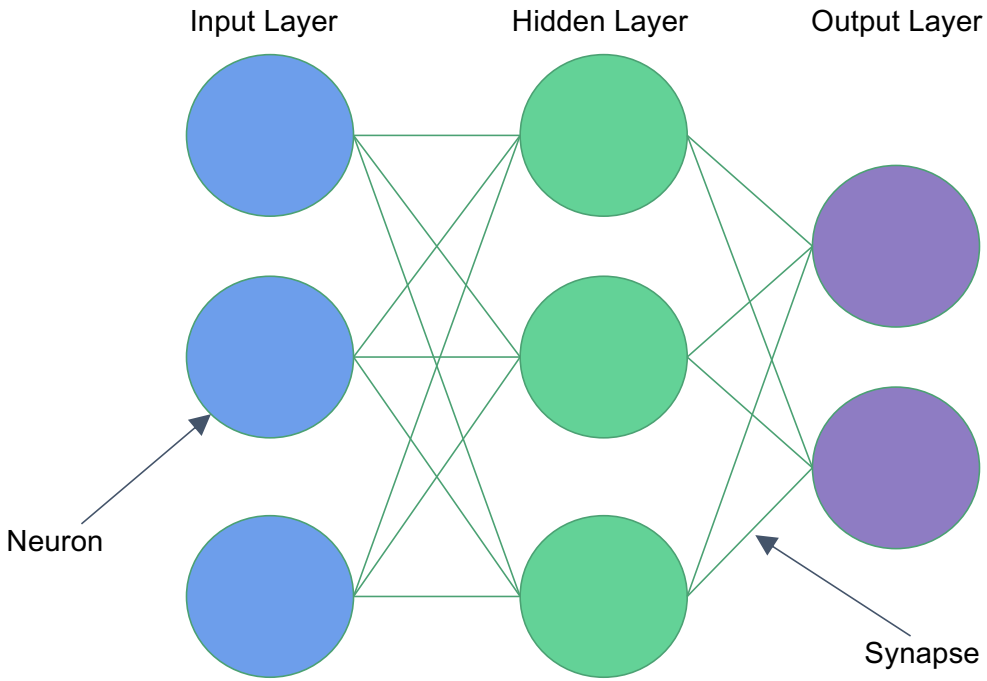
XOR Problem – Minsky paper

- Can’t solve with a single layer
- Exercise: create single node networks for OR and AND. Try to create single node network for XOR. Create and verify solution with 2 layers.

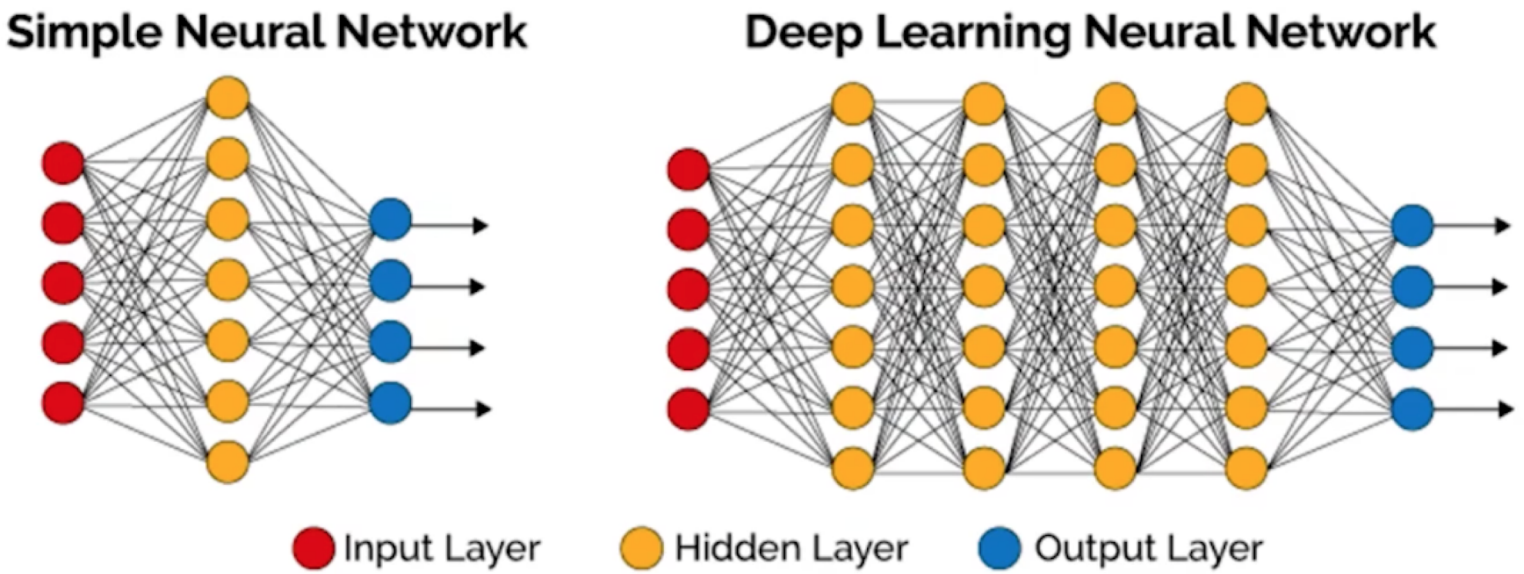


Neural Network Architecture

Multi-layer Perceptron

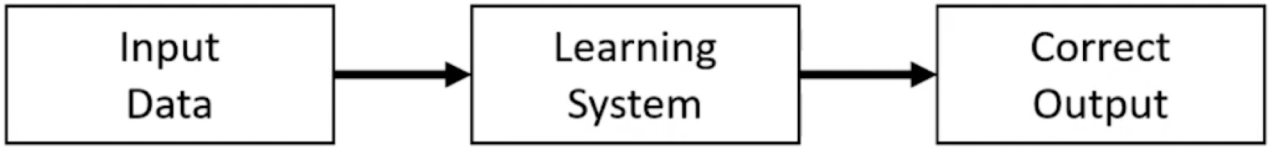


Combing Neurons in Hidden Layers:
The "Emergent" Power to Approximate



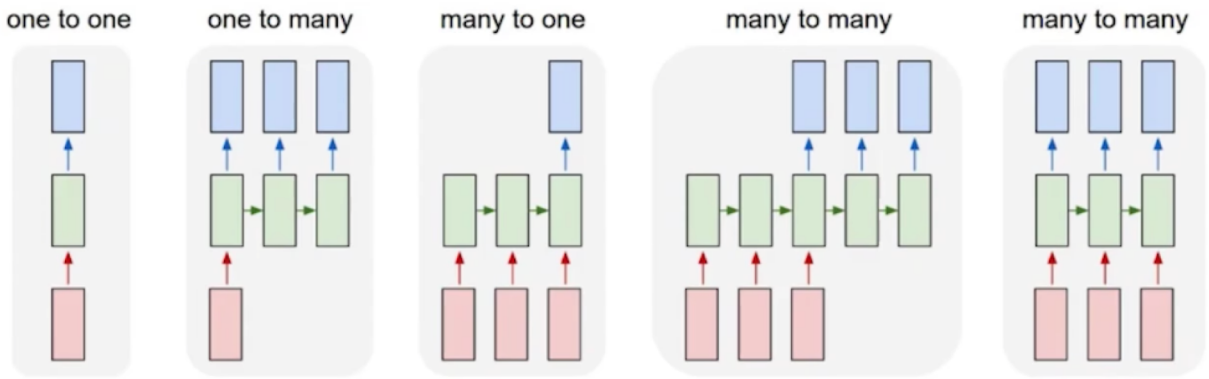
Universality: For any arbitrary function f(x), there exists a neural network that closely approximate it for any input x

What can we do with Deep Learning?



