

Social Computing and Big Data Analytics

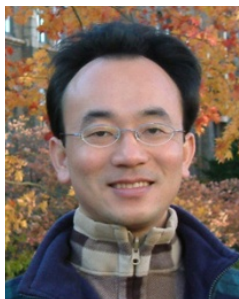
社群運算與大數據分析

Deep Learning with Theano and Keras in Python (Python Theano 和 Keras 深度學習)

1042SCBDA09

MIS MBA (M2226) (8628)

Wed, 8,9, (15:10-17:00) (B309)



Min-Yuh Day

戴敏育

Assistant Professor

專任助理教授

Dept. of Information Management, Tamkang University

淡江大學 資訊管理學系

<http://mail.tku.edu.tw/myday/>

2016-04-27



課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2016/02/17	Course Orientation for Social Computing and Big Data Analytics (社群運算與大數據分析課程介紹)
2	2016/02/24	Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data (資料科學與大數據分析： 探索、分析、視覺化與呈現資料)
3	2016/03/02	Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem (大數據基礎：MapReduce典範、 Hadoop與Spark生態系統)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
4	2016/03/09	Big Data Processing Platforms with SMACK: Spark, Mesos, Akka, Cassandra and Kafka (大數據處理平台SMACK : Spark, Mesos, Akka, Cassandra, Kafka)
5	2016/03/16	Big Data Analytics with Numpy in Python (Python Numpy 大數據分析)
6	2016/03/23	Finance Big Data Analytics with Pandas in Python (Python Pandas 財務大數據分析)
7	2016/03/30	Text Mining Techniques and Natural Language Processing (文字探勘分析技術與自然語言處理)
8	2016/04/06	Off-campus study (教學行政觀摩日)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
9	2016/04/13	Social Media Marketing Analytics (社群媒體行銷分析)
10	2016/04/20	期中報告 (Midterm Project Report)
11	2016/04/27	Deep Learning with Theano and Keras in Python (Python Theano 和 Keras 深度學習)
12	2016/05/04	Deep Learning with Google TensorFlow (Google TensorFlow 深度學習)
13	2016/05/11	Sentiment Analysis on Social Media with Deep Learning (深度學習社群媒體情感分析)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
14	2016/05/18	Social Network Analysis (社會網絡分析)
15	2016/05/25	Measurements of Social Network (社會網絡量測)
16	2016/06/01	Tools of Social Network Analysis (社會網絡分析工具)
17	2016/06/08	Final Project Presentation I (期末報告 I)
18	2016/06/15	Final Project Presentation II (期末報告 II)

Deep Learning

with

Theano

and

Keras

in

Python

LeCun, Yann,
Yoshua Bengio,
and Geoffrey Hinton.

"Deep learning."

Nature 521, no. 7553 (2015):
436-444.

Deep learning

Yann LeCun^{1,2}, Yoshua Bengio³ & Geoffrey Hinton^{4,5}

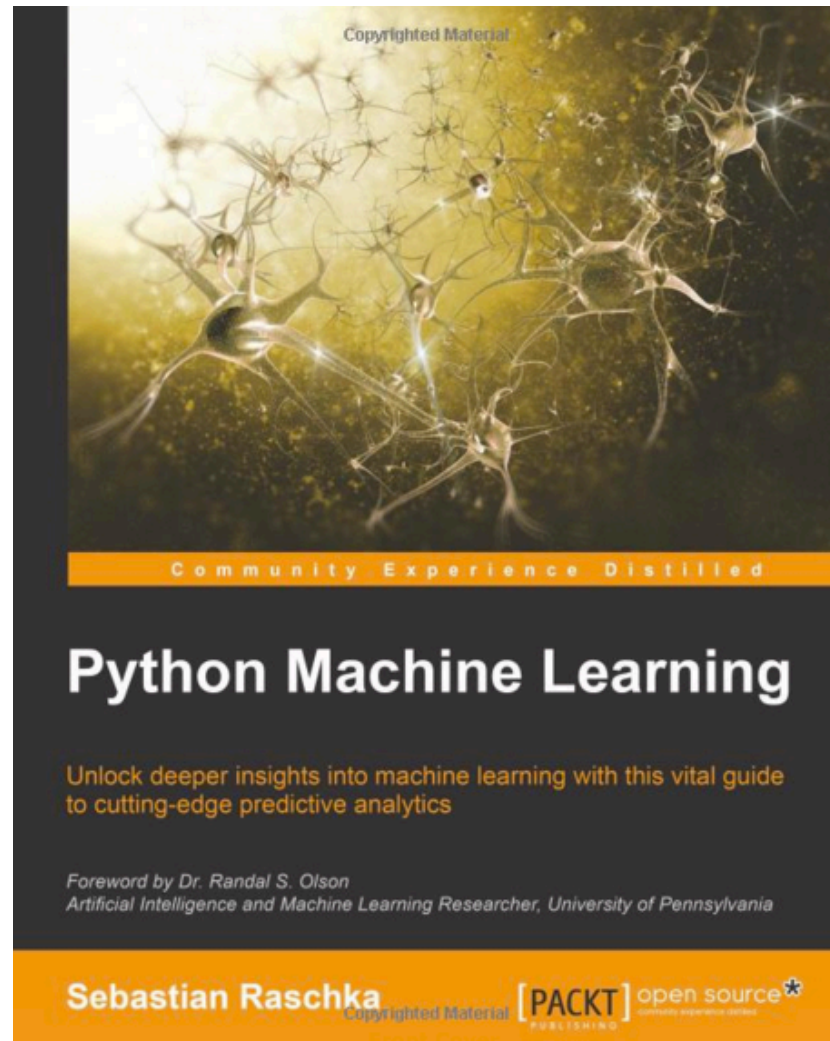
Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning.

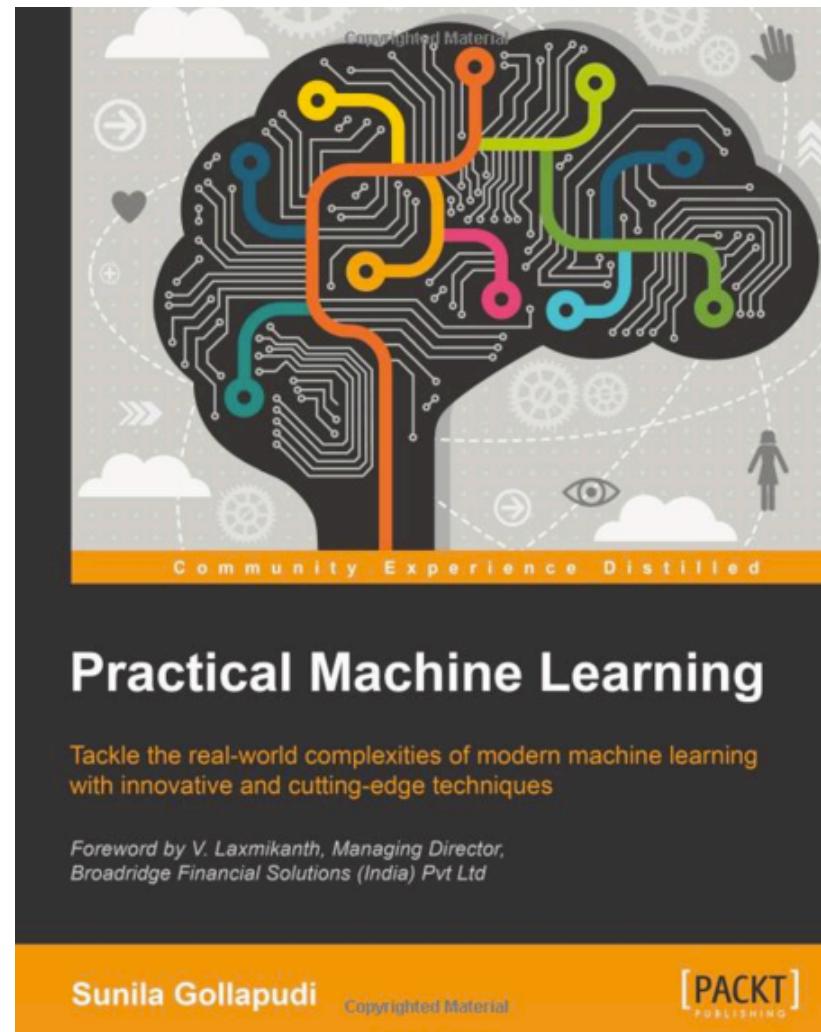
Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, con-

intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition^{1–4} and speech recognition^{5–7}, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules⁸, analysing particle accelerator data^{9,10}, reconstructing brain circuits¹¹, and predicting the effects of mutations in non-coding DNA on gene expression and disease^{12,13}. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding¹⁴, particularly topic classification, sentiment analysis, question answering¹⁵ and language translation^{16,17}.

Sebastian Raschka (2015),
Python Machine Learning,
Packt Publishing



Sunila Gollapudi (2016),
Practical Machine Learning,
Packt Publishing



Machine Learning Models

Deep Learning

Kernel

Association rules

Ensemble

Decision tree

Dimensionality reduction

Clustering

Regression Analysis

Bayesian

Instance based

Neural networks (NN) 1960

Multilayer Perceptrons (MLP) 1985

Restricted Boltzmann Machine (RBM) 1986

Support Vector Machine (SVM) 1995



Hinton presents the

Deep Belief Network (DBN)

**New interests in deep learning
and RBM**

State of the art MNIST

2005

Deep Recurrent Neural Network (RNN) 2009

Convolutional DBN 2010

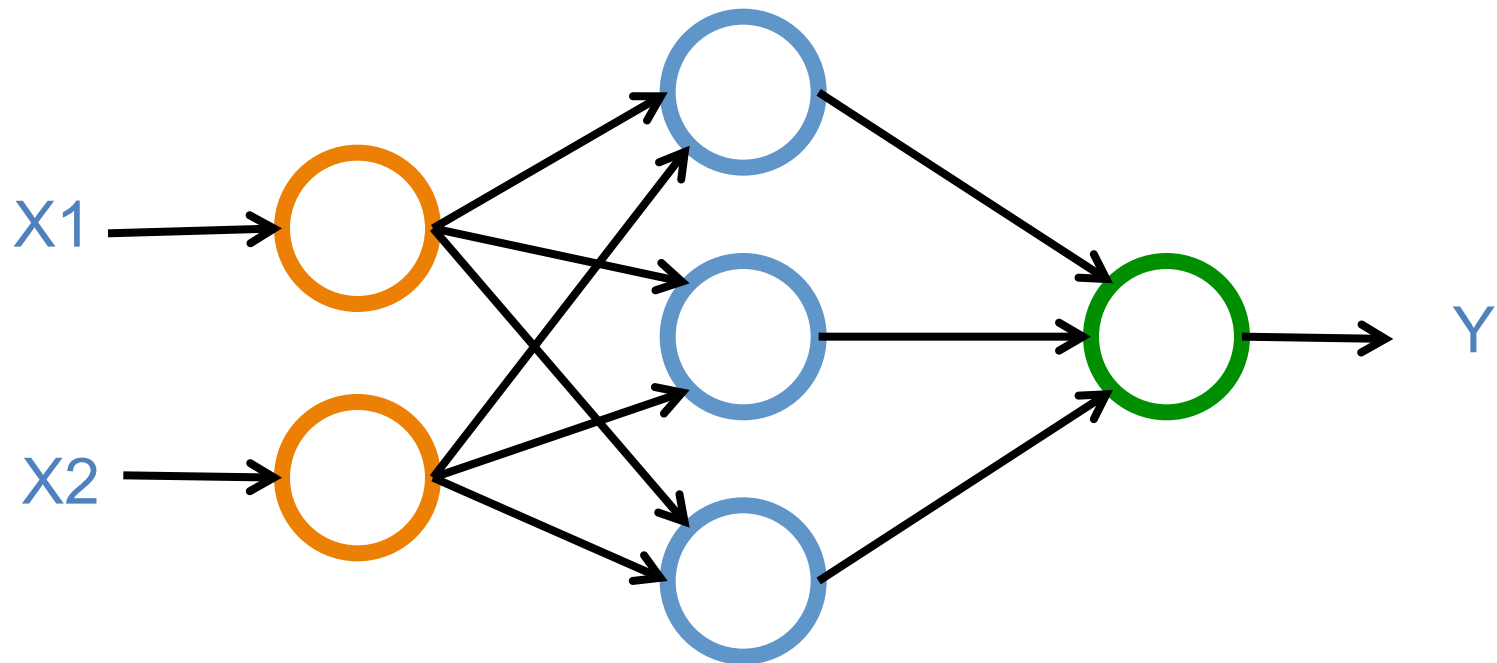
Max-Pooling CDBN 2011

Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



Deep Learning

Geoffrey Hinton

Yann LeCun

Yoshua Bengio

Andrew Y. Ng



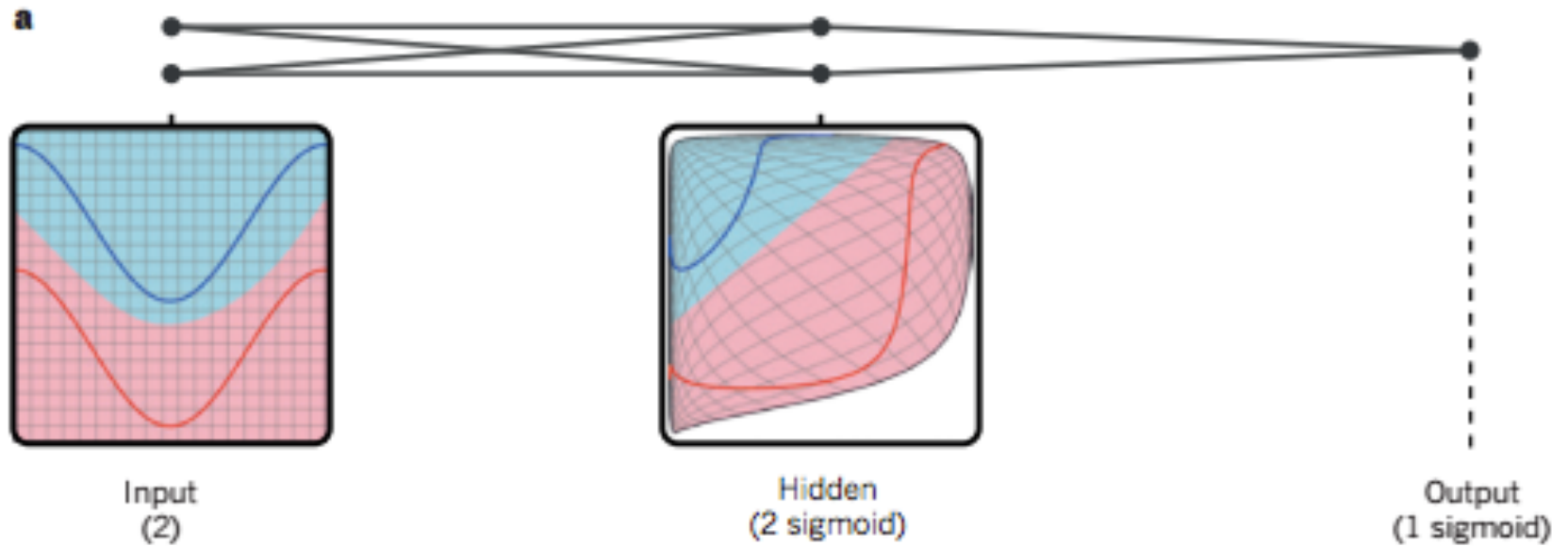
Geoffrey Hinton
Google
University of Toronto

LeCun, Yann,
Yoshua Bengio,
and Geoffrey Hinton.

