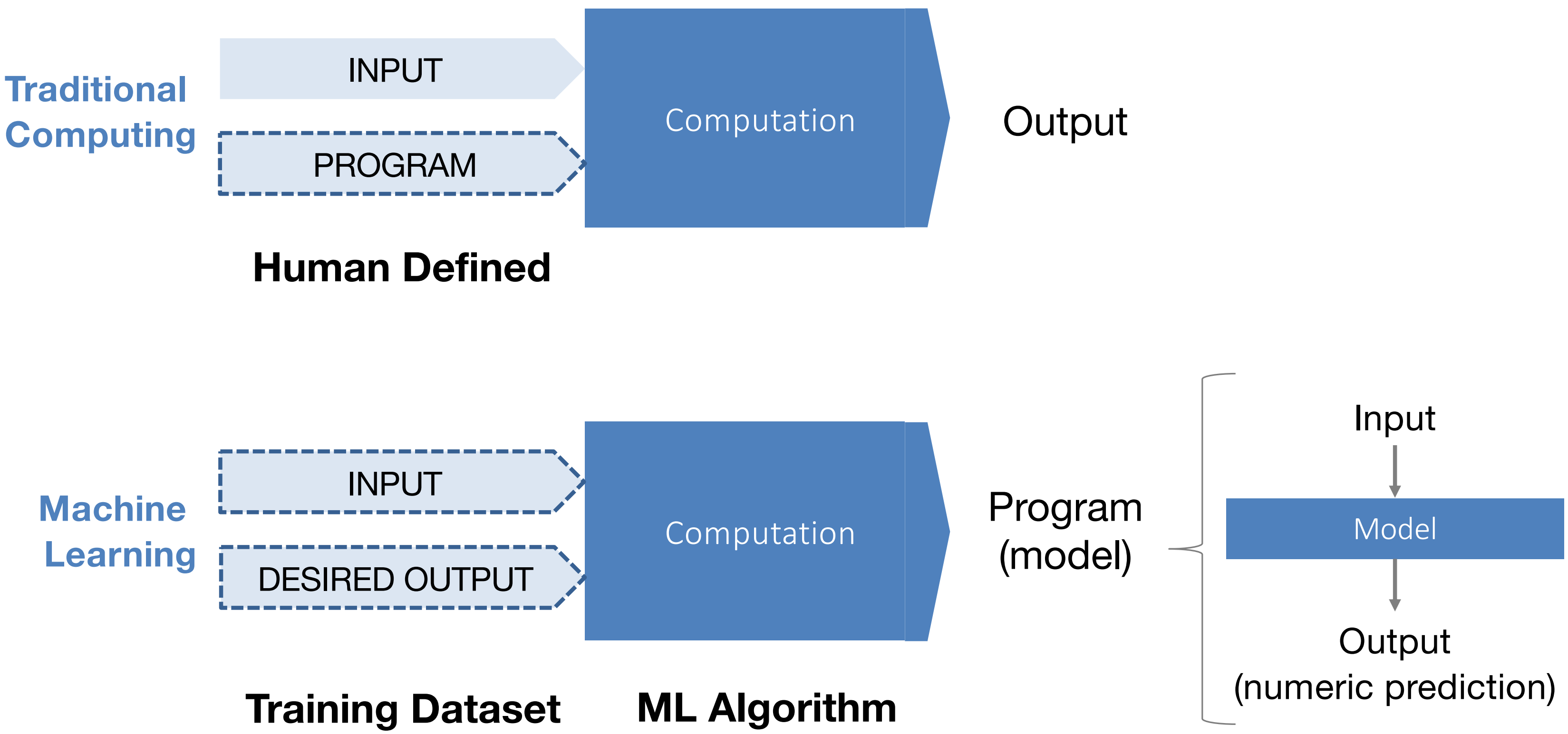
Introduction to Neural Networks and Deep Learning

Brian Keng
Chief Data Scientist
Rubikloud Technologies
brian.keng@rubikloud.com
@bjlkeng

Adjunct Professor, Data Science
Rotman School of Management, University of Toronto
brian.keng@rotman.utoronto.ca

A Brief Overview of Machine Learning

Traditional Programming vs. Machine Learning



Machine Learning: Learning by Example

Write a program with explicit rules:

```
if email contains "V!agr@"
  then mark as spam
else if email contains …
  then mark as spam
else if email contains …
  then mark as spam
…
otherwise mark as not-spam
```

Write a program to "learn" from labelled examples by changing itself:

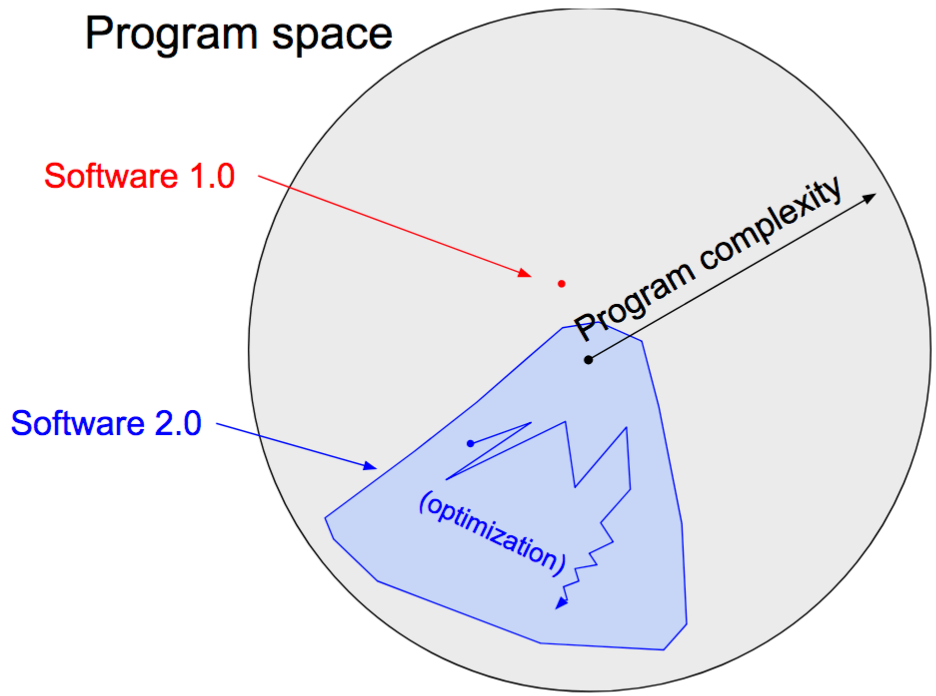
```
try to classify labelled emails
(spam, not-spam)

change self to reduce errors

repeat until satisfied with error

classify unlabeled emails
```

MAN VS. MACHINE

Programming without Humans (ML as Software 2.0)?

> It turns out that a large portion of real-world problems have the property that it is significantly easier to collect the data than to explicitly write the program.

Software 2.0
Andrej Karpathy



Software 2.0, Andrej Karpathy, https://medium.com/@karpathy/software-2-0-a64152b37c35

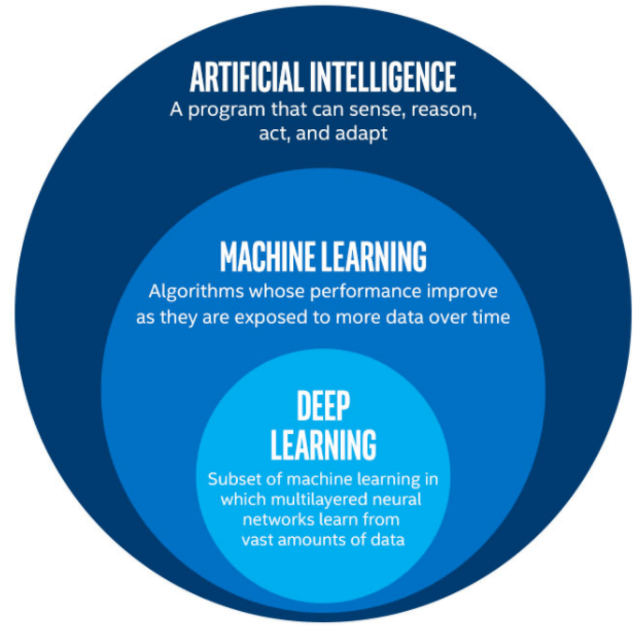
Artificial Intelligence

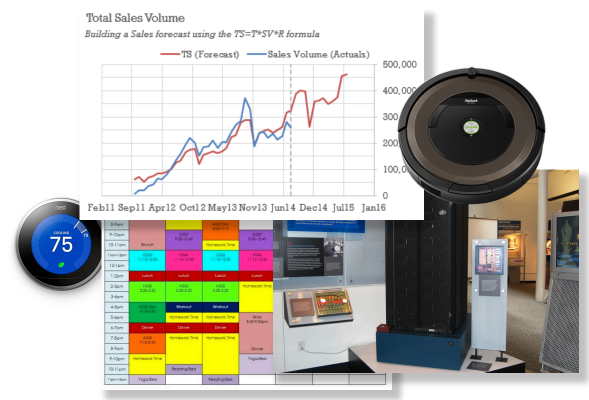
- **Artificial Intelligence:**
  - Academic discipline that attempts to build machines that *mimic human cognitive* functions such as "*learning*" and "*problem solving*"

- **Symbolic AI ("Good Old-Fashioned AI" - GOFAI)**
  - Logic, Search, Simulation, Expert Systems

- **Statistical Learning ("Learning from Data")**
  - Machine Learning, Deep Learning, Statistical Inference

- **Examples:**
  - Forecasting model, Scheduling a timetable, Roomba Vacuum, Computer Chess, Nest Thermostat





What Can Neural Networks Do?

Which one are real vs. AI-generated? (AI Face Generation)

Rotman

https://arxiv.org/pdf/1812.04948.pdf

Self Driving Cars

Rotman

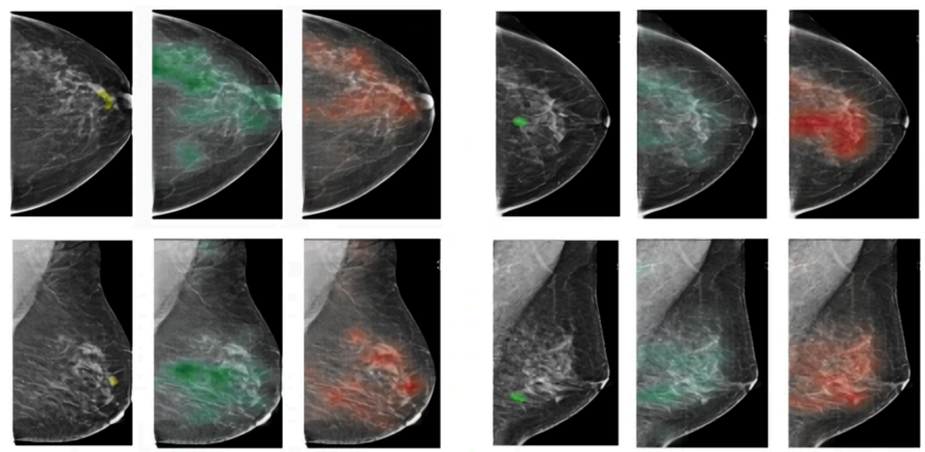
Source: https://www.techiexpert.com/tesla-using-artificial-intelligence-big-data/

