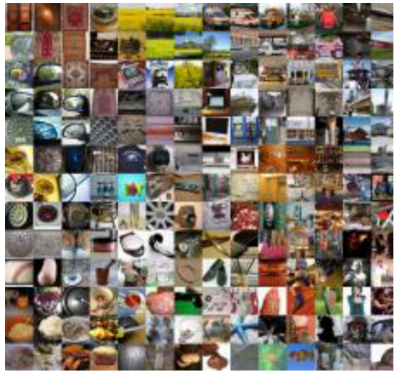
Lecture 22: Introduction to Deep Neural Networks

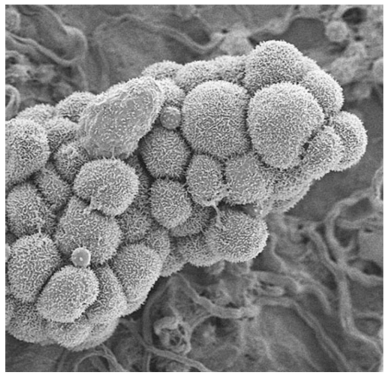
COMP90049
Knowledge Technology

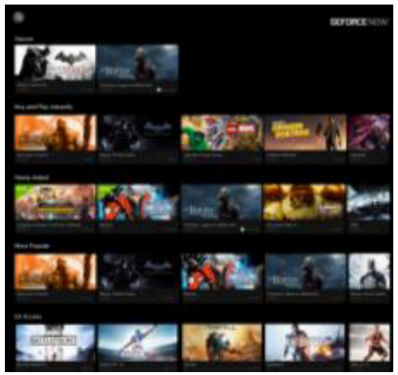
Sarah Erfani, CIS

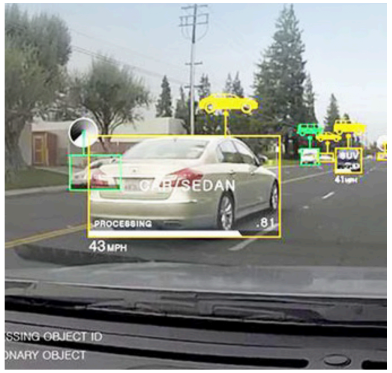
Semester 2, 2017

Deep Learning is Everywhere









INTERNET & CLOUD

Image Classification
Speech Recognition
Language Translation
Language Processing
Sentiment Analysis
Recommendation

MEDICINE & BIOLOGY

Cancer Cell Detection
Diabetic Grading
Drug Discovery

MEDIA & ENTERTAINMENT

Video Captioning
Video Search
Real Time Translation

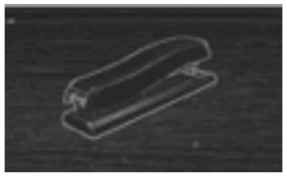
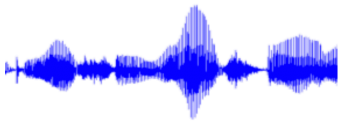
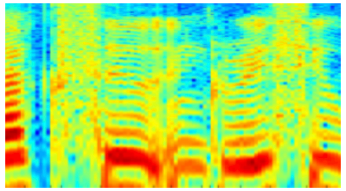
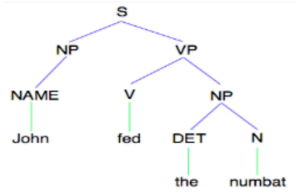
SECURITY & DEFENSE

Face Detection
Video Surveillance
Satellite Imagery

AUTONOMOUS MACHINES

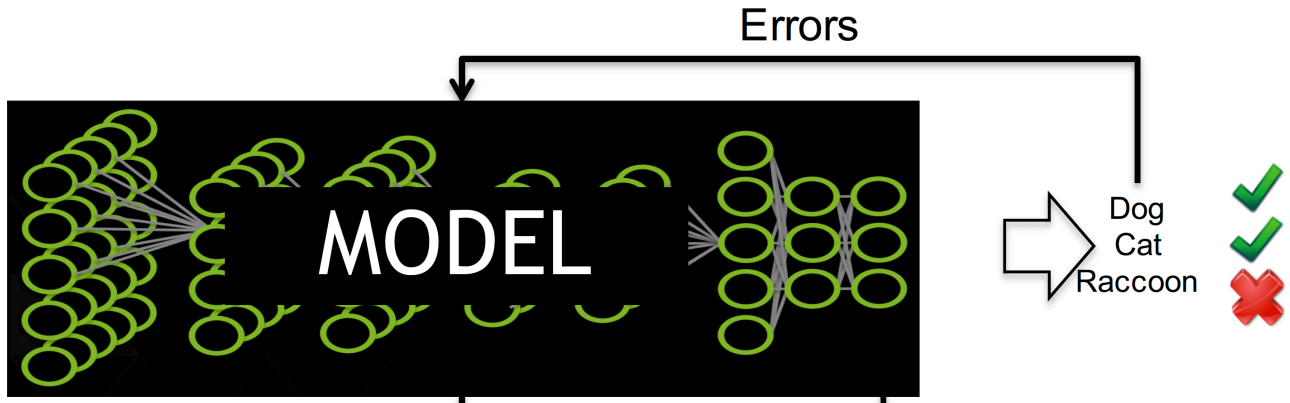
Pedestrian Detection
Lane Tracking
Recognize Traffic Sign

Traditional Machine Learning Perception

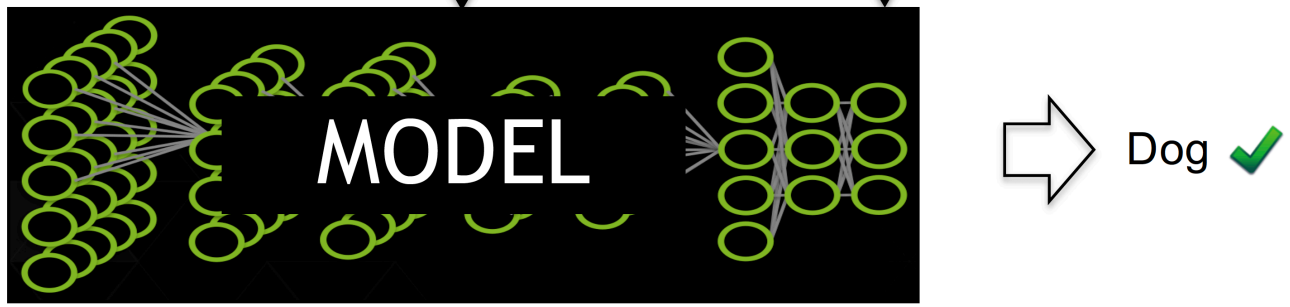
Raw data	Feature extraction	Classifier/ detector	Result
		SVM, shallow neural net, …	
		HMM, shallow neural net, …	Speaker ID, speech transcription, …
		Clustering, HMM, LDA, LSA …	Topic classification, machine translation, sentiment analysis…

Deep Learning Approach

Train:



Deploy:

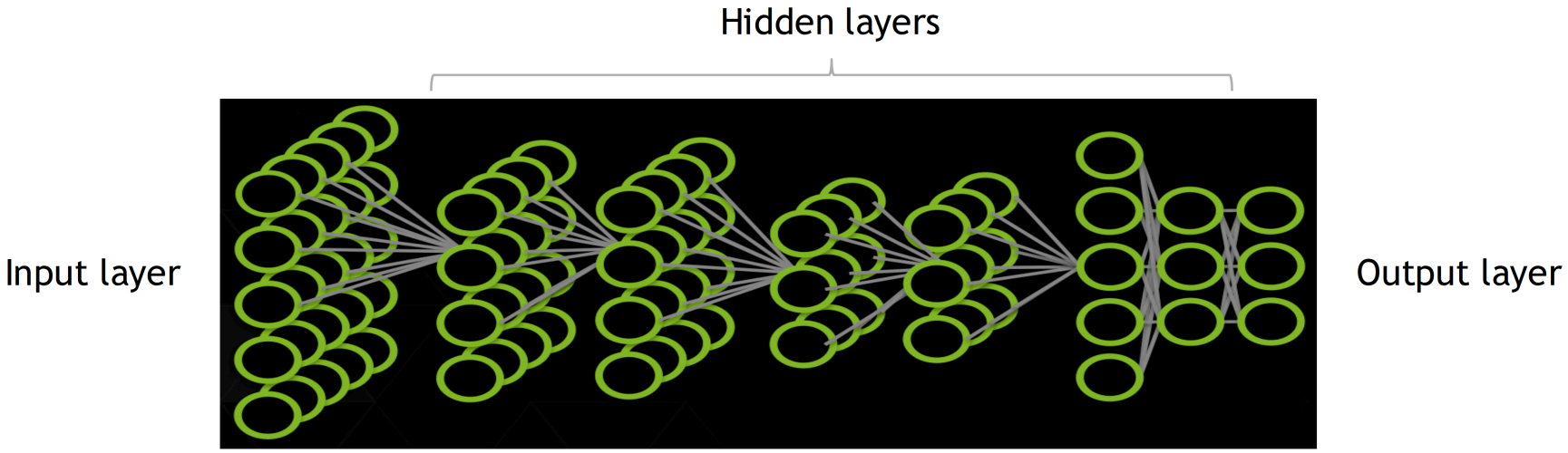


Deep Learning Benefits

- **Robust**
  - No need to design the features ahead of time – features are automatically learned to be optimal for the task at hand
  - Robustness to natural variations in the data is automatically learned
- **Generalizable**
  - The same neural net approach can be used for many different applications and data types
- **Scalable**
  - Performance improves with more data, method is massively parallelizable

Artificial Neural Networks

- A collection of simple, trainable mathematical units that collectively learn complex functions

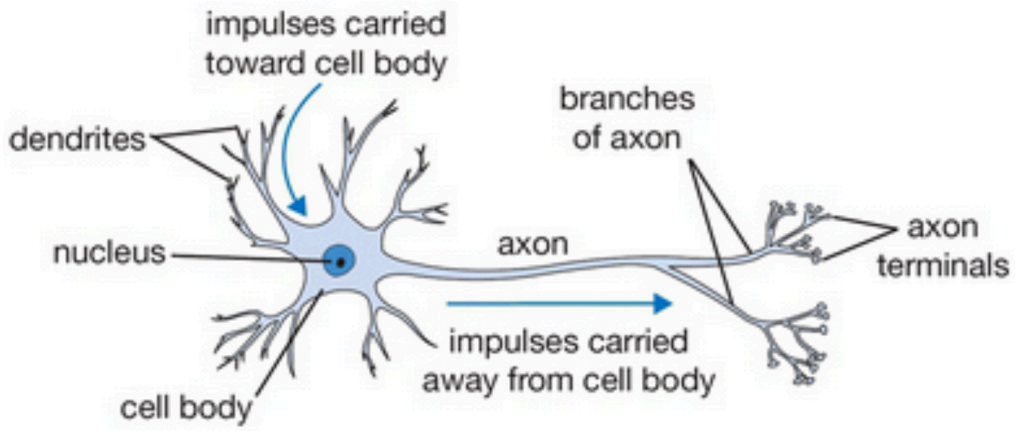


- Given sufficient training data an artificial neural network can approximate very complex functions mapping raw data to output decisions