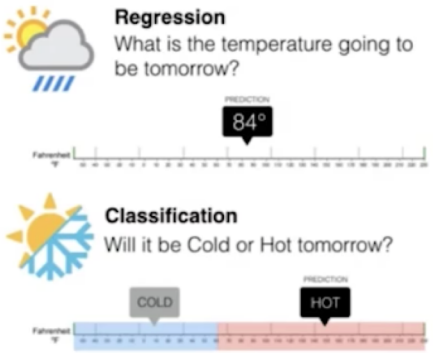
- Number
- Vector of numbers
- Sequence of numbers
- Sequence of vectors of numbers

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



Andrej Karpathy

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

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

Prediction

Ground Truth

Cross Entropy Loss

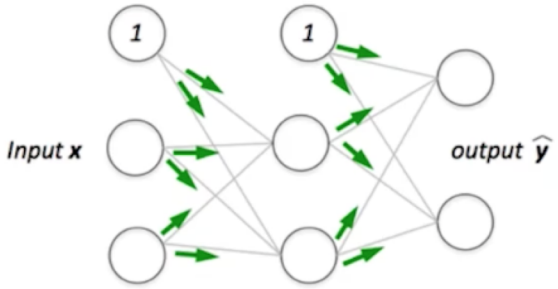
$$CE = -\sum_{i}^{C} t_i log(s_i)$$

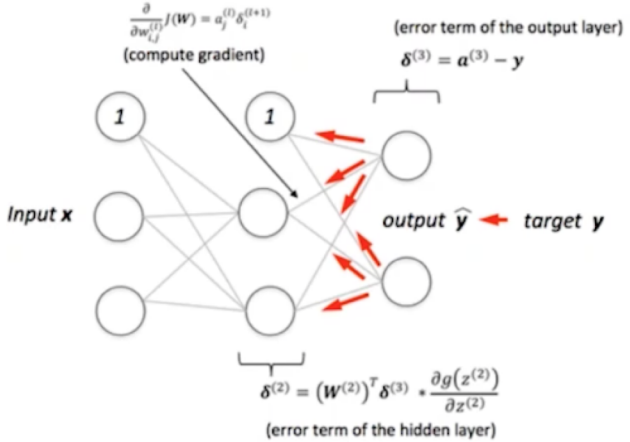
Classes

Prediction

Ground Truth {0,1}

Backpropagation





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

Subtasks:

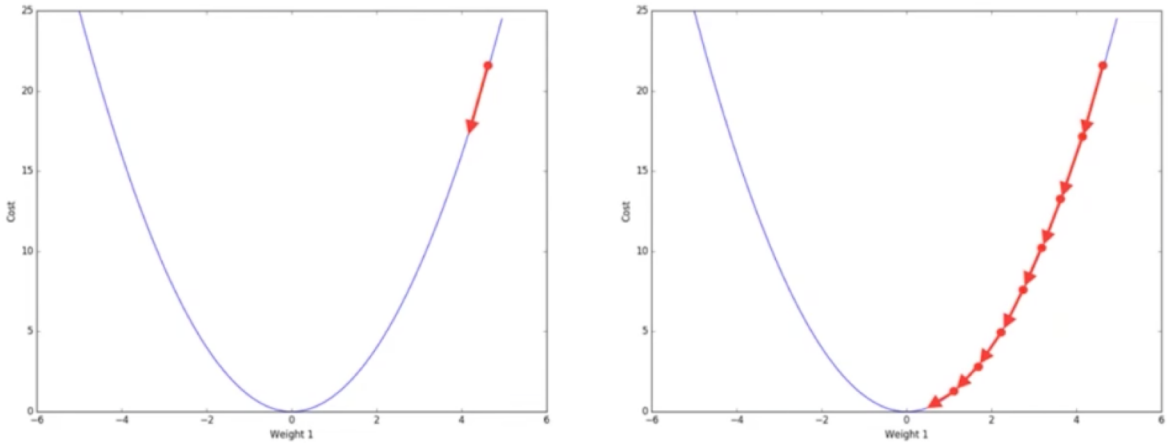
1. Forward pass to compute network output and "error"
2. Backward pass to compute gradients
3. A fraction of the weight's gradient is subtracted from the weight.

↑ Learning Rate

Numerical Method: Automatic Differentiation

Learning is an Optimization Problem

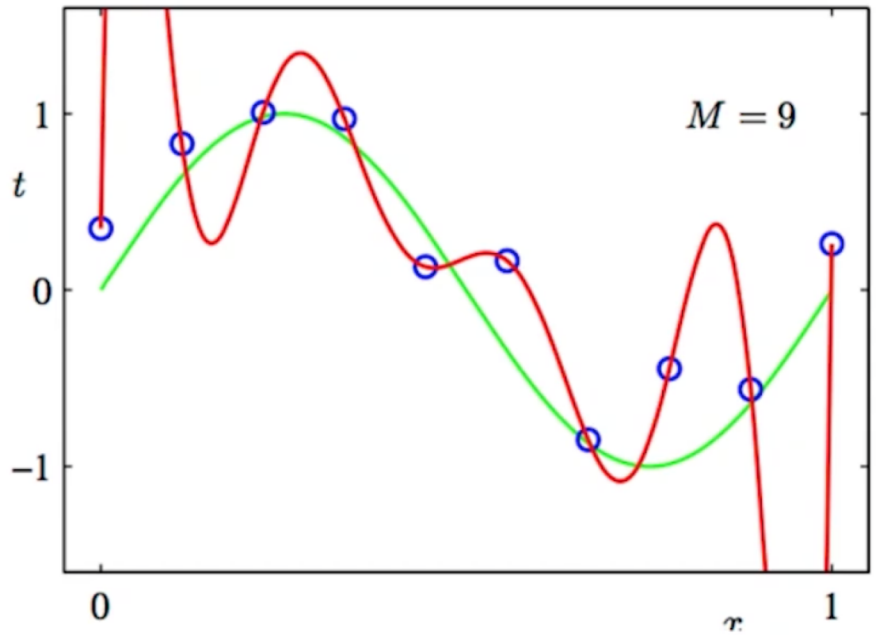
Task: Update the **weights** and **biases** to decrease **loss function**



