Deep Learning in R

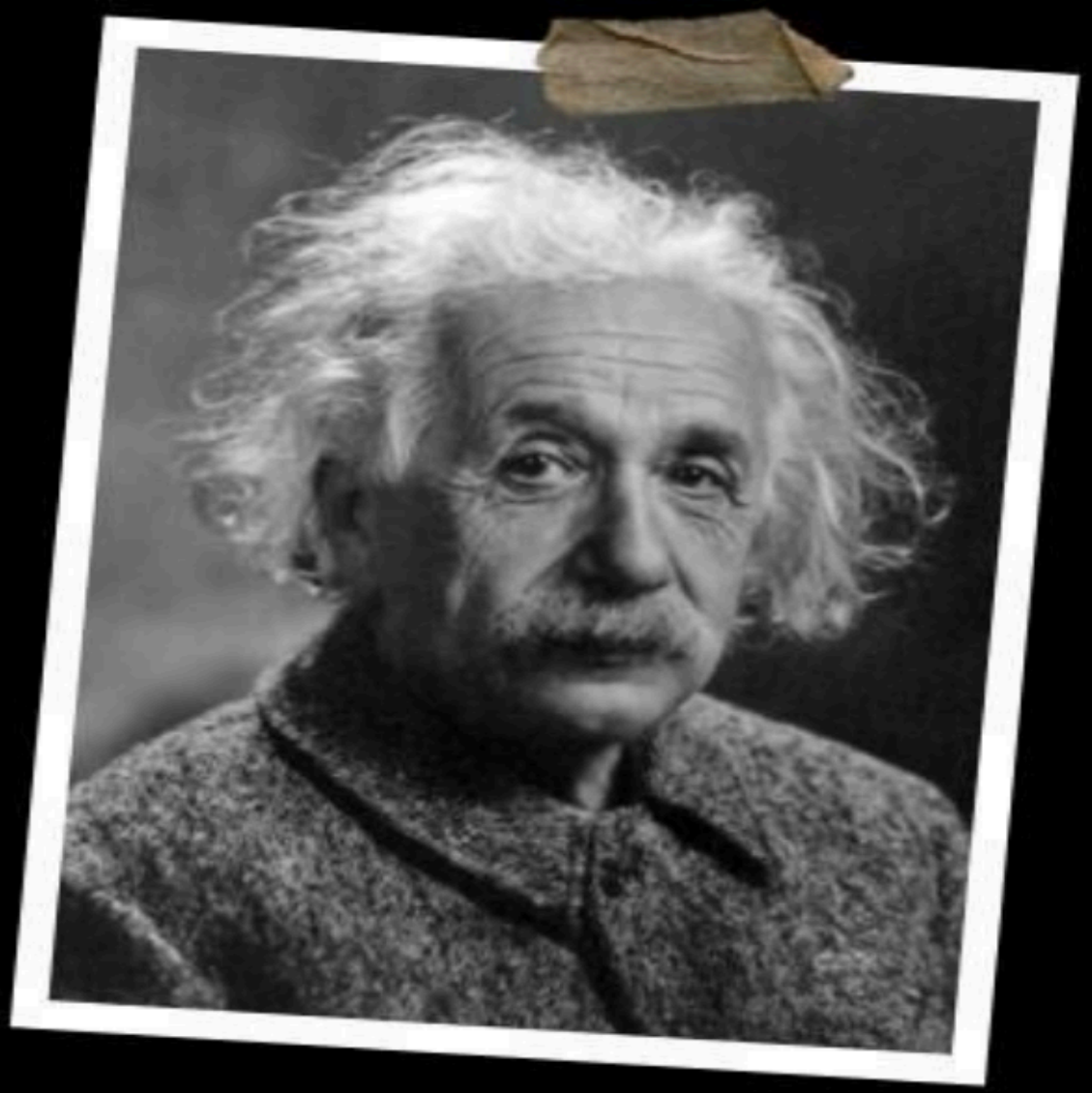
Kamal Mishra
08-Jun-2019

Disclosure:

Content is based on my experiences and do not intend to represent any future views of any organization or individuals

Motivation

"Education is not the learning of facts, but the training of the mind to think."
-Albert Einstein



Disclaimer

Disclaimer – The contents of this document are to best of my knowledge and experiences. Some data and names MAY BE tweaked to take care of data privacy and business sensitivity aspects. The information provided is purely to highlight experience gathered with clear business impact created as part of opportunities and NO WAY RELATES TO ANY ORGANIZATION.

Safe Harbor

This document contains forward-looking statements within the meaning of Section 27A of the Securities Act of 1933, as amended, and Section 21E of the Securities Exchange Act of 1934, as amended. The forward-looking statements contained herein are subject to certain risks and uncertainties that could cause actual results to differ materially from those reflected in the forward-looking statements. I undertake no duty to update any forward-looking statements.

Content

- Deep Learning
  - Background
  - Why Deep Learning and Trends
  - Similarity to Human Brain / Neural networks
  - Use cases / Applications
  - Image and feature representation
  - Layers – Input, Output and others
  - Example of usage in CIFAR10 dataset
- Deep Learning Libraries in R
- References

Disclosure:

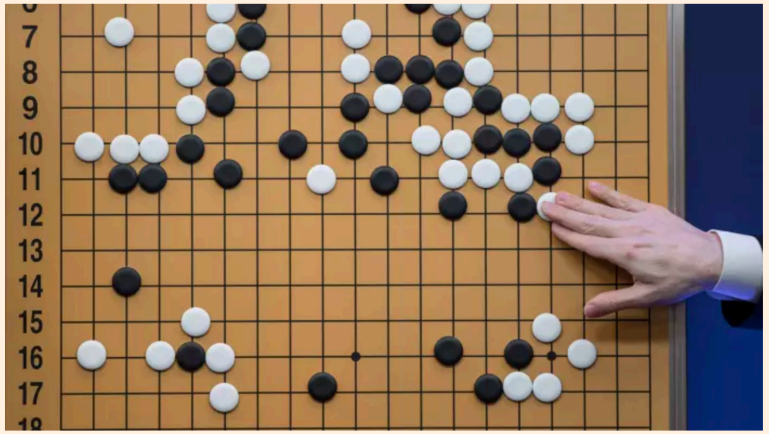
Content is based on my experiences and do not intend to represent any future views of any organization or individuals

Deep Learning..

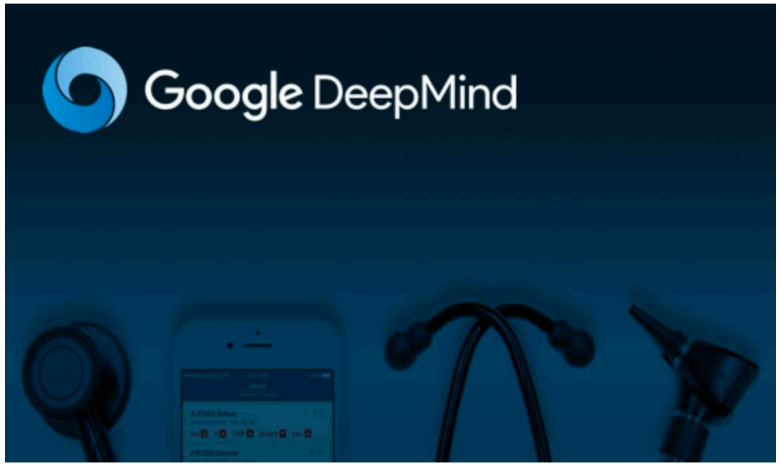
AlphaGo marks stark difference between AI and human intelligence

Google's robot relied on processing power and data storage, writes Daniel Susskind

DANIEL SUSSKIND + Add to myFT



Google Deepmind trial to improve cancer treatment



Google's DeepMind AI gains on human oncologists in planning radiation cancer treatments

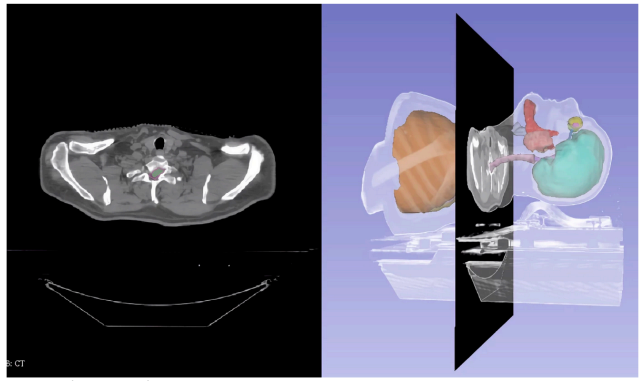
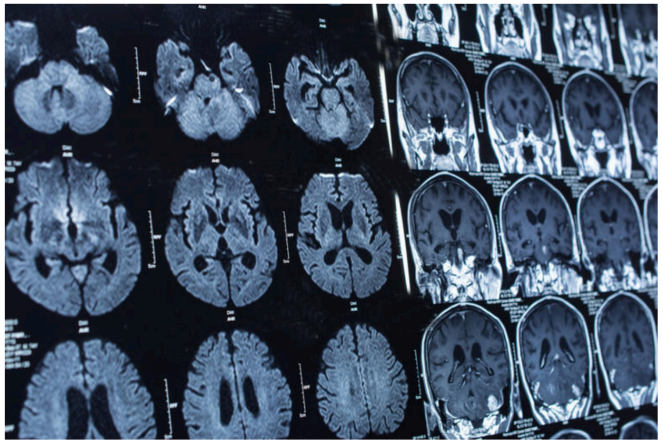


Image Credit: DeepMind

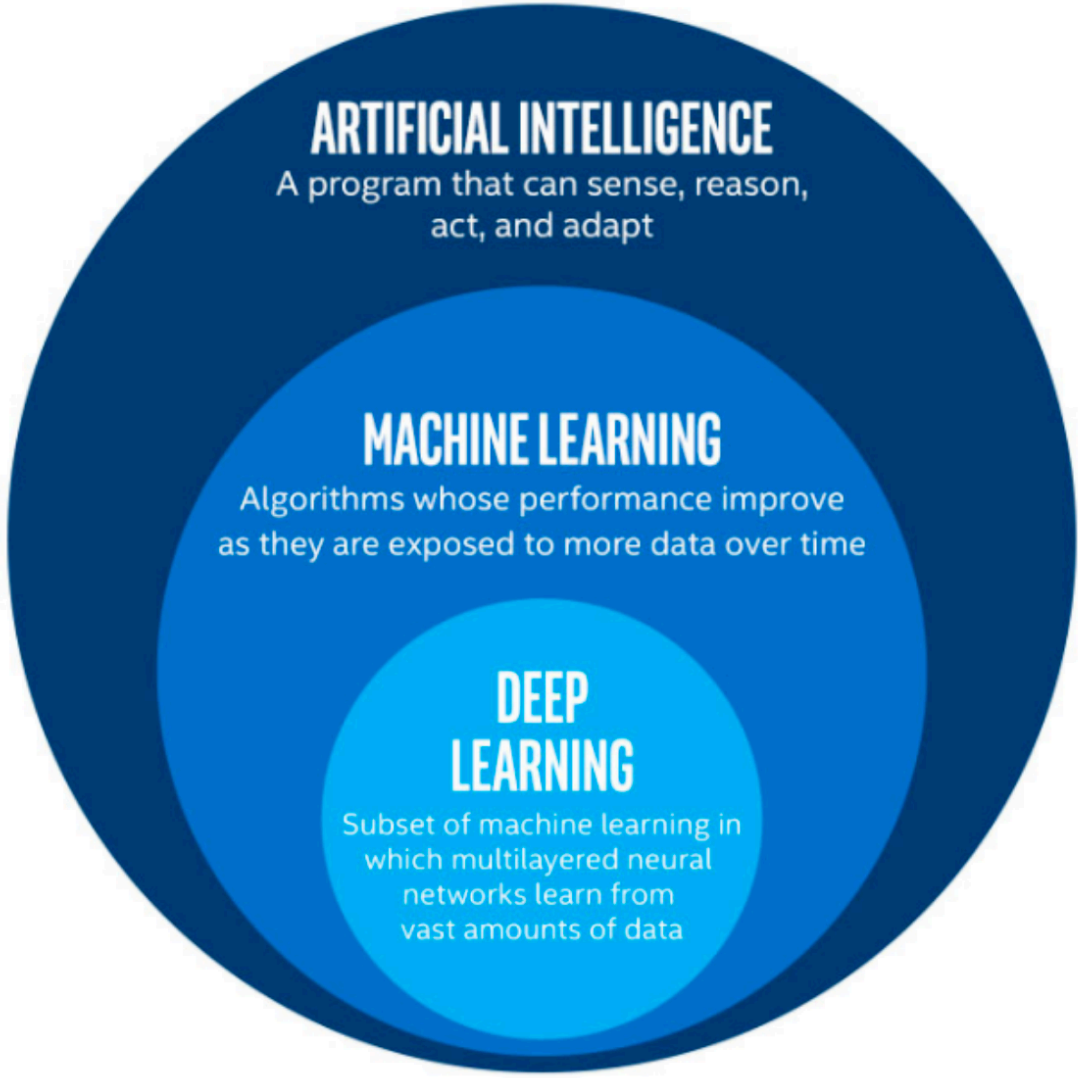
Google DeepMind AI to help doctors treat head and neck cancers

A new collaboration between DeepMind and the UK's National Health Service hopes to cut radiotherapy planning times from four hours to just one.

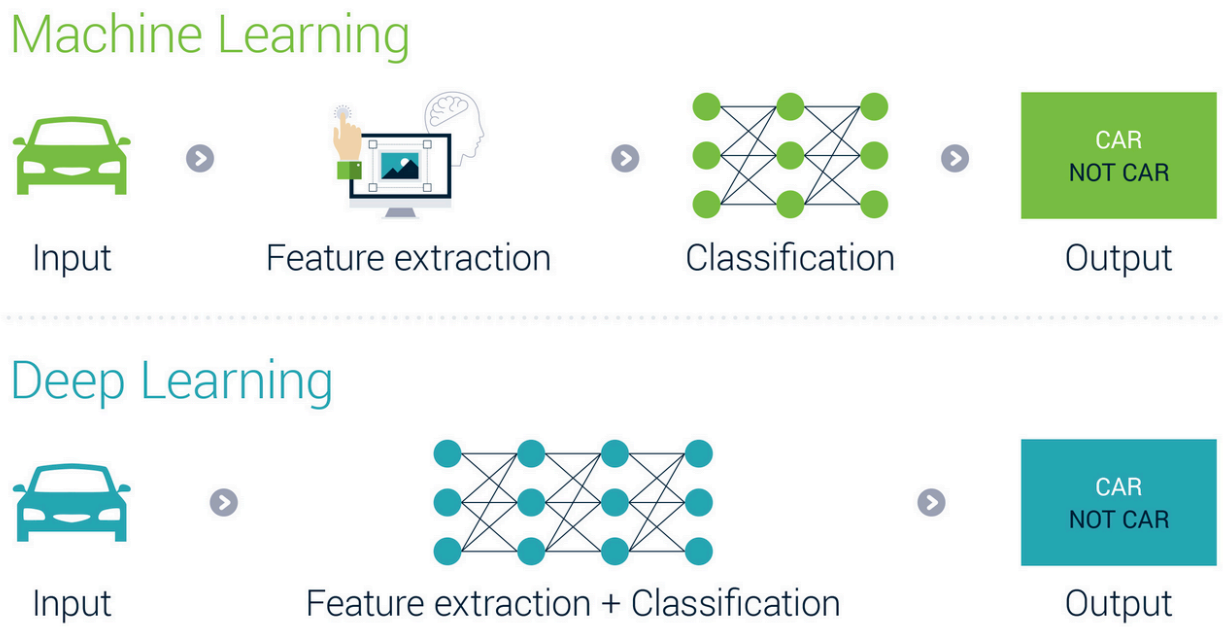


Google's DeepMind is aiming to help cut the time spent identifying key areas to treat, and avoid, in radiotherapy.