AI vs. Doctors

Rotman

AI can detect skin cancer better than doctors now

PTI | May 29, 2018, 11.58 AM IST

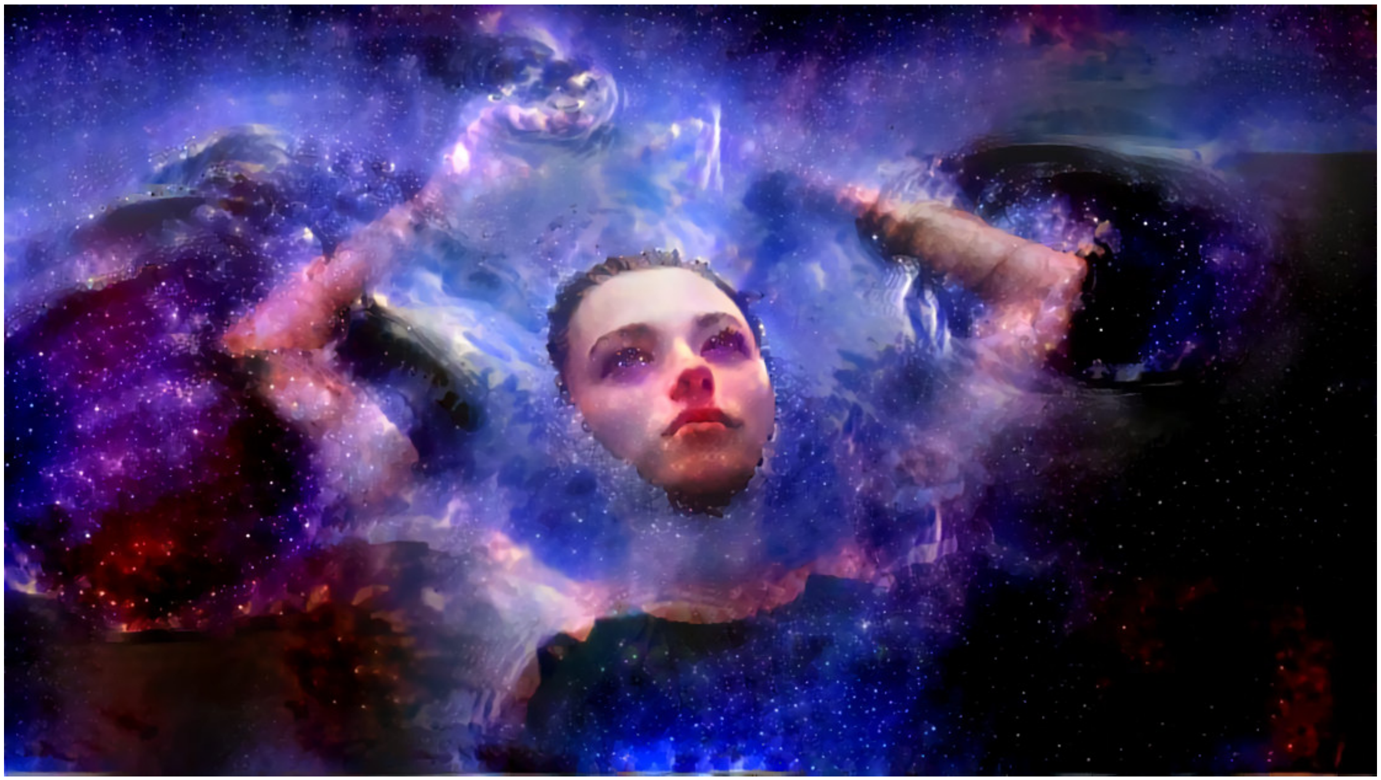


Source:

https://economictimes.indiatimes.com/magazines/panache/ai-can-detect-skin-cancer-better-than-doctors-now/articleshow/64365380.cms
http://www.sciencemag.org/news/2017/04/self-taught-artificial-intelligence-beats-doctors-predicting-heart-attacks
https://venturebeat.com/2019/03/21/nyu-open-sources-breast-cancer-screening-model-trained-on-over-200000-mammography-exams/

AI Art: Deep Dreaming

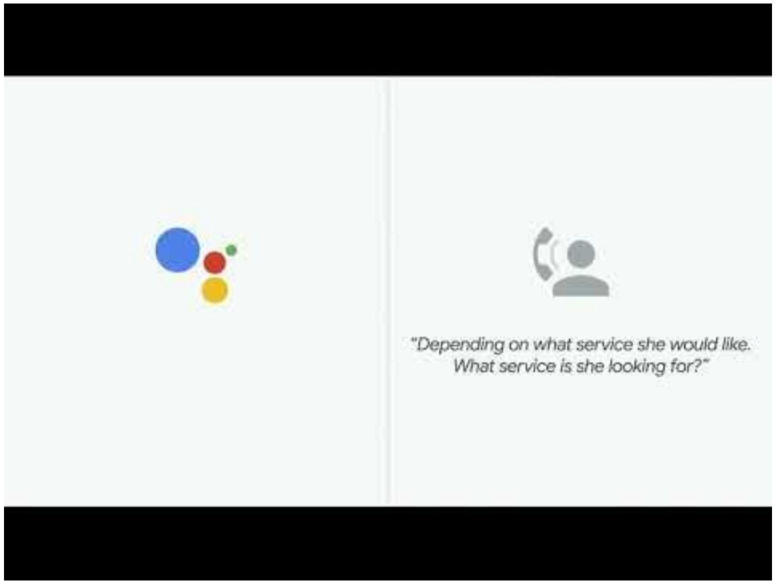
Rotman



Source: https://deepdreamgenerator.com/

AI Conversation Bots (Google's Duplex)

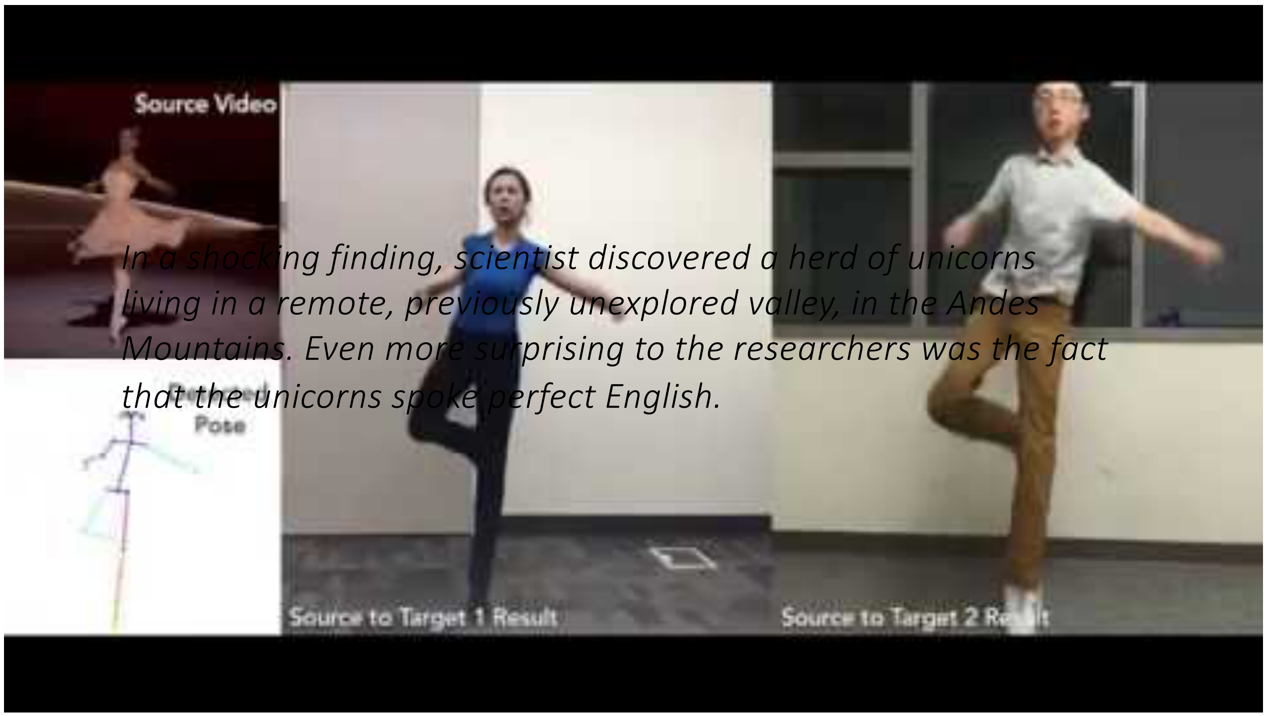
Rotman



https://www.youtube.com/watch?v=NO0-5MuJvew

AI Making You Dance

Rotman



In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

https://youtu.be/PCBTZh41Ris
https://arxiv.org/pdf/1808.07371.pdf

Human-Level Language Comprehension

Rotman

Seed Human Written Text:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Machine Generated Text Examples:

```
The scientist named the population, after
their distinctive horn, Ovid’s Unicorn. These
four-horned, silver-white unicorns were
previously unknown to science.
```

https://openai.com/blog/better-language-models/

Question and Answer Passage:

The 2008 Summer Olympics torch relay was run from March 24 until August 8, 2008, prior to the 2008 Summer Olympics, with the theme of “one world, one dream”.

...

Machine Generated Answers:

Q: What was the theme?
A: “one world, one dream”.

Q: What was the length of the race?
A: 137,000 km

Q: Was it larger than previous ones?
A: No

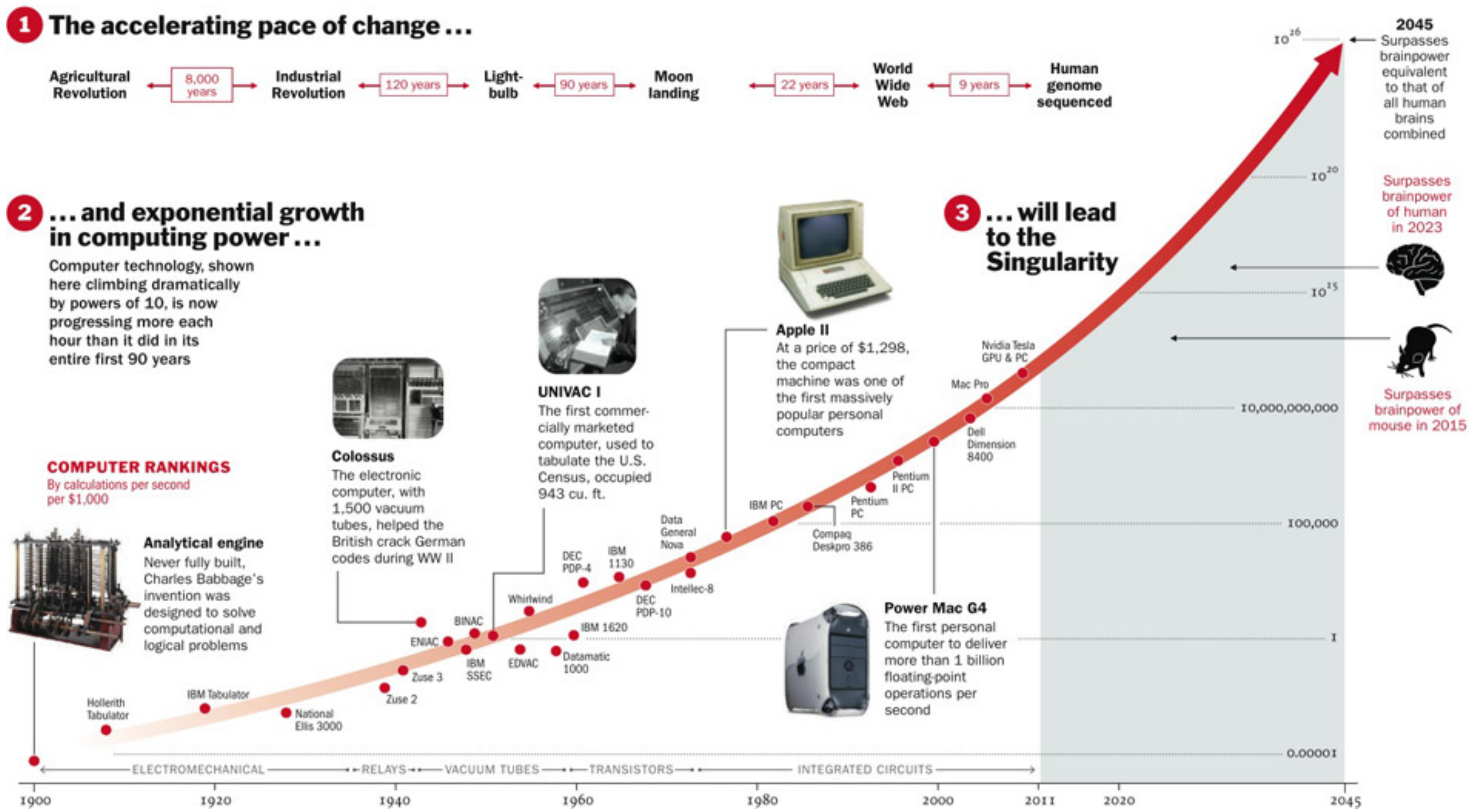
Q: Where did the race begin?
A: Olympia, Greece

Q: Is there anything notable about that place?
A: birthplace of Olympic Games

How Did We Get Here?