SGD: Stochastic Gradient Descent

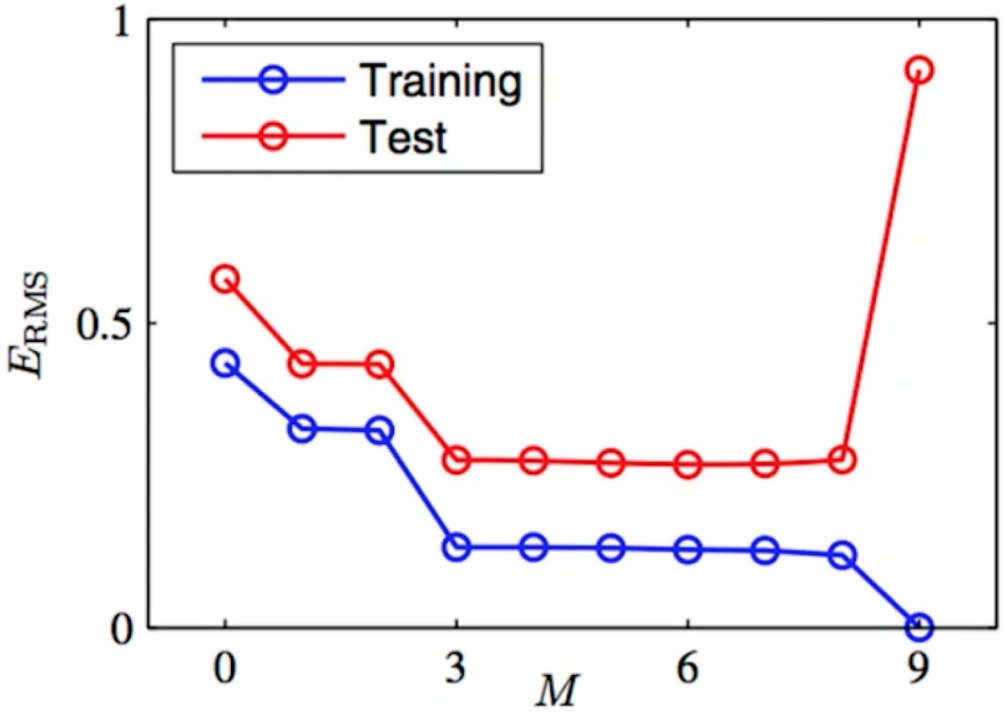
Overfitting and Regularization

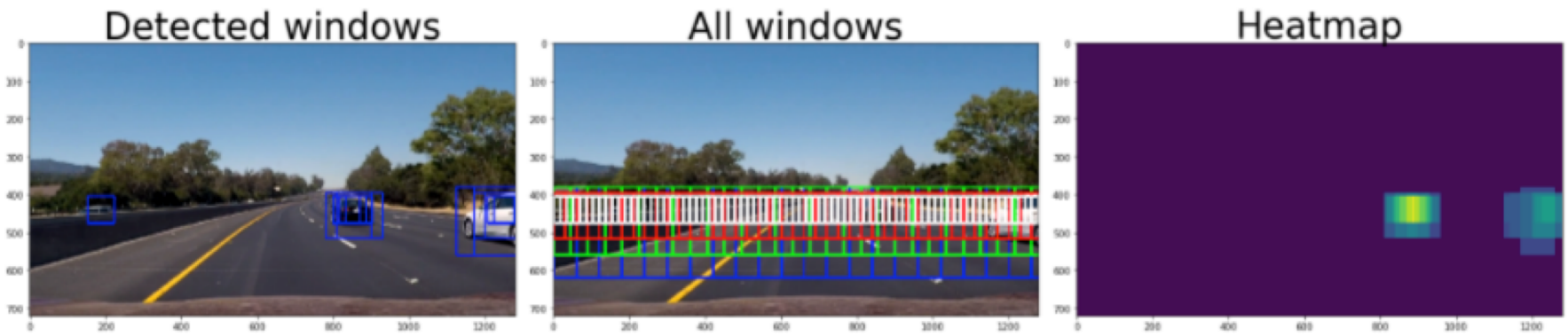
- Help the network **generalize** to data it hasn't seen.
- Big problem for **small datasets**.
- Overfitting example (a sine curve vs 9-degree polynomial):



Overfitting and Regularization

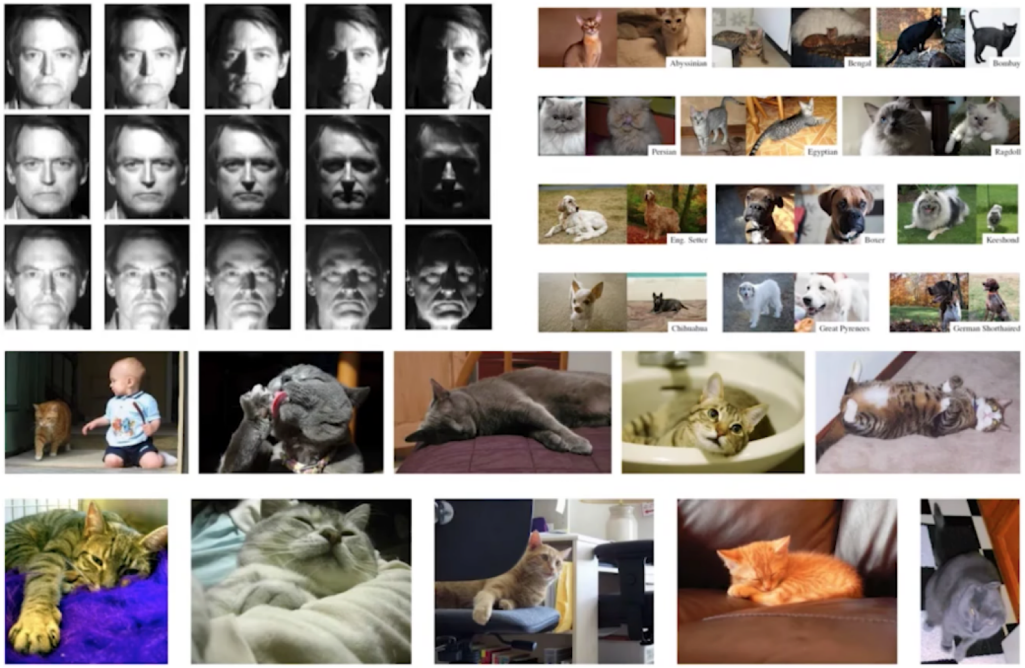
- Overfitting: The error decreases in the training set but increases in the test set.