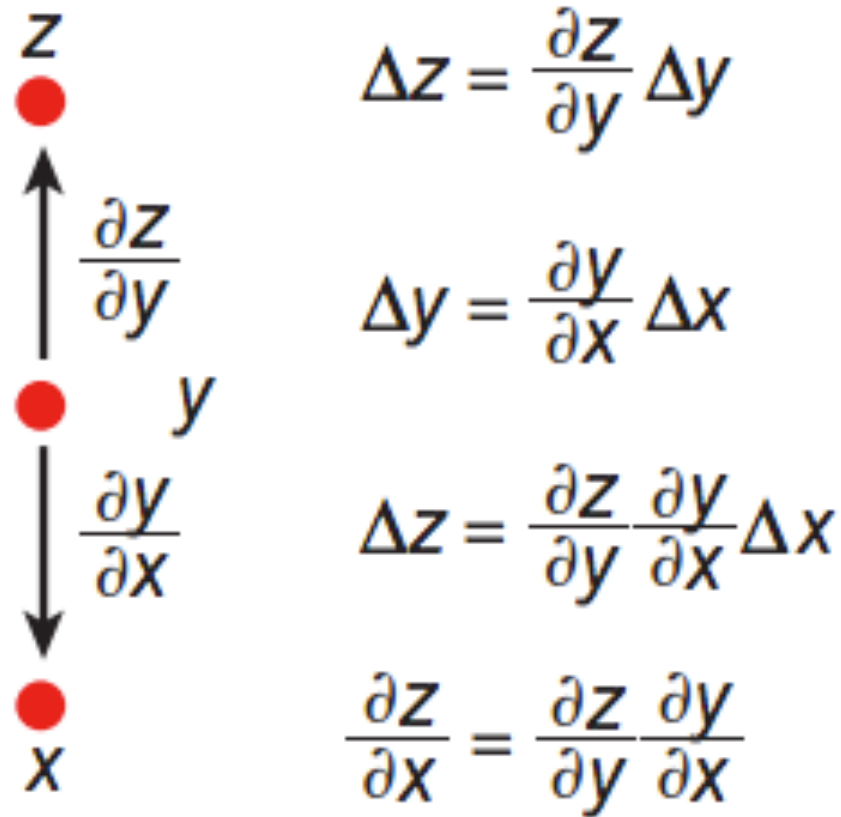
"Deep learning."

Nature 521, no. 7553 (2015):
436-444.

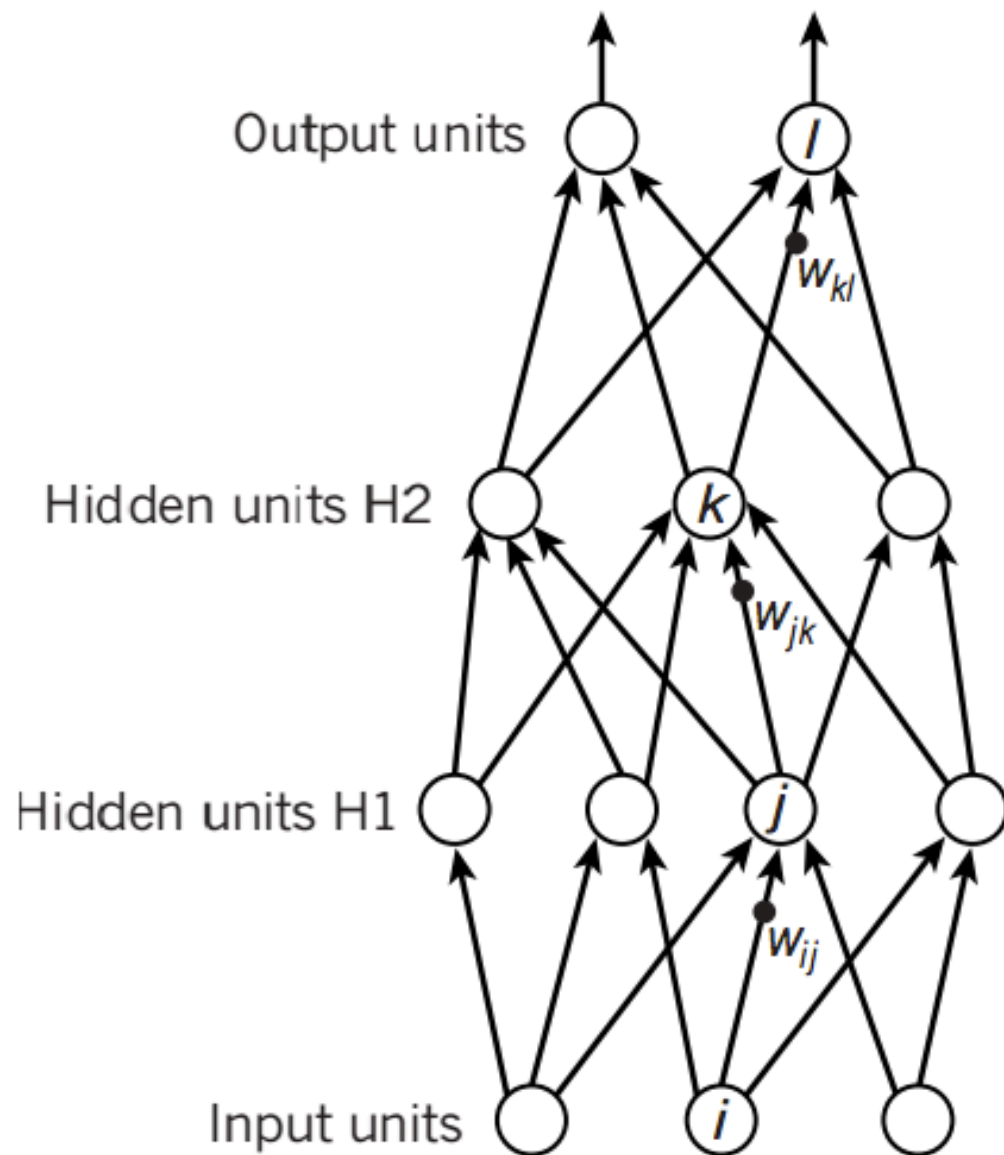
Deep Learning



Deep Learning



Deep Learning



$$y_l = f(z_l)$$

$$z_l = \sum_{k \in H2} w_{kl} y_k$$

$$y_k = f(z_k)$$

$$z_k = \sum_{j \in H1} w_{jk} y_j$$

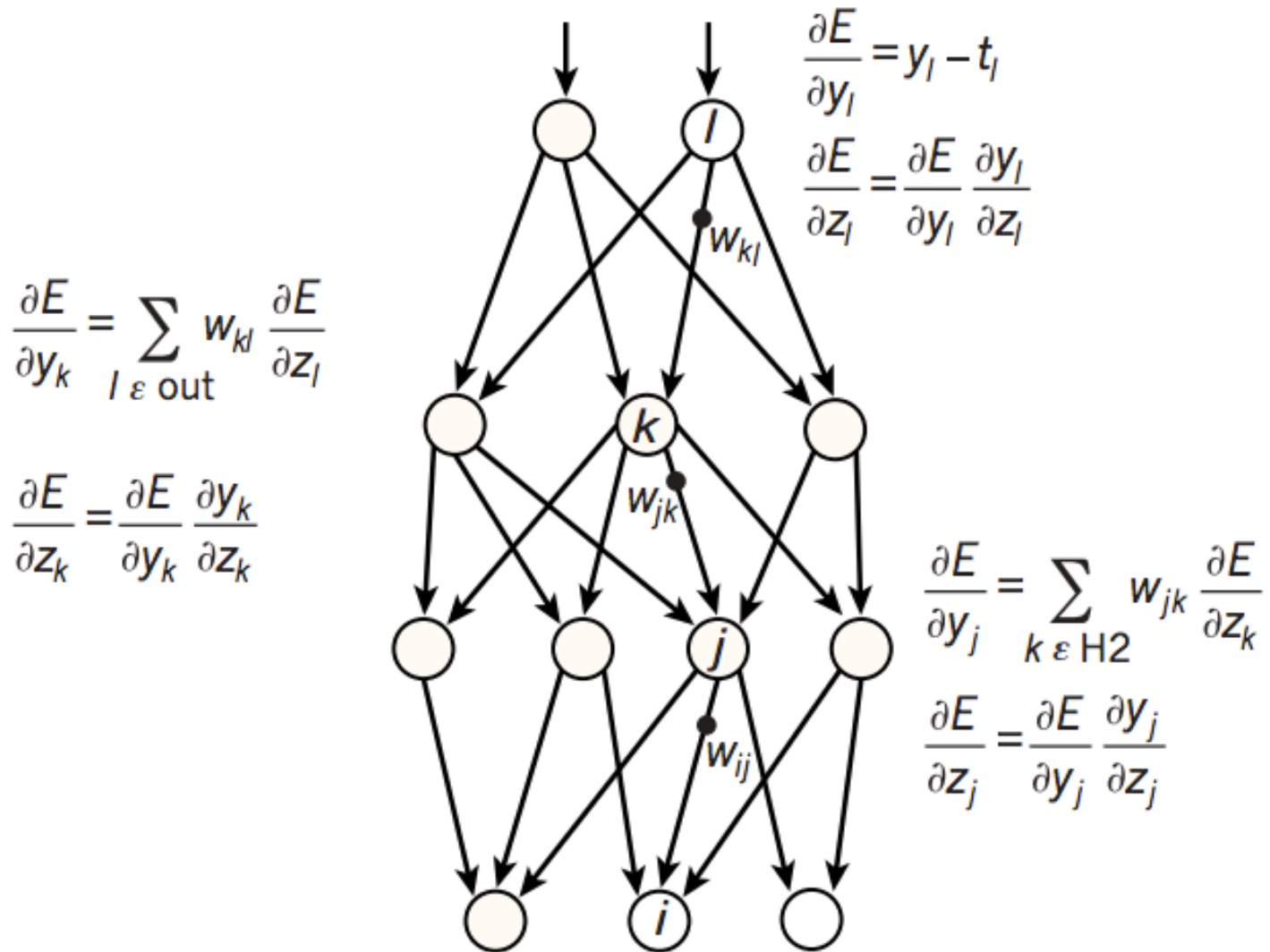
$$y_j = f(z_j)$$

$$z_j = \sum_{i \in \text{Input}} w_{ij} x_i$$

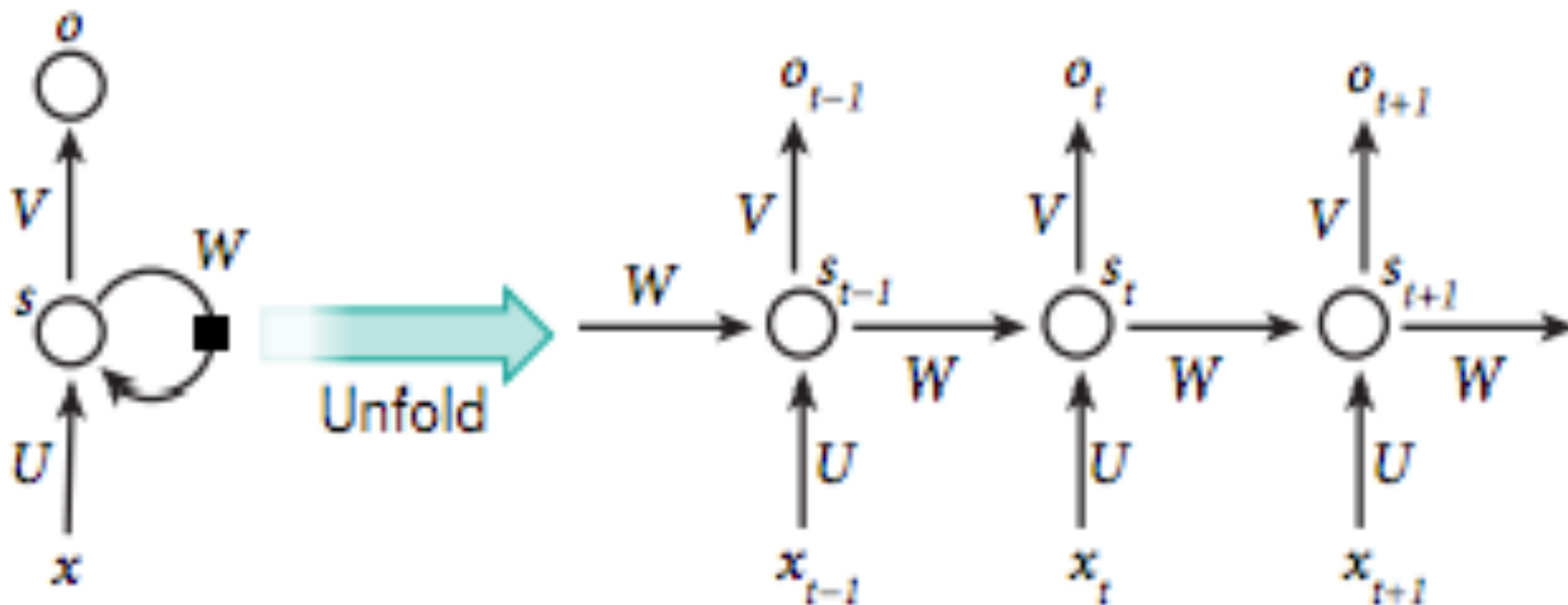
Deep Learning

d

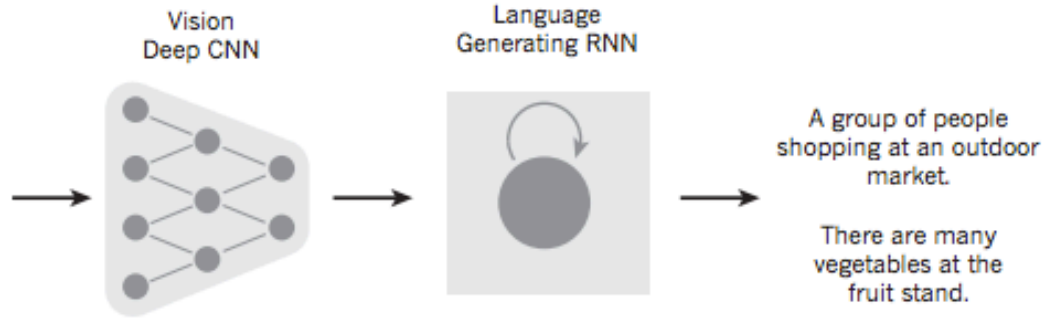
Compare outputs with correct answer to get error derivatives



Recurrent Neural Network (RNN)



From image to text



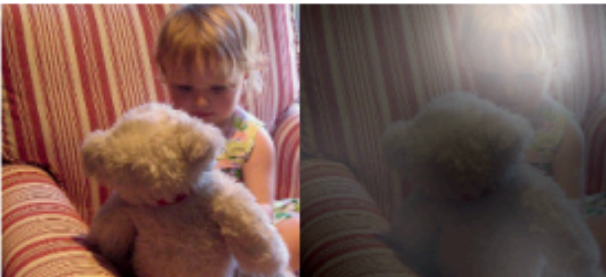
A woman is throwing a **frisbee** in a park.



