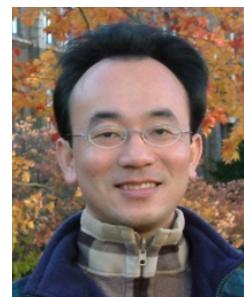


Big Data Mining 巨量資料探勘



Google TensorFlow 深度學習 (Deep Learning with Google TensorFlow)

1052DM10
MI4 (M2244) (3069)
Thu, 8, 9 (15:10-17:00) (B130)



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/>

2017-05-11



課程大綱 (Syllabus)

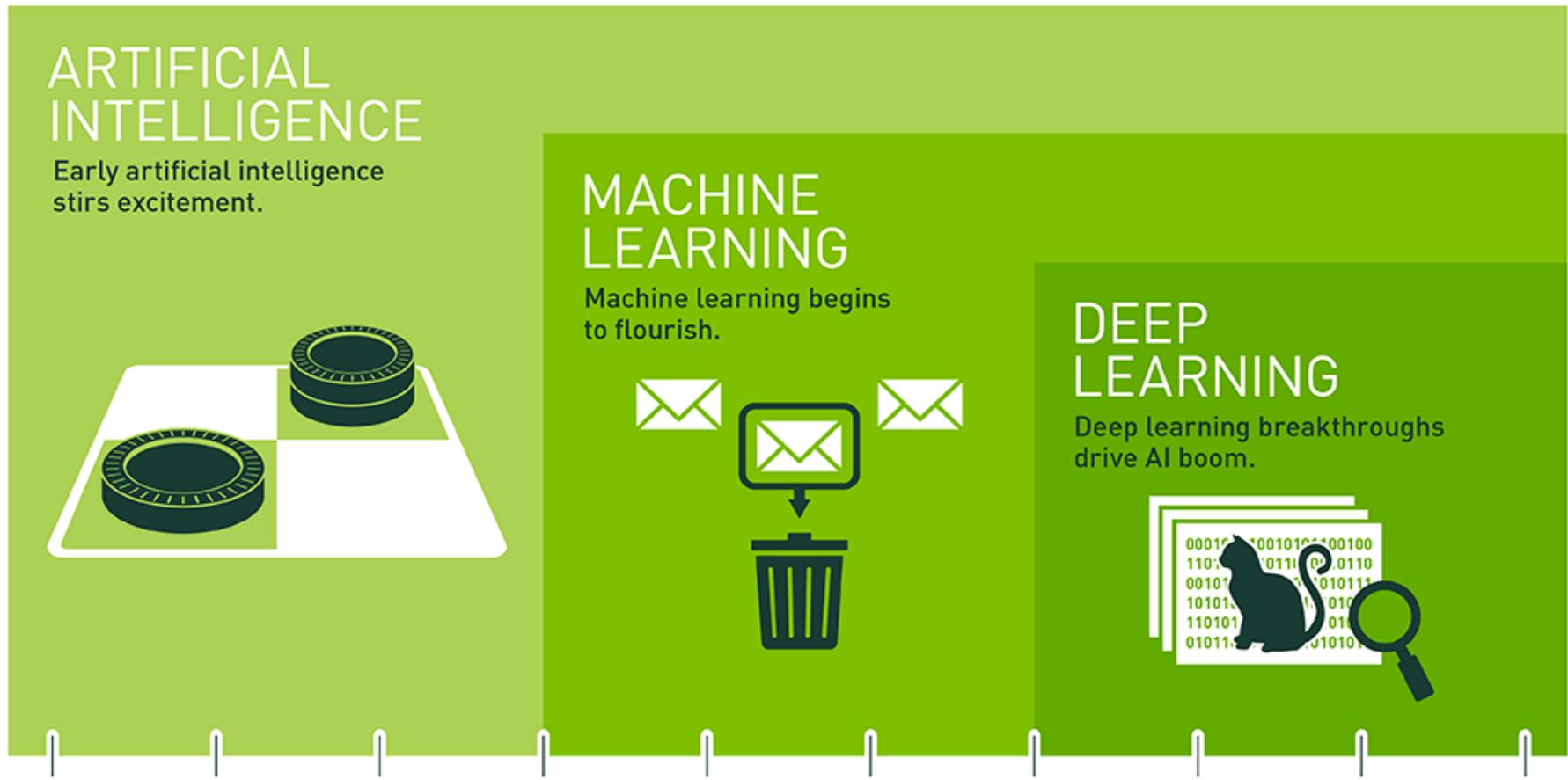
週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2017/02/16	巨量資料探勘課程介紹 (Course Orientation for Big Data Mining)
2	2017/02/23	巨量資料基礎：MapReduce典範、Hadoop與Spark生態系統 (Fundamental Big Data: MapReduce Paradigm, Hadoop and Spark Ecosystem)
3	2017/03/02	關連分析 (Association Analysis)
4	2017/03/09	分類與預測 (Classification and Prediction)
5	2017/03/16	分群分析 (Cluster Analysis)
6	2017/03/23	個案分析與實作一 (SAS EM 分群分析)： Case Study 1 (Cluster Analysis – K-Means using SAS EM)
7	2017/03/30	個案分析與實作二 (SAS EM 關連分析)： Case Study 2 (Association Analysis using SAS EM)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
8	2017/04/06	教學行政觀摩日 (Off-campus study)
9	2017/04/13	期中報告 (Midterm Project Presentation)
10	2017/04/20	期中考試週 (Midterm Exam)
11	2017/04/27	個案分析與實作三 (SAS EM 決策樹、模型評估) : Case Study 3 (Decision Tree, Model Evaluation using SAS EM)
12	2017/05/04	個案分析與實作四 (SAS EM 迴歸分析、類神經網路) : Case Study 4 (Regression Analysis, Artificial Neural Network using SAS EM)
13	2017/05/11	Google TensorFlow 深度學習 (Deep Learning with Google TensorFlow)
14	2017/05/18	期末報告 (Final Project Presentation)
15	2017/05/25	畢業班考試 (Final Exam)

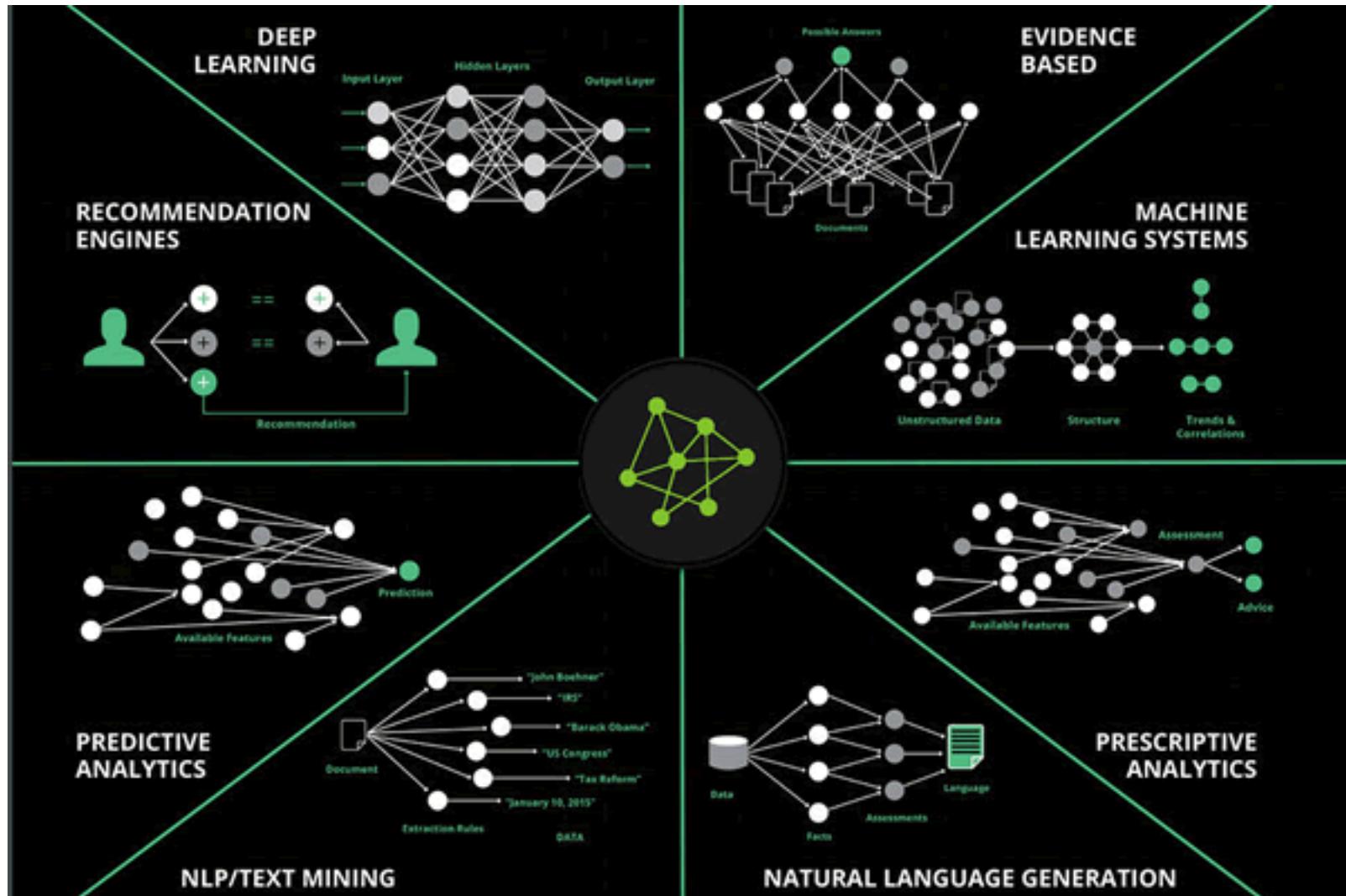
Artificial Intelligence

Machine Learning & Deep Learning



Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

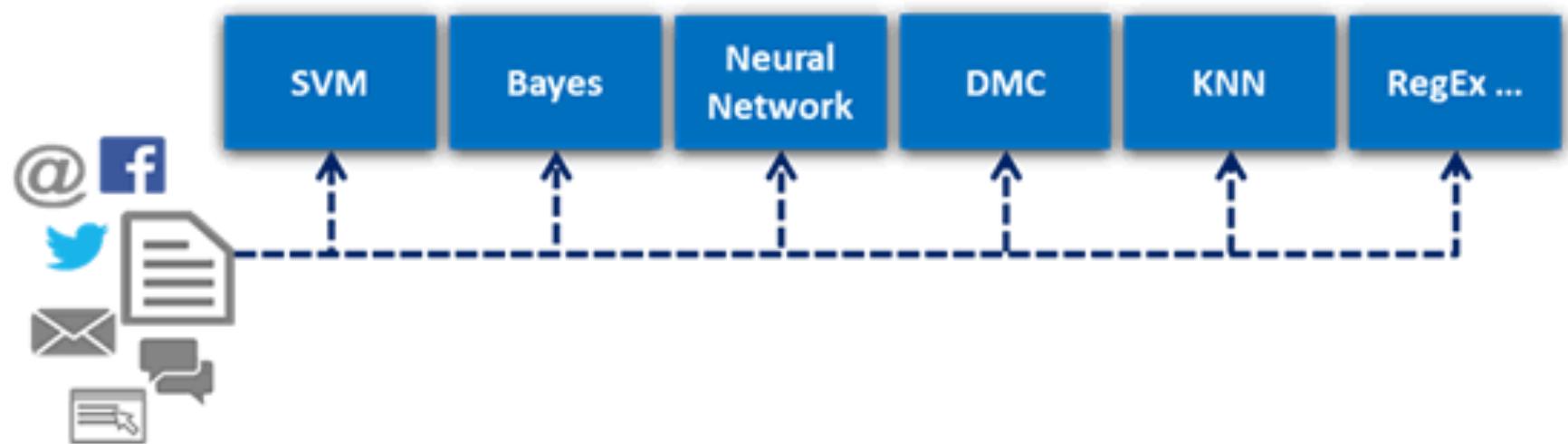
Artificial Intelligence (AI) is many things



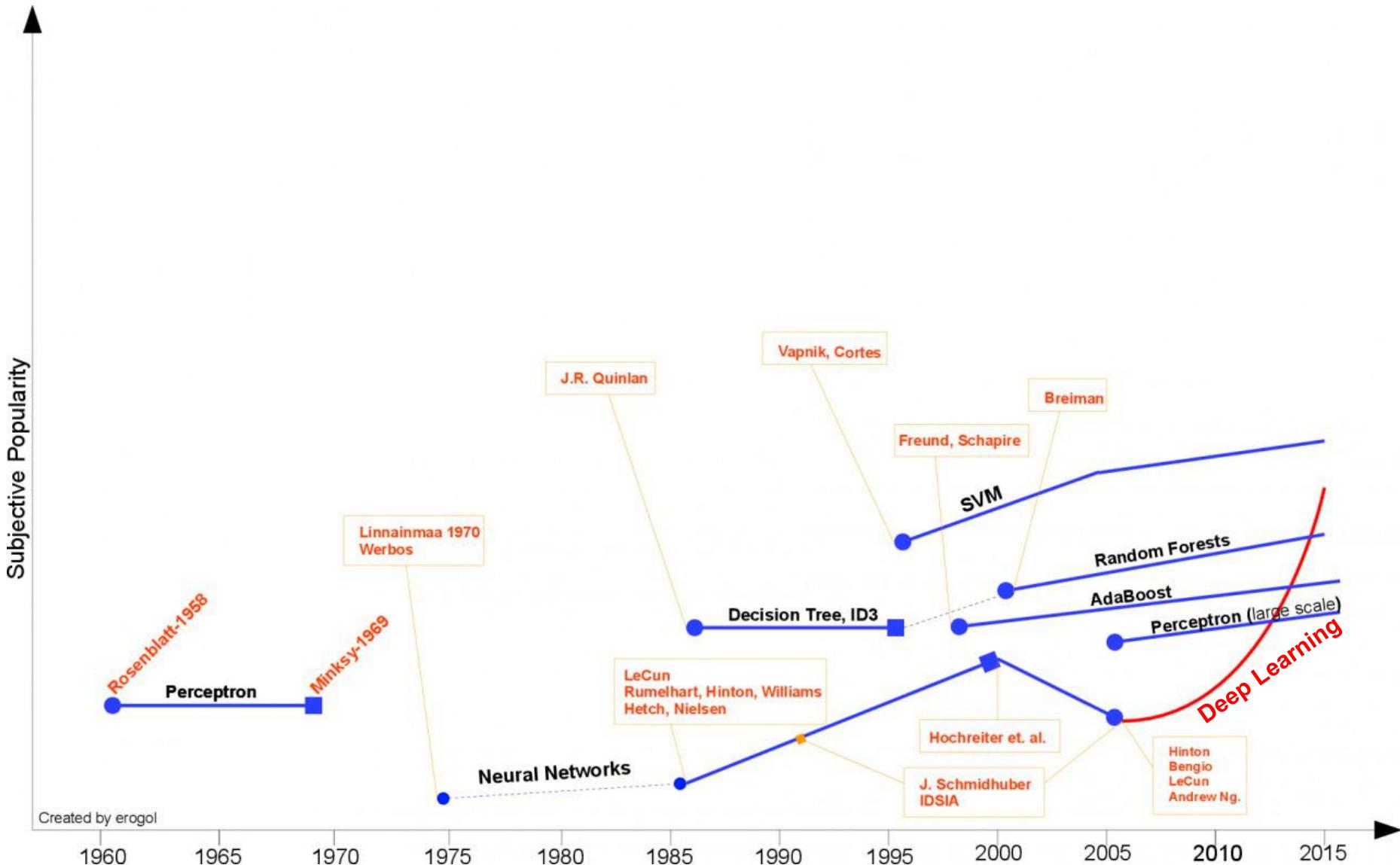
Ecosystem of AI

Artificial Intelligence (AI)

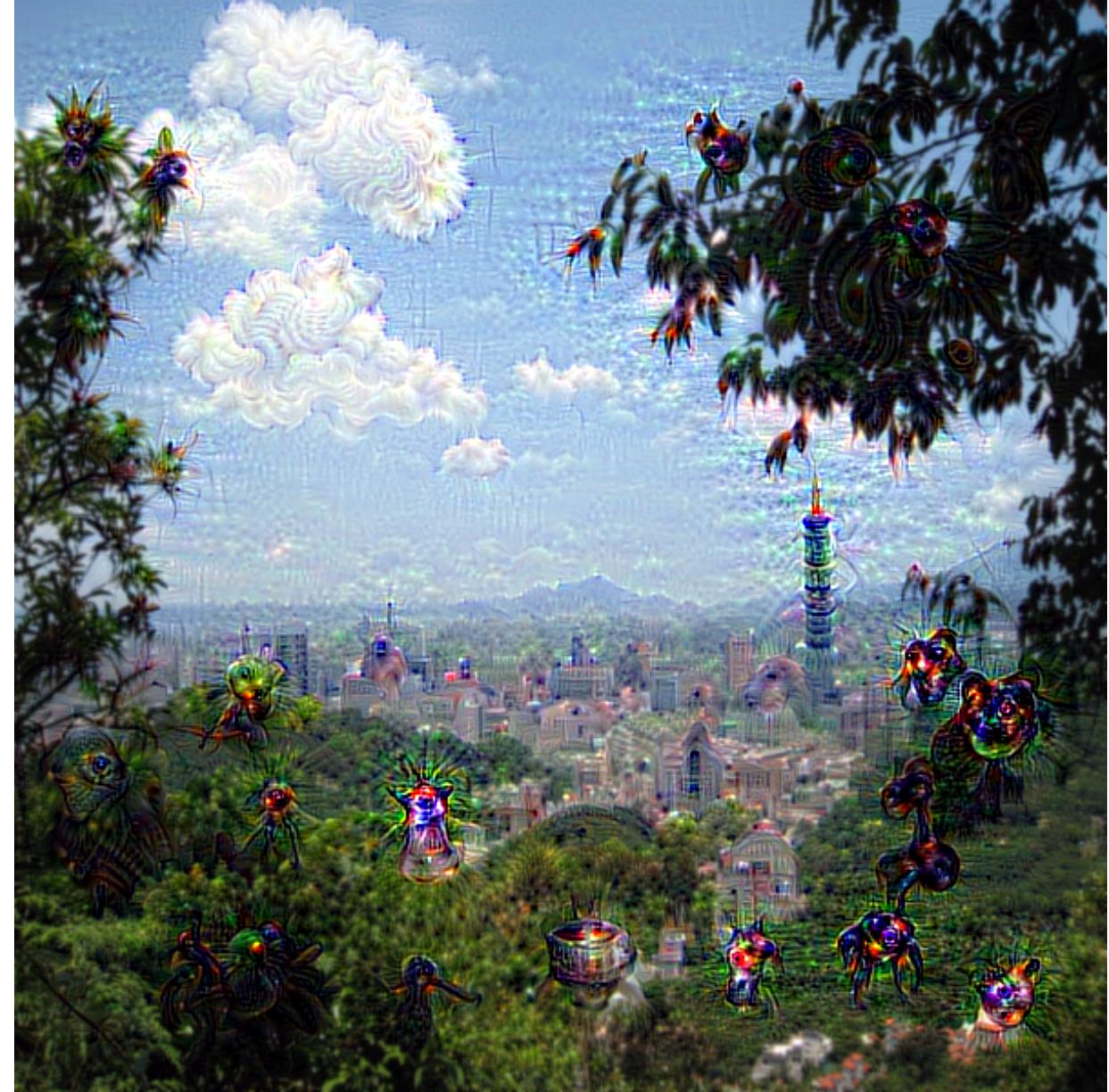
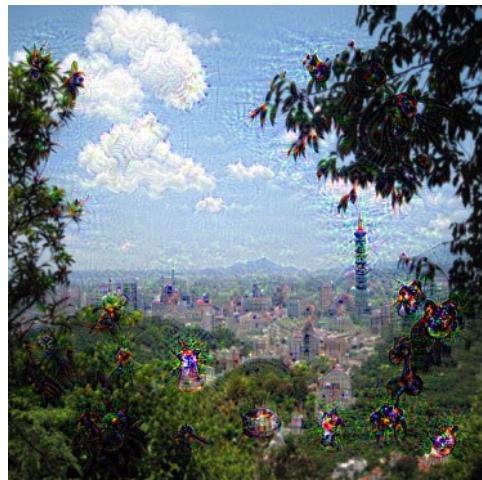
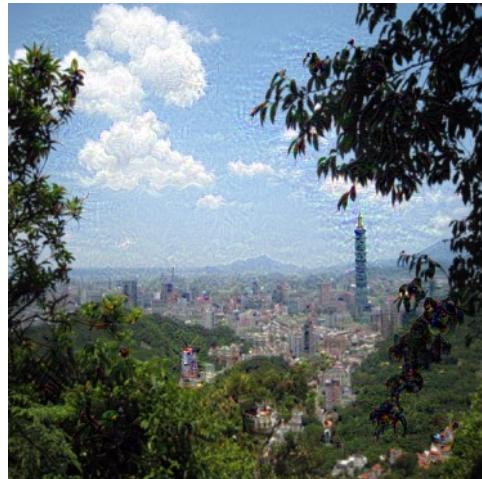
Intelligent Document Recognition algorithms



Deep Learning Evolution



Deep Dream



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}.

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

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

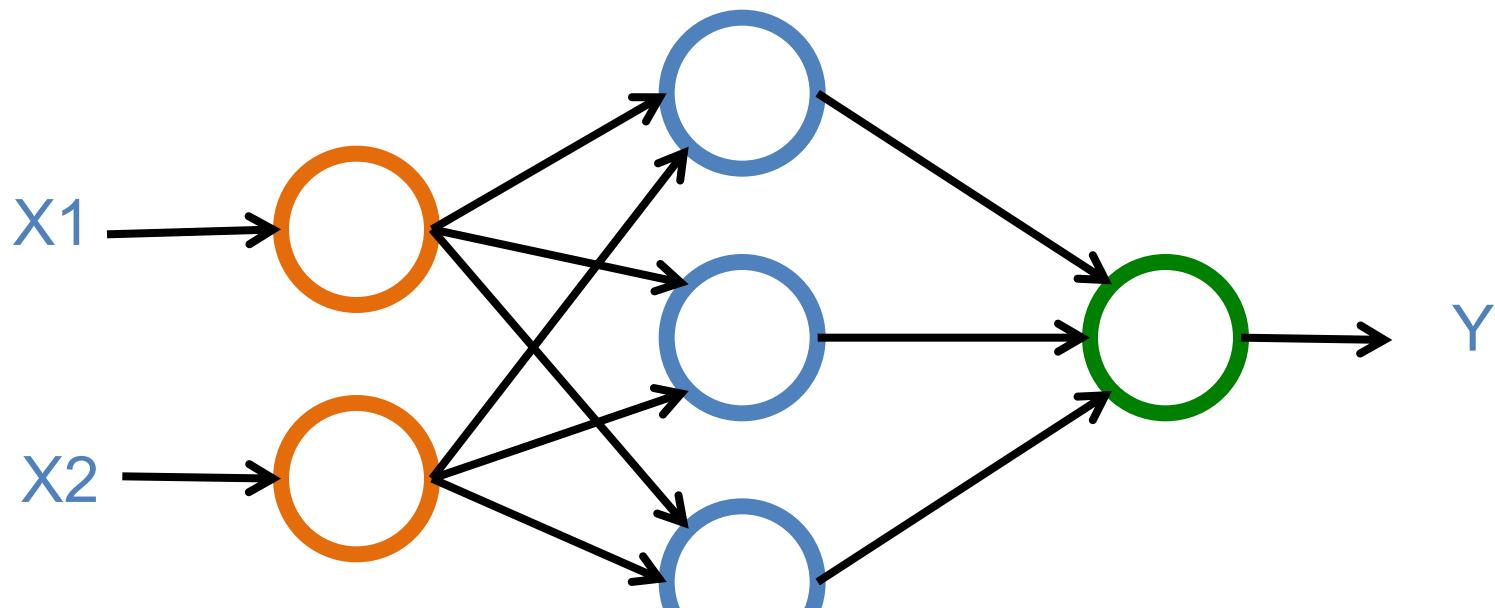
2010

Max-Pooling CDBN

2011

Neural Networks

Input Layer **Hidden Layer** **Output Layer**
(X) **(H)** **(Y)**



Deep Learning

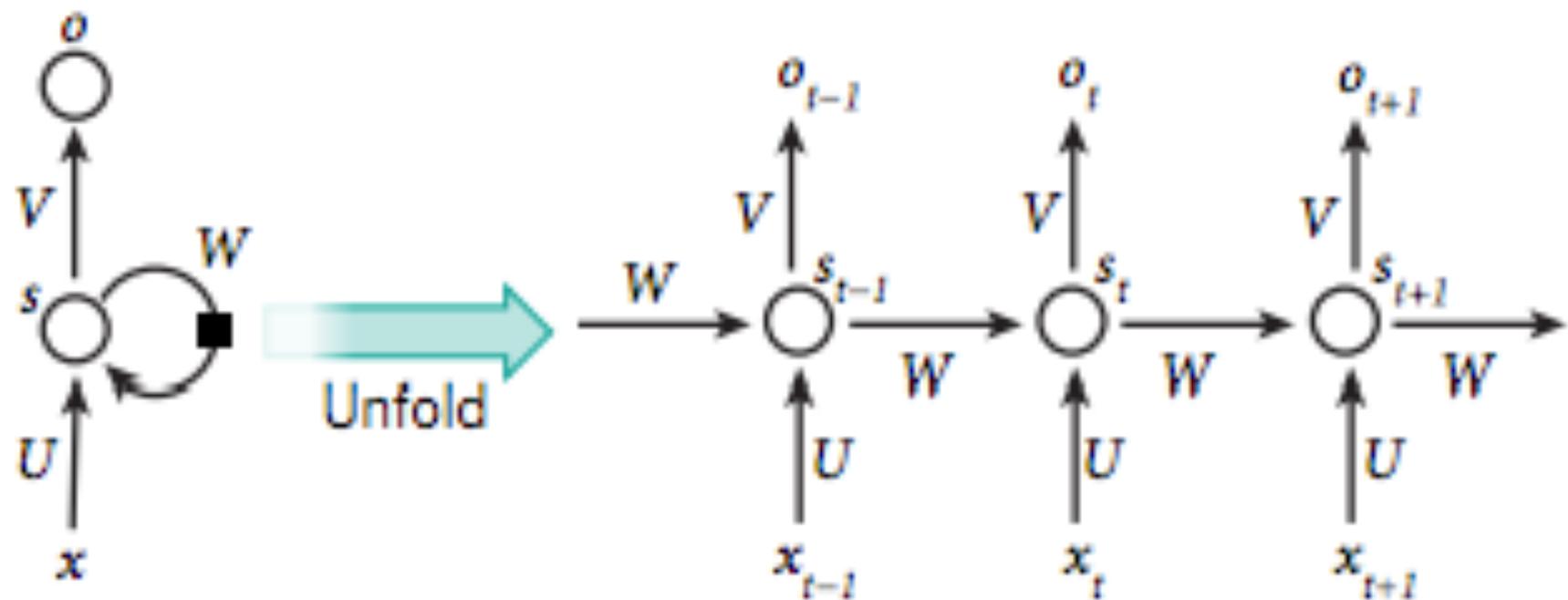
Geoffrey Hinton

Yann LeCun

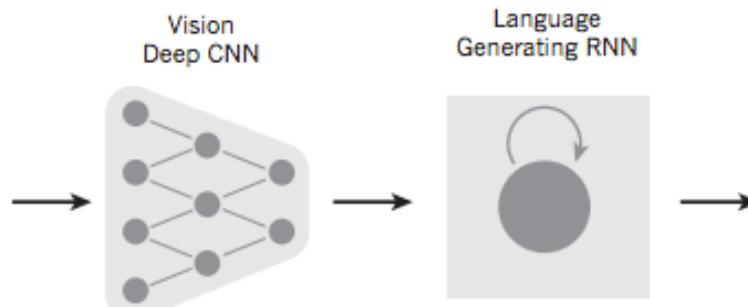
Yoshua Bengio

Andrew Y. Ng

Recurrent Neural Network (RNN)



From image to text



A group of people shopping at an outdoor market.

There are many vegetables at the fruit stand.



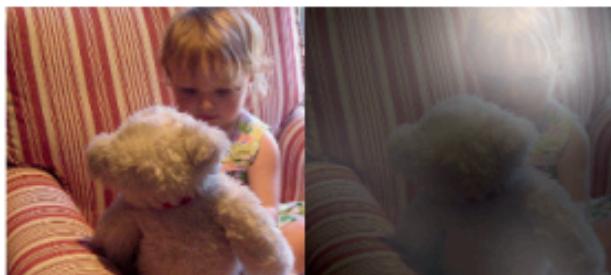
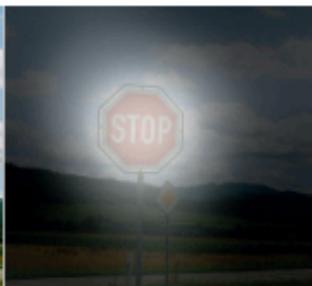
A woman is throwing a **frisbee** in a park.



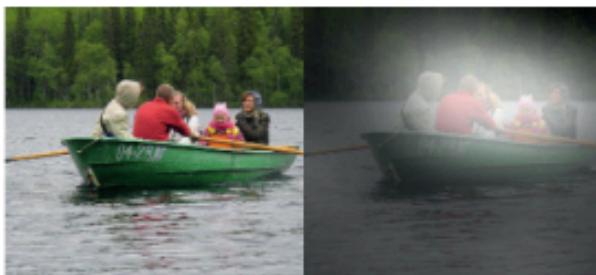
A **dog** is standing on a hardwood floor.



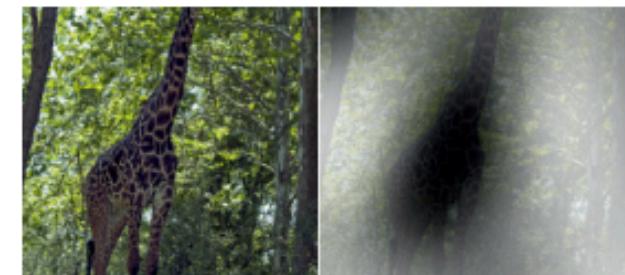
A **stop** sign is on a road with a mountain in the background



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

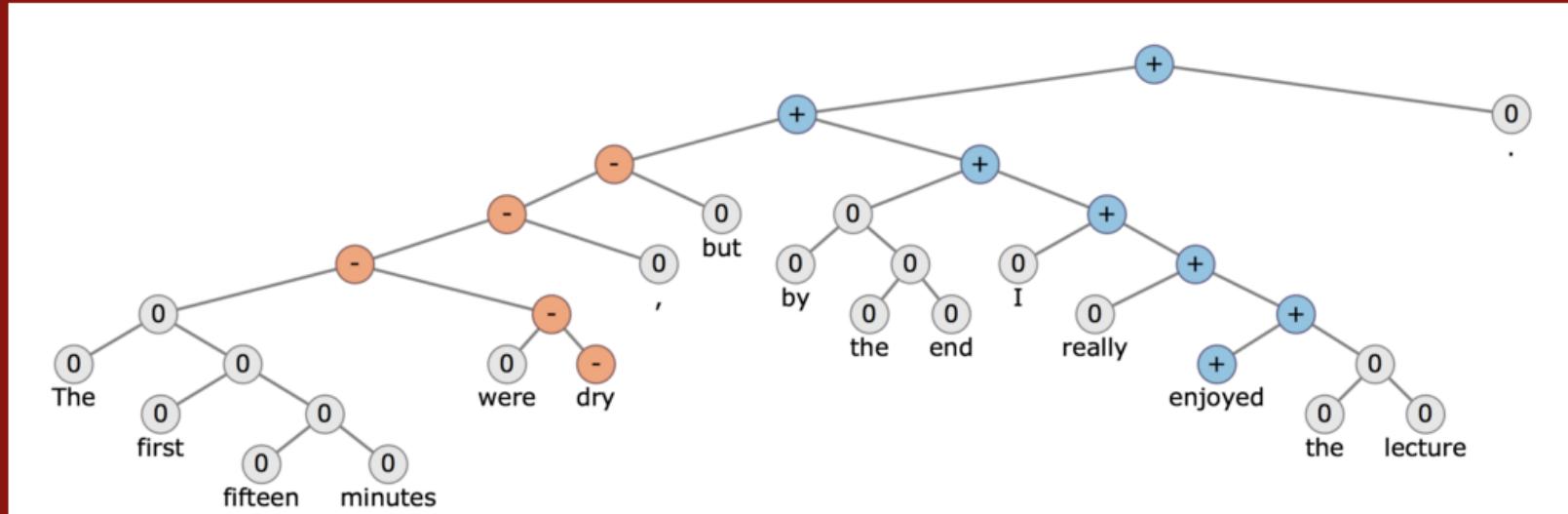
Image: deep convolution neural network (CNN)
Text: recurrent neural network (RNN)



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

CS224d: Deep Learning for Natural Language Processing

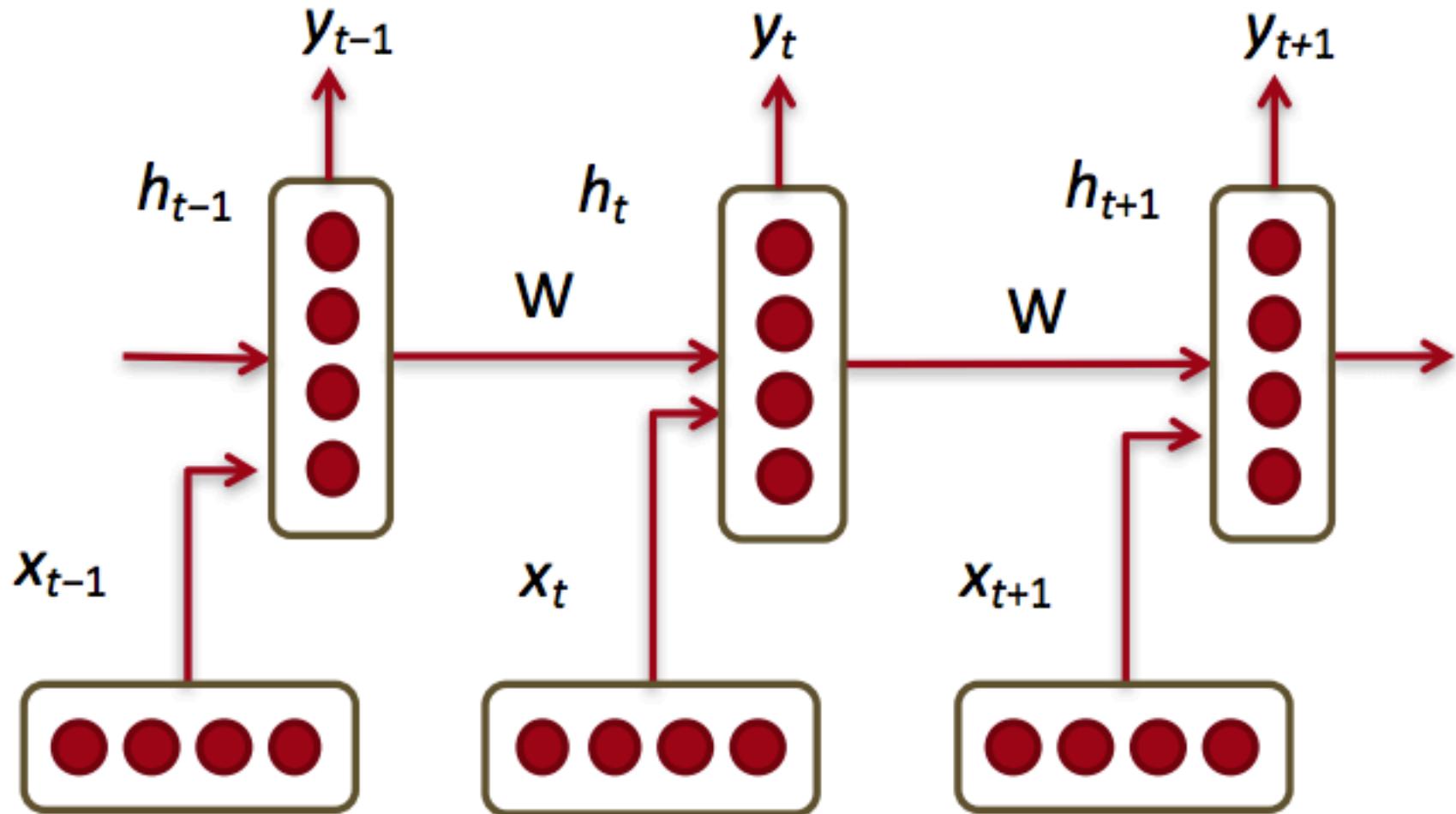
CS224d: Deep Learning for Natural Language Processing



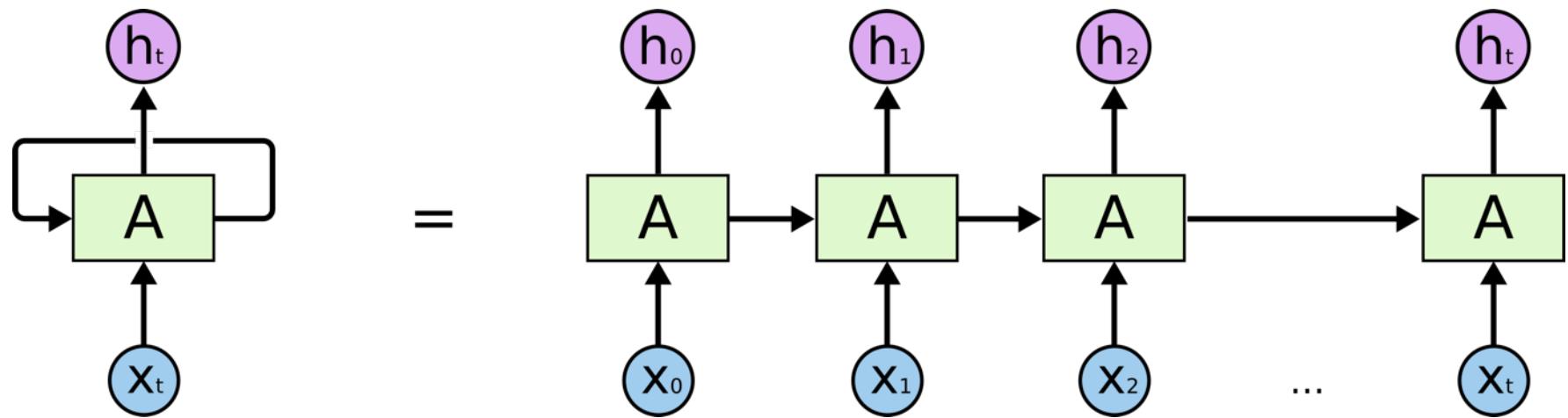
Course Description

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,

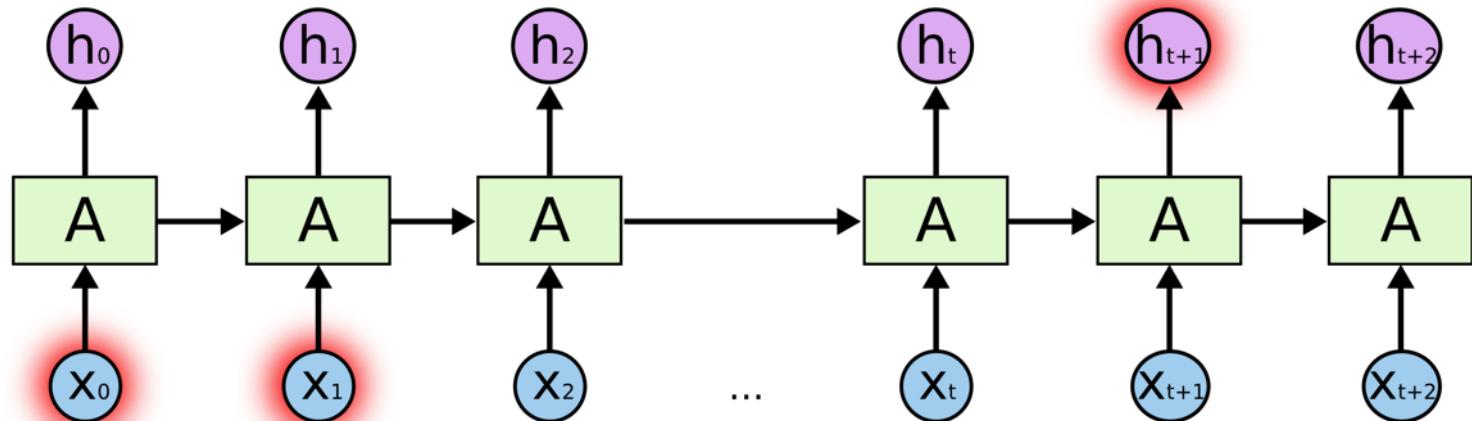
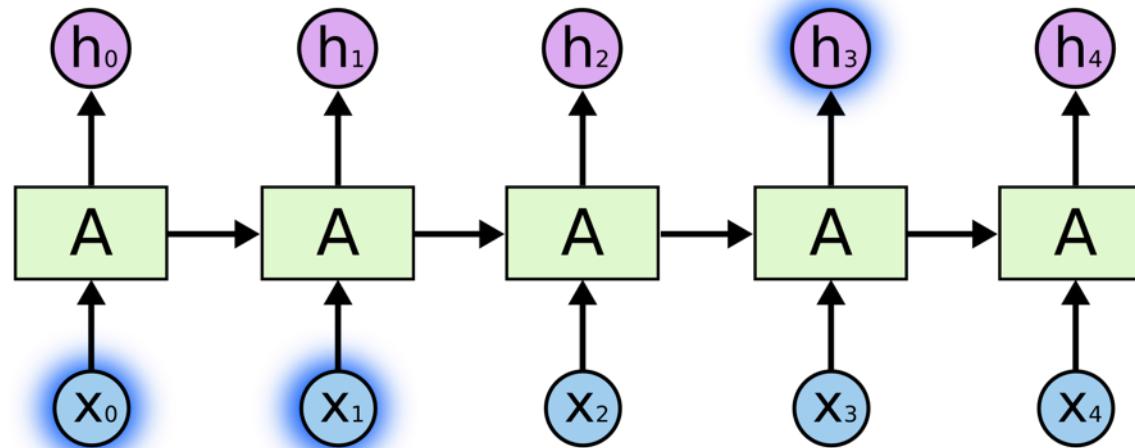
Recurrent Neural Networks (RNNs)



RNN

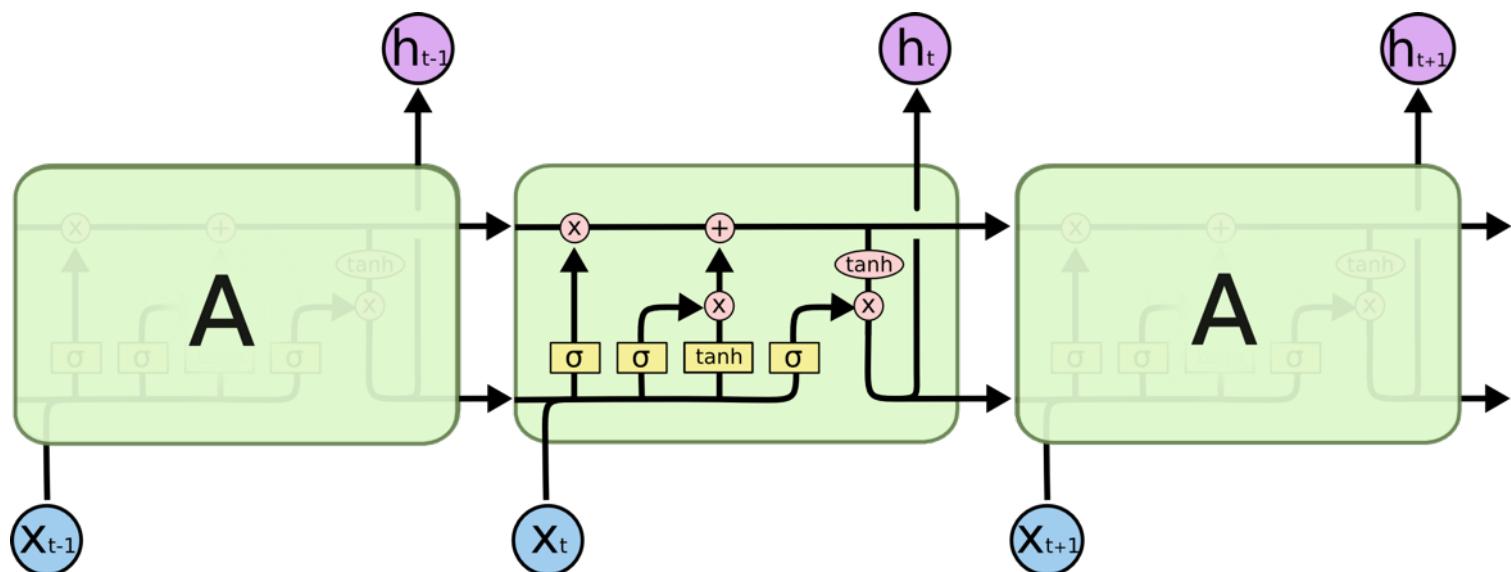
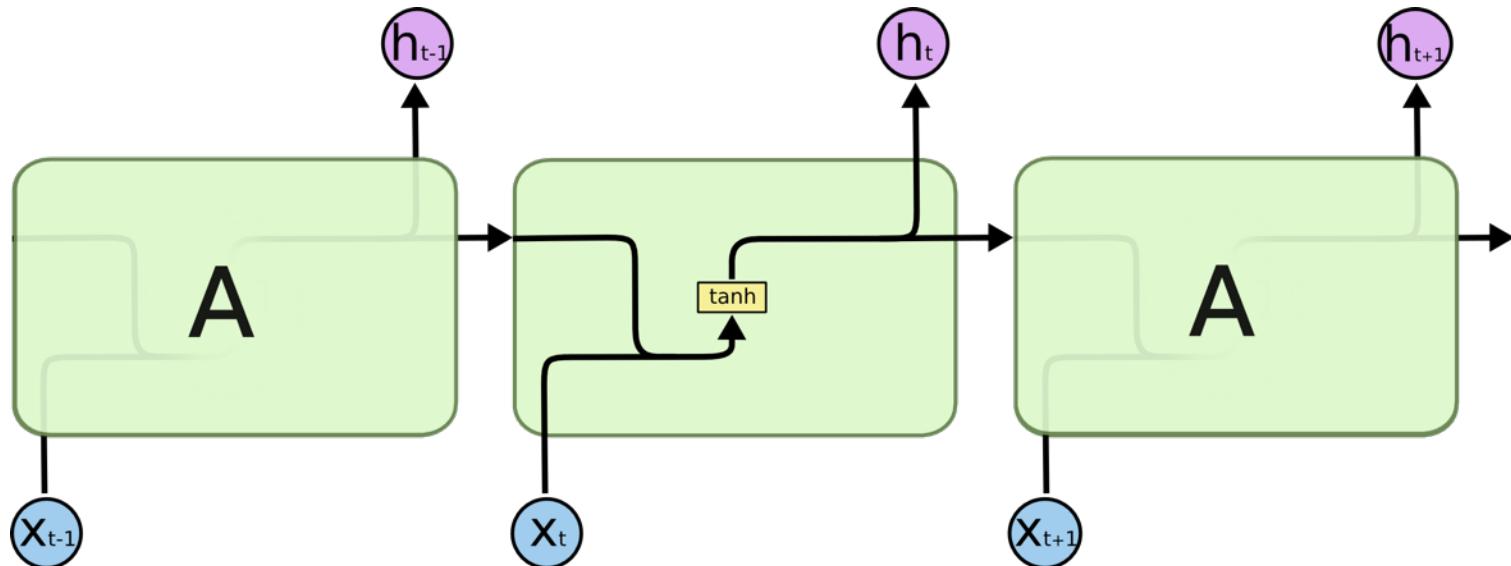


RNN long-term dependencies

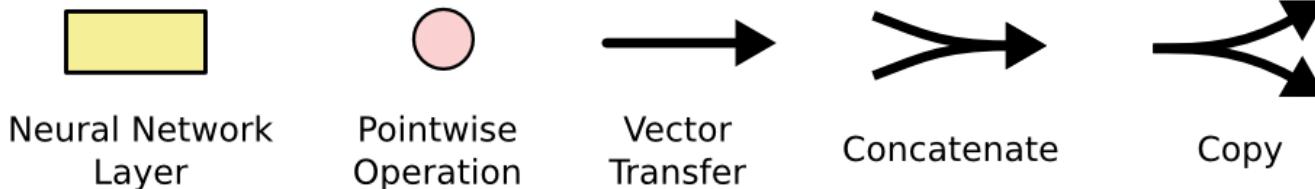
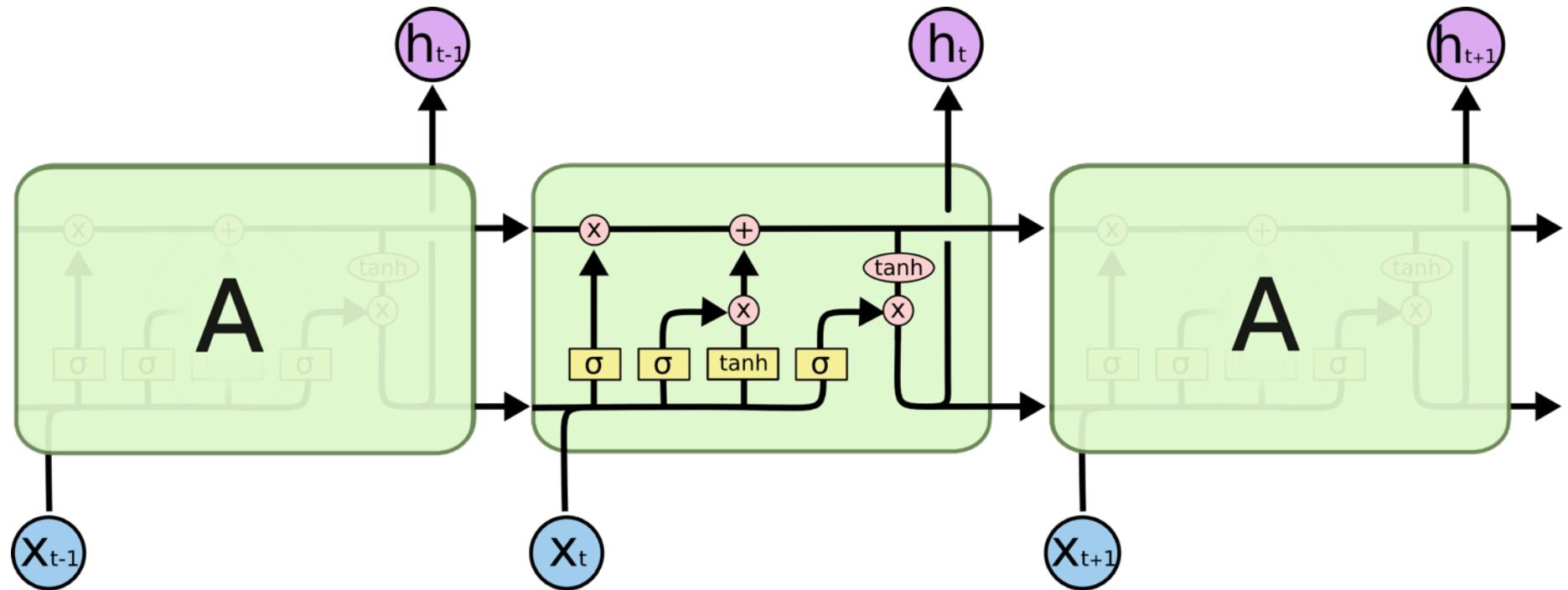


I grew up in France... I speak fluent French.