Artificial Neural Networks

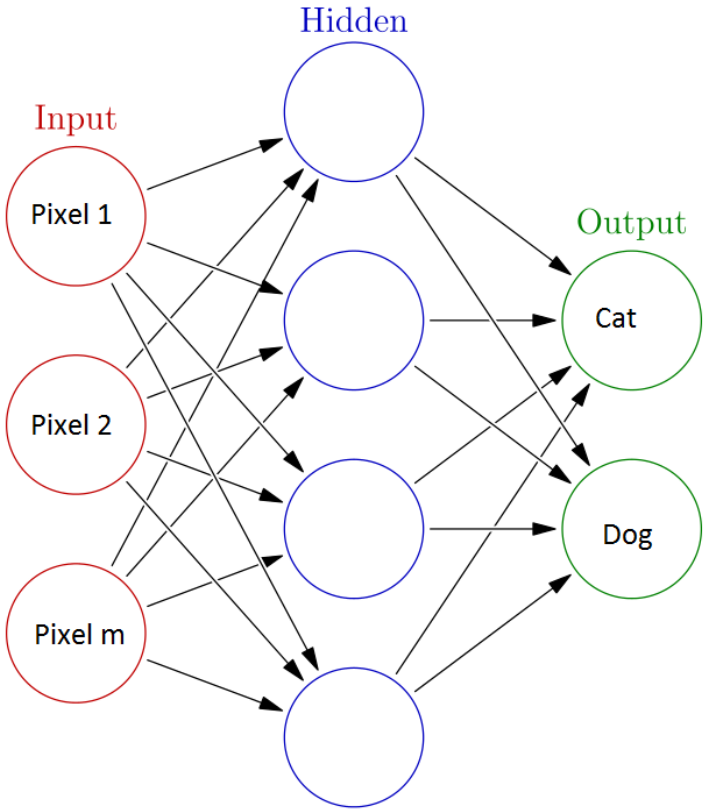
- The first artificial neural networks were invented in 1950's
- Inspired by the biological brain