A **dog** is standing on a hardwood floor.



A **stop** sign is on a road with a mountain in the background



A little **girl** sitting on a bed with a teddy bear.



A group of **people** sitting on a boat in the water.



A giraffe standing in a forest with **trees** in the background.

From image to text

Image: deep convolution neural network (CNN)

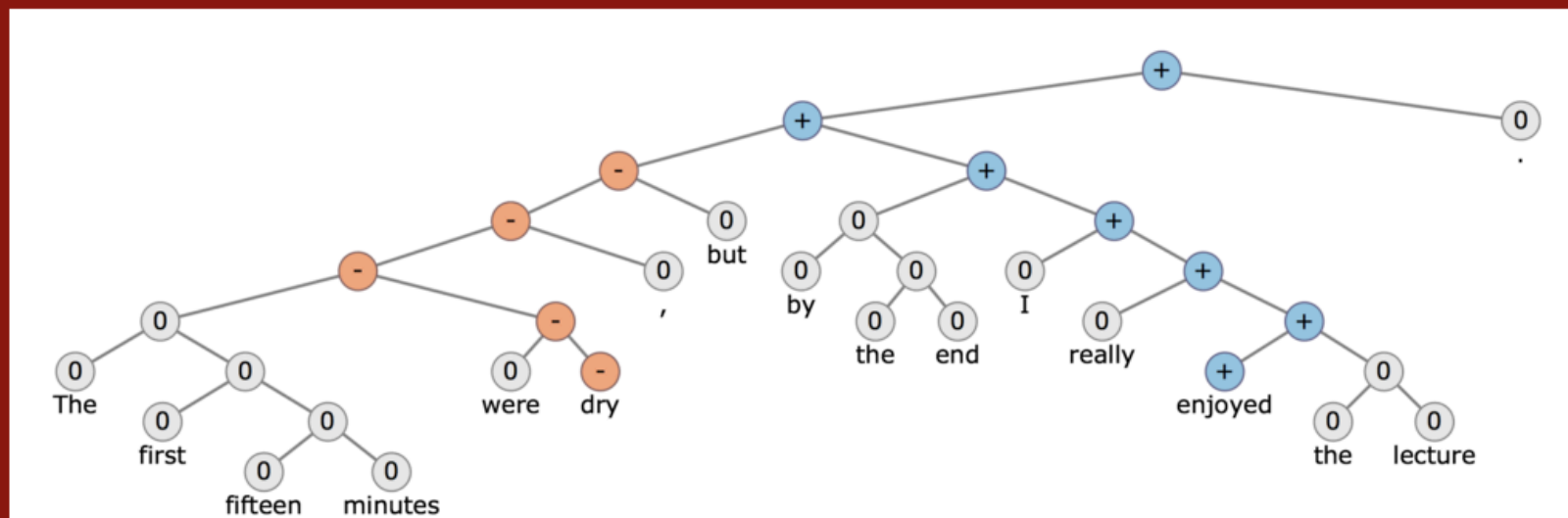
Text: recurrent neural network (RNN)



A group of **people** sitting on a boat in the water.

CS224d: Deep Learning for Natural Language Processing

CS224d: Deep Learning for Natural Language Processing

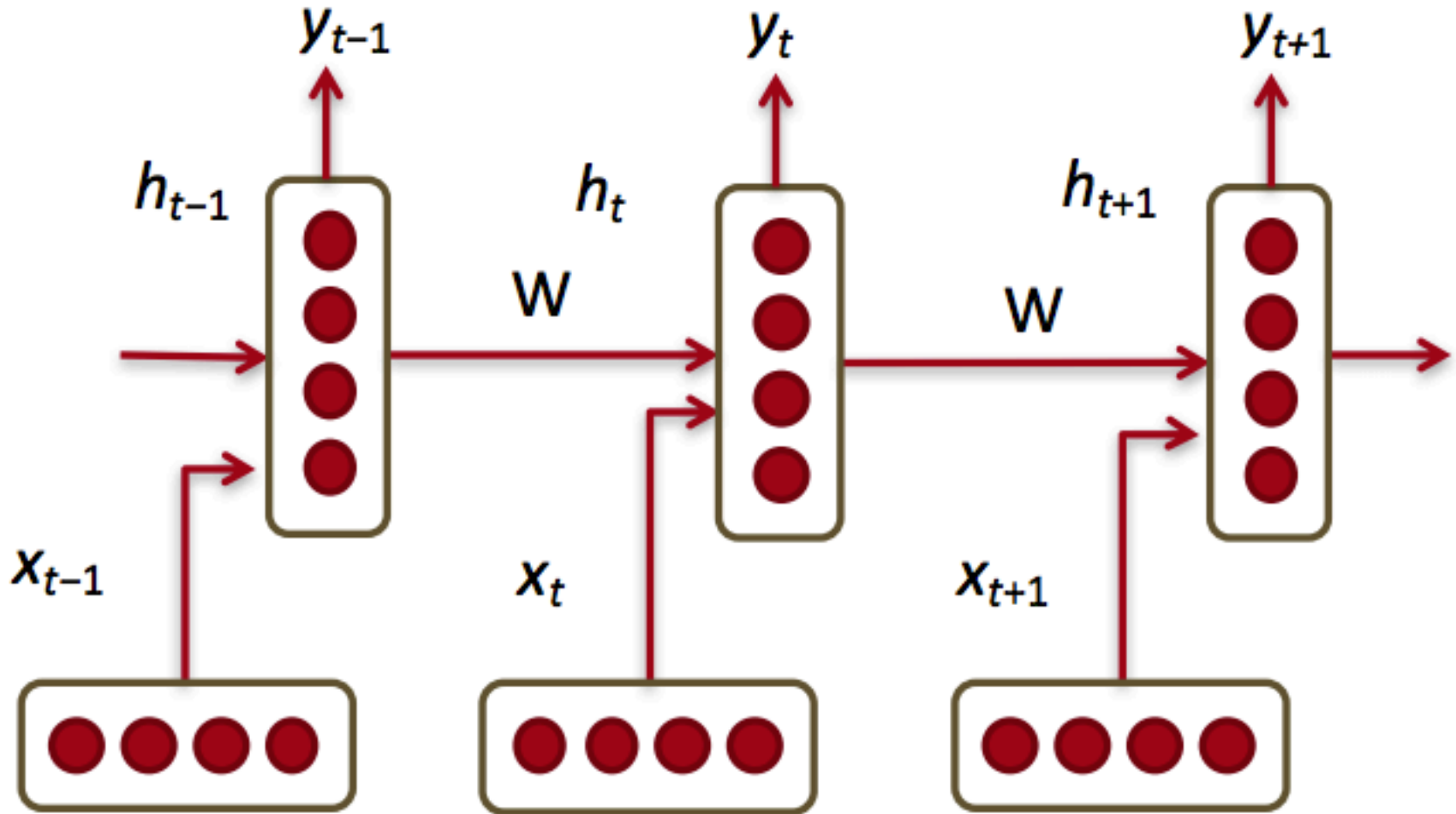


Course Description

Natural language processing (NLP) is one of the most important technologies of the information age. Understanding complex language utterances is also a crucial part of artificial intelligence. Applications of NLP are everywhere because people communicate most everything in language: web search, advertisement, emails, customer service, language translation, radiology reports, etc. There are a large variety of underlying tasks and machine learning models powering NLP applications. Recently, deep learning approaches have obtained very high performance across many different NLP tasks. These models can often be trained with a single end-to-end model and do not require traditional, task-specific feature engineering. In this spring quarter course students will learn to implement, train, debug, visualize and invent their own neural network models. The course provides a deep excursion into cutting-edge research in deep learning applied to NLP. The final project will involve training a complex recurrent neural network and applying it to a large scale NLP problem. On the model side we will cover word vector representations,

<http://cs224d.stanford.edu/>

Recurrent Neural Networks (RNNs)

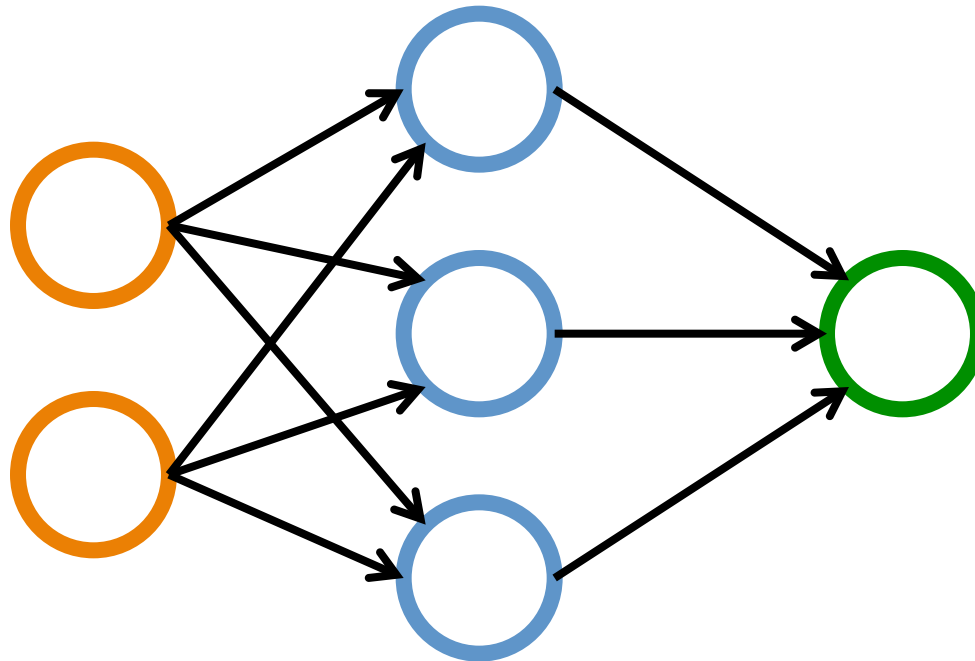


X		Y
Hours Sleep	Hours Study	Score
3	5	75
5	1	82
10	2	93
8	3	?

	X		Y
	Hours Sleep	Hours Study	Score
Training	3	5	75
	5	1	82
	10	2	93
Testing	8	3	?

Training a Network
=
Minimize the Cost Function

Neural Networks

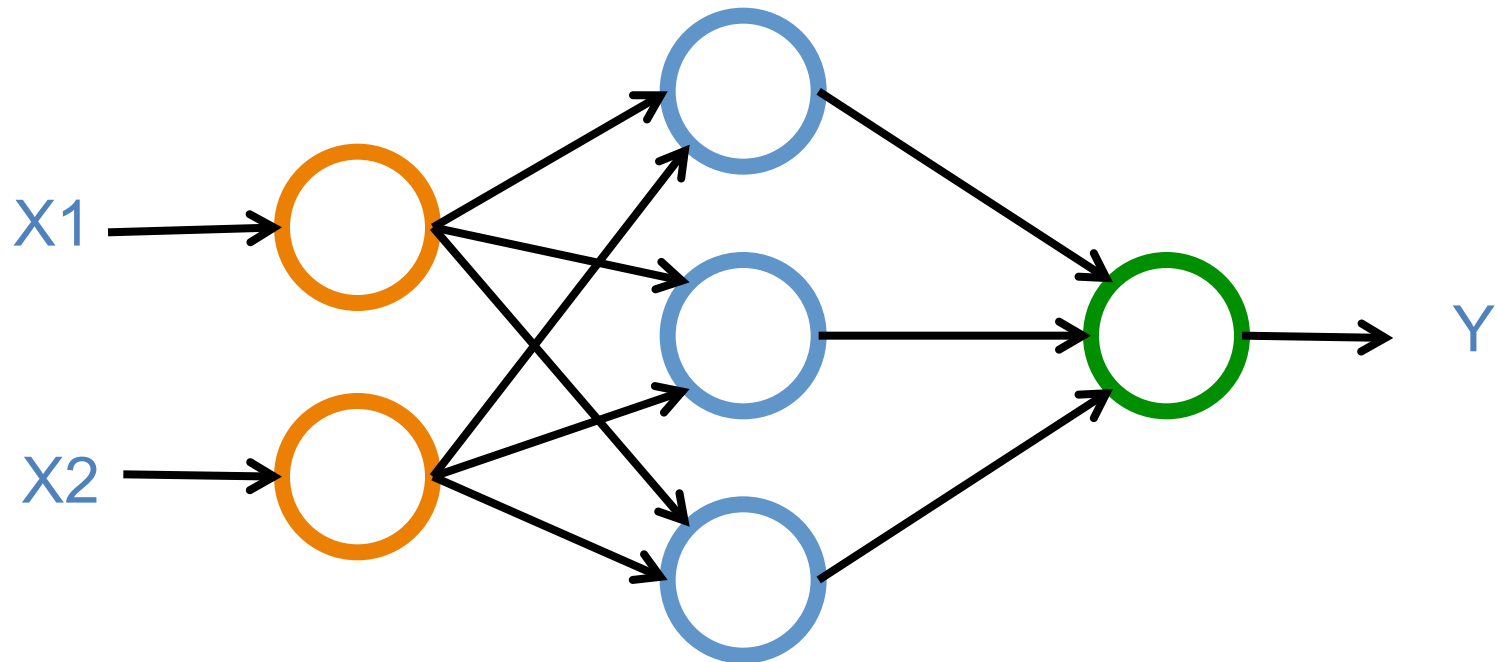


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



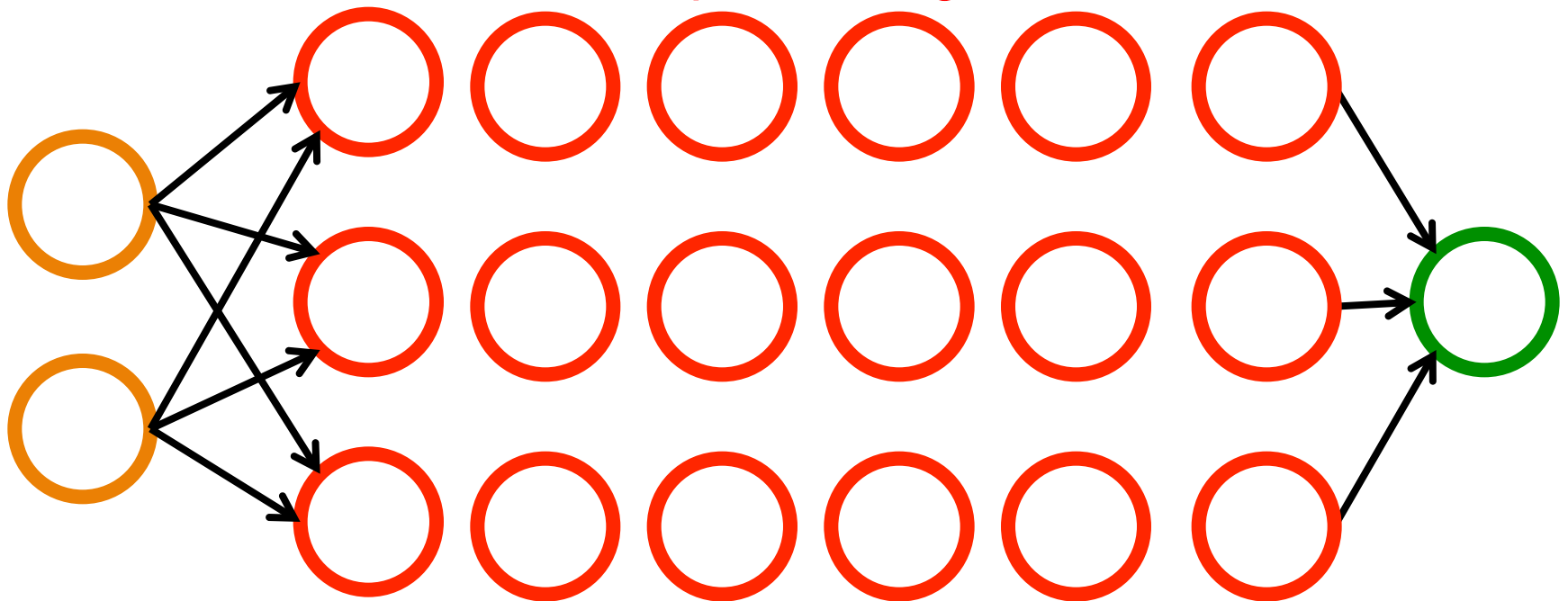
Neural Networks

Input Layer
(X)