Systems are currently good at basic prediction problems - deep learning + supervised data.

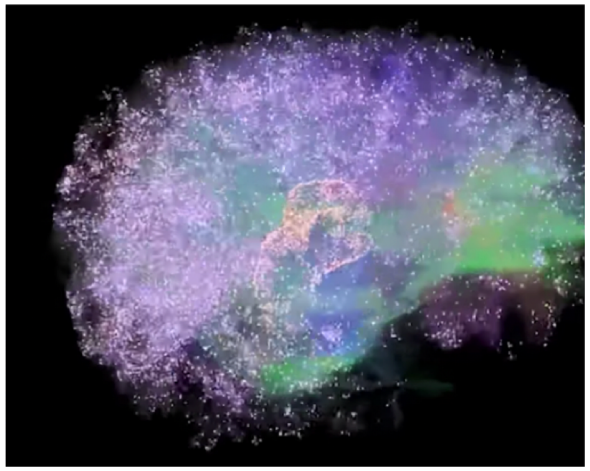
Pure perception, i.e., scene understanding is very hard.
Identifying objects in an image is good, but understanding how they relate is hard



nVidia

Deep Learning:

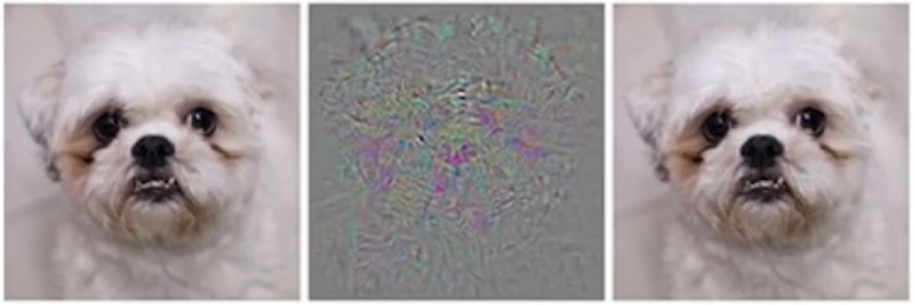
Our intuition about what's "hard" is flawed (in complicated ways)



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**

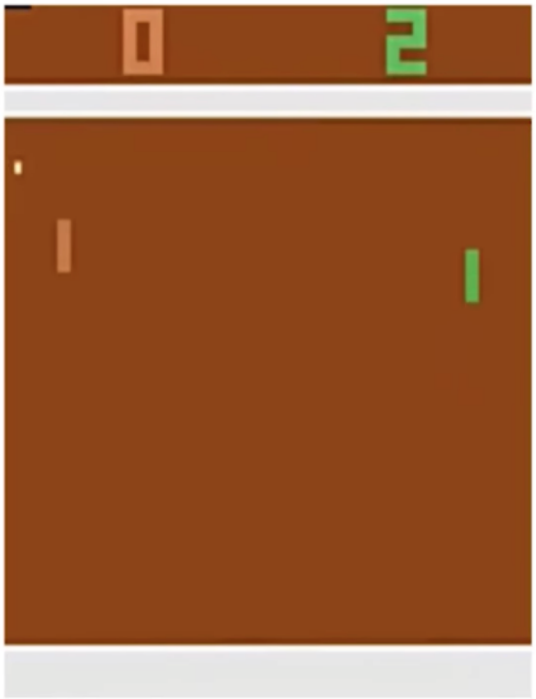
Generative Models may help

"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)*

STOP

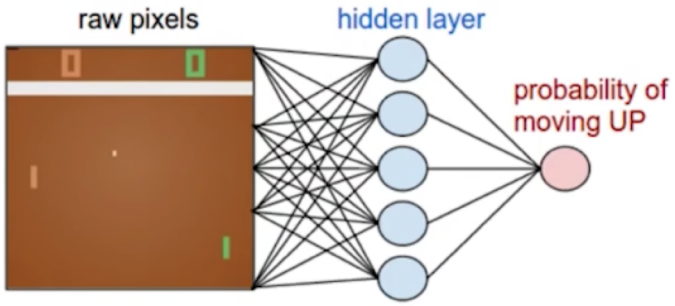
(Toward) General Purpose Intelligence:

Pong to Pixels



Andrej Karpathy. "Deep Reinforcement Learning: Pong from Pixels." 2016.

Policy Network:



- 80x80 image (difference image)
- 2 actions: up or down
- 200,000 Pong games

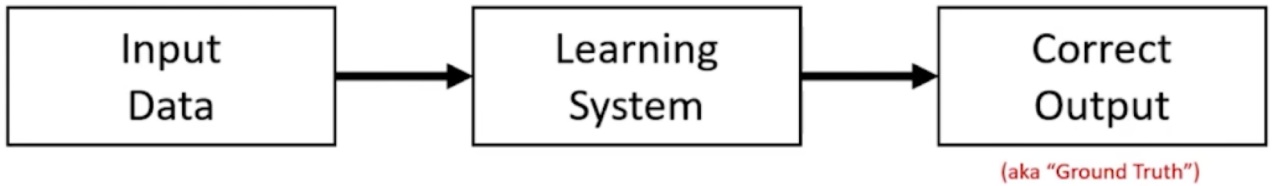
This is a step towards general purpose artificial intelligence!

Challenge for Deep Learning: Efficient Learning, One shot learning

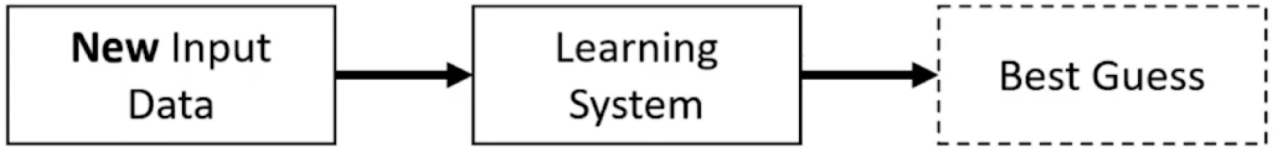
- Humans can learn from very few examples.
- Can learn by interacting with their environment
- Most machine learning models, especially deep learning models need tens of thousand of examples.