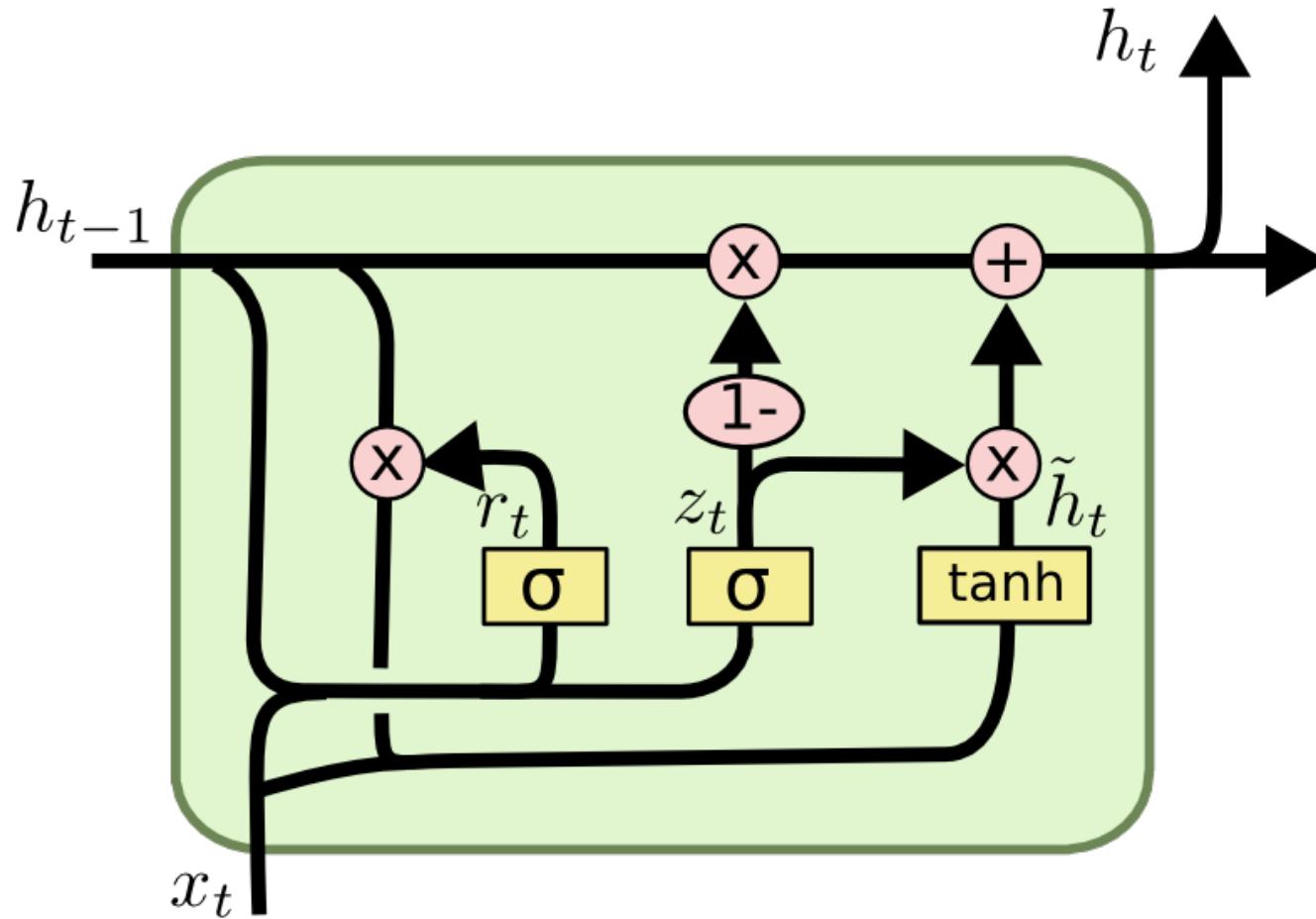
RNN LSTM



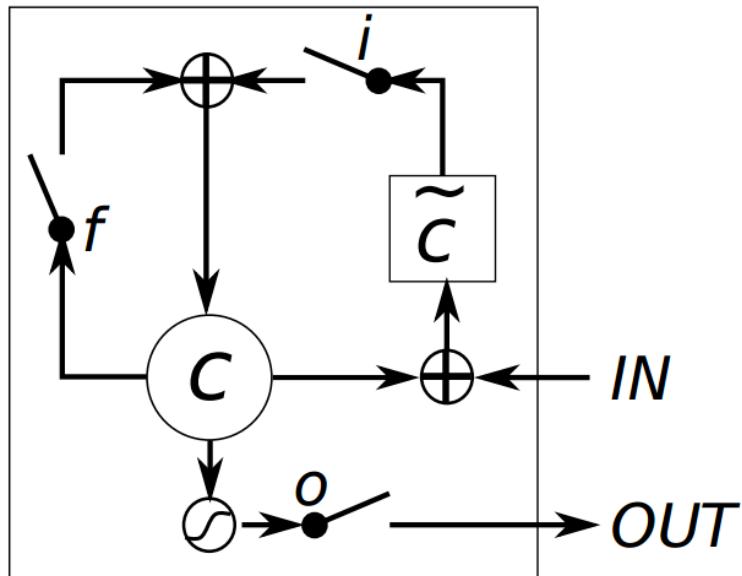
Long Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)



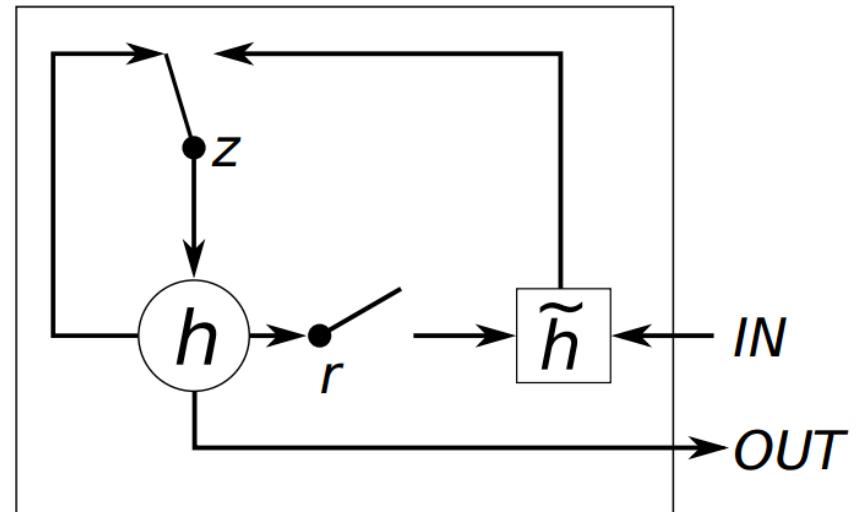
LSTM vs GRU



LSTM

i , f and o are the **input**, **forget** and **output** gates, respectively.

c and \tilde{c} denote the **memory cell** and the **new memory cell content**.

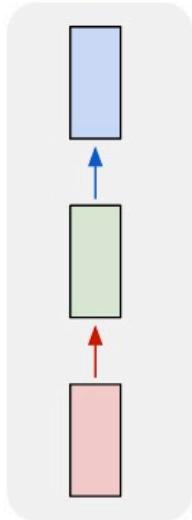


GRU

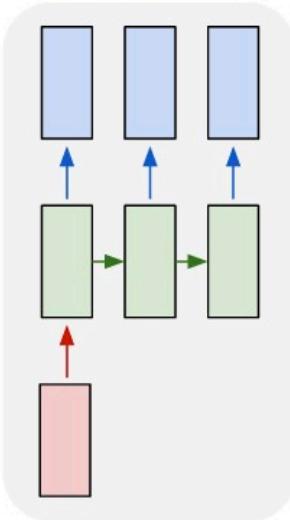
r and z are the **reset** and **update** gates, and h and \tilde{h} are the **activation** and the **candidate activation**.

LSTM Recurrent Neural Network

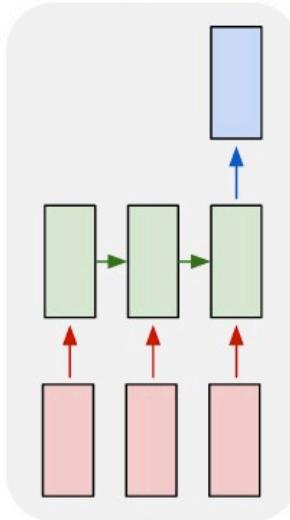
one to one



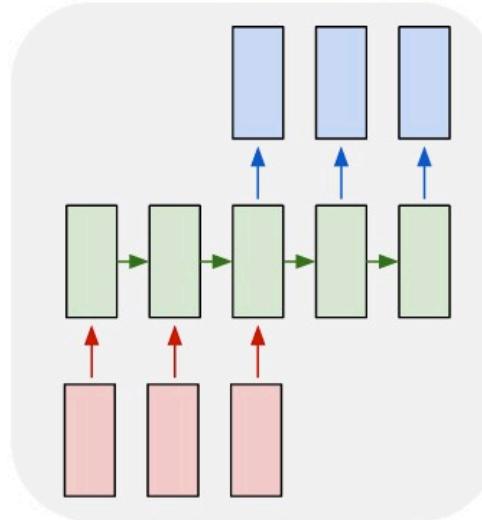
one to many



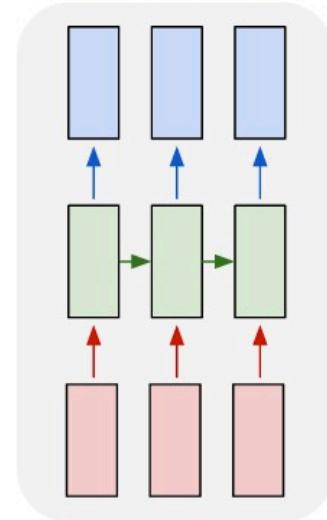
many to one



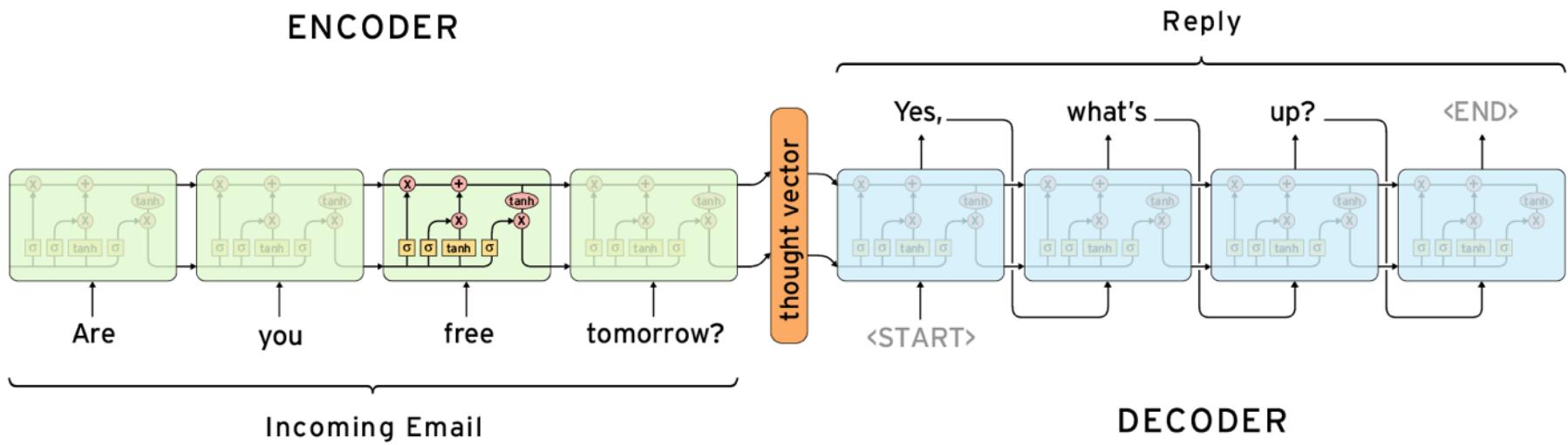
many to many



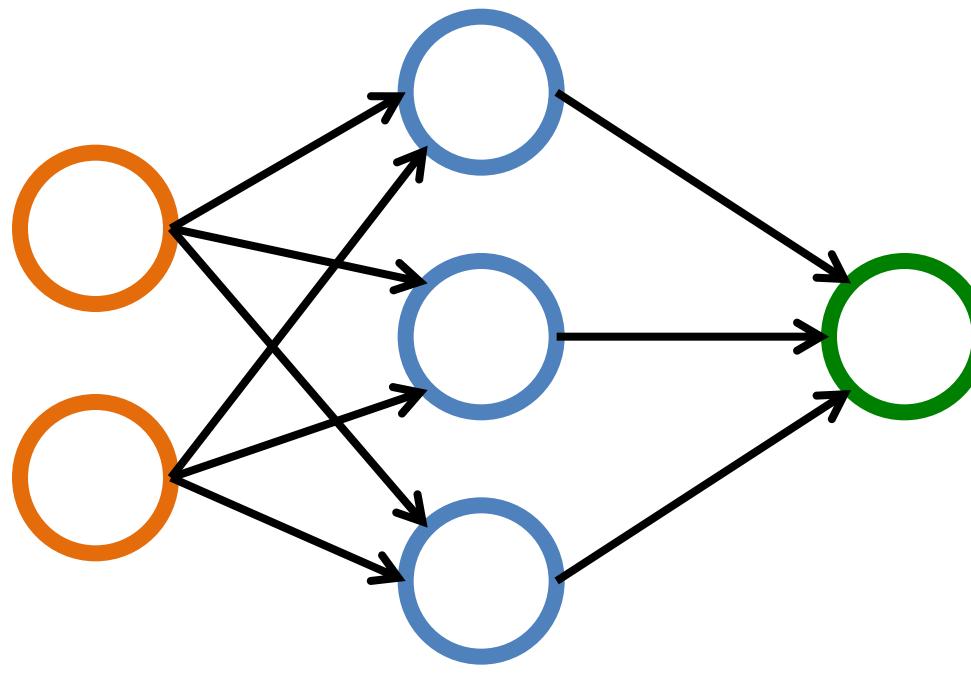
many to many



The Sequence to Sequence model (seq2seq)

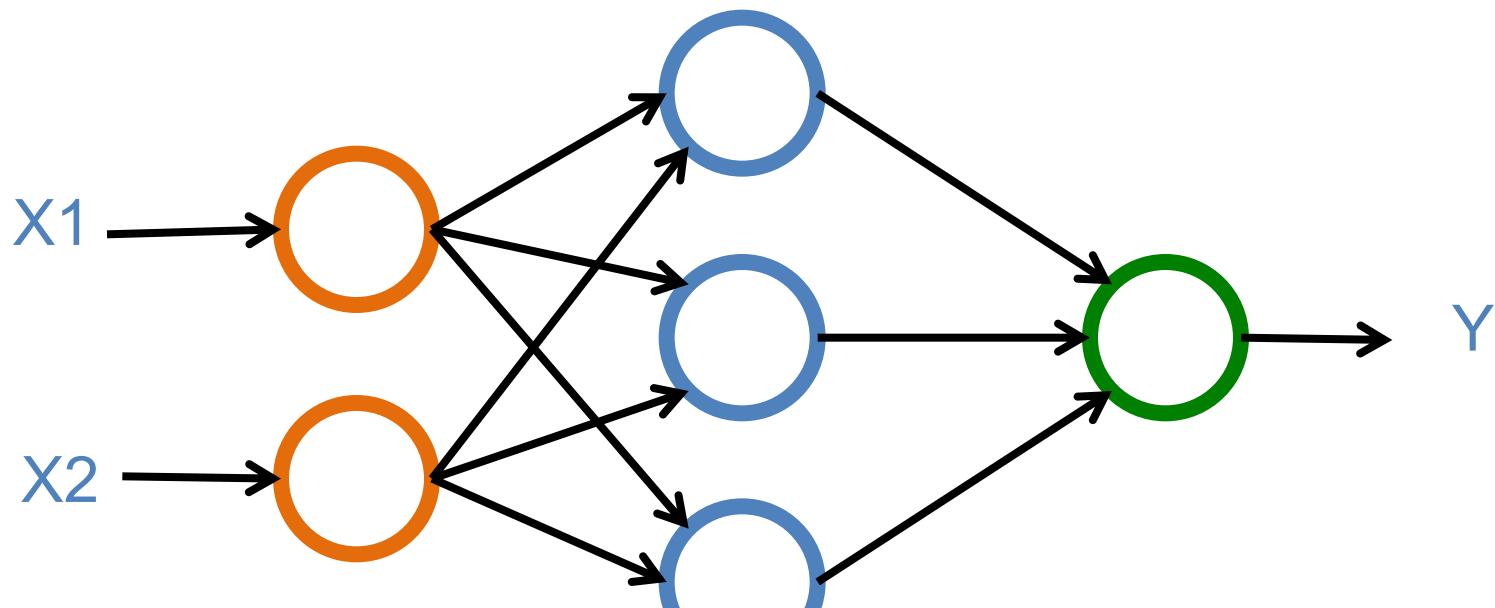


Neural Networks



Neural Networks

Input Layer **Hidden Layer** **Output Layer**
(X) **(H)** **(Y)**



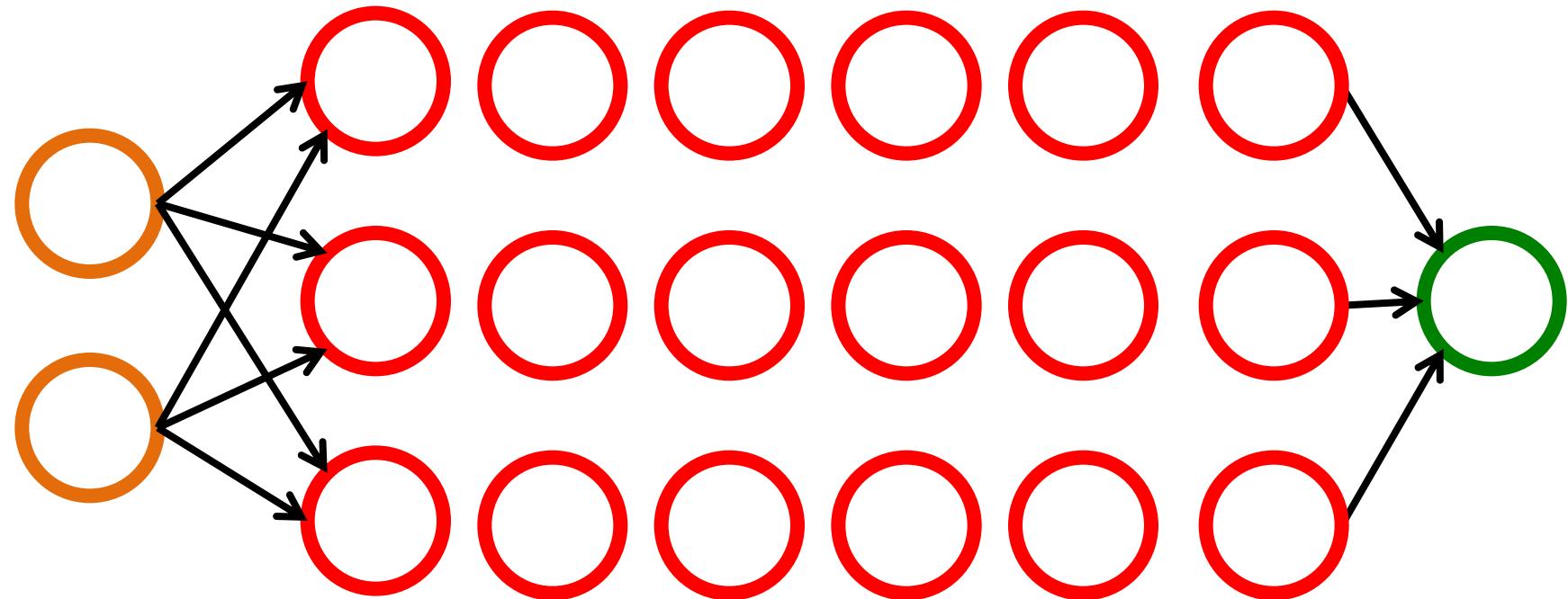
Neural Networks

Input Layer
(X)

Hidden Layers
(H)

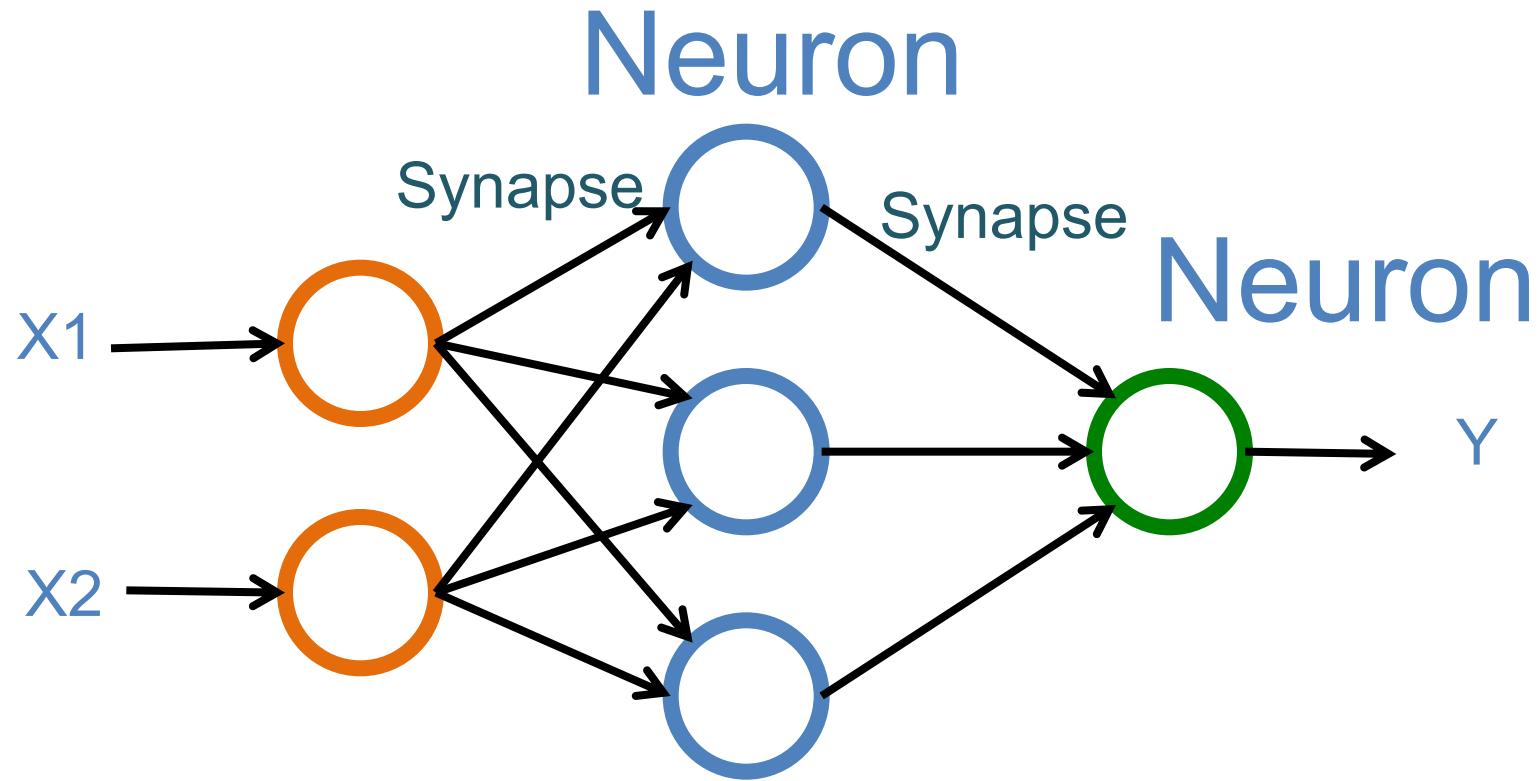
Output Layer
(Y)

Deep Neural Networks
Deep Learning

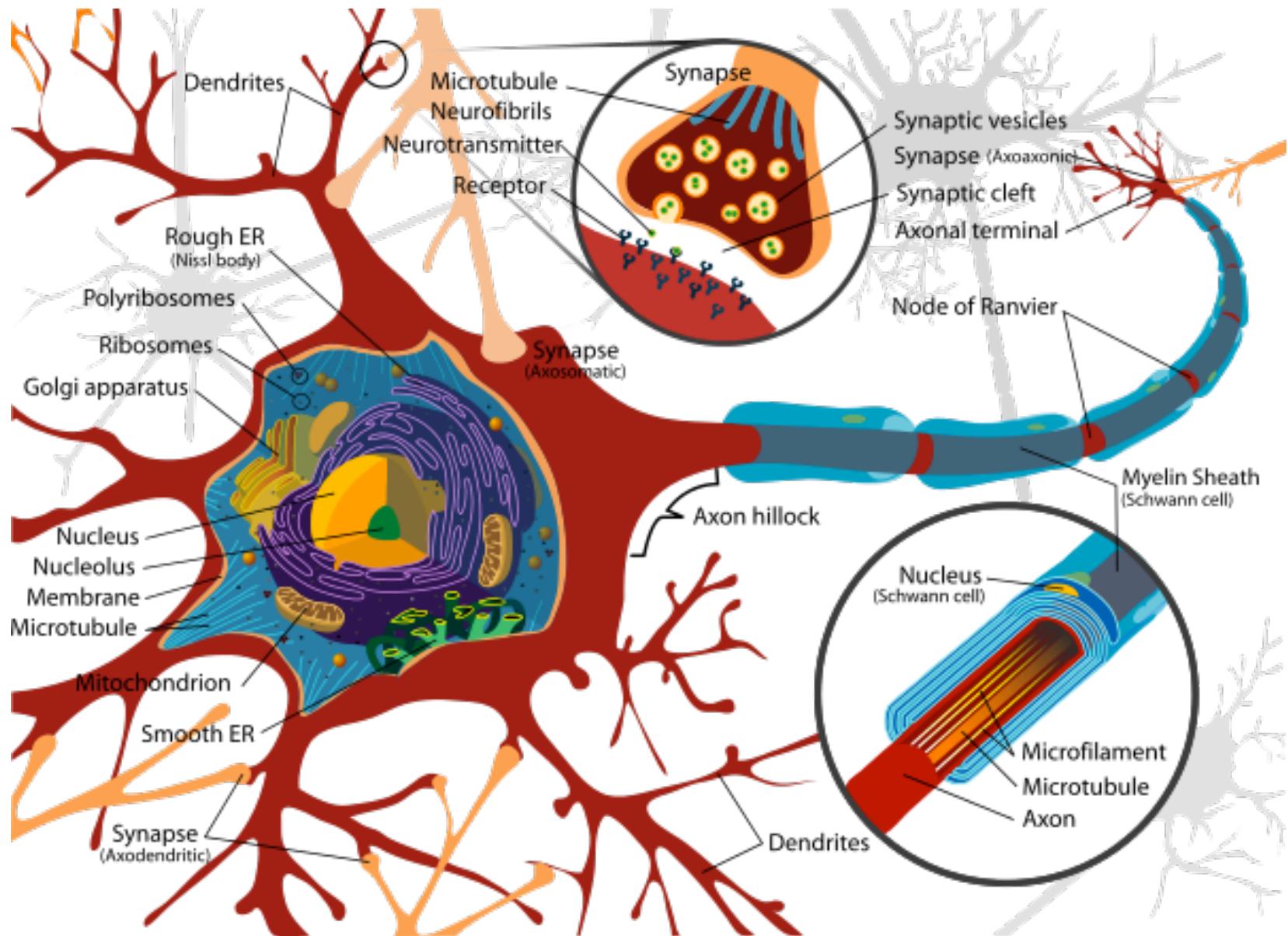


Neural Networks

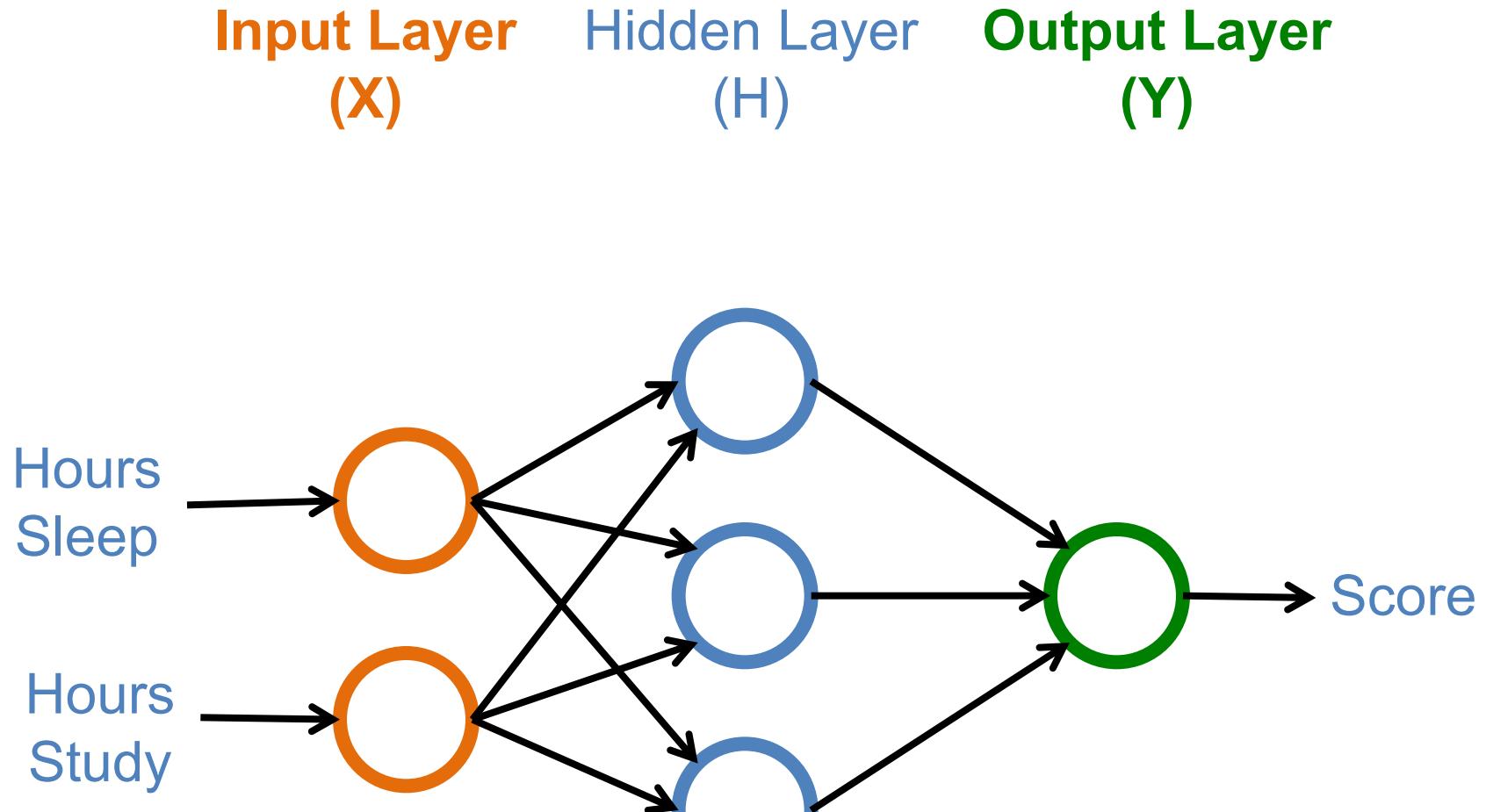
Input Layer **Hidden Layer** **Output Layer**
(X) **(H)** **(Y)**



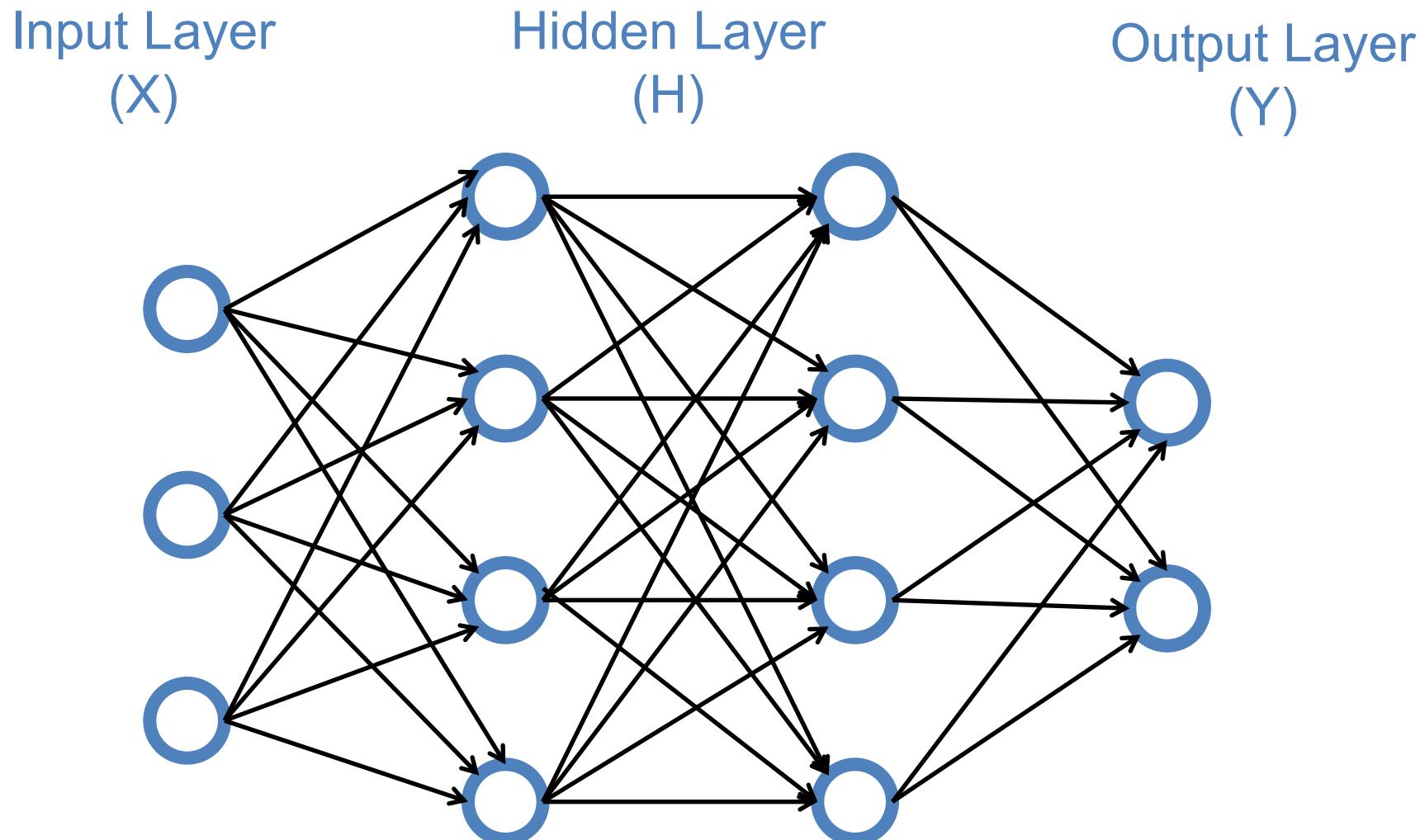
Neuron and Synapse



Neural Networks

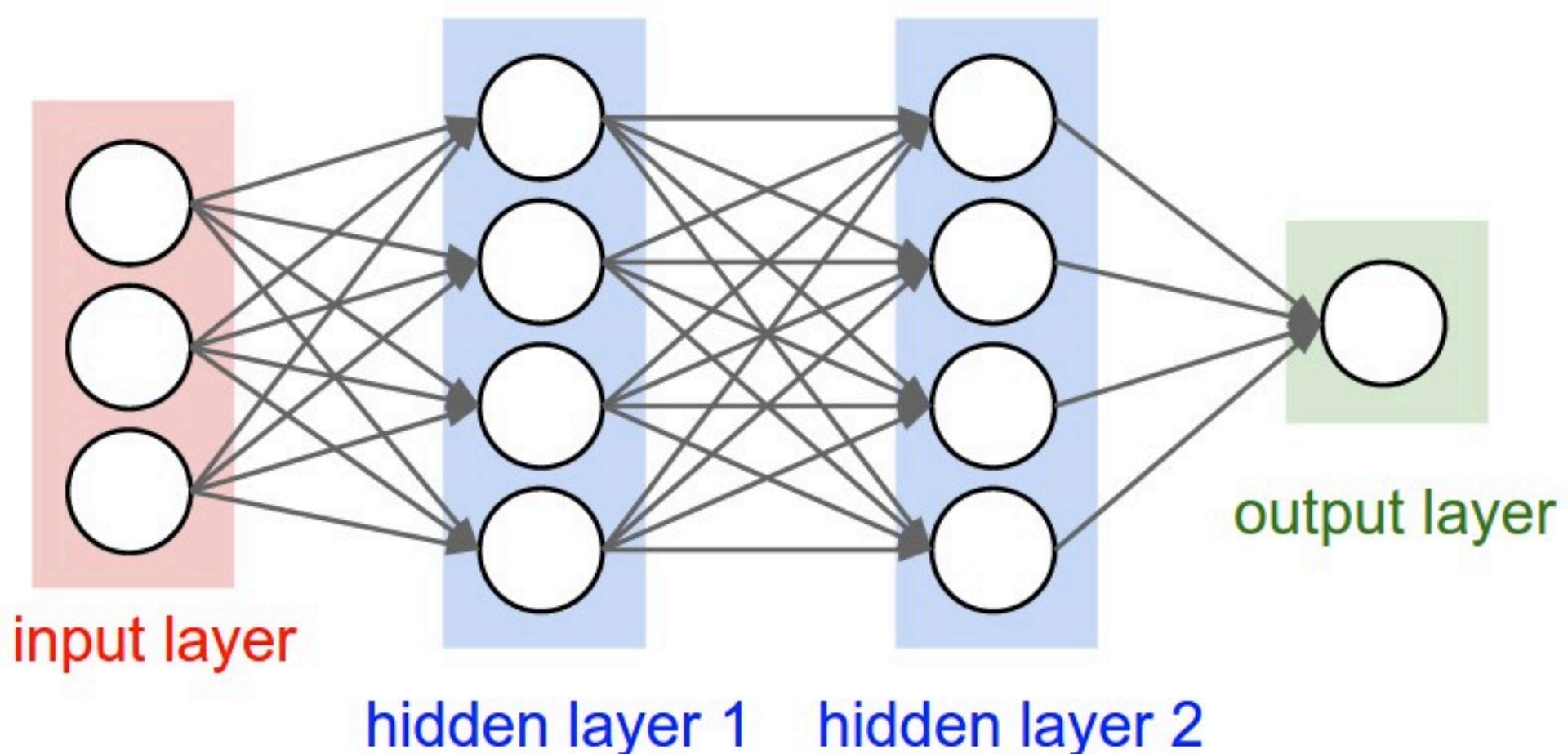


Neural Networks

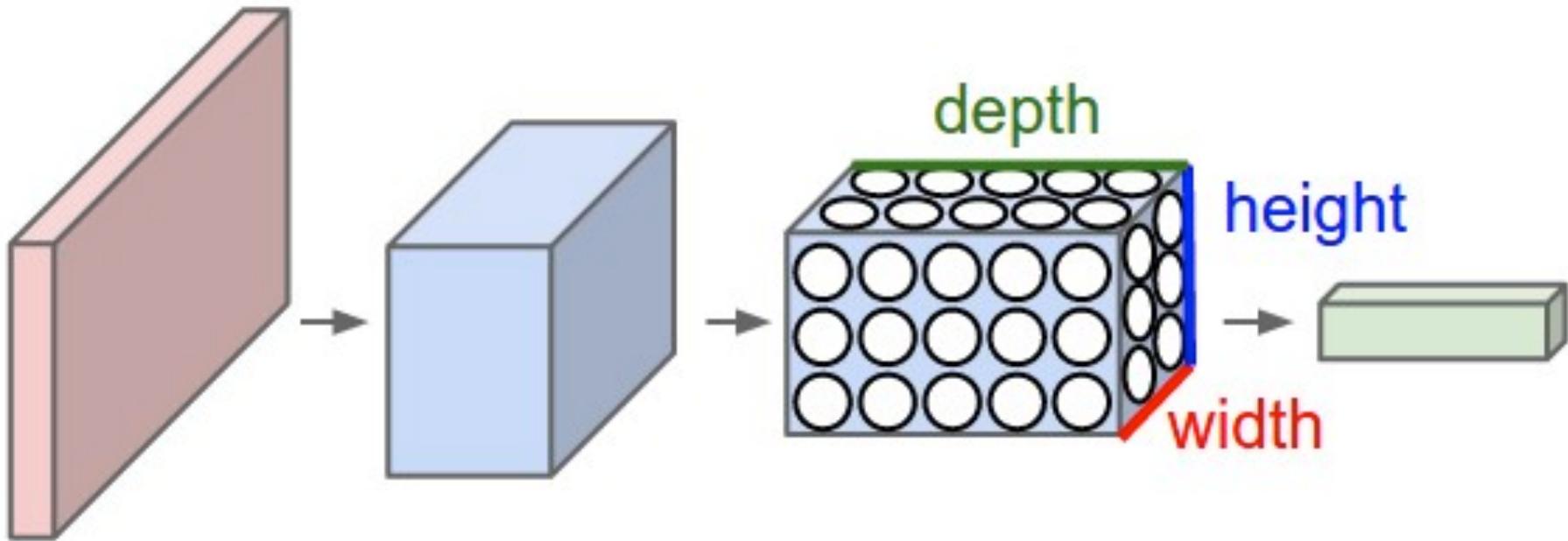


Convolutional Neural Networks (CNNs / ConvNets)

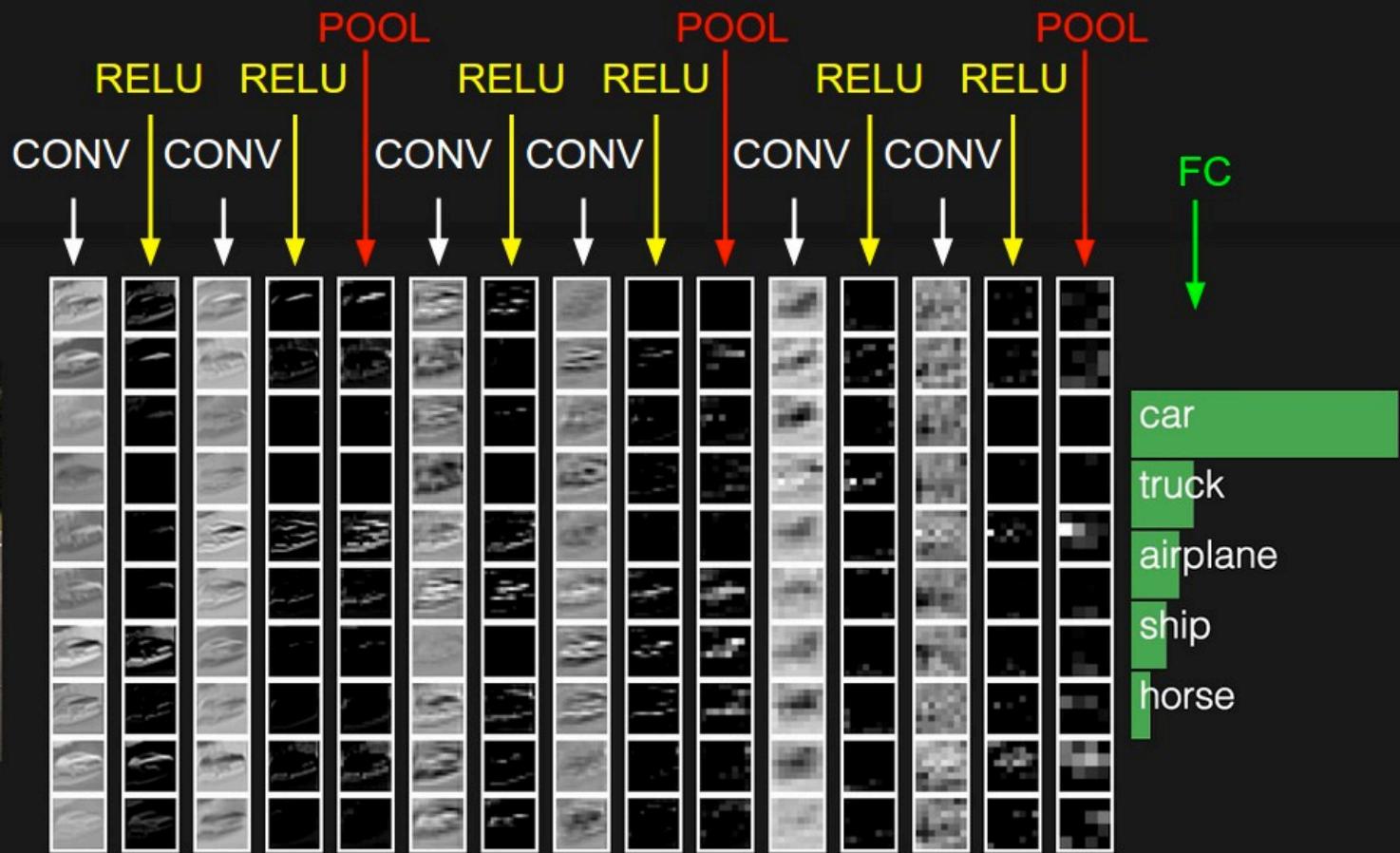
A regular 3-layer Neural Network



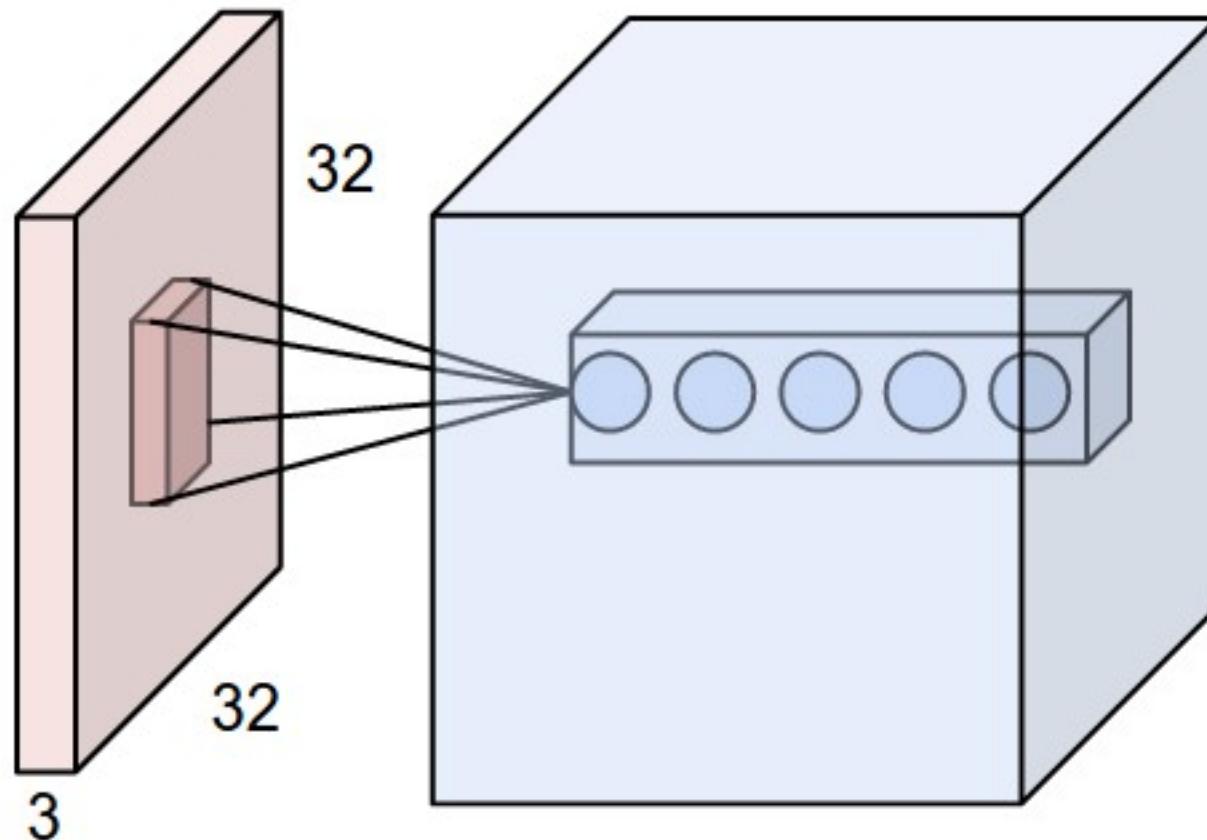
A ConvNet arranges its neurons in three dimensions (width, height, depth)



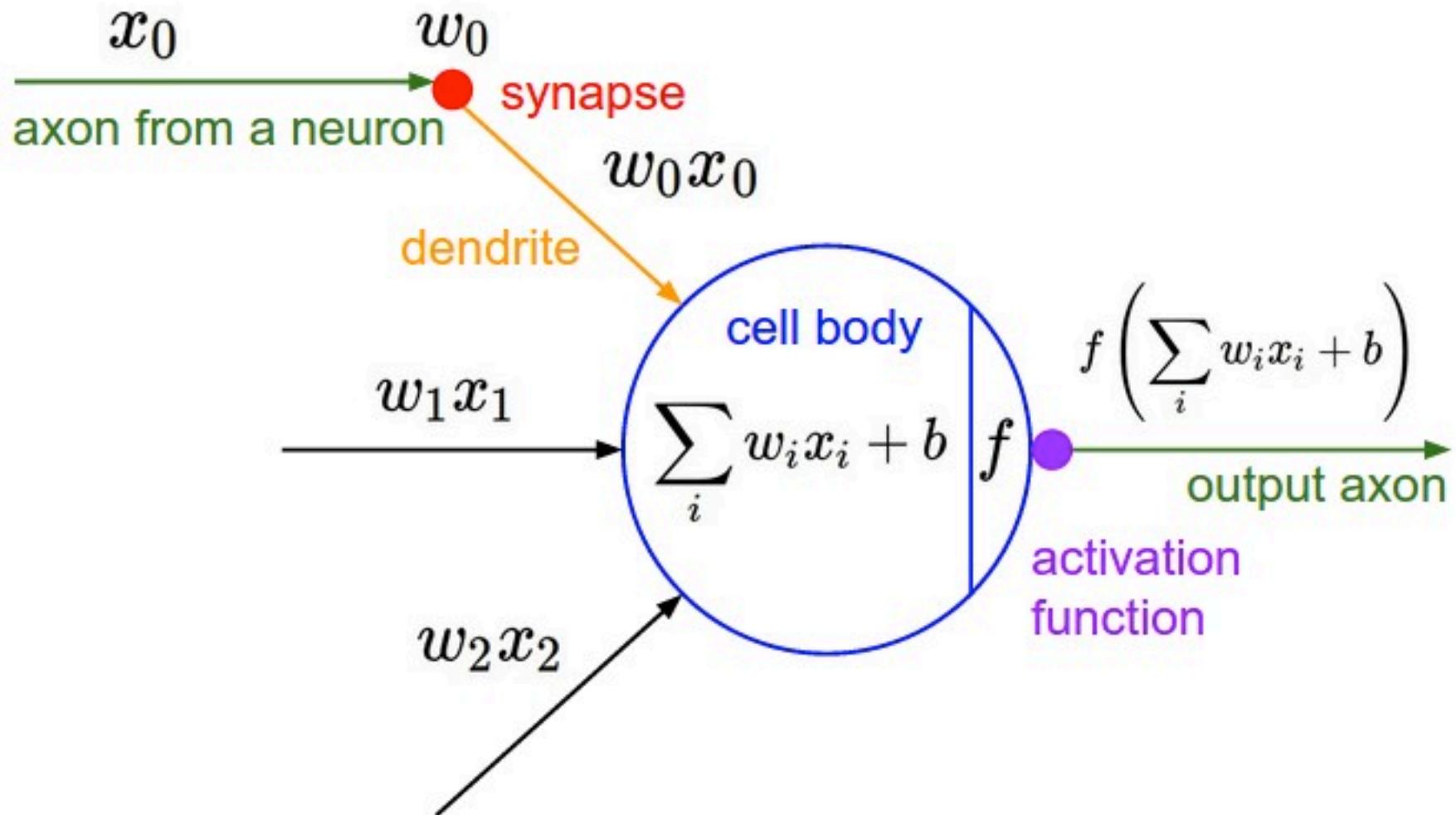
The activations of an example ConvNet architecture.