Biological neuron

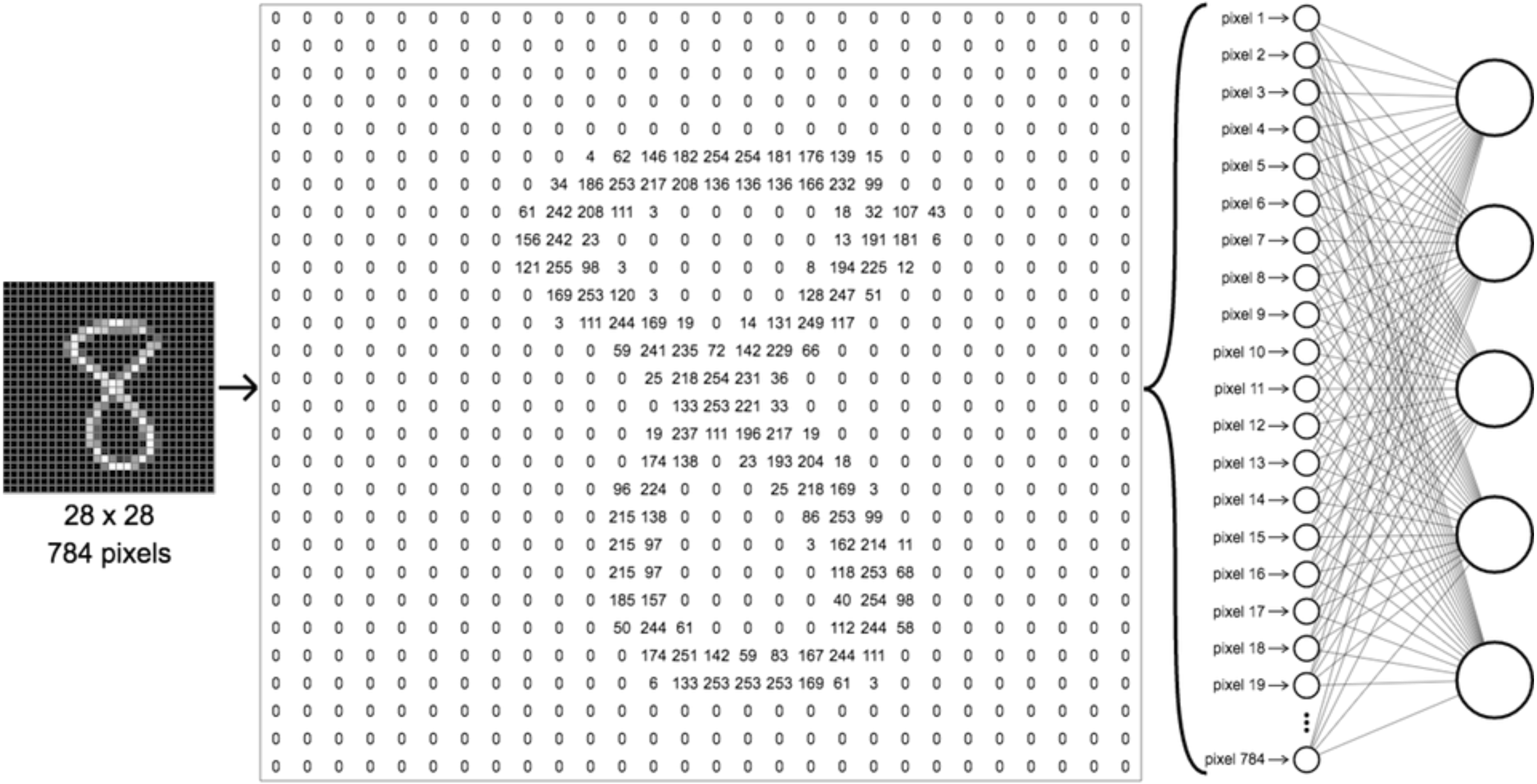


From Stanford cs231n lecture notes

Artificial Neural Networks

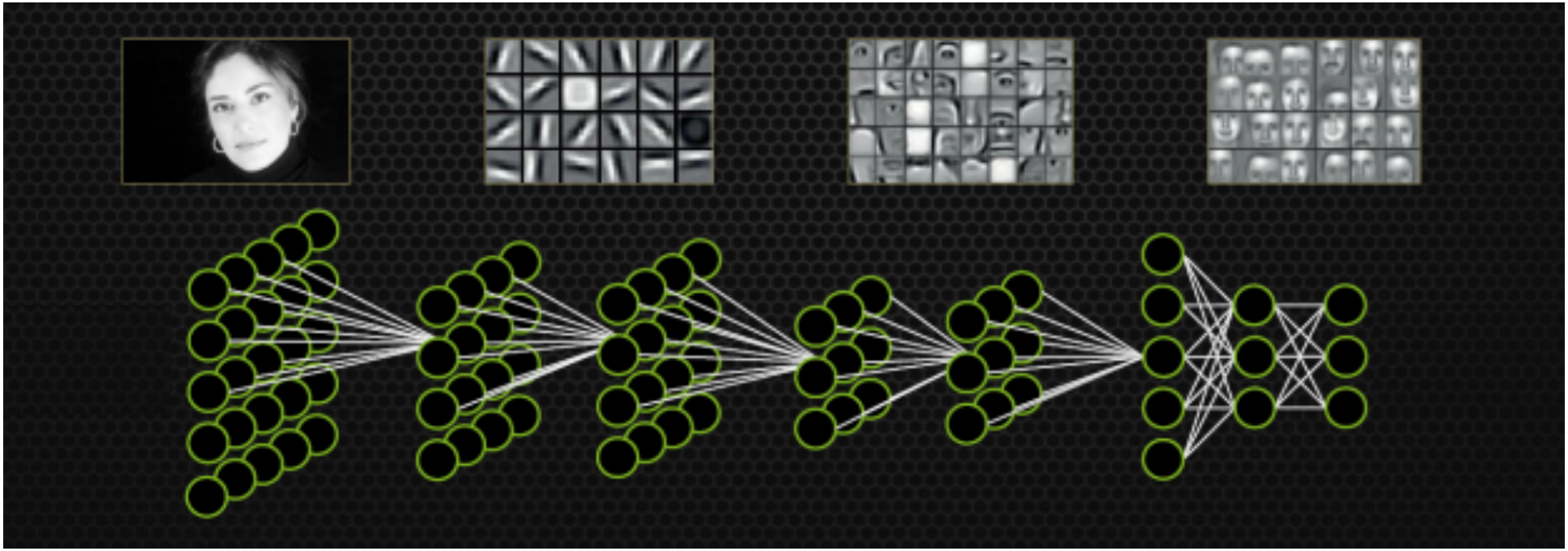


Artificial Neural Networks

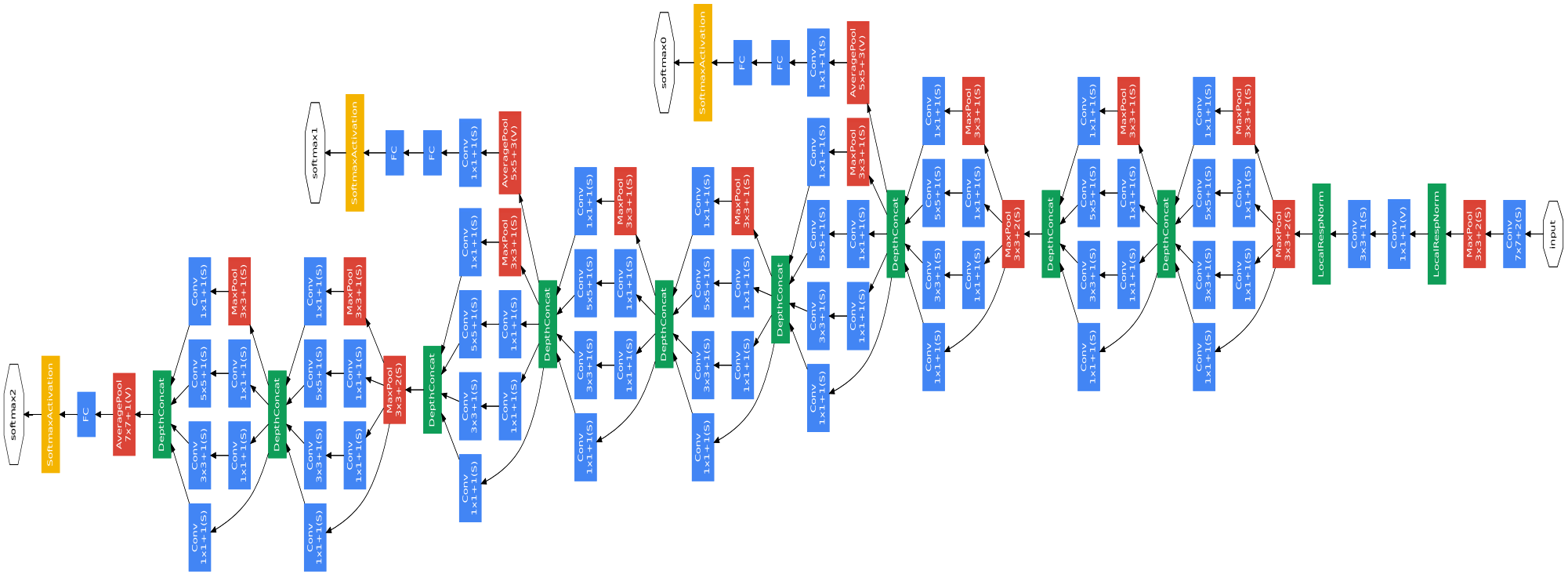


0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Artificial Neural Networks

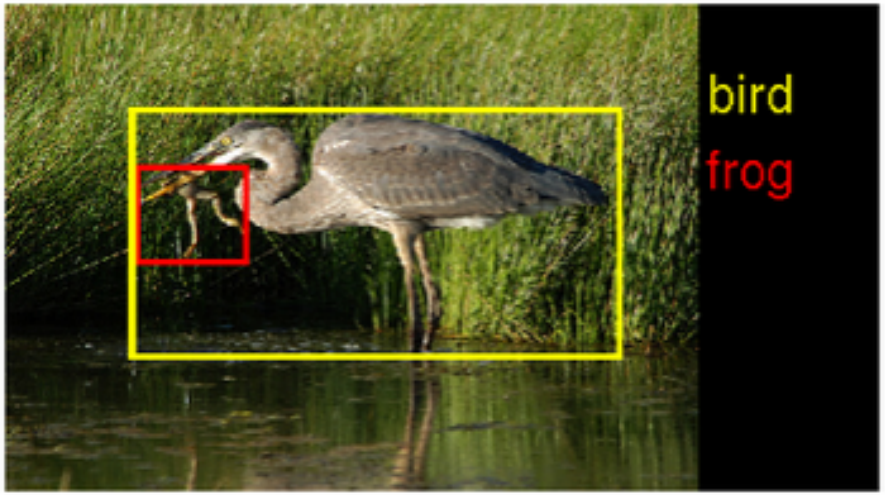


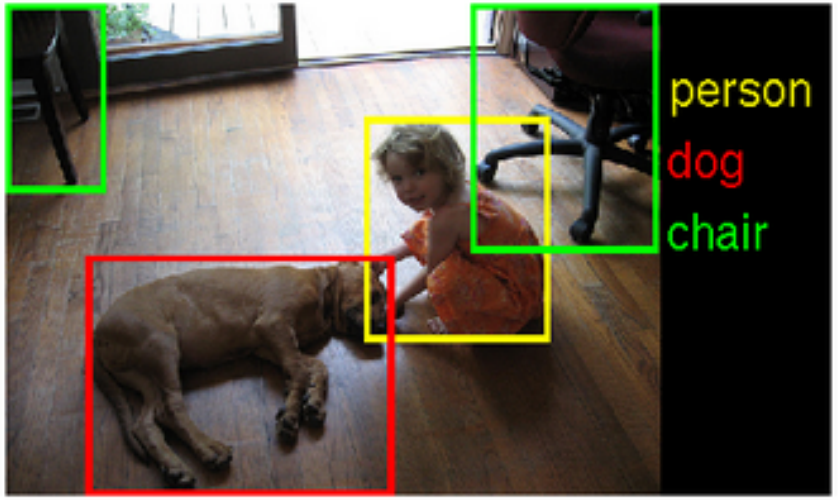
Example of Deep Learning Model

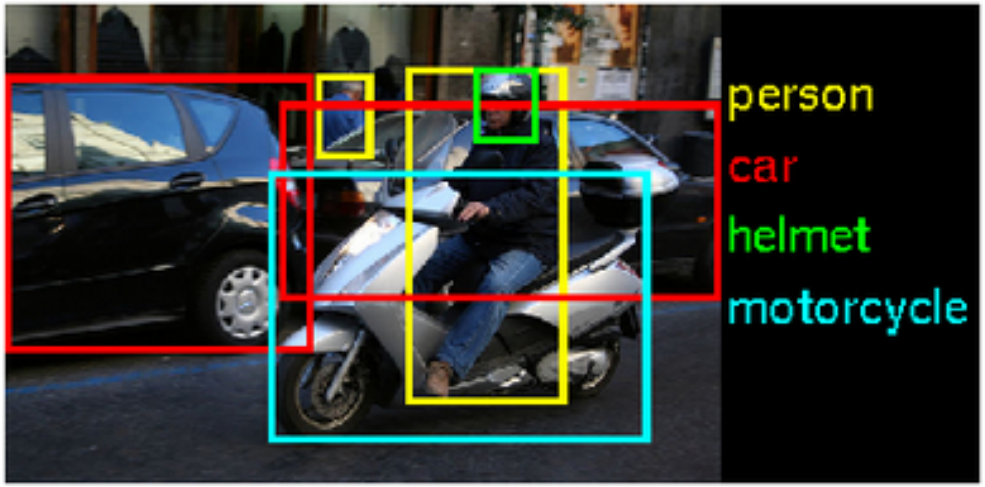


Google - Inception

Applications of DNNs





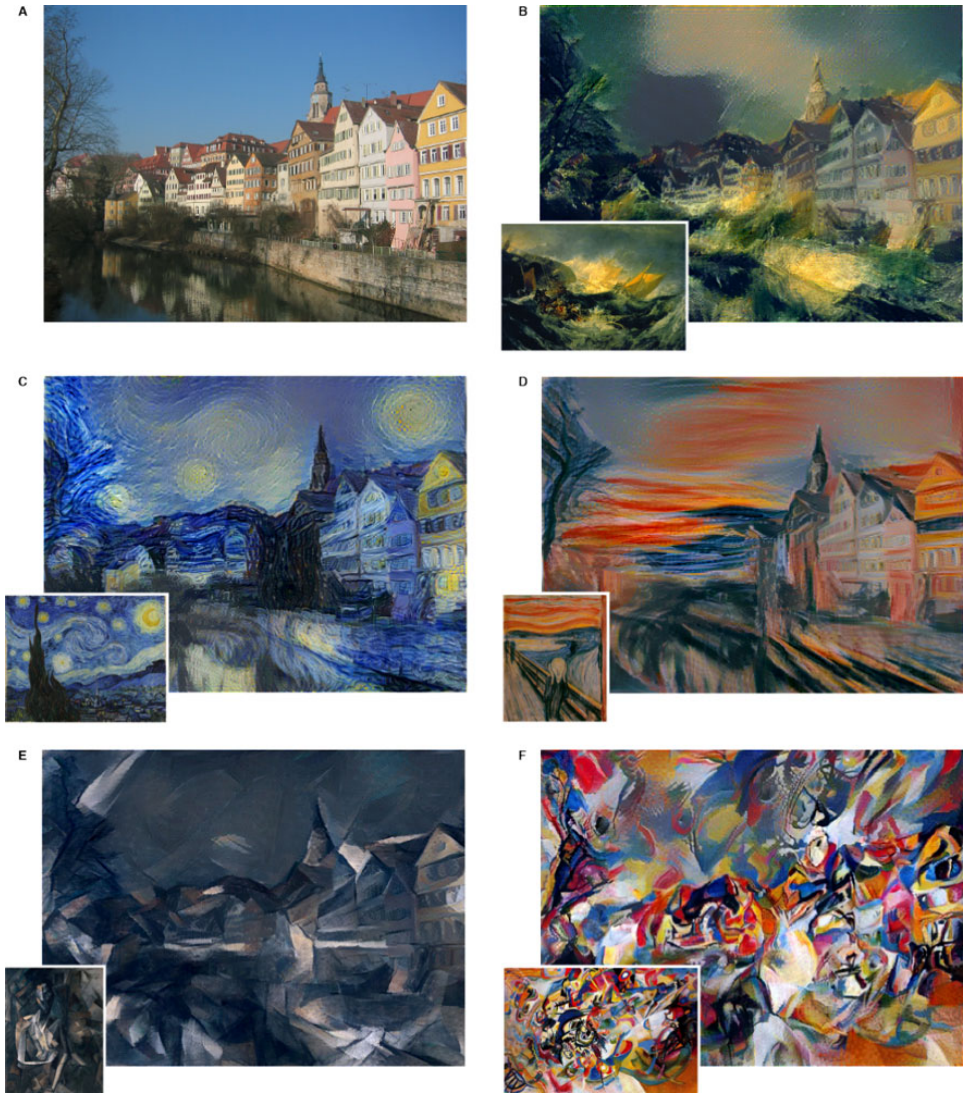


Applications of DNNs

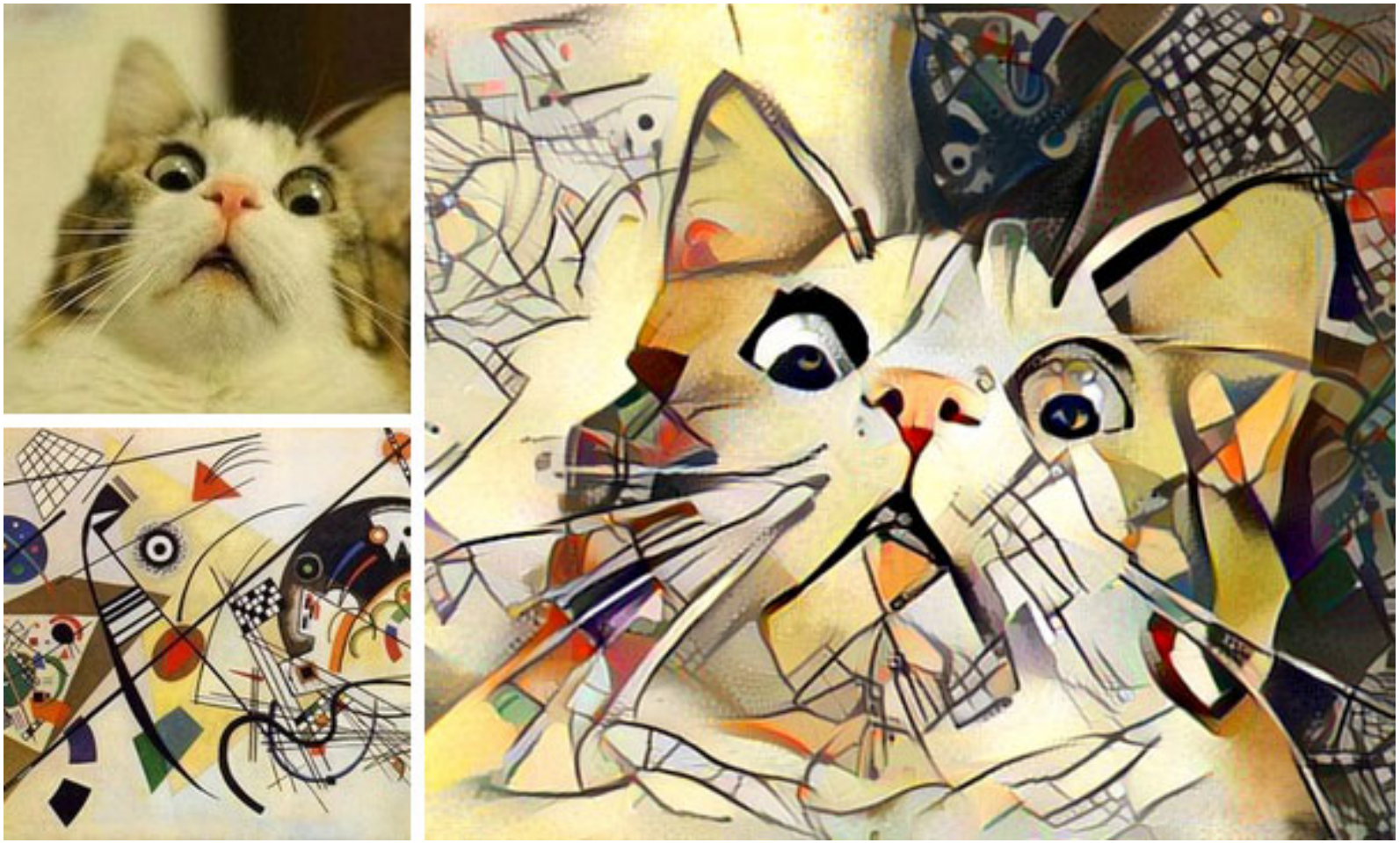
Applications of DNNs

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
 A person riding a motorcycle on a dirt road.	 Two dogs play in the grass.	 A skateboarder does a trick on a ramp.	 A dog is jumping to catch a frisbee.
 A group of young people playing a game of frisbee.	 Two hockey players are fighting over the puck.	 A little girl in a pink hat is blowing bubbles.	 A refrigerator filled with lots of food and drinks.

Applications of DNNs



Applications of DNNs



https://prisma-ai.com/

Applications of DNNs

SEED: Yes we can

Good morning. And as we mark the fact that they can stand with their companies that are consistent to the state of Pakistan and the United States of America.

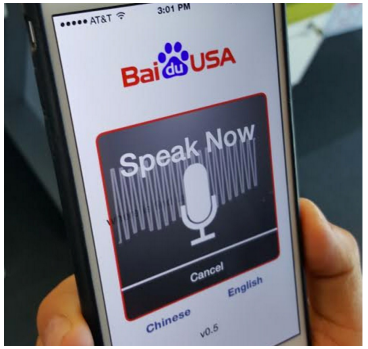
With the financial system we can do that. And the people of the United States will not be able to continue to support the people of the greatest problem of the American people to stay in the White House. And that's why I've got to recognize the private sector that there is no doubt that we've got to continue to shape the painful realisation that we are the United States of America. And these are the people of all Americans to be recognising the continent of the reason that they don't have to stay on the law.