Deep Learning: Training and Testing

Training Stage:

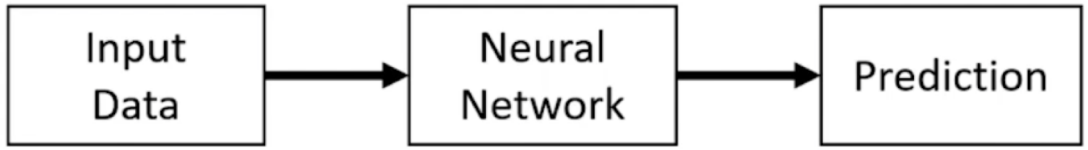


Testing Stage:

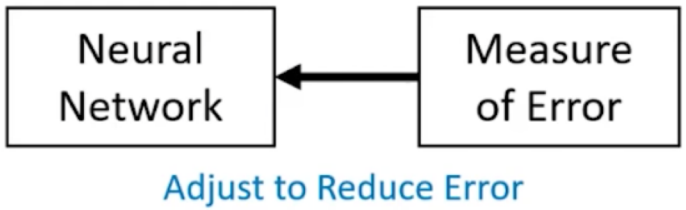


How Neural Networks Learn: Backpropagation

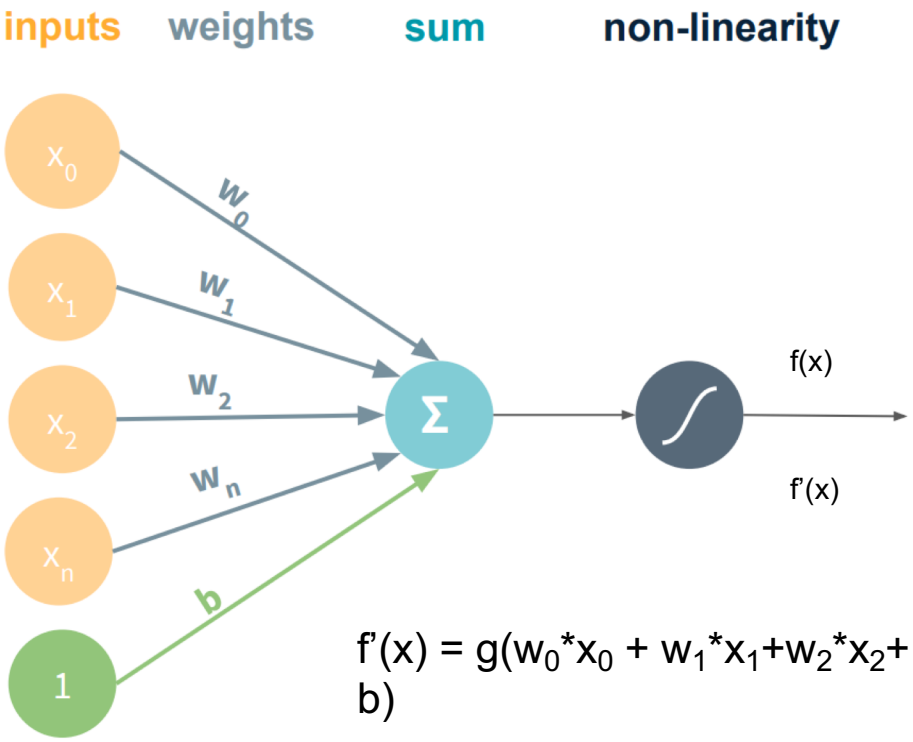
Forward Pass:



Backward Pass (aka Backpropagation):



Perceptron



$f'(x) = g(w_0*x_0 + w_1*x_1+w_2*x_2+\ldots+w_n*x_n + b)$

g(x) – non-linear activation function

Perceptron

$$output = g($$

(2*0.1) +

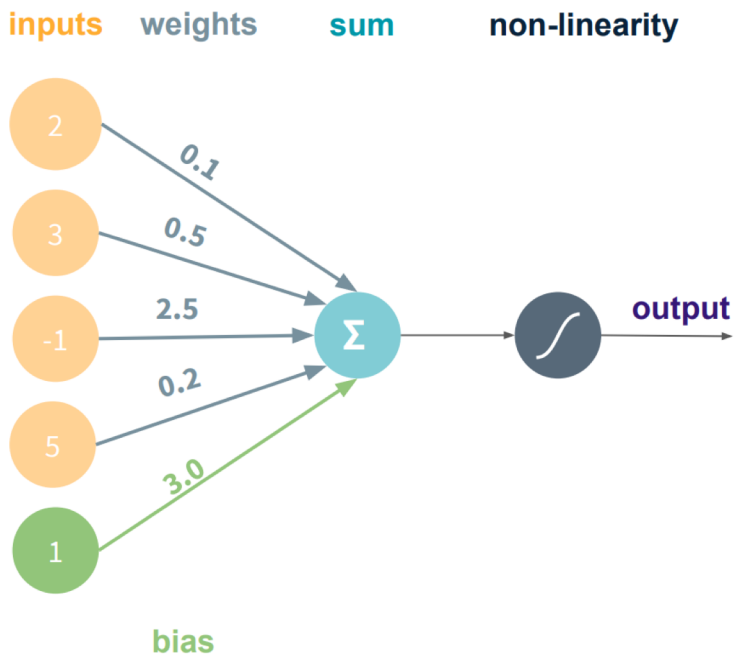
(3*0.5) +

(-1*2.5) +

(5*0.2) +

(1*3.0)

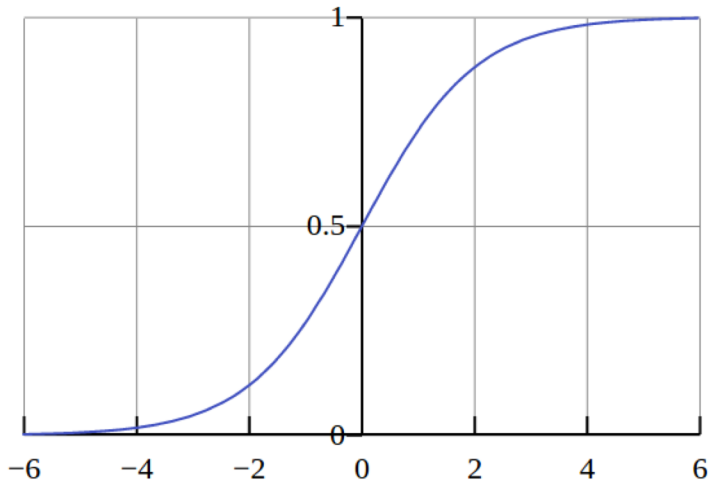
)



Activation Functions

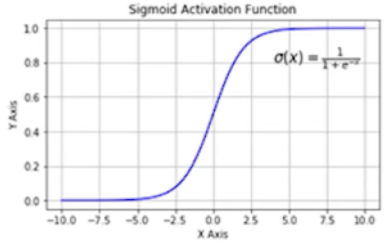
- Activation Functions are applied to the inputs at each neuron
  - A common activation function is the Sigmoid

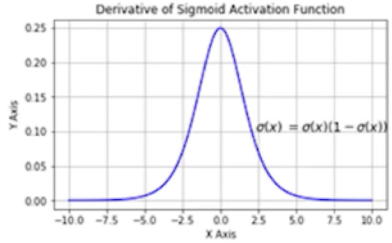
$$S(t) = \frac{1}{1 + e^{-t}}$$



Key Concepts:

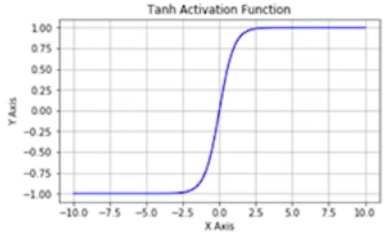
Activation Functions

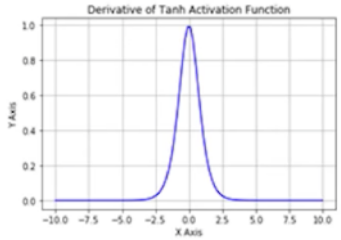




Sigmoid

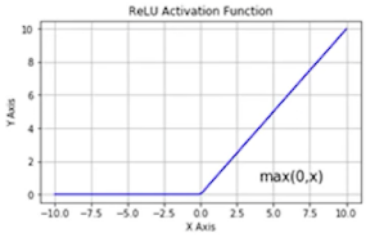
- Vanishing gradients
- Not zero centered

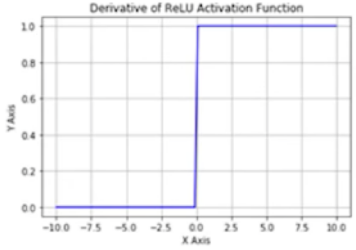




Tanh

- Vanishing gradients

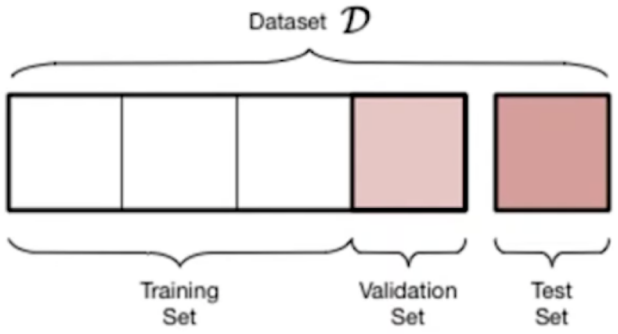


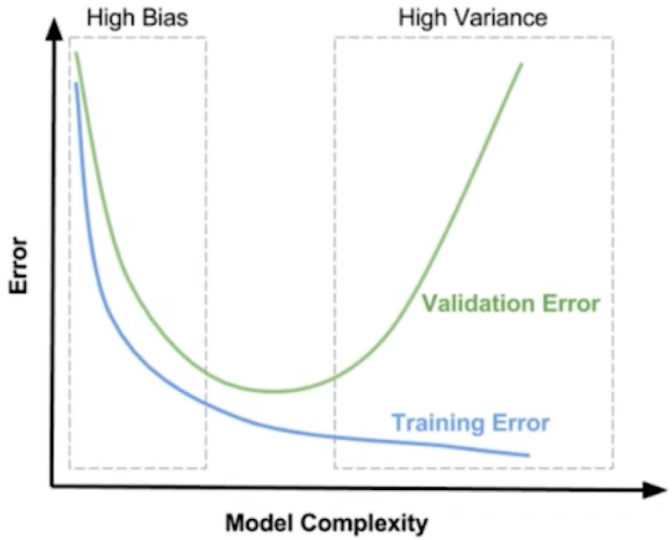


ReLU

- Not zero centered

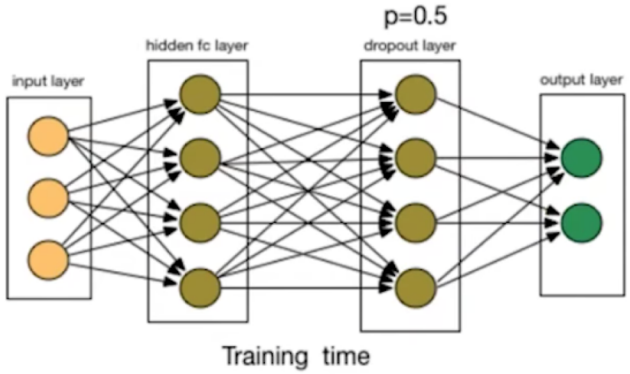
Regularization: Early Stoppage





- 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

Regularization: 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

Normalization

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

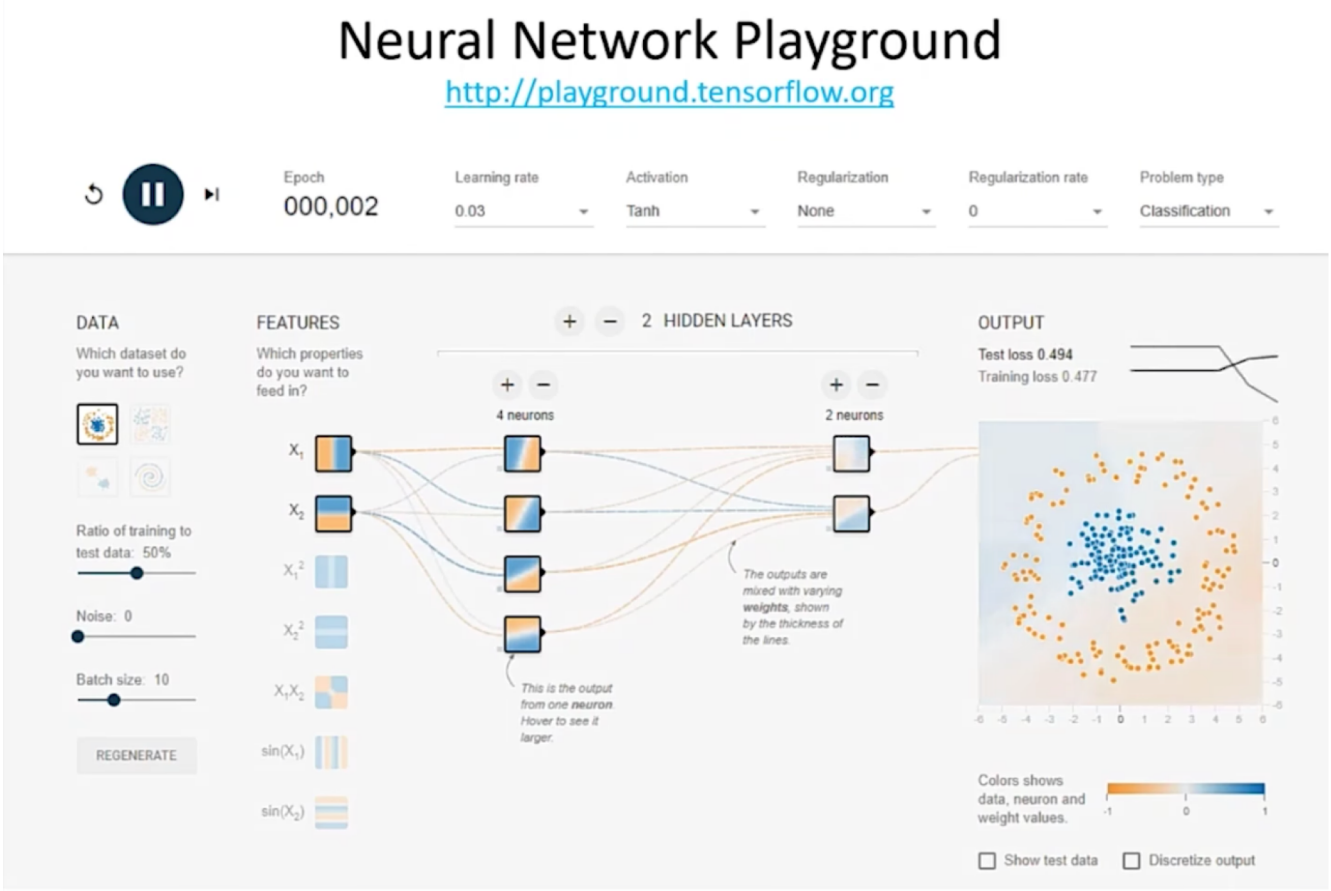
- Batch Normalization (BatchNorm, BN)
  - Normalize hidden layer inputs to mini-batch mean & variance
  - Reduces impact of earlier layers on later layers