Massive Growth in Computing Power…

Rotman

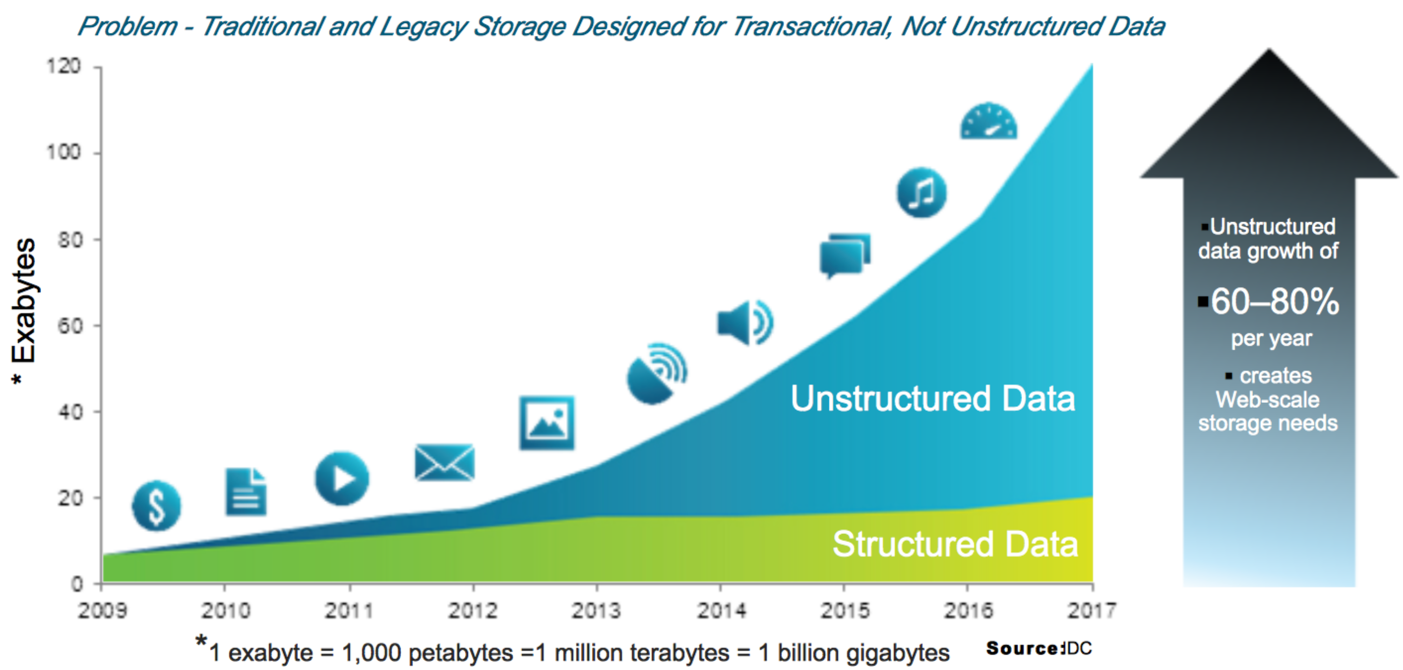


Source: http://content.time.com/time/interactive/0%2C31813%2C2048601%2C00.html

And Massive Growth in Data…

Rotman

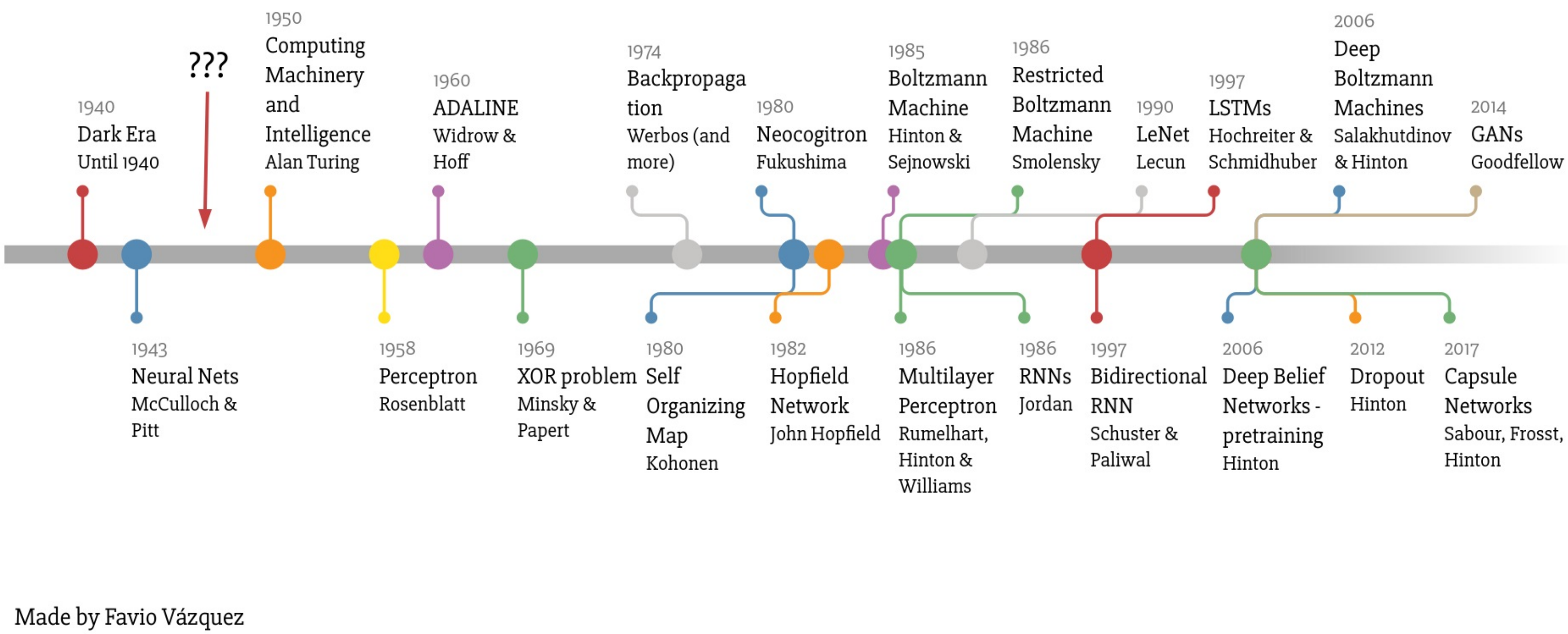
Data Growth



https://www.dubber.net/unlocking-unstructured-data-voice-processing-power-zoe/

And Some Research…

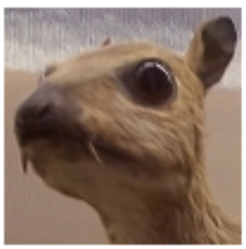
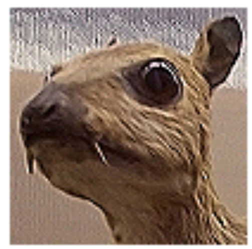
Rotman



https://towardsdatascience.com/a-weird-introduction-to-deep-learning-7828803693b0

Why Are Neural Networks So Good?

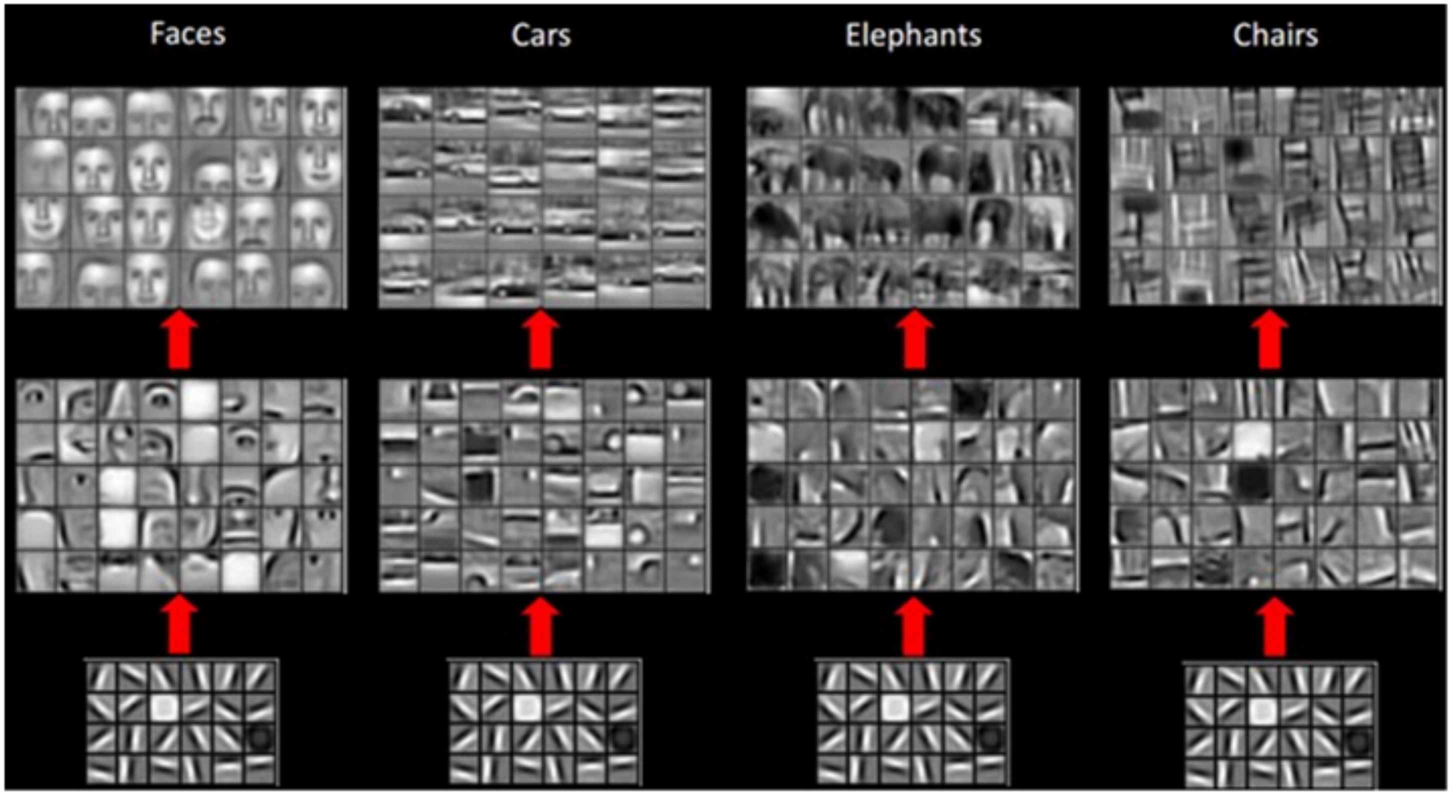
Manual Feature Engineering

Original	*Gaussian Blur*	*Sharpen*	*Edge Detection*
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$
			

Source: https://santexgroup.com/blog/tag/tensor-flow/

Automatic Feature "Learning"