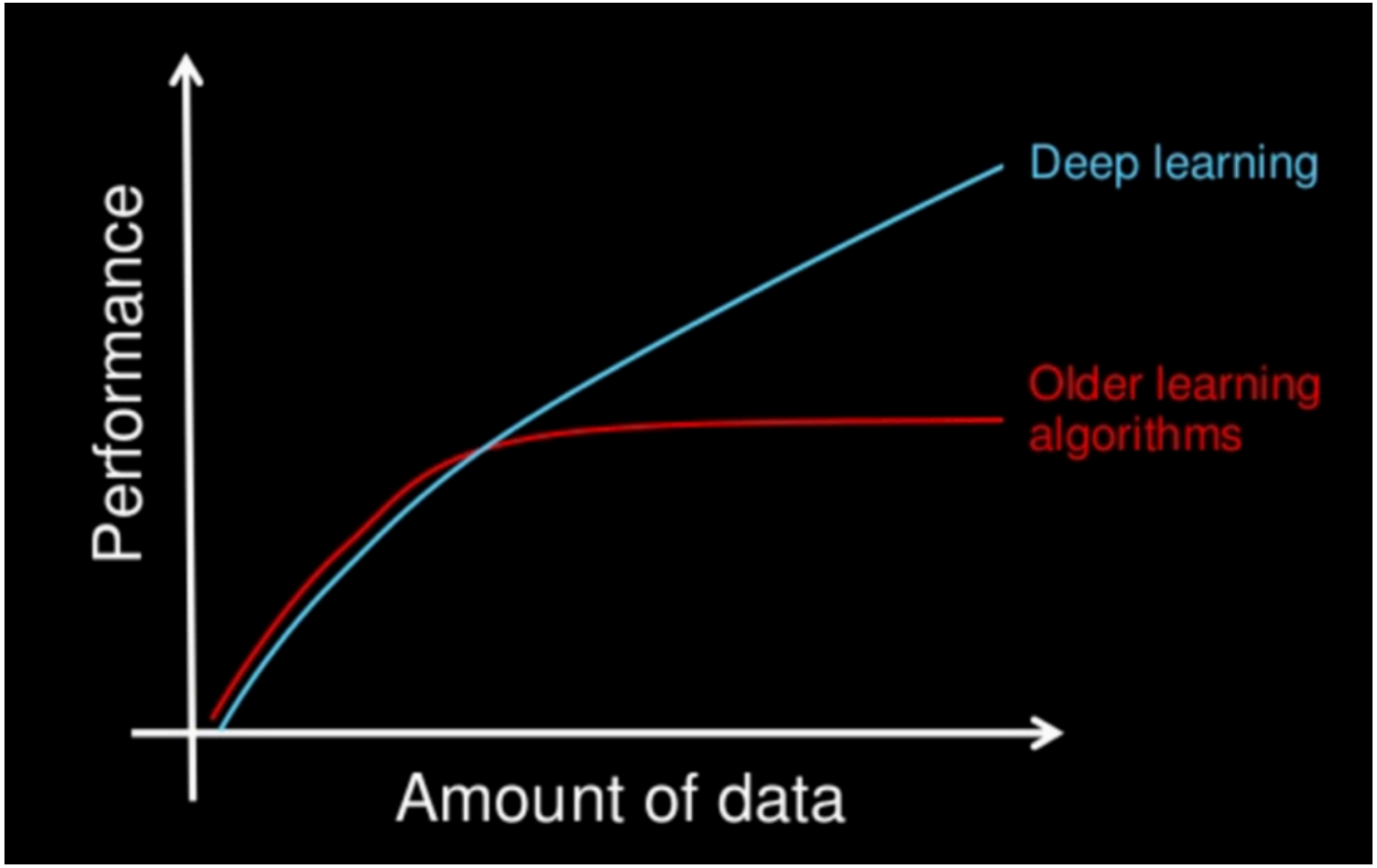
Deep Learning – subset of ML



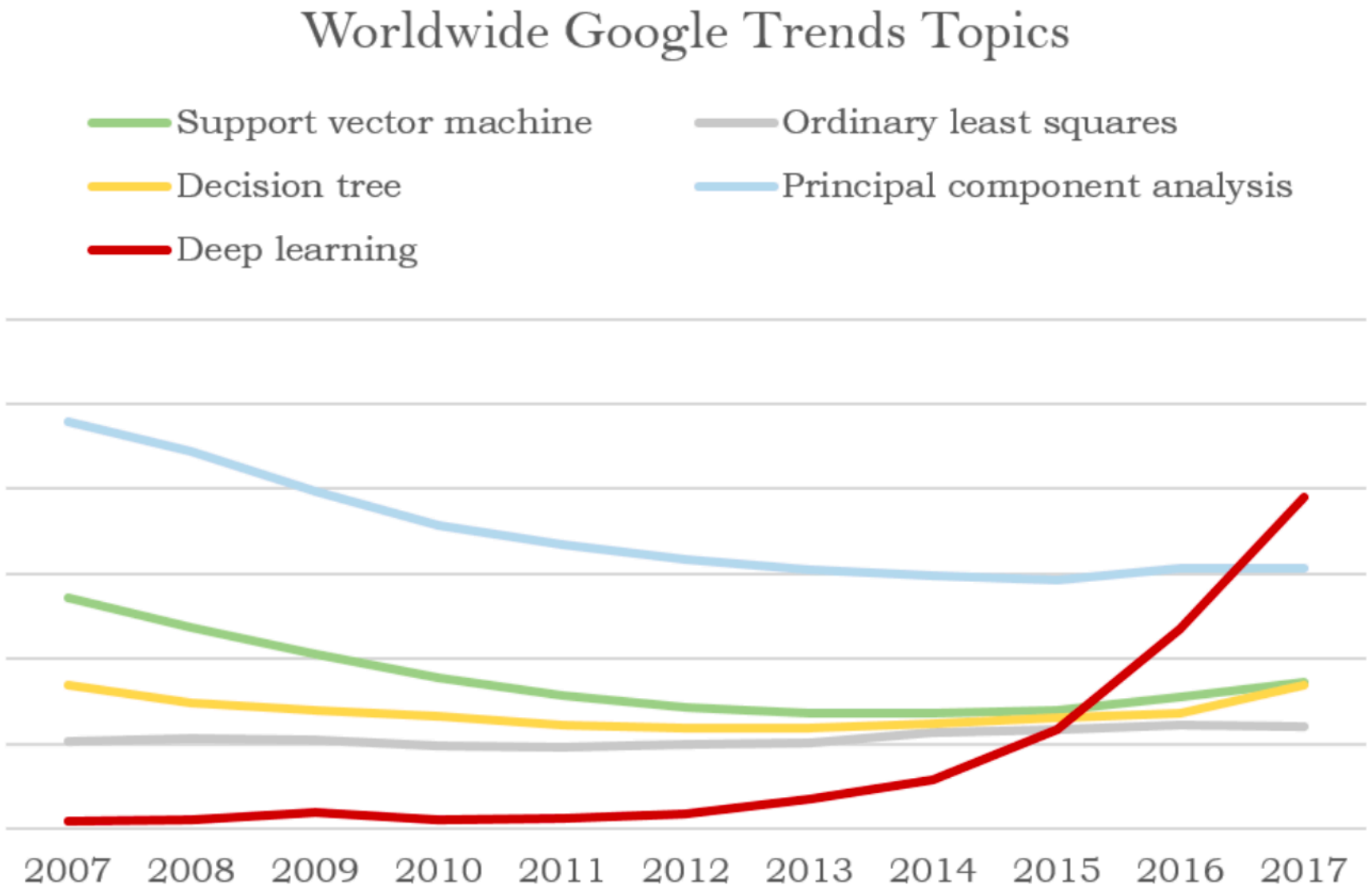
A crude example...



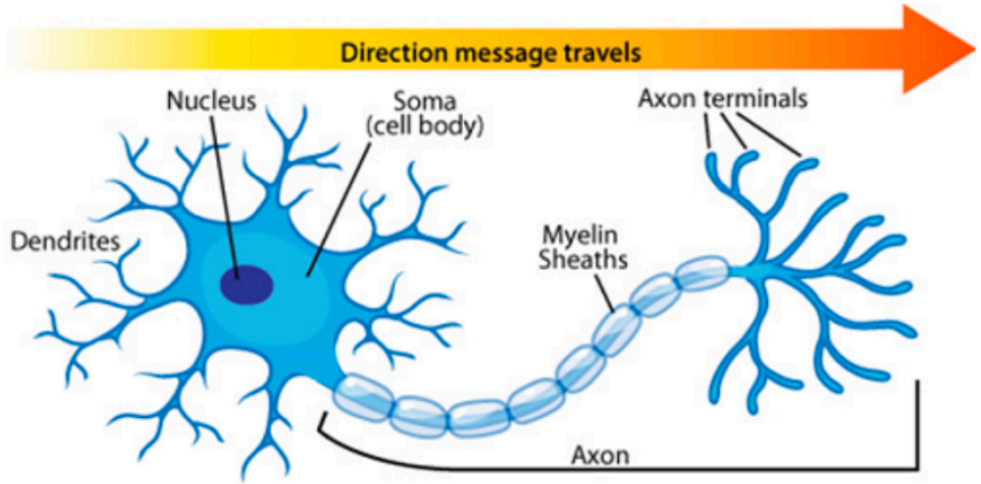
Why Deep Learning?



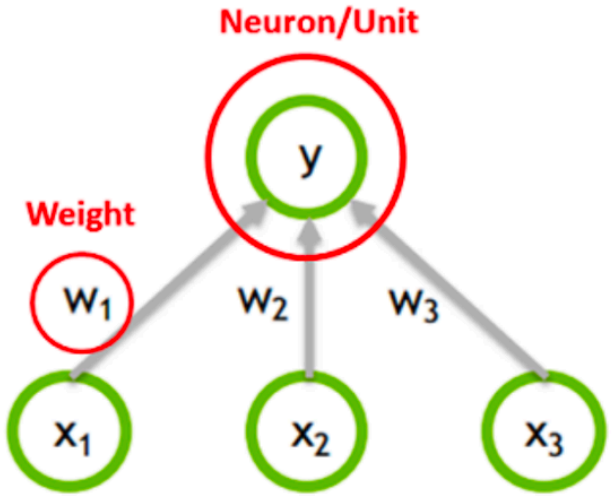
Trends


Worldwide Google Trends Topics
Support vector machine
Ordinary least squares
Decision tree
Principal component analysis
Deep learning
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

Human Brain



Biological Neuron



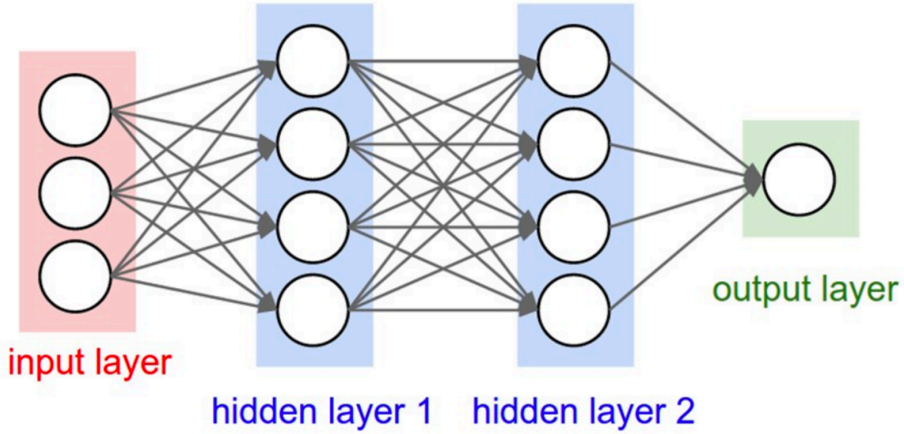
$$y=F(w_1x_1+w_2x_2+w_3x_3)$$

$$F(x)=\max(0,x)$$

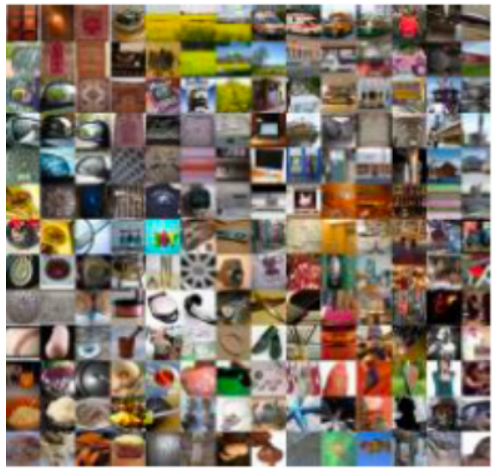
Artificial Neuron

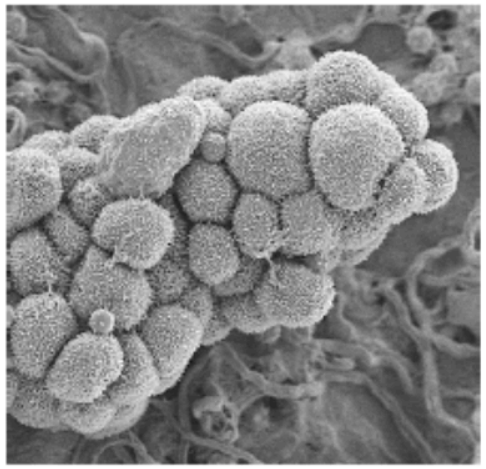
Neural Networks

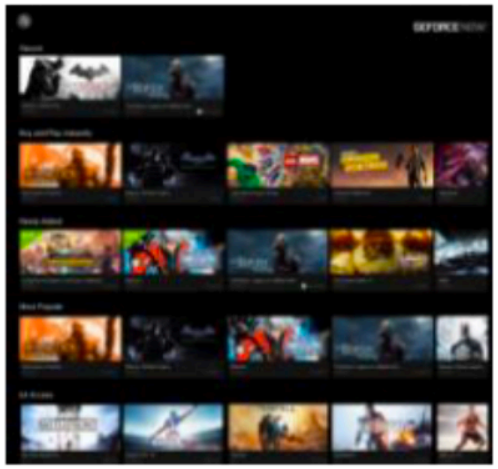
- Deep Learning is primarily about Neural Networks
- Interconnected nodes and edges
- Designed to perform complex tasks
- Neural Nets are highly structured networks and have three kind of layers – input, hidden and output layers [hidden layer is any layer(s) between input and output]
- Each node (called as neuron) in the hidden and output layers has a classifier

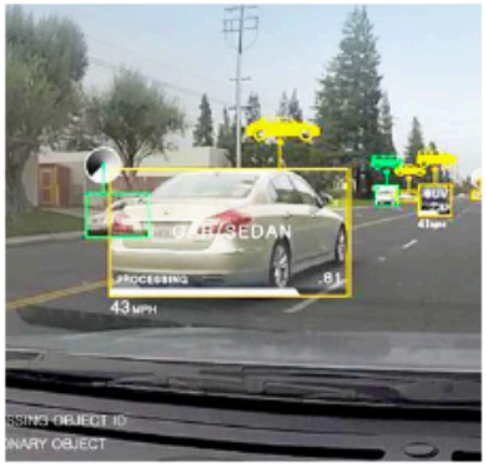


Typical applications









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

Source: developer.nvidia.com/deep--learning--courses

Deep Learning – Pros and Cons

Pros	Cons
• Robust – No need to design features ahead of time. Features are automatically learned to be optimal for task at hand. Natural variations in large data is automatically handled and learned. • Generalizable – The same neural net approach or method can be used for many different applications and data types • Scalable – Performance improves with more data • Parallelizable – Method supports massive parallelism	• Requires a large data set and hence longer training time. • Black Box. Learned features are very complex to understand. Many vision features are also not human interpretable. • Requires good understanding of how to model multiple aspects with different tools.