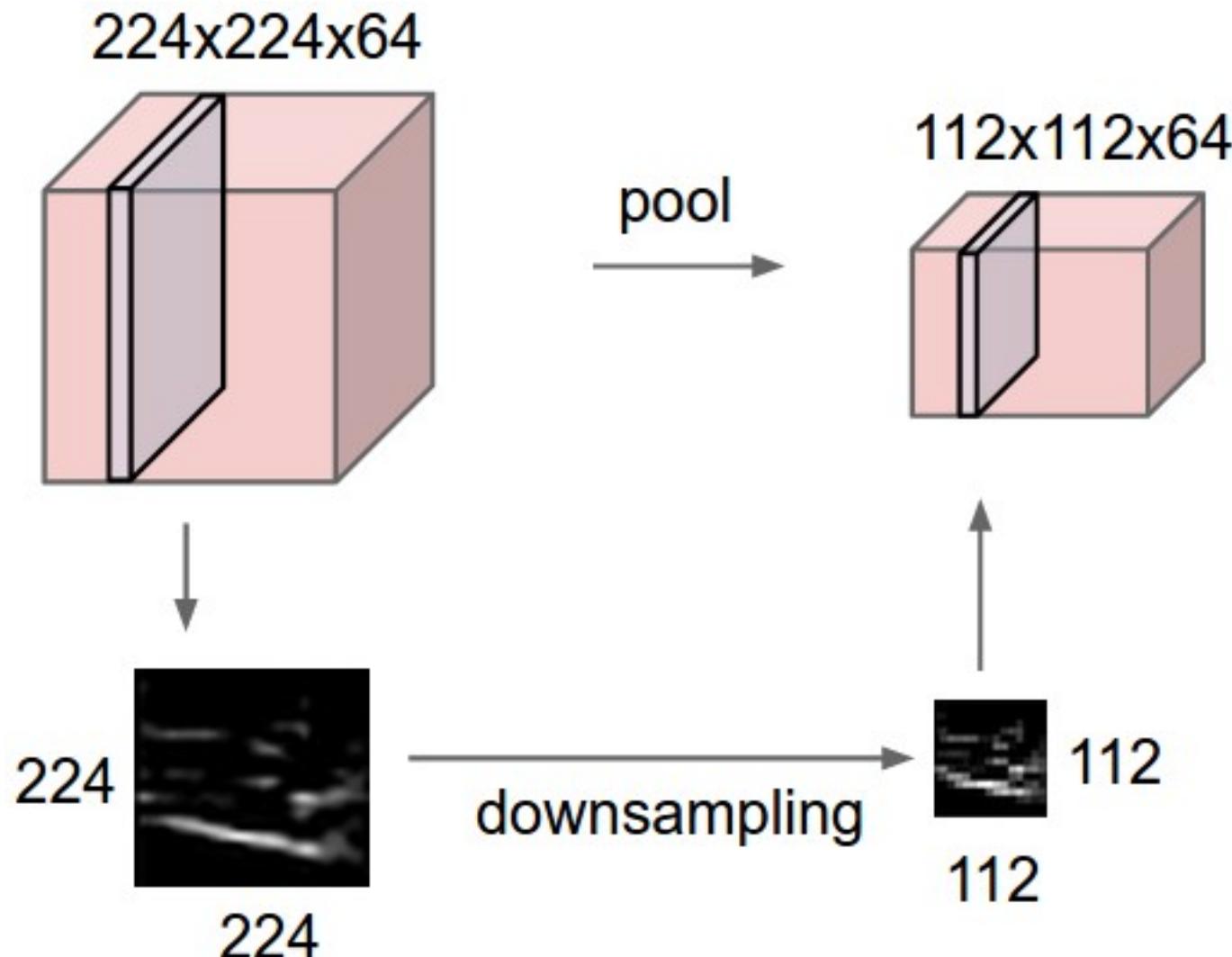
ConvNets



ConvNets



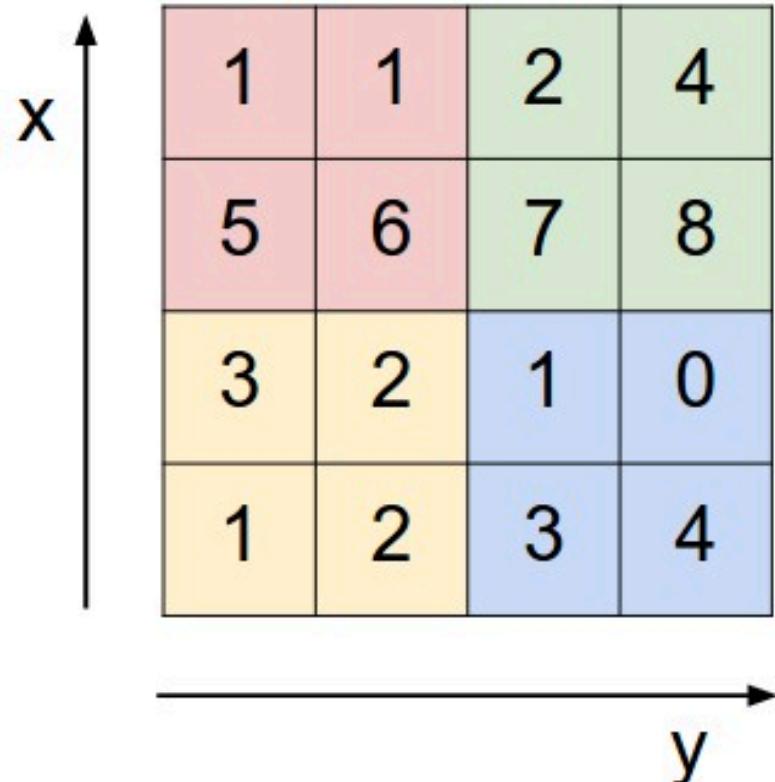
ConvNets



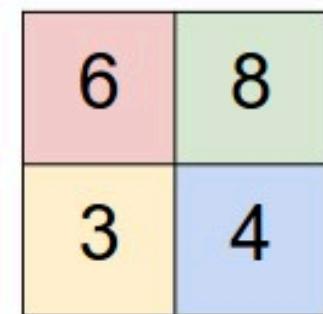
ConvNets

max pooling

Single depth slice



max pool with 2x2 filters
and stride 2

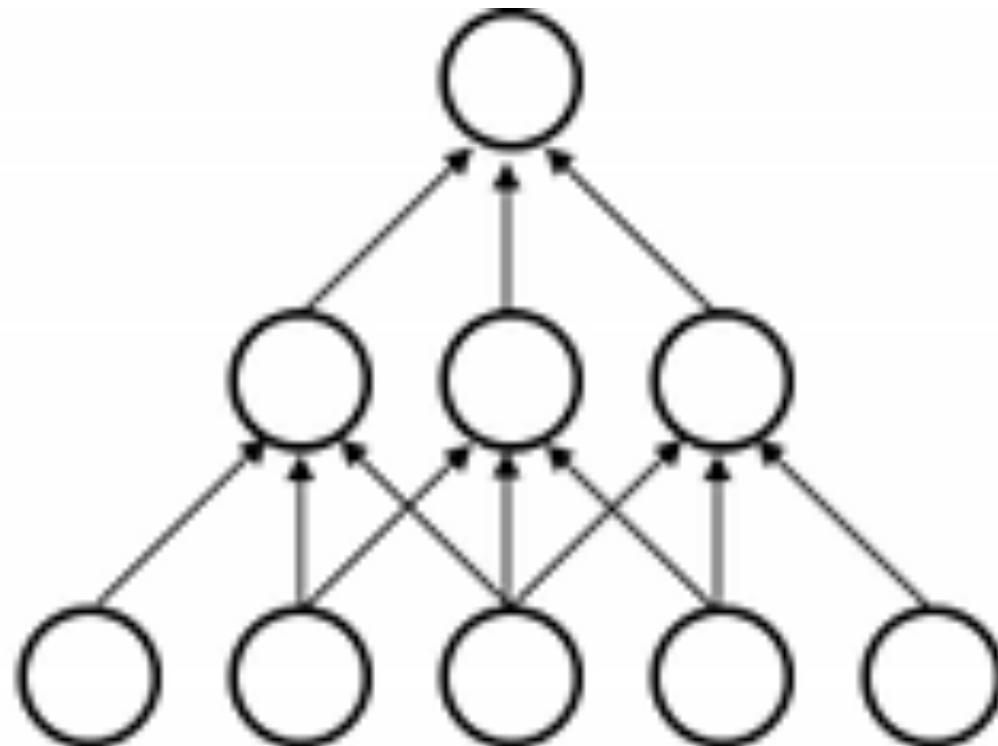


Convolutional Neural Networks (CNN) (LeNet) Sparse Connectivity

layer $m+1$

layer m

layer $m-1$

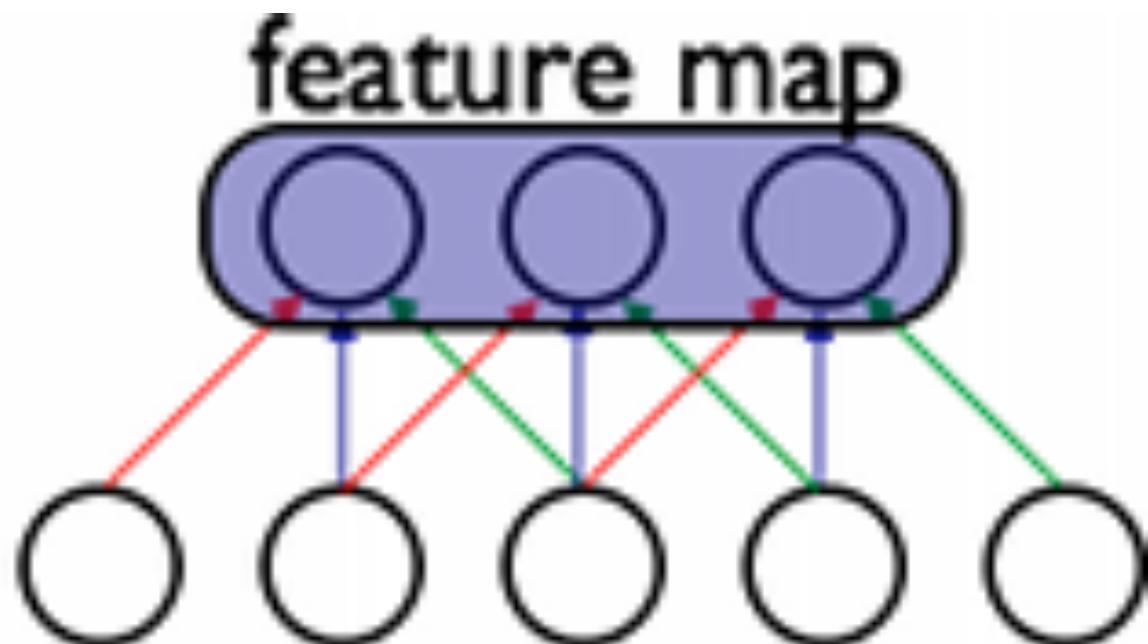


Convolutional Neural Networks (CNN) (LeNet)

Shared Weights

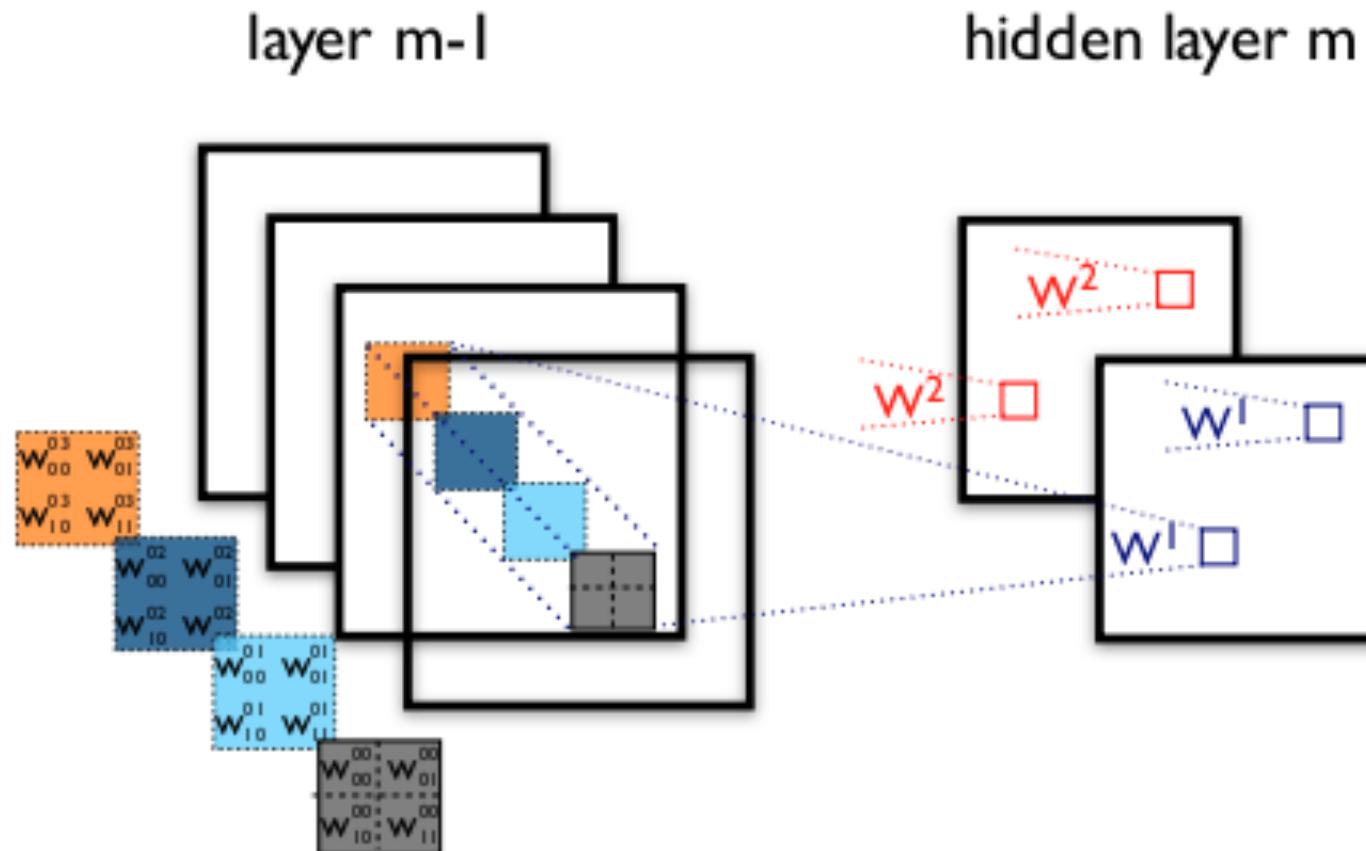
layer m

layer m-1

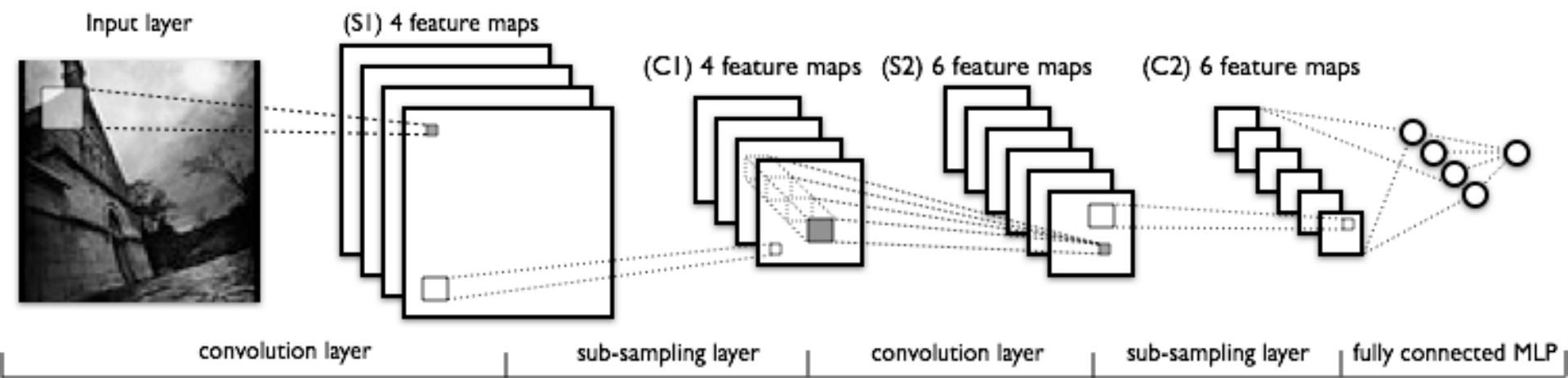


Convolutional Neural Networks (CNN) (LeNet)

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

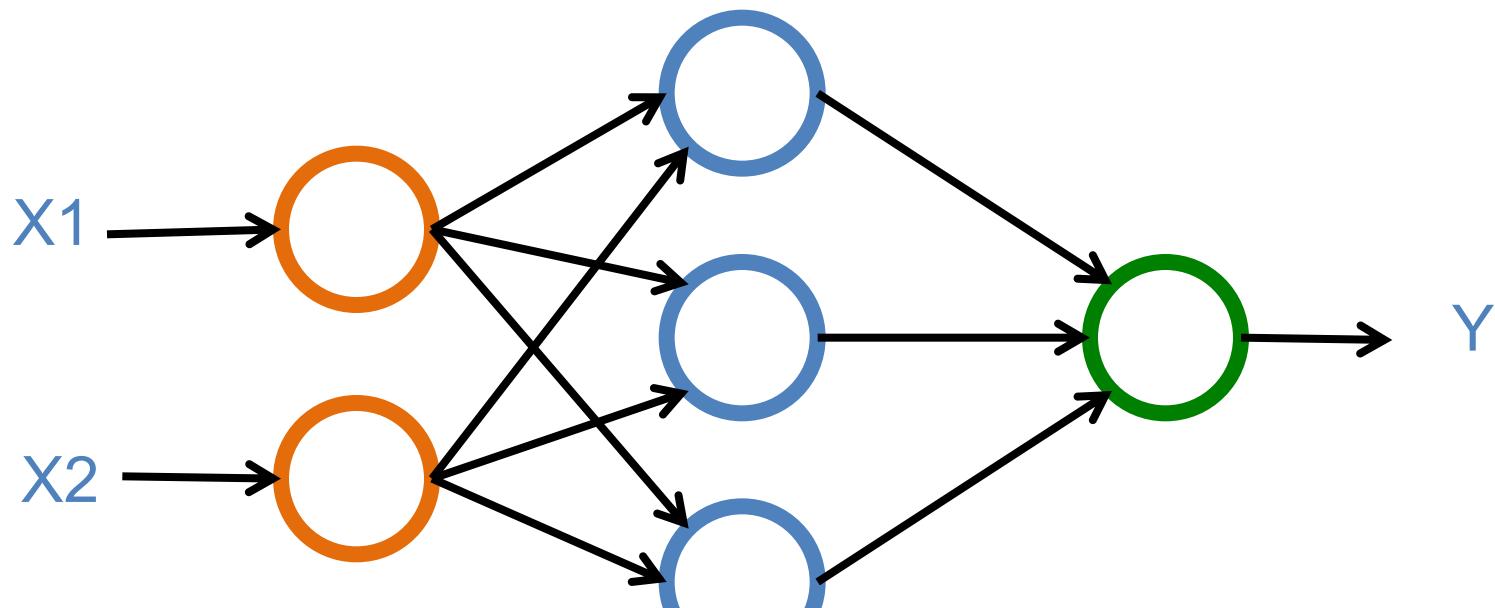
show flights from Boston to New York today

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

Neural Networks

Input Layer **Hidden Layer** **Output Layer**
(X) **(H)** **(Y)**



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

The diagram illustrates a scatter plot with three axes: X (orange), Y (green), and Z (blue). The X and Y axes are labeled "Hours Sleep" and "Hours Study" respectively, while the Z axis is labeled "Score". A vertical blue line separates the X and Y axes. A horizontal dashed blue line extends from the Z-axis label "Score" through the middle of the plot area.

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

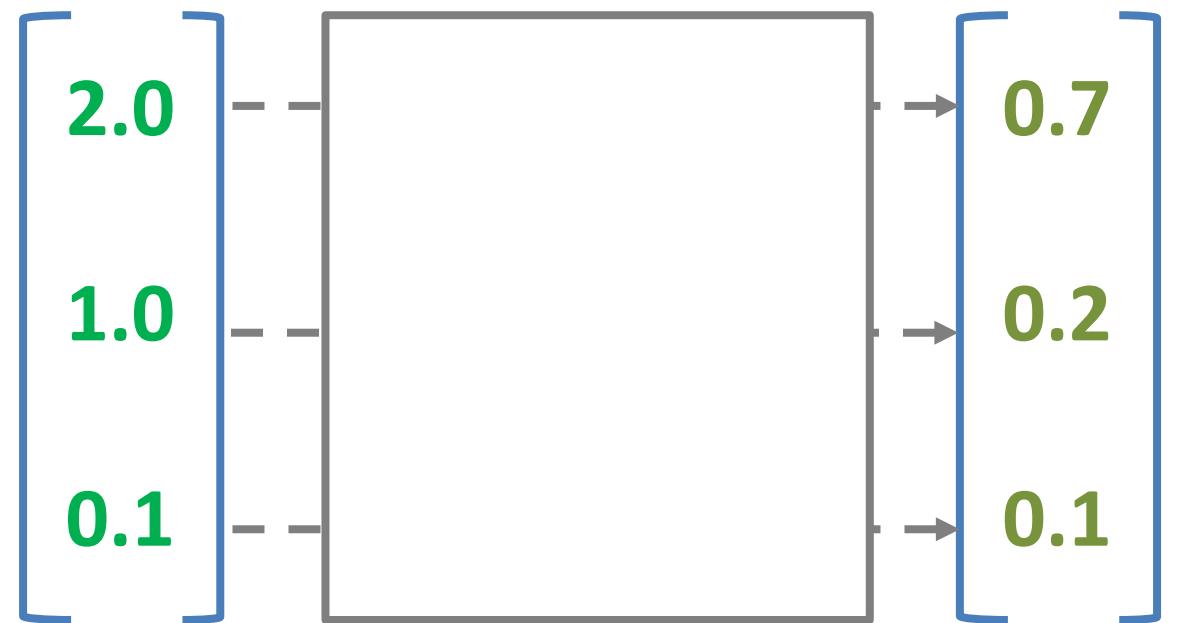
$$Y = wX + b$$

$$Y = W X + b$$

Diagram illustrating the components of a linear equation:

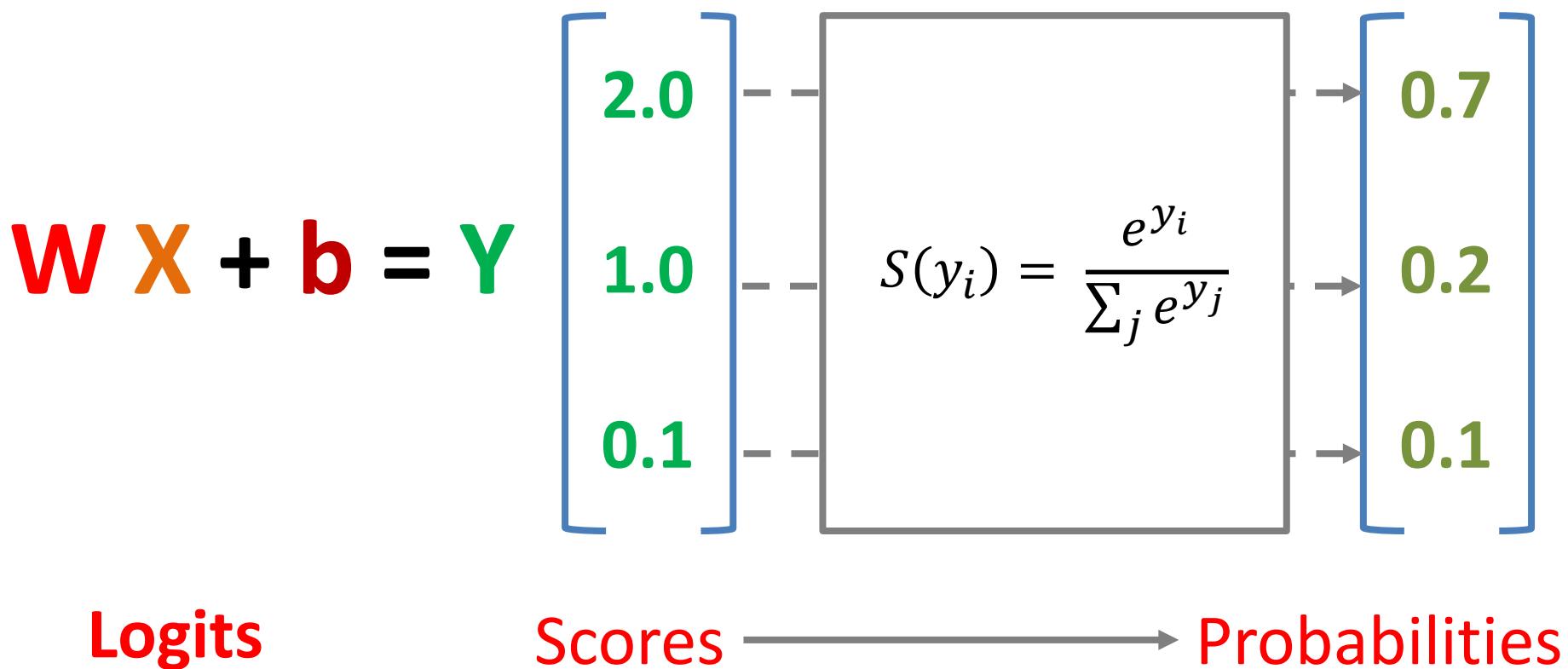
- Output** (green text) points to the term **Y**.
- input** (orange text) points to the term **X**.
- Weights** (red text) points to the term **W**.
- bias** (red text) points to the term **b**.
- Trained** (red text) points to the terms **W** and **b**.

$$W X + b = Y$$



Scores → Probabilities

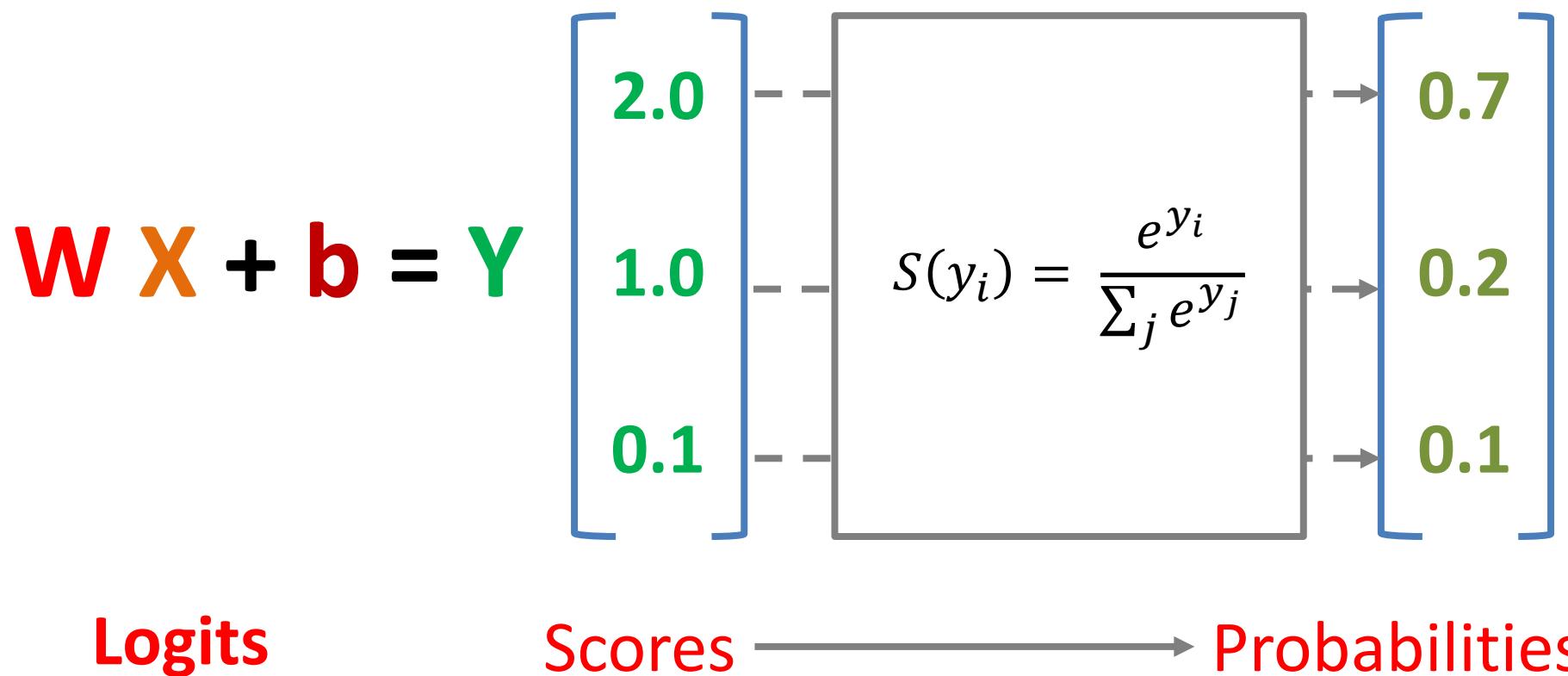
SoftMAX



$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{2.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.7$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{1.0}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{1.0}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.2$$

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{0.1}}{e^{2.0} + e^{1.0} + e^{0.1}} = \frac{2.7182^{0.1}}{2.7182^{2.0} + 2.7182^{1.0} + 2.7182^{0.1}} = 0.1$$



Training a Network

=

Minimize the Cost Function

Training a Network

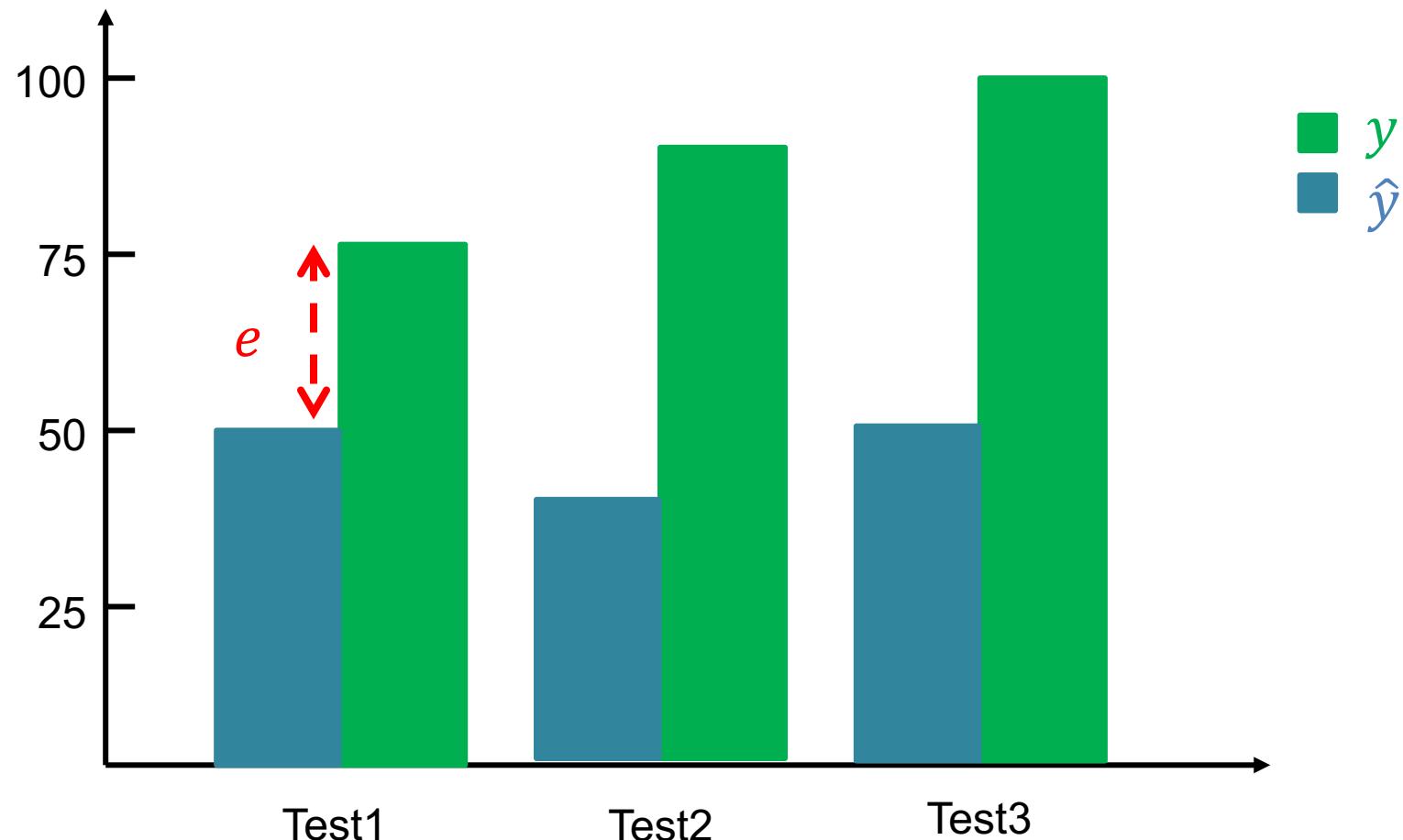
=

Minimize the **Cost Function**

Minimize the **Loss Function**

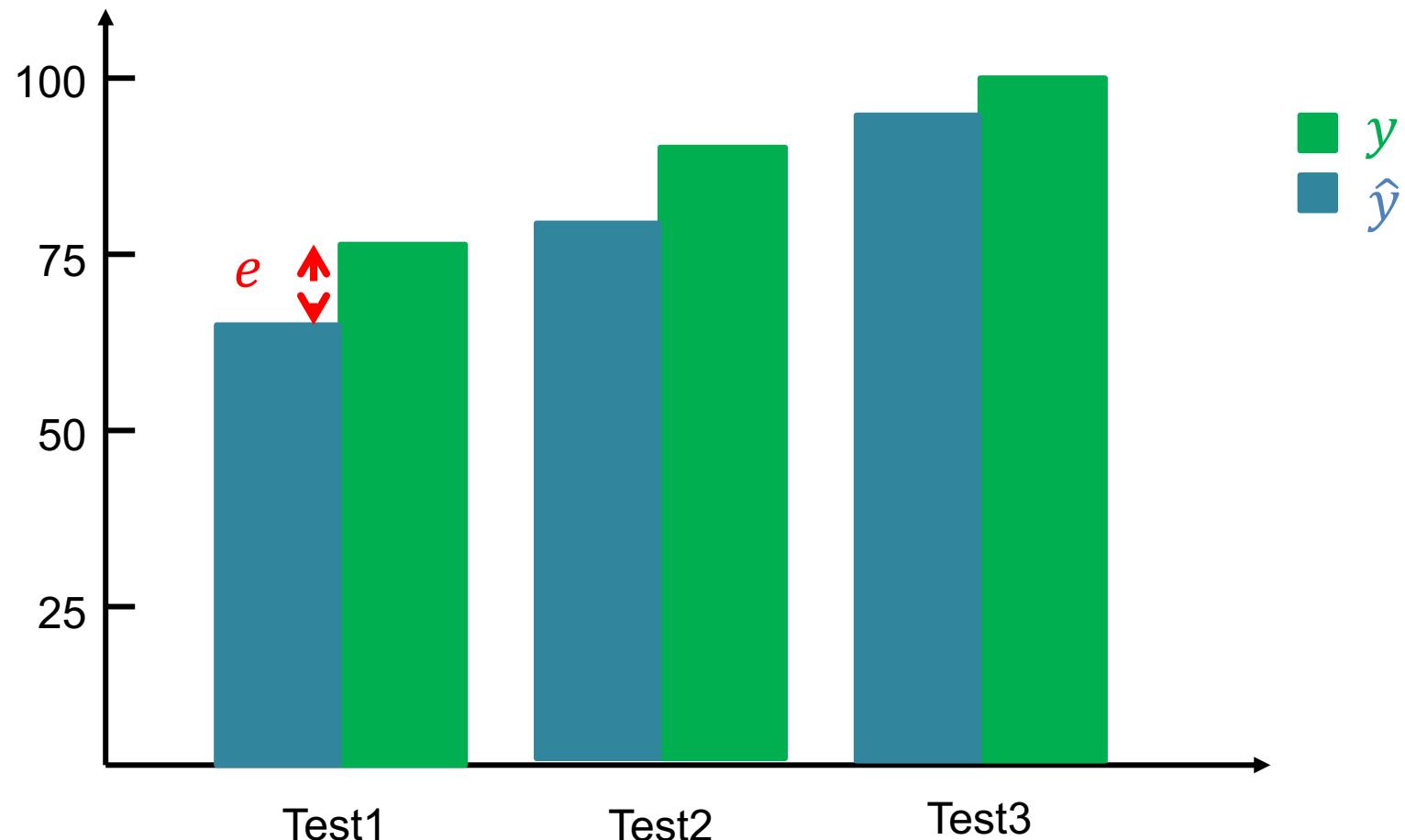
Error = Predict Y - Actual Y

Error : Cost : Loss



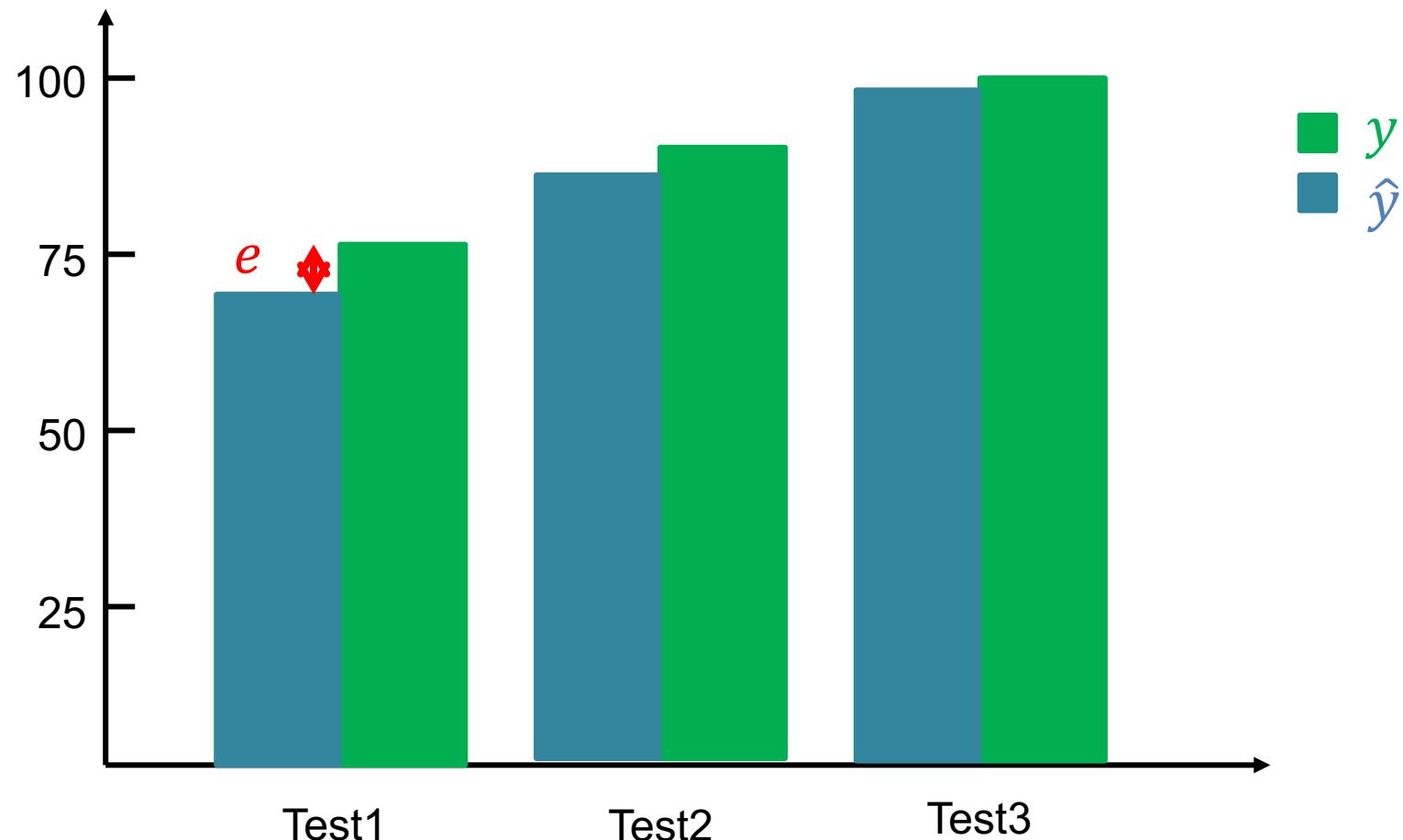
Error = Predict Y - Actual Y

Error : Cost : Loss



Error = Predict Y - Actual Y

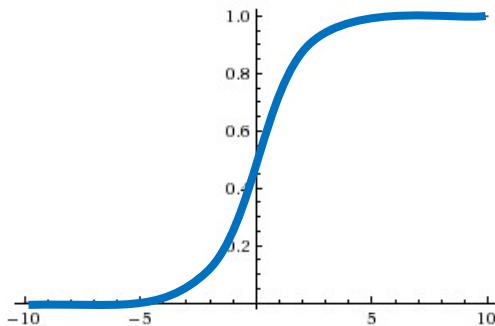
Error : Cost : Loss



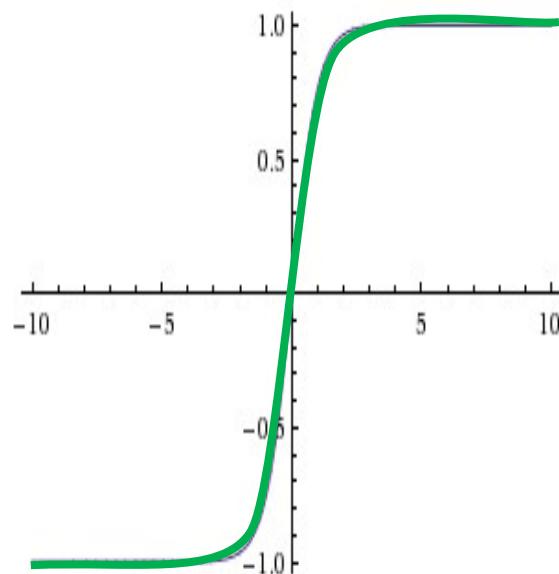
Activation Functions

Activation Functions

Sigmoid

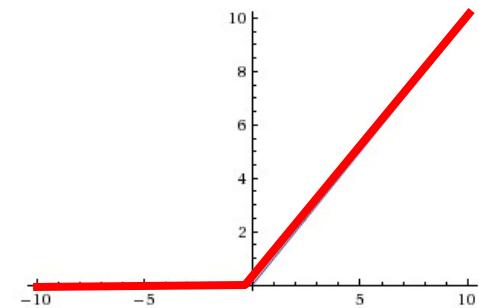


TanH



ReLU

(Rectified Linear Unit)



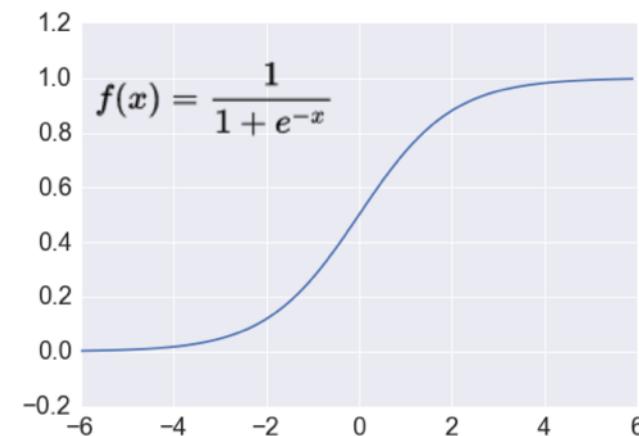
[0, 1]

[-1, 1]