Hidden Layers
(H)

Output Layer
(Y)

Deep Neural Networks
Deep Learning

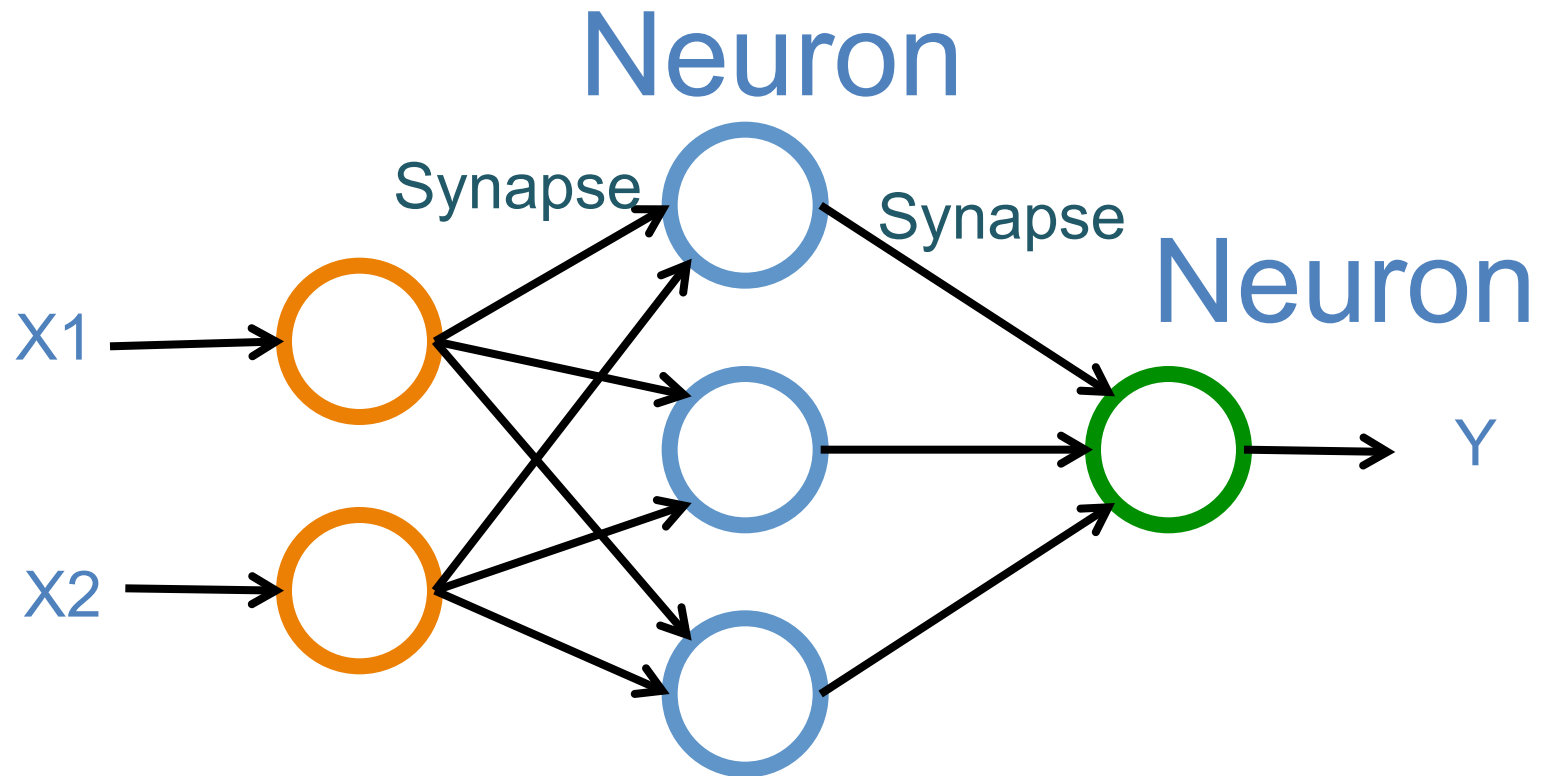


Neural Networks

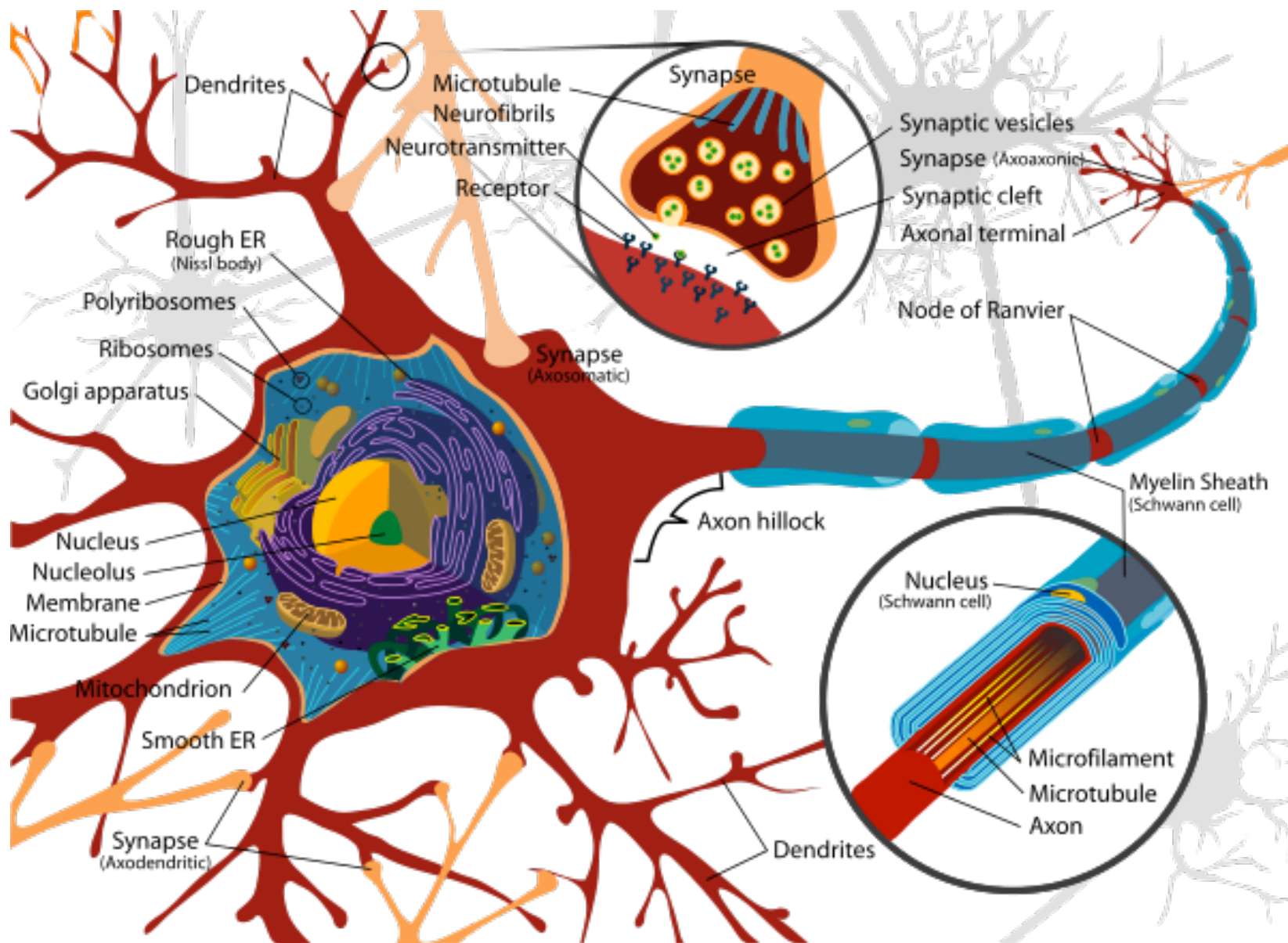
Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)



Neuron and Synapse

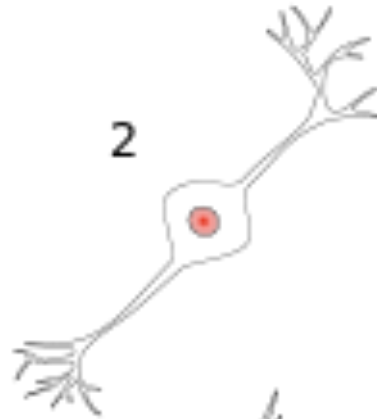


Neurons

1 Unipolar neuron



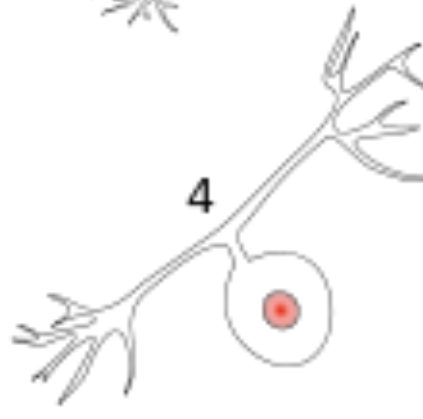
2 Bipolar neuron



3 Multipolar neuron



4 Pseudounipolar neuron

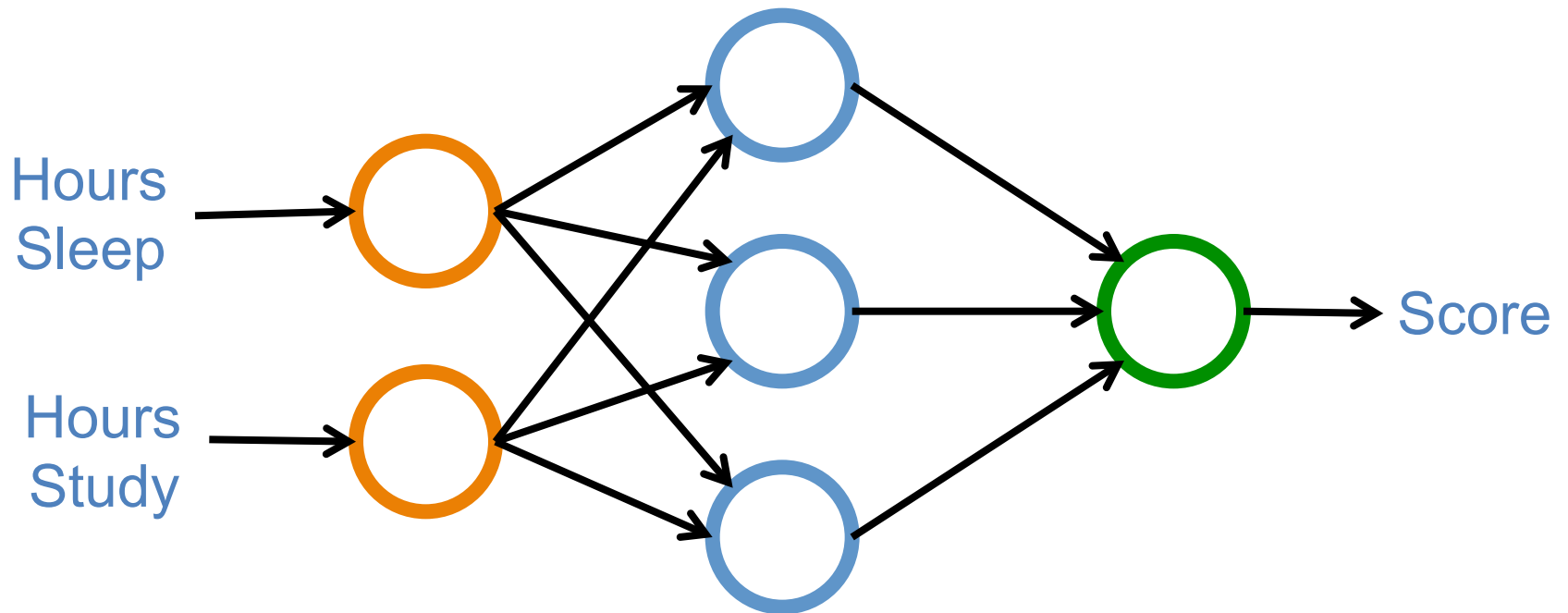


Neural Networks

Input Layer
(X)

Hidden Layer
(H)

Output Layer
(Y)