Why is it so complex?

Machine Learning

Input: X

Output: Y

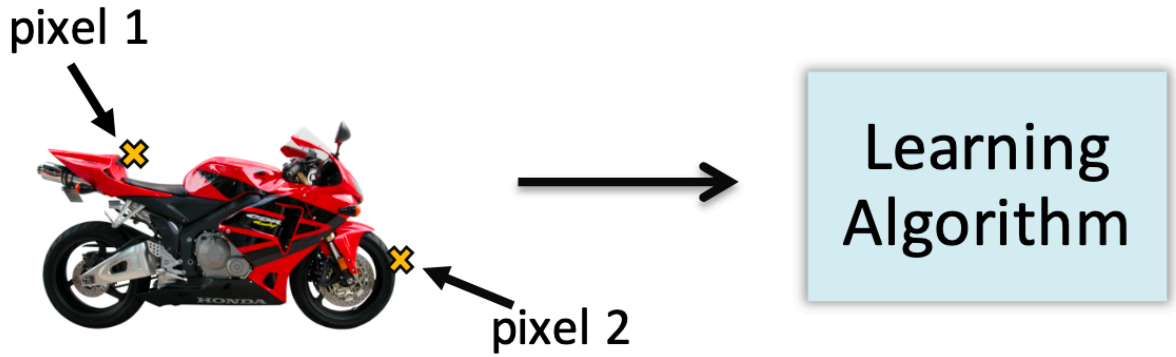
Label"motorcycle"

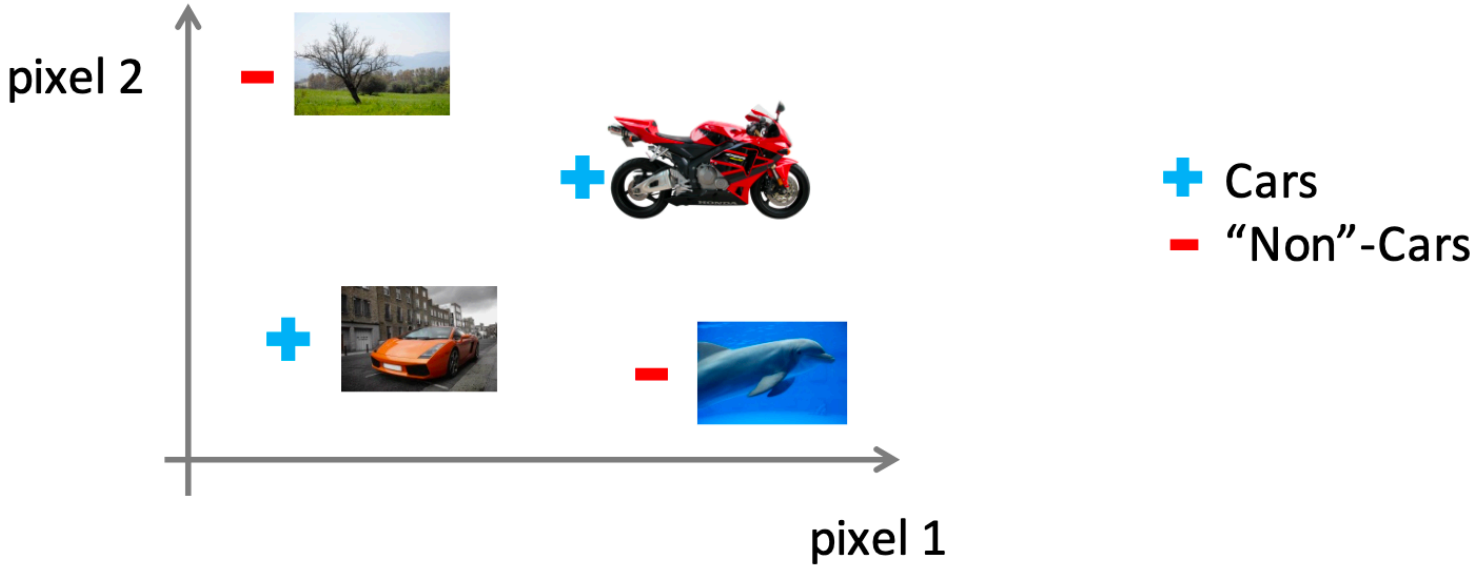
Deep Learning

But the camera sees this:

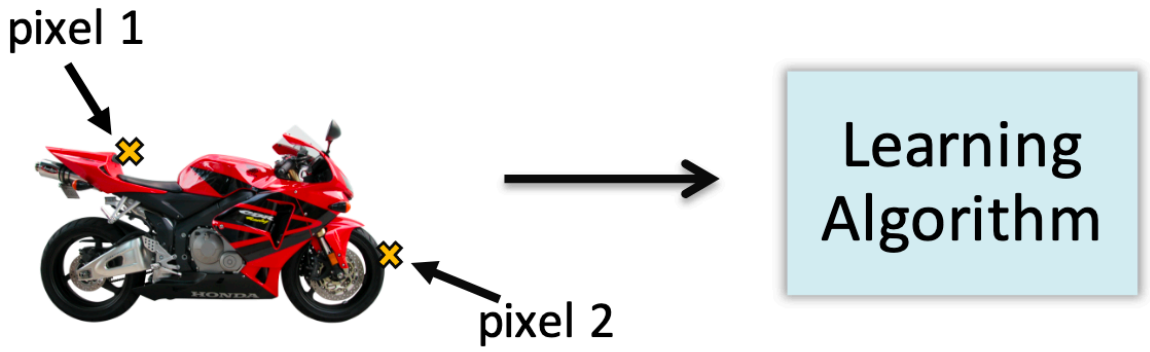
194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