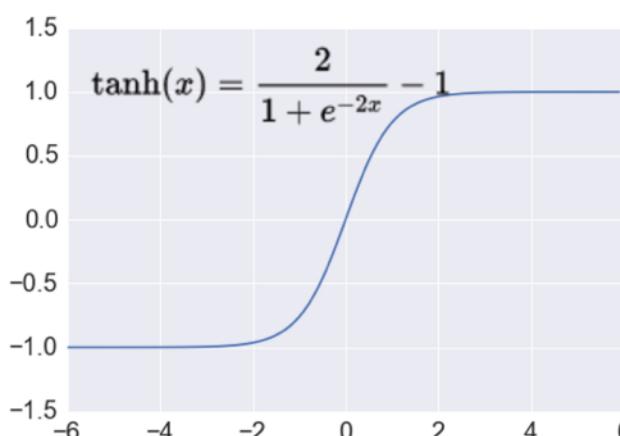
$$f(x) = \max(0, x)$$

Activation Functions

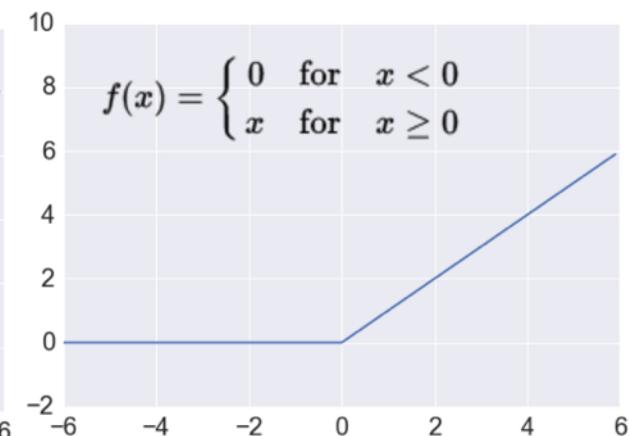
Sigmoid



TanH



ReLU



Loss Function

Binary Classification: 2 Class

Activation Function:
Sigmoid

Loss Function:
Binary Cross-Entropy

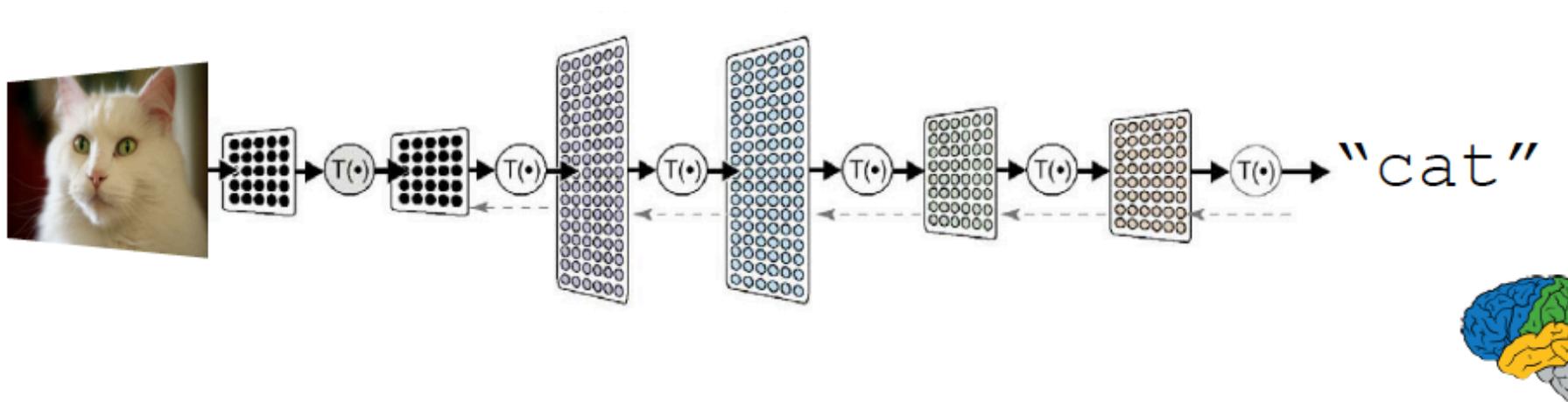
Multiple Classification: 10 Class

Activation Function:
SoftMAX

Loss Function:
Categorical Cross-Entropy

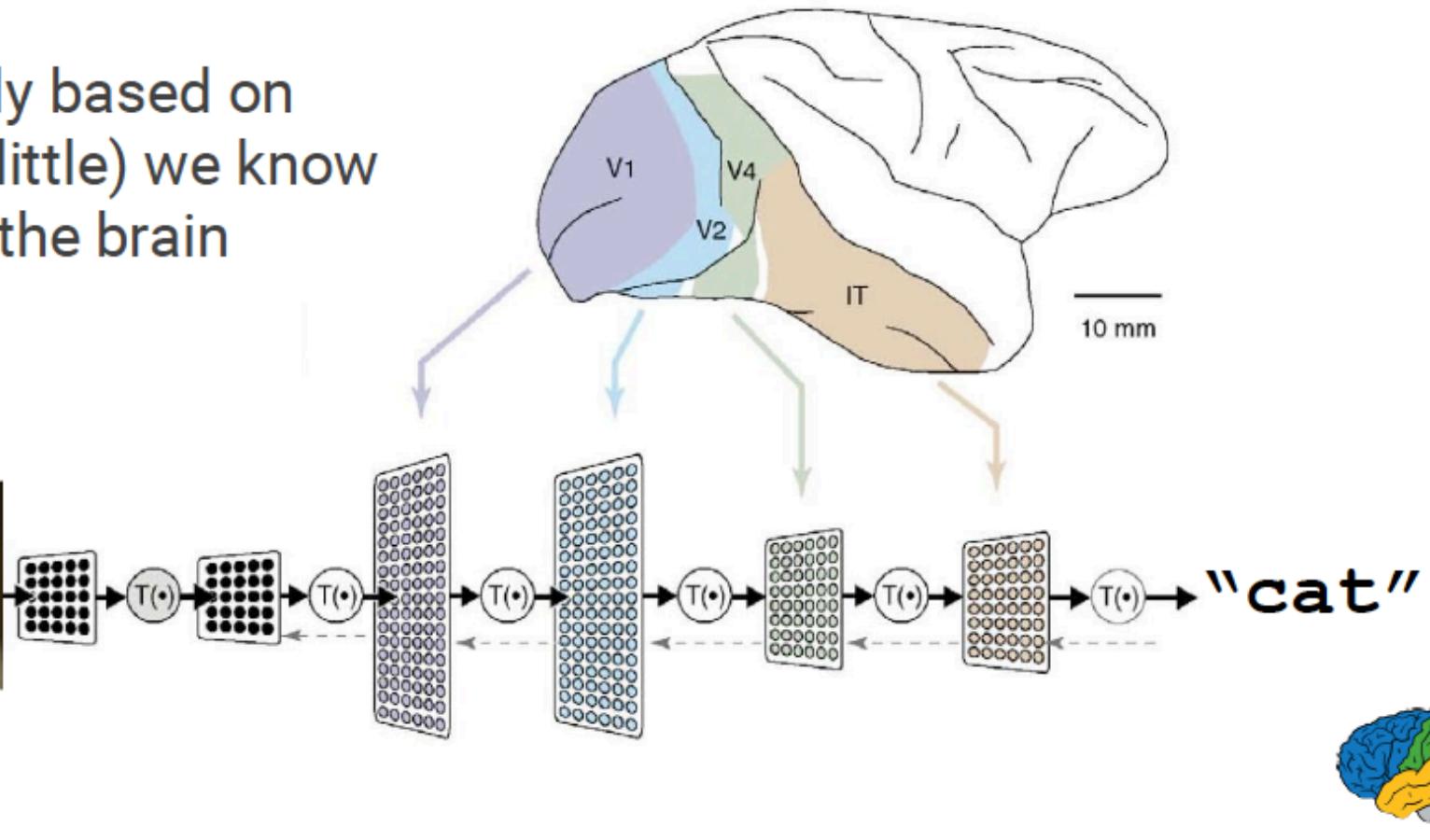
Deep Learning

- A powerful class of machine learning model
- Modern reincarnation of artificial neural networks
- Collection of simple, trainable mathematical functions
- Compatible with many variants of machine learning

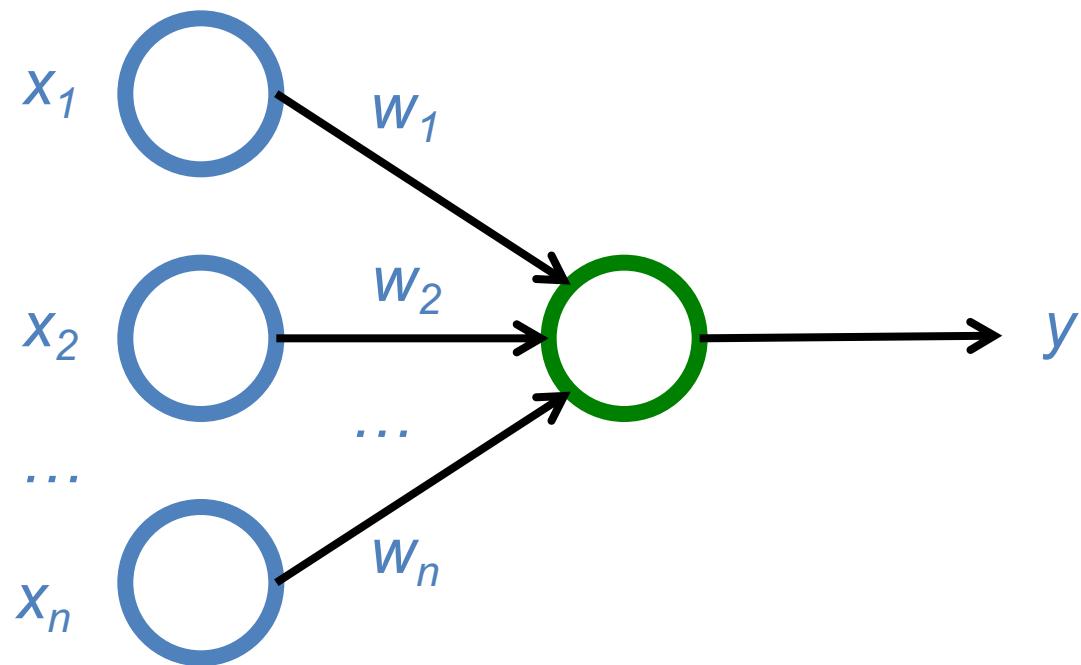


What is Deep Learning?

- Loosely based on (what little) we know about the brain

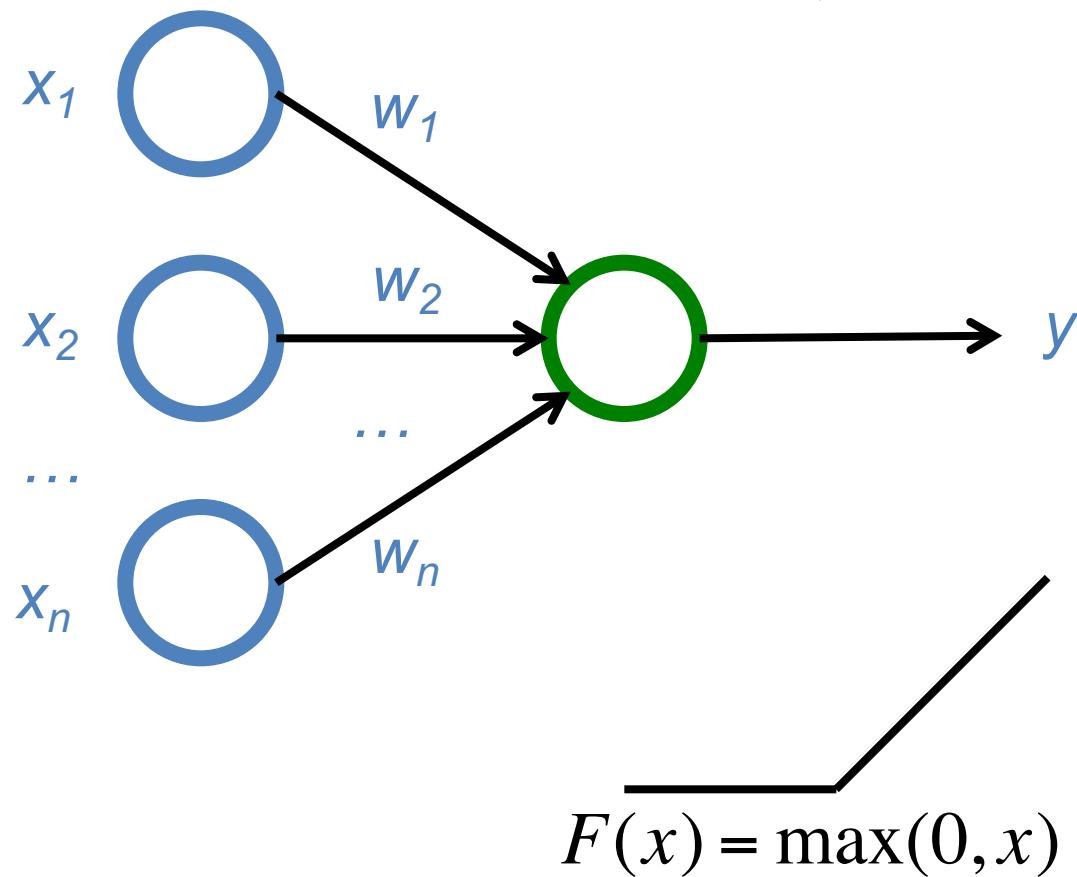


The Neuron

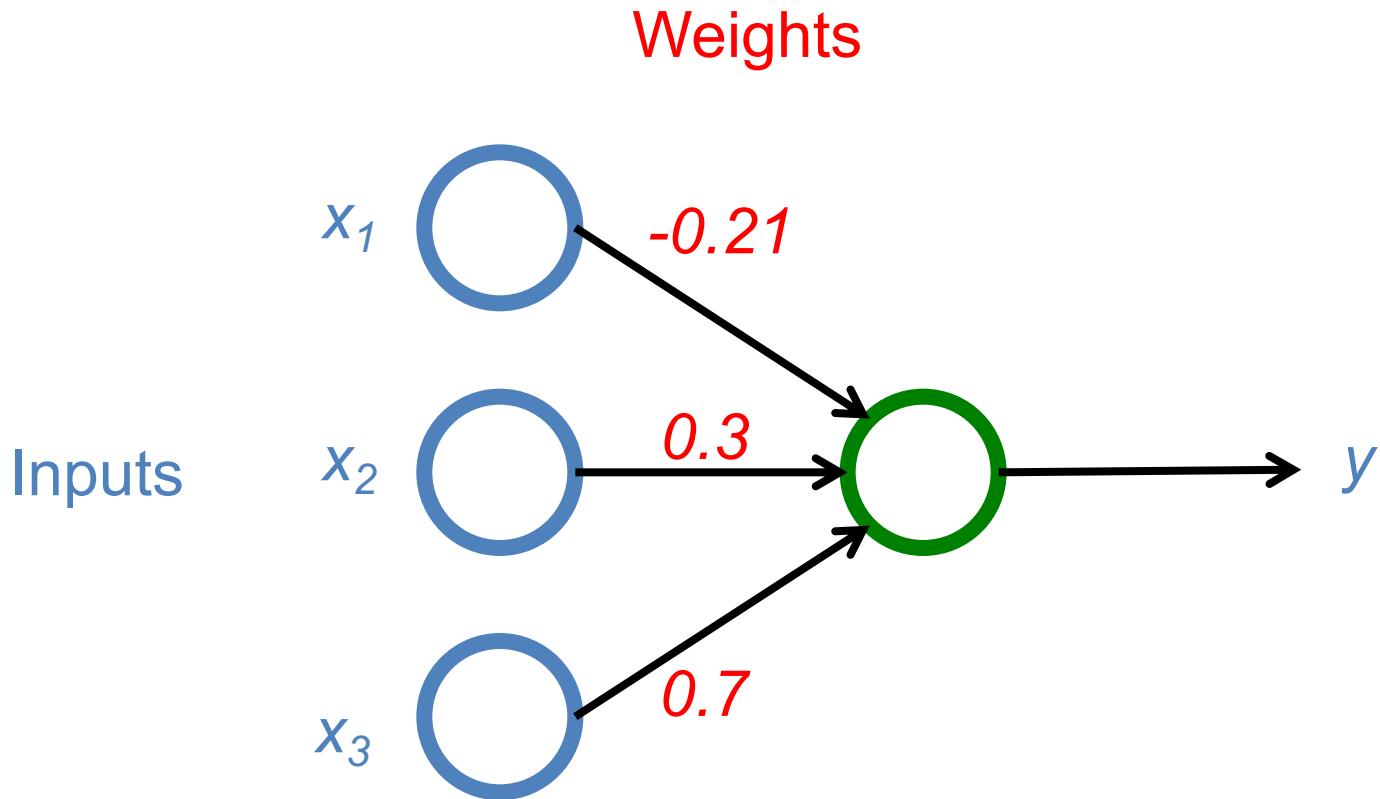


The Neuron

$$y = F\left(\sum_i w_i x_i\right)$$



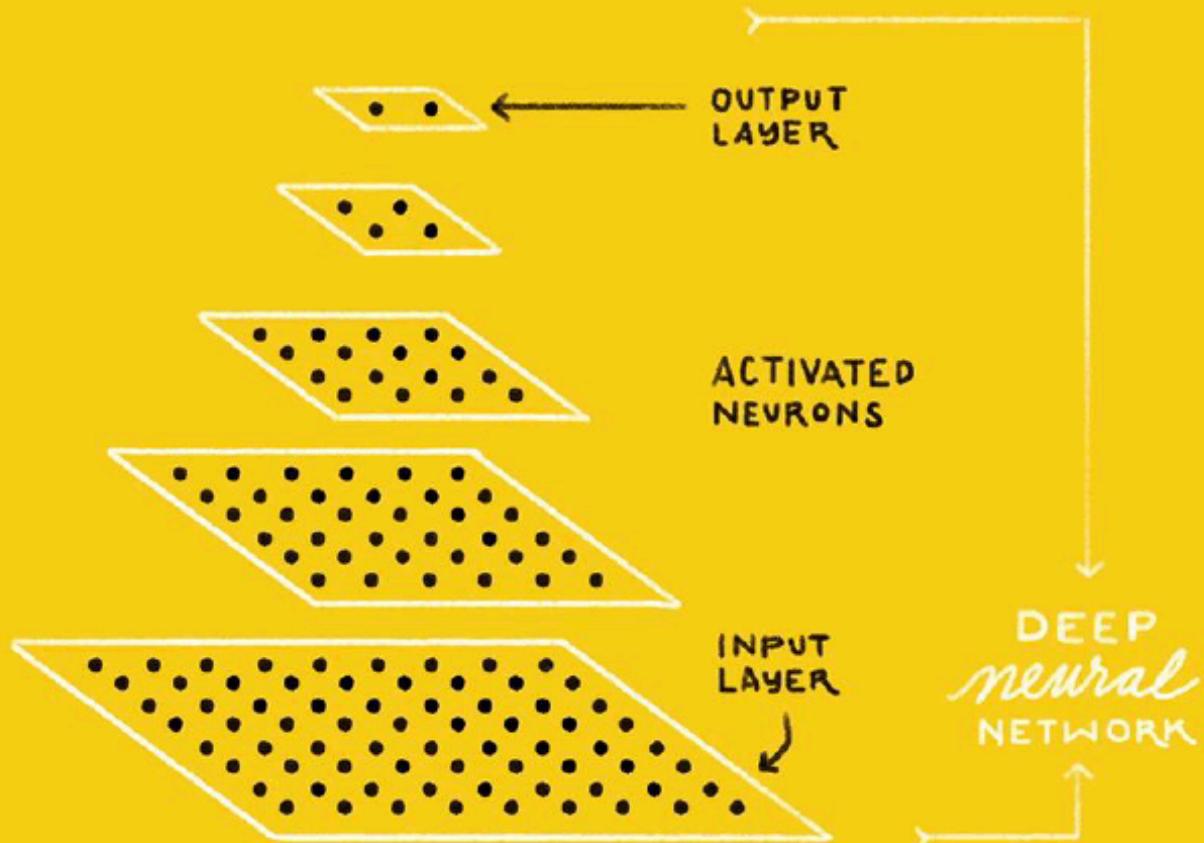
$$y = \max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$



IS THIS A
CAT or DOG?



CAT DOG



Learning Algorithm

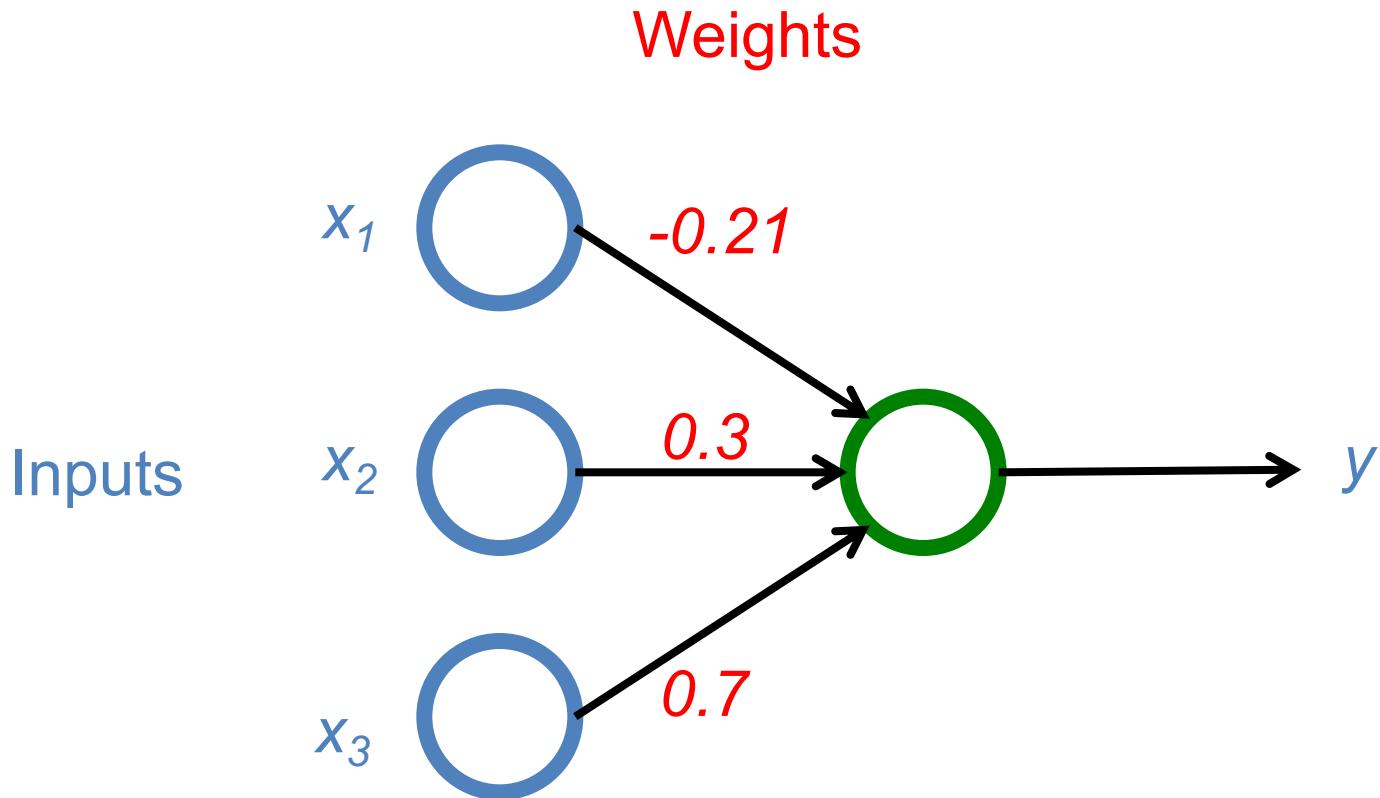
While not done:

Pick a random training example “(input, label)”

Run neural network on “input”

Adjust weights on edges to make output closer to “label”

$$y = \max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$

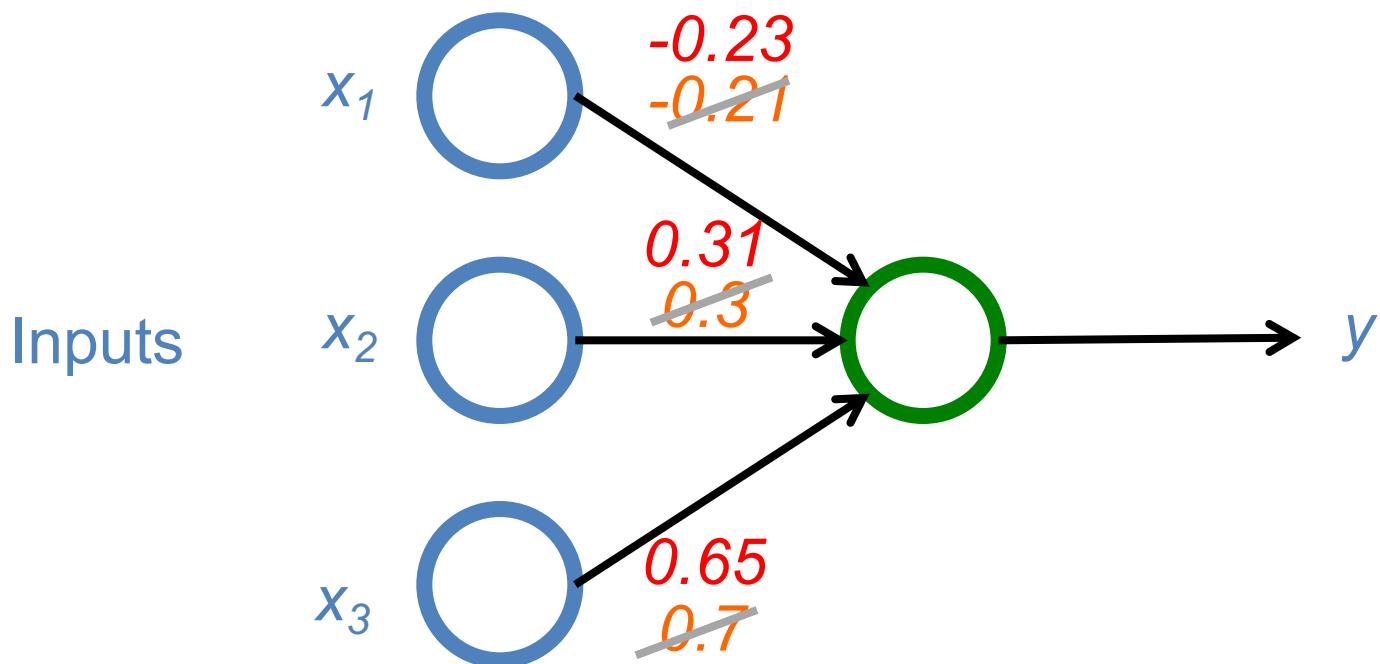


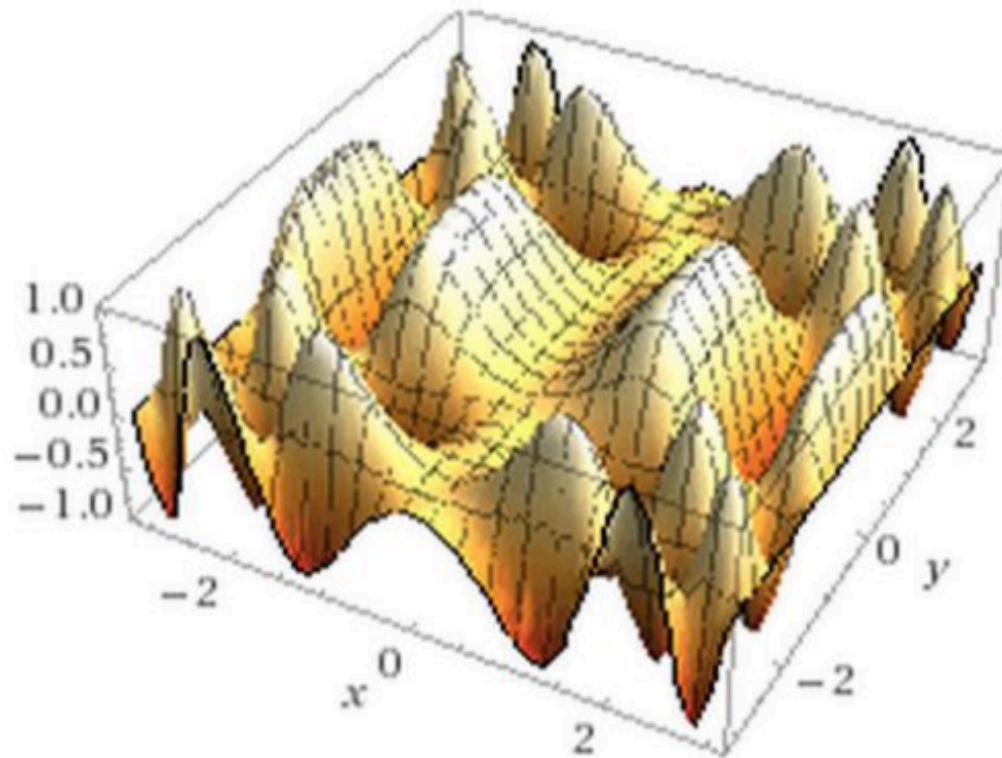
Next time:

$$y = \max(0, -0.23 * x_1 + 0.31 * x_2 + 0.65 * x_3)$$

$$y = \max(0, -0.21 * x_1 + 0.3 * x_2 + 0.7 * x_3)$$

Weights





This shows a function of 2 variables: real neural nets are functions of hundreds of millions of variables!

Important Property of Neural Networks

Results get better with

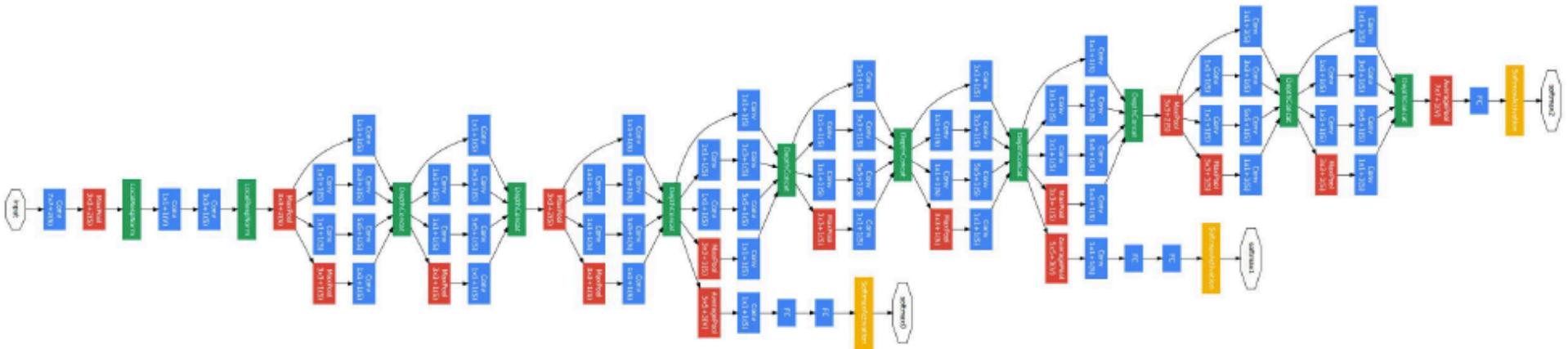
More data +

Bigger models +

More computation

(Better algorithms, new insights
and improved techniques always help, too!)

The Inception Architecture (GoogLeNet, 2014)



Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

ArXiv 2014, CVPR 2015



Deep Learning Software

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

Deep Learning

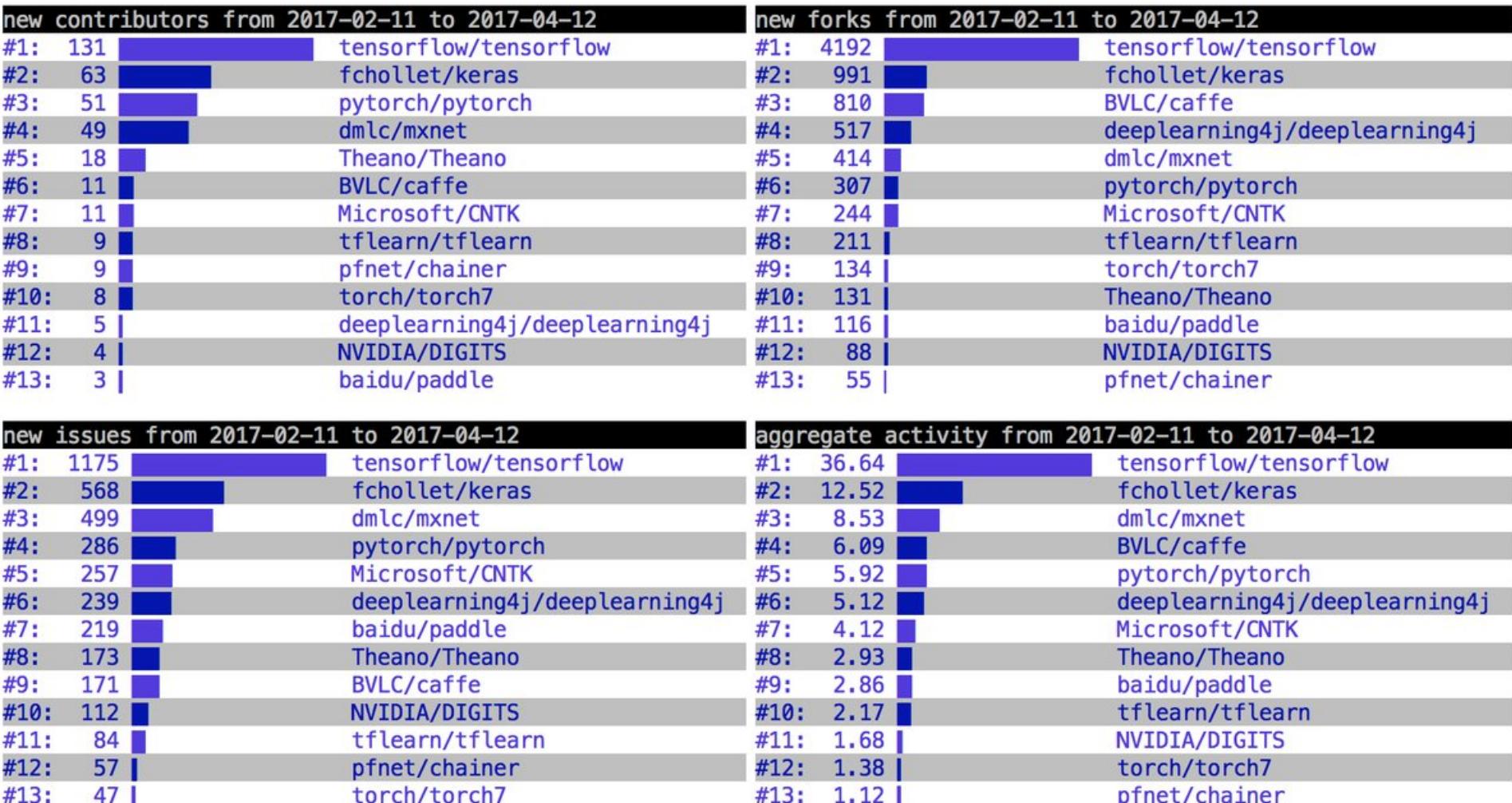
with

Google

TensorFlow

Deep Learning Libraries: Tensorflow and Keras

Deep learning libraries: GitHub activity from February 11 to April 12, 2017





TensorFlow

Google TensorFlow

TensorFlow™

GET STARTED TUTORIALS HOW TO API RESOURCES ABOUT

Fork me on GitHub

TensorFlow is an Open Source Software
Library for Machine Intelligence

GET STARTED

About TensorFlow

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.



<https://www.tensorflow.org/>

TensorFlow

is an

Open Source

Software Library

for

Machine Intelligence

numerical computation using data flow graphs

Tensor

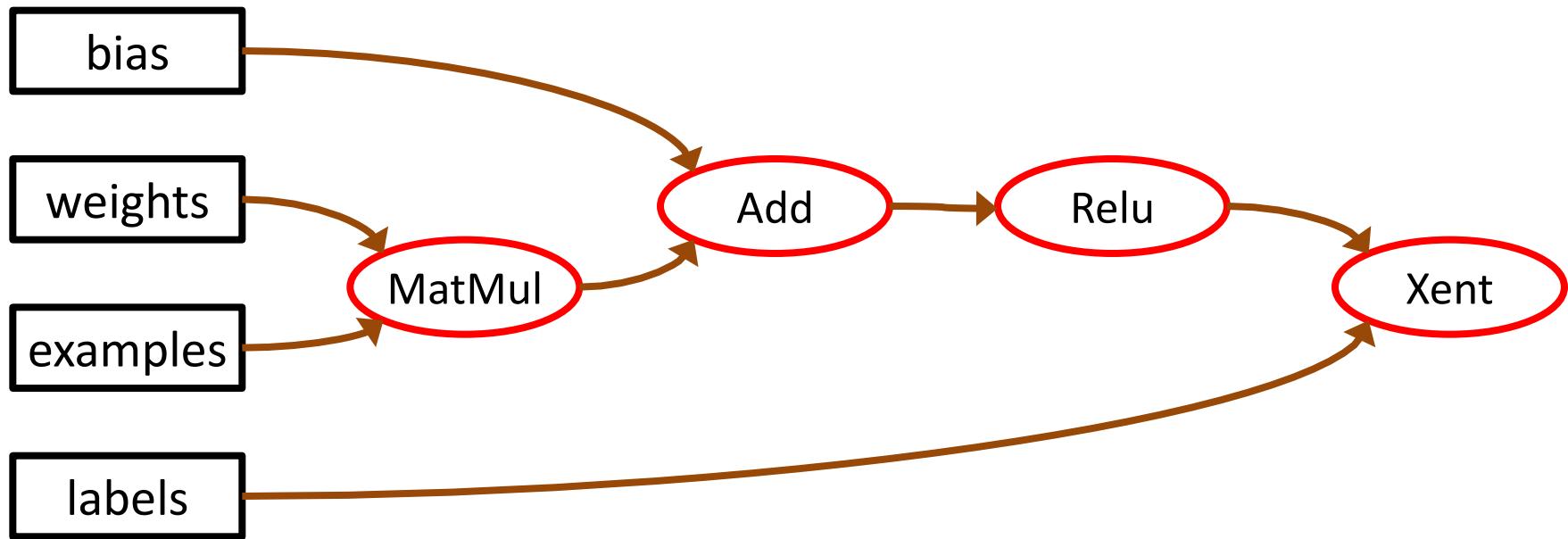
- **3**
 - # a rank 0 tensor; this is a **scalar** with shape []
- **[1., 2., 3.]**
 - # a rank 1 tensor; this is a **vector** with shape [3]
- **[[1., 2., 3.], [4., 5., 6.]]**
 - # a rank 2 tensor; a **matrix** with shape [2, 3]
- **[[[1., 2., 3.]], [[7., 8., 9.]]]**
 - # a rank 3 **tensor** with shape [2, 1, 3]

**Nodes:
mathematical operations**

**edges:
multidimensional data arrays
(tensors)
communicated between nodes**

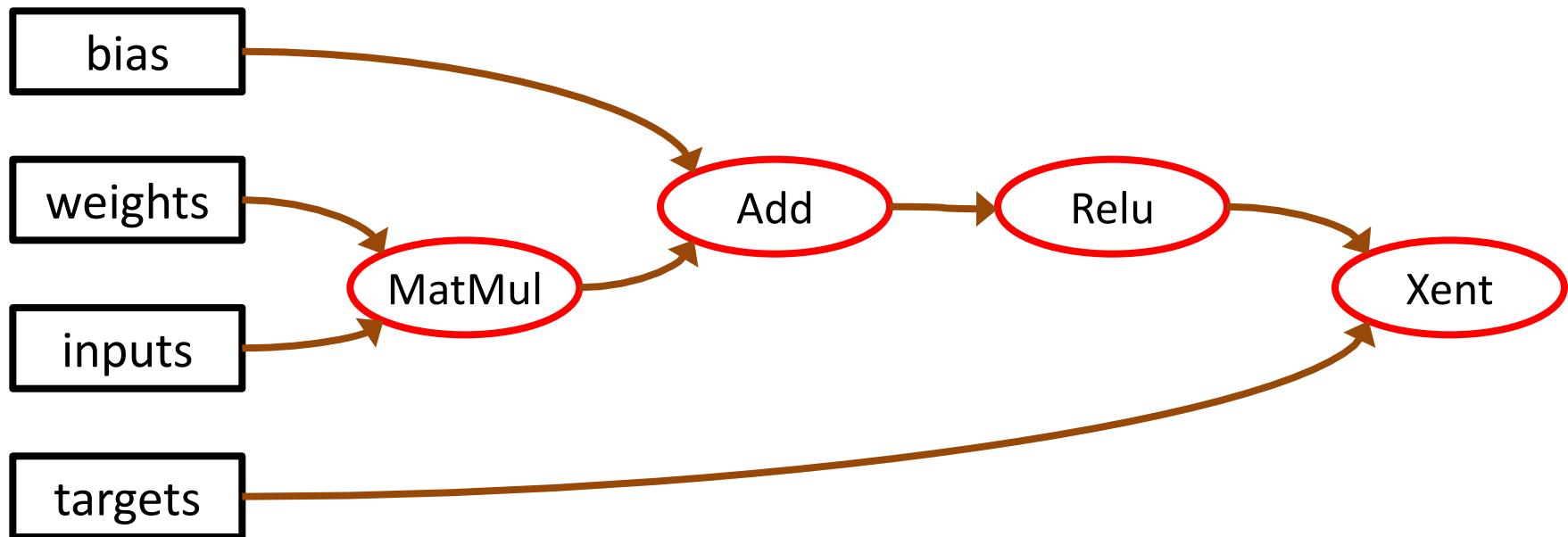
Computation is a Dataflow Graph

Graph of Nodes,
also called Operations or ops.

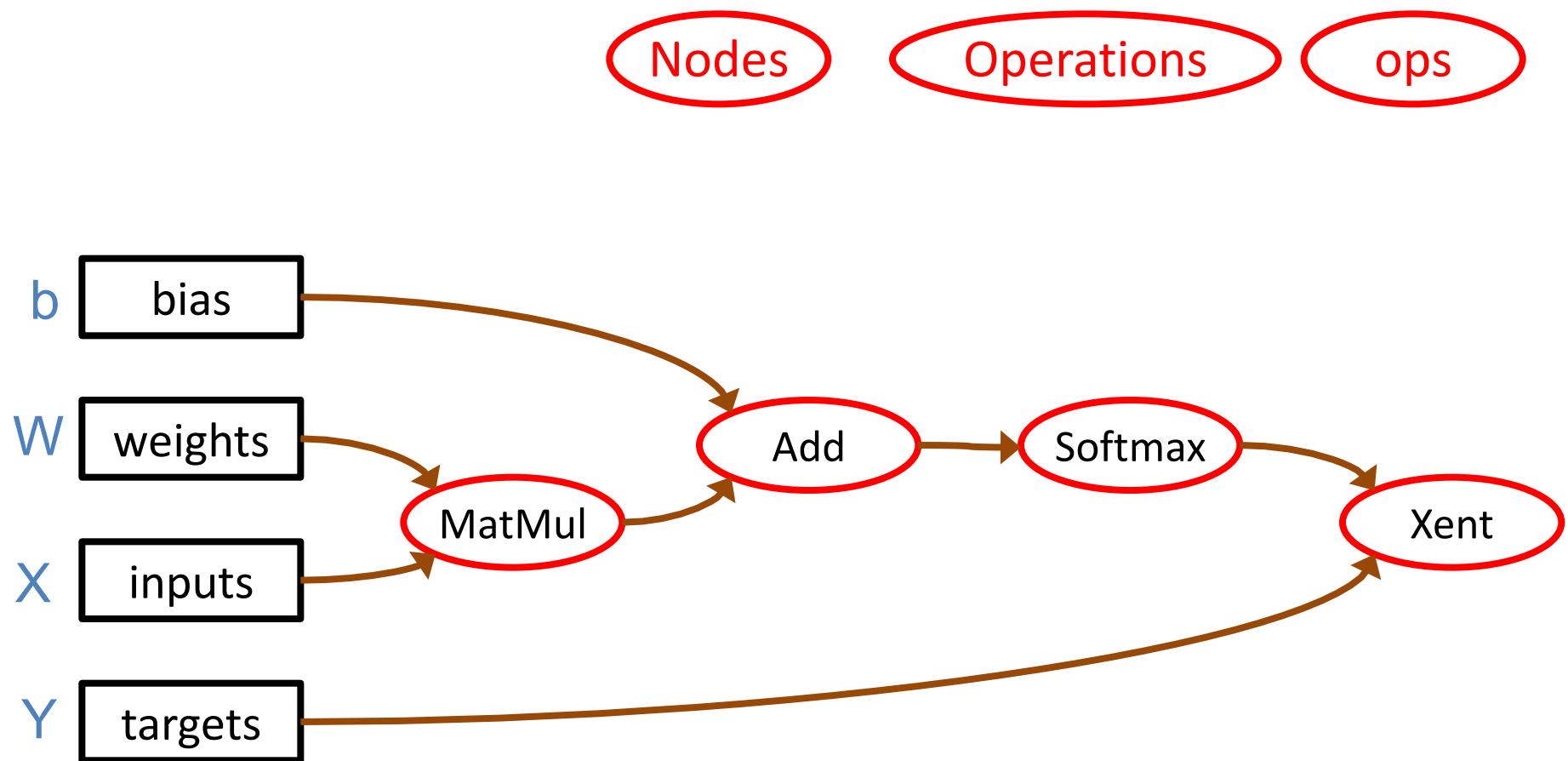


Computation is a Dataflow Graph

Edges are N-dimensional arrays: **Tensors**



Logistic Regression as Dataflow Graph

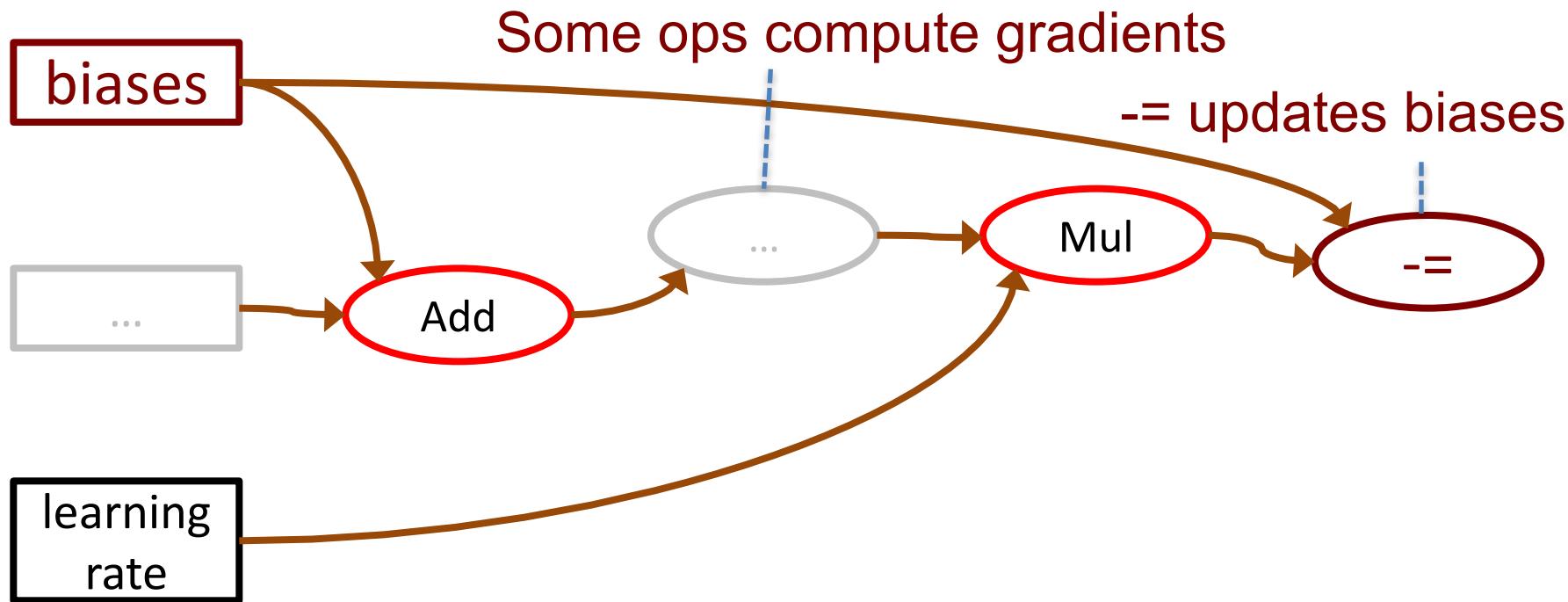


Edges are N-dimensional arrays: **Tensors**

Computation is a Dataflow Graph

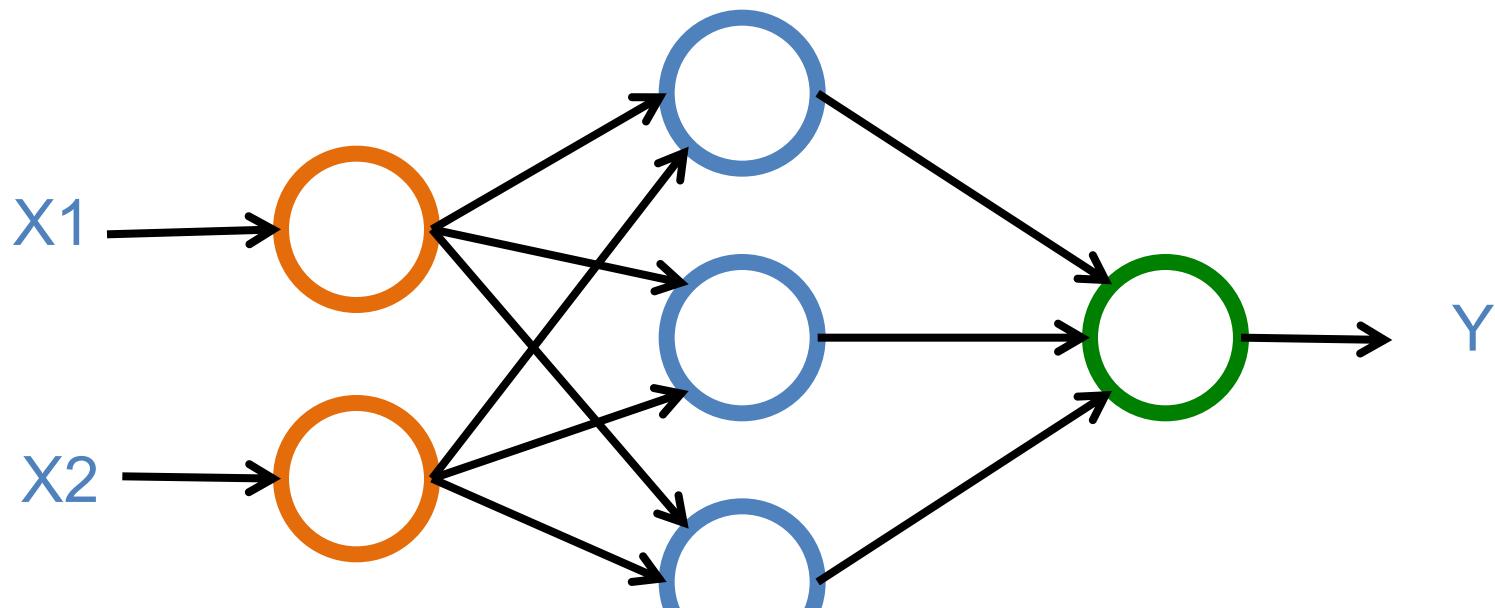
with state

'Biases' is a variable

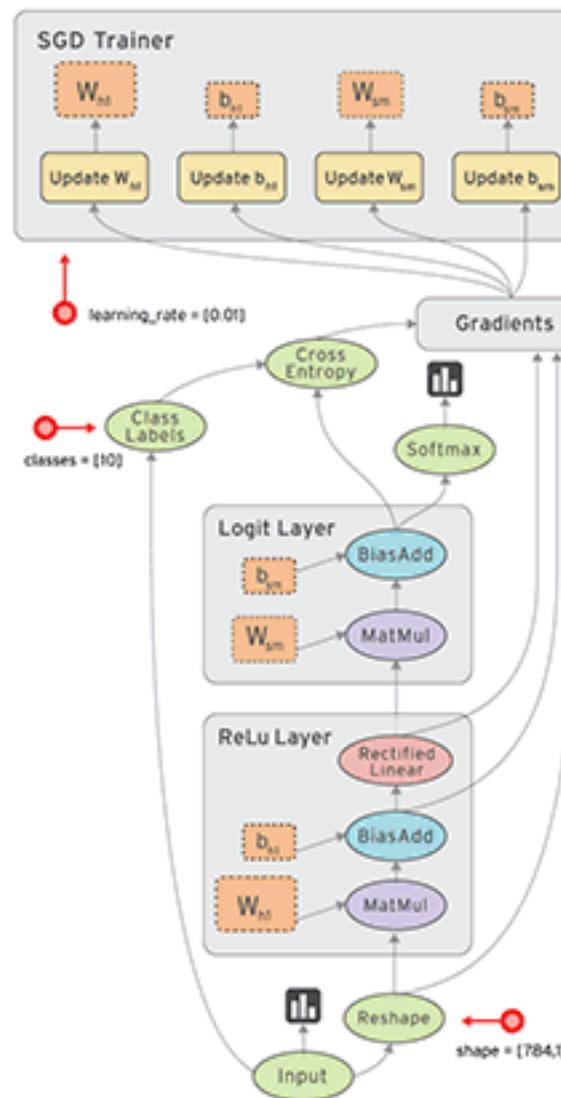


Neural Networks

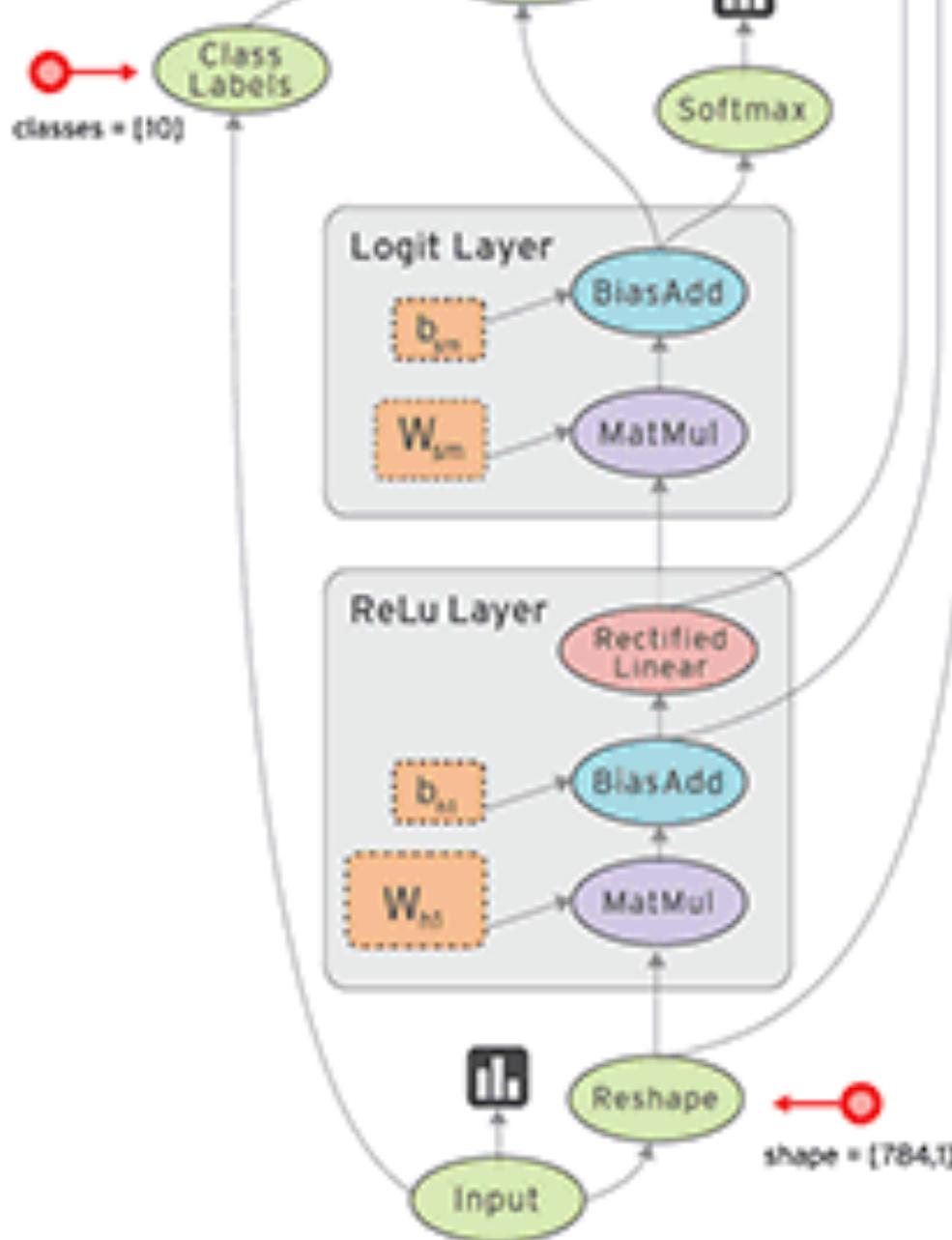
Input Layer **Hidden Layer** **Output Layer**
(X) **(H)** **(Y)**