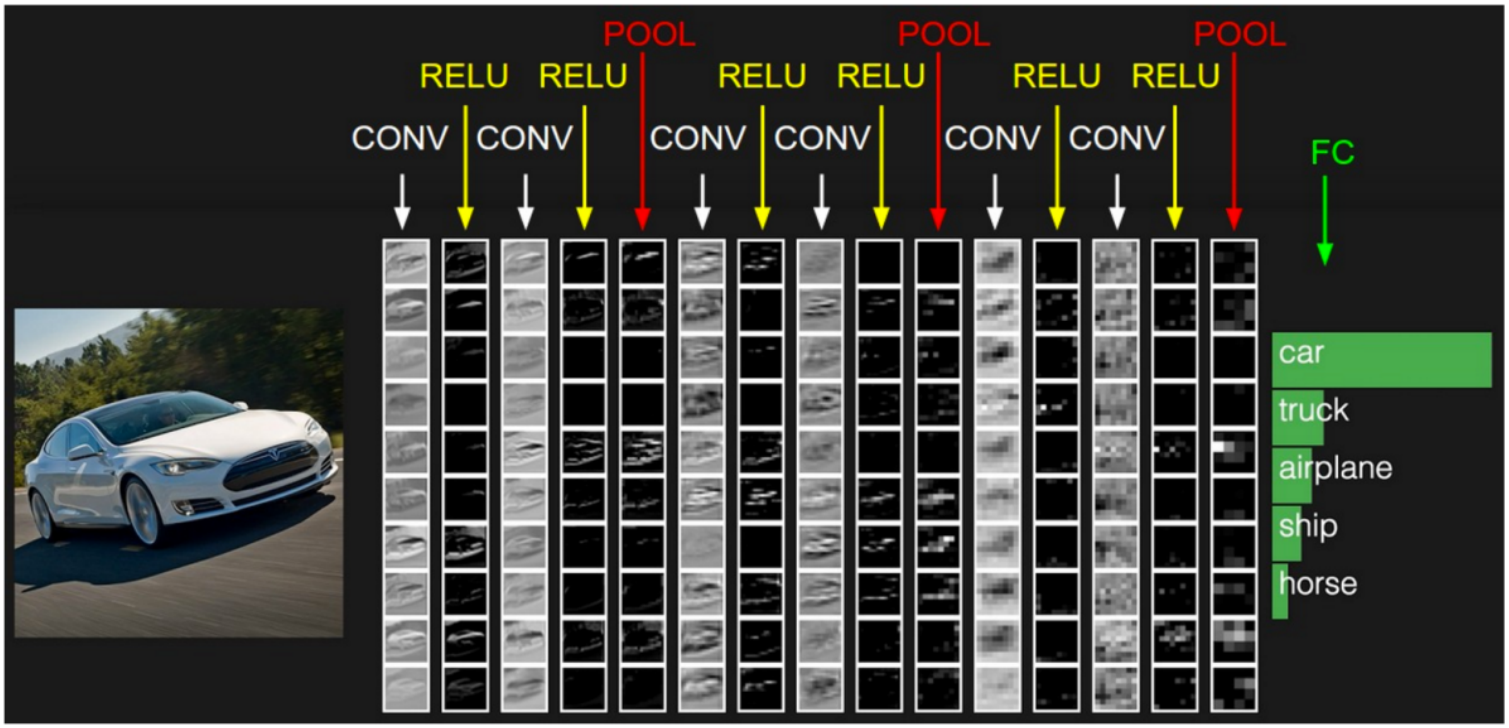
Rotman



Source: https://stats.stackexchange.com/questions/146413/why-convolutional-neural-networks-belong-to-deep-learning

Automatic Feature "Learning"

Rotman

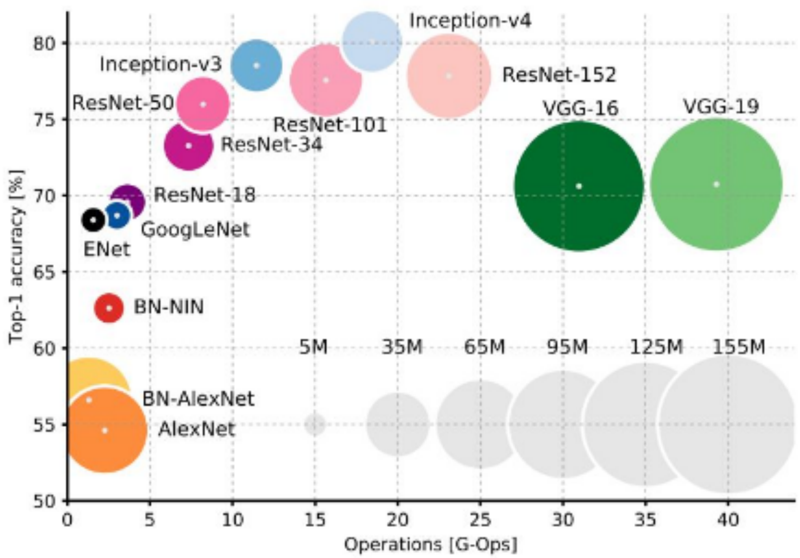


Source: https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3

Neural Networks are Scalable

Rotman

Model	Number of Parameters
Linear Regression	< 100
LeNet (1998)	60K
AlexNet (2012)	60M
VGG-16 (2014)	138M
Inception-v3 (2015)	23M
ResNet-152 (2015)	60M



Source: https://www.kdnuggets.com/2016/11/intuitive-explanation-convolutional-neural-networks.html/3
"AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS", https://arxiv.org/pdf/1605.07678.pdf

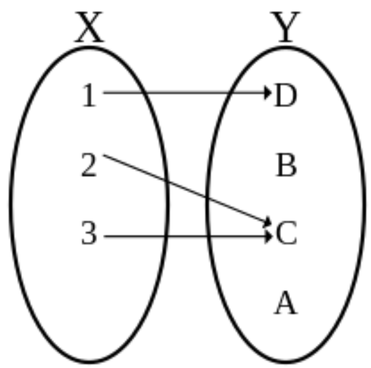
Neural Networks as Function Approximators

Functions and Machine Learning

Rotman

A function f maps each element in the ***domain*** (X) to a single element in the **range** (Y). Most functions we see are real-valued functions, e.g.:

- $f(x) = 2x^2 + 3$
- $g(x_0, x_1) = e^{\alpha x_0 + \beta x_1 + C}$

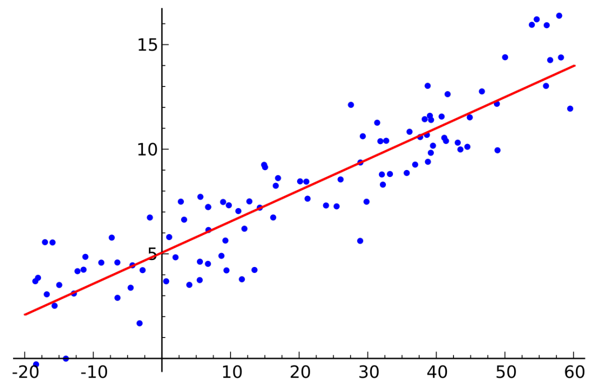


Data $= (X, Y)$ can be thought of as*:

- Inputs X (i.e. features, regressors, covariates etc.)
- Outputs Y (i.e. observations, response variables, labels etc.)

One useful way to think about machine learning is as ***function approximation***:

- Finding a function that "fits" the data according to some mathematical objective function



Approximating the data using a line of best fit

* This is only *one* (of many) ways to view ML; there are many lenses to understand it (e.g. probabilistically, algorithmically, optimization etc.)

What Makes a Good Function Approximator: Linear?

Rotman

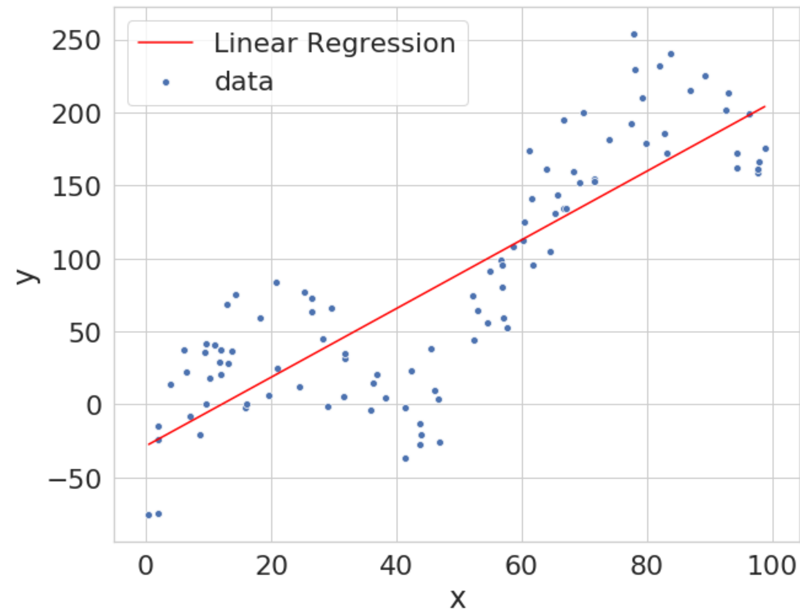
Parameters

Response Variable

$$y = f(x) = \beta_1 x + \beta_0$$

Features

- 2 *parameters* (β_1, β_0) provide limited flexibility
- 1 *feature* (aka *covariate, independent variable, predictor, regressor*)
- 1 *response* variable (aka *output, dependent variable*)
- Linear models are rigid (not much flexibility)

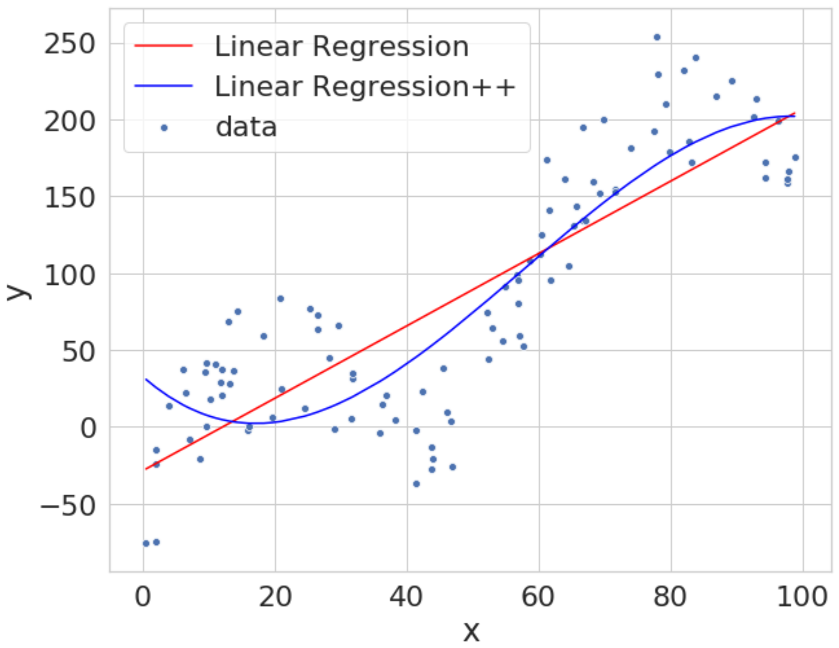


More Complex Linear Function?

Rotman

$$y = f(x) = \beta_3 x^3 + \beta_2 x^2 + \beta_1 x + \beta_0$$

- 4 parameters $(\beta_3, \beta_2, \beta_1, \beta_0)$
- 3 features $(\mathrm{x}^3, \mathrm{x}^2, \mathrm{x})$ where each feature is a function of our original covariate
- Need to use intuition to come up with right features (transformations of raw data observations)



How about a Neural Network? (Definitions)

Rotman

Define a non-linear ("activation") function:

$$ReLU(\boldsymbol{x}) \coloneqq \max(0, \boldsymbol{x}) \text{ (element-wise)}$$

Define a "hidden layer" function:

$$H_{n,d}(\boldsymbol{x}) = ReLU(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = ReLU\left(\begin{bmatrix} w_{11} & \cdots & w_{1d} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nd} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}\right)$$

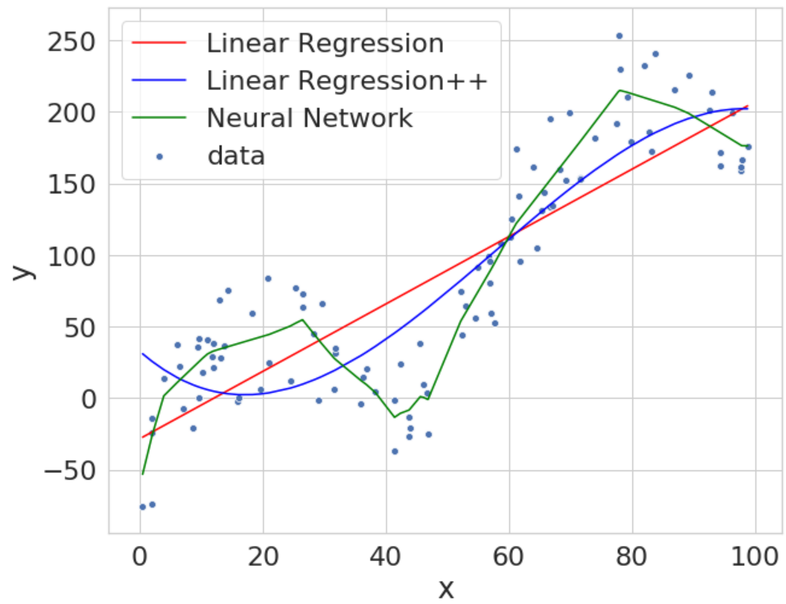
How about a Neural Network?