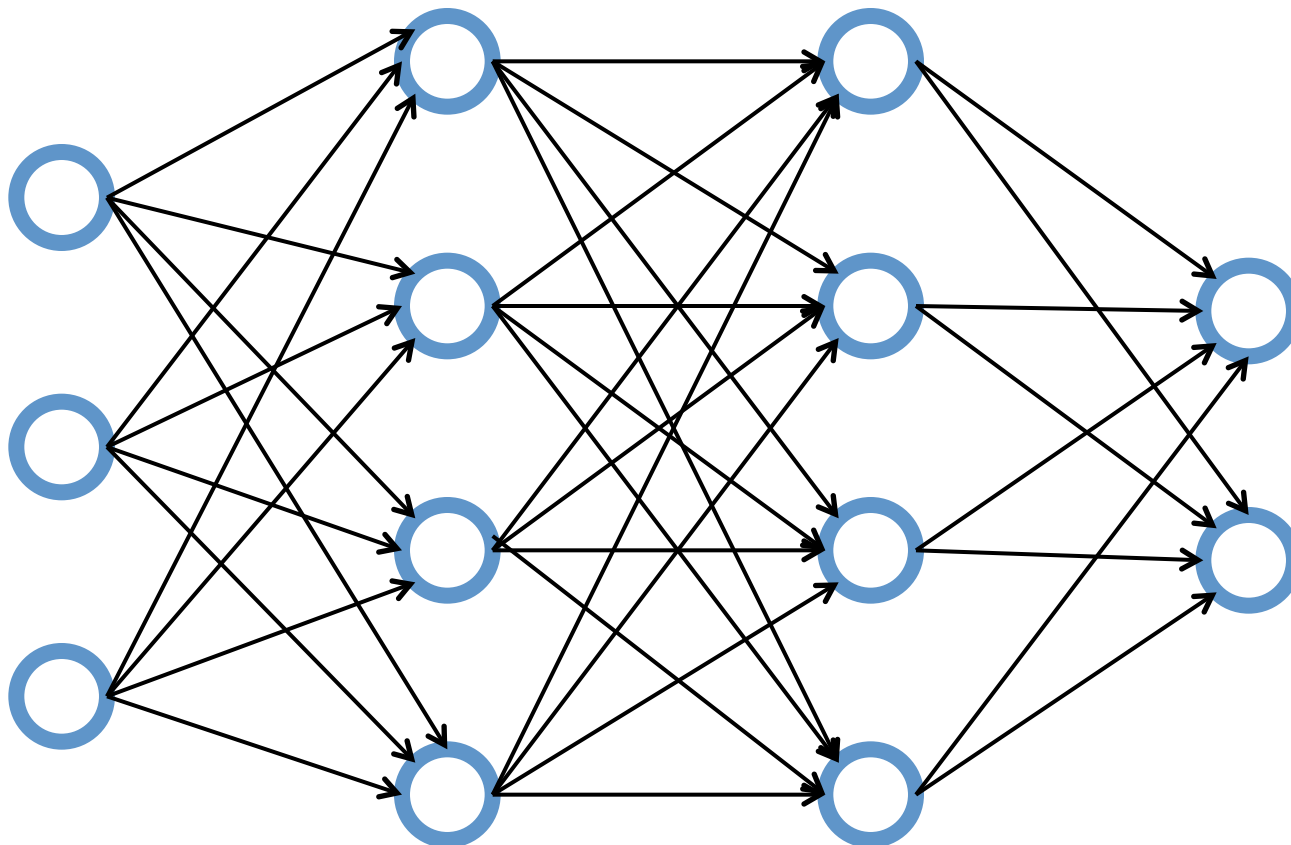


Neural Networks

Input Layer
(X)

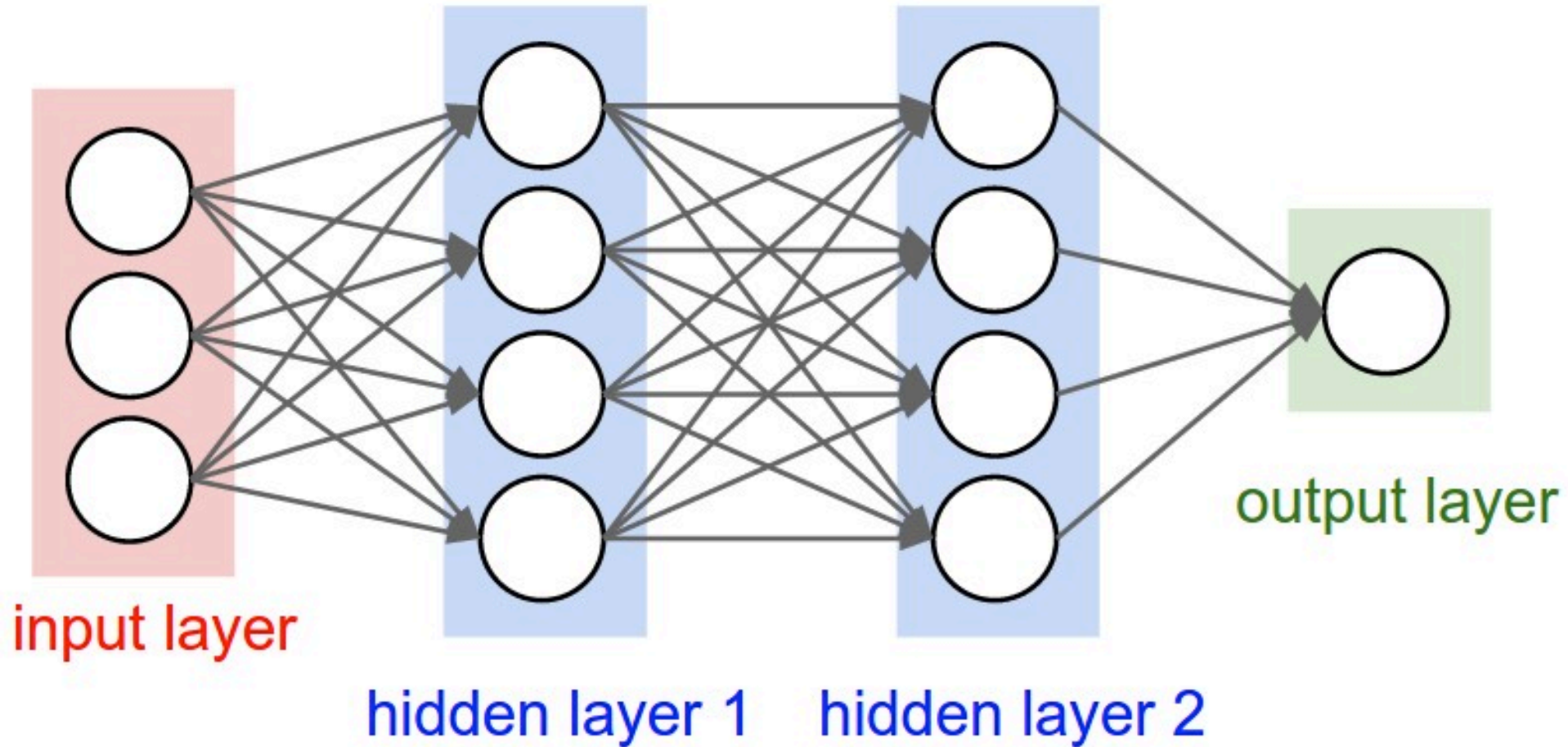
Hidden Layer
(H)

Output Layer
(Y)

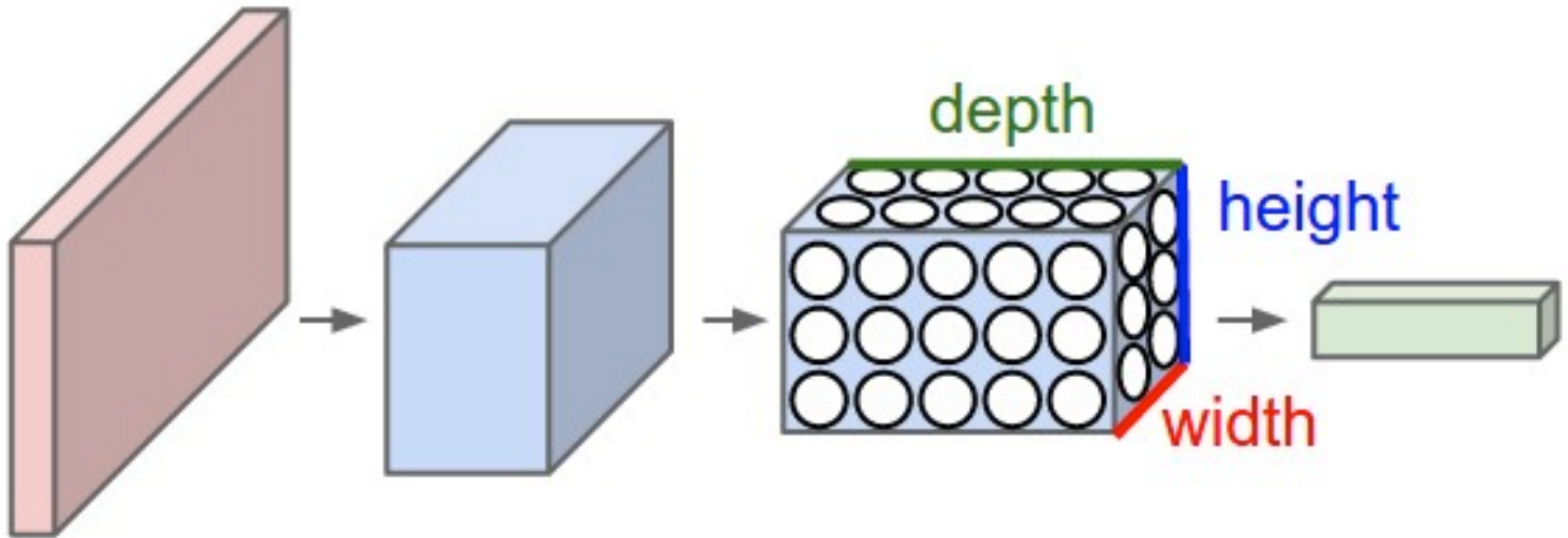


Convolutional Neural Networks (CNNs / ConvNets)

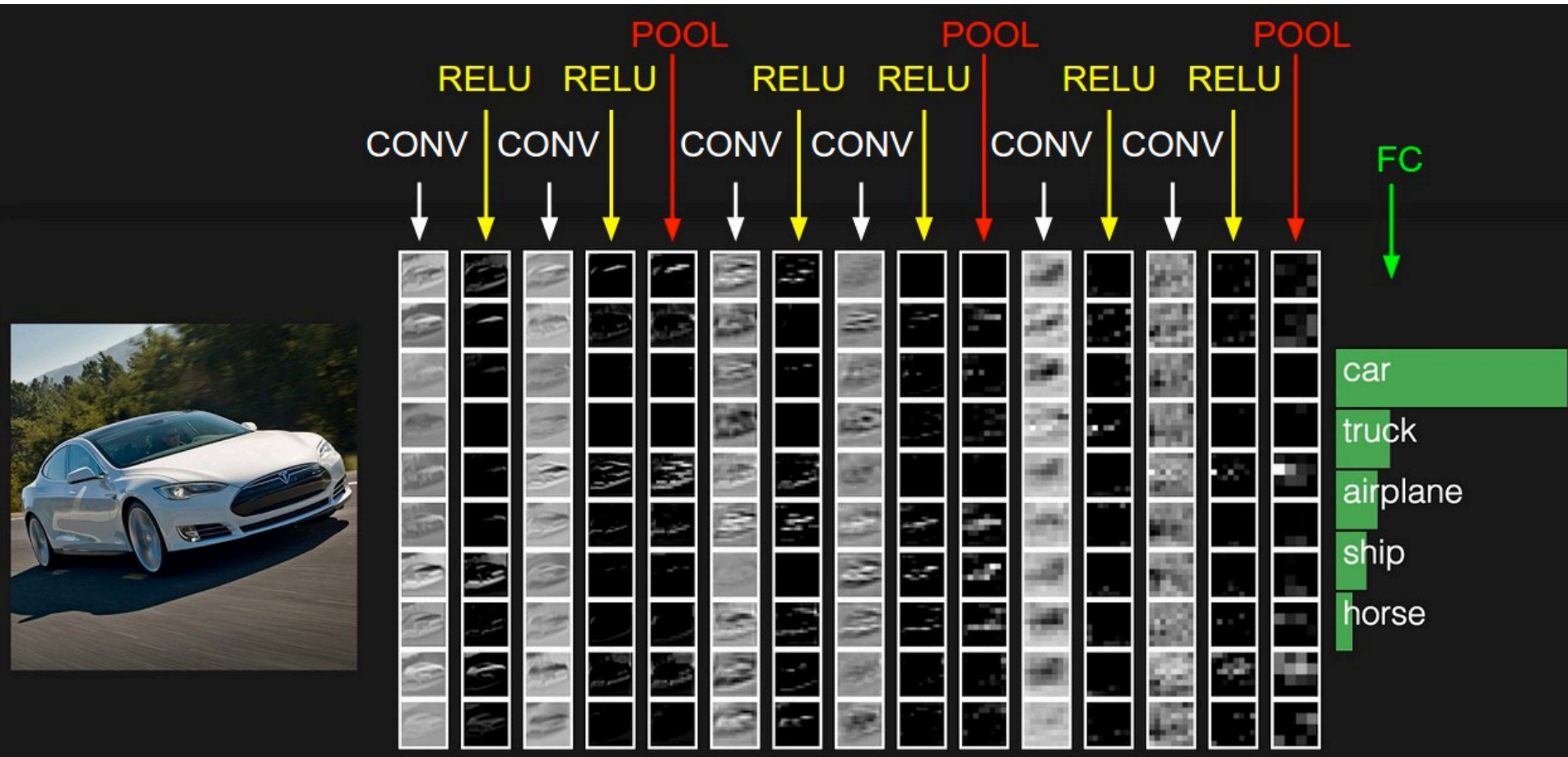
A regular 3-layer Neural Network



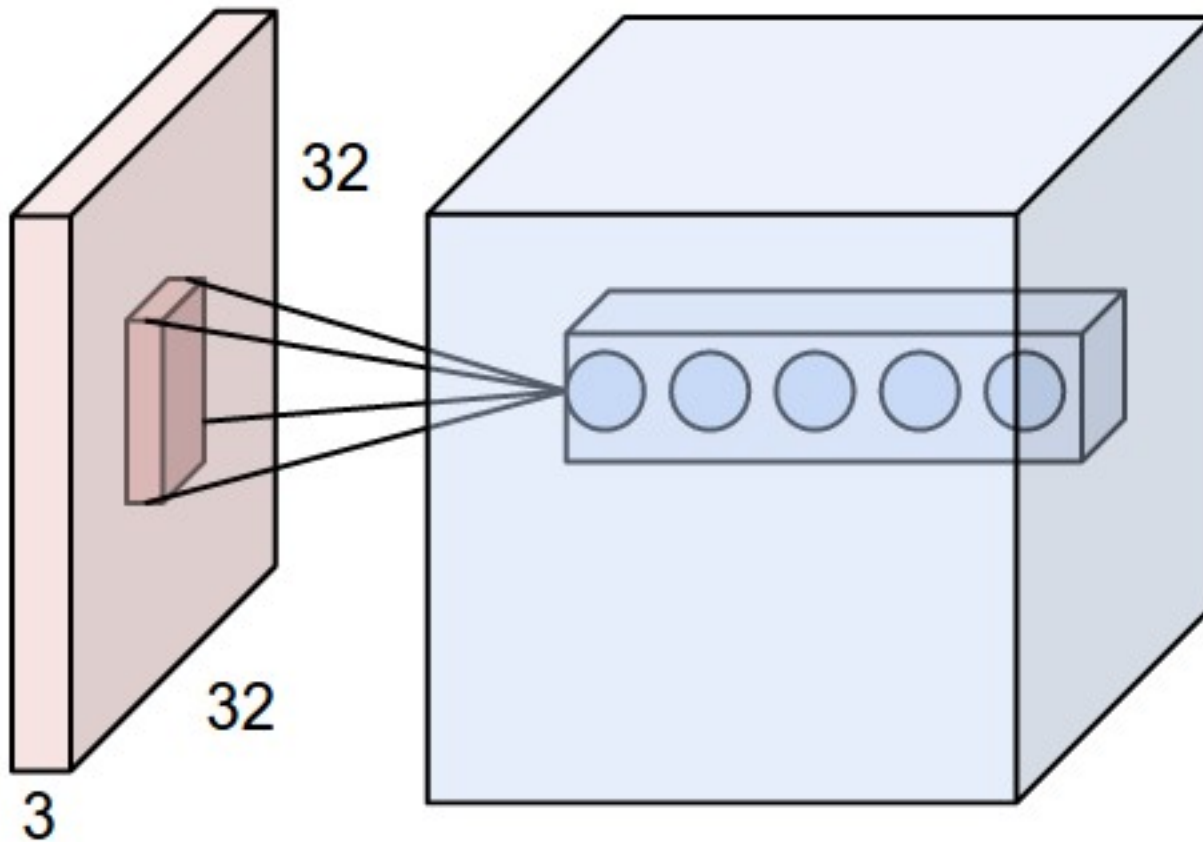
A ConvNet arranges its neurons in three dimensions (width, height, depth)