It was the first time in Muslim Americans to see your own lives. And the United States will be better profitable and security and prosperity to the American people —she was a candidate for the same recession that they have to make sure that they take a state in the United States of America.

Thank you very much. God bless you. God bless you. God bless you. God bless you.

We (will) Lose on Many Specific Tasks

Speech Recognition



Baidu Deep Speech 2:

- End-to-end Deep Learning for English and Mandarin Speech Recognition
- English and Mandarin speech recognition Transition from English to Mandarin made simpler by end-to-end DL
- No feature engineering or Mandarin-specifics required
- More accurate than humans

Error rate 3.7% vs. 4% for human tests

http://svail.github.io/mandarin/

https://arxiv.org/pdf/1512.02595.pdf

We (will) Lose on Many Specific Tasks

Strategic Games

AlphaGo:

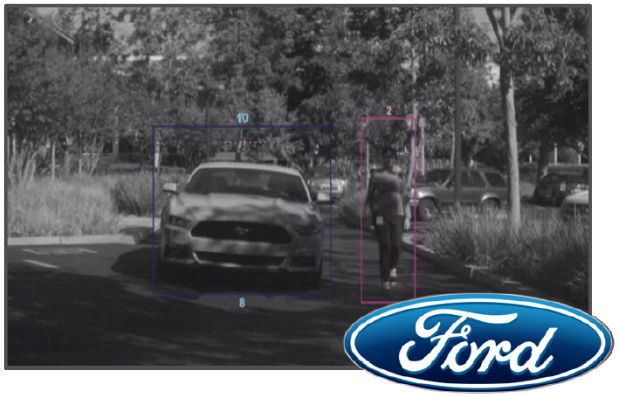
- First Computer Program to Beat a Human Go Professional
- Training DNNs: 3 weeks, 340 million training steps on 50 GPUs
- Play: Asynchronous multi-threaded search
- Simulations on CPUs, policy and value DNNs in parallel on GPUs
- Single machine: 40 search threads, 48 CPUs, and 8 GPUs
- Distributed version: 40 search threads, 1202 CPUs and 176 GPUs
- Outcome: Beat both European and World Go champions in best of 5 matches

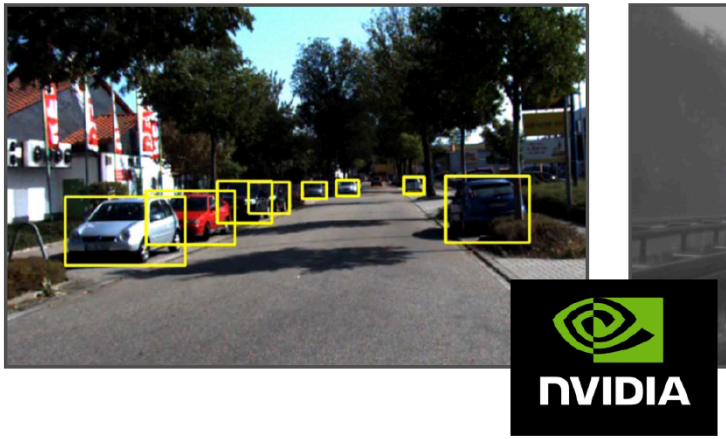
http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html

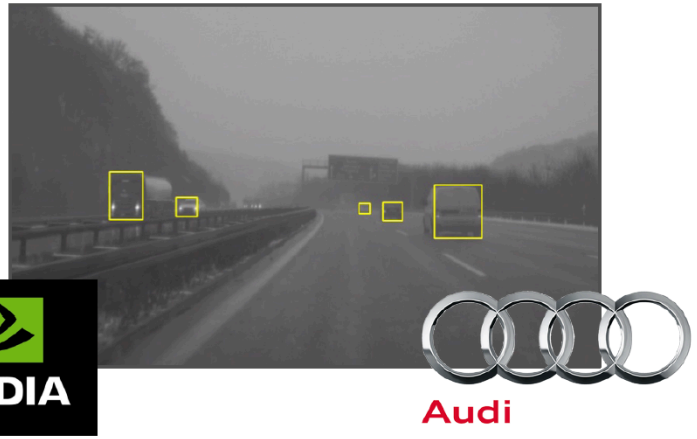
http://deepmind.com/alpha-go.html

We (will) Lose on Many Specific Tasks

Autonomous Vehicles







We (will) Lose on Many Specific Tasks

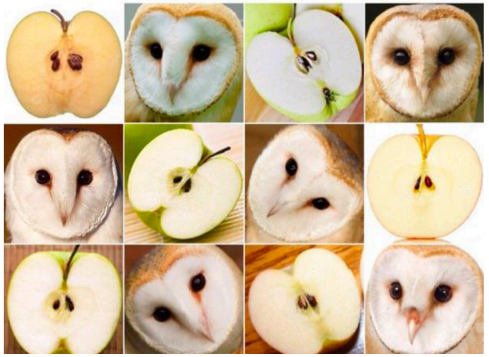
Image Recognition

GoogLeNet: http://cs.stanford.edu/people/karpathy/ilsvrc/

Labrapoodle or Fried chicken

Sheepdog or Mop

Parrot or Guacamole



Barn owl or Apple

Raw chicken or Donald Trump

Some Examples

- Fake Obama Speech

https://www.youtube.com/watch?v=AmUC4m6w1wo

- Generating music:

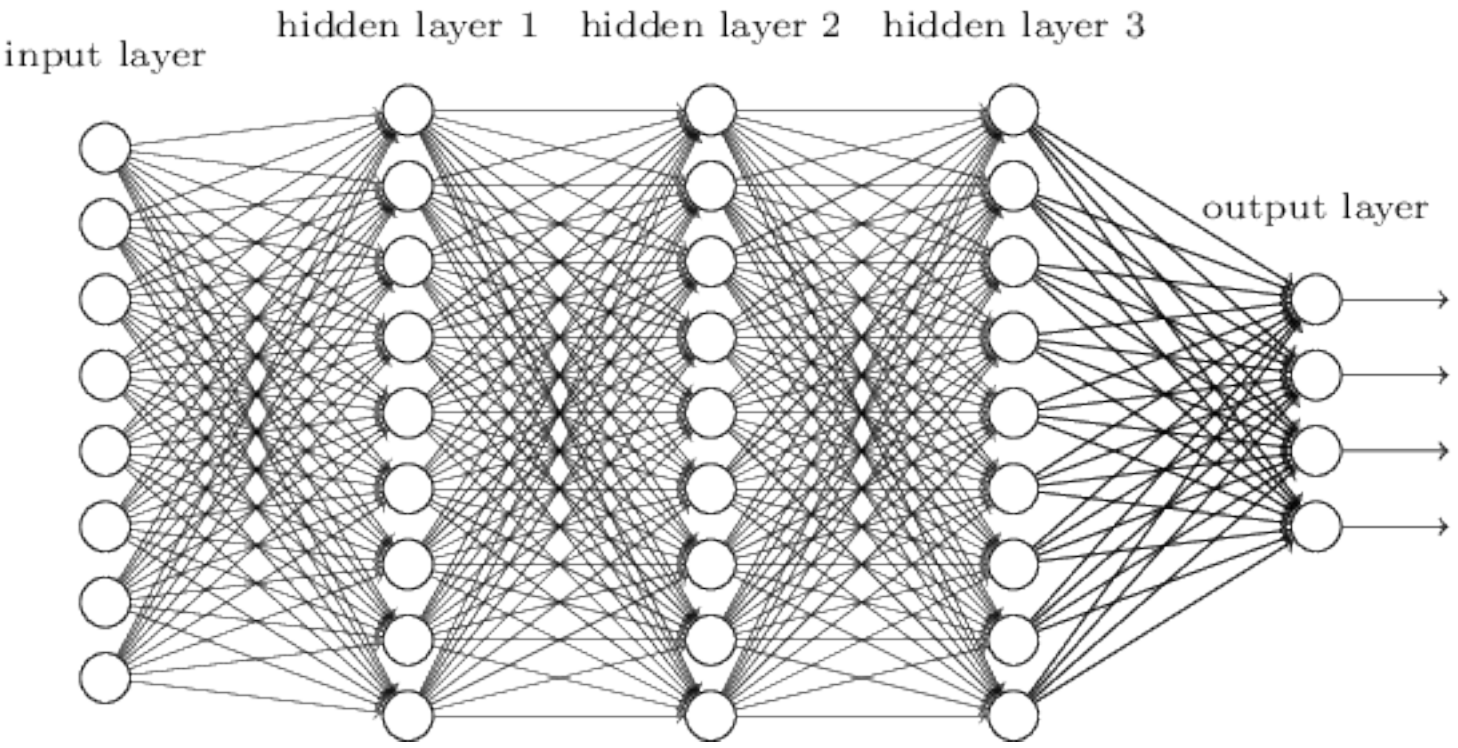
https://experiments.withgoogle.com/ai/sound-maker/view/

https://magenta.tensorflow.org/nsynth

https://deepmind.com/blog/wavenet-generative-model-raw-audio/

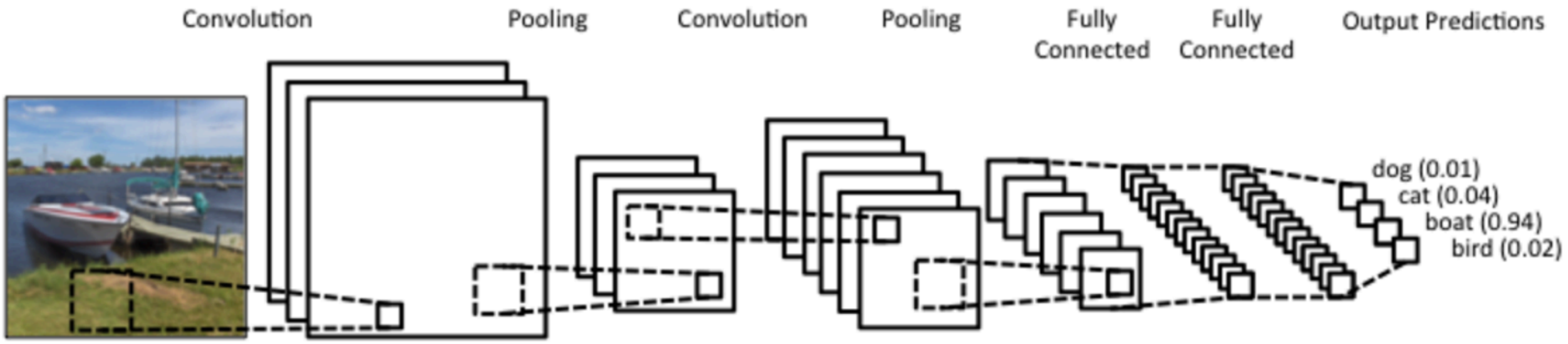
Types of Deep Neural Networks

1. DNN – all fully connected layers

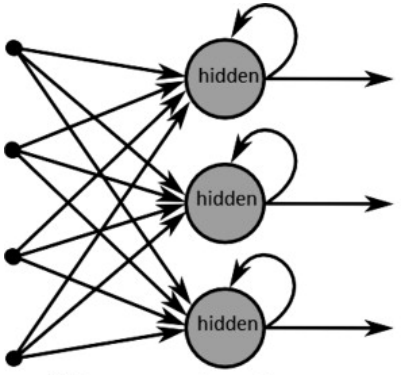


Types of Deep Neural Networks

2. CNN (Convolution Neural Networks) – some convolutional layers

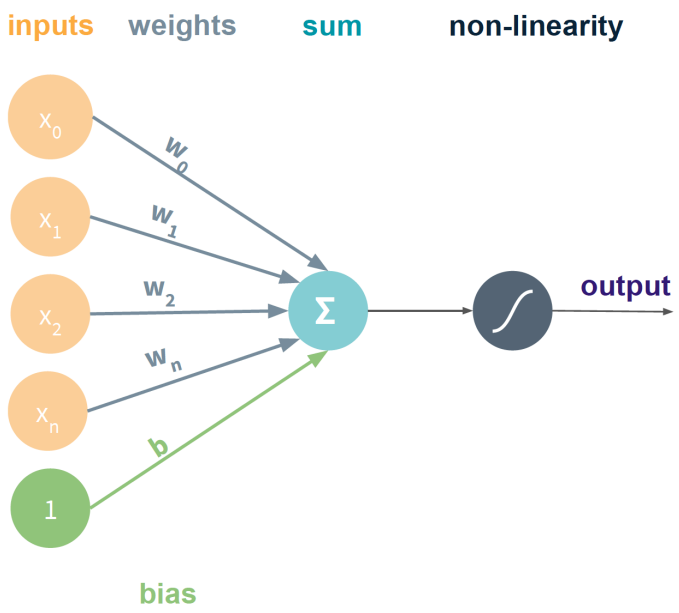


3. RNN (Recurrent Neural Network) – LSTM



Fundamentals of Neural Networks

- Receive signals from input neurons: $x_1, x_2, \ldots, x_n$
- Weight signals according to the link strength between neurons: $w_1 x_1, w_2 x_2, \ldots, w_n x_n$
- Add the input signals and bias: $w_1 x_1, w_2 x_2, \ldots, w_n x_n + b = \sum_{i=1}^{n} w_i x_i + b$
- Emit an output signal: activation function f



Activation Function

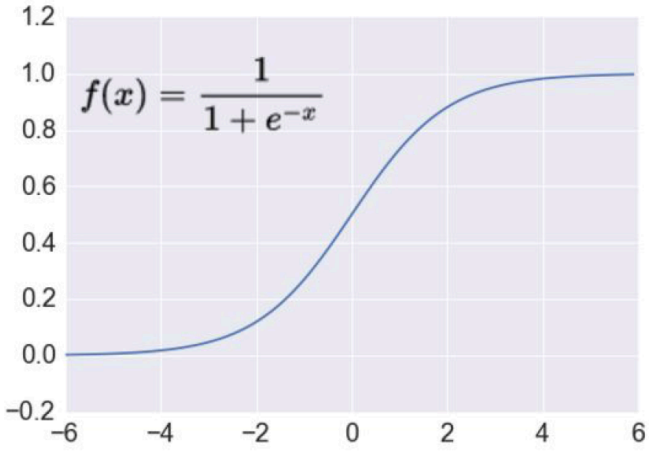
$$output = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Activation Functions

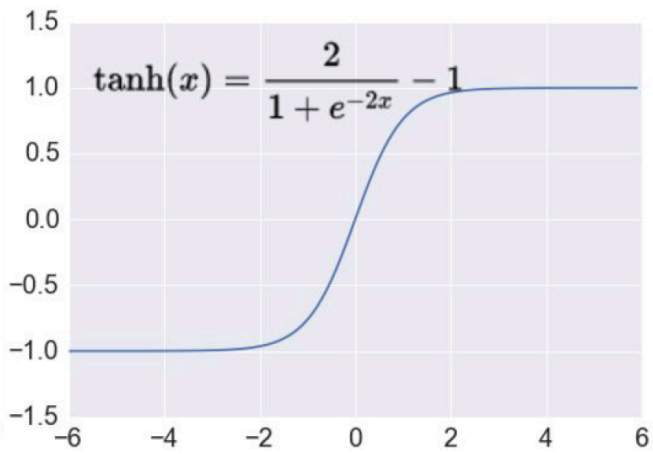
- Activation functions add non-linearity to our network's function
- Most real-world problems + data are non-linear

Common Activation Functions:

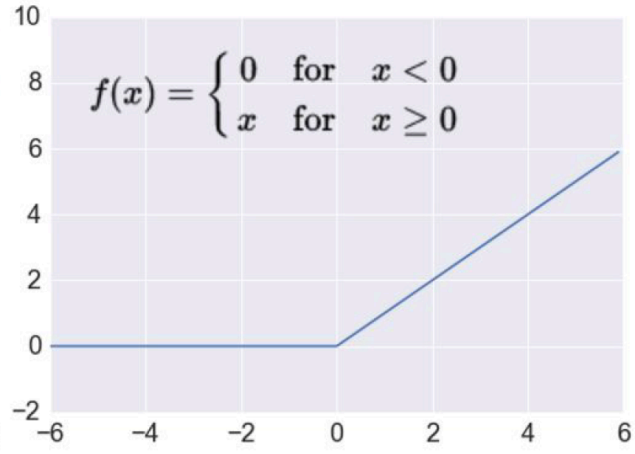
Sigmoid



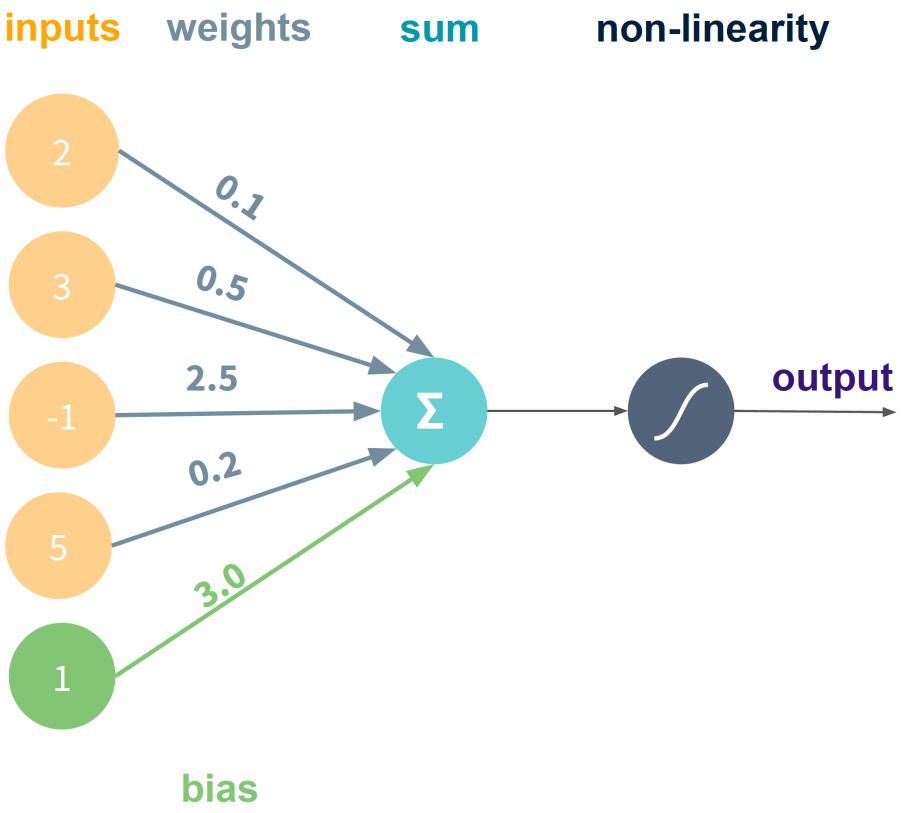
TanH



ReLU

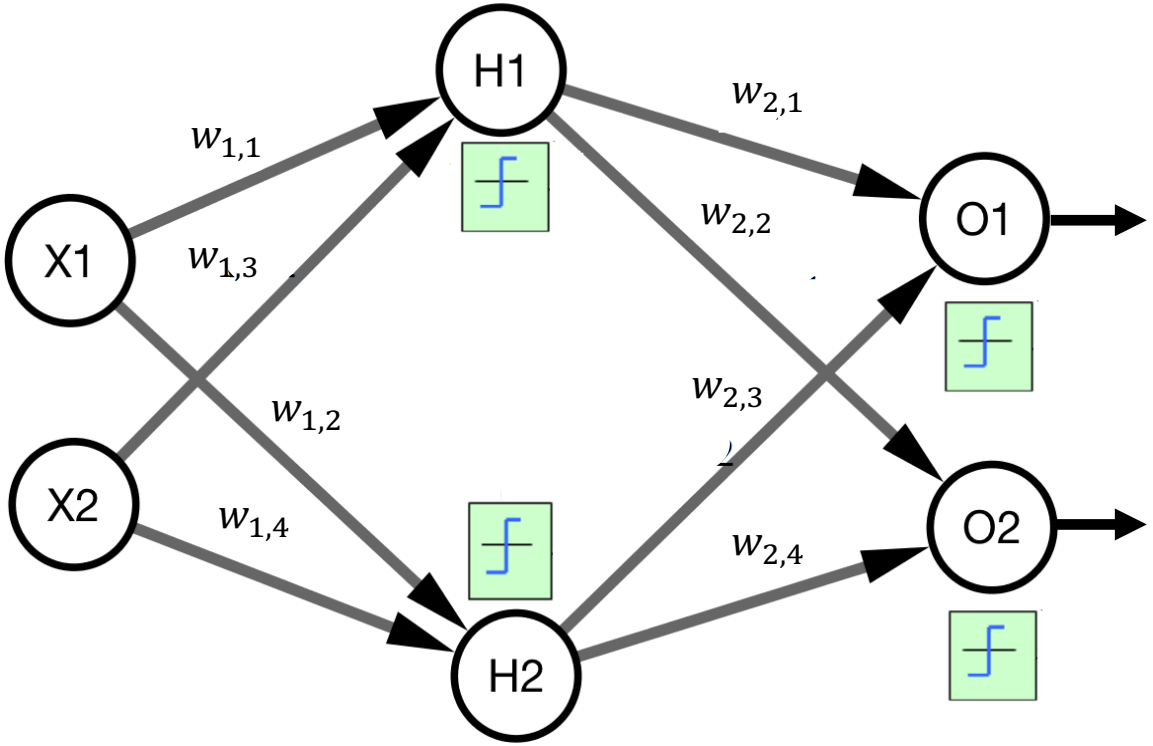


Fundamentals of Neural Networks



$$output = \sigma\,(2\times0.1 + 3\times0.5 + (-1)\times2.5 + 5\times0.2 + 1\times3) = 0.96$$

Functional Form of Neural Networks



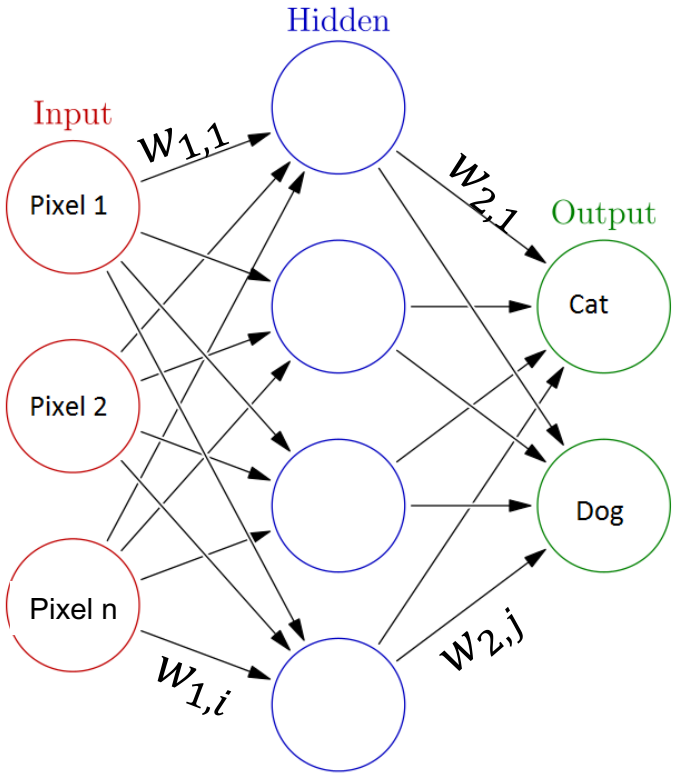
Feed-Forward pass:

$$H_1 = f_1(X_1 w_{1,1} + X_2 w_{1,3})$$
$$H_2 = f_1(X_1 w_{1,2} + X_2 w_{1,4})$$
$$O_1 = f_2(H_1 w_{2,1} + H_2 w_{2,3})$$
$$O_2 = f_2(H_1 w_{2,2} + H_2 w_{2,4})$$

Training a Network

- Find a set of weights so that the network exhibits the desired behaviour
- Example: cat vs. dog



Output	Label	
0.8,0.6	1,1	(Cat & dog)
0.08,0.3	0,1	(Dog)
0.92,0.01	1,0	(Cat)
0.02, 0.0	0,0	(No cat or dog)

Error Function

- Measure the difference between actual output and expected output
- One popular measure: sum of squared error

$$E(input, weight, label) = \sum(output - label)^2$$

- Note: Neural network is a composite/nested function that map the input to the output.

$$output = f_{NN}(input, weights)$$

Gradient Decent Learning

- A training example is a vector of inputs and the desirable output(s), i.e., $(\{x_1, x_2, \dots, x_n\}, \{t_1, t_2, \dots, t_c\}), t_i = 1$ iff the data point $\{x_1, x_2, \dots, x_n\}$, belongs to the i-th class.
- **Objective:** finding the weights w that minimise the difference between t and o (predicted output) for each of our training inputs.
- Define an error function *E* to be the sum of squared errors.

$$E = \frac{1}{2}\sum_{k=1}^{c}(o_k - t_k)^2$$

- If we think of *E* as height, it defines an error landscape on the weight space. *The aim is to find a set of weights for which E is very low.*
- This is done by moving in the steepest downhill direction (Gradient descent), i.e., $-\frac{\partial E}{\partial w_i}$

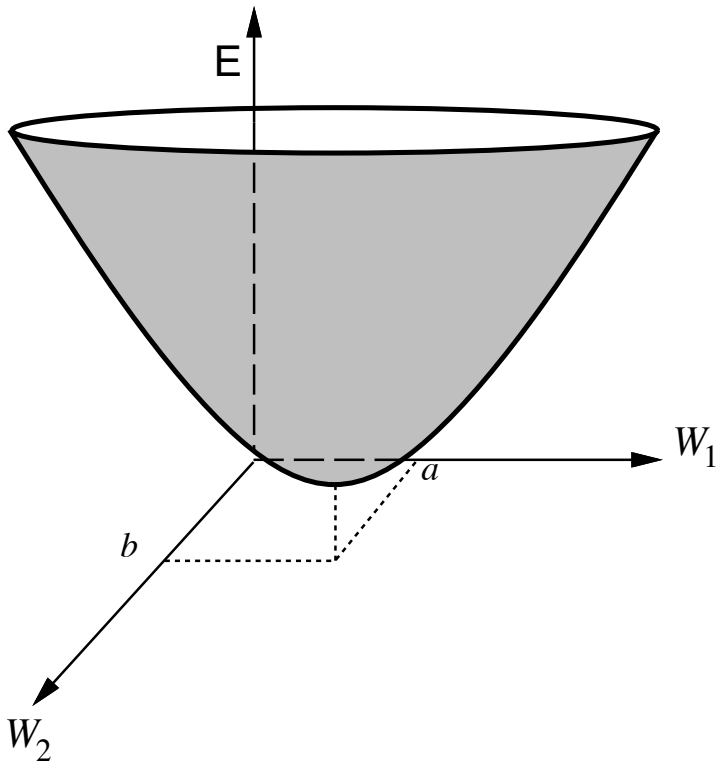
$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