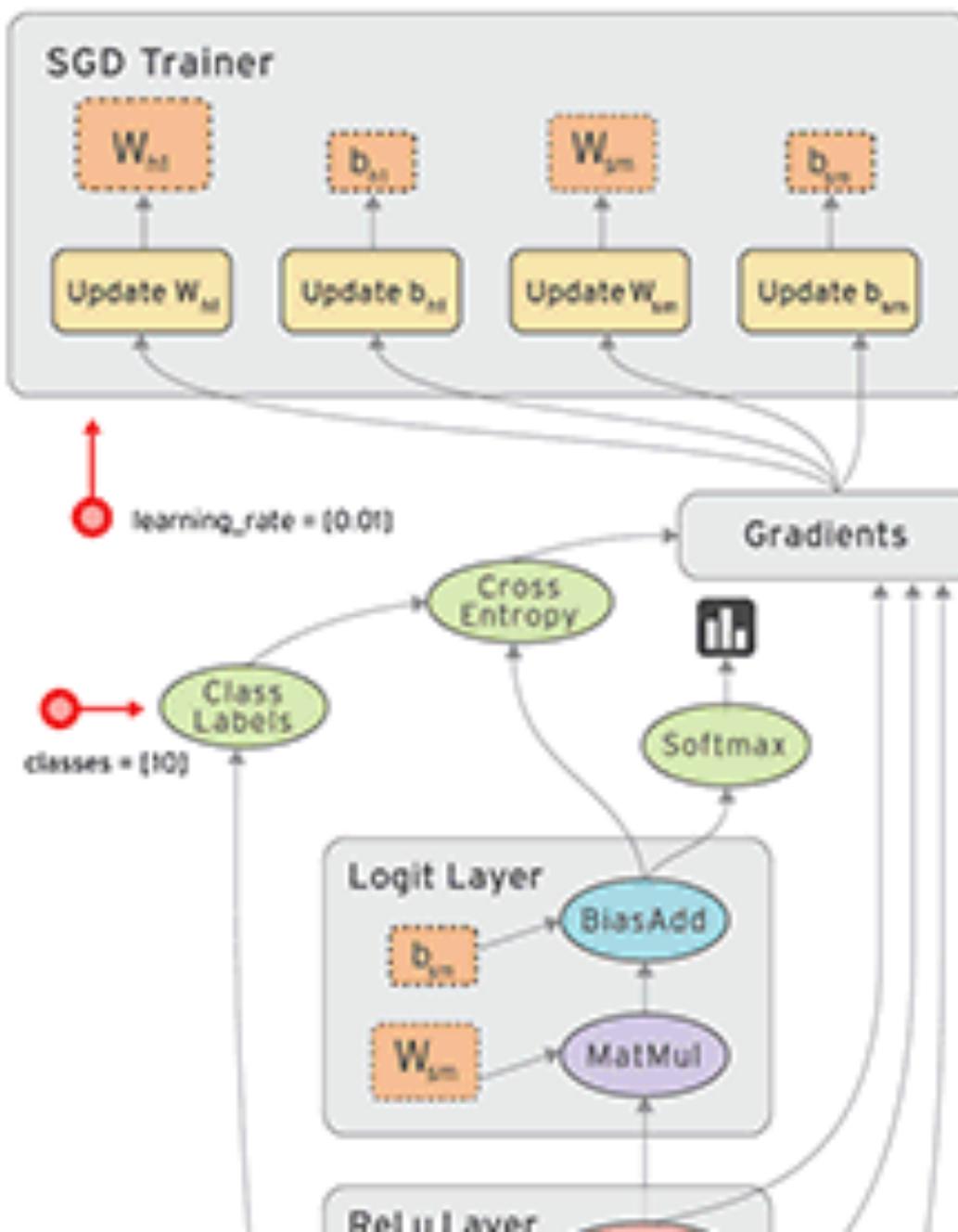
Data Flow Graph



Data Flow Graph



Data Flow Graph





Python

Download Anaconda



[Log In](#) [Get Support](#) [Search](#) [Contact](#)

[PRODUCTS](#) [COMMUNITY](#) [CONSULTING](#) [TRAINING](#) [ABOUT](#) [RESOURCES](#)

DOWNLOAD ANACONDA NOW

Download for



GET SUPERPOWERS WITH ANACONDA

Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science.

Which version should I download and install?

With Anaconda you can run multiple versions of Python in isolated environments, so choose the download with the Python version that you use more often, as that will be your default Python version.

<https://www.continuum.io/downloads>

Download Anaconda Python 3.6

Download for Windows

Download for macOS

Download for Linux

Anaconda 4.3.1

For macOS

macOS 10.12.2 users: To prevent permissions problems, we recommend that you upgrade to macOS 10.12.3 or later before installing Anaconda.

Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

Changelog

Graphical Installer

1. Download the graphical installer
2. Double-click the downloaded .pkg file and follow the instructions

Command Line Installer

1. Download the command-line installer
2. Optional: Verify data integrity with [MD5](#) or [SHA-256](#) [More info](#)
3. In your terminal window type one of the below and follow the instructions:
Python 3.6 version

Python 3.6 version

GRAPHICAL INSTALLER (424M)

COMMAND-LINE INSTALLER (363M)

64-Bit

Python 2.7 version

GRAPHICAL INSTALLER (419M)

COMMAND-LINE INSTALLER (358M)

64-Bit

GET ANACONDA SUPPORT

OS X Anaconda Python 3.6

Installation

Command Line Installer

Download the command-line installer

In your terminal window type one of the below
and follow the instructions:

Python 3.6 version

```
bash Anaconda3-4.3.1-MacOSX-x86_64.sh
```

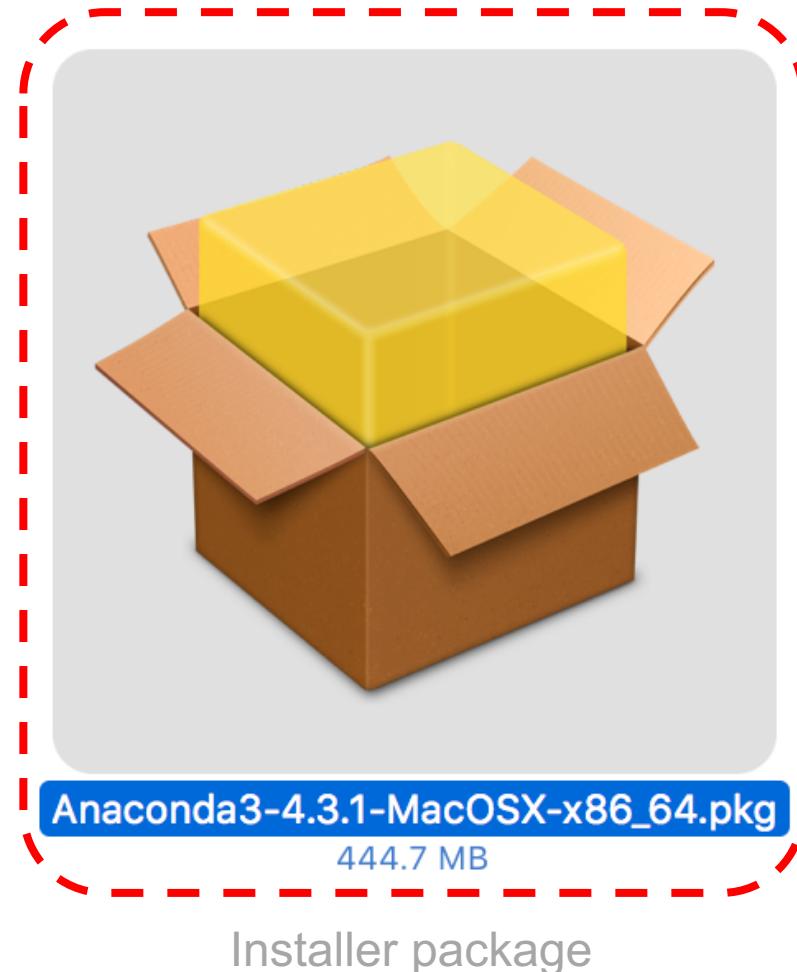
Python 2.7 version

```
bash Anaconda2-4.3.1-MacOSX-x86_64.sh
```

OS X Anaconda 3 - 4.3.1

Python 3.6 Installation

Anaconda3-4.3.1-MacOSX-x86_64.pkg



Install Anaconda 3

Install Anaconda3

Welcome to the Anaconda3 Installer

- **Introduction**
- Read Me
- License
- Destination Select
- Installation Type
- Installation
- Summary

You will be guided through the steps necessary to install this software.

ANACONDA

Go Back Continue

Install Anaconda 3

Install Anaconda3

Important Information

- Introduction
- Read Me**
- License
- Destination Select
- Installation Type
- Installation
- Summary

 **ANACONDA®**

Anaconda is a modern open source analytics platform powered by Python. See <https://www.continuum.io/downloads/>.

By default, this installer modifies your bash profile to put Anaconda in your PATH. To disable this, choose "Customize" at the "Installation Type" phase, and disable the "Modify PATH" option. If you do not do this, you will need to add `~/anaconda/bin` to your PATH manually to run the commands, or run all anaconda commands explicitly from that path.

To install to a different location, select "Change Install Location..." at the "Installation Type" phase, the choose "Install on a specific disk...", choose the disk you wish to install on, and click "Choose Folder...". The "Install for me only" option will install anaconda to the default location, `~/anaconda`.

The packages included in this installation are:

- alabaster 0.7.9

Print...

Save...

Go Back

Continue

110

Install Anaconda 3

Install Anaconda3

Software License Agreement

=====

Anaconda License

=====

Copyright 2016, Continuum Analytics, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of Continuum Analytics, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS

Print... Save... Go Back Continue



ANACONDA

Install Anaconda 3

Install Anaconda3

Software License Agreement

=====

Anaconda License

=====

Copyright 2016, Continuum Analytics, Inc.

To continue installing the software you must agree to the terms of the software license agreement.

Click Agree to continue or click Disagree to cancel the installation and quit the Installer.

[Read License](#) [Disagree](#) [Agree](#)

* Neither the name of Continuum Analytics, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS

[Print...](#) [Save...](#) [Go Back](#) [Continue](#)



ANACONDA®

Install Anaconda 3

Install Anaconda3

Standard Install on "Macintosh HD"

- Introduction
- Read Me
- License
- Destination Select
- Installation Type**
- Installation
- Summary

This will take 1.4 GB of space on your computer.

Click Install to perform a standard installation of this software in your home folder. Only the current user of this computer will be able to use this software.

Change Install Location...

Customize Go Back Install



ANACONDA®

Install Anaconda 3

Install Anaconda3

Select a Destination

- Introduction
- Read Me
- License
- **Destination Select**
- Installation Type
- Installation
- Summary

How do you want to install this software?

 Install for all users of this computer

 **Install for me only**

 Install on a specific disk...

Installing this software requires 1.4 GB of space.
You have chosen to install this software in your home folder.
Only the current user will be able to use this software.

Go Back Continue



Install Anaconda 3

Install Anaconda3

Standard Install on "Macintosh HD"

- Introduction
- Read Me
- License
- Destination Select
- Installation Type**
- Installation
- Summary

This will take 1.4 GB of space on your computer.

Click Install to perform a standard installation of this software in your home folder. Only the current user of this computer will be able to use this software.

Change Install Location...

Customize Go Back **Install**



ANACONDA®

Install Anaconda 3

Install Anaconda3

Installing Anaconda3

- Introduction
- Read Me
- License
- Destination Select
- Installation Type
- **Installation**
- Summary

Registering updated applications...

Install time remaining: About a minute

Go Back Continue



ANACONDA®

Install Anaconda 3

The installation was completed successfully.

Anaconda is the leading open data science platform powered by Python.

Share your notebooks and packages on Anaconda Cloud!
[Sign up for free](#)

178 python packages included.

**Supported packages:
453**

 **ANACONDA®**

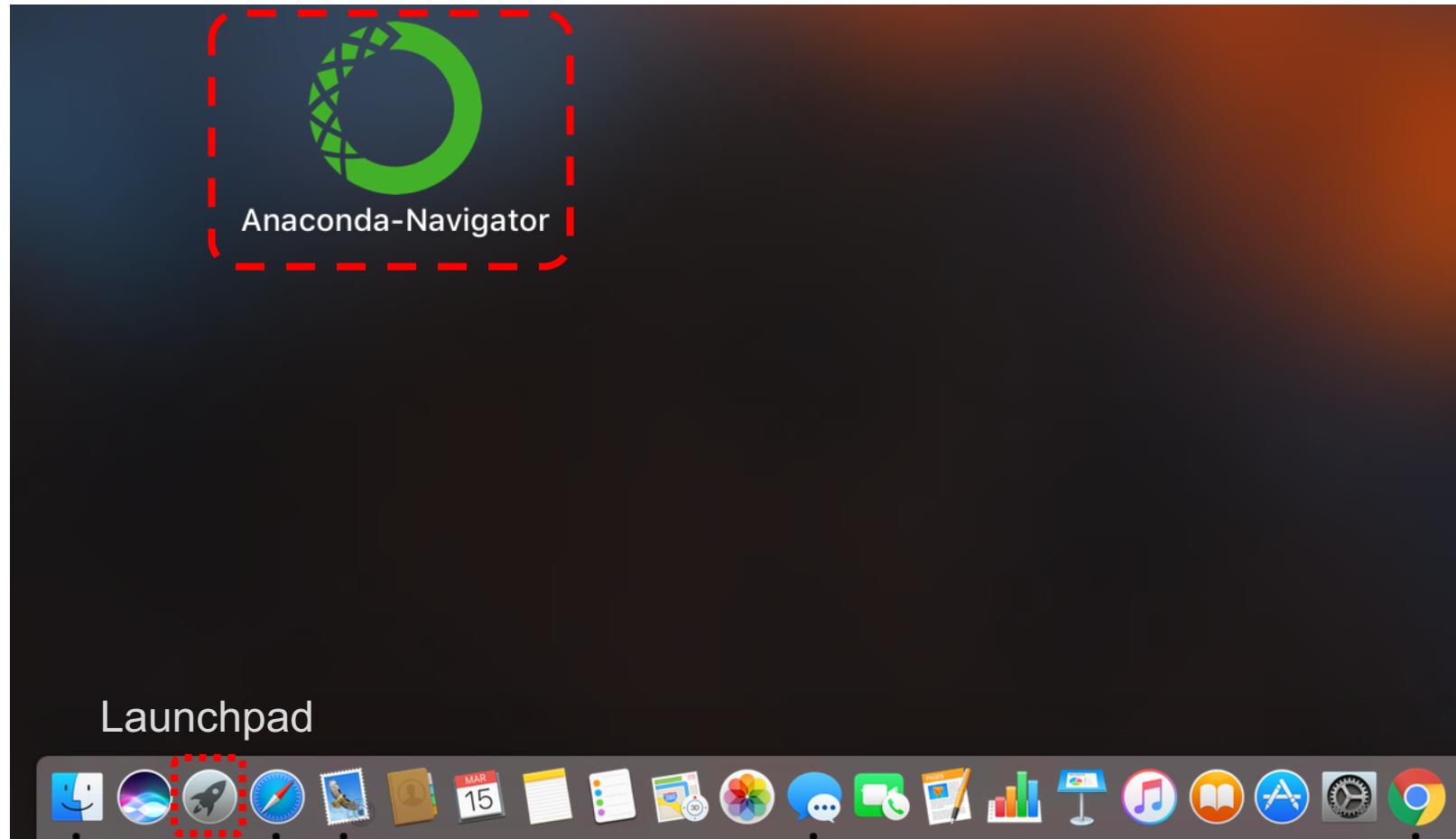
Go Back Close

Install Anaconda 3

1	_license	1.1	51	heapdict	1.0.0	101	partd	0.3.7	151	sip	4.18	py36_0
2	alabaster	0.7.9	52	icu	54.1	102	path.py	10.0	152	six	1.10.0	py36_0
3	anaconda	4.3.1	53	idna	2.2	103	pathlib2	2.2.0	153	snowballstemmer	1.2.1	py36_0
4	anaconda-client	1.6.0	54	imagesize	0.7.1	104	patsy	0.4.1	154	sockjs-tornado	1.0.3	py36_0
5	anaconda-navigator	1.5.0	55	ipykernel	4.5.2	105	pep8	1.7.0	155	sphinx	1.5.1	py36_0
6	anaconda-project	0.4.1	56	ipython	5.1.0	106	pexpect	4.2.1	156	spyder	3.1.2	py36_0
7	appnope	0.1.0	57	ipython_genutils	0.1.0	107	pickleshare	0.7.4	157	sqlalchemy	1.1.5	py36_0
8	appscript	1.0.1	58	ipywidgets	5.2.2	108	pillow	4.0.0	158	sqlite	3.13.0	0
9	astroid	1.4.9	59	isort	4.2.5	109	pip	9.0.1	159	statsmodels	0.6.1	np111py36_1
10	astropy	1.3	60	itsdangerous	0.24	110	ply	3.9	160	sympy	1.0	py36_0
11	babel	2.3.4	61	jbig	2.1	111	prompt_toolkit	1.0.9	161	terminado	0.6	py36_0
12	backports	1.0	62	jdcal	1.3	112	psutil	5.0.1	162	tk	8.5.18	0
13	beautifulsoup4	4.5.3	63	jedi	0.9.0	113	ptyprocess	0.5.1	163	toolz	0.8.2	py36_0
14	bitarray	0.8.1	64	jinja2	2.9.4	114	py	1.4.32	164	tornado	4.4.2	py36_0
15	blaze	0.10.1	65	jpeg	9b	115	pyasn1	0.1.9	165	traitlets	4.3.1	py36_0
16	bokeh	0.12.4	66	jsonschema	2.5.1	116	pycosat	0.6.1	166	unicodecsv	0.14.1	py36_0
17	boto	2.45.0	67	jupyter	1.0.0	117	pycparser	2.17	167	wcwidth	0.1.7	py36_0
18	bottleneck	1.2.0	68	jupyter_client	4.4.0	118	pycrypto	2.6.1	168	werkzeug	0.11.15	py36_0
19	cffi	1.9.1	69	jupyter_console	5.0.0	119	pycurl	7.43.0	169	wheel	0.29.0	py36_0
20	chardet	2.3.0	70	jupyter_core	4.2.1	120	pyflakes	1.5.0	170	widgetsnbextension	1.2.6	py36_0
21	chest	0.2.3	71	lazy-object-proxy	1.2.2	121	pygments	2.1.3	171	wrapt	1.10.8	py36_0
22	click	6.7	72	libiconv	1.14	122	pylint	1.6.4	172	xlrd	1.0.0	py36_0
23	cloudpickle	0.2.2	73	libpng	1.6.27	123	pyopenssl	16.2.0	173	xlsxwriter	0.9.6	py36_0
24	clyent	1.2.2	74	libtiff	4.0.6	124	pyparsing	2.1.4	174	xlwings	0.10.2	py36_0
25	colorama	0.3.7	75	libxml2	2.9.4	125	pyqt	5.6.0	175	xlwt	1.2.0	py36_0
26	conda	4.3.14	76	libxslt	1.1.29	126	pytables	3.3.0	176	xz	5.2.2	1
27	conda-env	2.6.0	77	llvmlite	0.15.0	127	pytest	3.0.5	177	yaml	0.1.6	0
28	configobj	5.0.6	78	locket	0.2.0	128	python	3.6.0	178	zlib	1.2.8	3
29	contextlib2	0.5.4	79	lxml	3.7.2	129	python-dateutil	2.6.0		py36_0		
30	cryptography	1.7.1	80	markupsafe	0.23	130	python.app	1.2		py36_4		
31	curl	7.52.1	81	matplotlib	2.0.0	131	pytz	2016.10		py36_0		
32	cycler	0.10.0	82	mistune	0.7.3	132	pyyaml	3.12		py36_0		
33	cython	0.25.2	83	mkl	2017.0.1	133	pyzmq	16.0.2		py36_0		
34	cytoolz	0.8.2	84	mkl-service	1.1.2	134	qt	5.6.2		0		
35	dask	0.13.0	85	mpmath	0.19	135	qtawesome	0.4.3		py36_0		
36	datashape	0.5.4	86	multipledispatch	0.4.9	136	qtconsole	4.2.1		py36_1		
37	decorator	4.0.11	87	nbconvert	4.2.0	137	qtpy	1.2.1		py36_0		
38	dill	0.2.5	88	nbformat	4.2.0	138	readline	6.2		2		
39	docutils	0.13.1	89	networkx	1.11	139	redis	3.2.0		0		
40	entrypoints	0.2.2	90	nltk	3.2.2	140	redis-py	2.10.5		py36_0		
41	et_xmlfile	1.0.1	91	nose	1.3.7	141	requests	2.12.4		py36_0		
42	fastcache	1.0.2	92	notebook	4.3.1	142	rope	0.9.4		py36_1		
43	flask	0.12	93	numba	0.30.1	143	ruamel_yaml	0.11.14		py36_1		
44	flask-cors	3.0.2	94	numexpr	2.6.1	144	scikit-image	0.12.3		np111py36_1		
45	freetype	2.5.5	95	numpy	1.11.3	145	scikit-learn	0.18.1		np111py36_1		
46	get_terminal_size	1.0.0	96	numpydoc	0.6.0	146	scipy	0.18.1		np111py36_1		
47	gevent	1.2.1	97	odo	0.5.0	147	seaborn	0.7.1		py36_0		
48	greenlet	0.4.11	98	openpyxl	2.4.1	148	setuptools	27.2.0		py36_0		
49	h5py	2.6.0	99	openssl	1.0.2k	149	simplegeneric	0.8.1		py36_1		
50	hdf5	1.8.17	100	pandas	0.19.2	150	singledispatch	3.4.0.3		py36_0		

178
python
packages
included.

Anaconda-Navigator



Anaconda-Navigator

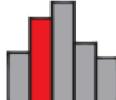
The screenshot shows the Anaconda Navigator application window. At the top, there's a navigation bar with the Anaconda Navigator logo, a search bar labeled "Applications on root", and buttons for "Upgrade Now" and "Sign in to Anaconda Cloud". On the left, a sidebar menu includes "Home", "Environments", "Projects (beta)", "Learning", "Community", "Documentation", "Developer Blog", and "Feedback", along with social media links for Twitter, YouTube, and GitHub.

The main content area displays a grid of application icons. A central modal dialog box is open, displaying a welcome message: "Thanks for installing Anaconda! Anaconda Navigator helps you easily start important Python applications and manage the packages in your local Anaconda installation. It also connects you to online resources for learning and engaging with the Python, SciPy, and PyData community." It also contains a note about usage data collection and a checkbox for opting out. The "Ok" button is highlighted with a red dashed box, and a green "Ok, and don't show again" button is visible below it.

The applications listed in the grid are:

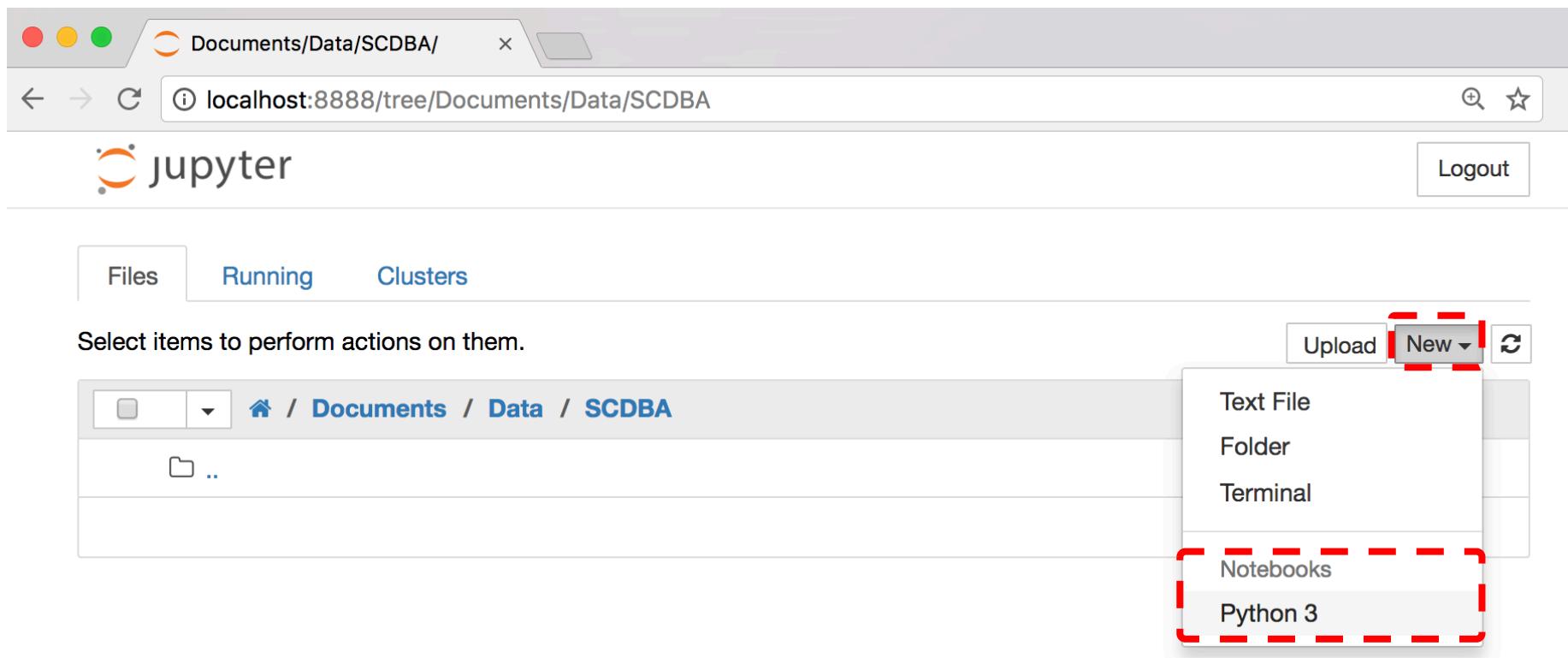
- jupyter notebook (4.3.1) - Web-based, interactive computing environment. Edit and run human docs while describing the data. Includes a "Launch" button.
- anaconda-fusion (1.0.2) - Integration between Excel® and Anaconda via Notebooks. Run data science functions, interact with results and create advanced visualizations in a code-free app inside Excel. Includes an "Install" button.
- glueviz (0.9.1) - Multidimensional data visualization across files. Explore relationships within and among related datasets. Includes an "Install" button.
- spyder (3.1.2) - Python Development. Powerful Python IDE with editing, interactive testing, introspection features. Includes a "Launch" button.
- rstudio (1.0.136) - A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. Includes an "Install" button.

Jupyter Notebook

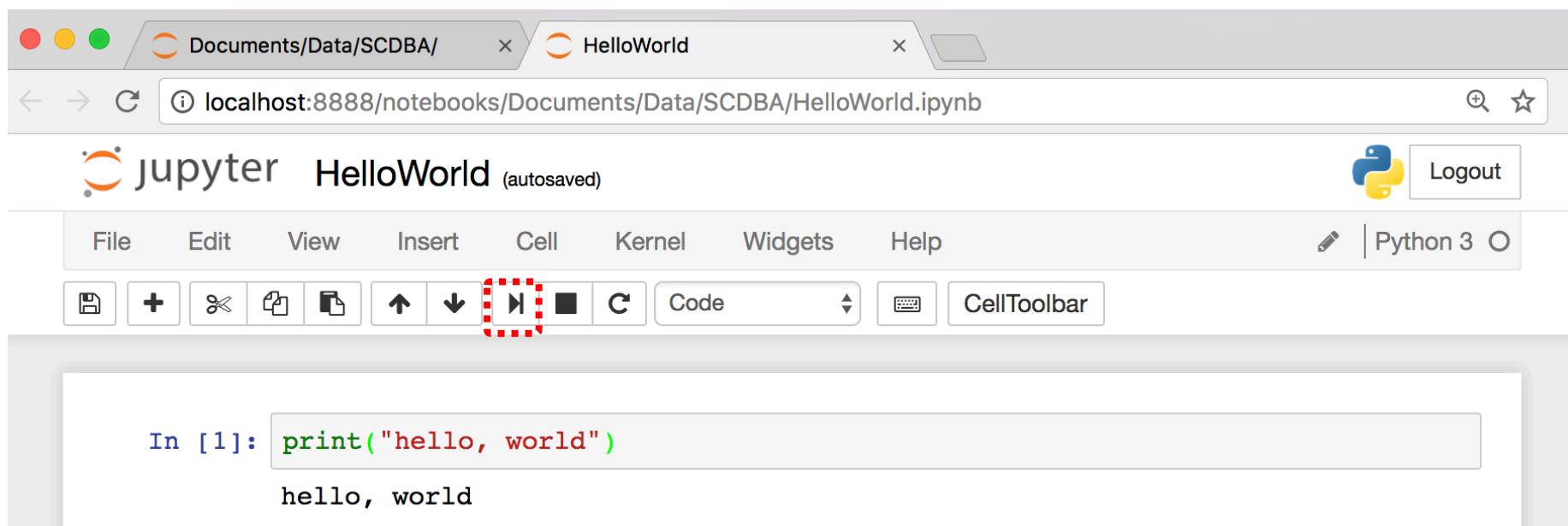
 <p>jupyter notebook  4.3.1</p> <p>Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.</p> <p>Launch</p>	 <p>qtconsole 4.2.1</p> <p>PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.</p> <p>Launch</p>	 <p>spyder  3.1.2</p> <p>Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features</p> <p>Launch</p>
 <p>anaconda-fusion 1.0.2</p> <p>Integration between Excel ® and Anaconda via Notebooks. Run data science functions, interact with results and create advanced visualizations in a code-free app inside Excel</p> <p>Install</p>	 <p>glueviz 0.9.1</p> <p>Multidimensional data visualization across files. Explore relationships within and among related datasets.</p> <p>Install</p>	 <p>rstudio 1.0.136</p> <p>A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.</p> <p>Install</p>

Jupyter Notebook

New Python 3



print("hello, world")



```
from platform import python_version
print("Python Version:", python_version())
```

The screenshot shows a Jupyter Notebook interface with the following details:

- Header:** Shows tabs for "Documents/Data/SCDBA/" and "HelloWorld". The "HelloWorld" tab is active.
- Breadcrumbs:** Shows the URL: "localhost:8888/notebooks/Documents/Data/SCDBA>HelloWorld.ipynb".
- User Information:** Shows "jupyter HelloWorld (autosaved)" and a "Logout" button.
- Toolbar:** Includes standard file operations (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a "CellToolbar" button.
- Cells:**
 - In [1]:** `print("hello, world")`
Output: `hello, world`
 - In [2]:** `from platform import python_version
print("Python Version:", python_version())`
Output: `Python Version: 3.6.0`

Create Python Environments with Anaconda

- Python 3.6
- Python 3.5
 - Python 3.5.3
 - Python 3.5.2
- Python 2.7

Anaconda Create New Python 3.5 Environment (py35)

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

Home Environments Projects (beta) Learning Community Documentation Developer Blog Feedback

Search Environments root

Installed Channels Update index... Search Packages

Create new environment

Environment name: py35

Python version: 3.5

Create Cancel

186 packages available (root)

py35 Python 3.5

The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with links for Home, Environments (which is selected and highlighted with a red dashed border), Projects (beta), Learning, Community, Documentation, Developer Blog, and Feedback. Below the sidebar are social media icons for Twitter, YouTube, and GitHub. The main area has tabs for Installed, Channels, and Update index... At the top right is a 'Sign in to Anaconda Cloud' button. In the center, there's a search bar for environments and a table of installed packages. A modal window titled 'Create new environment' is open, showing fields for 'Environment name' (set to 'py35'), 'Python' (checked), 'R' (unchecked), and 'Python version' (set to '3.5'). Below the modal, the package list starts with '_license' and includes 'astroid', 'astropy', 'babel', 'backports', 'backports.shutil-get-terminal-size', and 'beautifulsoup4'. At the bottom of the package list, it says '186 packages available (root)'. A large red text overlay 'py35 Python 3.5' is overlaid on the left side of the central area.

Anaconda Create New Python 2.7 Environment (py27)

The screenshot shows the Anaconda Navigator interface. On the left, there's a sidebar with icons for Home, Environments (selected), Projects (beta), Learning, and Community. Below the sidebar are links for Documentation, Developer Blog, and Feedback, along with social media icons for Twitter, YouTube, and GitHub.

In the main area, there's a search bar for environments and a list of environments: root and py35. A red dashed box highlights the 'Environments' section. Red text overlays on the left side read "py35 Python 3.5" and "py27 Python 2.7".

The central part of the interface shows a table of installed packages in the 'root' environment. The table includes columns for Name, Description, and Version. Packages listed include openssl, pip, python, readline, setuptools, sqlite, tk, wheel, xz, and zlib.

A modal window titled "Create new environment" is open in the bottom right. It has fields for "Environment name" (set to "py27"), "Python" (checked), "R" (unchecked), and "Python version" (set to "2.7"). There are "Cancel" and "Create" buttons at the bottom.

At the bottom of the main pane, it says "10 packages available (/Users/imyday/anaconda/envs/py35)".

Verify that conda is installed, check current conda version

- **conda --version**
- Update conda to the current version
 - **conda update conda**

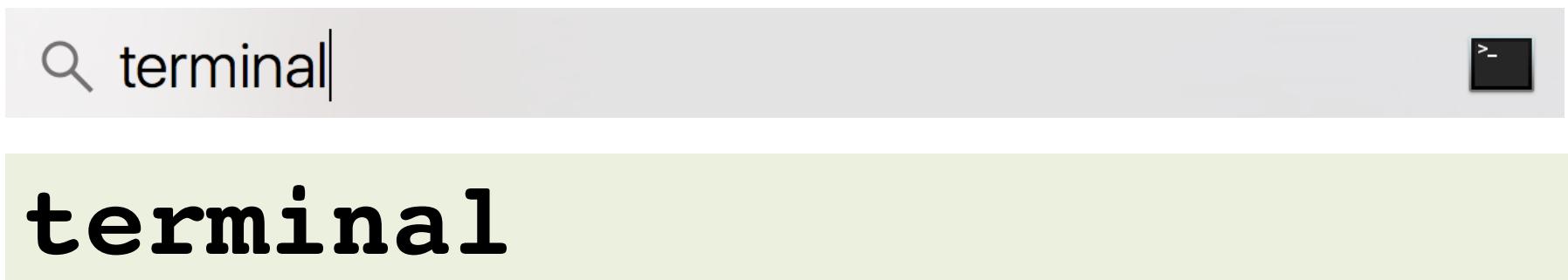
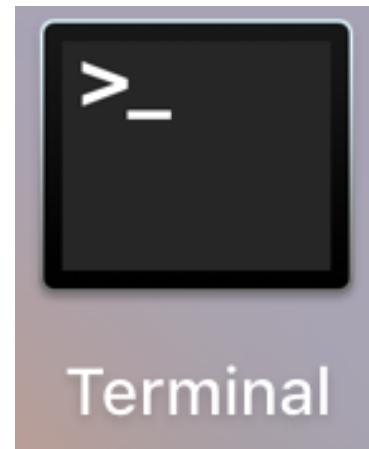
Check current conda version

Check current python version

Check conda environments

- **conda --version**
- **python --version**
- **conda info --envs**

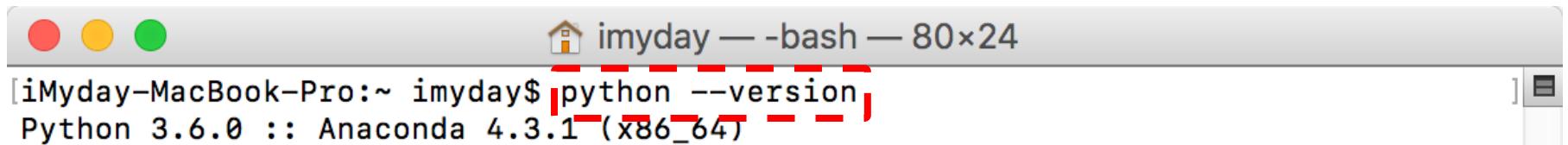
Terminal



conda list

```
iMyday-MacBook-Pro:~ imyday$ conda list
# packages in environment at /Users/imyday/anaconda:
#
_license           1.1                  py36_1
alabaster          0.7.9                py36_0
anaconda           4.3.1      np111py36_0
anaconda-client    1.6.0                py36_0
anaconda-navigator 1.5.0                py36_0
anaconda-project   0.4.1                py36_0
appnope             0.1.0                py36_0
appscript            1.0.1                py36_0
astroid              1.4.9                py36_0
astropy             1.3      np111py36_0
babel               2.3.4                py36_0
backports           1.0                  py36_0
beautifulsoup4     4.5.3                py36_0
bitarray             0.8.1                py36_0
blaze                0.10.1               py36_0
bokeh                0.12.4               py36_0
boto                 2.45.0                py36_0
bottleneck          1.2.0      np111py36_0
cffi                 1.9.1                py36_0
chardet              2.3.0                py36_0
chest                 0.2.3                py36_0
```

python --version



A screenshot of a macOS terminal window titled "imyday — -bash — 80x24". The window has red, yellow, and green close buttons. The command "python --version" is typed at the prompt, and the output "Python 3.6.0 :: Anaconda 4.3.1 (x86_64)" is displayed. The word "python" in the command and the first part of the output are highlighted with a red dashed underline.

```
[iMyday-MacBook-Pro:~ imyday$ python --version
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)
```

conda --version

```
iMyday-MacBook-Pro:~ imyday$ python --version  
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)  
[iMyday-MacBook-Pro:~ imyday$ conda --version  
conda 4.3.14  
[iMyday-MacBook-Pro:~ imyday$ conda info --envs  
# conda environments:  
#  
py27          /Users/imyday/anaconda/envs/py27  
py35          /Users/imyday/anaconda/envs/py35  
root          * /Users/imyday/anaconda
```

python --version
conda --version
conda info --envs

```
[iMyday-MacBook-Pro:~ imyday$ source activate py35  
[(py35) iMyday-MacBook-Pro:~ imyday$ python --version  
Python 3.5.3 :: Continuum Analytics, Inc.
```

source activate py35

```
[(py35) iMyday-MacBook-Pro:~ imyday$ conda --version  
conda 4.3.14
```

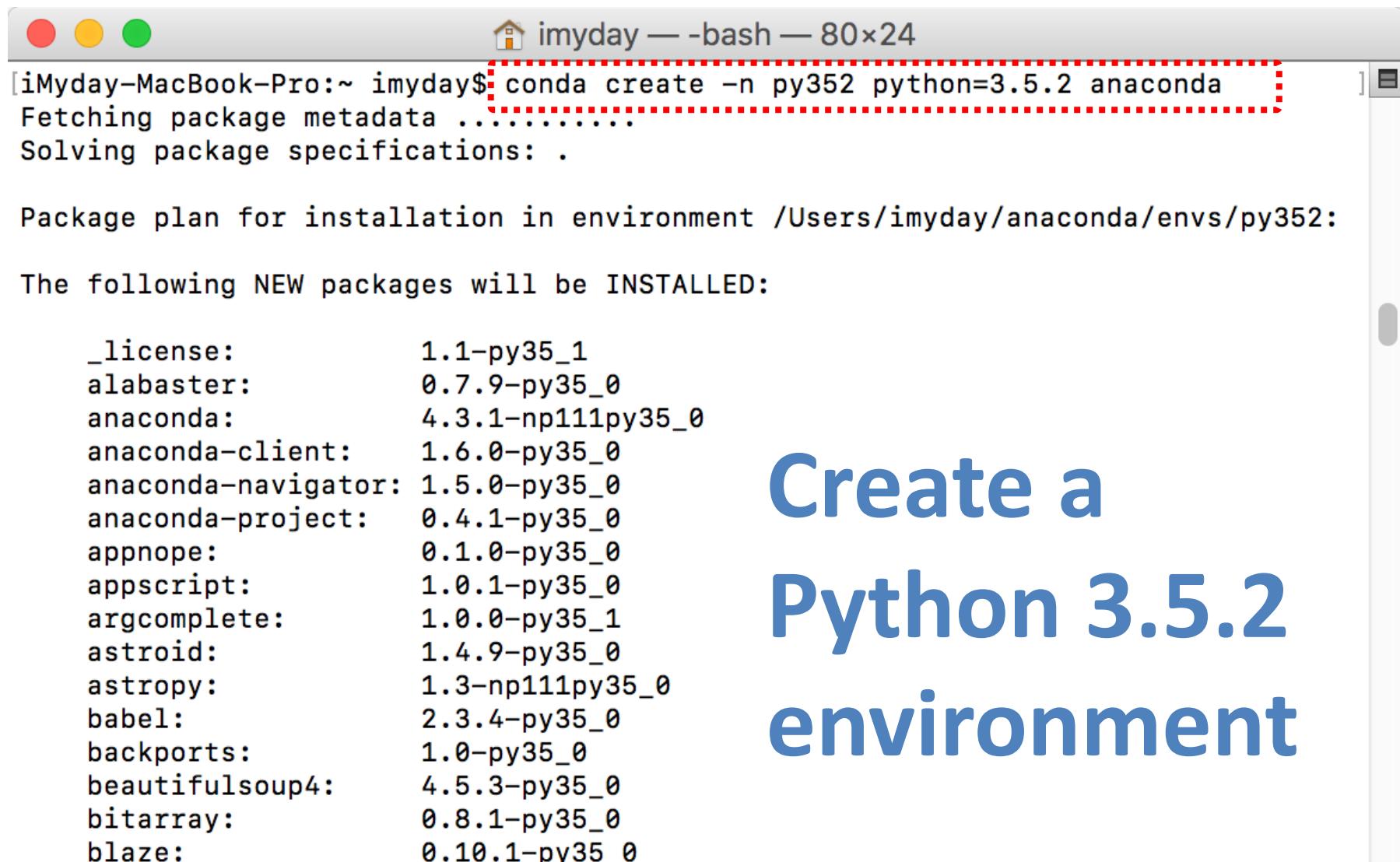
```
[(py35) iMyday-MacBook-Pro:~ imyday$ source deactivate py35
```

```
[iMyday-MacBook-Pro:~ imyday$ conda info --envs  
# conda environments:  
#
```

```
py27          /Users/imyday/anaconda/envs/py27  
py35          /Users/imyday/anaconda/envs/py35  
root          * /Users/imyday/anaconda
```

source deactivate py35

```
conda create -n py352 python=3.5.2 anaconda
```



A screenshot of a macOS terminal window titled "imyday — -bash — 80x24". The window shows the command "conda create -n py352 python=3.5.2 anaconda" being run. The output of the command is displayed below, including package metadata fetching, solving specifications, and a package plan for installation. A red dashed box highlights the command input area.

```
[iMyday-MacBook-Pro:~ imyday$ conda create -n py352 python=3.5.2 anaconda
Fetching package metadata .....
Solving package specifications: .

Package plan for installation in environment /Users/imyday/anaconda/envs/py352:

The following NEW packages will be INSTALLED:

_license:          1.1-py35_1
alabaster:         0.7.9-py35_0
anaconda:          4.3.1-np111py35_0
anaconda-client:   1.6.0-py35_0
anaconda-navigator: 1.5.0-py35_0
anaconda-project:  0.4.1-py35_0
appnope:           0.1.0-py35_0
appscript:          1.0.1-py35_0
argcomplete:        1.0.0-py35_1
astroid:            1.4.9-py35_0
astropy:            1.3-np111py35_0
babel:              2.3.4-py35_0
backports:          1.0-py35_0
beautifulsoup4:     4.5.3-py35_0
bitarray:           0.8.1-py35_0
blaze:               0.10.1-py35_0
```

Create a Python 3.5.2 environment

```
conda create -n py352 python=3.5.2 anaconda
```

```
i myday — -bash — 80x24
pyopenssl-16.2 100% | #####| Time: 0:00:00 1.40 MB/s
scikit-image-0 100% | #####| Time: 0:00:17 1.05 MB/s
seaborn-0.7.1- 100% | #####| Time: 0:00:00 1.05 MB/s
statsmodels-0. 100% | #####| Time: 0:00:04 1.06 MB/s
anaconda-navig 100% | #####| Time: 0:00:04 1.05 MB/s
blaze-0.10.1-p 100% | #####| Time: 0:00:00 1.05 MB/s
ipykernel-4.5. 100% | #####| Time: 0:00:00 1.21 MB/s
nbconvert-4.2. 100% | #####| Time: 0:00:00 1.22 MB/s
jupyter_consol 100% | #####| Time: 0:00:00 2.74 MB/s
notebook-4.3.1 100% | #####| Time: 0:00:05 1.05 MB/s
qtconsole-4.2. 100% | #####| Time: 0:00:00 1.03 MB/s
spyder-3.1.2-p 100% | #####| Time: 0:00:03 1.06 MB/s
widgetsnbexten 100% | #####| Time: 0:00:01 1.05 MB/s
ipywidgets-5.2 100% | #####| Time: 0:00:00 1.08 MB/s
jupyter-1.0.0- 100% | #####| Time: 0:00:00 2.53 MB/s
anaconda-4.3.1 100% | #####| Time: 0:00:00 4.49 MB/s
#
# To activate this environment, use:
# > source activate py352
#
# To deactivate this environment, use:
# > source deactivate py352
#
```

```
source activate py352
```

conda info --envs

```
iMyday-MacBook-Pro:~ imyday$ conda info --envs
# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
py352         /Users/imyday/anaconda/envs/py352
root          * /Users/imyday/anaconda

[iMyday-MacBook-Pro:~ imyday$ python --version
Python 3.6.0 :: Anaconda 4.3.1 (x86_64)
[iMyday-MacBook-Pro:~ imyday$ source activate py352
(py352) iMyday-MacBook-Pro:~ imyday$ conda info --envs
# conda environments:
#
py27          /Users/imyday/anaconda/envs/py27
py35          /Users/imyday/anaconda/envs/py35
py352         * /Users/imyday/anaconda/envs/py352
root          /Users/imyday/anaconda

(py352) iMyday-MacBook-Pro:~ imyday$ python --version
Python 3.5.2 :: Anaconda 4.3.1 (x86_64)
(py352) iMyday-MacBook-Pro:~ imyday$ 
```

TensorFlow

```
conda info --envs
```

```
conda --version
```

```
python --version
```

```
conda list
```

```
conda create -n tensorflow python=3.5
```

```
source activate tensorflow
```

```
activate tensorflow
```

```
sudo pip install keras
```

```
pip install keras
```

```
pip install tensorflow
```

```
pip install ipython[all]
```

Source: <https://github.com/martin-gorner/tensorflow-mnist-tutorial/blob/master/INSTALL.txt>

Source: <http://deeplearning.net/software/theano/install.html#anaconda>

pip install tensorflow

```
bash-3.2$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.1.0-cp36-cp36m-macosx_10_11_x86_64.whl (31.3MB)
    100% |████████████████████████████████| 31.3MB 23kB/s
Requirement already satisfied: wheel>=0.26 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: six>=1.10.0 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Collecting protobuf>=3.2.0 (from tensorflow)
  Downloading protobuf-3.2.0-py2.py3-none-any.whl (360kB)
    100% |████████████████████████████████| 368kB 453kB/s
Requirement already satisfied: werkzeug>=0.11.10 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: numpy>=1.11.0 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: setuptools in ./anaconda/lib/python3.6/site-packages/setuptools-27.2.0-py3.6.
egg (from protobuf>=3.2.0->tensorflow)
Installing collected packages: protobuf, tensorflow
Successfully installed protobuf-3.2.0 tensorflow-1.1.0
bash-3.2$
```

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?