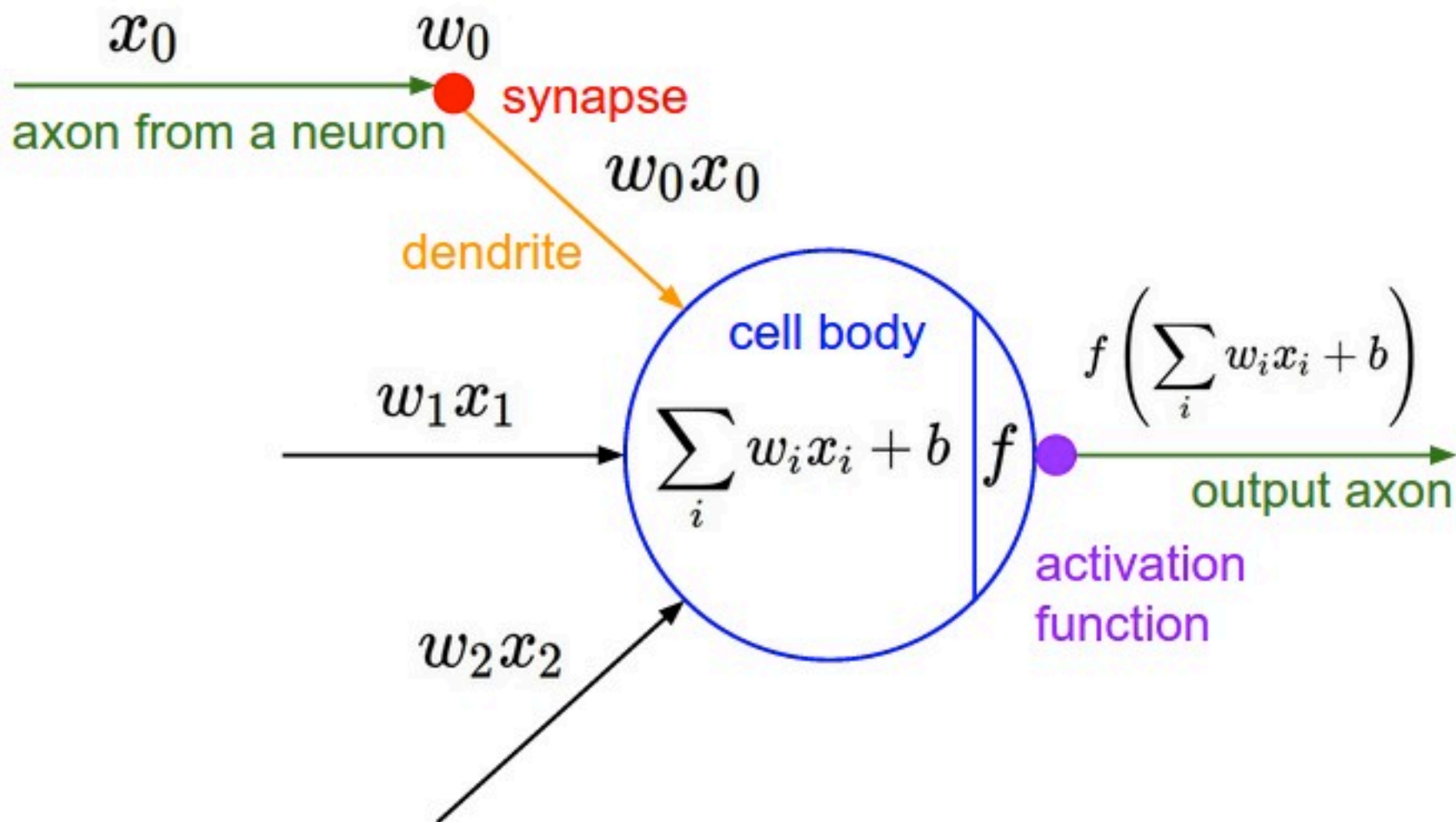
The activations of an example ConvNet architecture.