Rotman

5 hidden layer neural network with 500 hidden units:

$$y = f(x) = (H_{1,500}) \circ (H_{500,500}) \circ (H_{500,500}) \circ (H_{500,500}) \circ (H_{500,500}) \circ \left(H_{500,1}\right)(x)$$
$$= H_{1,500}\left(H_{500,500}\left(H_{500,500}\left(H_{500,500}\left(H_{500,500}\left(H_{500,1}(x)\right)\right)\right)\right)\right)$$

- 1M+ parameters (each w_{ij}'s in each $\boldsymbol{W}$ matrix is a parameter)
- 1 features (x)
- No need for any feature engineering

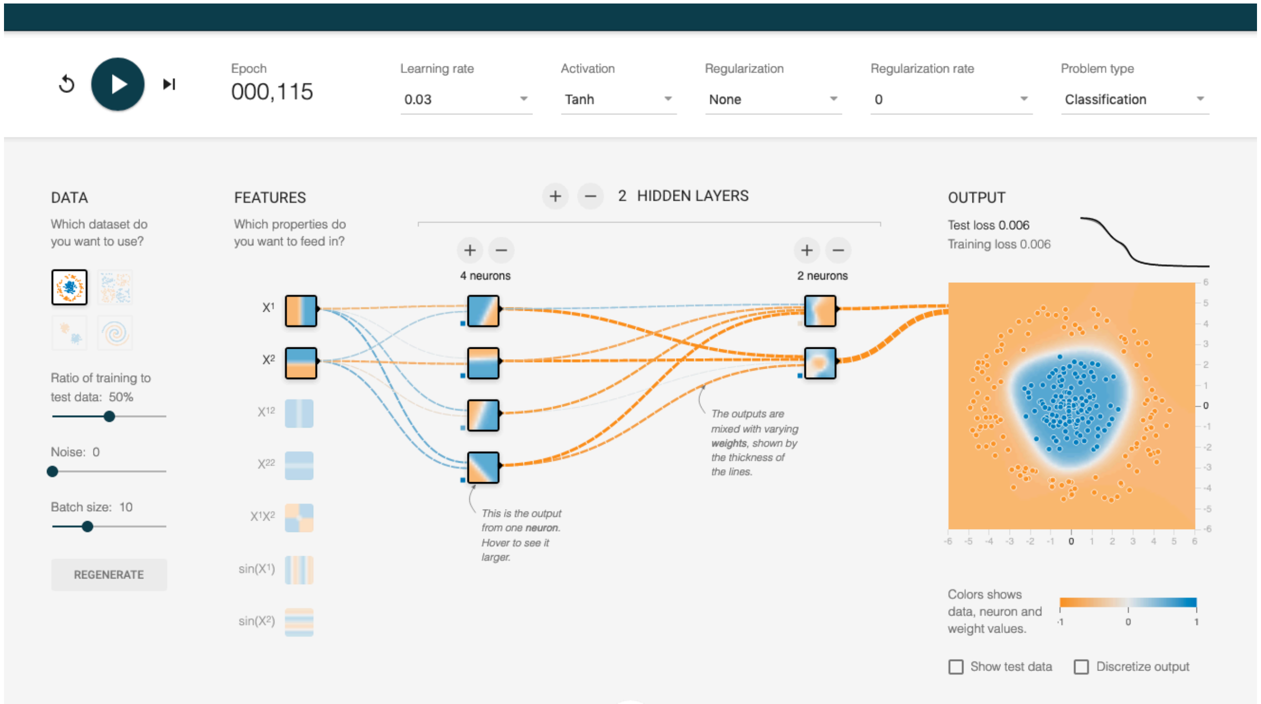


Basics of Feed Forward Neural Networks

(aka Multi-Layer Perceptrons, Feed Forward Neural Networks, Deep Feedforward Networks, Dense Neural Networks, Fully Connected Neural Networks)

Demo: Neural Network Playground

Rotman

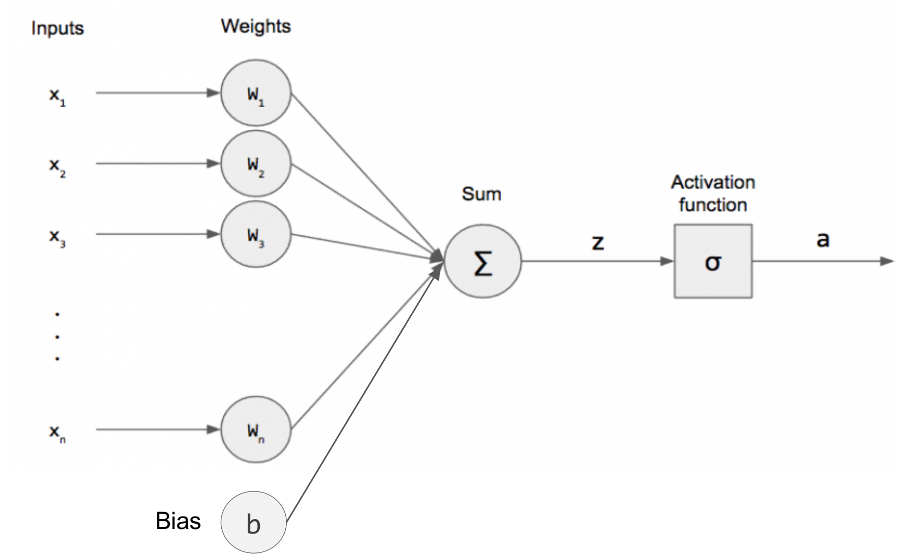


https://playground.tensorflow.org/

The Anatomy of a Perceptron (aka neurons)

Rotman

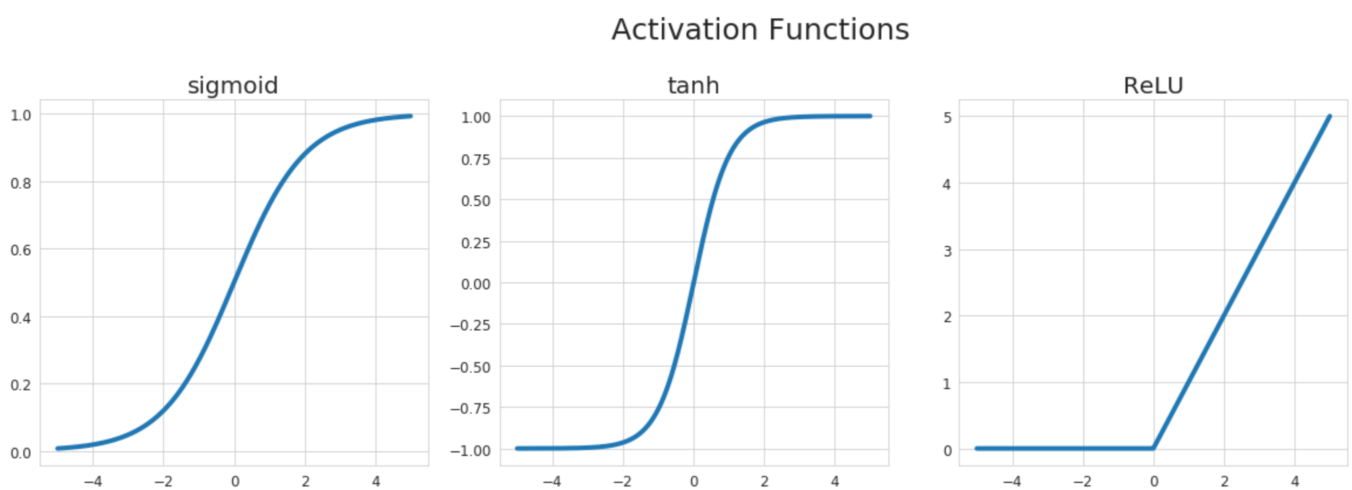
- **Inputs ($\boldsymbol{x_i}$)**: input features (or outputs from previous layers)
- **Weights ($\boldsymbol{w_j}$)**: Learnable real-valued parameters (this is akin to learning the "slope" and "intercept" of a line)
- **Sum ($\boldsymbol{\Sigma}$)**: Summation of product of weights and inputs
- **Activation Function ($\boldsymbol{\sigma}$)**: Non-linear function mapping z to a (usually monotonic and continuous)
- **Bias ($\boldsymbol{b}$)**: "Intercept" for the perceptron (can be implemented by making one input constant)



$$f(\boldsymbol{x}) = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right) = \sigma\left(\begin{bmatrix} w_1 & \cdots & w_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b\right) = \sigma(\boldsymbol{W}^T \boldsymbol{x} + b)$$

What Makes a Perceptron Special?

Rotman



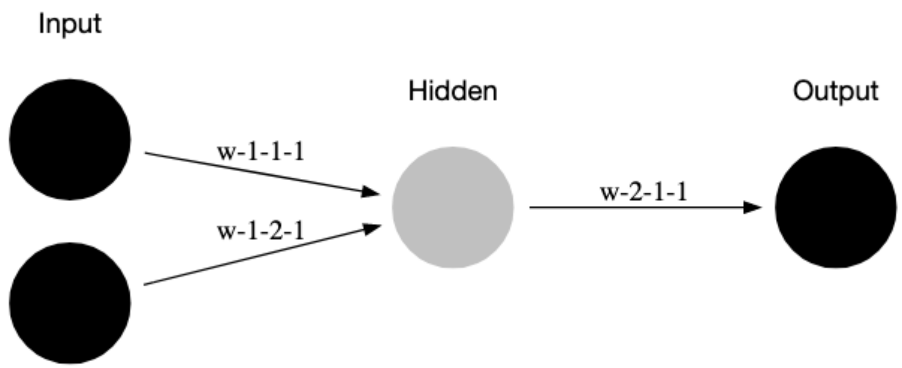
$$sigmoid(x) = \frac{1}{1 + e^{-x}} \qquad tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \qquad \text{ReLU}(x) = \max(0, x)$$

- Non-linear function allows perceptron to learn "interesting" functions
- The more perceptrons you have, the more "interesting" functions you can learn

A Very Simple Neural Network

Rotman

- **Input Layer**: represents input features
- **Hidden Layer**: represents perceptrons in any non-input/non-output layers
- **Output Layer**: represents perceptrons used to generate final output(s) (for now just the identify function)



Name	Value
Inputs	2
Parameters	(1·2+1) + (1·1+1) = 3+2 = 5
Hidden Layers	1
Output Layers	1
Depth	2
Width	[2, 1, 1]

$$f(\boldsymbol{x}) = y_1 = Id\left(w_{2,1,1} \cdot \sigma\left(\begin{bmatrix} w_{1,1,1} & w_{1,1,2} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b_{1,1}\right) + b_{2,1}\right)$$
$$= \boldsymbol{w_2} \cdot \sigma\left(\boldsymbol{w_1 x} + b_{1,1}\right) + b_{2,1}$$

Adding More Perceptrons

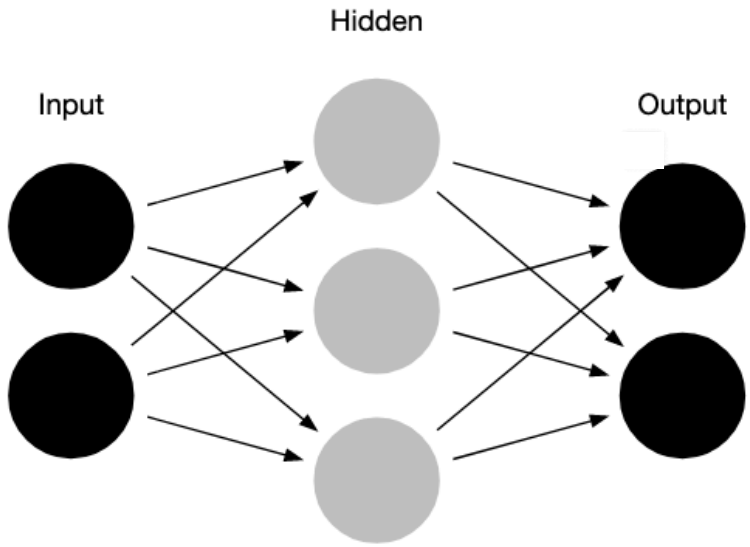
Rotman

$$f(\boldsymbol{x}) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

$$= Id\left(\begin{bmatrix} w_{2,1,1} & w_{2,2,1} & w_{2,3,1} \\ w_{2,1,2} & w_{2,2,2} & w_{2,3,2} \end{bmatrix} \cdot \sigma\left(\begin{bmatrix} w_{1,1,1} & w_{1,2,1} \\ w_{1,1,2} & w_{1,2,2} \\ w_{1,1,3} & w_{1,2,3} \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_{1,1} \\ b_{1,2} \\ b_{1,3} \end{bmatrix}\right) + \begin{bmatrix} b_{2,1} \\ b_{2,2} \end{bmatrix}\right)$$

$$= \boldsymbol{W_2} \cdot \sigma(\boldsymbol{W_1 x} + \boldsymbol{b_1}) + \boldsymbol{b}_2$$