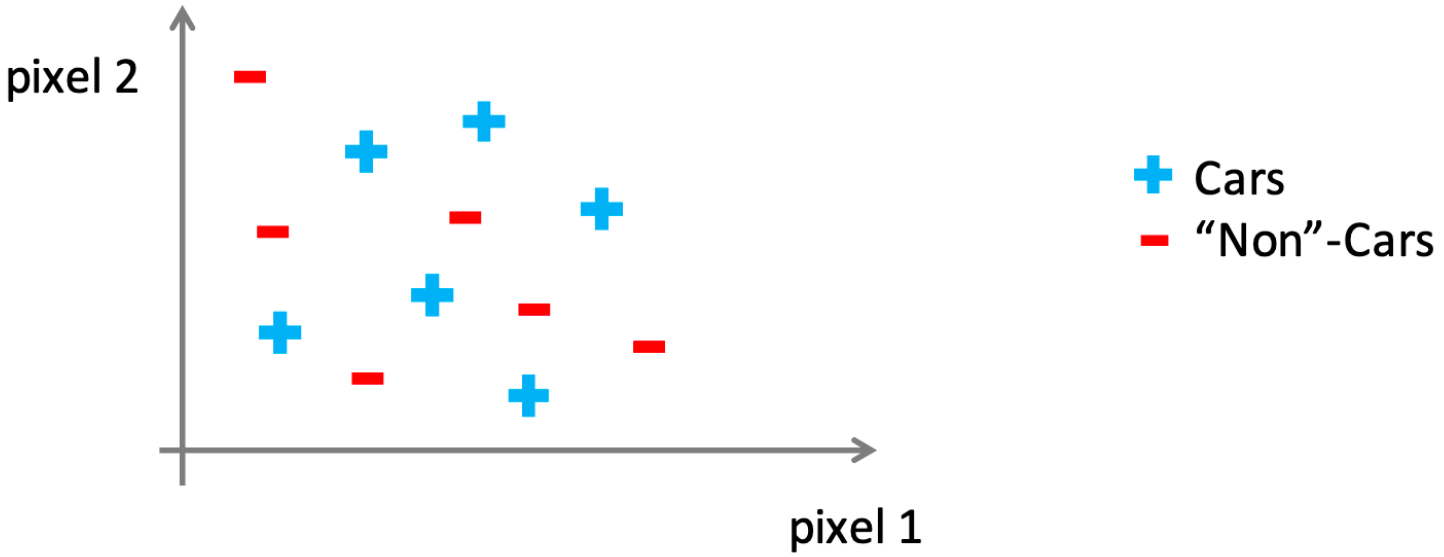
Raw image representation



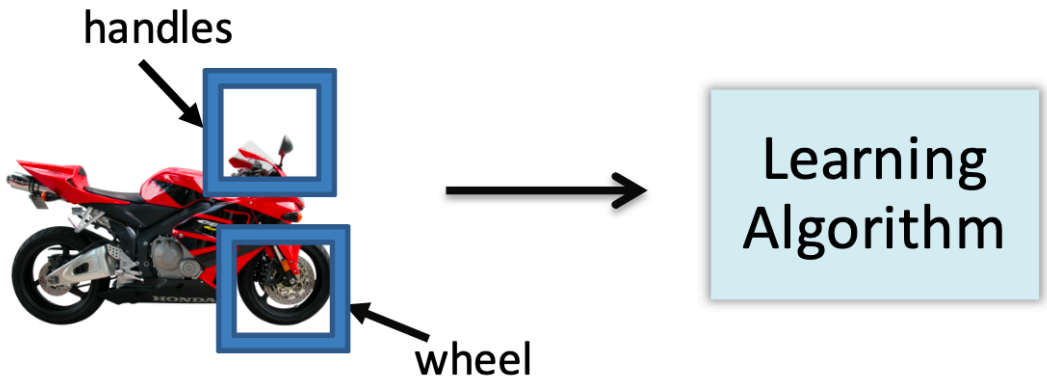


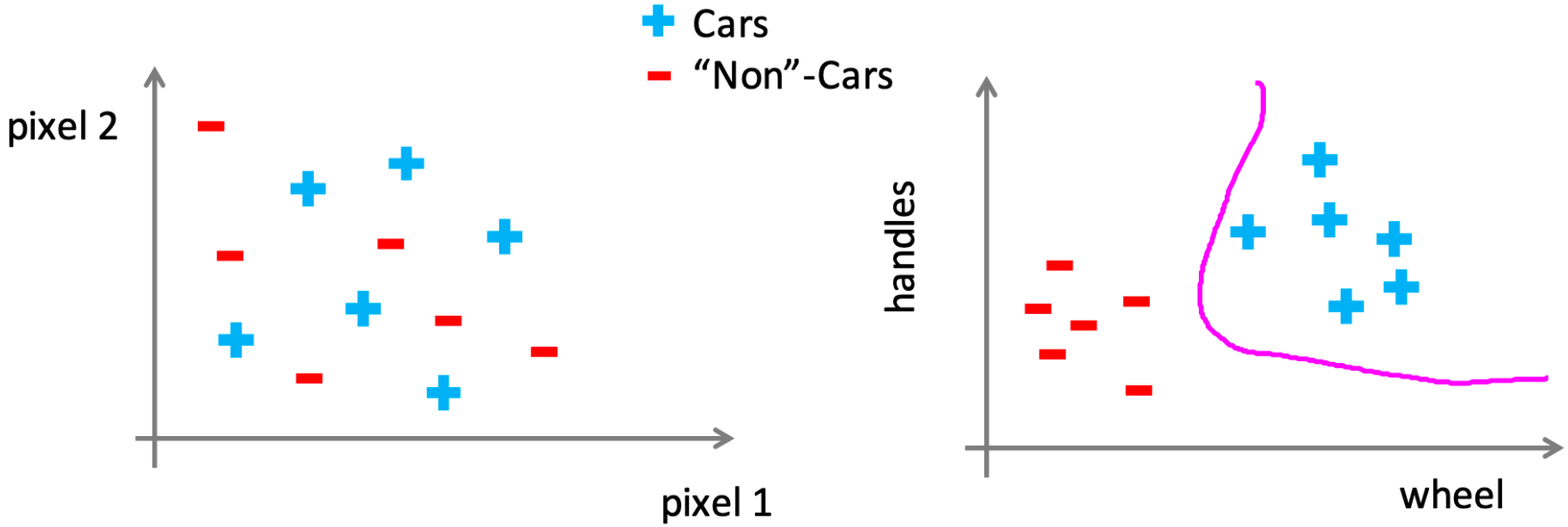
Raw image representation



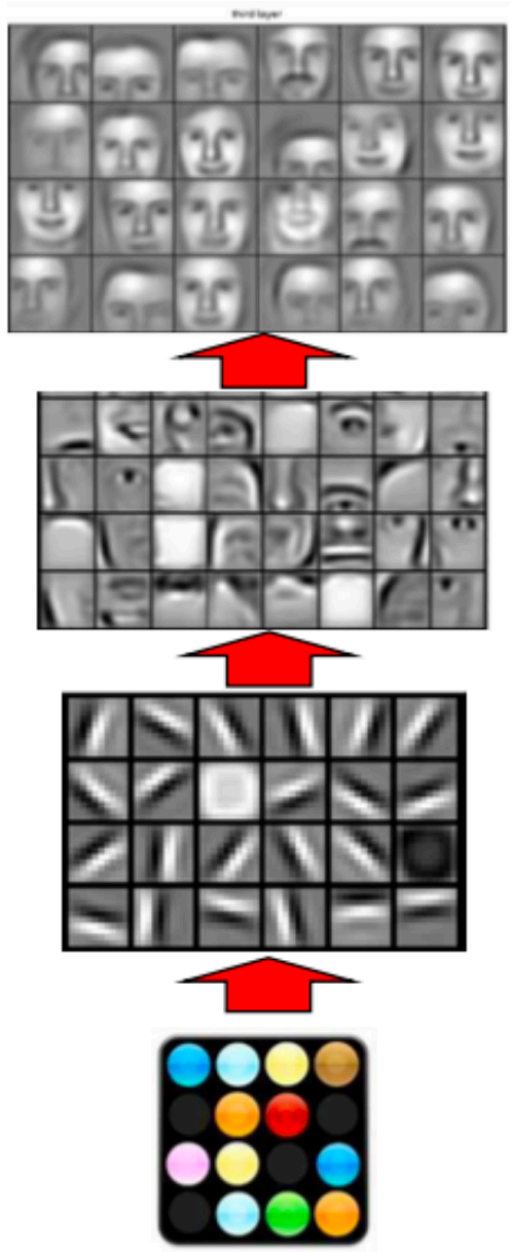


Better feature representation





Learning feature representations



object models

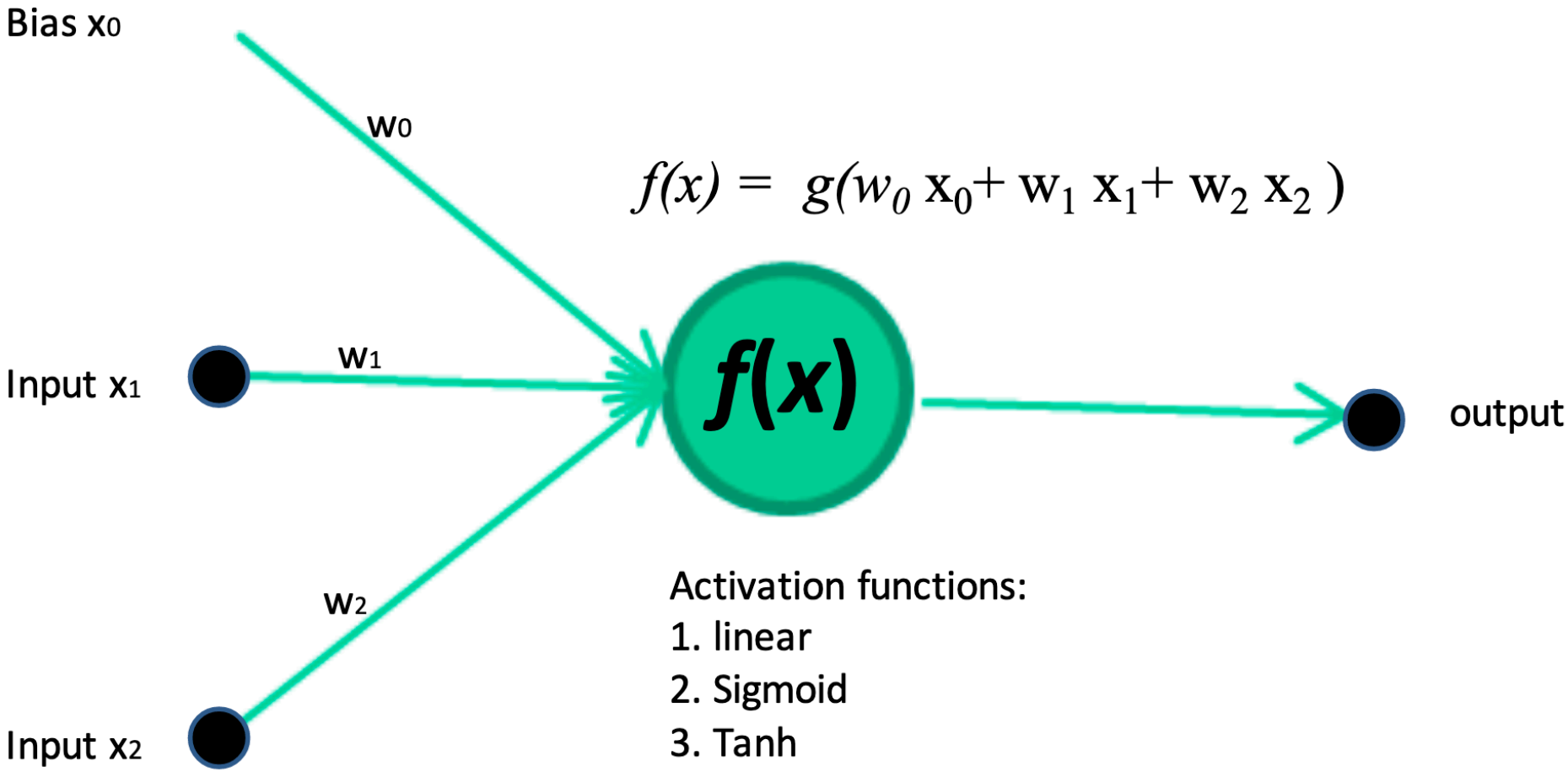
object parts (combination of edges)

edges

pixels

Source: Stanford University

Input, Bias, Weight, Activation Function, Output



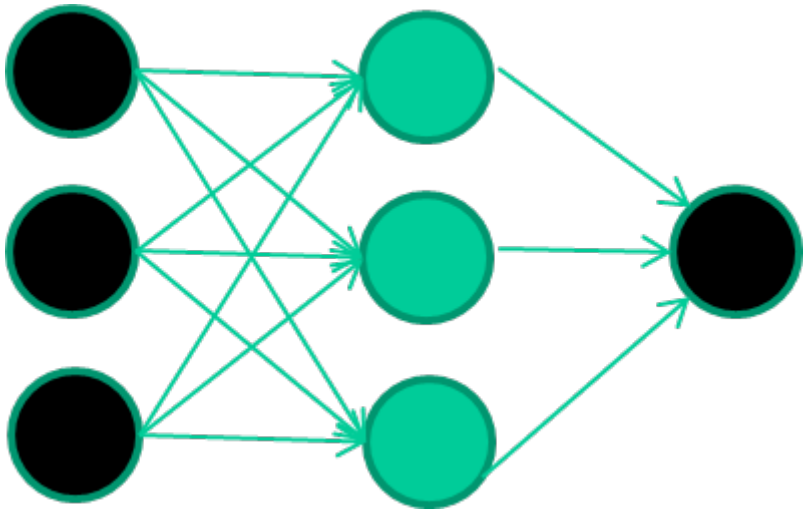
Activation functions:

1. linear
2. Sigmoid
3. Tanh
4. Relu
5. Softmax

etc.

A dataset

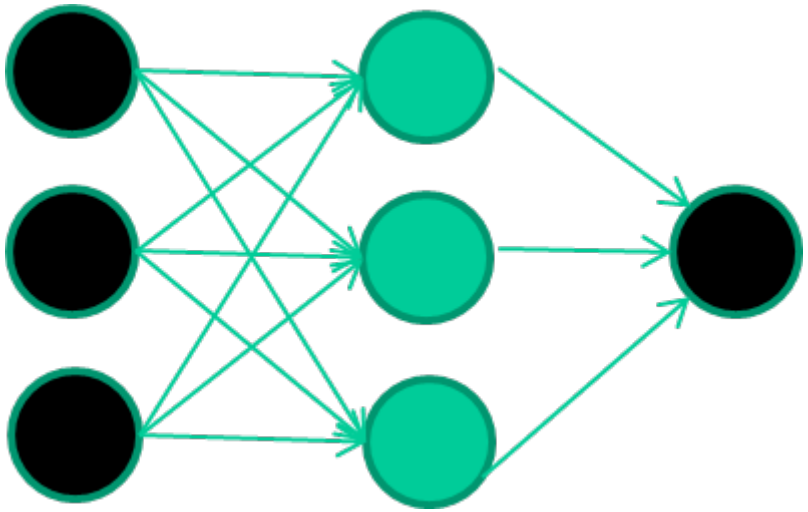
Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			



Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training the neural network

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

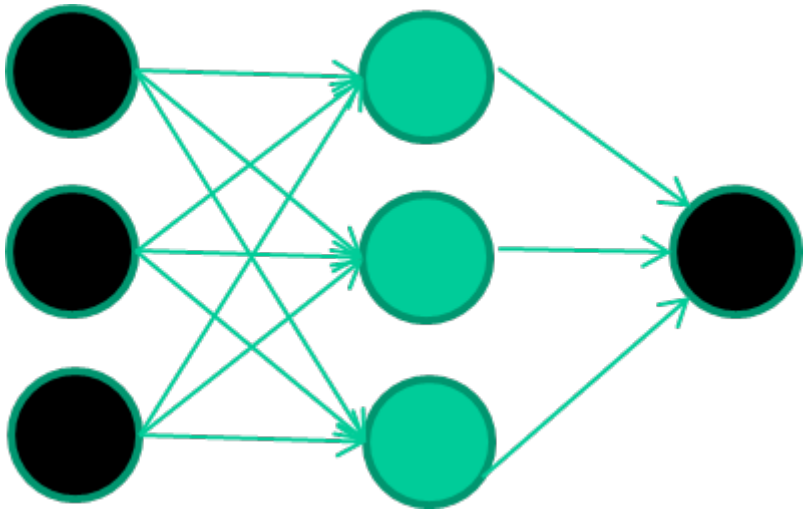


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Initialise with random weights