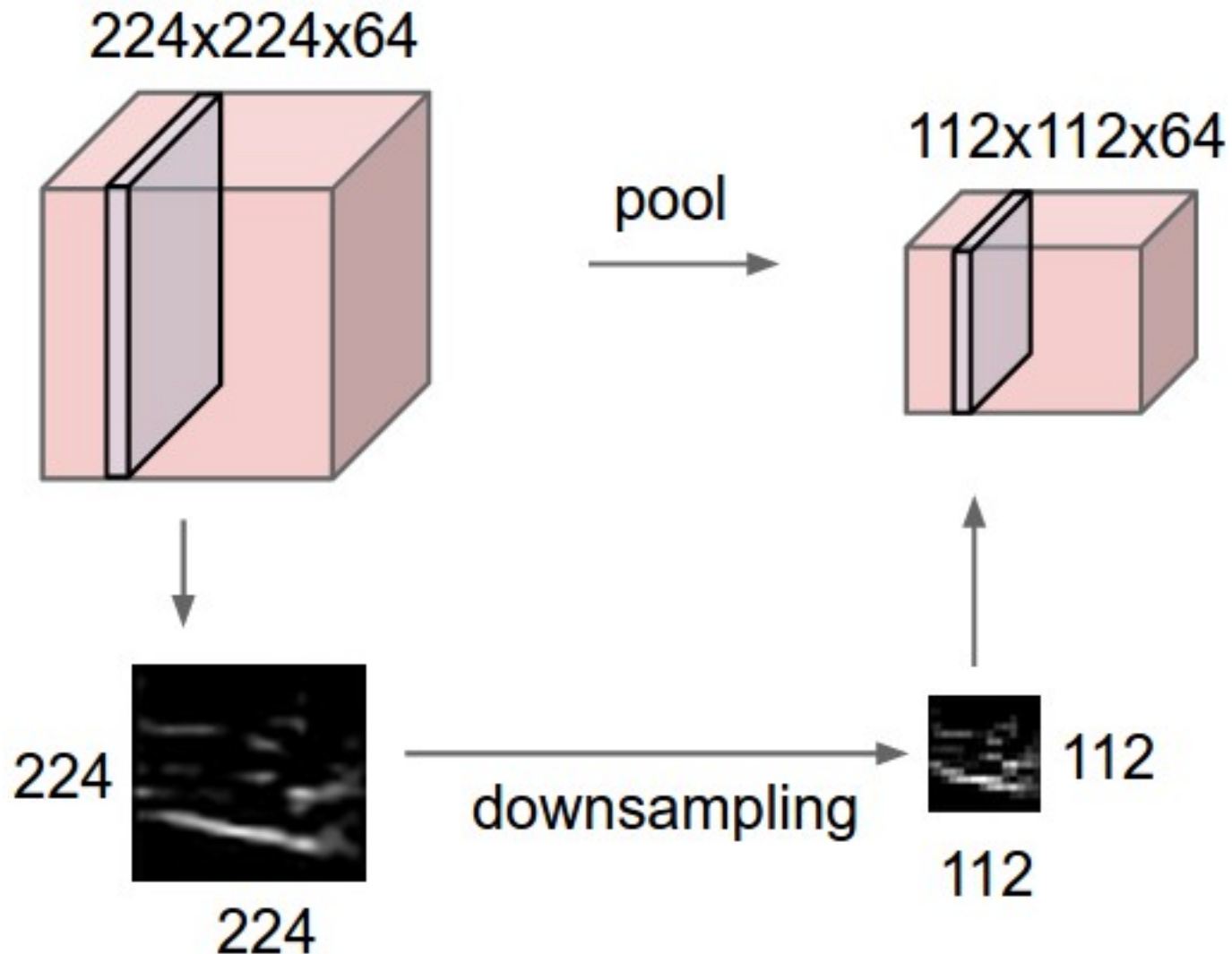
ConvNets



ConvNets



ConvNets



ConvNets

max pooling

Single depth slice

x

1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

y

max pool with 2x2 filters
and stride 2

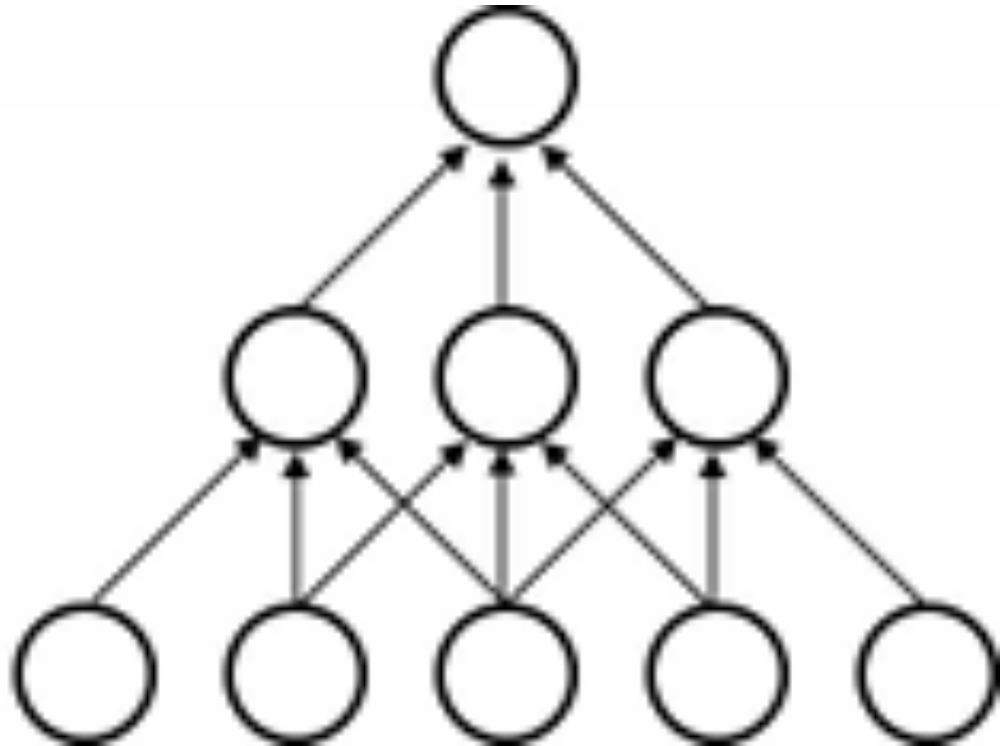
6	8
3	4

Convolutional Neural Networks (CNN) (LeNet) Sparse Connectivity

layer $m+1$

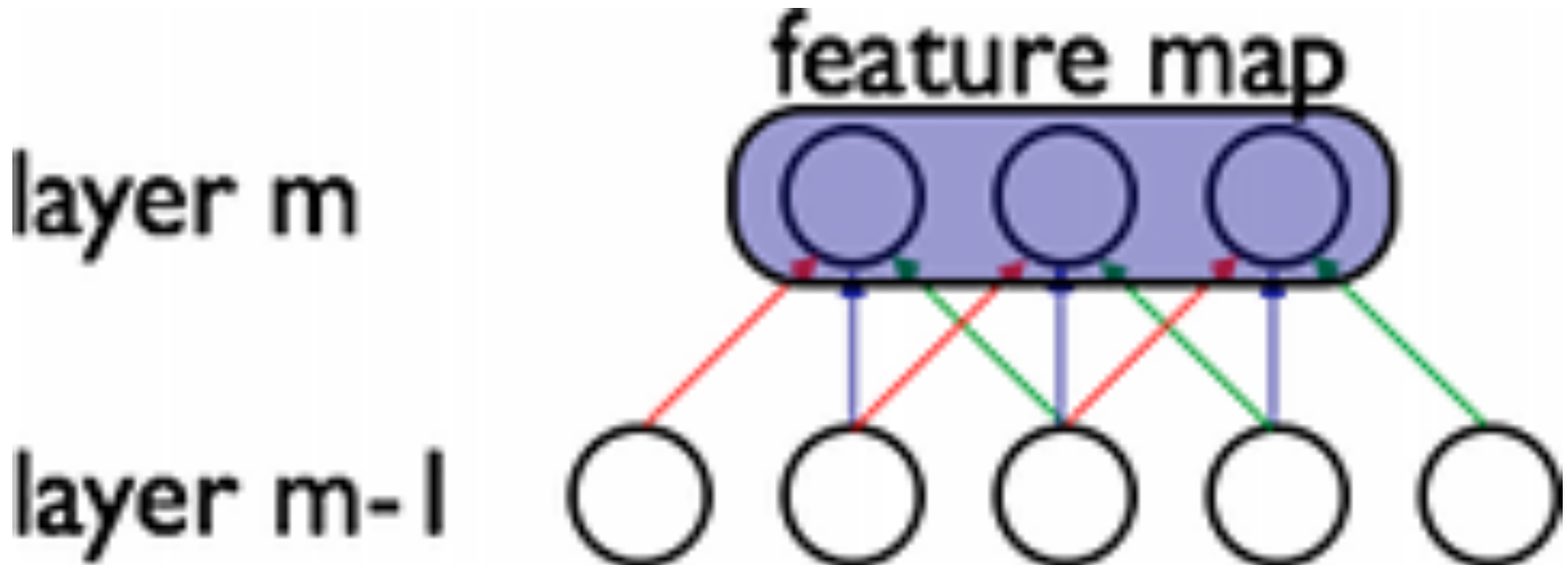
layer m

layer $m-1$



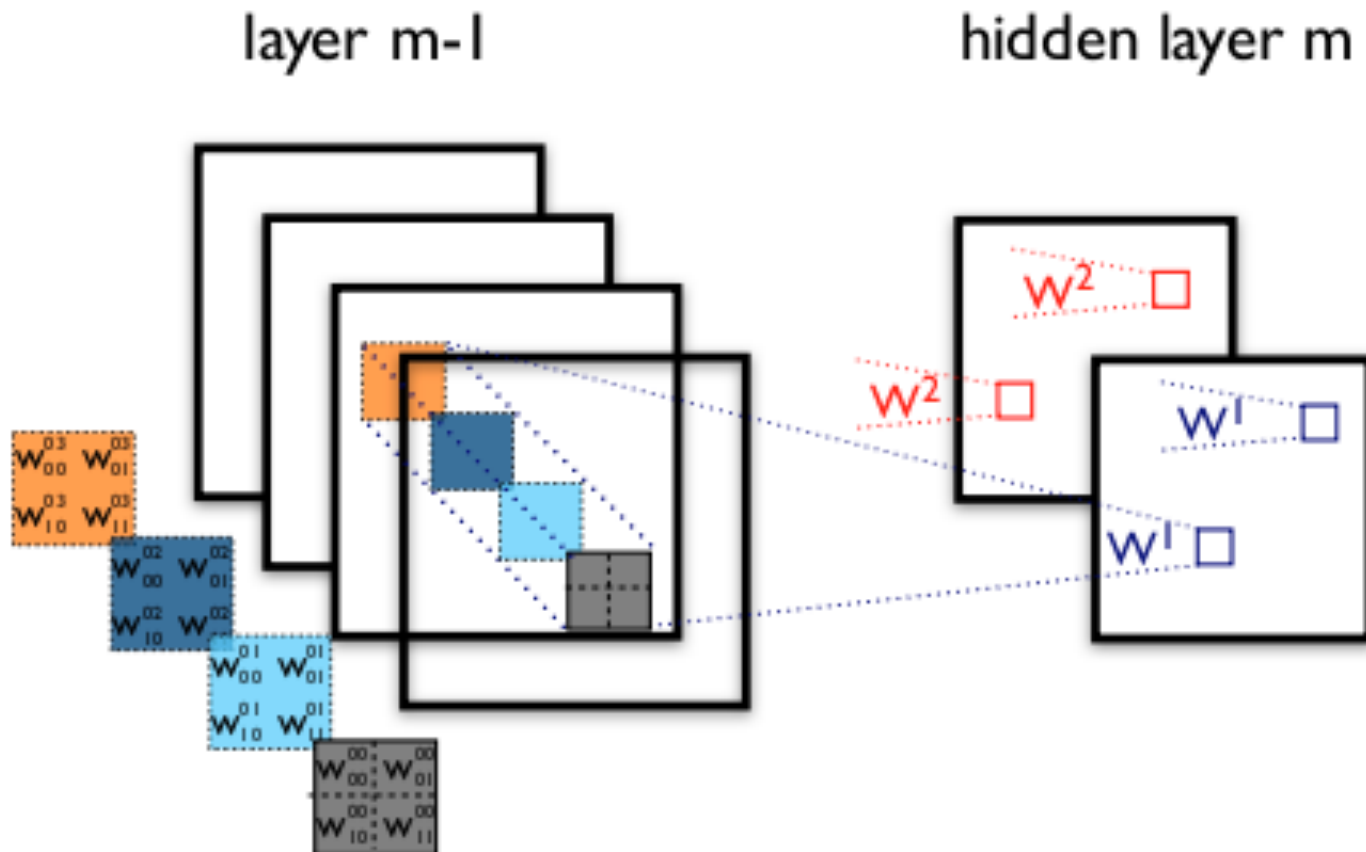
Convolutional Neural Networks (CNN) (LeNet)

Shared Weights

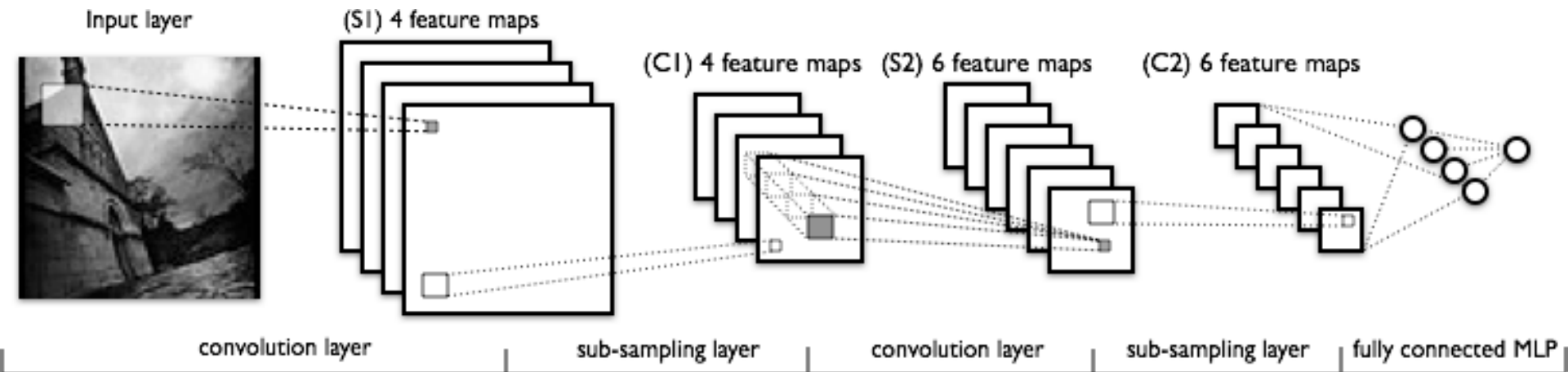


Convolutional Neural Networks (CNN) (LeNet)

example of a convolutional layer



Convolutional Neural Networks (CNN) (LeNet)



show flights from Boston to New York today

Recurrent Neural Networks with Word Embeddings

Semantic Parsing / Slot-Filling (Spoken Language Understanding)

Input (words)	show	flights	from	Boston	to	New	York	today
Output (labels)	O	O	O	B-dept	O	B-arr	I-arr	B-date

show flights from Boston to New York today

show flights from Boston to New York today

Input (words)	show	flights	from	Boston	to	New	York	today
Output (labels)	O	O	O	B-dept	O	B-arr	I-arr	B-date

Deep Learning Software

- **Theano**
 - CPU/GPU symbolic expression compiler in python (from MILA lab at University of Montreal)
- **Keras**
 - A theano based deep learning library.
- **Tensorflow**
 - TensorFlow™ is an open source software library for numerical computation using data flow graphs.

Welcome

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

- **tight integration with NumPy** – Use `numpy.ndarray` in Theano-compiled functions.
- **transparent use of a GPU** – Perform data-intensive calculations up to 140x faster than with CPU.(float32 only)
- **efficient symbolic differentiation** – Theano does your derivatives for function with one or many inputs.
- **speed and stability optimizations** – Get the right answer for `log(1+x)` even when `x` is really tiny.
- **dynamic C code generation** – Evaluate expressions faster.
- **extensive unit-testing and self-verification** – Detect and diagnose many types of errors.

Theano has been powering large-scale computationally intensive scientific investigations since 2007. But it is also approachable enough to be used in the classroom (University of Montreal's [deep learning/machine learning](#) classes).

News

- Theano 0.8 was released 21th March 2016. Everybody is encouraged to update.
- Multi-GPU.
- We added support for [CuDNN v5](#).
- We added support for `CNMeM` to speed up the GPU memory allocation.
- Theano 0.7 was released 26th March 2015. Everybody is encouraged to update.
- We support [cuDNN](#) if it is installed by the user.
- Open Machine Learning Workshop 2014 [presentation](#).
- Colin Raffel [tutorial on Theano](#).
- Ian Goodfellow did a [12h class with exercises on Theano](#).
- New technical report on Theano: [Theano: new features and speed improvements](#).
- [HPCS 2011 Tutorial](#). We included a few fixes discovered while doing the Tutorial.



You can watch a quick (20 minute) introduction to Theano given as a talk at [SciPy 2010](#) via streaming (or downloaded) video:

[Transparent GPU Computing With Theano](#). James Bergstra, SciPy 2010, June 30, 2010.

<http://deeplearning.net/software/theano/>

theano

Table Of Contents

[Welcome](#)
[News](#)
[Download](#)
[Status](#)
[Citing Theano](#)
[Documentation](#)
[Community](#)
[Help!](#)

- [How to Seek Help](#)
- [How to provide help](#)

Next topic

Release Notes

This Page

Show Source

Quick search

Enter search terms or a module, class or function name.



HOME

MILA

RESEARCH

TRAINING

FRANÇAIS

Montreal
Institute for
Learning
Algorithms



Dedicated to understanding the fundamentals of learning and intelligence

FACULTY



Yoshua Bengio



Pascal Vincent



Christopher Pal



Aaron Courville




Roland Memisevic



Laurent Charlin

<https://mila.umontreal.ca/en/>

Keras

 Keras Documentation

Search docs

Home

Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Guiding principles

Getting started: 30 seconds to Keras

Installation

Switching from Theano to TensorFlow

Support

Why this name, Keras?

Getting started

Guide to the Sequential model

Guide to the Functional API

FAQ

Models


About Keras models

Sequential

Model (functional API)

Layers

About Keras layers

 GitHub

Next »

Docs » Home

 Edit on GitHub

Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Keras is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either **TensorFlow** or **Theano**. It was 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.

Use Keras if you need a deep learning library that:

- allows for easy and fast prototyping (through total modularity, minimalism, and extensibility).
- supports both convolutional networks and recurrent networks, as well as combinations of the two.
- supports arbitrary connectivity schemes (including multi-input and multi-output training).
- runs seamlessly on CPU and GPU.

Read the documentation at [Keras.io](https://keras.io).

Keras is compatible with: **Python 2.7-3.5**.

Guiding principles

- **Modularity.** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.

<http://keras.io/>

pip install Theano

```
imyday — -bash — 80x24
iMydaytekiMacBook-Pro:~ imyday$ pip install Theano
Collecting Theano
  Downloading Theano-0.8.2.tar.gz (2.9MB)
    100% |████████████████████████████████████████| 2.9MB 46kB/s
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.7.1 in /anaconda/lib/python2.7/site-packages (from Theano)
Requirement already satisfied (use --upgrade to upgrade): scipy>=0.11 in /anaconda/lib/python2.7/site-packages (from Theano)
Requirement already satisfied (use --upgrade to upgrade): six>=1.9.0 in /anaconda/lib/python2.7/site-packages (from Theano)
Building wheels for collected packages: Theano
  Running setup.py bdist_wheel for Theano ... done
  Stored in directory: /Users/imyday/Library/Caches/pip/wheels/96/2b/3d/71d54e24a7171a4afb7144d1e944a7be643b448b23a35b9937
Successfully built Theano
Installing collected packages: Theano
Successfully installed Theano-0.8.2
You are using pip version 8.1.0, however version 8.1.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
iMydaytekiMacBook-Pro:~ imyday$
```


conda install pydot

```
imyday — -bash — 82x36
[iMydaytekiMacBook-Pro:~ imyday$ conda install pydot
Using Anaconda Cloud api site https://api.anaconda.org
Fetching package metadata: ....
Solving package specifications: .....

Package plan for installation in environment //anaconda:

The following packages will be downloaded:


```

package	build	
pyarsing-1.5.6	py27_0	61 KB
pydot-1.0.28	py27_0	35 KB
Total:		96 KB

```


The following NEW packages will be INSTALLED:

pydot:      1.0.28-py27_0

The following packages will be DOWNGRADED:

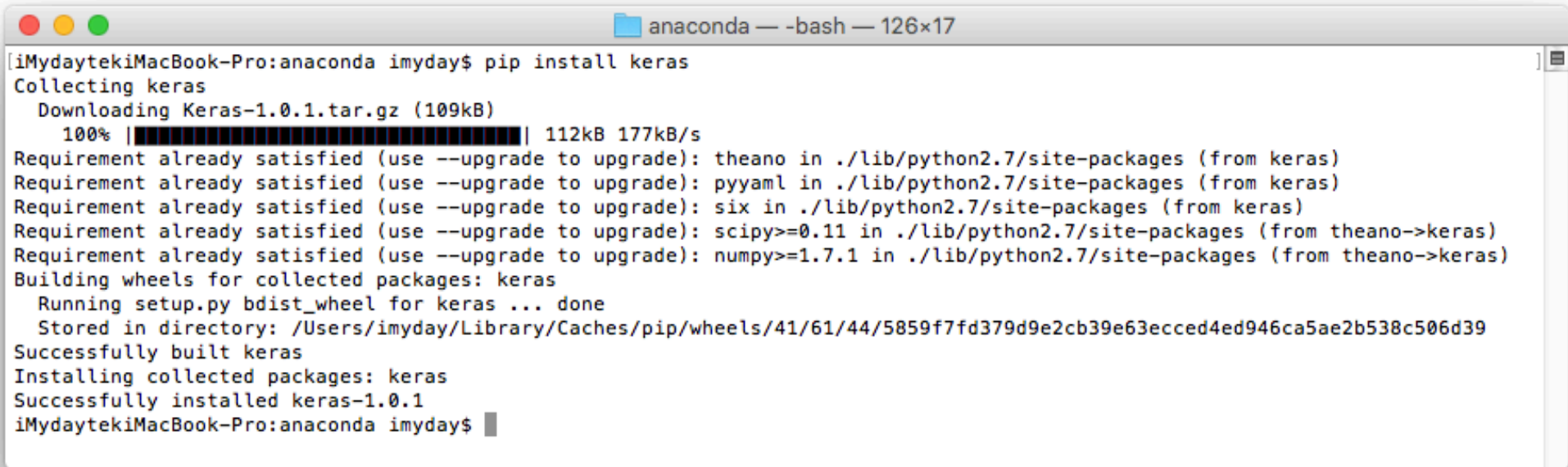
pyarsing:  2.0.3-py27_0 --> 1.5.6-py27_0

Proceed ([y]/n)? y

Fetching packages ...
pyarsing-1.5. 100% |#####| Time: 0:00:00 110.47 kB/s
pydot-1.0.28-p 100% |#####| Time: 0:00:00 71.53 kB/s
Extracting packages ...
[ COMPLETE ] |#####| 100%
Unlinking packages ...
[ COMPLETE ] |#####| 100%
Linking packages ...
[ COMPLETE ] |#####| 100%
iMydaytekiMacBook-Pro:~ imyday$
```


`sudo pip install keras`

`pip install keras`



```
anaconda — -bash — 126x17
iMydaytekiMacBook-Pro:anaconda imyday$ pip install keras
Collecting keras
  Downloading Keras-1.0.1.tar.gz (109kB)
    100% |#####| 112kB 177kB/s
Requirement already satisfied (use --upgrade to upgrade): theano in ./lib/python2.7/site-packages (from keras)
Requirement already satisfied (use --upgrade to upgrade): pyyaml in ./lib/python2.7/site-packages (from keras)
Requirement already satisfied (use --upgrade to upgrade): six in ./lib/python2.7/site-packages (from keras)
Requirement already satisfied (use --upgrade to upgrade): scipy>=0.11 in ./lib/python2.7/site-packages (from theano->keras)
Requirement already satisfied (use --upgrade to upgrade): numpy>=1.7.1 in ./lib/python2.7/site-packages (from theano->keras)
Building wheels for collected packages: keras
  Running setup.py bdist_wheel for keras ... done
  Stored in directory: /Users/imyday/Library/Caches/pip/wheels/41/61/44/5859f7fd379d9e2cb39e63ecced4ed946ca5ae2b538c506d39
Successfully built keras
Installing collected packages: keras
Successfully installed keras-1.0.1
iMydaytekiMacBook-Pro:anaconda imyday$
```

New Deep Learning in Jupyter Notebook



Files

Running

Clusters

Select items to perform actions on them.