Name	Value
Inputs	2
Parameters	(3·2+3) + (2·3+2) = 17
Hidden Layers	1
Output Layers	1
Depth	2
Width	[2, 3, 2]

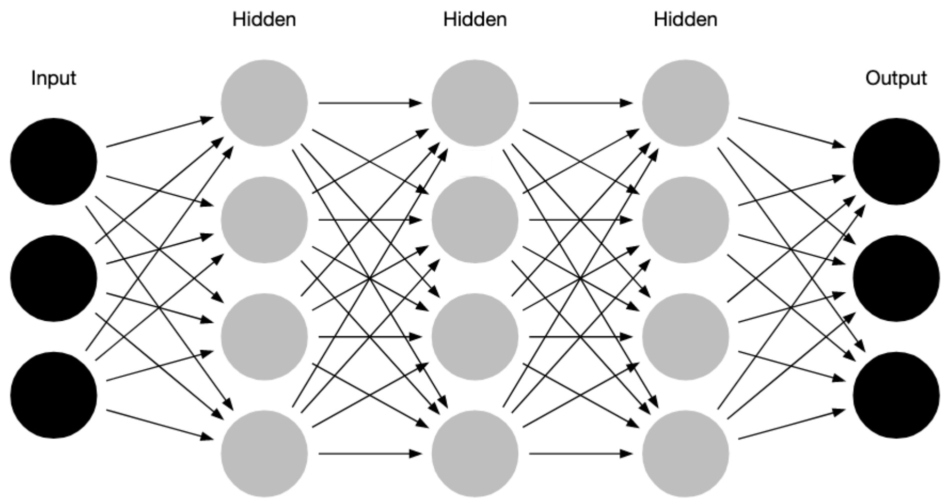


Adding More Layers…

Rotman

$$f(\boldsymbol{x}) = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$
$$= \boldsymbol{W_4} \cdot \sigma(\boldsymbol{W_3} \cdot \sigma(\boldsymbol{W_2} \cdot \sigma(\boldsymbol{W_1}\boldsymbol{x} + \boldsymbol{b_1}) + \boldsymbol{b}_2) + \boldsymbol{b_3}) + \boldsymbol{b_4}$$

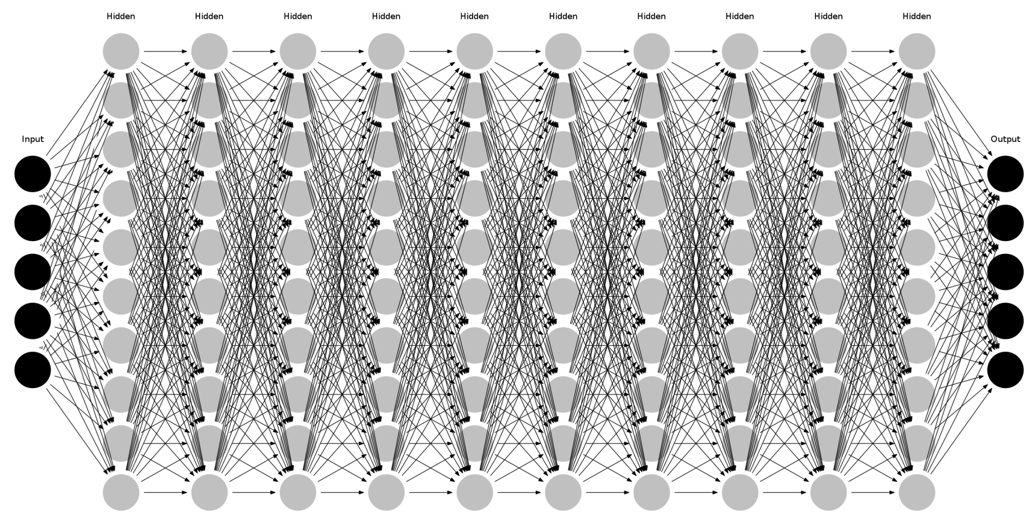
Name	Value
Inputs	3
Parameters	(4·3+4) + (4·4+4) + (4·4+4) + (3·4+3) = 71
Hidden Layers	3
Output Layers	1
Depth	4
Width	[3, 4, 4, 4, 3]



A “Deep” Neural Network…

Rotman

Name	Value
Inputs	5
Parameters	(10·5+10) + 9·(10·10+10) + (5·10+5) = 1105
Hidden Layers	10
Output Layers	1
Depth	11
Width	[5, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 5]

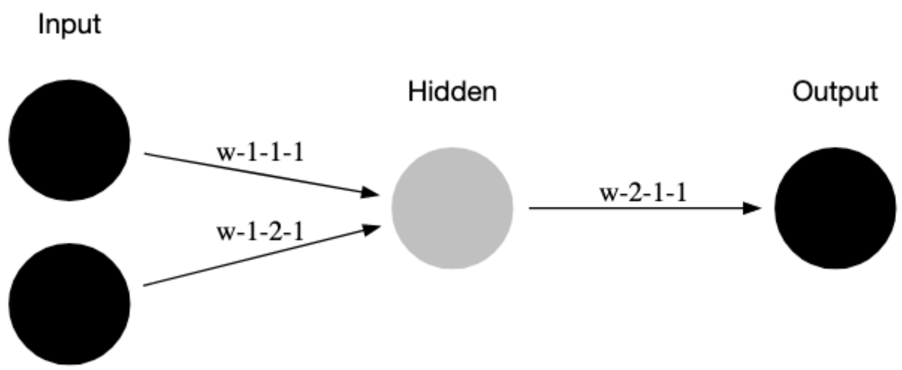


“Deep” = Many of Layers (definition of deep changes as deeper networks are built)

How Do We Define the Output Perceptrons?

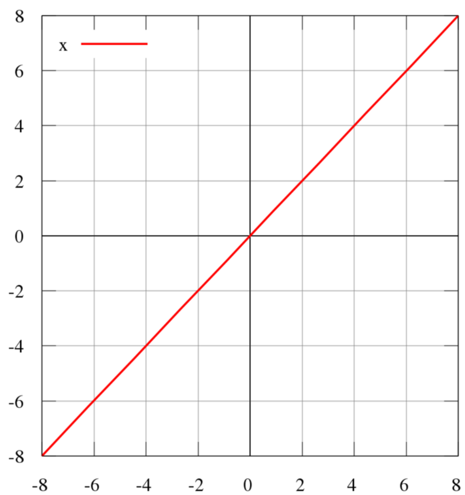
Rotman

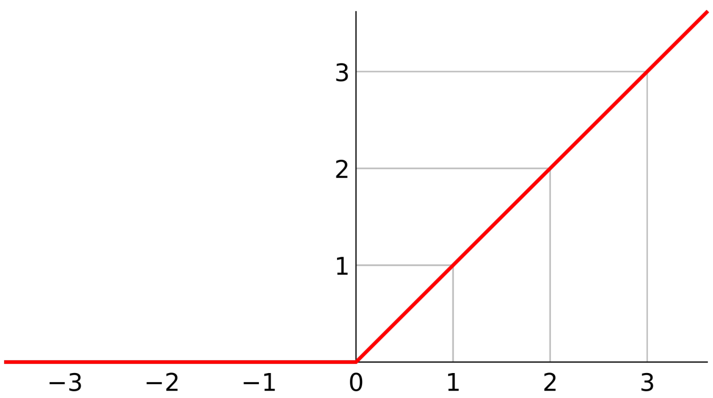
- Output neurons model the “y” values (or labels)
- Activation function should match your response variable
- Considerations:
  - Regression vs Classification problem
  - Range of output variable
  - Loss function (see this later)
  - Discrete vs. Continuous
- We’ll look at 4 common activation functions for output units:
  - Identity
  - ReLU
  - Sigmoid
  - Softmax



Output Units (Linear, Positive Real-Valued)

Rotman





Linear Output (Identity Activation Function)

- Real-valued output ($y \in [-\infty, \infty]$)
- Appropriate for general regression problems

$$\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{h} + \boldsymbol{b}$$

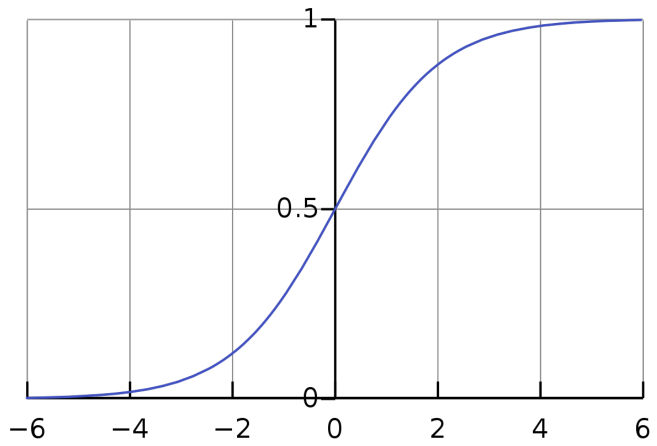
Positive-Valued Output

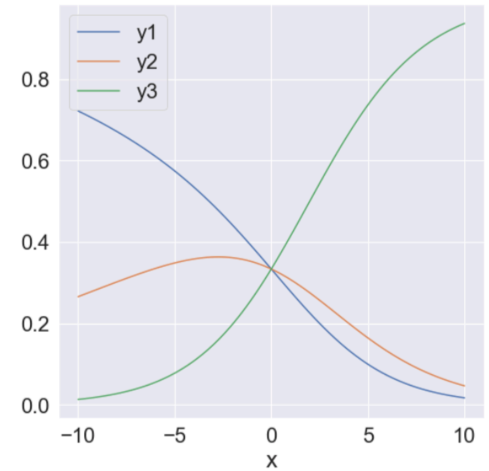
- Real-valued output ($y \in [0, \infty]$)
- Appropriate for general regression problems

$$\boldsymbol{y} = ReLU(\boldsymbol{W}^T \boldsymbol{h} + \boldsymbol{b})$$

Output Units (Binary, Categorical)

Rotman





Binary Probability Output

- Real-valued output ($y \in [0,1]$)
- Appropriate for binary classification problems
- Output variable is probability of "1" label

$$\boldsymbol{y} = sigmoid(\boldsymbol{W}^T \boldsymbol{h} + \boldsymbol{b})$$

Softmax (Multinomial/Categorical Probability) Output

- Categorical Probability ($y_i \in [0,1], \sum_i y_i = 1$)
- Appropriate when output labels are categorical labels (e.g. Red, Green Blue, Yellow)
- N output variables corresponding to a probability distribution across N categories

$$\boldsymbol{z} = \boldsymbol{W}^T \boldsymbol{h} + \boldsymbol{b} \qquad y_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Example: A Neural Network for the XOR Function

Rotman

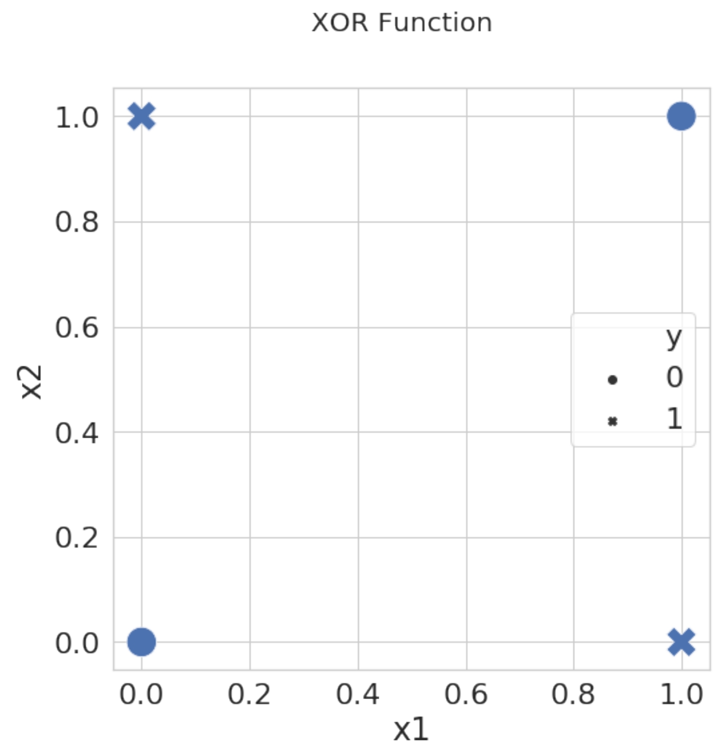
XOR Truth Table

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

- XOR is a basic logic binary operation
- XOR is not *linearly separable* (can’t draw a line to separate 0/1 classes)

Question:

- XOR is only defined on inputs with {0,1}, what should our network return for other inputs (e.g. $x_1 = 0.5, x_2 = 0.5$)?