X_1

X_2

X_1^2

X_2^2

$X_1 X_2$

X_1^3

X_2^3



3 HIDDEN LAYERS



2 neurons



2 neurons



2 neurons

4 neurons

4 neurons

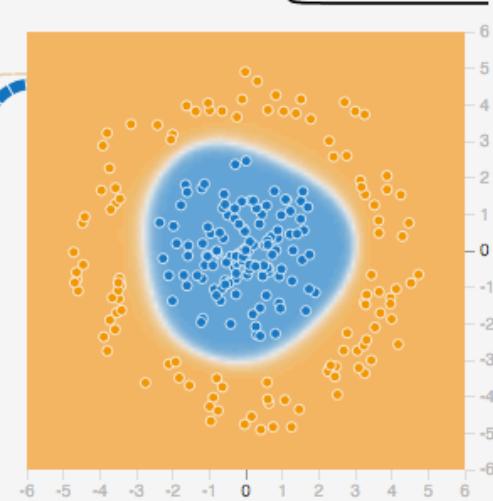
4 neurons

The outputs are mixed with varying **weights**, shown by the thickness of the lines.

This is the output from one **neuron**. Hover to see it larger.

OUTPUT

Test loss 0.000
Training loss 0.000



TensorBoard

TensorBoard EVENTS IMAGES GRAPHS **HISTOGRAMS**

Fit to screen
 Download PNG

Run **train** (1)

Session runs (0)

Upload

Color Structure Device
color: same substructure gray; unique substructure

Graph (* = expandable)
 Namespace*
 OpNode
 Unconnected series*
 Connected series*
 Constant
 Summary
 Dataflow edge
 Control dependency edge
 Reference edge

Main Graph

The Main Graph visualization shows a neural network architecture. It consists of several nodes: **input**, **layer1**, **layer2**, **cross_entropy**, and **accuracy**. The **input** node has two outgoing edges labeled "train" to **layer1** and **cross_entropy**. The **layer1** node has two outgoing edges labeled "train" to **layer2** and **cross_entropy**. The **layer2** node has one outgoing edge labeled "train" to **cross_entropy**. The **cross_entropy** node has one outgoing edge labeled "train" to **accuracy**. The **accuracy** node has one outgoing edge labeled "train" to a separate **train** node. A legend on the right, titled "Auxiliary nodes", shows the **train** node with its internal structure: it receives "train" inputs from **cross_entropy**, **input**, **predictions**, and **layer2**, and produces a "train" output to **accuracy**.

Auxiliary nodes

Try your first TensorFlow

```
$ python
```

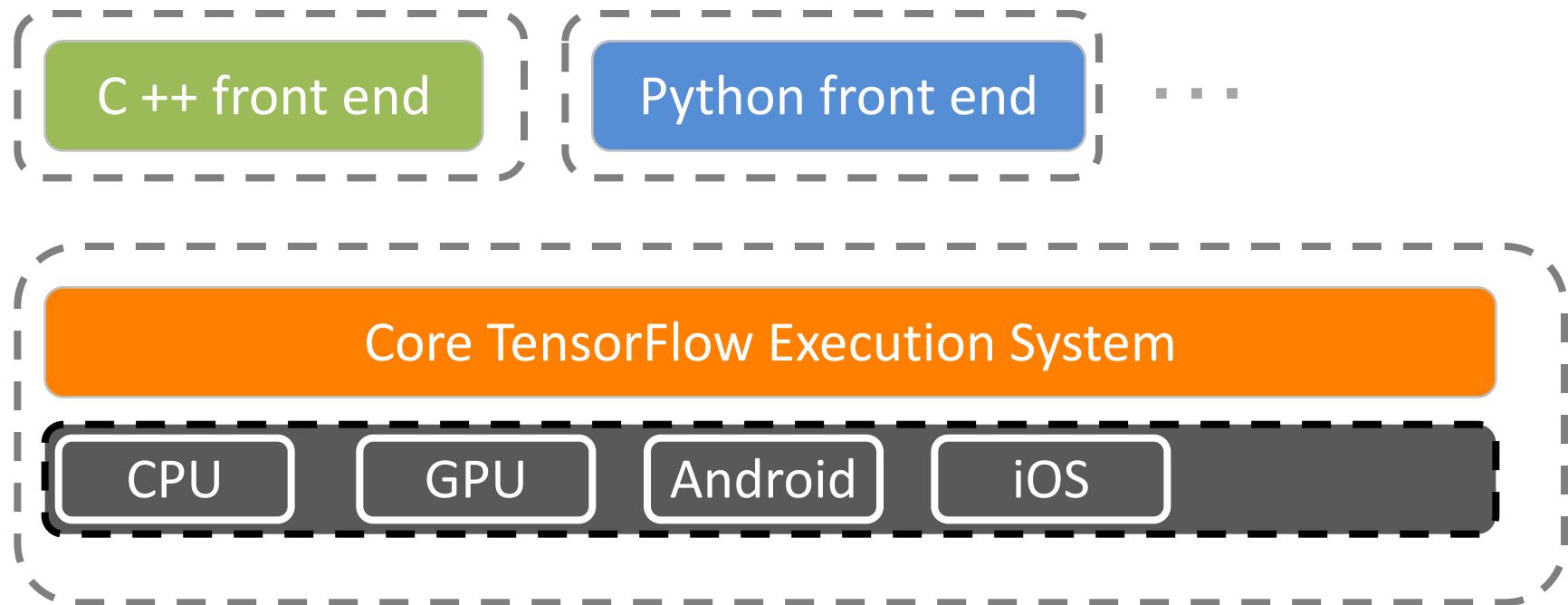
```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
Hello, TensorFlow!
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
>>>
```

Try your first TensorFlow

```
$ python
```

```
>>> import tensorflow as tf
>>> hello = tf.constant('Hello, TensorFlow!')
>>> sess = tf.Session()
>>> sess.run(hello)
'Hello, TensorFlow!'
>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> sess.run(a+b)
42
>>>
```

Architecture of TensorFlow



TensorFlow and Deep Learning

TensorFlow and Deep Learning

1 Overview

Preparation: Install TensorFlow, get the sample code

Theory: train a neural network

Theory: a 1-layer neural network

Theory: gradient descent

Lab: let's jump into the code

Lab: adding layers

Lab: special care for deep networks

Lab: learning rate decay

Lab: dropout, overfitting

Theory: convolutional networks

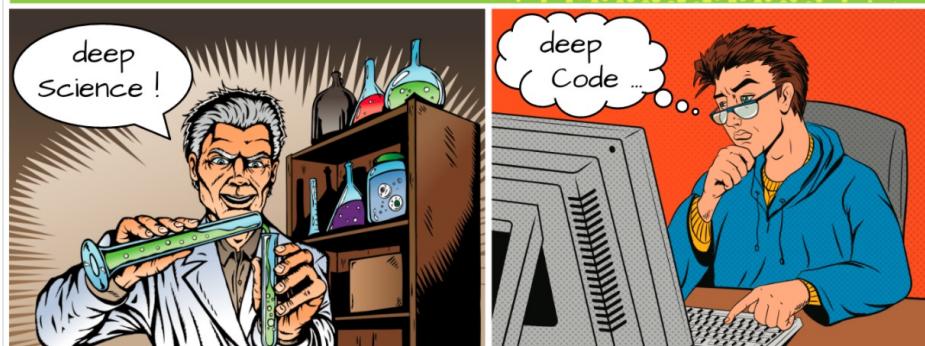
Did you find a mistake? [Please file a bug.](#)

← TensorFlow and deep learning, without a PhD

⌚ 149 min remaining

1. Overview

>TensorFlow and deep learning_ without a PhD



In this codelab, you will learn how to build and train a neural network that recognises handwritten digits. Along the way, as you enhance your neural network to achieve 99% accuracy, you will also discover the tools of the trade that deep learning professionals use to train their models efficiently.

This codelab uses the [MNIST](#) dataset, a collection of 60,000 labeled digits that has kept generations of PhDs busy for almost two decades. You will solve the problem with less than 100 lines of Python / TensorFlow code.

What you'll learn



TensorFlow MNIST Tutorial



Features Business Explore Pricing

This repository

Search

Sign in or Sign up

martin-gorner / tensorflow-mnist-tutorial

Watch

50

Star

489

Fork

204

Code

Issues 6

Pull requests 2

Projects 0

Pulse

Graphs

Sample code for "Tensorflow and deep learning, without a PhD" presentation and code lab.

102 commits

1 branch

0 releases

4 contributors

Apache-2.0

Branch: master ▾

New pull request

Find file

Clone or download ▾



martin-gorner committed on GitHub Update INSTALL.txt ...

Latest commit ed331aa 25 days ago

mlengine

added example using the Tensorflow high level layers API

26 days ago

.gitignore

small bug fix in batch norm

6 months ago

CONTRIBUTING.md

initial commit 2

4 months ago

INSTALL.txt

Update INSTALL.txt

25 days ago

LICENSE

Initial commit

a year ago

README.md

better image URL

3 months ago

mnist_1.0_softmax.py

global_variables_initializer used everywhere instead of inirialize_al...

2 months ago

mnist_2.0_five_layers_sigmoid.py

Fix spacing in the network structure comment

a month ago

mnist_2.1_five_layers_relu_lrdecay...

Fix spacing in the network structure comment

a month ago

TensorFlow and Deep Learning

- What is a neural network and how to train it
- How to build a basic 1-layer neural network using TensorFlow
- How to add more layers
- Training tips and tricks: overfitting, dropout, learning rate decay ...
- How to troubleshoot deep neural networks
- How to build convolutional networks

TensorFlow MNIST Tutorial

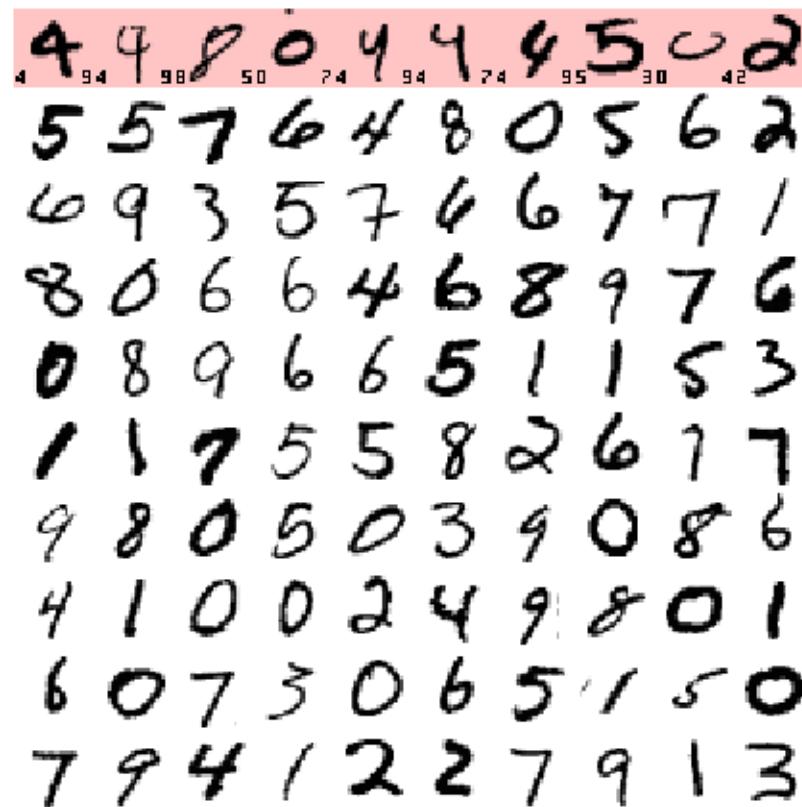
```
git clone https://github.com/martin-gorner/tensorflow-mnist-tutorial.git
```

```
cd tensorflow-mnist-tutorial
```

```
python3 mnist_1.0_softmax.py
```

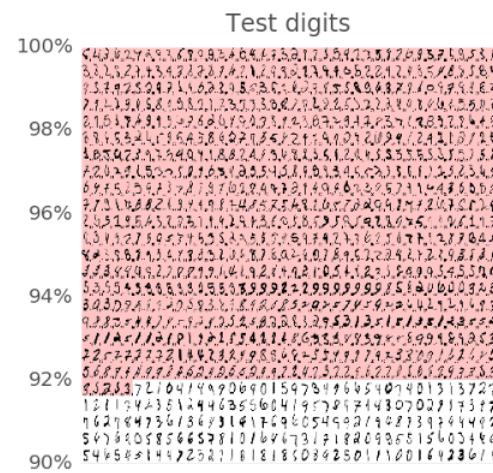
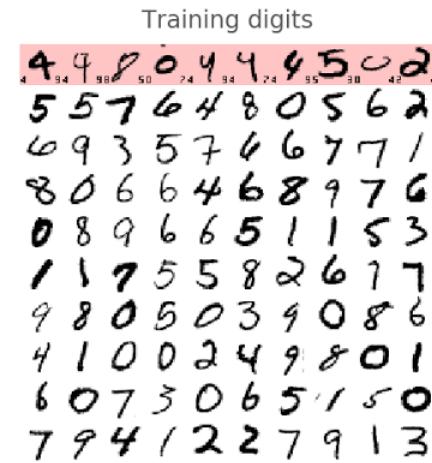
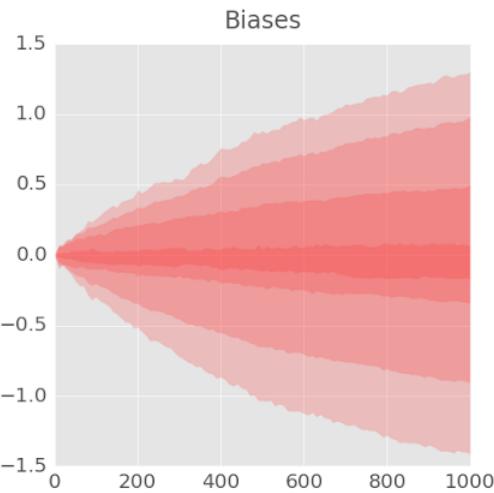
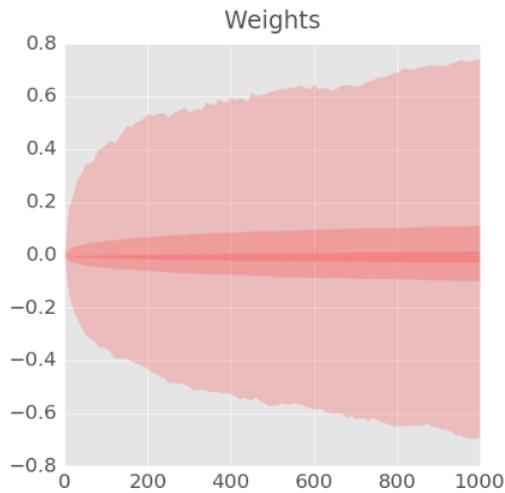
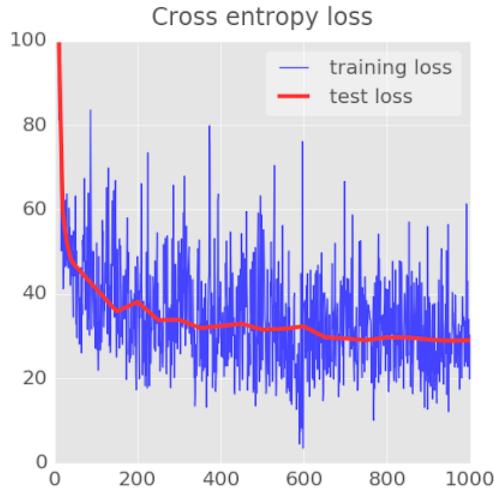
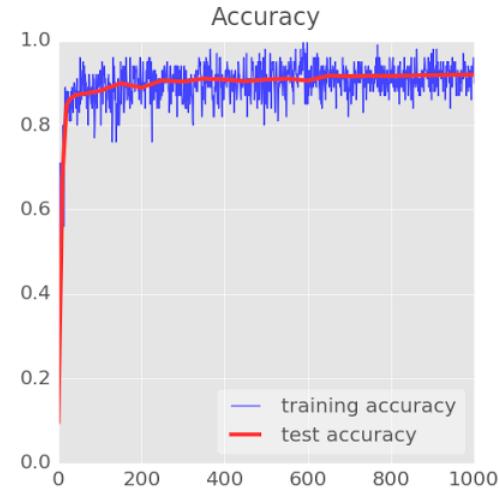
MNIST dataset: 60,000 labeled digits

Training digits



```
cd tensorflow-mnist-tutorial
```

```
python3 mnist_1.0_softmax.py
```

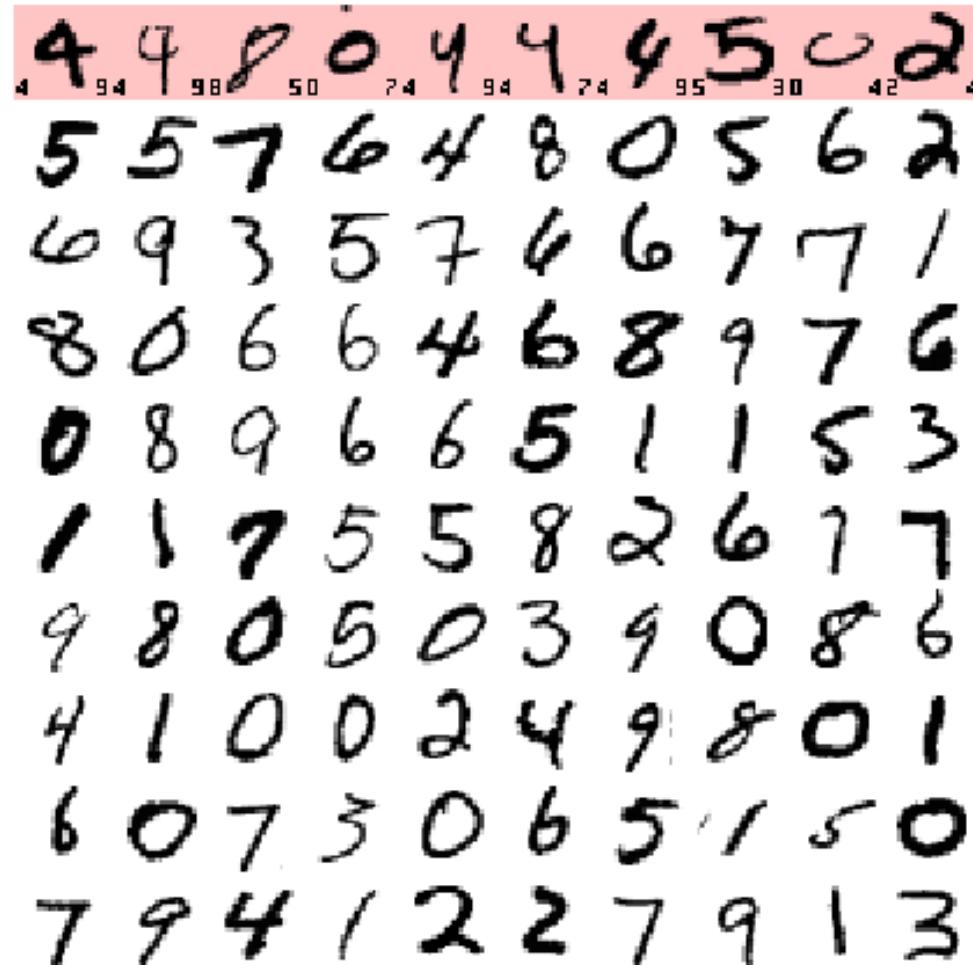


Train a Neural Network

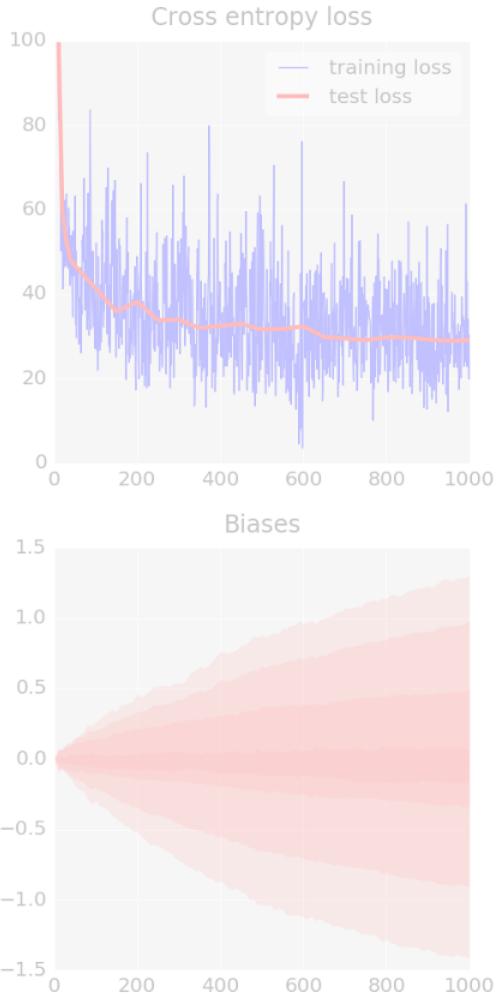
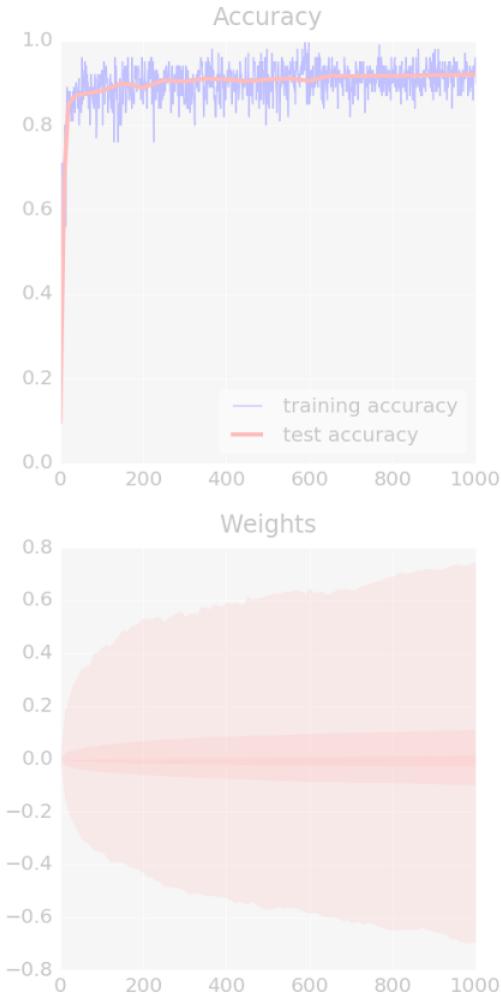
Training digits
updates to **weights** and **biases** =>
better recognition (loop)

Training digits

Training digits



Training digits

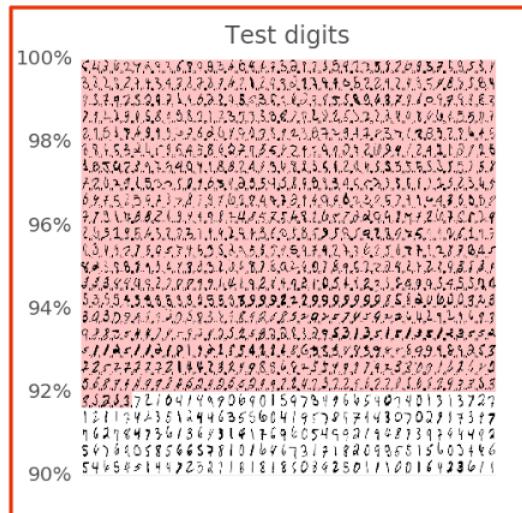
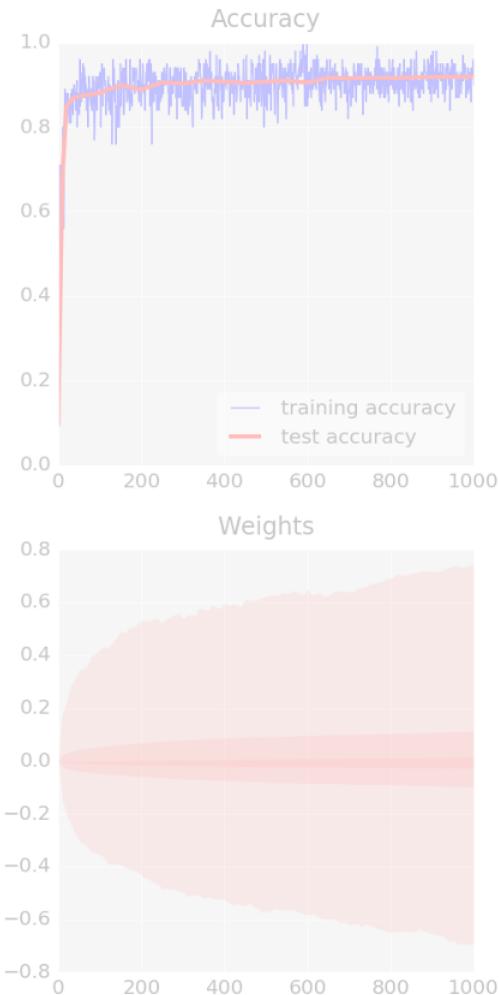


Test digits

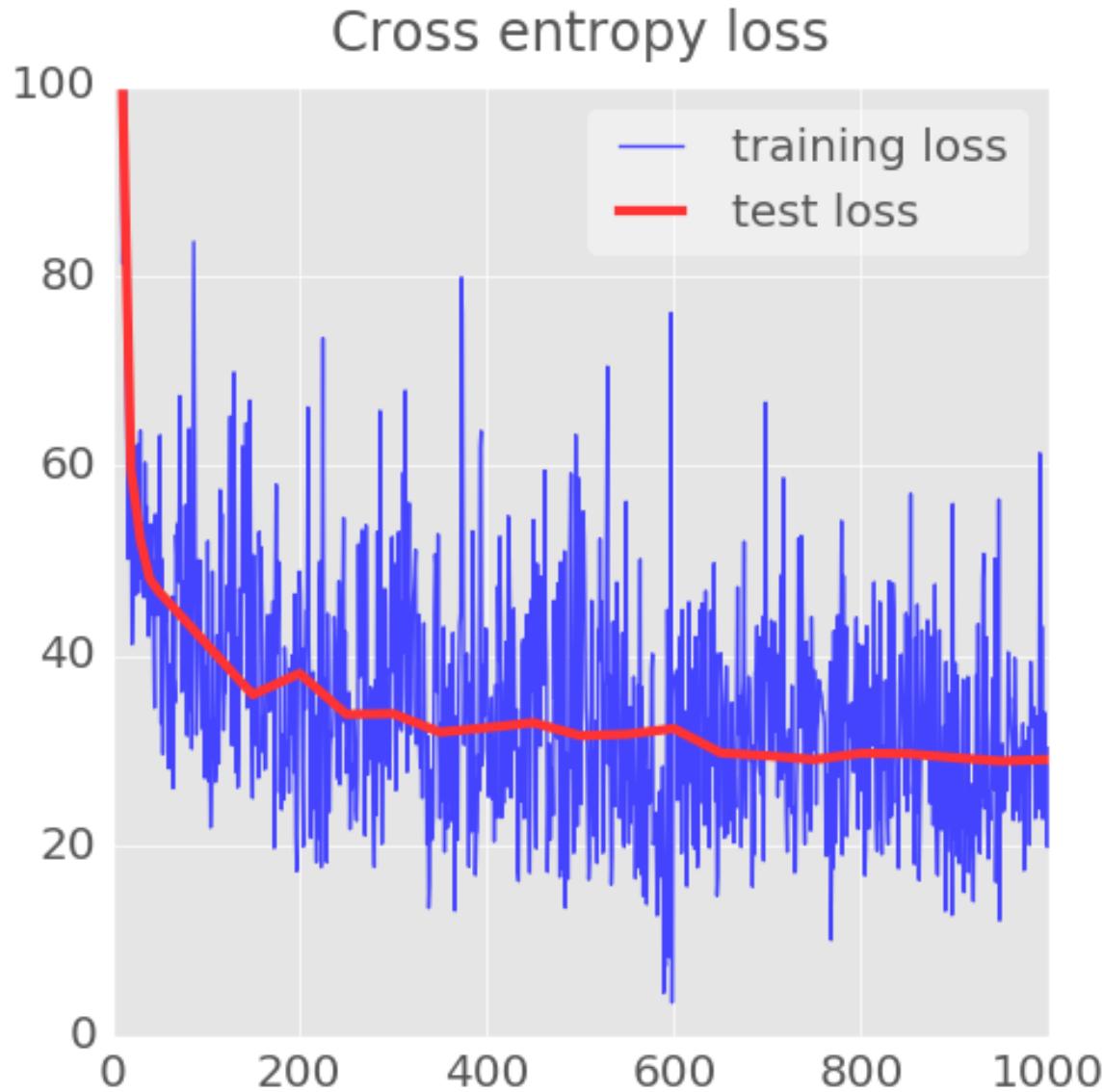


Source: <https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2>

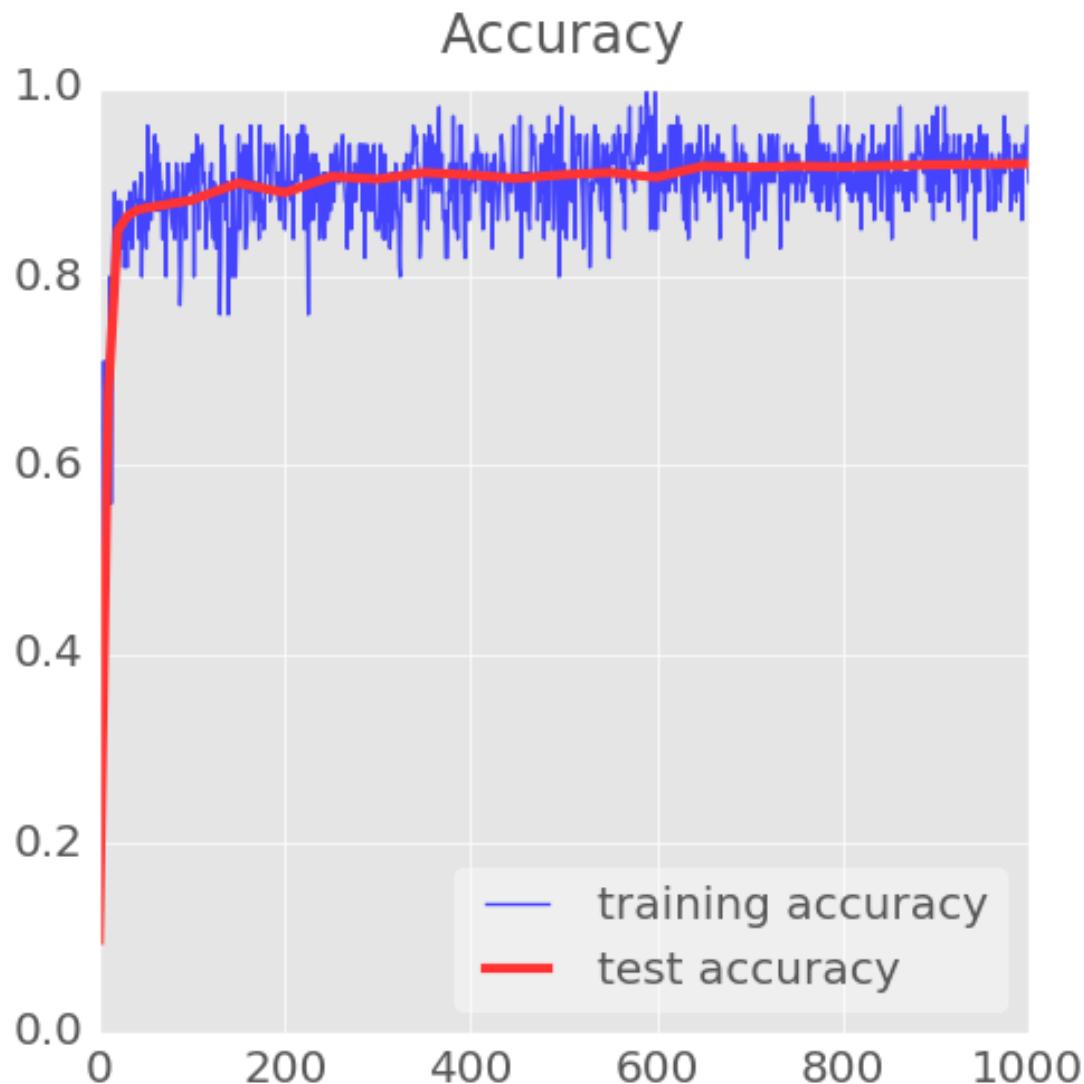
Test digits



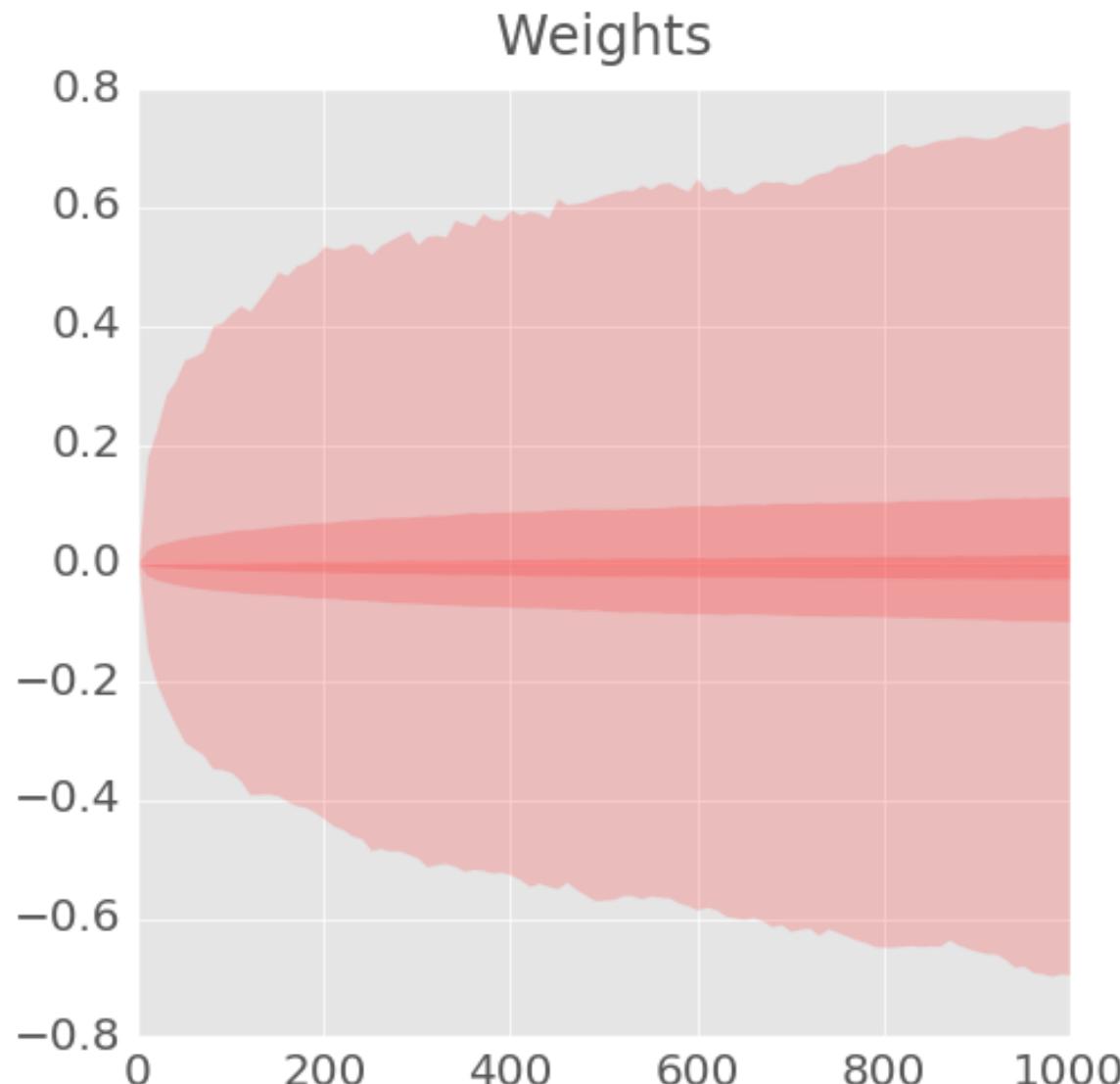
Cross entropy loss



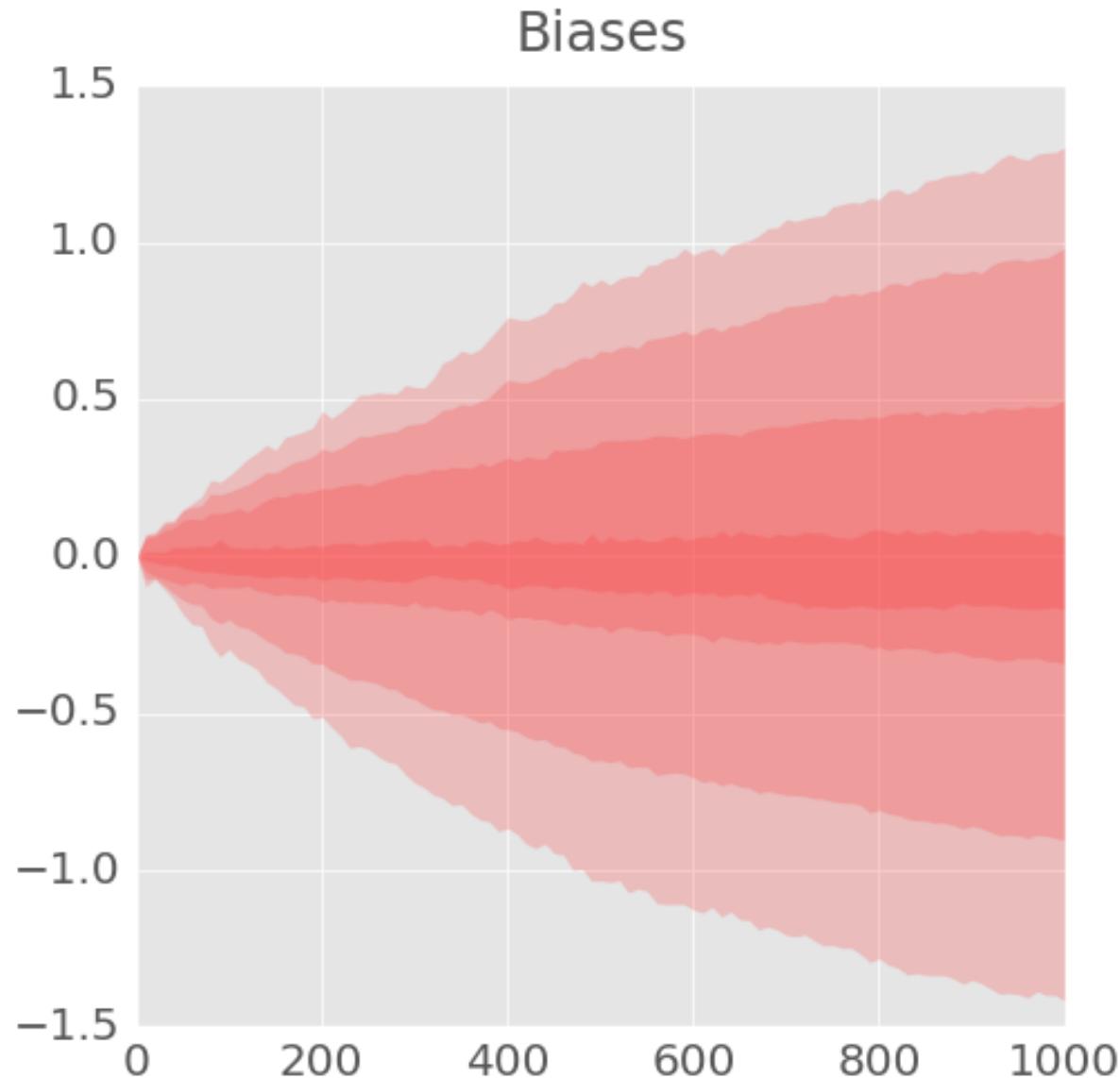
Accuracy



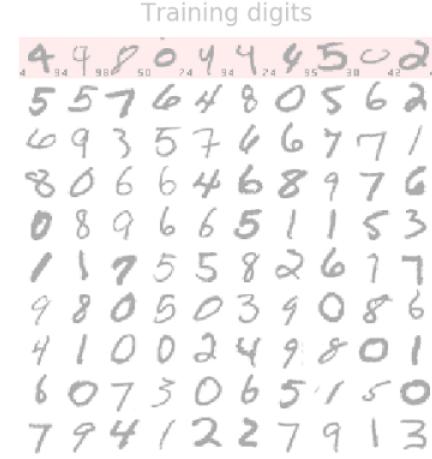
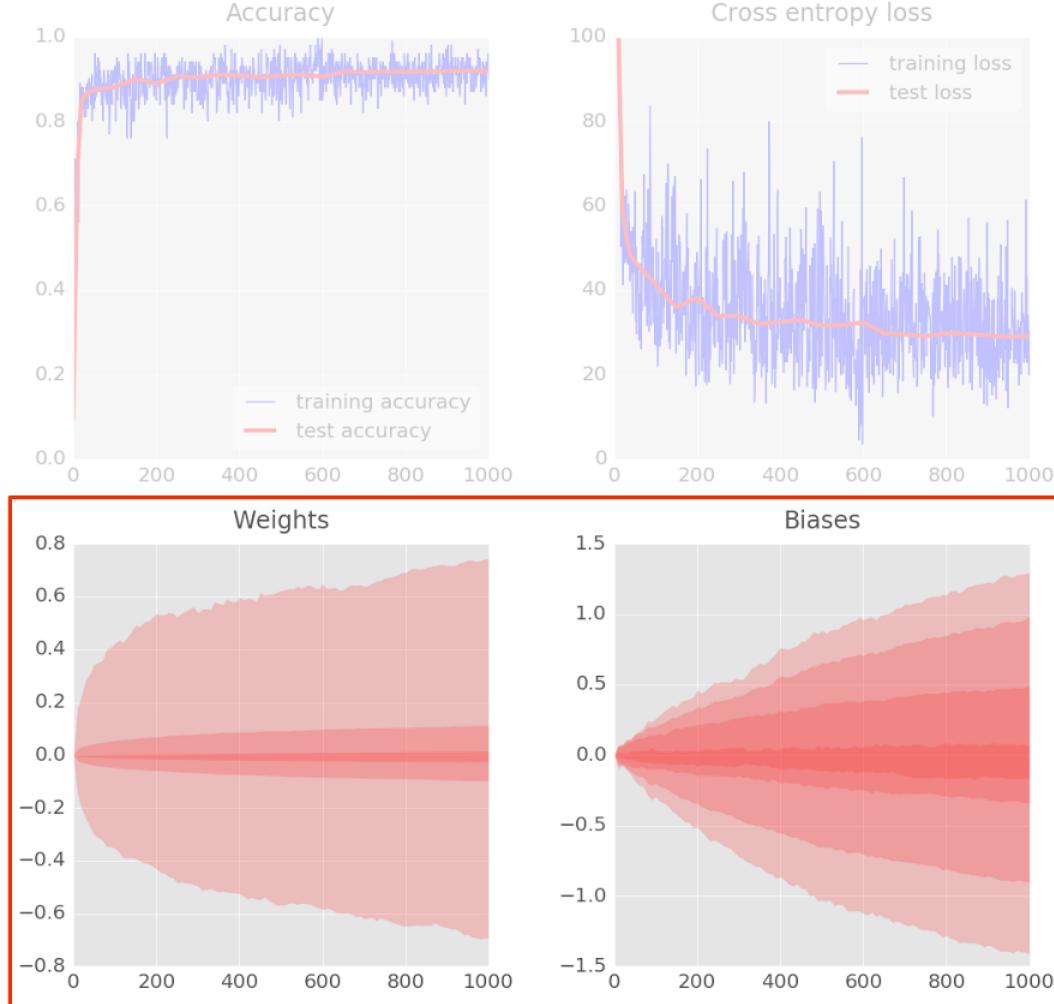
Weights