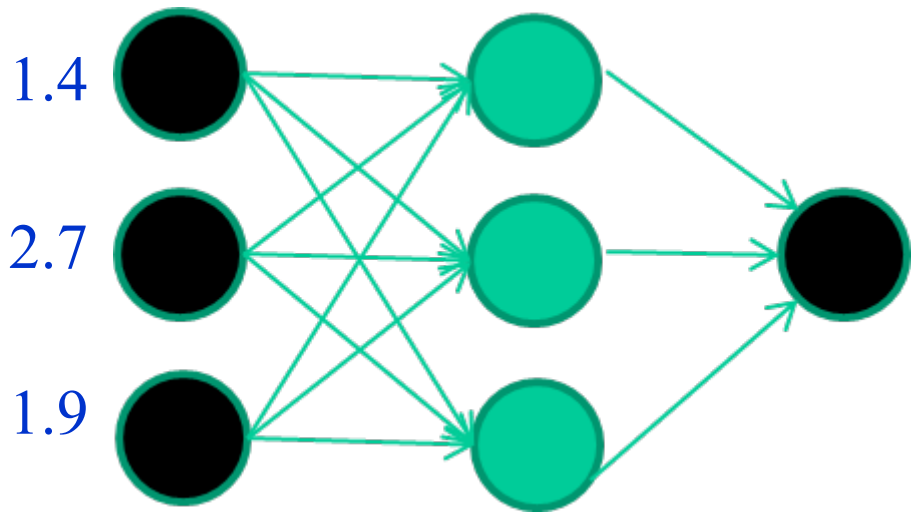


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern

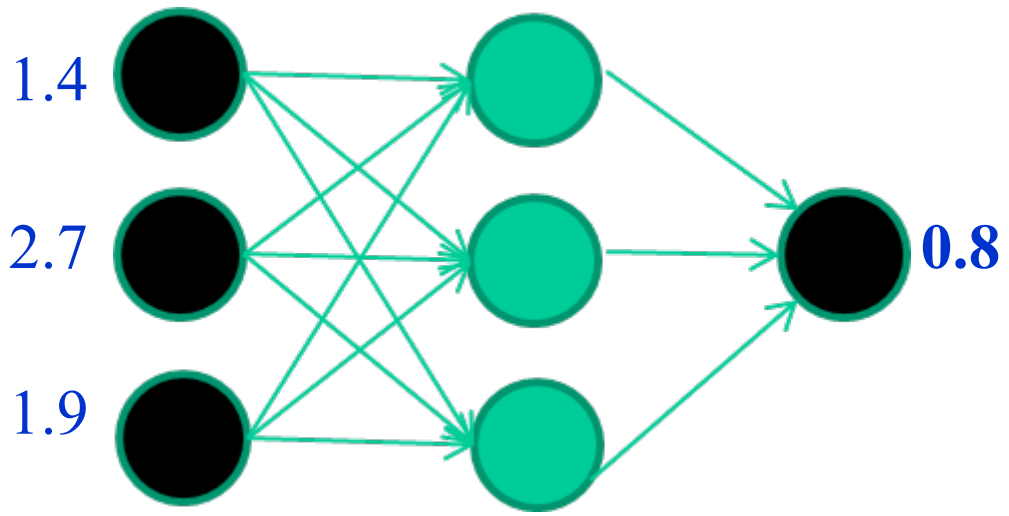


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output

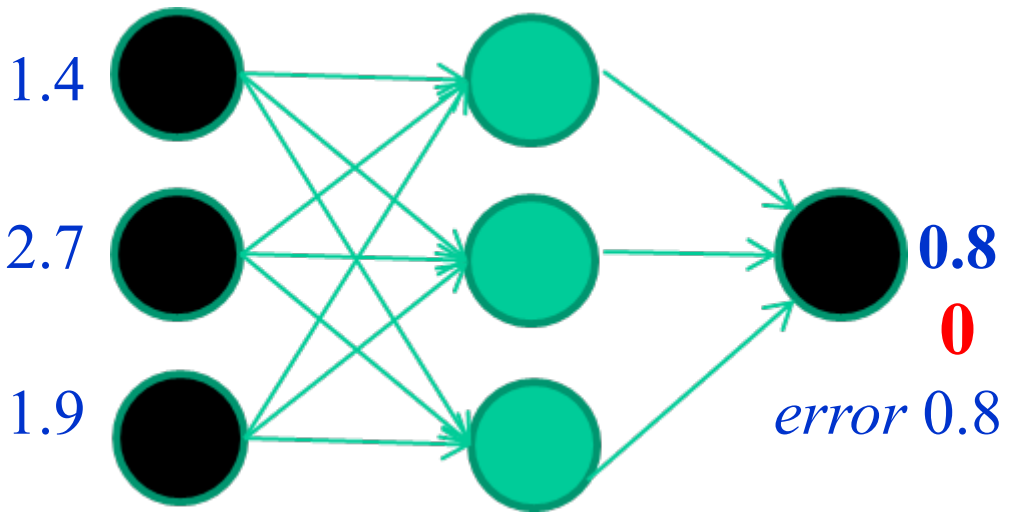


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output

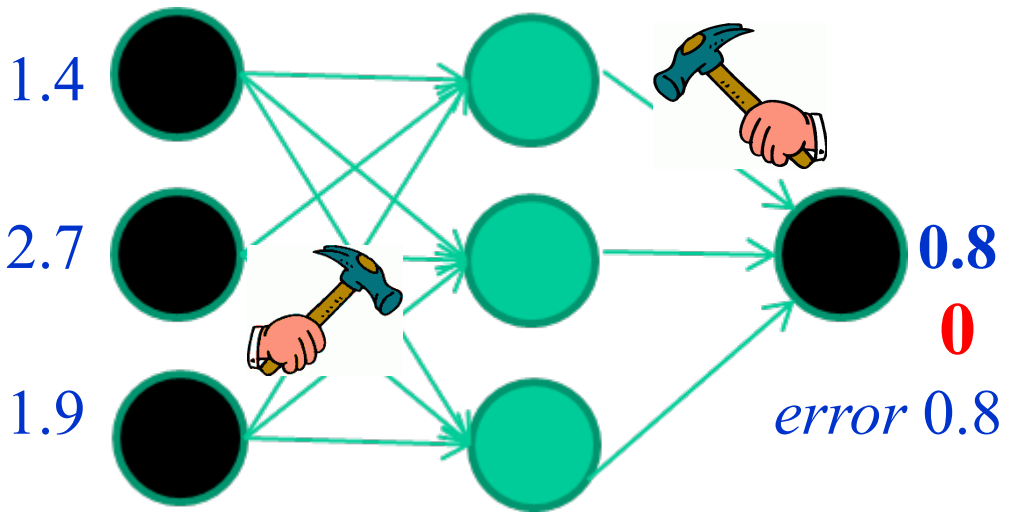


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

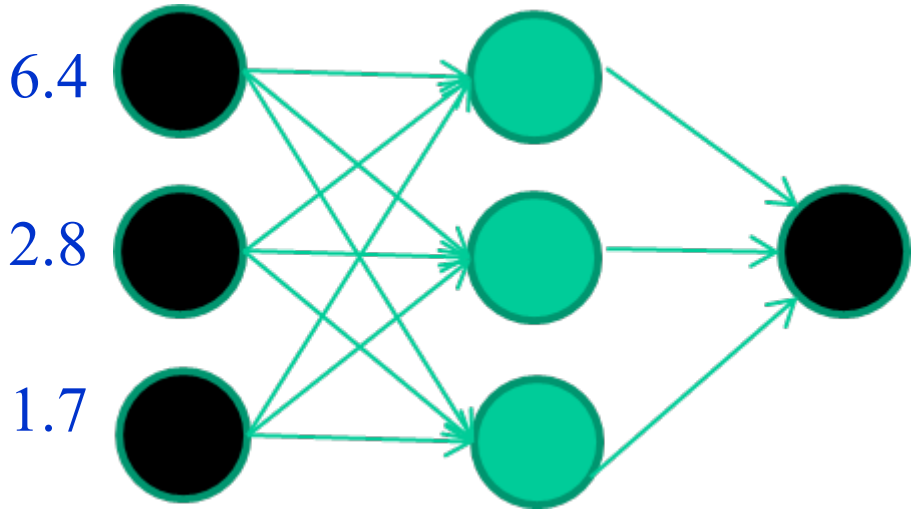


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Present a training pattern

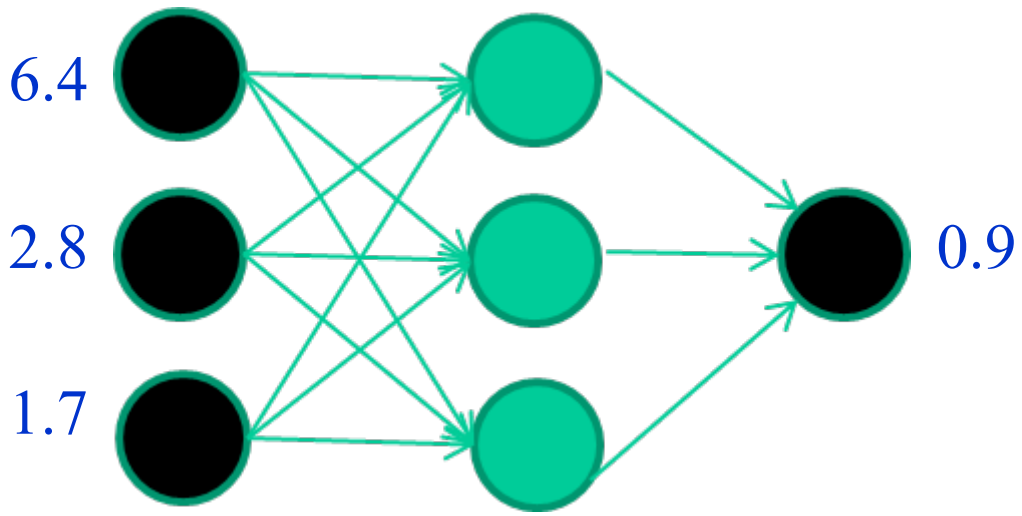


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Feed it through to get output

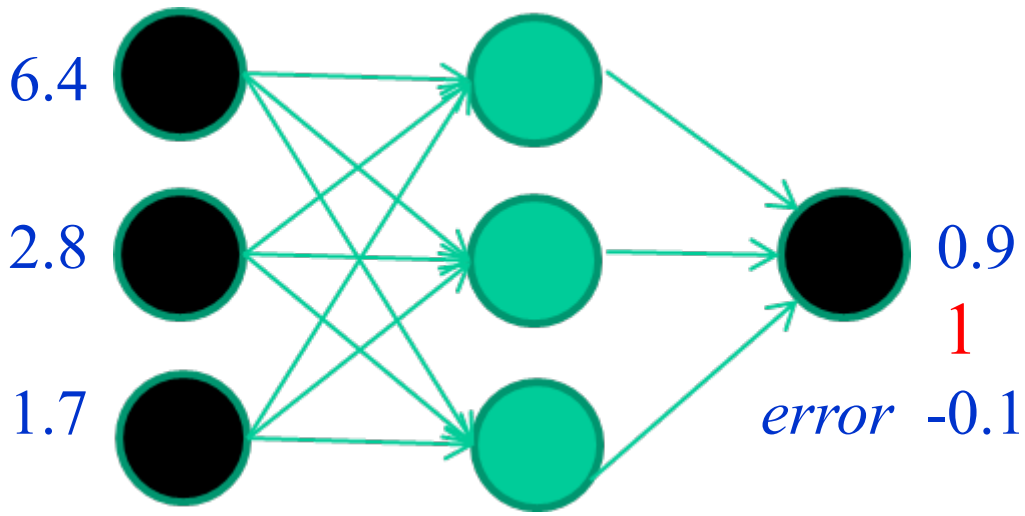


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Compare with target output

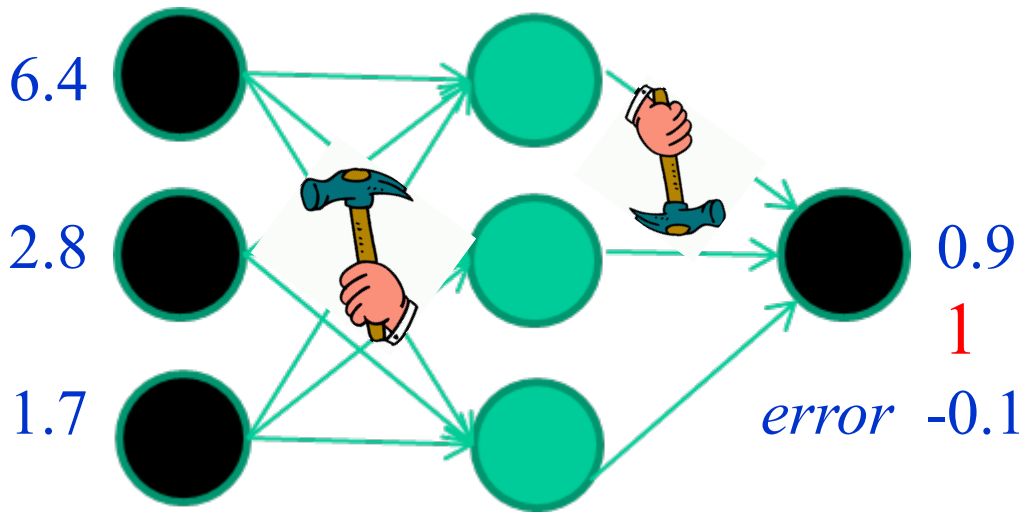


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

Adjust weights based on error

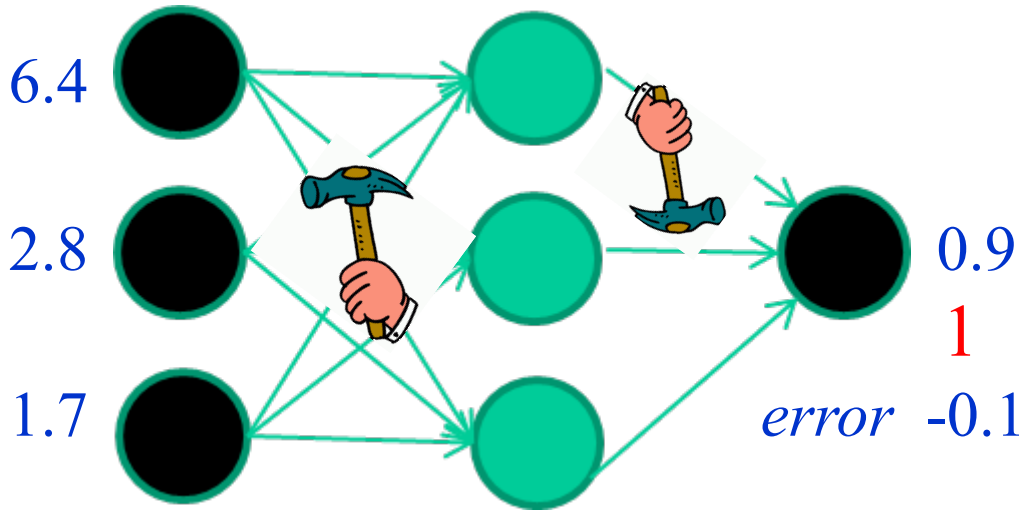


Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

Training data

Fields			***class***
1.4	2.7	1.9	0
3.8	3.4	3.2	0
6.4	2.8	1.7	1
4.1	0.1	0.2	0
etc …			

And so on ….



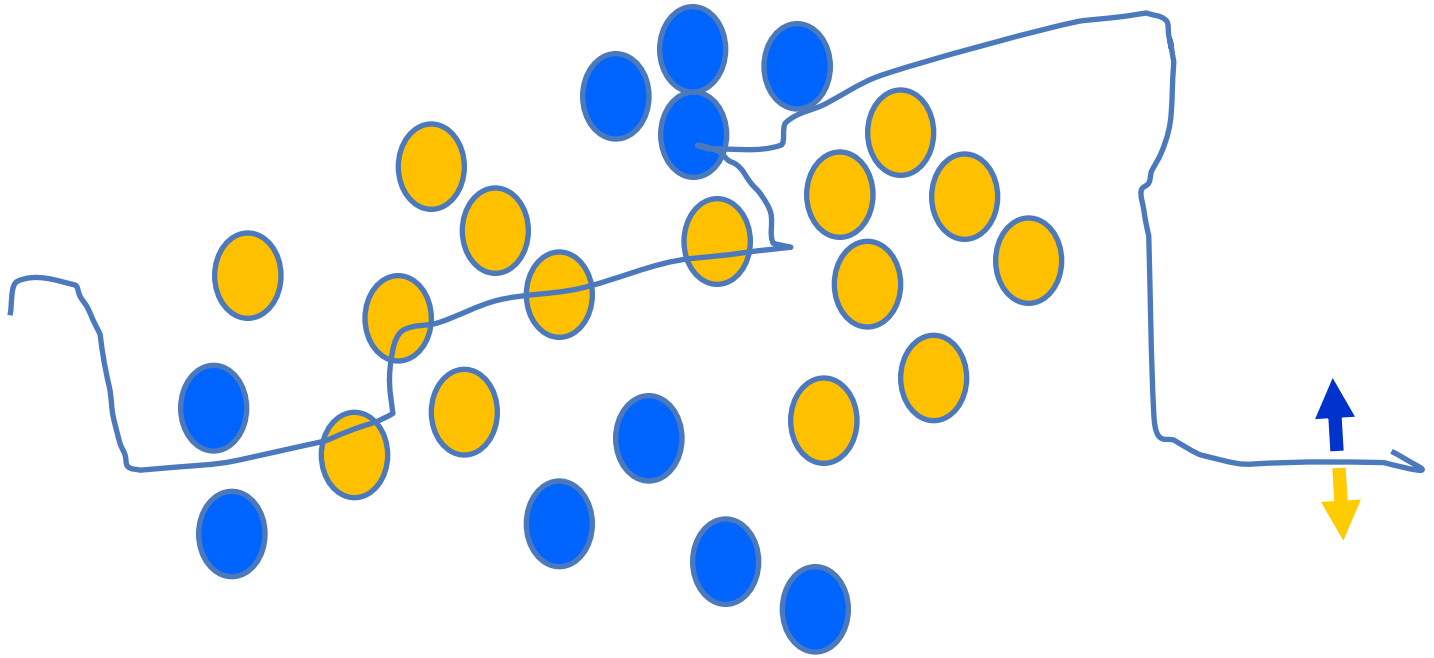
Repeat this thousands, maybe millions of times – each time taking a random training instance, and making slight weight adjustments

Algorithms for weight adjustment are designed to make changes that will reduce the error

Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

The decision boundary perspective…

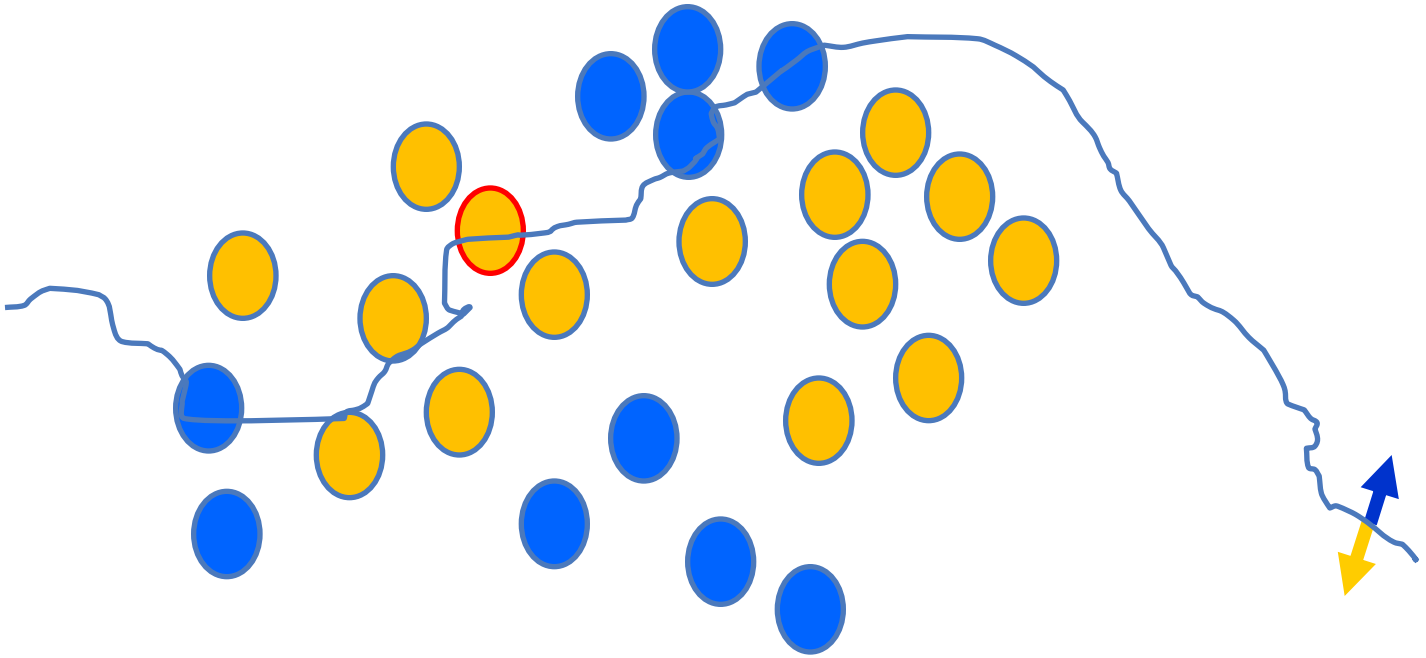
Initial random weights



Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

The decision boundary perspective…

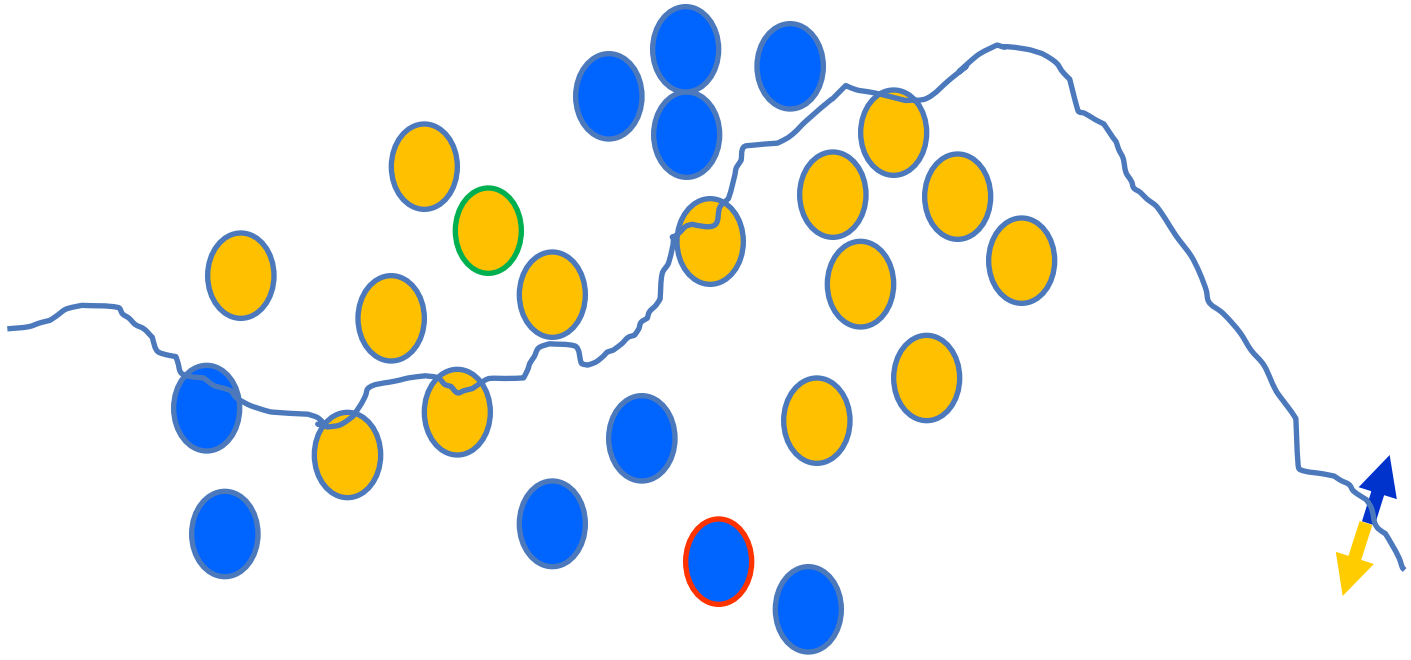
Present a training instance / adjust the weights



Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

The decision boundary perspective…

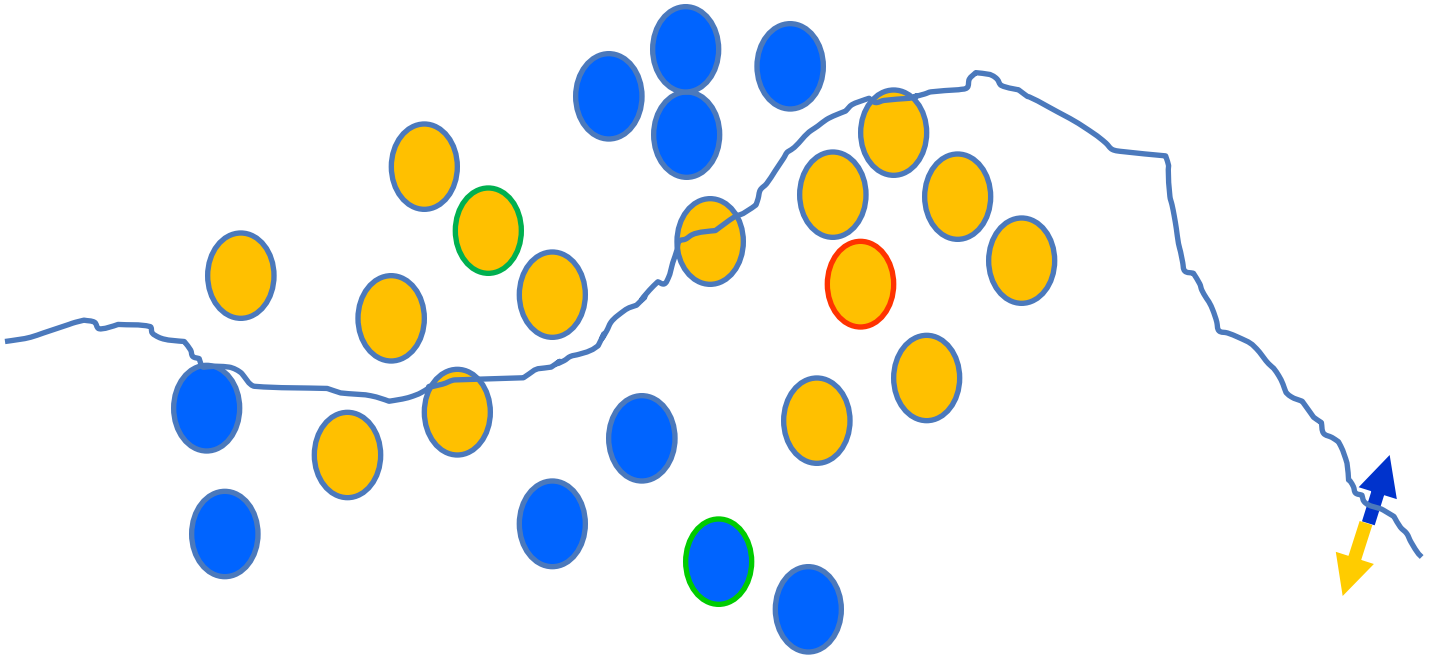
Present a training instance / adjust the weights



Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

The decision boundary perspective...

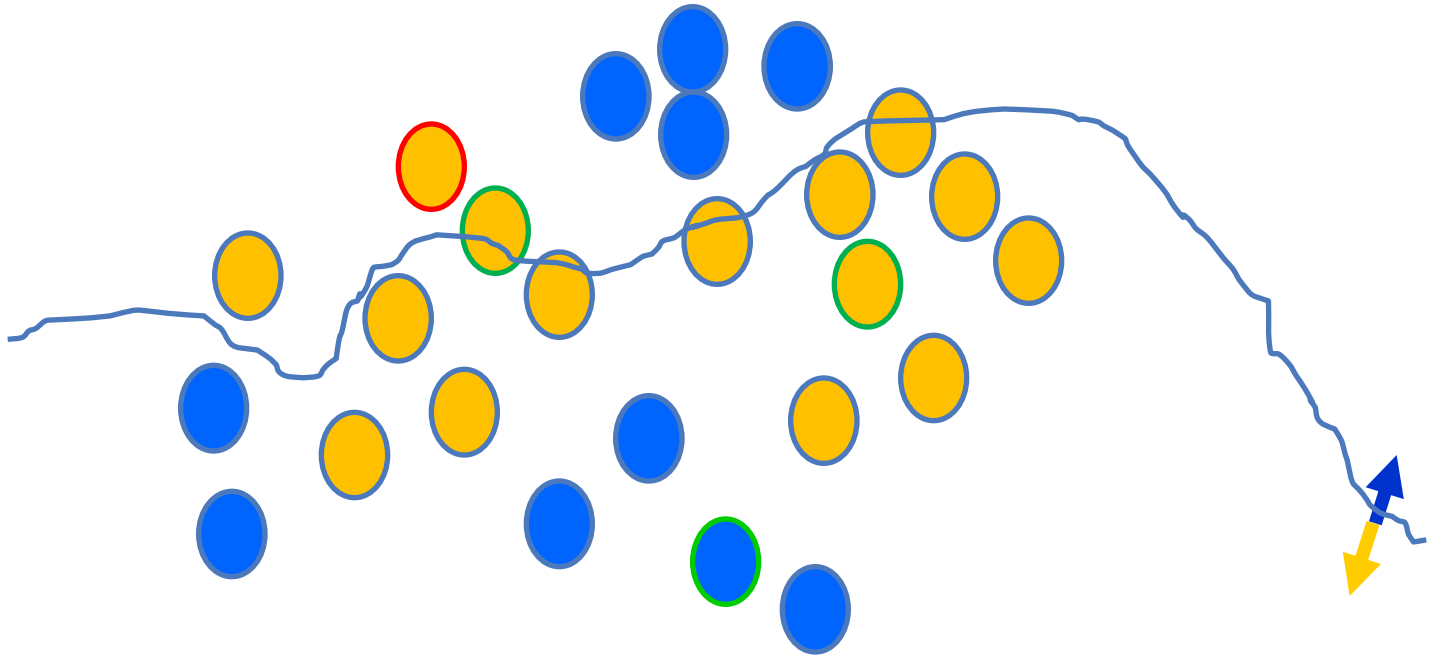
Present a training instance / adjust the weights



Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

The decision boundary perspective...

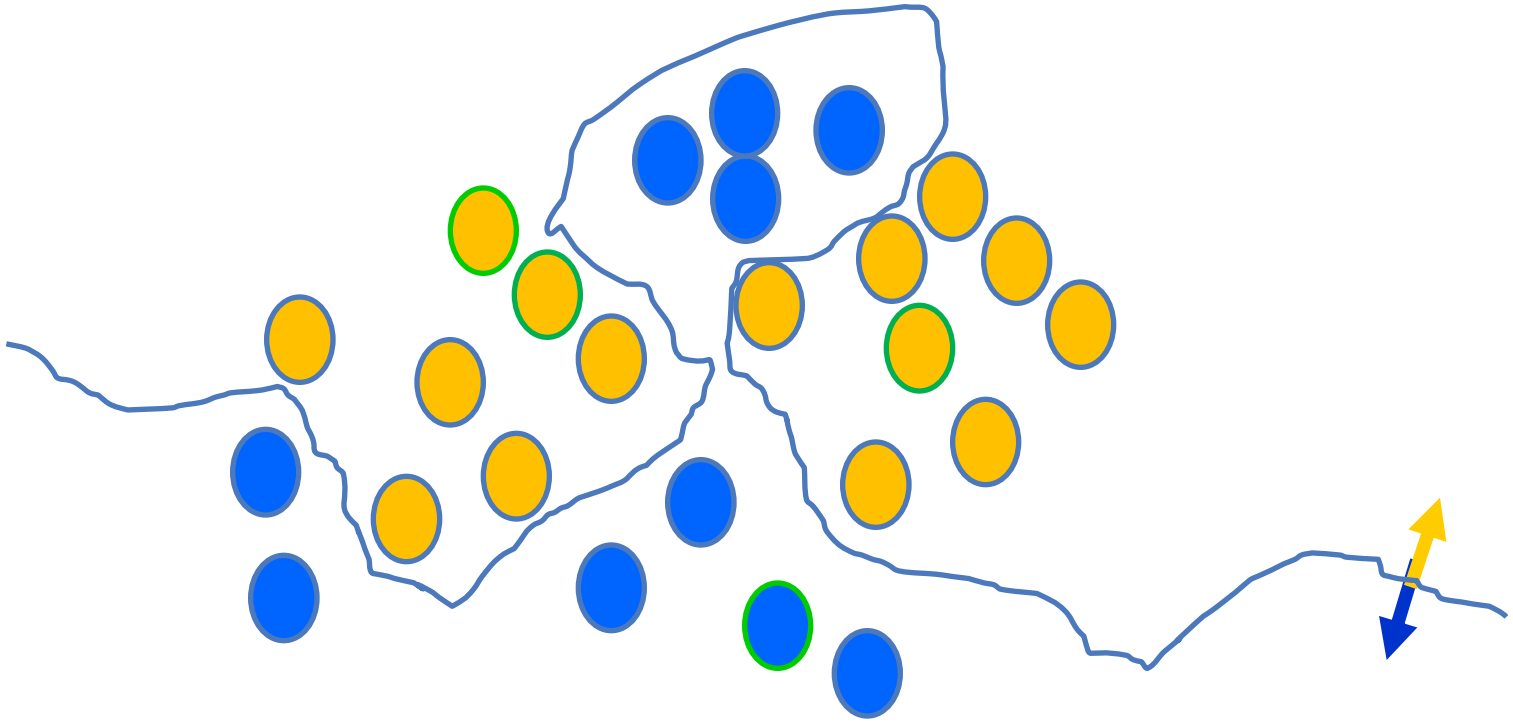
Present a training instance / adjust the weights



Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

The decision boundary perspective…

Eventually ….