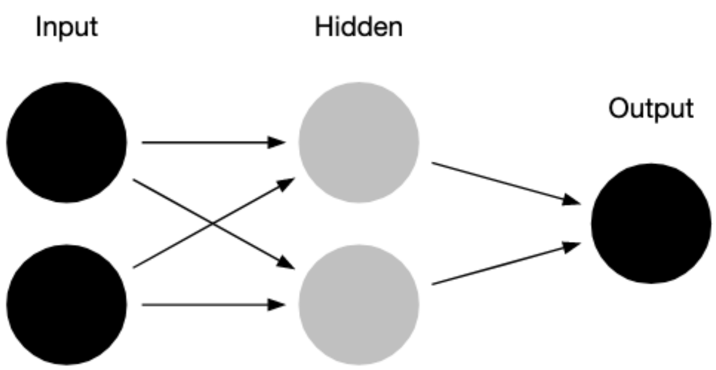
Example: Define Our Neural Network

Rotman

Write out as equations (using the RELU activation and linear output):

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^{\boldsymbol{T}} \max\left(0, \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{x} + \boldsymbol{c}\right) + b$$
$$= \begin{bmatrix} w_{2,1} & w_{2,2} \end{bmatrix} \max\left(0, \begin{bmatrix} w_{1,1,1} & w_{1,2,1} \\ w_{1,1,2} & w_{1,2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}\right) + b$$

Given training input data: $\boldsymbol{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \quad \boldsymbol{y} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$



Find values of $\boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b$ that minimize error (in this case mean squared error):

$$\sum_{x_i, y_i \in (\boldsymbol{X}, \boldsymbol{y})} (y_i - f(\boldsymbol{x_i}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b)^2$$

Example: Solving our Network

Rotman

Here's a solution that solves this problem:

$$W = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \qquad \boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \qquad \boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \qquad b = 0$$

Let's see how these weights (or parameters) solve the XOR problem by doing a "forward pass" of a neural network:

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^{\boldsymbol{T}} \max\left(0, \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{x} + \boldsymbol{c}\right) + b$$

First, compute the first weight matrix with our features $\boldsymbol{W}^T \boldsymbol{x}$ (we'll compute all four data points at once):

$$\boldsymbol{XW} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix}$$

Example: Solving our Network (continued)

Rotman

Next, add our constant bias $\boldsymbol{W^T x} + \boldsymbol{c}$:

$$\begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 1 & 1 \\ 2 & 2 \end{bmatrix} + \begin{bmatrix} 0 & -1 \\ 0 & -1 \\ 0 & -1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

Next, add apply the activation function (i.e. ReLU) $\max(0, \boldsymbol{W^T x} + \boldsymbol{c})$:

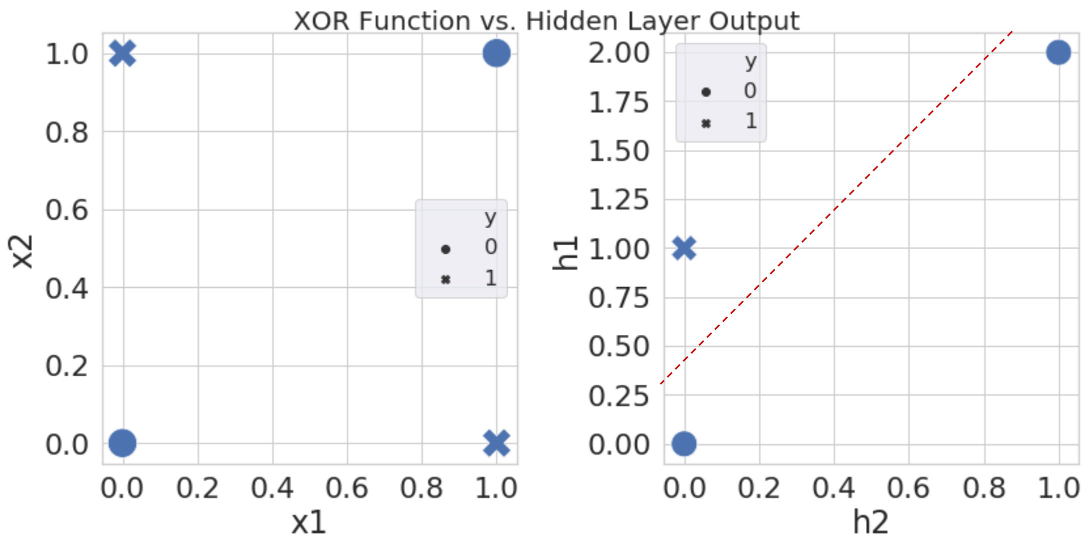
$$\max(\boldsymbol{0}, \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$$

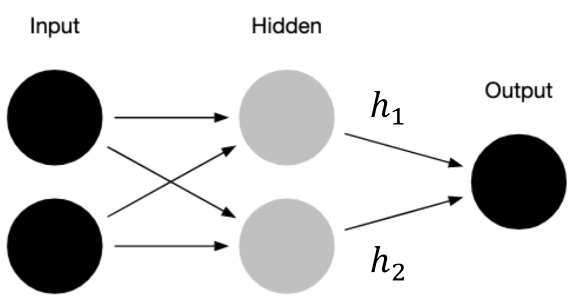
Finally, apply weights from the last layer $\boldsymbol{w^T} \max(0, \boldsymbol{W^T x} + \boldsymbol{c}) + b$:

$$\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -2 \end{bmatrix} + \boldsymbol{0} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Example: Illustration of Non-Linear Hidden Layers

Rotman





$$
\boldsymbol{H} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}
$$

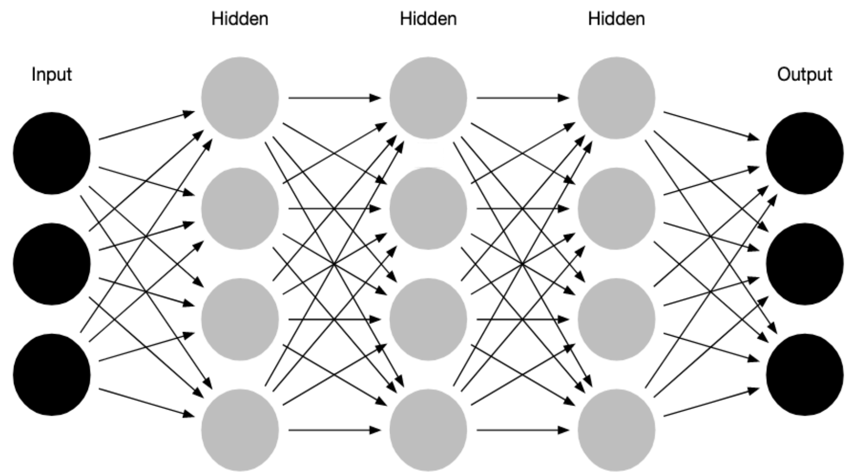
- Non-linear hidden layers transformed XOR problem in a linearly separable problem

Much More to Neural Networks…

Rotman

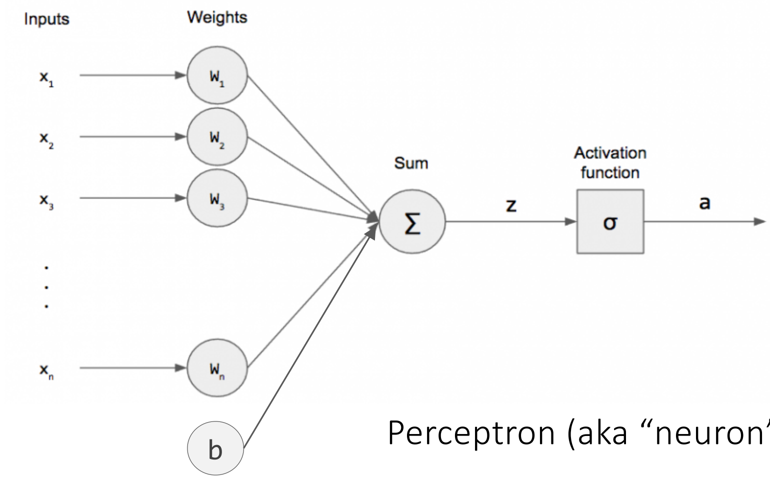
Fundamentals Concepts of Neural Networks

- Training Deep Neural Networks
  - Computation graphs and computing gradients
  - Optimization through Stochastic Gradient Descent
- Neural Network Architectures
  - Width, depth
  - Types of layers
  - Connectivity
  - Activation functions
  - Loss functions
- Tuning Deep Neural Networks
  - Overfitting vs. Underfitting
  - Regularization
  - Learning rate
  - Hyper-parameters tuning

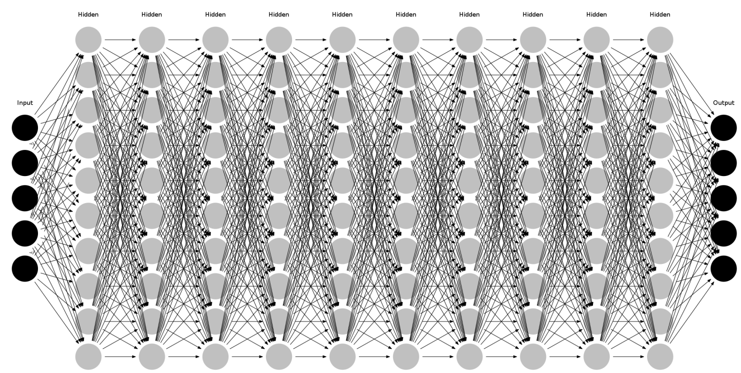


Summary of Neural Networks

- **Neural networks** are *big* function approximators with 100Ms of parameters
  - Allows them to approximate *very* complicated functions
- Feedforward Neural Networks are composed of simple functions called **perceptrons:**
  - Compute a linear combination of inputs and parameters with a non-linear activation function
- **”Deep” neural networks** (i.e. deep learning) are just neural networks with many layers
- Neural networks have only recently (~10 years) become very useful because of exponential growth in computing power (and correspondingly data)



Perceptron (aka “neuron”)