η : Learning rate

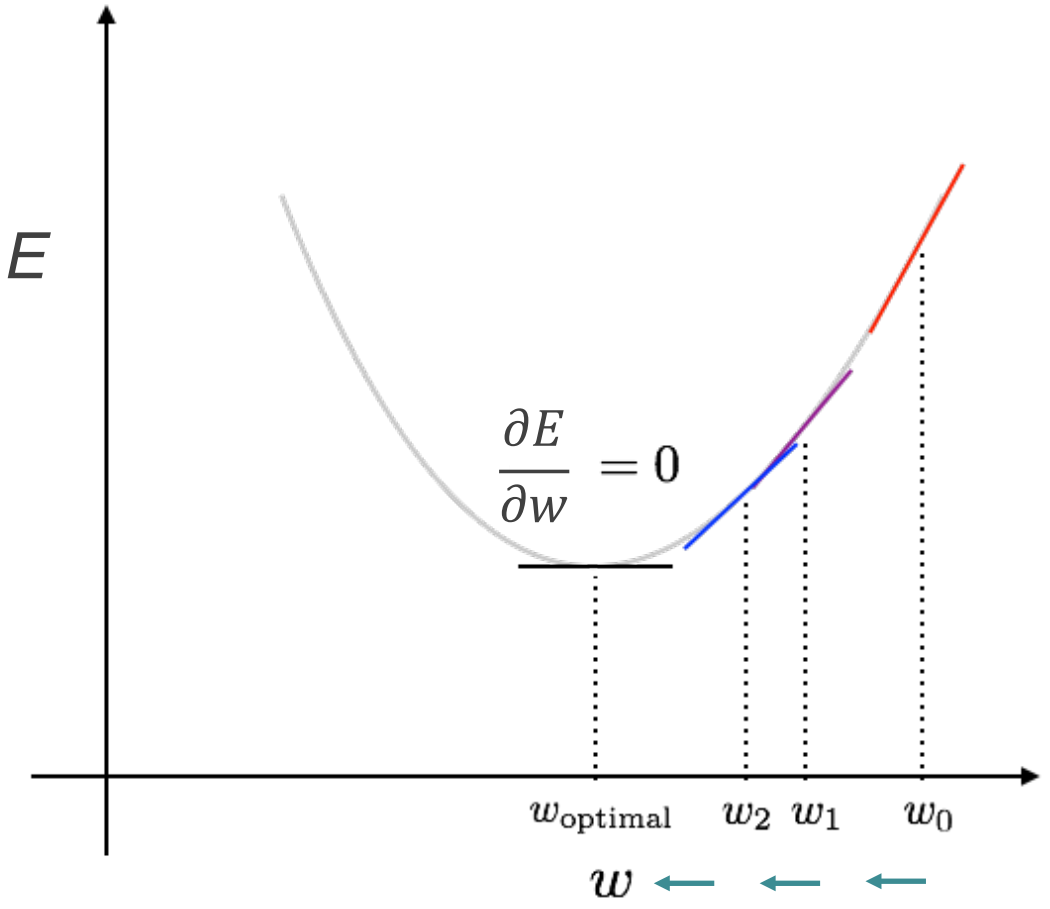
- The gradient descent approach is the basis of the *back-propagation algorithm*

Example Error Landscape

- An example error landscape for gradient descent search in weight space



Intuition behind Gradient Descent



$$w_1 = w_0 - \eta \frac{\partial E}{\partial w}(w_0)$$

$$w_2 = w_1 - \eta \frac{\partial E}{\partial w}(w_1)$$

$$\vdots$$

$$w_{\text{optimal}}$$

Negative gradient points towards a local minimum.

Weight update rule

To derive the weight update rule, we need to remember the chain rule from differential calculus:

if $y = y(u)$ and $u = u(x)$

then $\partial y/\partial x = (\partial y/\partial u)(\partial u/\partial x)$

So, if $\mathrm{E} = \frac{1}{2}\sum_{k=1}^{c}(o-t)^2$ and t = true output

$$\frac{\partial E}{\partial w_i} = \frac{\partial\left[\frac{1}{2}(o-t)^2\right]}{\partial o}\frac{\partial o}{\partial w_i}$$

$$= (o-t)\frac{\partial o}{\partial w_i}$$

$$= (o-t)x$$

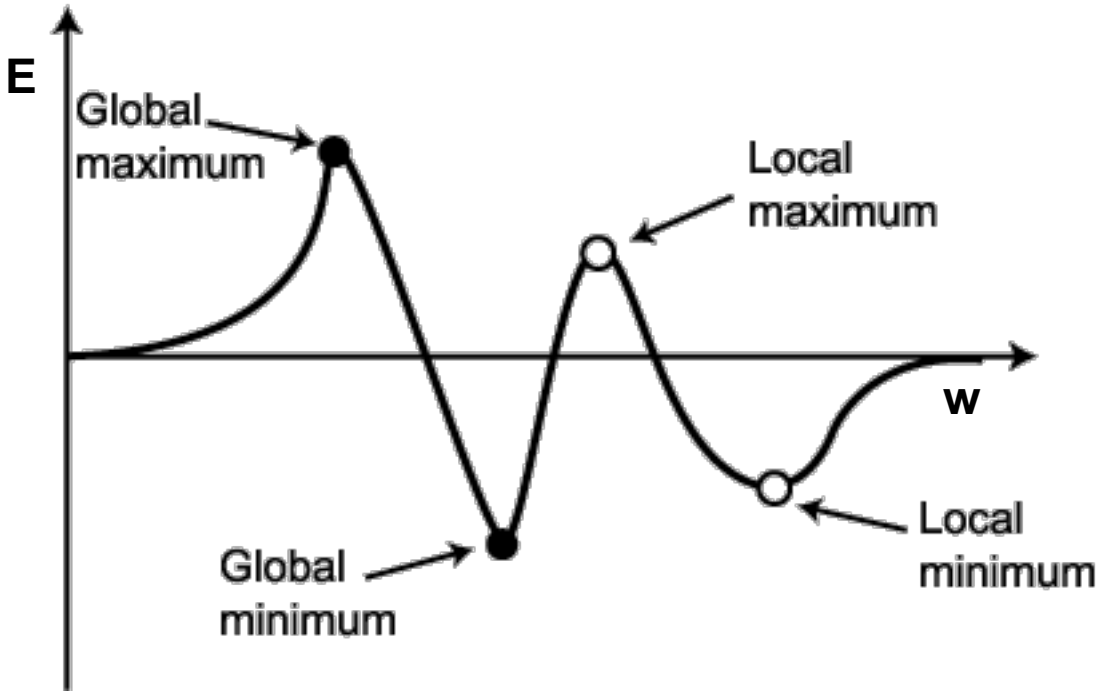
Normally we include a learning rate parameter η to control the update of weights in a stable manner

$$w_i \leftarrow w_i - \eta(o-t)x$$

We repeatedly update the weights based on each example until the weights converge

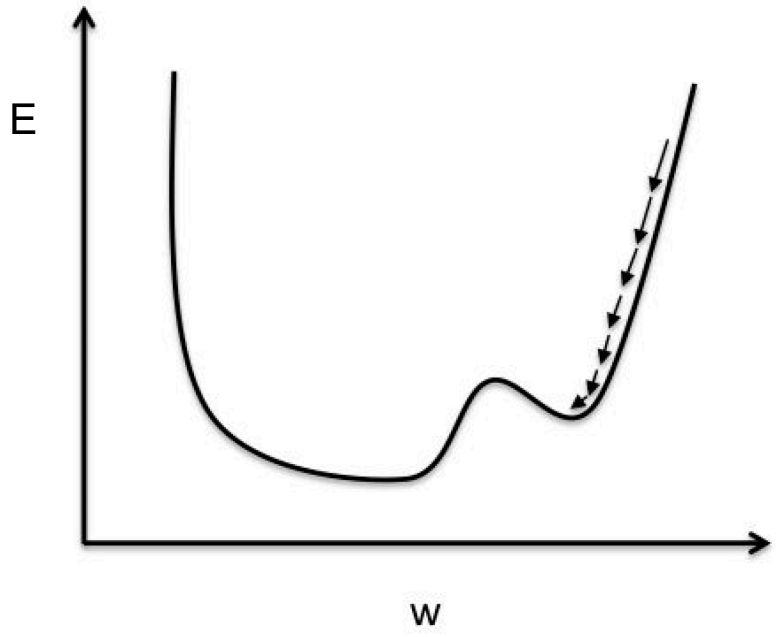
Intuition behind Gradient Descent

- The iterative algorithm might converge to one of the many local minima

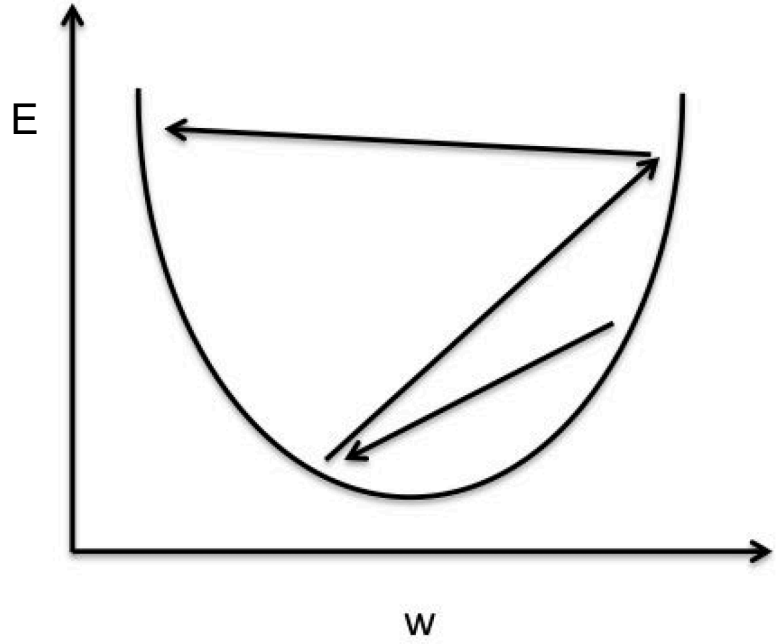


Adjusting Learning Rate

- Learning rate parameter η is a small value (usually $0.0001 - 0.1$) to control the update of weights in a stable manner.



Small learning rate: Many iterations until convergence and trapping in local minima.



Large learning rate: Overshooting.