- 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



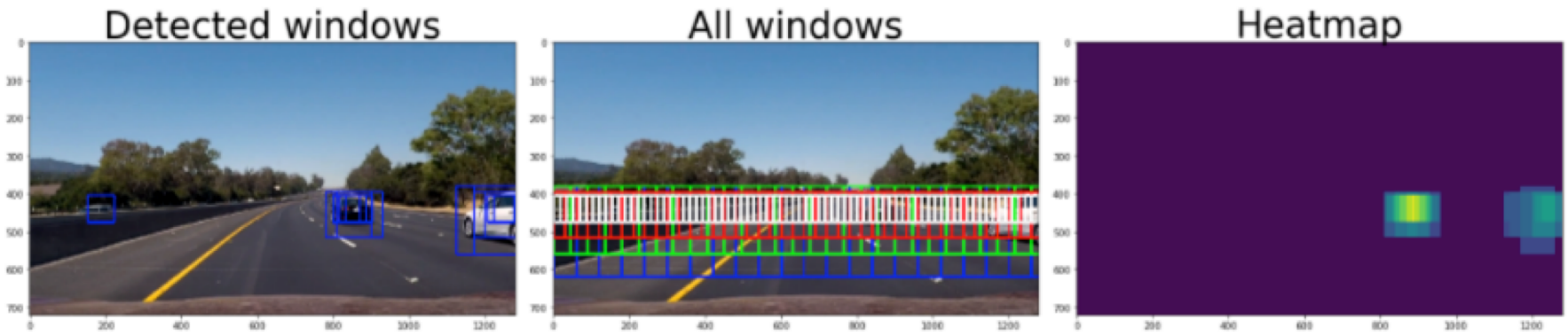
Summary Rationale for Deep Learning

Costs of acquiring and storing large quantities of heterogeneous data have dropped significantly.

The ability to extract actionable knowledge from such datasets has become critical for companies to maintain a competitive advantage.

Significant improvements in deep learning algorithms enabled by GPU processing provide new application opportunities across industries.

Deep Learning has set new performance standards in many machine learning applications.


Detected windows
All windows
Heatmap

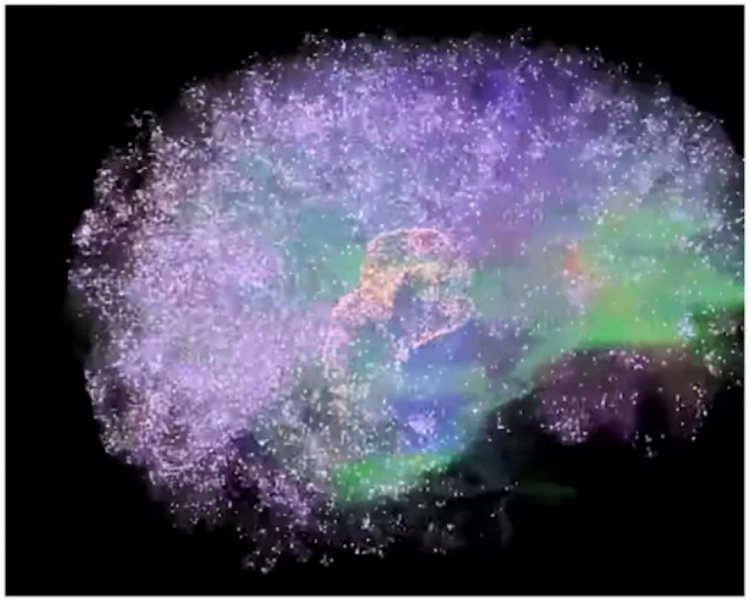
Visual Understanding is Harder

Examples of what we can't do well:

- Mirrors
- Sparse information
- 3D Structure
- Physics
- What's on peoples' minds?
- What happens next?
- Humor

Andrej Karpathy

Biological and Artificial Neural Networks



Human Brain

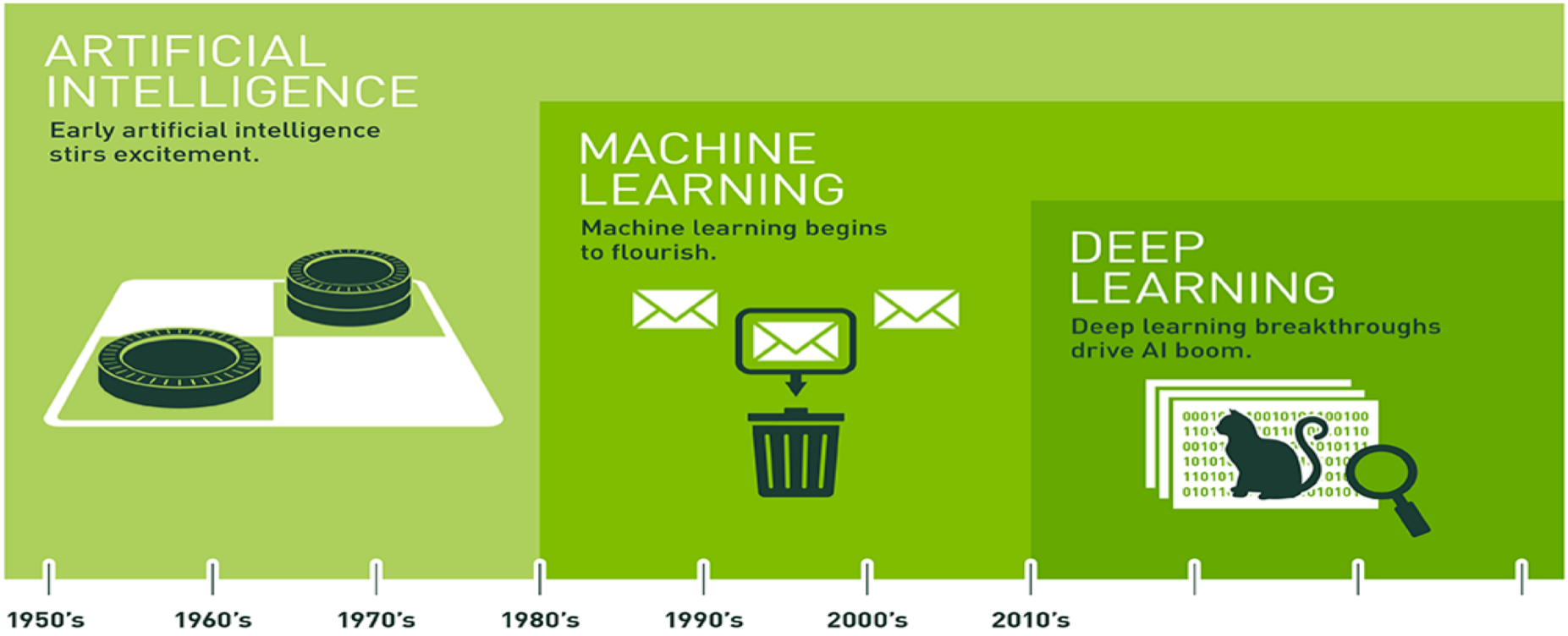
- Thalamocortical system:
  3 million neurons
  476 million synapses
- Full brain:
  100 billion neurons
  1,000 trillion synapses