Source: https://www.macs.hw.ac.uk/~dwcorne/Teaching/introdl.ppt

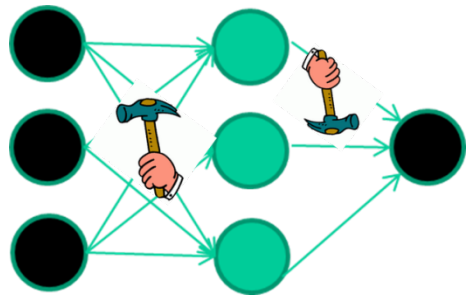
The key point here is…

weight-learning algorithms for Neural Nets are dumb

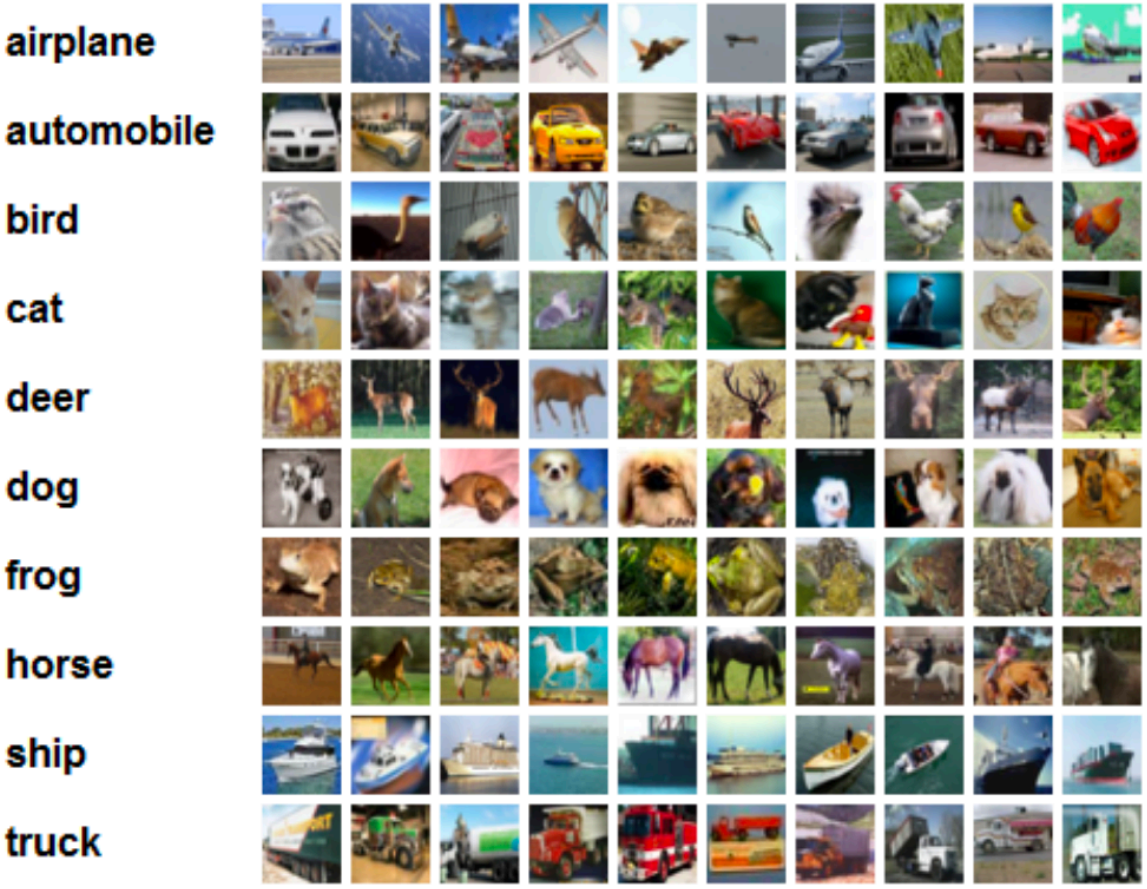
they work by making thousands and thousands of tiny adjustments, each making the network do better at the most recent pattern, but perhaps a little worse on many others

However, by sheer luck, eventually this tends to be good enough to

learn effective classifiers for many real applications



CIFAR 10 and Convolutional Neural Network



CIFAR 10 dataset:
50,000 training images
10,000 testing images
10 categories (classes)

Accuracies from different methods:

Human: ~94%

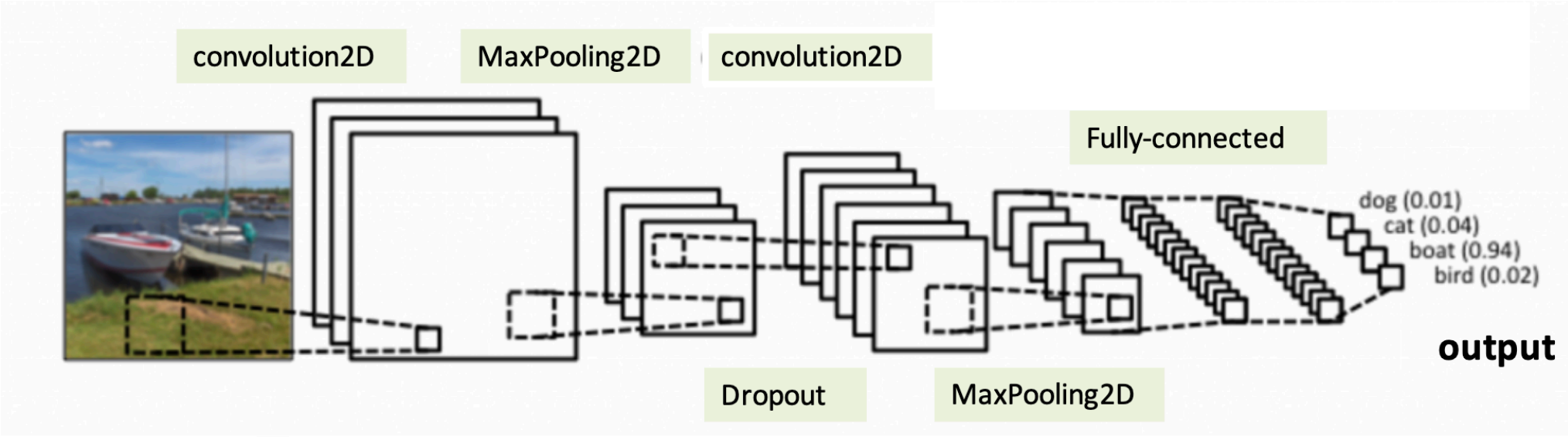
Whitening K-mean: 80%

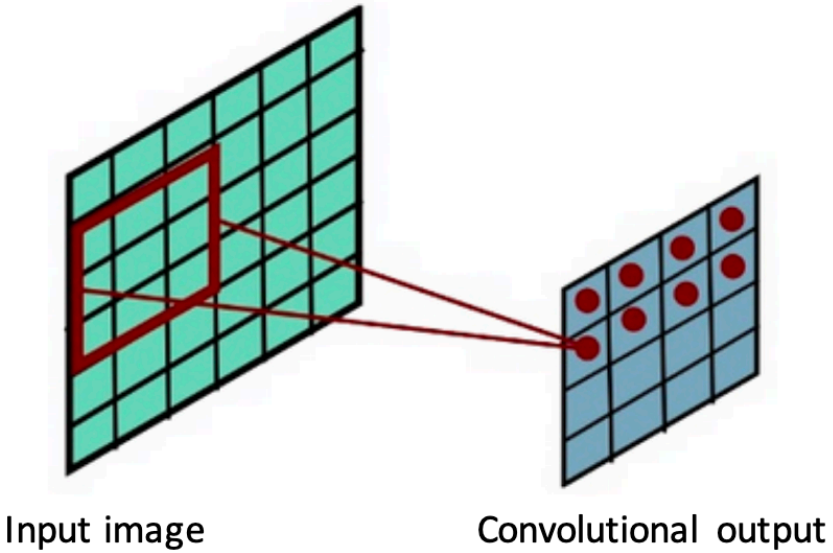
.....

Deep CNN: 95.5%

Source: http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/

Deep CNN on CIFAR10

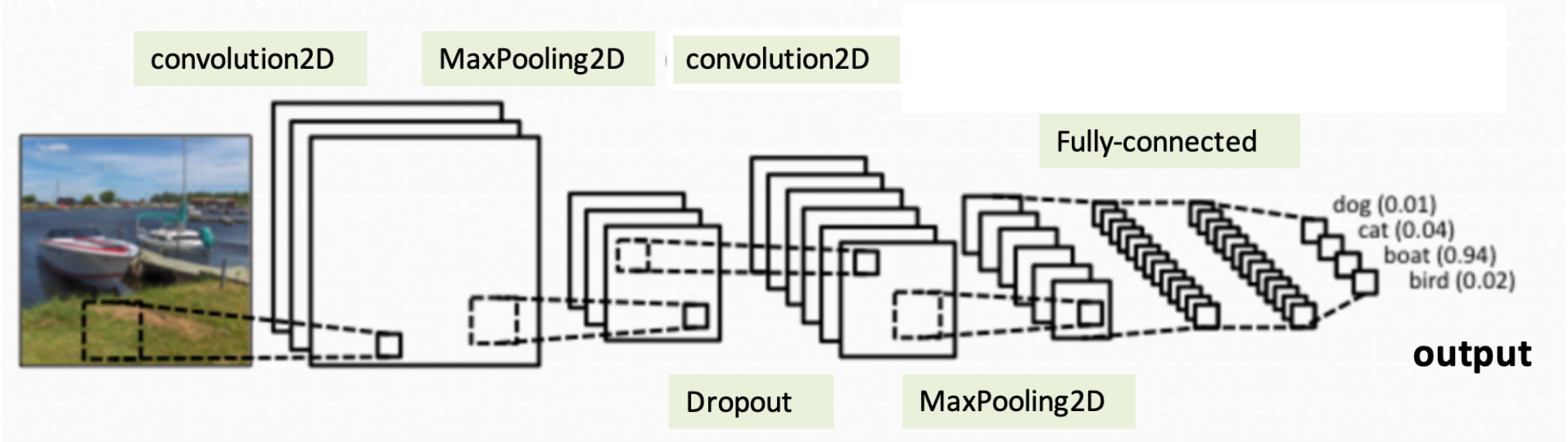


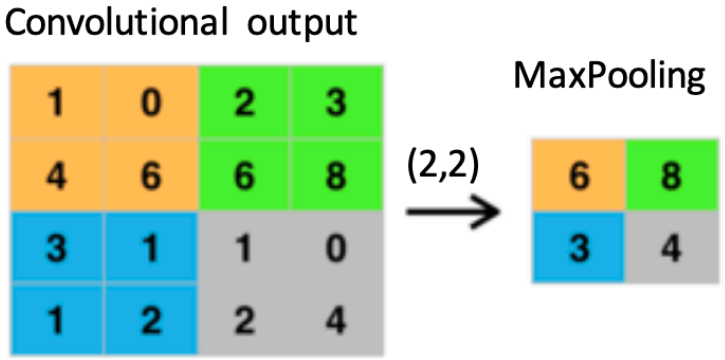


Convolutional Layer: filters work on every part of the image, therefore, they are searching for the same feature everywhere in the image.

Source: http://www.slideshare.net/perone/deep--learning--convolutional--neural--networks

Deep CNN on CIFAR10

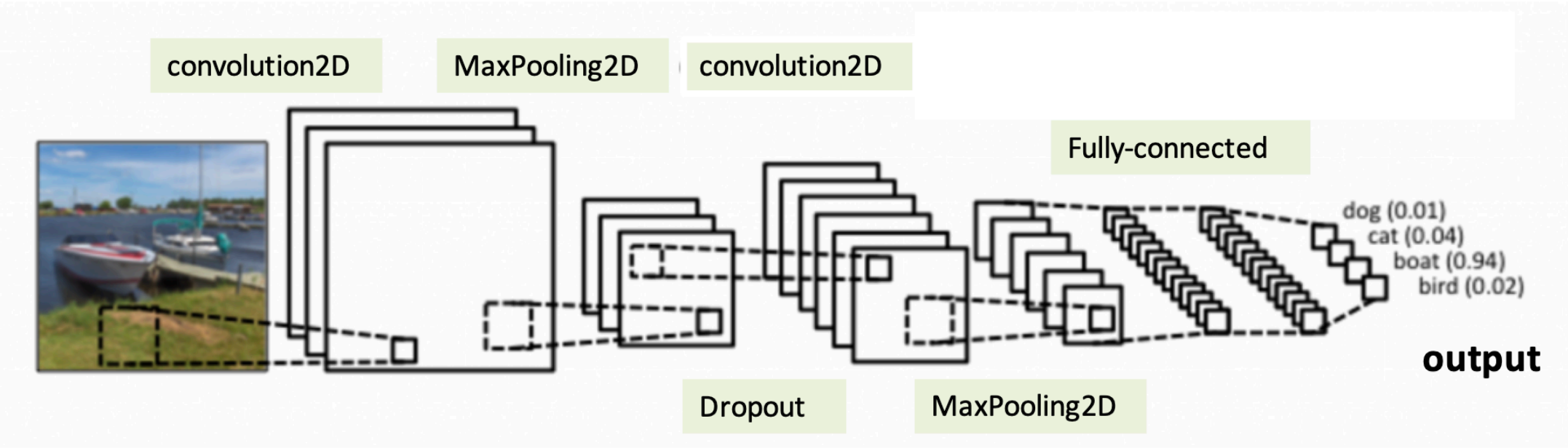


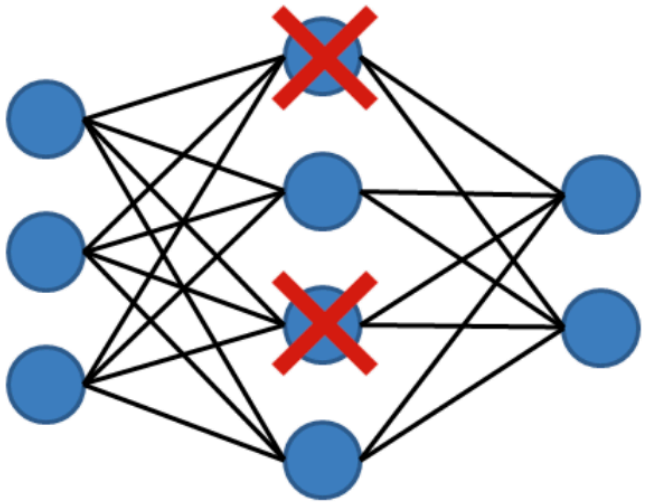


MaxPooling: usually present after the convolutional layer. It provides a down-sampling of the convolutional output

Source: http://www.slideshare.net/perone/deep--learning--convolutional--neural--networks

Deep CNN on CIFAR10

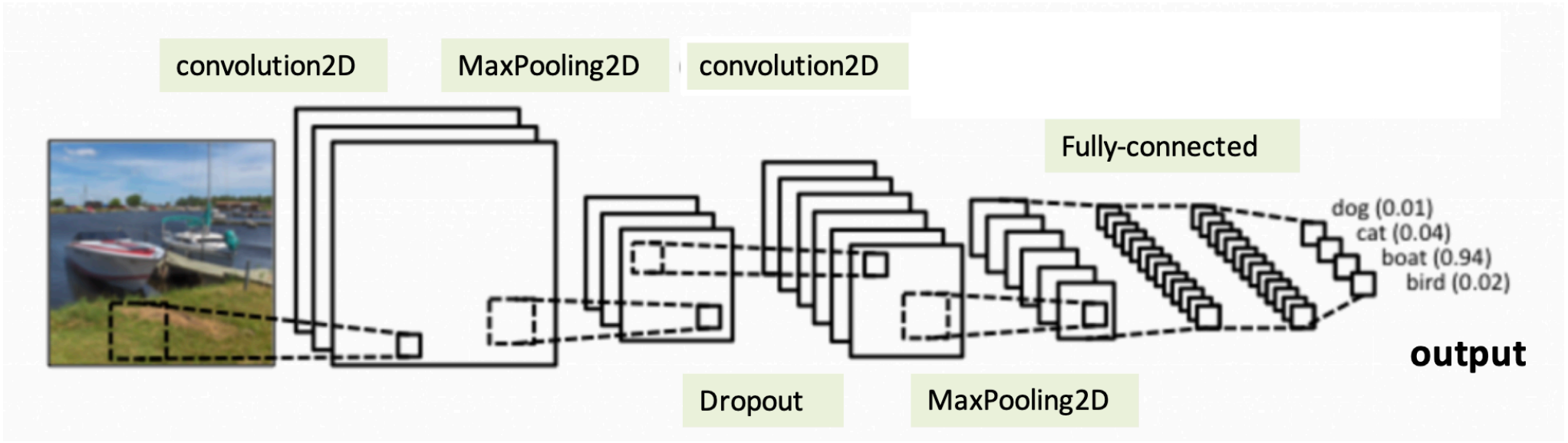


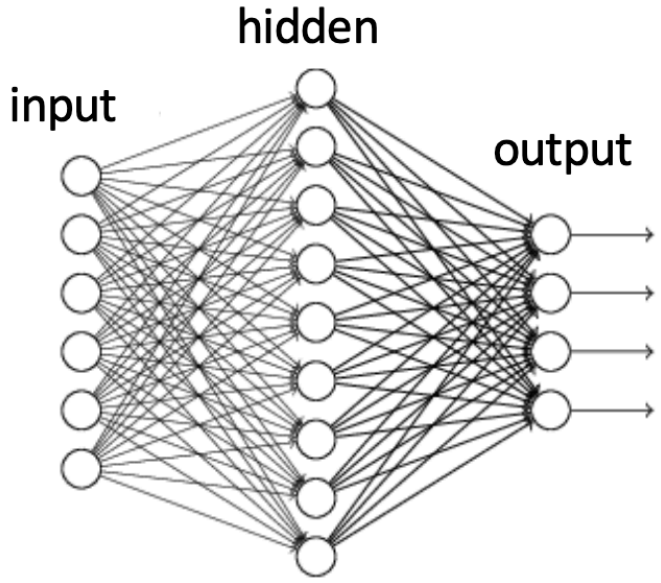


Dropout: randomly drop units along with their connections during training. It helps to learn more robust features by reducing complex co-adaptations of units and alleviate overfitting issue as well.