Backpropagation Algorithm

Backpropagation(network, training data D, label T)

Initialise the weights ***w*** to small random values

Repeat

For each $d \in D, d =< x_1, \ldots x_n, t_1, \ldots, t_c >$

Forward pass*:* calculate hidden H_i and output O_j values

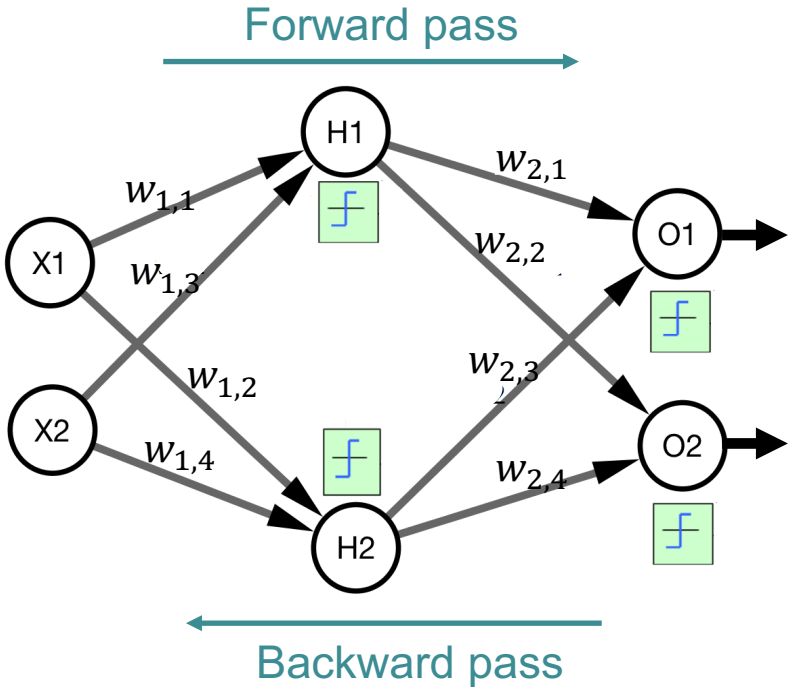
Backward pass:

Calculate error between o_k and t_k at out

Update the weights in each layer $w_{i,j}$

(in proportion to their effect on the error using *gradient descent* $w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$)

Until network has converged



Network Architectures/Parameters

- Number of input nodes = number of features
- Number of output nodes = number of classes

  $d = (\{x_1, x_2, \ldots, x_n\}, \{t_1, t_2, \ldots, t_c\}), t_i = 1$ *iff the data point belongs to the i-th class.*

  Neural networks can deal naturally with multi-class classification problems (compare to SVM?)
- Number of hidden layers
- Number of nodes in each hidden layers
- Regularization parameters (similar to C in SVM): control the complexity of the model, preventing overfitting.

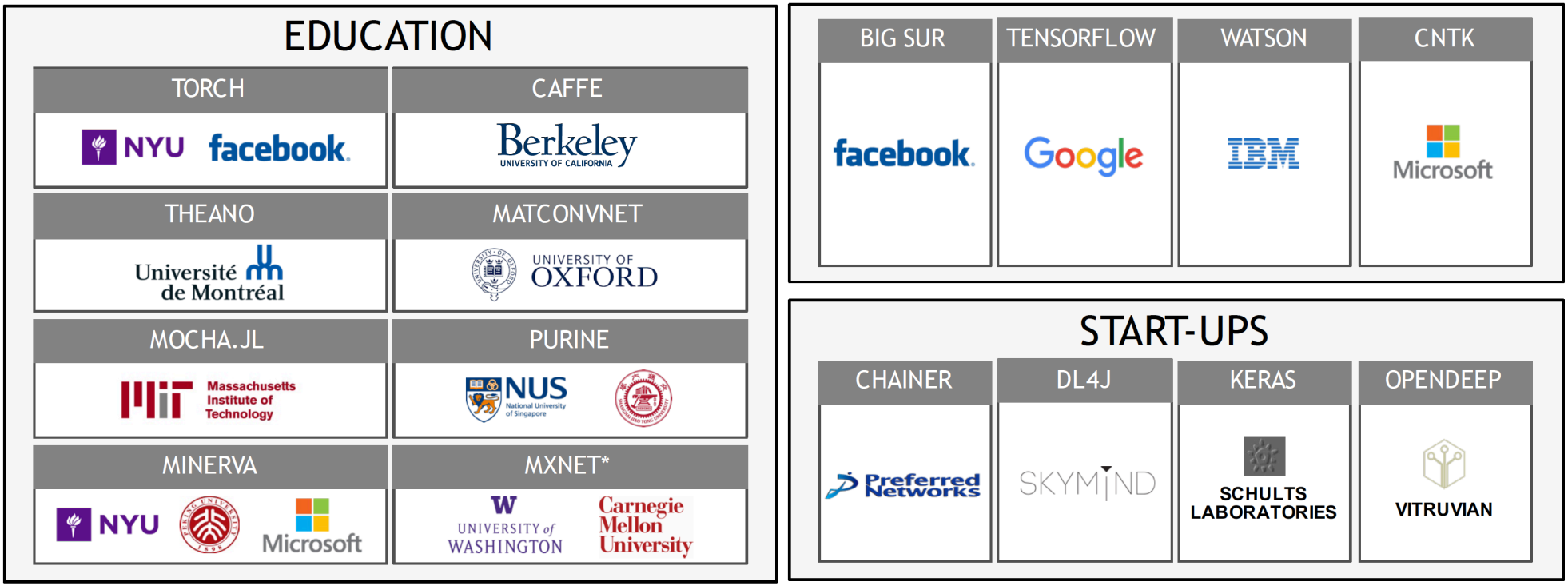
Live Demo:

http://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=circle&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=4,2&seed=0.13820&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTim

Why Deep Learning Works Now? (but not 30 years ago)

- Lots more computing power
  - GPU: mini-super computer: 3000 cores, 12GB RAM, 100x speed up over CPU
  - Much deeper and larger networks, trained for a few days-a week
- Lots more data
  - phone, camera, sensors
  - Internet, Crowd-Sourcing (Amazon Mechanical Turk) ! huge labeled data sets of millions of training samples
- More sophisticated algorithms
  - Clever initialization: initial weights are close to a good minimum
  - Stochastic gradient descent: fast approximate gradient descent algorithms
  - Activation function that facilitate training of deep nets, e.g., Rectified Linear Unit: ReLU(x) = max(0; x)
  - Strong regularization that prevent deep nets from overfitting

The Engine of Modern AI

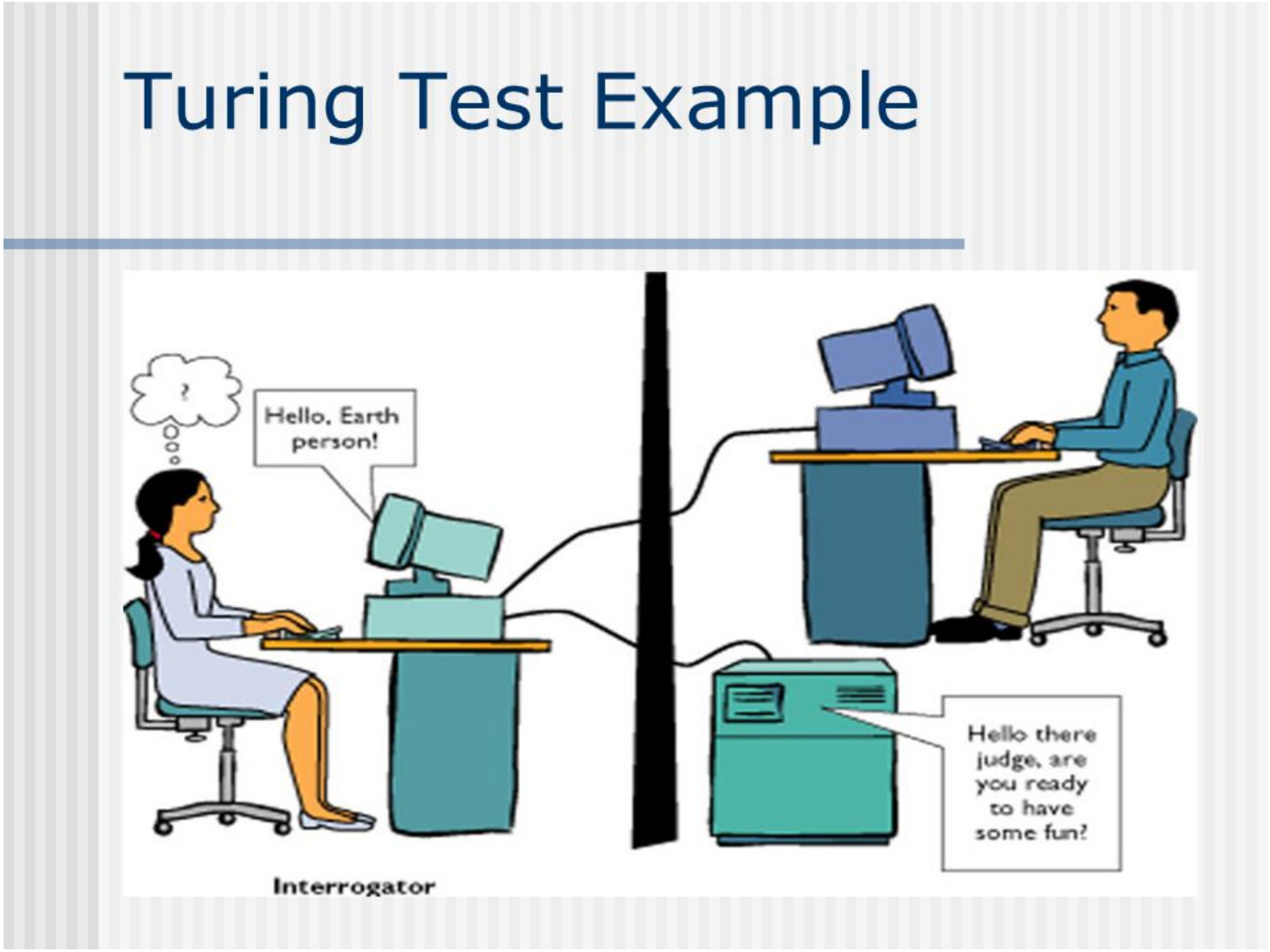


How to Apply Neural Networks

Many readily available implementations:

- Scikit-learn, Matlab, Weka: Suitable for small scale problems
- Modern implementations: TensorFlow (Google), Torch (Facebook), CNTK (Microsoft), Theano (U. Toronto), Keras (Python wrapper)
- Support training on (multiple) GPUs
- Support advanced network structures: convolutional neural networks (for image recognition), recurrent neural networks (for sequential data, e.g., text)

Exercise – Generating Music



Resources

Reading List

- http://neuralnetworksanddeeplearning.com/chap1.html
- http://neuralnetworksanddeeplearning.com/chap2.html

Further Resources

- http://www.wired.com/2014/01/geoffrey-hinton-deep-learning
- http://chronicle.com/article/The-Believers/190147/

Courses

- https://class.coursera.org/neuralnets-2012-001
- https://www.coursera.org/course/ml

Book

- http://www.deeplearningbook.org/