Biases



Weights and Biases



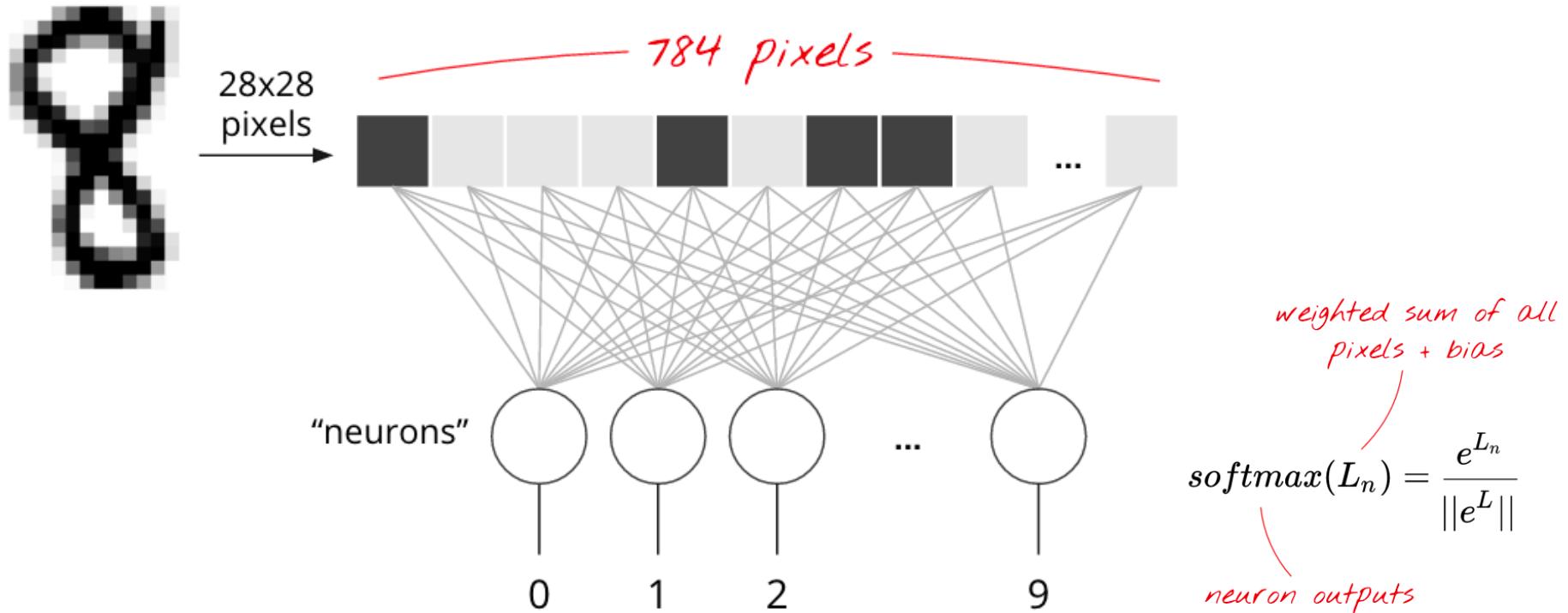
Test digits
4 4 8 0 4 4 4 5 0 2
5 5 7 6 4 8 0 5 6 2
6 9 3 5 7 4 6 7 7 1
8 0 6 6 4 6 8 9 7 6
0 8 9 6 6 5 1 1 5 3
1 1 7 5 5 8 2 6 1 7
9 8 0 5 0 3 9 0 8 6
4 1 0 0 2 4 9 8 0 1
6 0 7 3 0 6 5 1 5 0
7 9 4 1 2 2 7 9 1 3

Cookbook

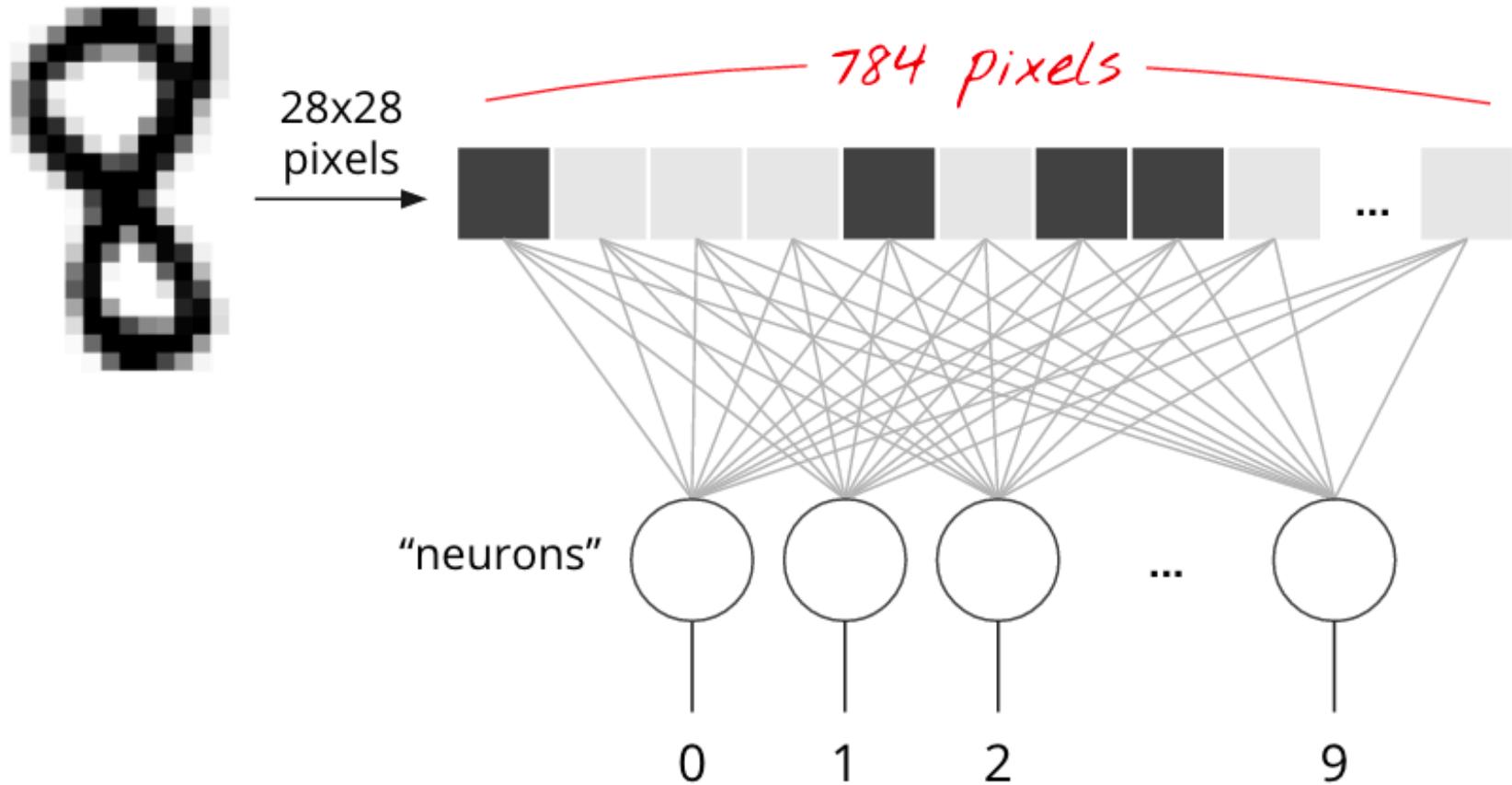
Softmax
Cross-entropy
Mini-batch



Very Simple Model: Softmax Classification



Very Simple Model: Softmax Classification



Very Simple Model: Softmax Classification

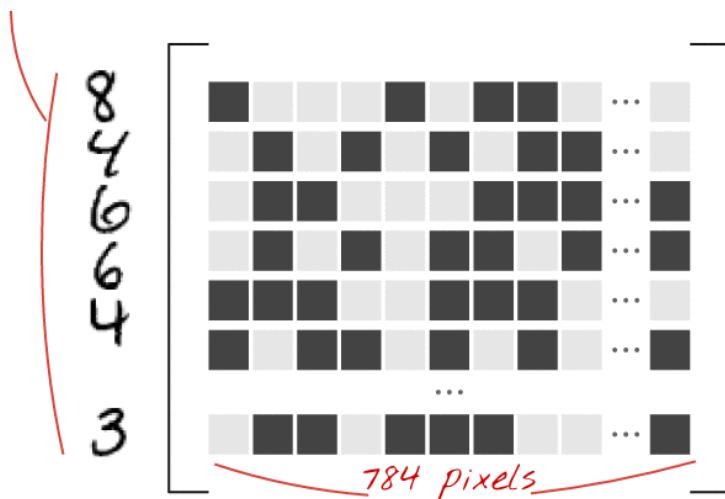
weighted sum of all
pixels + bias

$$\text{softmax}(L_n) = \frac{e^{L_n}}{\|e^L\|}$$

neuron outputs

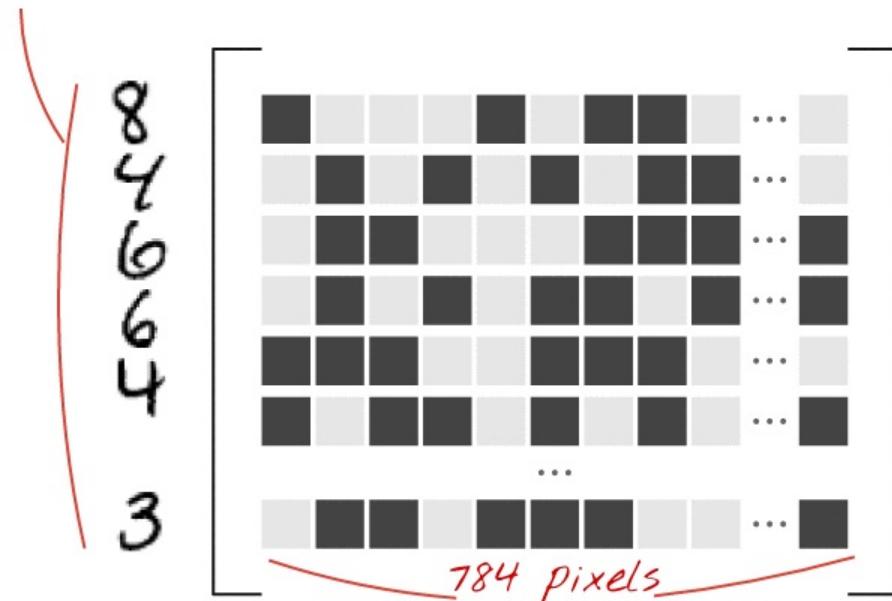
In Matrix notation, 100 images at a time

*X: 100 images,
one per line,
flattened*



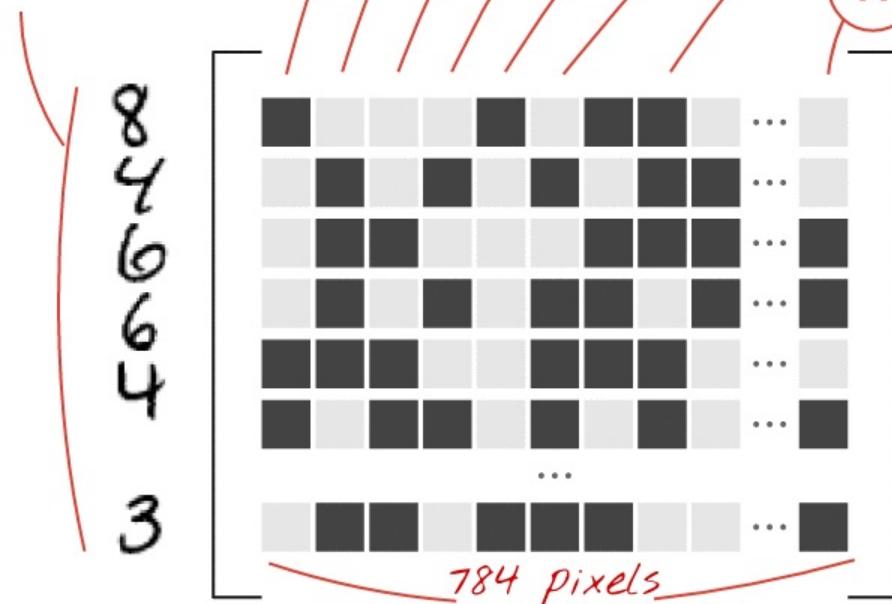
10 columns									
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$...	$W_{0,9}$				
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$...	$W_{1,9}$				
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$...	$W_{2,9}$				
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$...	$W_{3,9}$				
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$...	$W_{4,9}$				
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$...	$W_{5,9}$				
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$...	$W_{6,9}$				
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$...	$W_{7,9}$				
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$...	$W_{8,9}$				
...									
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	$W_{783,3}$...	$W_{783,9}$				

*X: 100 images,
one per line,
flattened*



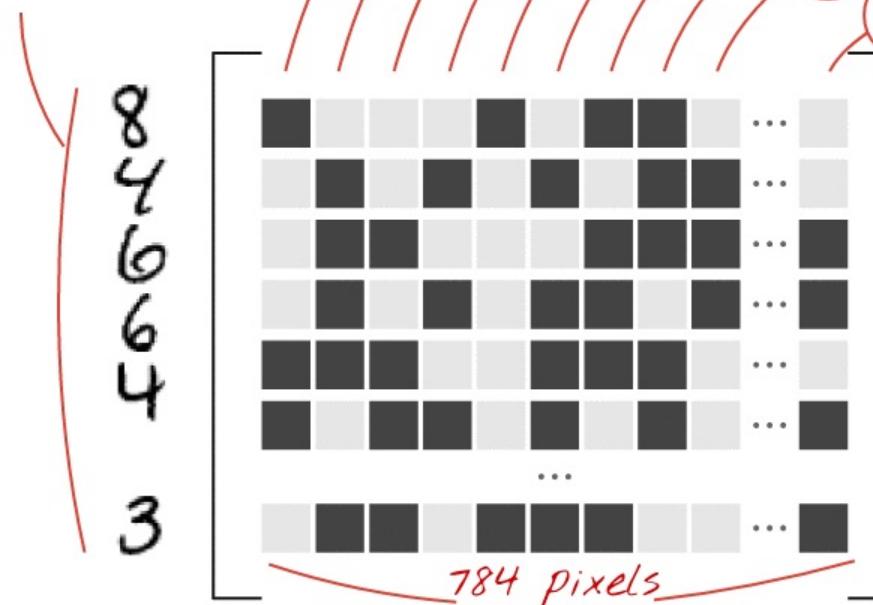
10 columns									
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$...	$W_{0,9}$				
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$...	$W_{1,9}$				
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$...	$W_{2,9}$				
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$...	$W_{3,9}$				
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$...	$W_{4,9}$				
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$...	$W_{5,9}$				
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$...	$W_{6,9}$				
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$...	$W_{7,9}$				
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$...	$W_{8,9}$				
...									
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	$W_{783,3}$...	$W_{783,9}$				

*X: 100 images,
one per line,
flattened*



10 columns									
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	\dots	$W_{0,9}$				
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	\dots	$W_{1,9}$				
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	\dots	$W_{2,9}$				
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	\dots	$W_{3,9}$				
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	\dots	$W_{4,9}$				
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	\dots	$W_{5,9}$				
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	\dots	$W_{6,9}$				
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	\dots	$W_{7,9}$				
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	\dots	$W_{8,9}$				
\dots									
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	$W_{783,3}$	\dots	$W_{783,9}$				

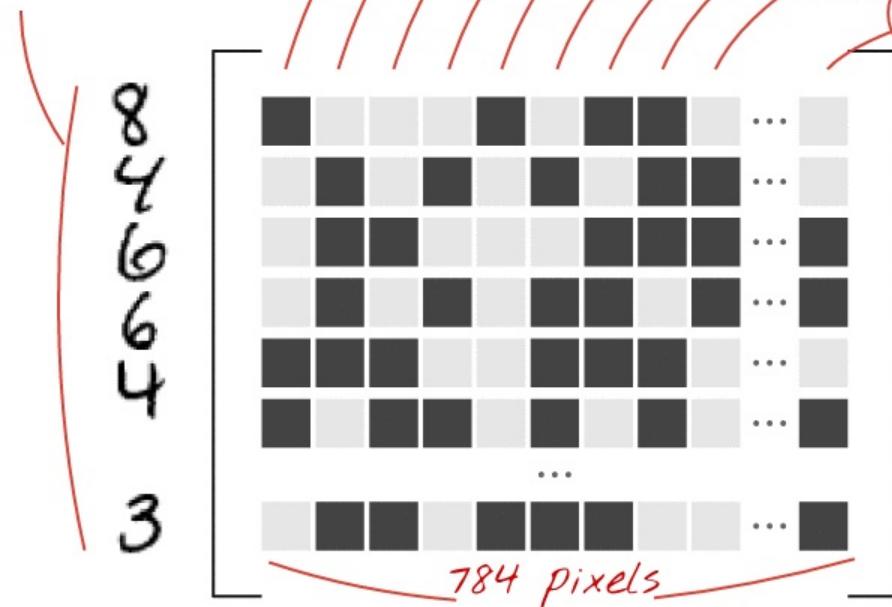
X : 100 images,
one per line,
flattened



10 columns									
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	\dots	$W_{0,9}$				
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	\dots	$W_{1,9}$				
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	\dots	$W_{2,9}$				
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	\dots	$W_{3,9}$				
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	\dots	$W_{4,9}$				
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	\dots	$W_{5,9}$				
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	\dots	$W_{6,9}$				
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	\dots	$W_{7,9}$				
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	\dots	$W_{8,9}$				
\dots									
	$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	\dots	$W_{783,9}$				

$L_{0,0}$ $L_{0,1}$

*X: 100 images,
one per line,
flattened*



10 columns									
$W_{0,0}$	$W_{0,1}$	$W_{0,2}$	$W_{0,3}$	\dots	$W_{0,9}$				
$W_{1,0}$	$W_{1,1}$	$W_{1,2}$	$W_{1,3}$	\dots	$W_{1,9}$				
$W_{2,0}$	$W_{2,1}$	$W_{2,2}$	$W_{2,3}$	\dots	$W_{2,9}$				
$W_{3,0}$	$W_{3,1}$	$W_{3,2}$	$W_{3,3}$	\dots	$W_{3,9}$				
$W_{4,0}$	$W_{4,1}$	$W_{4,2}$	$W_{4,3}$	\dots	$W_{4,9}$				
$W_{5,0}$	$W_{5,1}$	$W_{5,2}$	$W_{5,3}$	\dots	$W_{5,9}$				
$W_{6,0}$	$W_{6,1}$	$W_{6,2}$	$W_{6,3}$	\dots	$W_{6,9}$				
$W_{7,0}$	$W_{7,1}$	$W_{7,2}$	$W_{7,3}$	\dots	$W_{7,9}$				
$W_{8,0}$	$W_{8,1}$	$W_{8,2}$	$W_{8,3}$	\dots	$W_{8,9}$				
\dots									
$W_{783,0}$	$W_{783,1}$	$W_{783,2}$	$W_{783,3}$	\dots	$W_{783,9}$				

$L_{0,0}$	$L_{0,1}$	$L_{0,2}$	$L_{0,3}$	\dots	$L_{0,9}$
$L_{1,0}$	$L_{1,1}$	$L_{1,2}$	$L_{1,3}$	\dots	$L_{1,9}$
$L_{2,0}$	$L_{2,1}$	$L_{2,2}$	$L_{2,3}$	\dots	$L_{2,9}$
$L_{3,0}$	$L_{3,1}$	$L_{3,2}$	$L_{3,3}$	\dots	$L_{3,9}$
$L_{4,0}$	$L_{4,1}$	$L_{4,2}$	$L_{4,3}$	\dots	$L_{4,9}$
\dots					
$L_{99,0}$	$L_{99,1}$	$L_{99,2}$	$L_{99,3}$	\dots	$L_{99,9}$

What are "weights" and "biases" ?

**How is the "cross-entropy"
computed ?**

**How exactly does the
training algorithm work ?**

$$Y = f(X)$$

Predictions

$Y[100, 10]$



Images

$X[100, 784]$



Weights

$W[784, 10]$



Biases

$b[10]$



$$Y = \text{softmax}(X \cdot W + b)$$

applied line
by line

matrix multiply

broadcast
on all lines

tensor shapes in []

Y = tf.nn.softmax(tf.matmul(X, W) + b)

TensorFlow (Python) Softmax

Predictions:

$Y[100, 10]$

tensor shapes: $X[100, 784]$ $W[784, 10]$ $b[10]$

$Y = \text{tf.nn.softmax}(\text{tf.matmul}(X, W) + b)$

matrix multiply

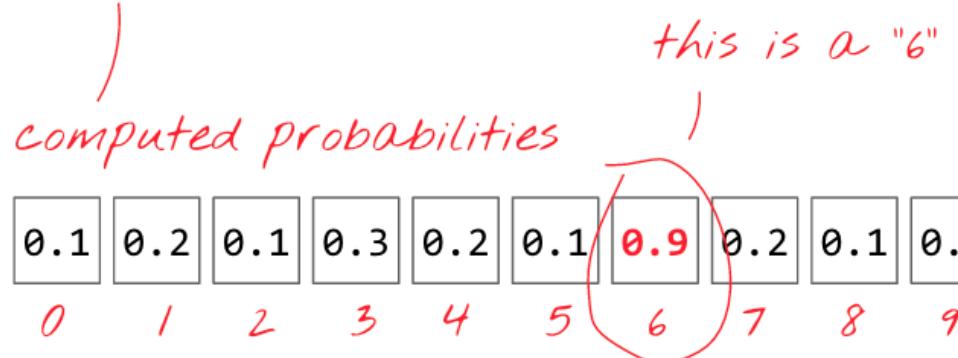
broadcast
on all lines

Cross Entropy

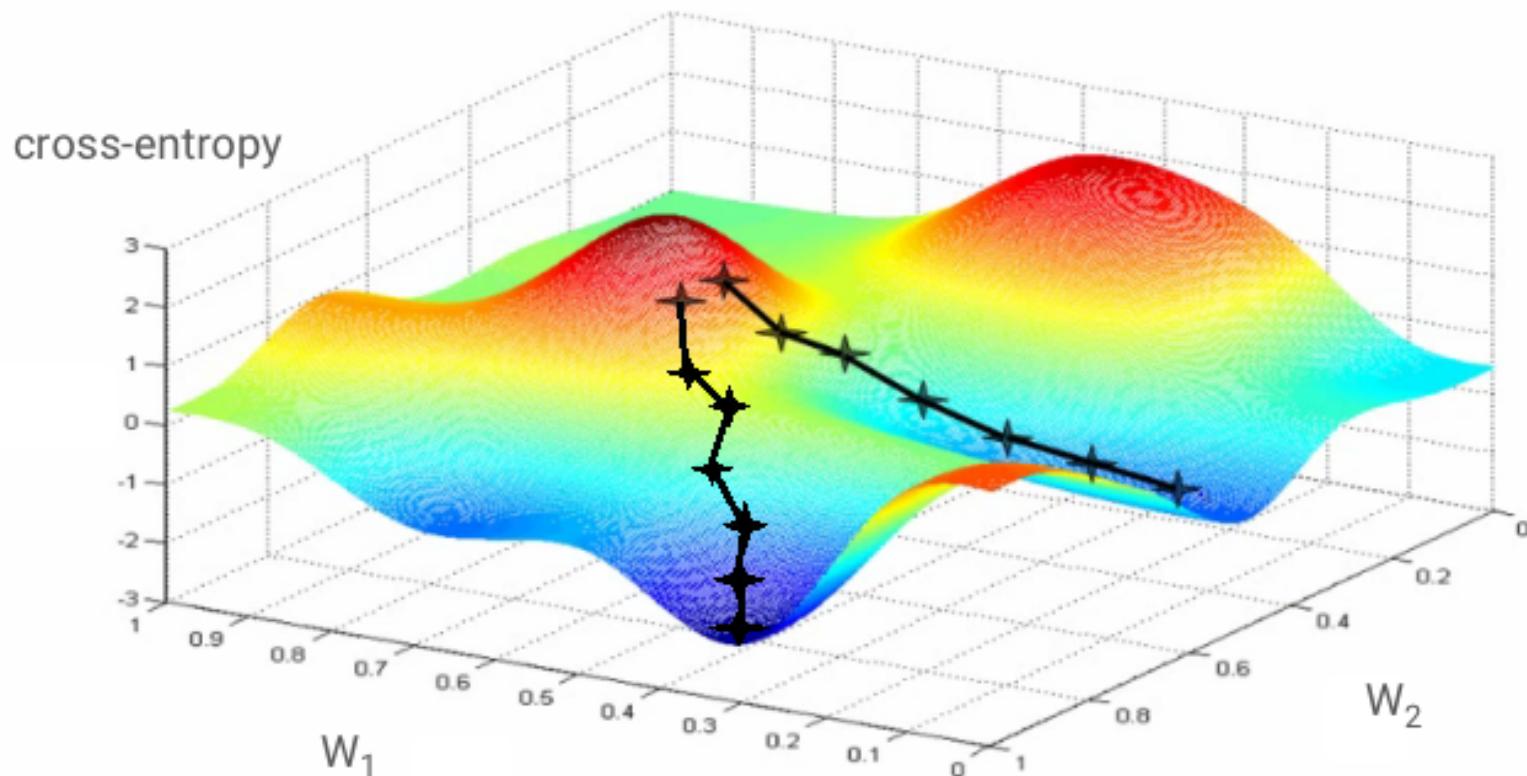
0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	0	0	0

actual probabilities, "one-hot" encoded

Cross entropy: $-\sum Y'_i \cdot \log(Y_i)$



Minimizing Cross Entropy (Minimizing Loss)



Training Loop

Training digits and labels

=> loss function

=> gradient (partial derivatives)

=> steepest descent

=> update weights and biases

**=> repeat with next mini-batch of
training images and labels**

**"mini-batches":
100 images and labels**

```
import tensorflow as tf
```

mnist_1.0_softmax.py

```
import tensorflow as tf
X = tf.placeholder(tf.float32, [None, 28, 28, 1])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))

init = tf.initialize_all_variables()
```

mnist_1.0_softmax.py

```
# model
Y = tf.nn.softmax(tf.matmul(tf.reshape(X, [-1, 784]), W) + b)
# placeholder for correct labels
Y_ = tf.placeholder(tf.float32, [None, 10])

# loss function
cross_entropy = -tf.reduce_sum(Y_ * tf.log(Y))
# % of correct answers found in batch
is_correct = tf.equal(tf.argmax(Y, 1), tf.argmax(Y_, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

mnist_1.0_softmax.py

```
sess = tf.Session()
sess.run(init)

for i in range(1000):
    # load batch of images and correct answers
    batch_X, batch_Y = mnist.train.next_batch(100)
    train_data={X: batch_X, Y_: batch_Y}

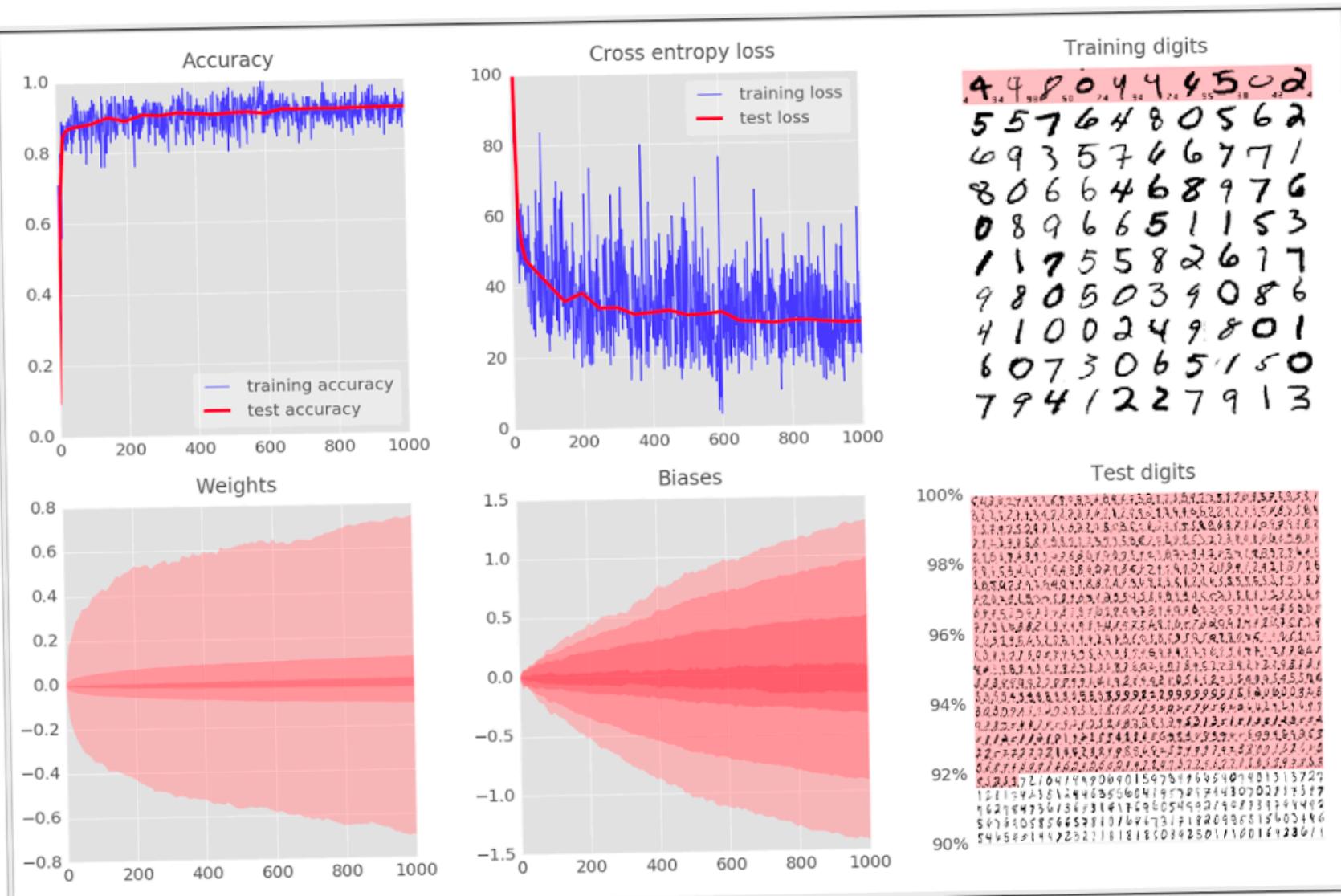
    # train
    sess.run(train_step, feed_dict=train_data)
```

mnist_1.0_softmax.py

```
# success ?
a,c = sess.run([accuracy, cross_entropy],
feed_dict=train_data)