Source: Srivastavaet al. Dropout: A Simple way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15(2014):1929--1958

Deep CNN on CIFAR10





Fully-connected layer (dense): each node is fully connected to all input nodes, each node computes weighted sum of all input nodes. It has one-dimensional structure. It helps to classify input pattern with high-level features extracted by previous layers.

Source: http://www.slideshare.net/datasciencekorea/5--20141107deeplearning

Deep Learning – R Libraries

R Package	Description
Nnet	Software for feed-forward neural networks with a single hidden layer and for multinomial log linear models
Neuralnet	Training of neural networks using backpropagation
H2O	The R interface to H2O deep learning framework
MXNet	The R interface to the MXNet deep learning library
darch	A R package for deep architectures and Restricted Boltzmann Machines
Tensorflow	Interface to tensorflow
Deepnet	Deep learning toolkit in R

Installation of Keras with TensorFlow at backend

```
install.packages("devtools")
```

```
devtools::install_github("rstudio/keras")
```

The above step will load the keras library from the GitHub repository. Now it is time to load keras into R and install tensorflow.

```
library(keras)
```

By default RStudio loads the CPU version of tensorflow. Use the below command to download the CPU version of tensorflow.

```
install_tensorflow()
```

To install the tensorflow version with GPU support for a single user/desktop system, use the below command.

```
install_tensorflow(gpu=TRUE)
```

For multi user installation guide pls refer here - https://tensorflow.rstudio.com/tools/local_gpu.html

R Interface to Keras

R interface to Keras

R interface to Keras

Keras is a high-level neural networks API developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.* Keras has the following key features:

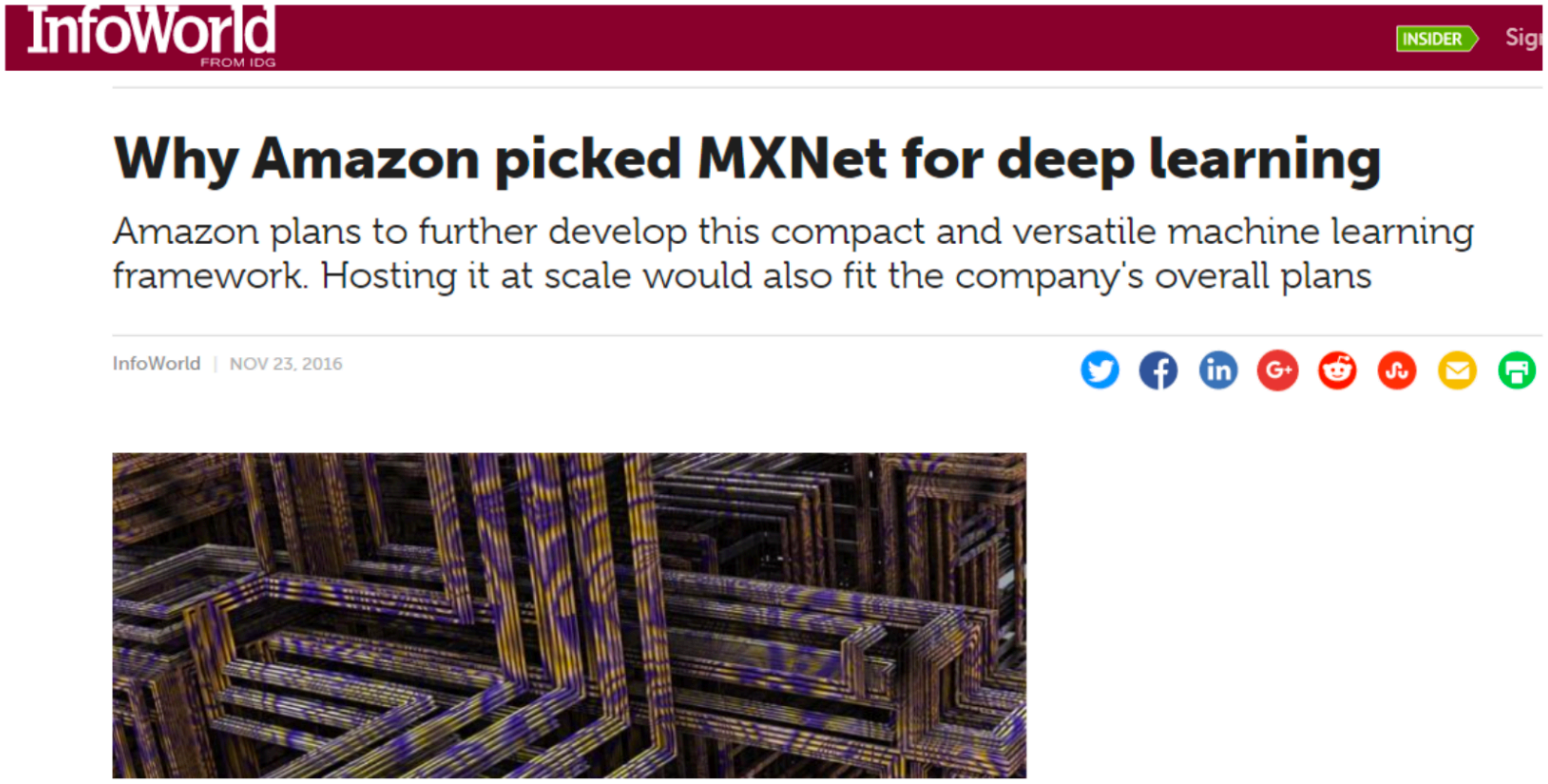
- Allows the same code to run on CPU or on GPU, seamlessly.
- User-friendly API which makes it easy to quickly prototype deep learning models.
- Built-in support for convolutional networks (for computer vision), recurrent networks (for sequence processing), and any combination of both.
- Supports arbitrary network architectures: multi-input or multi-output models, layer sharing, model sharing, etc. This means that Keras is appropriate for building essentially any deep learning model, from a memory network to a neural Turing machine.
- Is capable of running on top of multiple back-ends including TensorFlow, CNTK, or Theano.

For additional details on why you might consider using Keras for your deep learning projects, see the Why Use Keras? article.

This website provides documentation for the R interface to Keras. See the main Keras website at https://keras.io for additional information on the project.

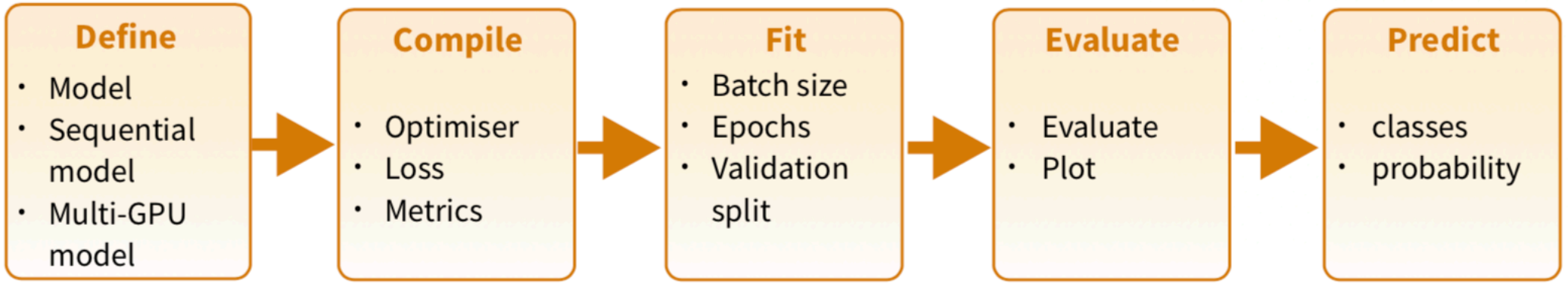
MXNet

```
install.packages("drat", repos="https://cran.rstudio.com")
drat:::addRepo("dmlc")
install.packages("mxnet")
```

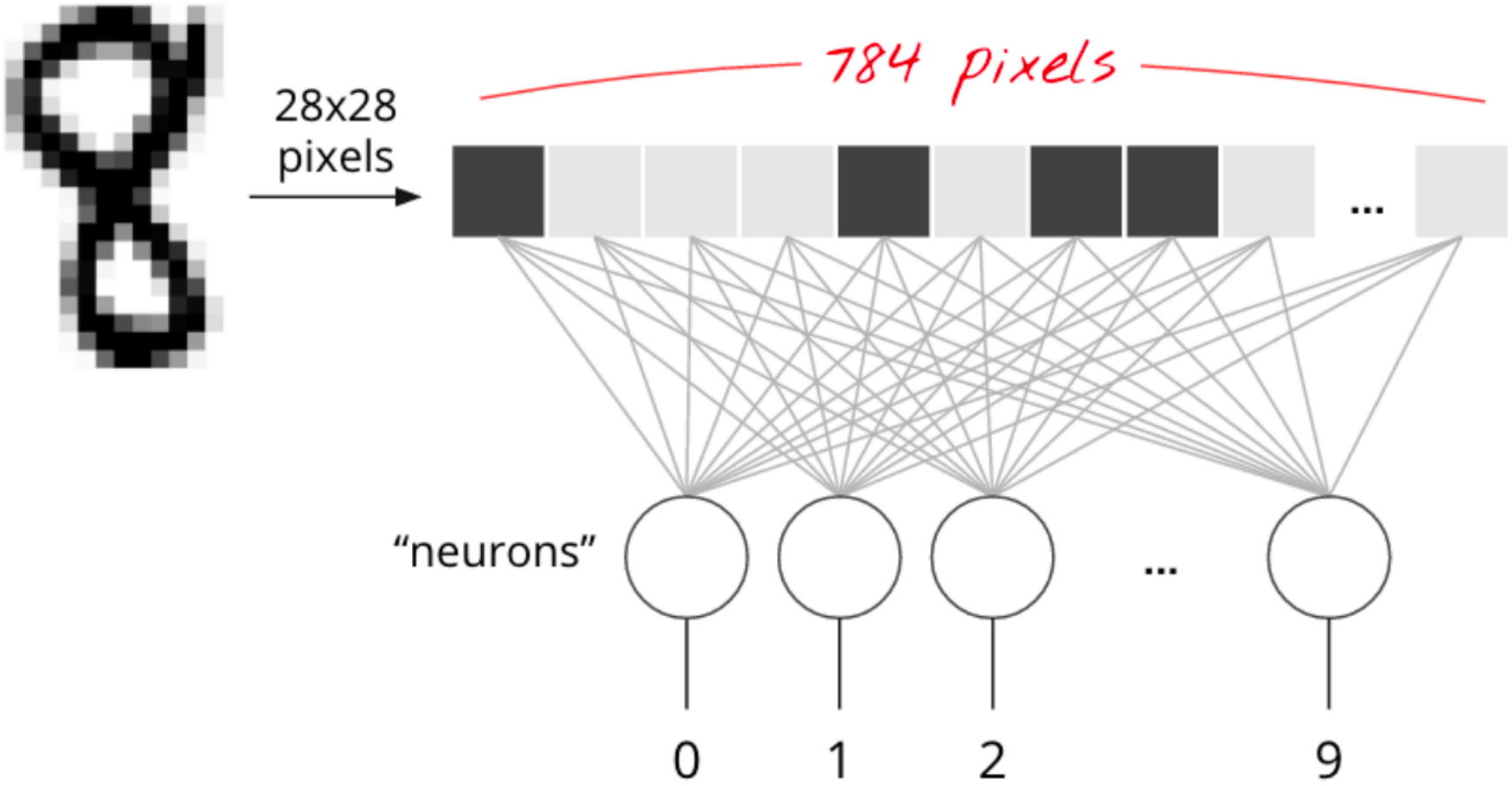


For multi user installation guide pls refer here - https://tensorflow.rstudio.com/tools/local_gpu.html

With the help of Keras



Classify with MNIST data



Training the image recognizer

```
# input layer: use MNIST images
mnist <- dataset_mnist()
x_train <- mnist$train$x;  y_train <- mnist$train$y
x_test <- mnist$test$x;  y_test <- mnist$test$y

# reshape and rescale
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))
x_train <- x_train / 255;  x_test <- x_test / 255

y_train <- to_categorical(y_train, 10)
y_test <- to_categorical(y_test, 10)

# defining the model and layers
model <- keras_model_sequential()
model %>%
 layer_dense(units = 256, activation = 'relu',
             input_shape = c(784)) %>%
 layer_dropout(rate = 0.4) %>%
 layer_dense(units = 128, activation = 'relu') %>%
 layer_dense(units = 10, activation = 'softmax')
```

```
# compile (define loss and optimizer)
model %>% compile(
 loss = 'categorical_crossentropy',
 optimizer = optimizer_rmsprop(),
 metrics = c('accuracy')
)

# train (fit)
model %>% fit(
 x_train, y_train,
 epochs = 30, batch_size = 128,
 validation_split = 0.2
)
model %>% evaluate(x_test, y_test)
model %>% predict_classes(x_test)
```

References

- Deep Learning in R using H2O: http://didericksen.github.io/deeplearning-r-h2o/
- Keras Cheat sheets: https://www.rstudio.com/resources/cheatsheets/#keras
- http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/
- Video reference: https://livevideo.manning.com/module/52_1_2/deep-learning-with-r-in-motion/getting-started/what-is-deep-learning%3f?

- https://blog.rstudio.com/2018/09/12/getting-started-with-deep-learning-in-r/
- MNIST dataset related –
  - Distorting the MNIST image dataset - https://msdn.microsoft.com/en-us/magazine/dn754573.aspx
  - Optical Recognition of handwritten digits dataset - http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
  - Pen-Based Recognition of Handwritten digits dataset - http://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits
  - Semeion Handwritten digit dataset - http://archive.ics.uci.edu/ml/datasets/Semeion+Handwritten+Digit
  - The EMNIST dataset's digit part - https://www.nist.gov/node/1298471/emnist-dataset

Thank you

kkm_007

https://www.linkedin.com/in/kamalmishra07/

https://github.com/kkm24132