”Deep” Neural Network

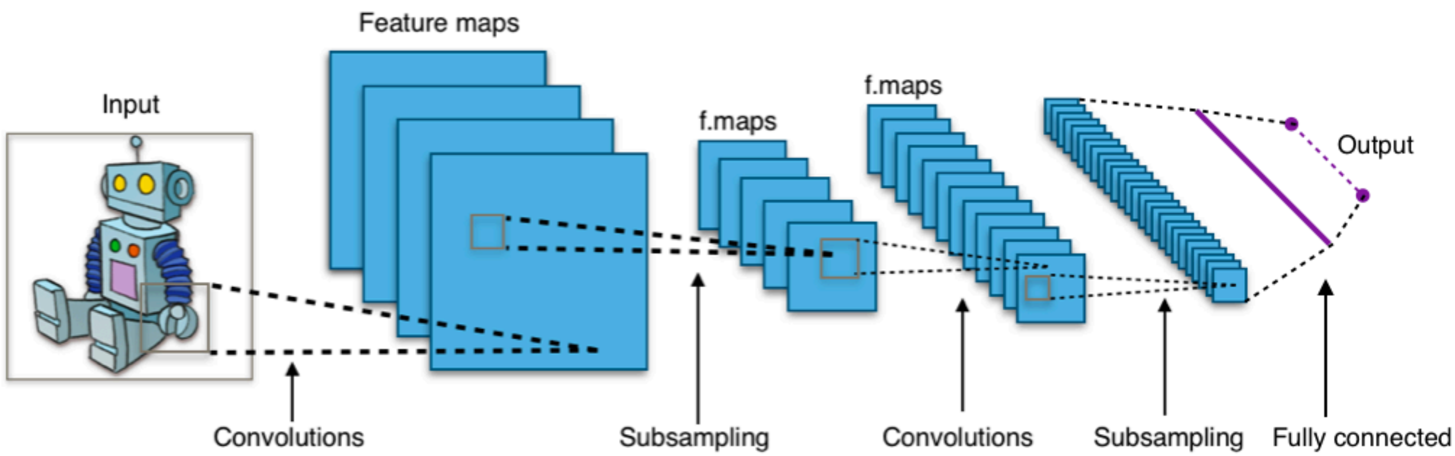
Rotman

Deep Learning and Beyond

Convolutional Neural Networks

Rotman

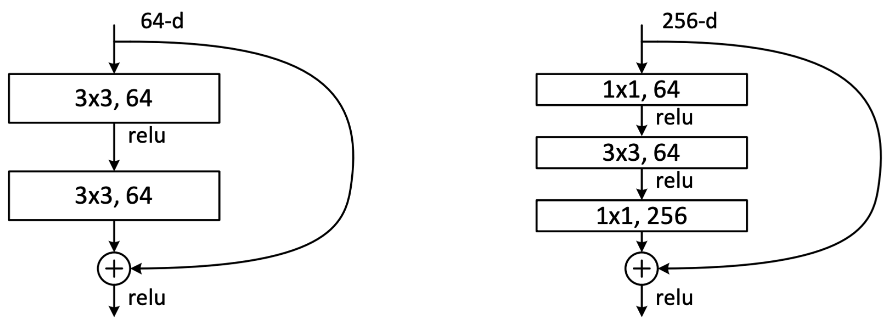
- CNNs are composed on multiple successive layers of convolutional layers and fully-connected layers
  - Backbone of modern image processing ML to produce state-of-the-art results
  - Convolutional layers are used to extract features from the original input size
  - Final few layers are usually FC layers to produce output (e.g. image classification)
  - Non-linear activation functions are implicit in diagrams



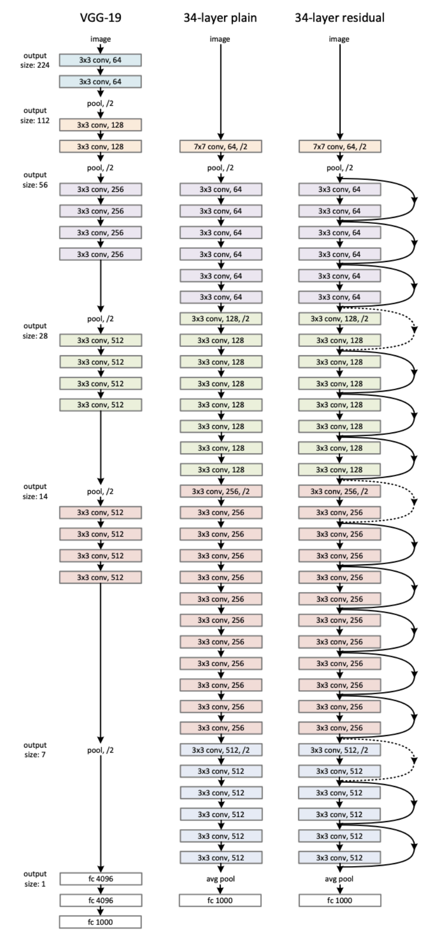
Source: https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

“Deep” Neural Networks (“ResNet”)

- He, Zhang, Ren, Sun, “Deep Residual Learning for Image Recognition”, 2015
    - Achieved 4.49% top-5 error rate; 60M parameters
    - https://arxiv.org/abs/1512.00567
- Introduced concept of “Residual Learning”:
    - Try to learn H$(x) - x$ (left branch) instead of directly learning H(x)
    - Adds “shortcut” connections that are added back to the CNN operations
    - Residual blocks enable “deep” networks up to 152 layers



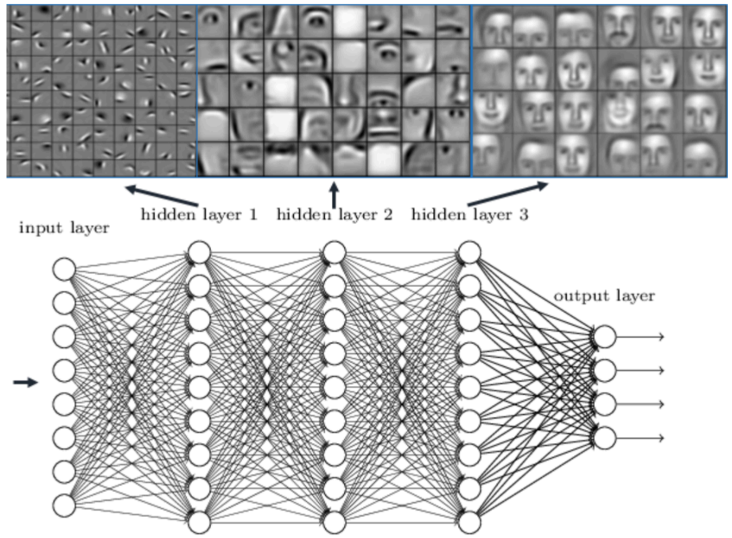
Two Different Residual Blocks



Representation Learning

Rotman

- ***Representation Learning*** (or feature learning) is a broad set of techniques to automatically discover the representation needed for a specific task from raw data.
  - **Supervised techniques**: Use labelled data to infer the right feature representation (e.g. deep nets)
  - **Unsupervised techniques**: Use unlabeled data to infer (usually compact) representations of the data
- Examples:
  - Images & CNNs: Each successive convolutional layer is a "feature map", forming a hierarchy of features
  - Word embeddings (shown later): Representing words as a high-dimensional continuous real vector
  - Dimensionality Reduction e.g. Principal Component Analysis (PCA)

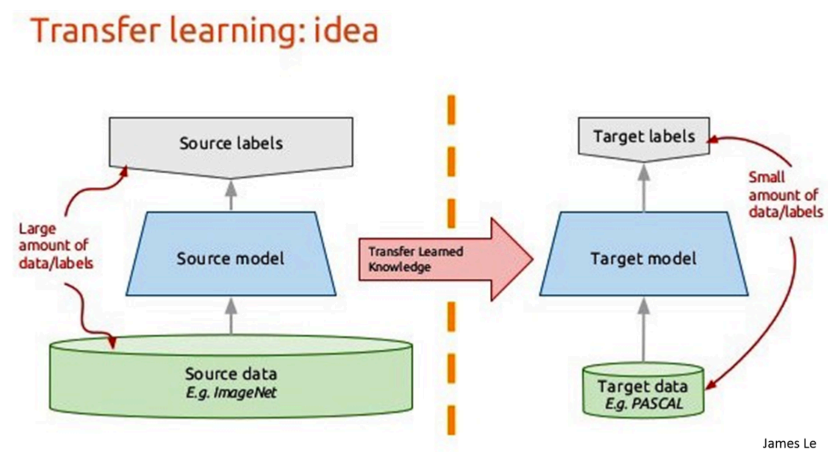


Source: https://cse.iitk.ac.in/users/piyush/courses/ml_autumn16/771A_lec23_slides.pdf

Transfer Learning

Rotman

- ***Transfer Learning***: transfer "knowledge" learned in one or more ML source tasks to improve performance on a related target task
  - Labelling data is expensive, thus most data is unlabeled
  - Utilize the large base of labelled data to improve performance on tasks with small amounts of data
- Example:
  - Task: Classifying male/female dolphins images
    - Not much labelled data, expensive to obtain
  - Pre-train a deep net network (e.g. ResNet-152) on ImageNet (14M labelled images)
  - Use the pre-trained network to generate features from existing dolphin images
  - Train a new classifier using these new features

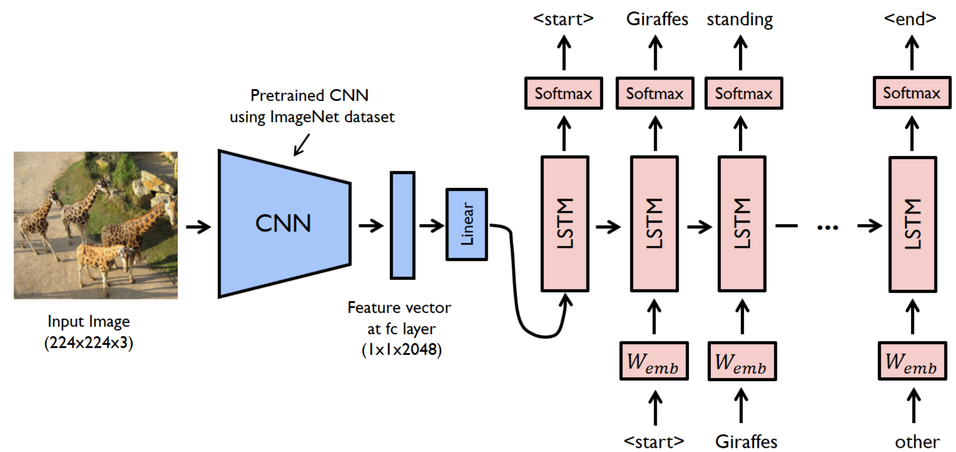


https://www.datasciencecentral.com/profiles/blogs/transfer-learning-deep-learning-for-everyone

Modeling Sequence Data with RNNs

- **Recurrent Neural Networks** (RNN) are a family of neural networks for processing sequential data
- Key RNN ideas:
  - Neural networks have a time varying *state* (h_t)
  - Utilize the *same* function (aka "*RNN cell*") $f(\ldots)$ to map h_t, x_t to h_{t+1}
  - Computational graph is *unfolded* through time to perform stochastic gradient descent
- Applications:
  - Time-series problems (e.g. stock prices, sales forecasting)
  - Video / audio / text prediction tasks (e.g. image caption generation, machine translation etc.)

Image Captioning Architecture

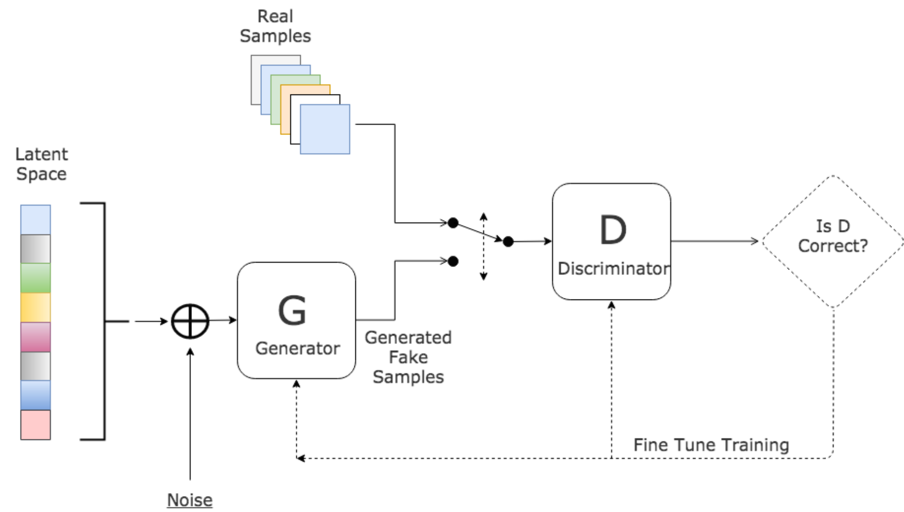


Source: https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/

Generative Adversarial Networks (GAN)

Rotman

- GANs are composed of two deep nets:
  - ***Generator*** network which generates fake samples of the data conditioned on latent space (random noise)
  - ***Discriminator*** network which attempts to tell the real samples from the fake (generator) samples
- The generator and discriminator play an **adversarial game** where:
  - Generator tries to fool the discriminator; and
  - Discriminator tries to guess that the data is fake
- The generator and discriminator networks alternate updating via SGD in attempt to "win" via a two player minimax game
- Applications:
  - Image / video / audio synthesis
  - Text generation



GAN Architecture

$$\min_G \max_D \begin{array}{l} V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + \\ E_{z\sim p_z(z)}[\log\left(1 - D\left(G(Z)\right)\right)] \end{array}$$

Source: https://stats.stackexchange.com/questions/277756/some-general-questions-on-generative-adversarial-networks