Artificial Neural Network

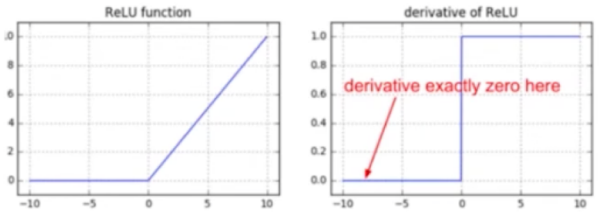
- ResNet-152:
  60 million synapses

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

ACCOMPLISHING COMPLEX GOALS

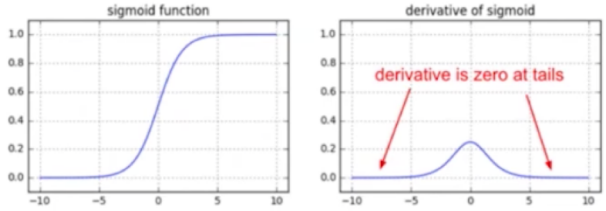


Dying ReLUs



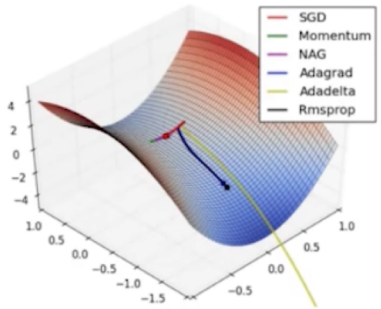
- If a neuron is initialized poorly, it might not fire for entire training dataset.
- Large parts of your network could be dead ReLUs!

Vanishing Gradients:

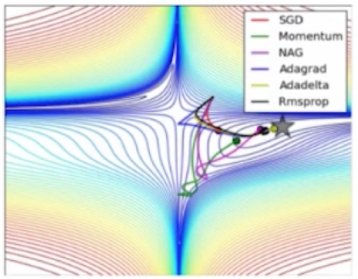


$$\frac{d\sigma(x)}{dx} = (1 - \sigma(x))\,\sigma(x)$$

Partial derivatives are small = Learning is slow



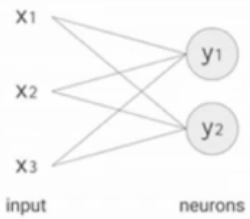
Hard to break symmetry



Vanilla SGD gets your there, but can be slow

Neural Networks are Parallelizable

Step 1



Step 2

Step 3

$$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3$$

Step 4

$$y_1 = f(w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

activation function

Step 5

$$y_1 = f(w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

$$y_2 = f(w_{21}x_1 + w_{22}x_2 + w_{23}x_3)$$

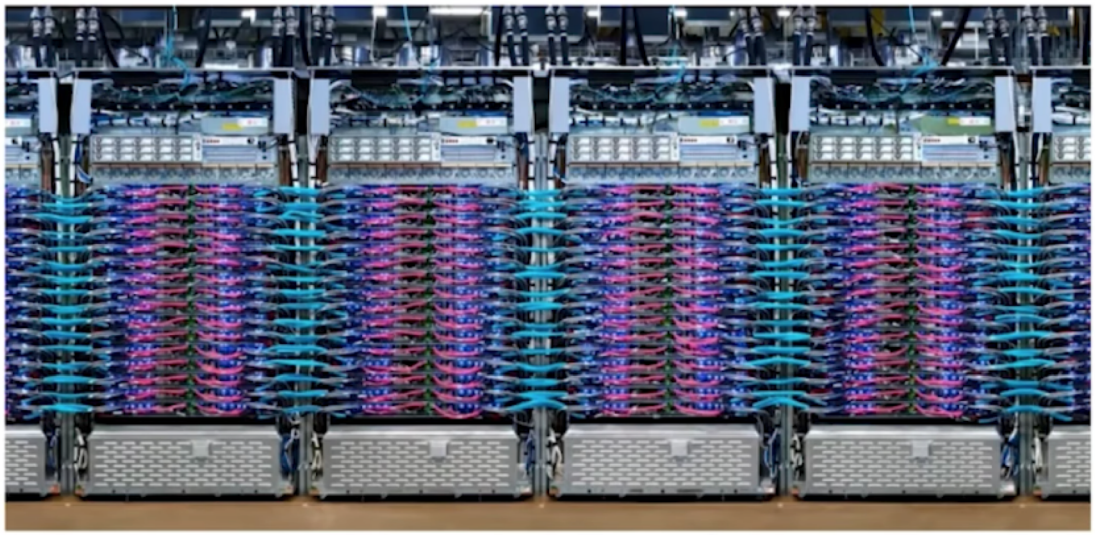
Animated

$$y_1 = f(w_{11}x_1 + w_{12}x_2 + w_{13}x_3)$$

activation function

Compute Hardware

- **CPU** – serial, general purpose, everyone has one
- **GPU** – parallelizable, still general purpose
- **TPU** – custom ASIC (Application-Specific Integrated Circuit) by Google, specialized for machine learning, low precision



DEEP LEARNING iN PRODUCTION

Speech Recognition

Recommender Systems

Autonomous Driving

Real-time Object Recognition

Robotics

Real-time Language Translation

Many More…