Upload

New ▾



Text File

Folder

Terminal

Notebooks

Python 2



/ SCDBA / DeepLearning



..

Notebook list empty.

Theano Example

```
import theano
from theano import tensor as T

a = T.scalar()
b = T.scalar()

y = a * b

multiply = theano.function(inputs=[a, b], outputs=y)

print(multiply(2, 3)) #6
print(multiply(4, 5)) #20
```

Theano Example

```
In [1]: #https://github.com/Newmu/Theano-Tutorials
import theano
from theano import tensor as T

a = T.scalar()
b = T.scalar()

y = a * b

multiply = theano.function(inputs=[a, b], outputs=y)

print(multiply(2, 3)) #6
print(multiply(4, 5)) #20
```

WARNING (theano.configdefaults): g++ not detected ! Theano will be unable to execute optimized C-implementations (for both CPU and GPU) and will default to Python implementations. Performance will be severely degraded. To remove this warning, set Theano flags cxx to an empty string.

WARNING:theano.configdefaults:g++ not detected ! Theano will be unable to execute optimized C-implementations (for both CPU and GPU) and will default to Python implementations. Performance will be severely degraded. To remove this warning, set Theano flags cxx to an empty string.

```
6.0
20.0
```

Keras Example: imdb.lstm.py

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility

from keras.preprocessing import sequence
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM, SimpleRNN, GRU
from keras.datasets import imdb

max_features = 20000
maxlen = 80 # cut texts after this number of words (among top max_features most common words)
batch_size = 32

print('Loading data...')
(X_train, y_train), (X_test, y_test) = imdb.load_data(nb_words=max_features,
                                                         test_split=0.2)

print(len(X_train), 'train sequences')
print(len(X_test), 'test sequences')

print('Pad sequences (samples x time)')
X_train = sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = sequence.pad_sequences(X_test, maxlen=maxlen)
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)
```

Keras Example: imdb.lstm.py

```
print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128, input_length=maxlen, dropout=0.2))
model.add(LSTM(128, dropout_W=0.2, dropout_U=0.2)) # try using a GRU instead, for fun
model.add(Dense(1))
model.add(Activation('sigmoid'))

# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

print('Train...')
print(X_train.shape)
print(y_train.shape)
model.fit(X_train, y_train, batch_size=batch_size, nb_epoch=15,
          validation_data=(X_test, y_test))
score, acc = model.evaluate(X_test, y_test,
                             batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Keras Example: imdb.Lstm.py

```
from __future__ import print_function
import numpy as np
np.random.seed(1337) # for reproducibility

from keras.preprocessing import sequence
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers.embeddings import Embedding
from keras.layers.recurrent import LSTM, SimpleRNN, GRU
from keras.datasets import imdb

max_features = 20000
maxlen = 80 # cut texts after this number of words (among top max_features most common words)
batch_size = 32

print('Loading data...')
(X_train, y_train), (X_test, y_test) = imdb.load_data(nb_words=max_features,
                                                    test_split=0.2)

print(len(X_train), 'train sequences')
print(len(X_test), 'test sequences')

print('Pad sequences (samples x time)')
X_train = sequence.pad_sequences(X_train, maxlen=maxlen)
X_test = sequence.pad_sequences(X_test, maxlen=maxlen)
print('X_train shape:', X_train.shape)
print('X_test shape:', X_test.shape)

print('Build model...')
model = Sequential()
model.add(Embedding(max_features, 128, input_length=maxlen, dropout=0.2))
model.add(LSTM(128, dropout_W=0.2, dropout_U=0.2)) # try using a GRU instead, for fun
model.add(Dense(1))
model.add(Activation("sigmoid"))

# try using different optimizers and different optimizer configs
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

print('Train...')
print(X_train.shape)
print(y_train.shape)
model.fit(X_train, y_train, batch_size=batch_size, nb_epoch=15,
          validation_data=(X_test, y_test))
score, acc = model.evaluate(X_test, y_test,
                             batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Keras Example: imdb.lstm.py

Deep Learning (Training...)

```
Using Theano backend.
WARNING (theano.configdefaults): g++ not detected ! Theano will be unable to execute optimize
d C-implementations (for both CPU and GPU) and will default to Python implementations. Perform
ance will be severely degraded. To remove this warning, set Theano flags cxx to an empty str
ing.
WARNING:theano.configdefaults:g++ not detected ! Theano will be unable to execute optimized C
-implementations (for both CPU and GPU) and will default to Python implementations. Performan
ce will be severely degraded. To remove this warning, set Theano flags cxx to an empty strin
g.

Loading data...
20000 train sequences
5000 test sequences
Pad sequences (samples x time)
X_train shape: (20000, 80)
X_test shape: (5000, 80)
Build model...
Train...
(20000, 80)
(20000,)
Train on 20000 samples, validate on 5000 samples
Epoch 1/15
 9888/20000 [=====>.....] - ETA: 164444s - loss: 0.6393 - acc: 0.6339
```


Keras Example: imdb.lstm.py

localhost:8888/notebooks/SCDBA/DeepLearning/DeepLearningKeras.ipynb

View site information



DeepLearningKeras

Last Checkpoint: Last Wednesday at 6:28 AM (autosaved)



File Edit View Insert Cell Kernel Help

Python 2

Code CellToolbar

```
print('Train...')
print(X_train.shape)
print(y_train.shape)
model.fit(X_train, y_train, batch_size=batch_size, nb_epoch=15,
          validation_data=(X_test, y_test))
score, acc = model.evaluate(X_test, y_test,
                             batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Using Theano backend.

WARNING (theano.configdefaults): g++ not detected ! Theano will be unable to execute optimized C-implementations (for both CPU and GPU) and will default to Python implementations. Performance will be severely degraded. To remove this warning, set Theano flags cxx to an empty string.

WARNING:theano.configdefaults:g++ not detected ! Theano will be unable to execute optimized C-implementations (for both CPU and GPU) and will default to Python implementations. Performance will be severely degraded. To remove this warning, set Theano flags cxx to an empty string.

Loading data...

20000 train sequences

5000 test sequences

Pad sequences (samples x time)

X_train shape: (20000, 80)

X_test shape: (5000, 80)

Build model...

Train...

(20000, 80)

(20000,)

Train on 20000 samples, validate on 5000 samples

Epoch 1/15

9920/20000 [=====>.....] - ETA: 163855s - loss: 0.6388 - acc: 0.6342

Keras Example: imdb_lstm.py



DeepLearningKeras

Last Checkpoint: Last Wednesday at 6:28 AM (autosaved)



File Edit View Insert Cell Kernel Help

Python 2

Code CellToolbar

```
print('Train...')
print(X_train.shape)
print(y_train.shape)
model.fit(X_train, y_train, batch_size=batch_size, nb_epoch=15,
          validation_data=(X_test, y_test))
score, acc = model.evaluate(X_test, y_test,
                             batch_size=batch_size)
print('Test score:', score)
print('Test accuracy:', acc)
```

Using Theano backend.

WARNING (theano.configdefaults): g++ not detected ! Theano will be unable to execute optimized C-implementations (for both CPU and GPU) and will default to Python implementations. Performance will be severely degraded. To remove this warning, set Theano flags cxx to an empty string.

WARNING:theano.configdefaults:g++ not detected ! Theano will be unable to execute optimized C-implementations (for both CPU and GPU) and will default to Python implementations. Performance will be severely degraded. To remove this warning, set Theano flags cxx to an empty string.

Loading data...

20000 train sequences

5000 test sequences

Pad sequences (samples x time)

X_train shape: (20000, 80)

X_test shape: (5000, 80)

Build model...

Train...

(20000, 80)

(20000,)

Train on 20000 samples, validate on 5000 samples

Epoch 1/15

16384/20000 [=====>.....] - ETA: 55247s - loss: 0.6005 - acc: 0.6699

Keras Example: imdb.lstm.py

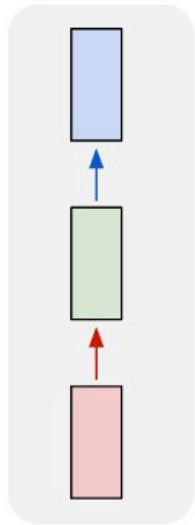
Deep Learning (Training...)

```
Loading data...
20000 train sequences
5000 test sequences
Pad sequences (samples x time)
X_train shape: (20000, 80)
X_test shape: (5000, 80)
Build model...
Train...
(20000, 80)
(20000,)
Train on 20000 samples, validate on 5000 samples
Epoch 1/15
20000/20000 [=====] - 312436s - loss: 0.5810 - acc: 0.6886 - val_loss: 0.4336 - val_acc: 0.8062
Epoch 2/15
1728/20000 [=>.....] - ETA: 250077s - loss: 0.4478 - acc: 0.7882
Epoch 2/15
928/20000 [>.....] - ETA: 260258s - loss: 0.4399 - acc: 0.7931
Epoch 1/15
19968/20000 [=====>.] - ETA: 479s - loss: 0.5812 - acc: 0.6885
Epoch 1/15
9888/20000 [=====>.....] - ETA: 164444s - loss: 0.6393 - acc: 0.6339
```

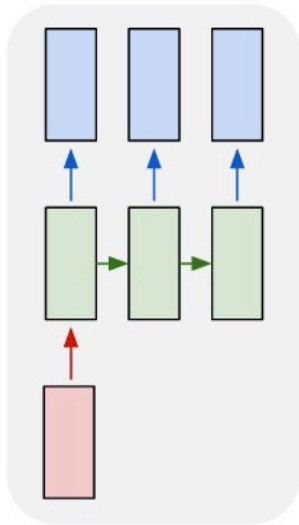
Keras

LSTM recurrent neural network

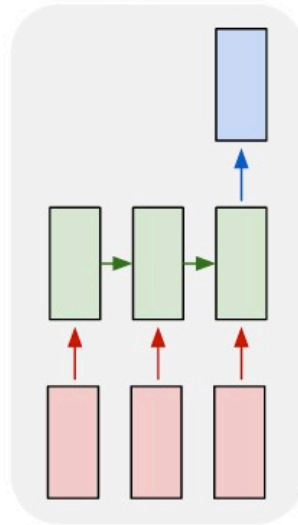
one to one



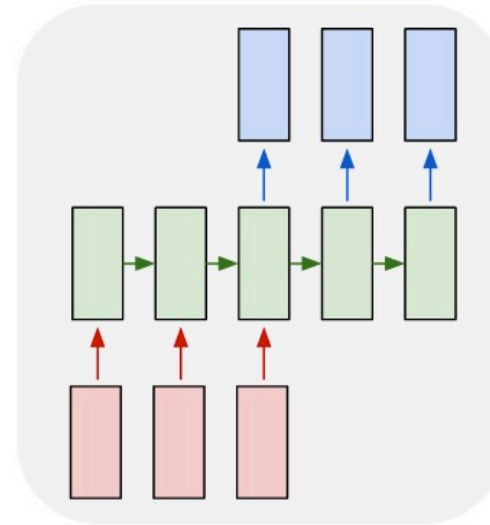
one to many



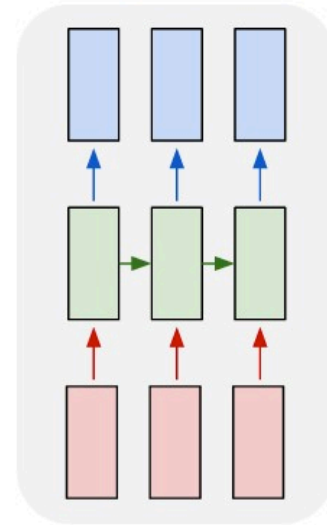
many to one



many to many



many to many



TensorFlow Playground

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.



Iterations
000,582

Learning rate

0.03

Activation

Tanh

Regularization

None

Regularization rate

0

Problem type

Classification

DATA

Which dataset do you want to use?



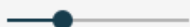
Ratio of training to test data: 50%



Noise: 0



Batch size: 10



INPUT

Which properties do you want to feed in?



3 HIDDEN LAYERS



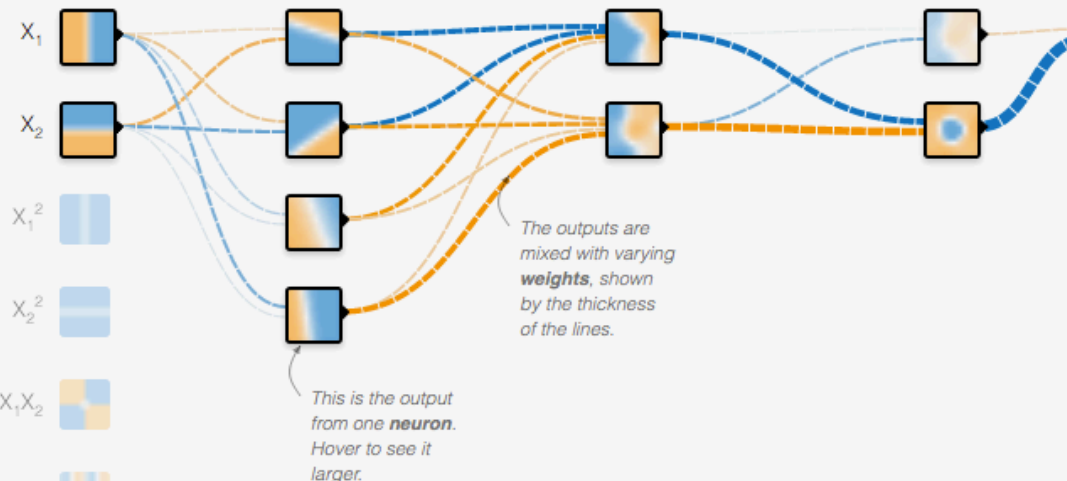
4 neurons



2 neurons

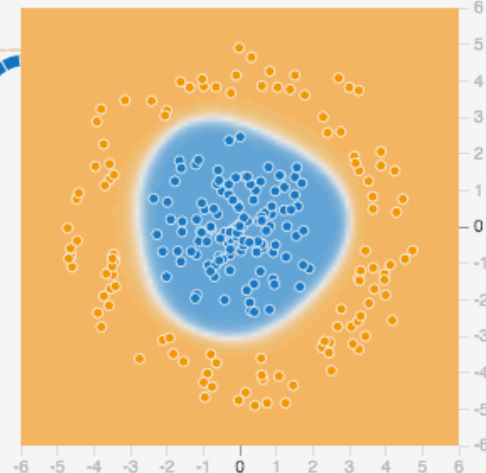


2 neurons



OUTPUT

Test loss 0.000
Training loss 0.000



References

- Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
- Sebastian Raschka (2015), Python Machine Learning, Packt Publishing
- Theano: <http://deeplearning.net/software/theano/>
- Keras: <http://keras.io/>
- Deep Learning Basics: Neural Networks Demystified, <https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU>
- Deep Learning SIMPLIFIED, <https://www.youtube.com/playlist?list=PLjJh1vISEYgvGod9wWiydumYl8hOXixNu>
- Deep Learning for NLP - Lecture 6 - Recurrent Neural Network, <https://www.youtube.com/watch?v=an9x3vXQYYQ>