# success on test data ?
test_data={X: mnist.test.images, Y_: mnist.test.labels}
a,c = sess.run([accuracy, cross_entropy], feed=test_data)
```

mnist_1.0_softmax.py



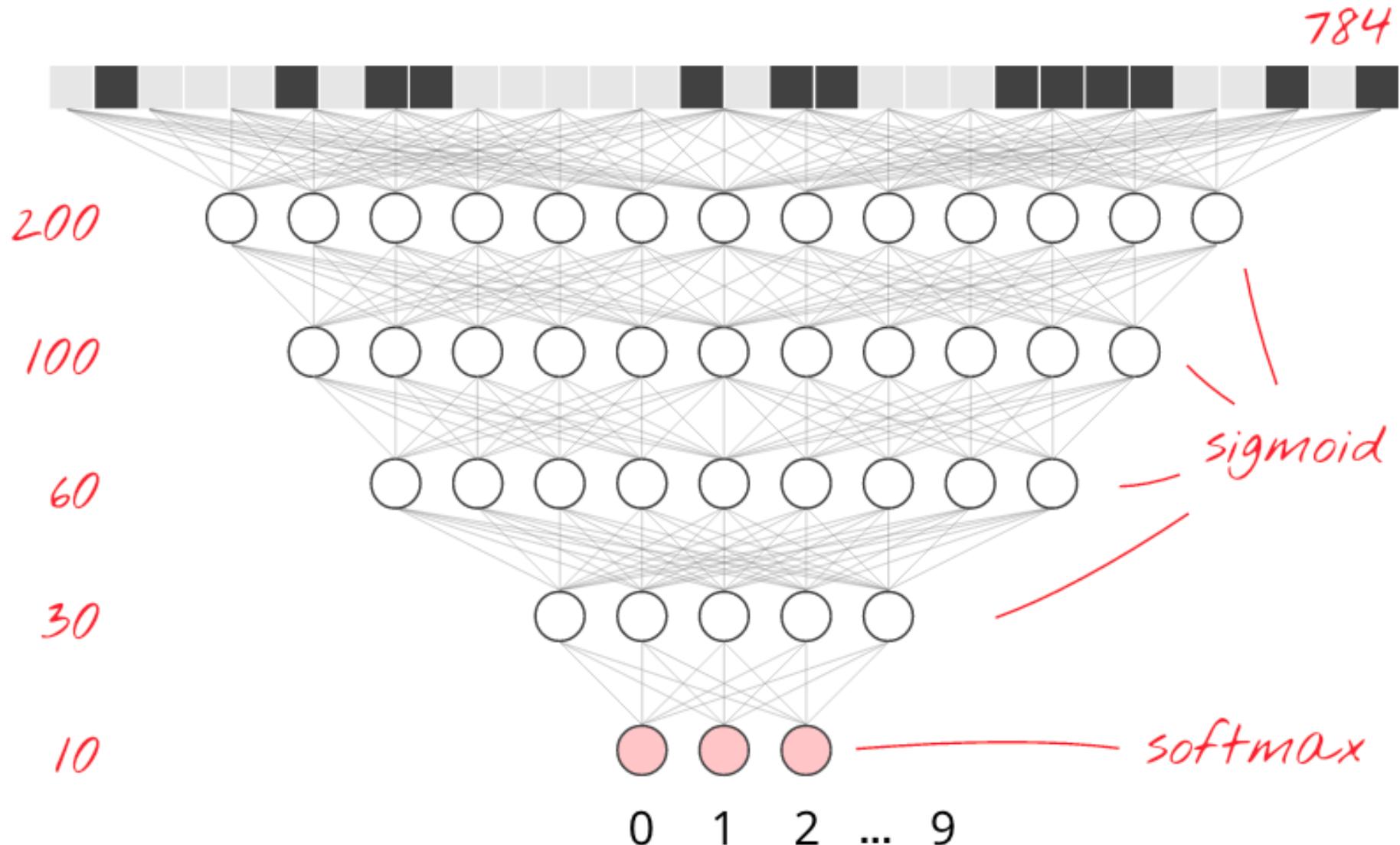
92%



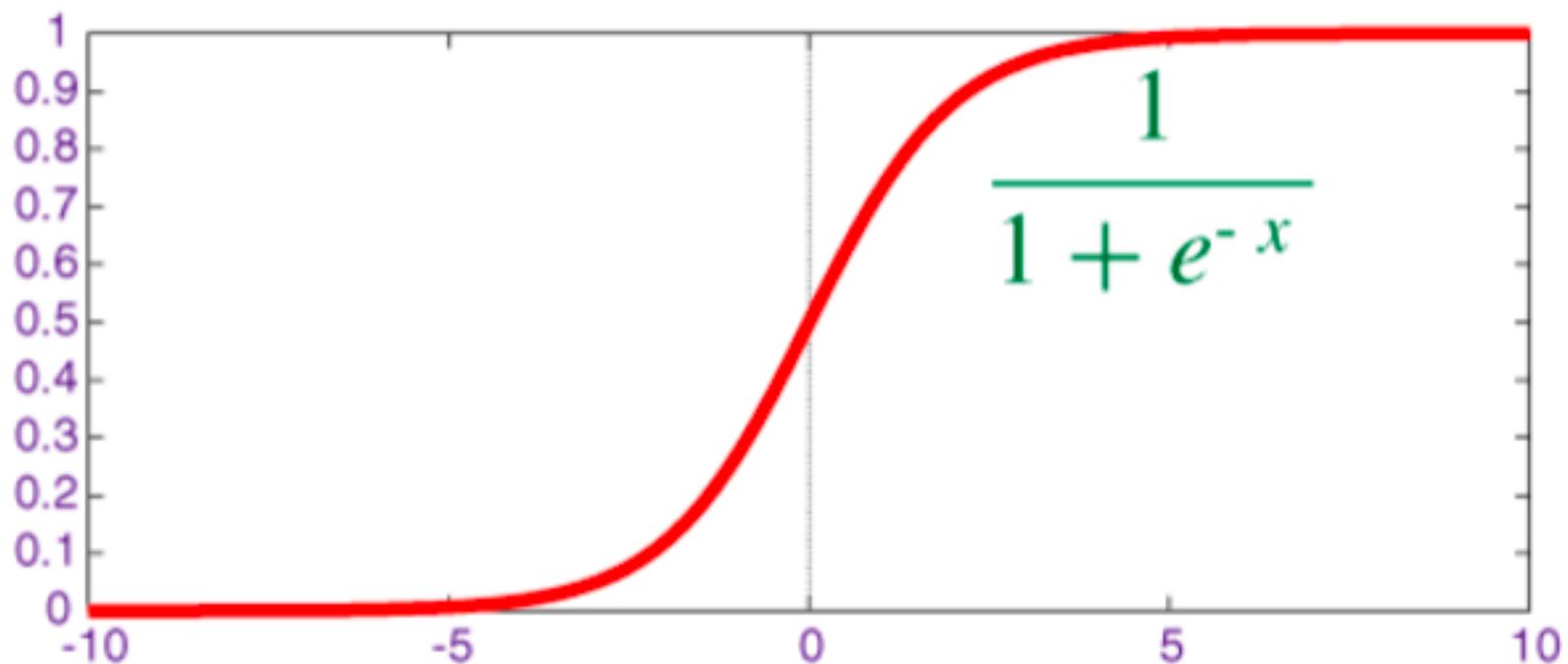
Deep Learning



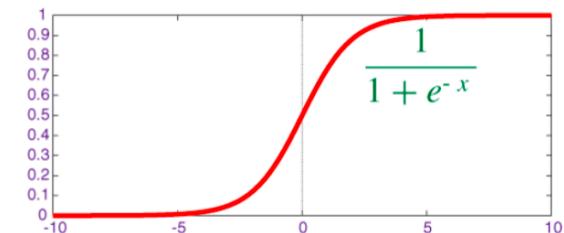
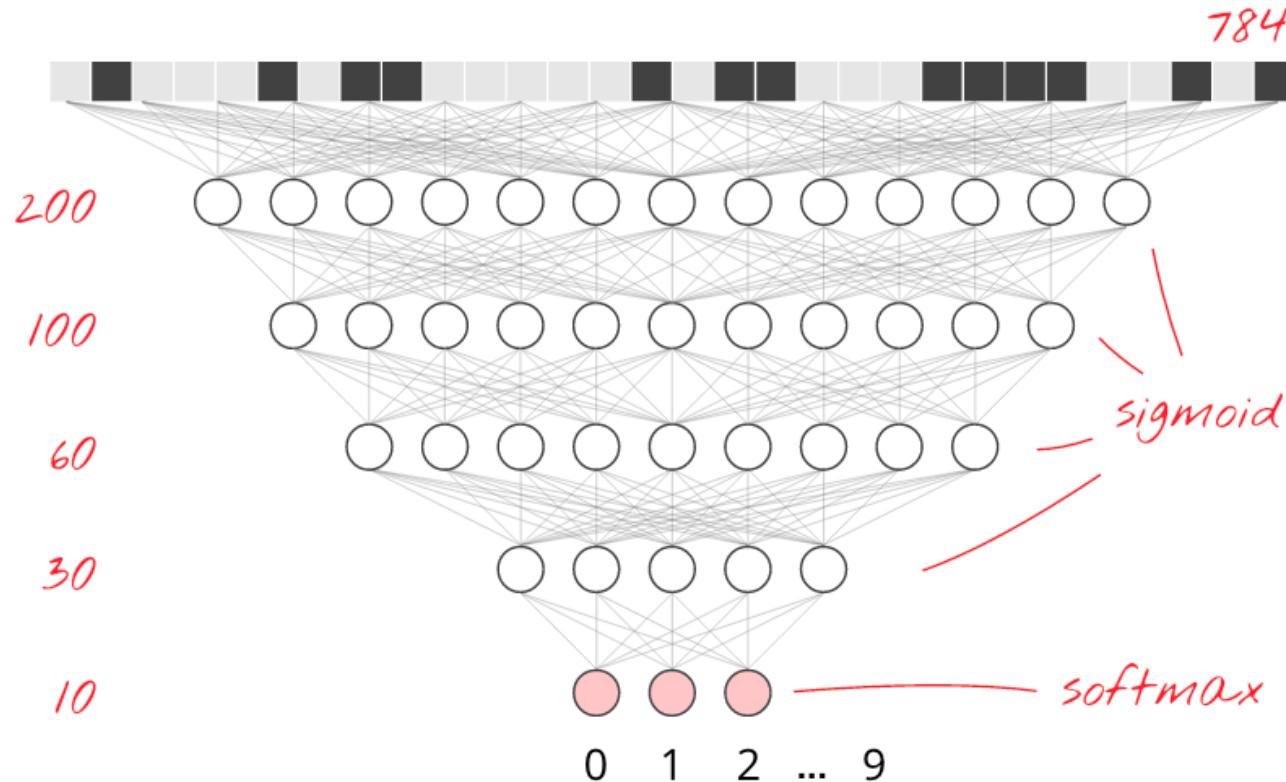
5 fully-connected layers



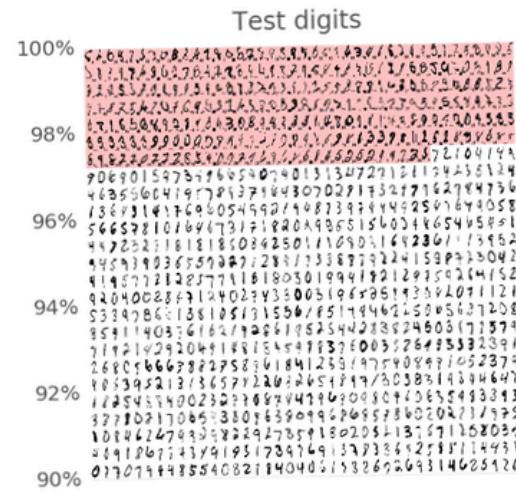
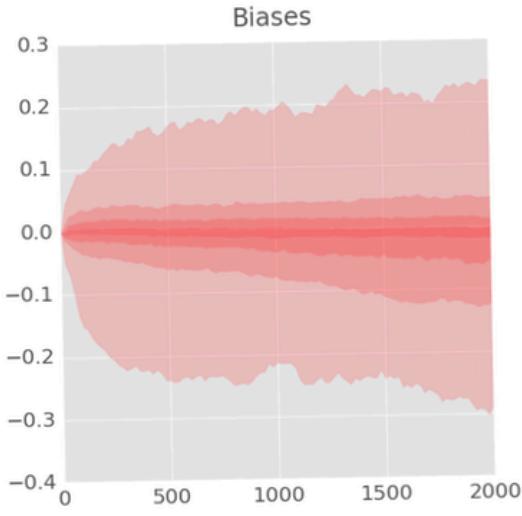
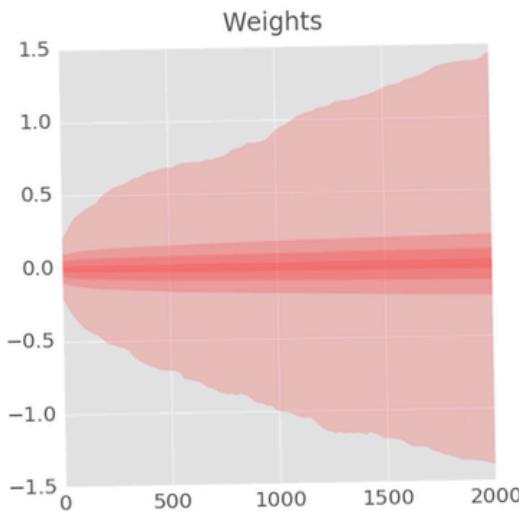
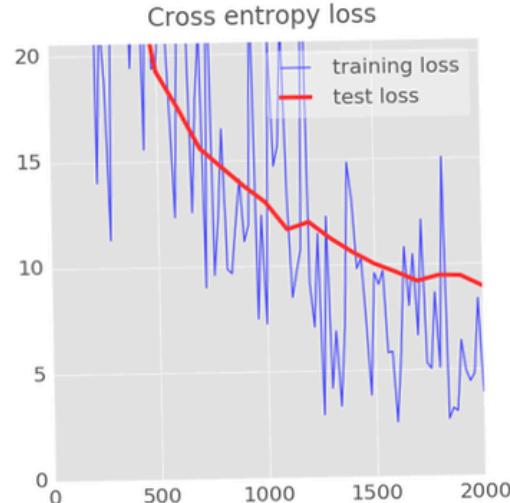
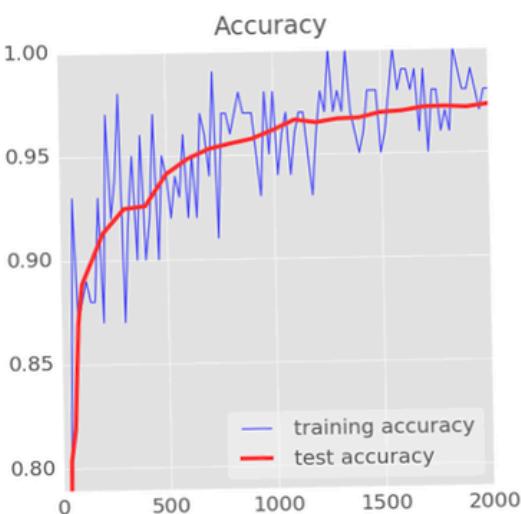
Sigmoid



5 fully-connected layers



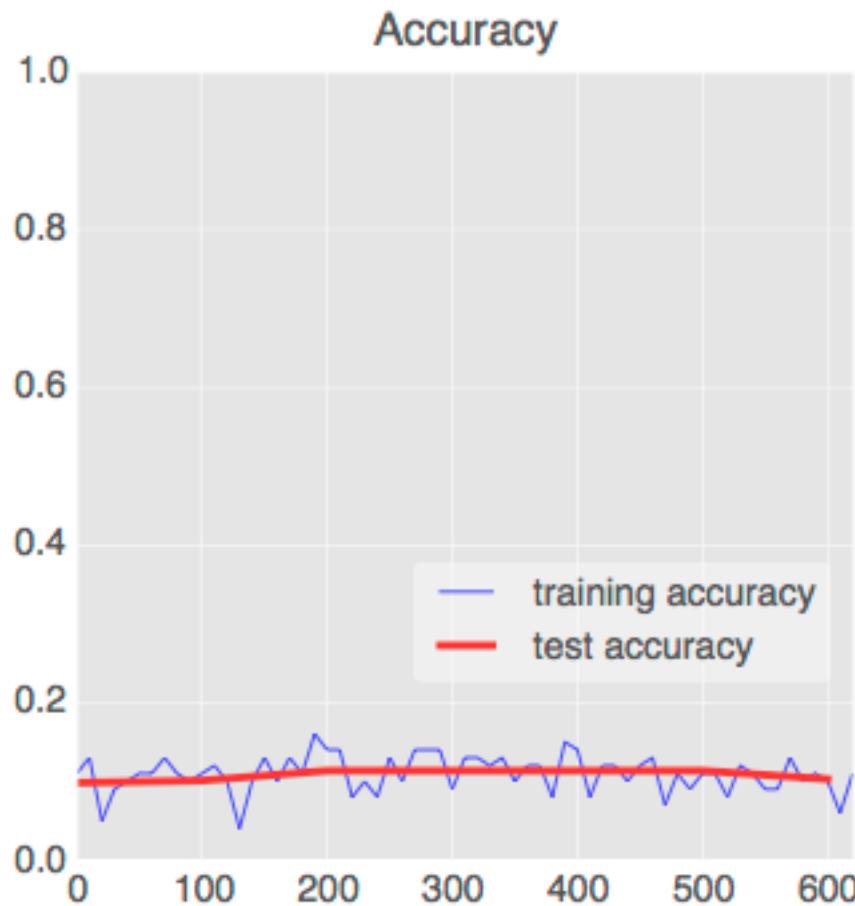
TensorFlow MNIST Tutorial



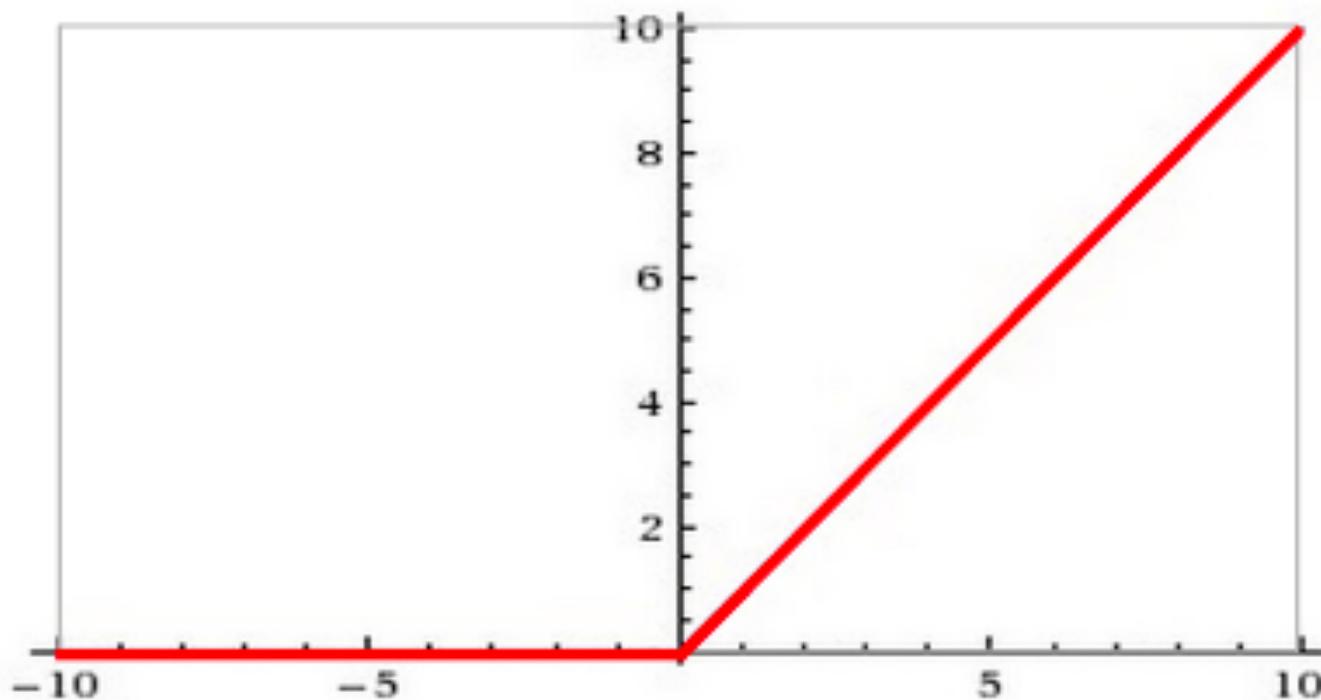
ReLU



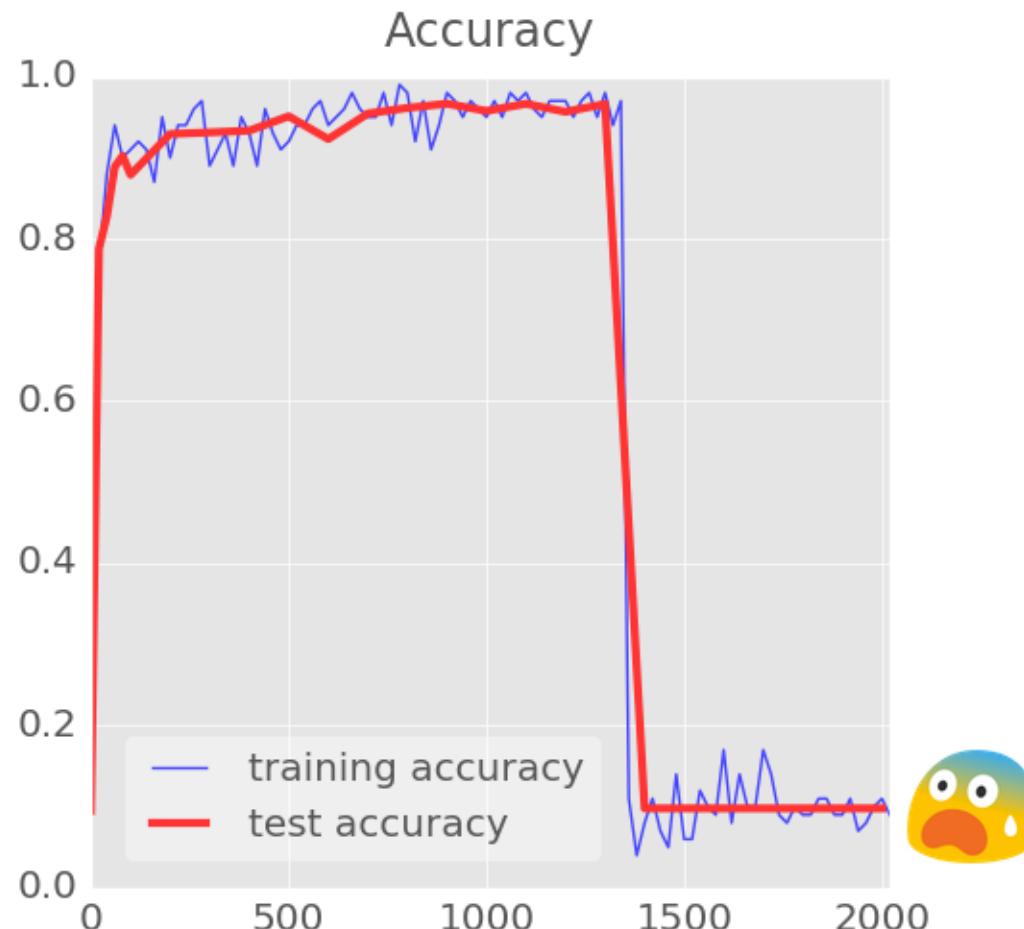
TensorFlow MNIST Tutorial



ReLU



TensorFlow MNIST Tutorial



Learning Rate

Slow down...

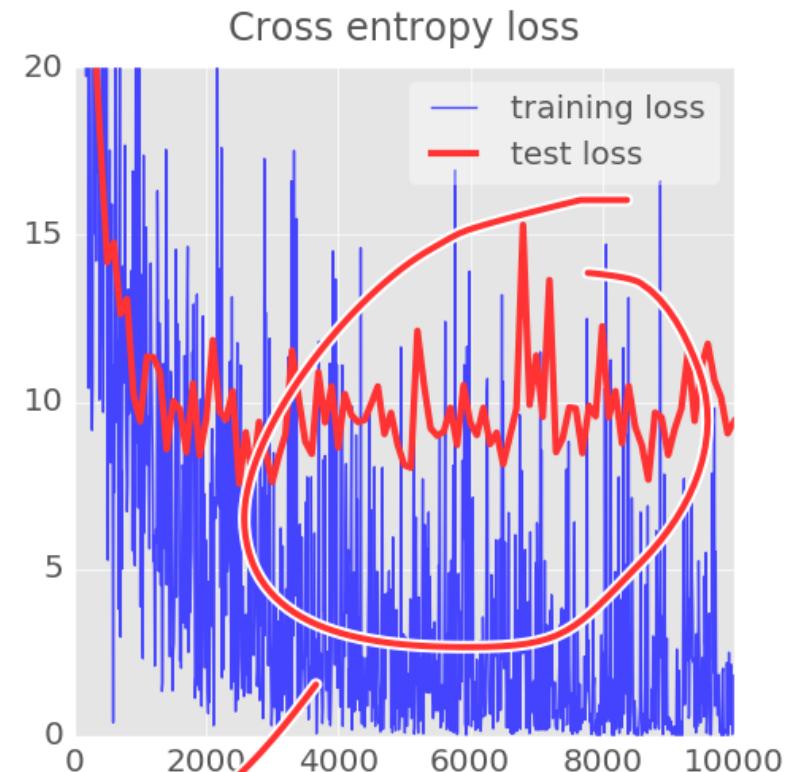
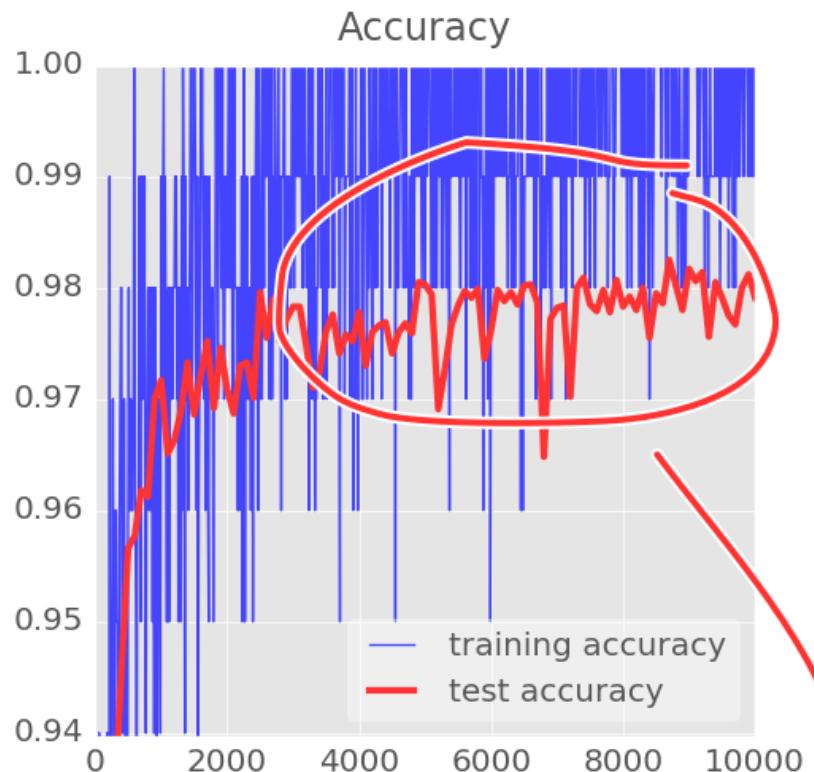


Learning
rate decay



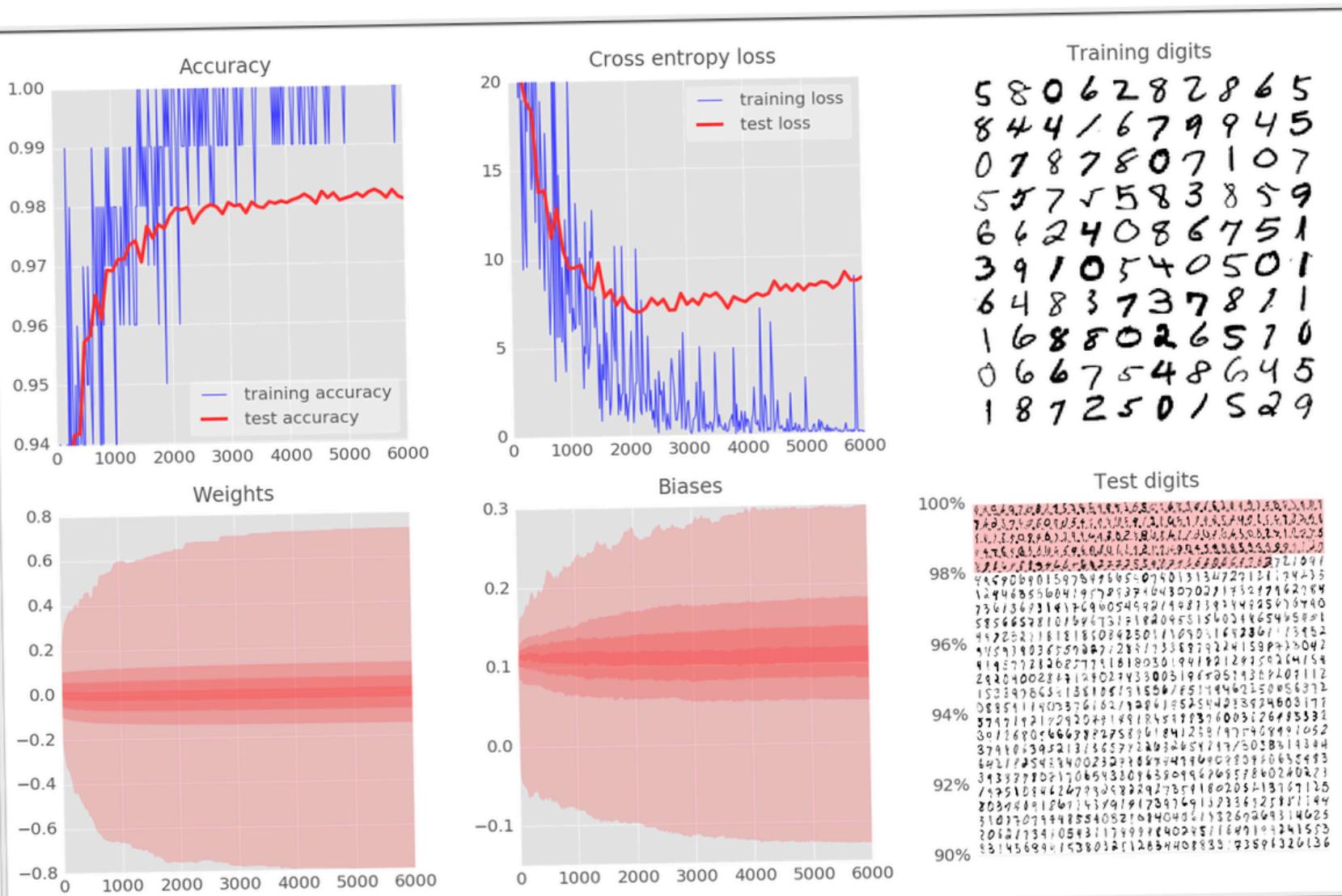
TensorFlow MNIST Tutorial

LR =
0.003



yuck!

TensorFlow MNIST Tutorial

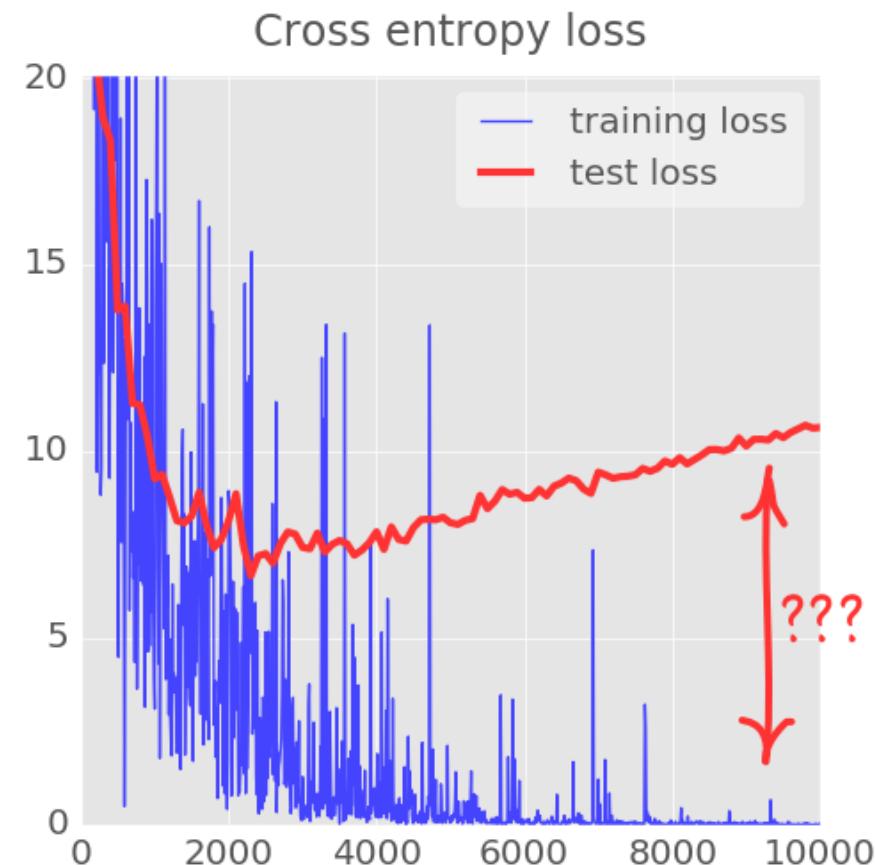
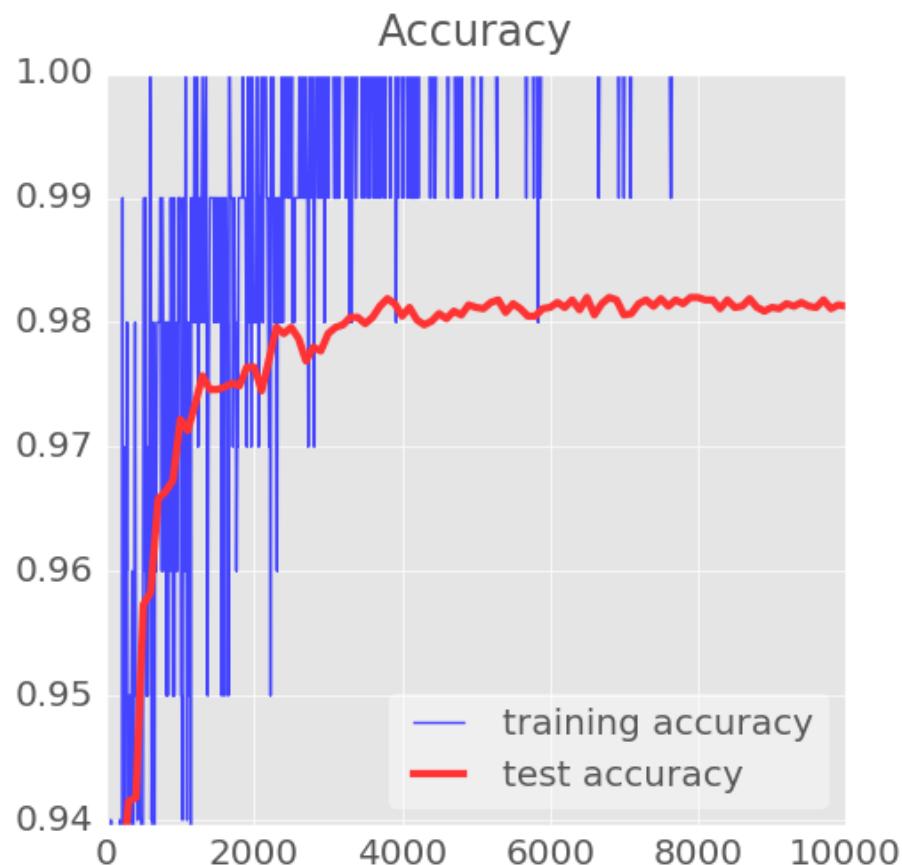


Dropout

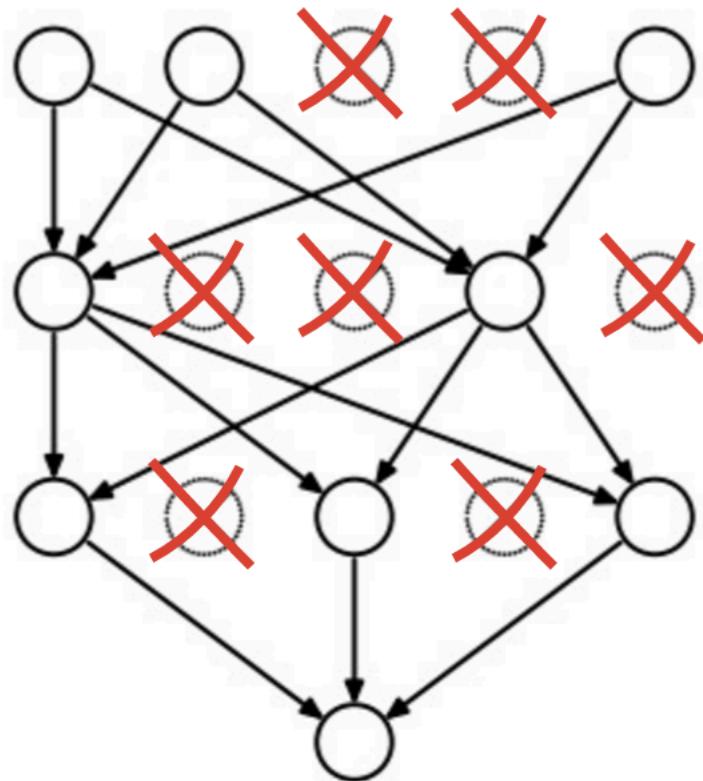
Dropout



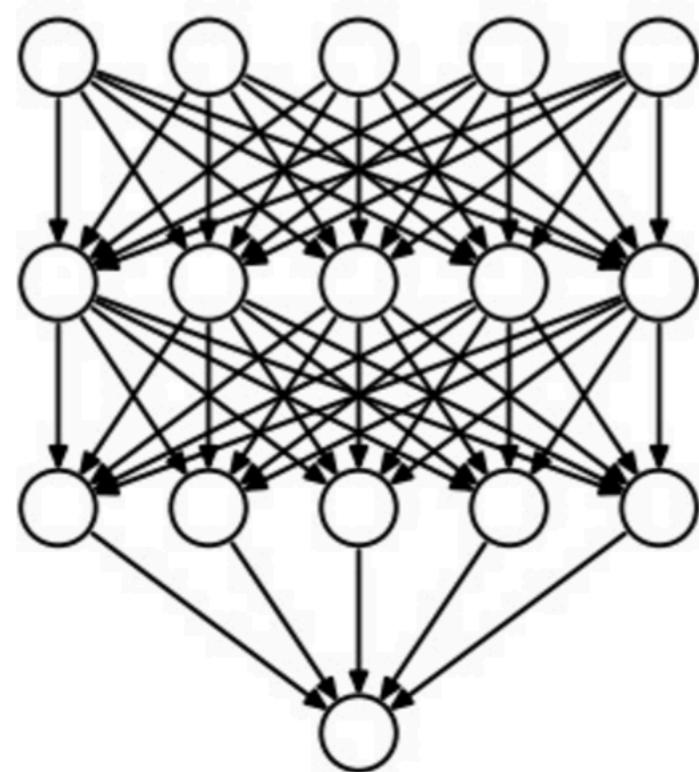
Overfitting



Dropout

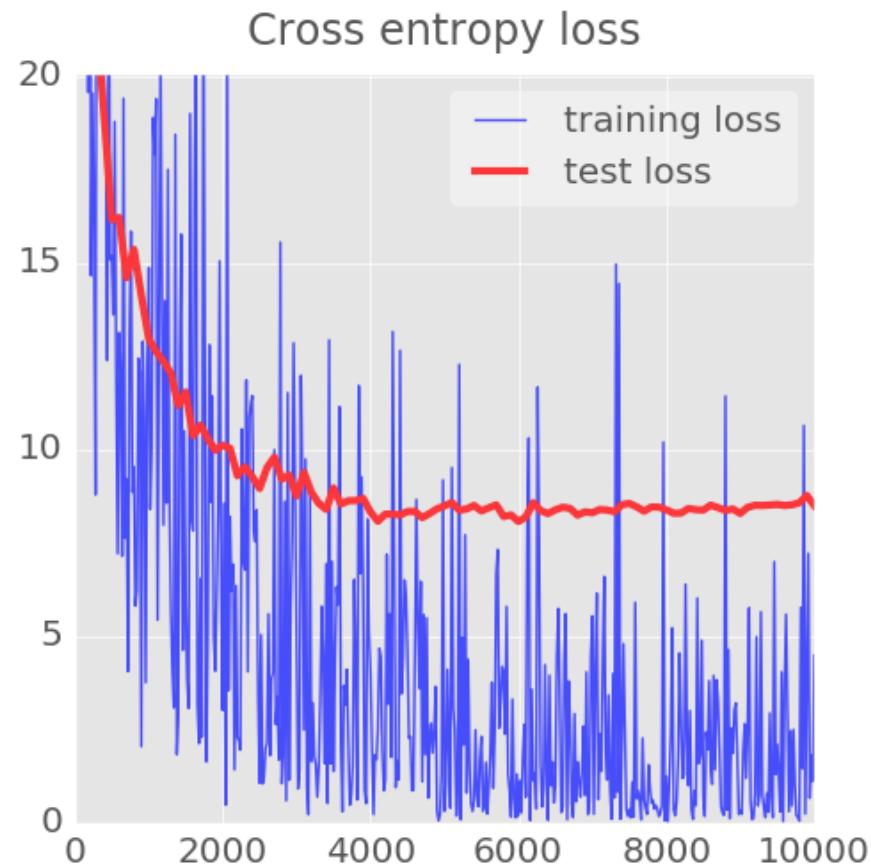
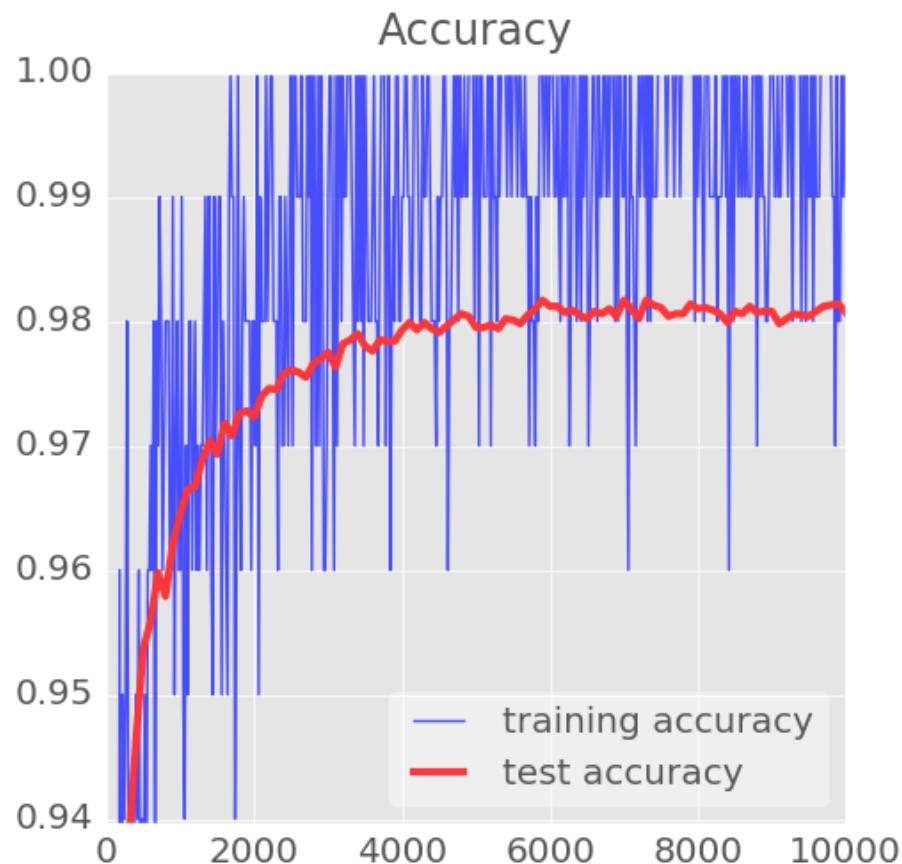


Training
 $p_{keep} = 0.75$



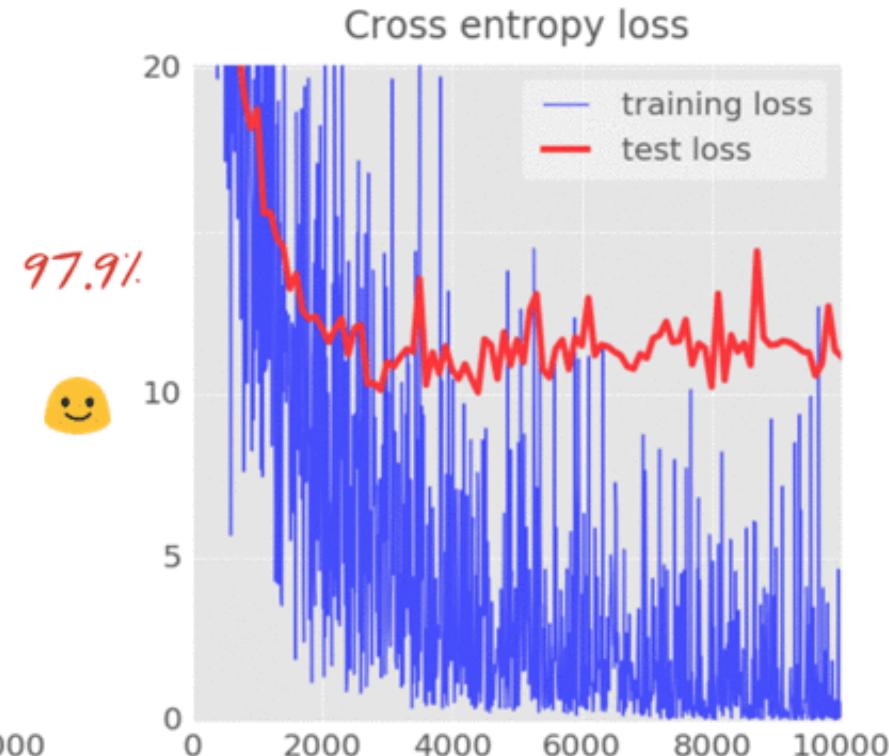
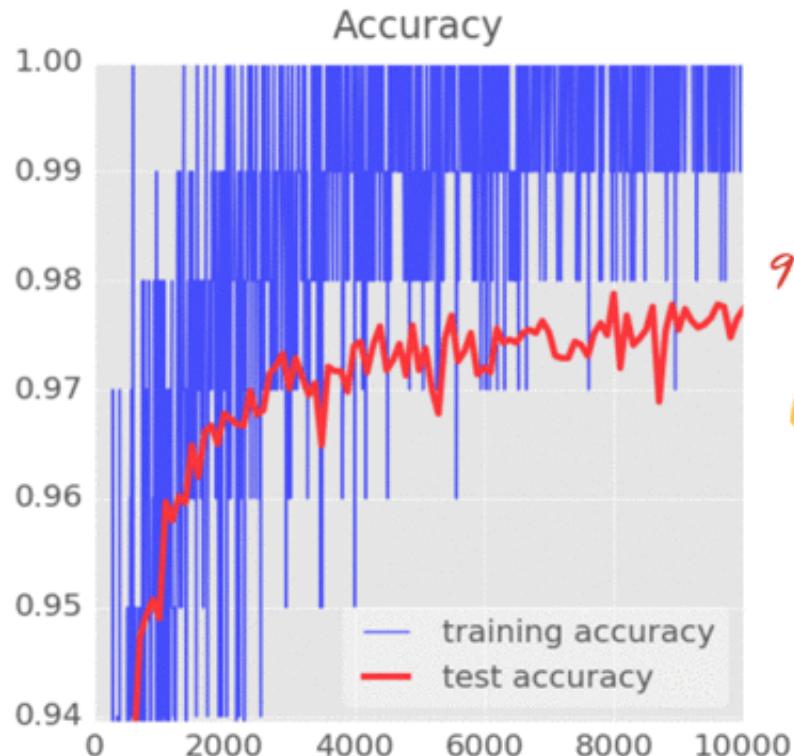
Test
 $p_{keep} = 1.0$

TensorFlow MNIST Tutorial



TensorFlow MNIST Tutorial

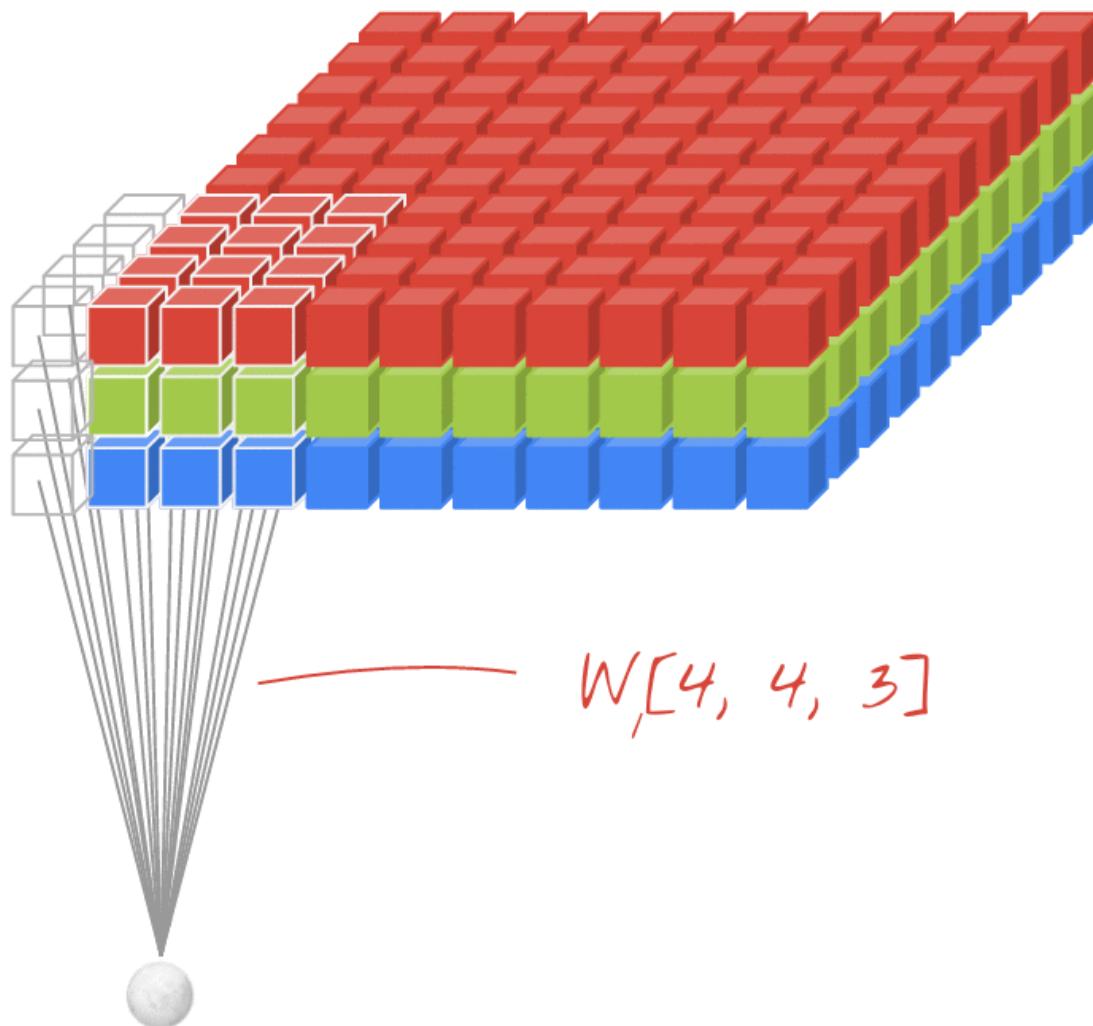
5 layers
Sigmoid



Overfitting



Convolutional Layer

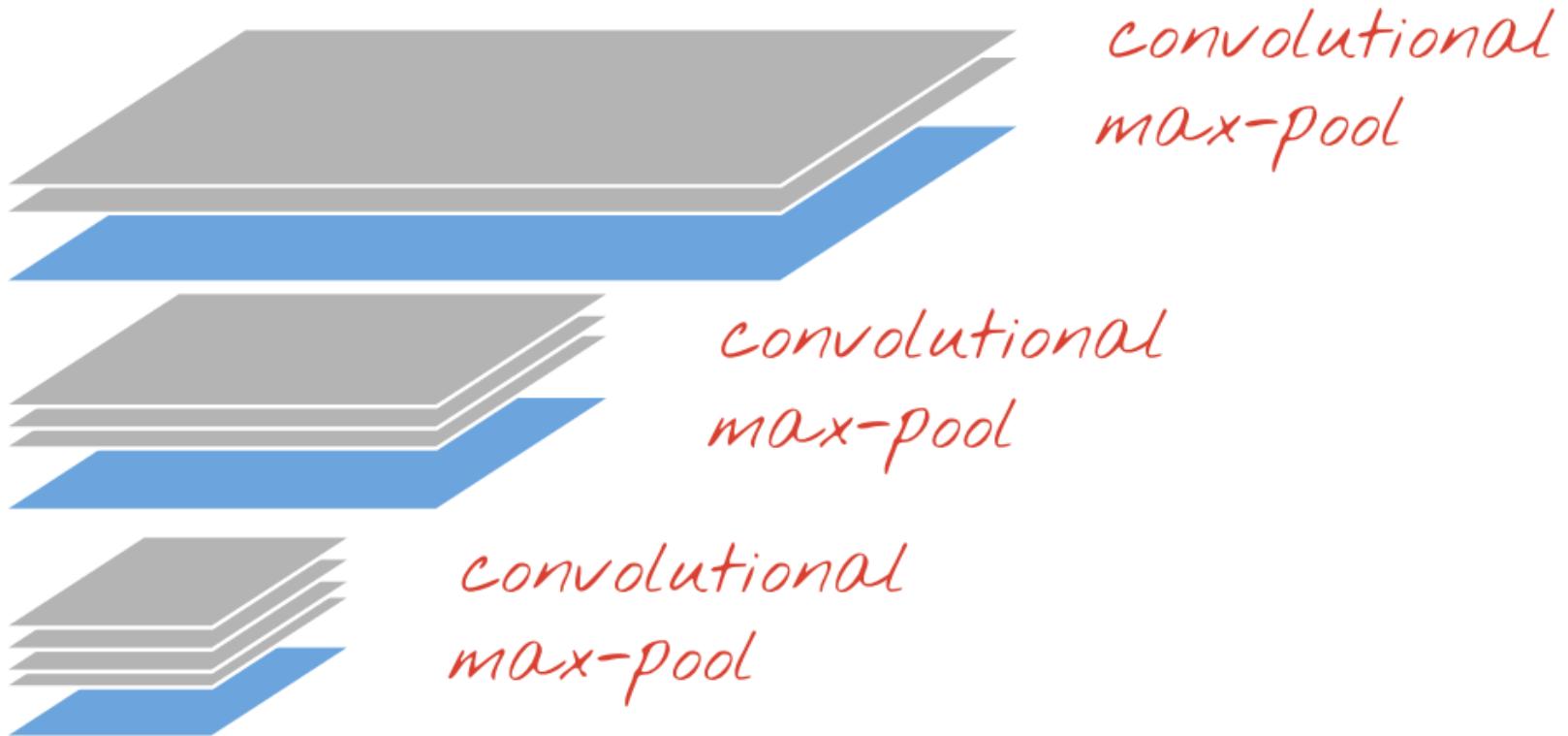


Convolutional Layer

$W[4, 4, 3]$ |
 $W_2[4, 4, 3]$ | $W[4, \underline{4}, 3, 2]$

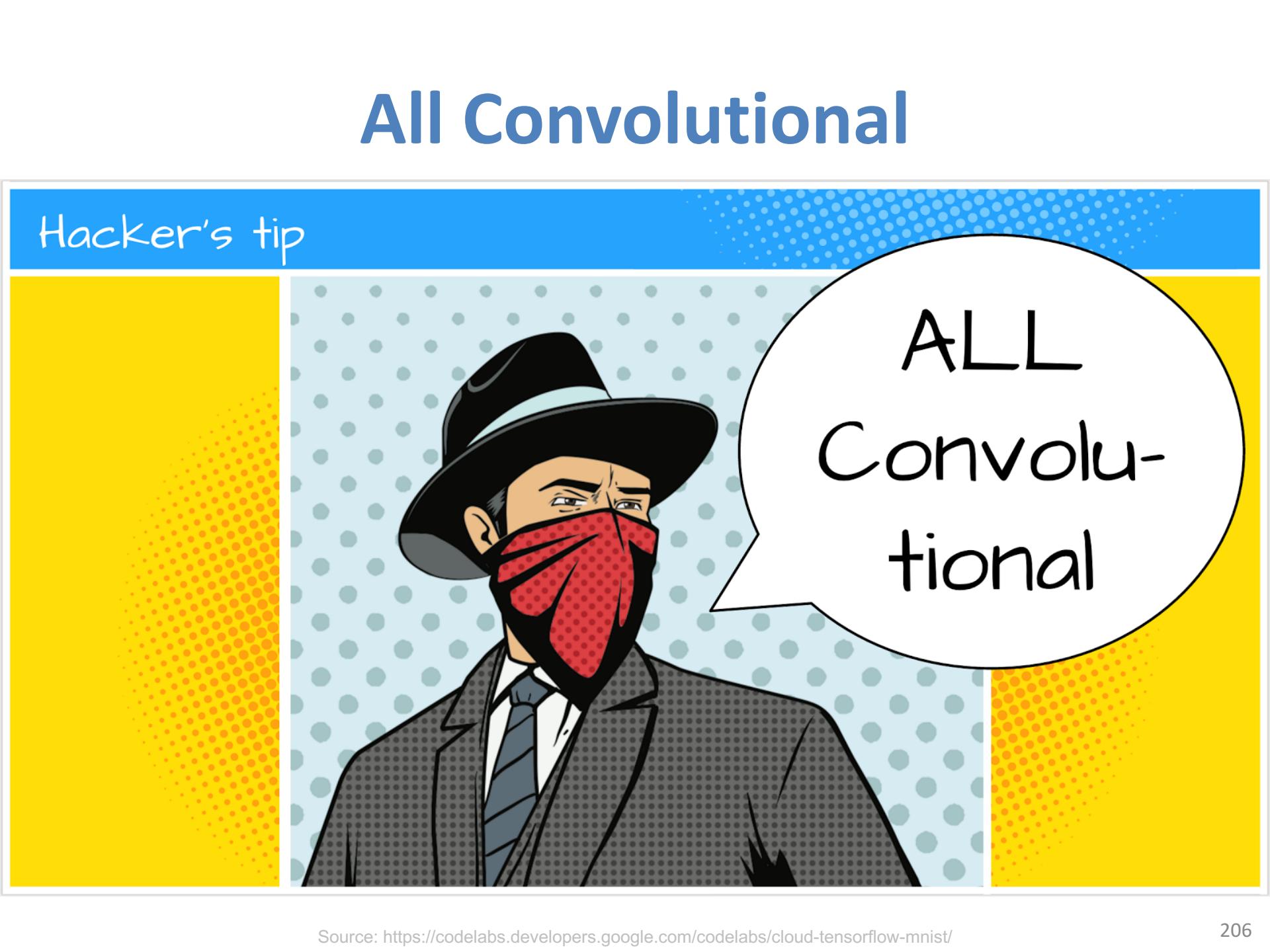
filter size input channels output channels

Convolutional Max-Pool



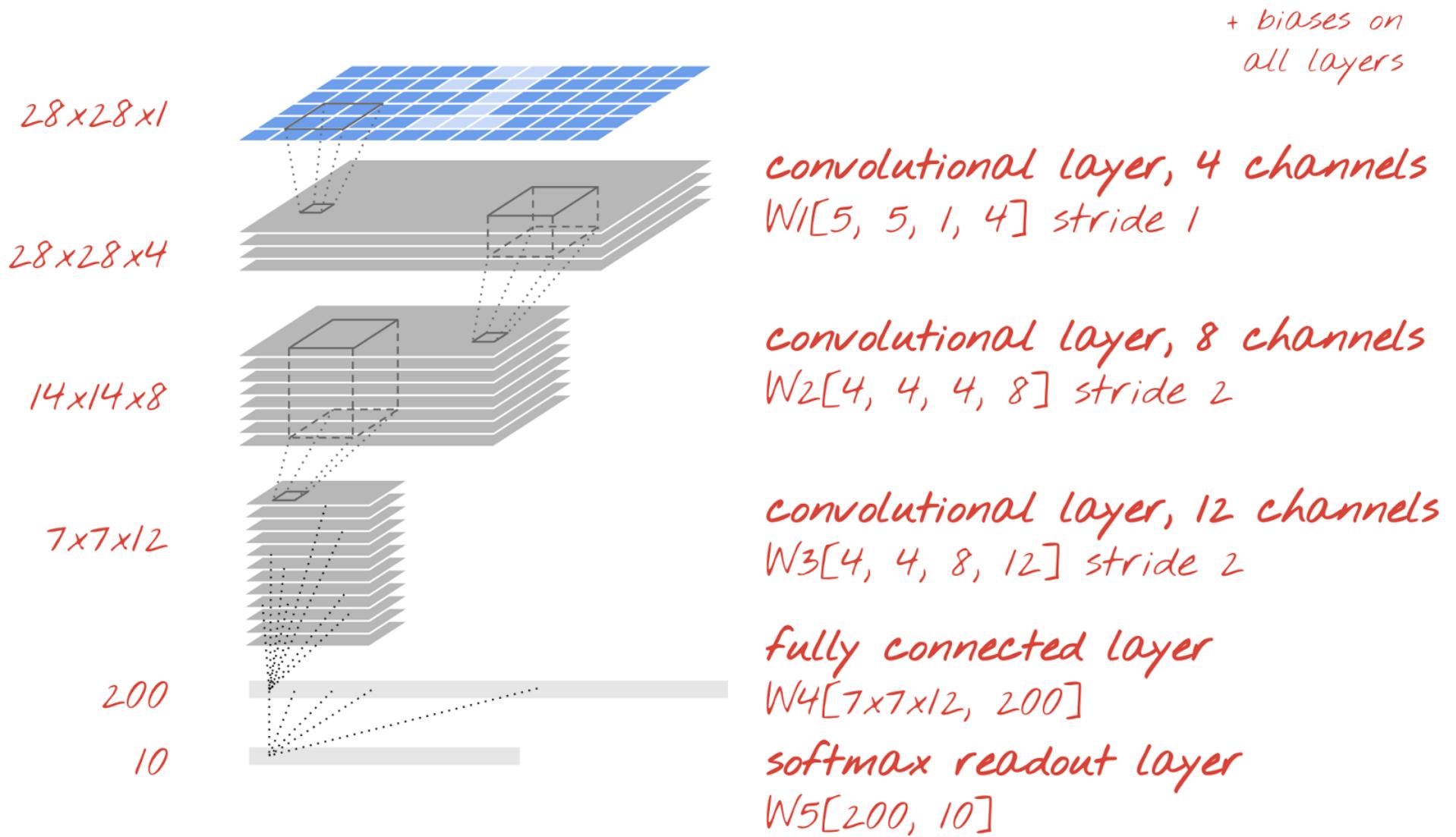
All Convolutional

Hacker's tip

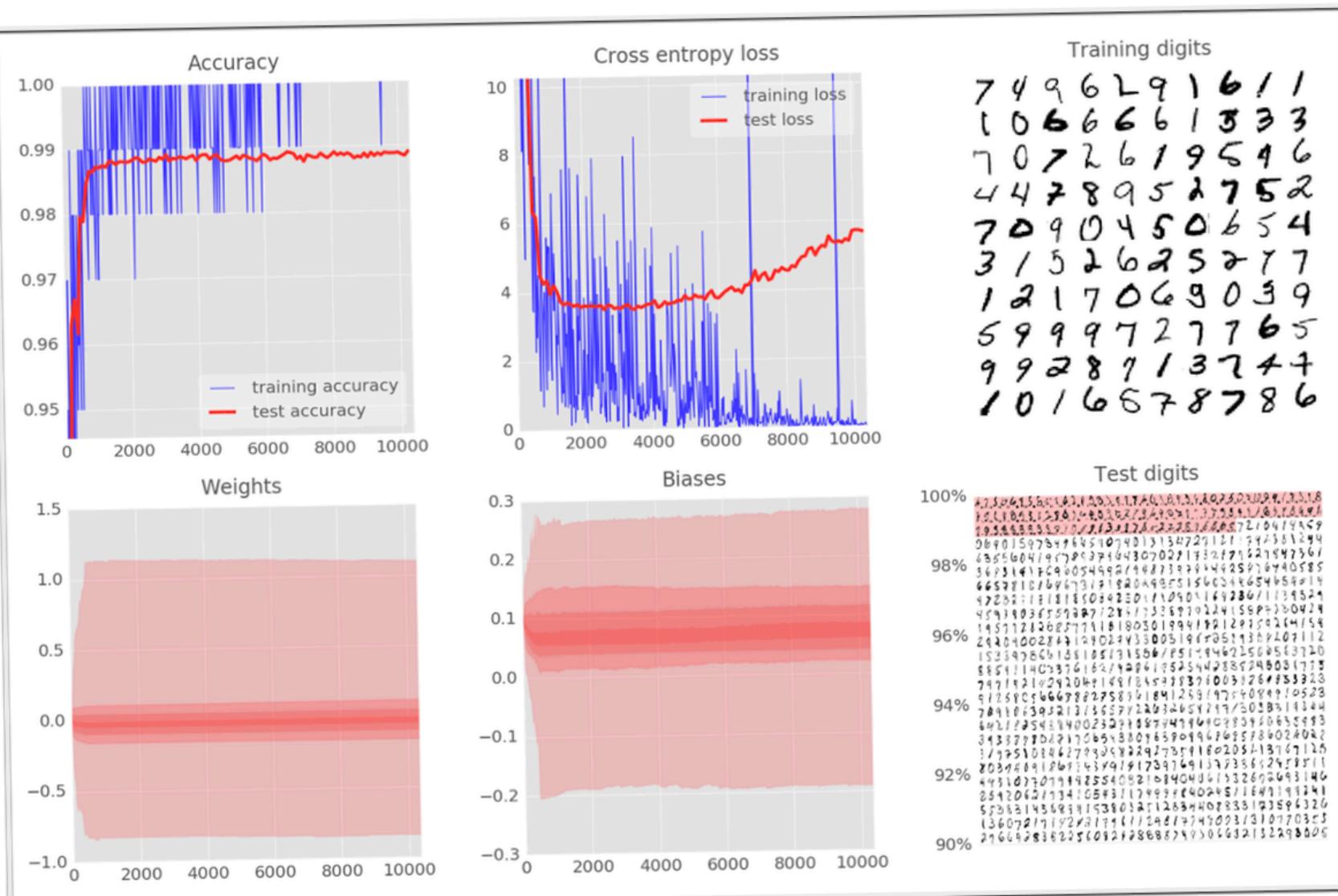


ALL
Convolu-
tional

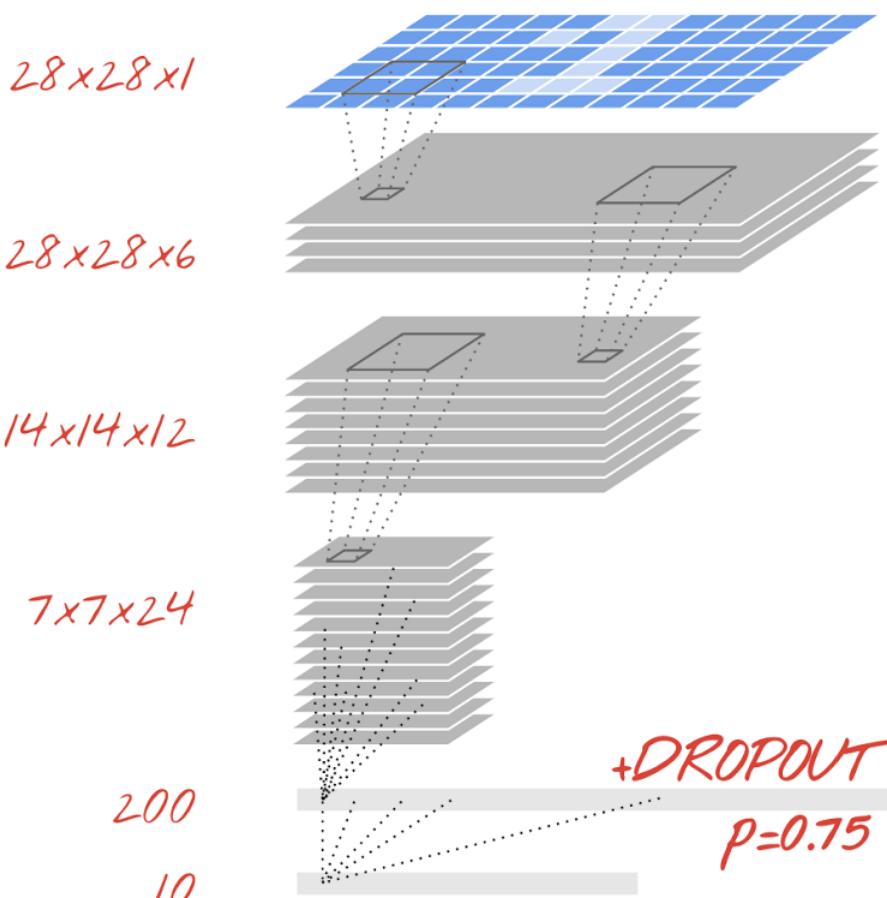
Bigger Convolutional Neural Network



Bigger Convolutional Neural Network



Bigger Convolutional Neural Network + Dropout



+ biases on all layers

convolutional layer, 6 channels

$W1[6, 6, 1, 6]$ stride 1

convolutional layer, 12 channels

$W2[5, 5, 6, 12]$ stride 2

convolutional layer, 24 channels

$W3[4, 4, 12, 24]$ stride 2

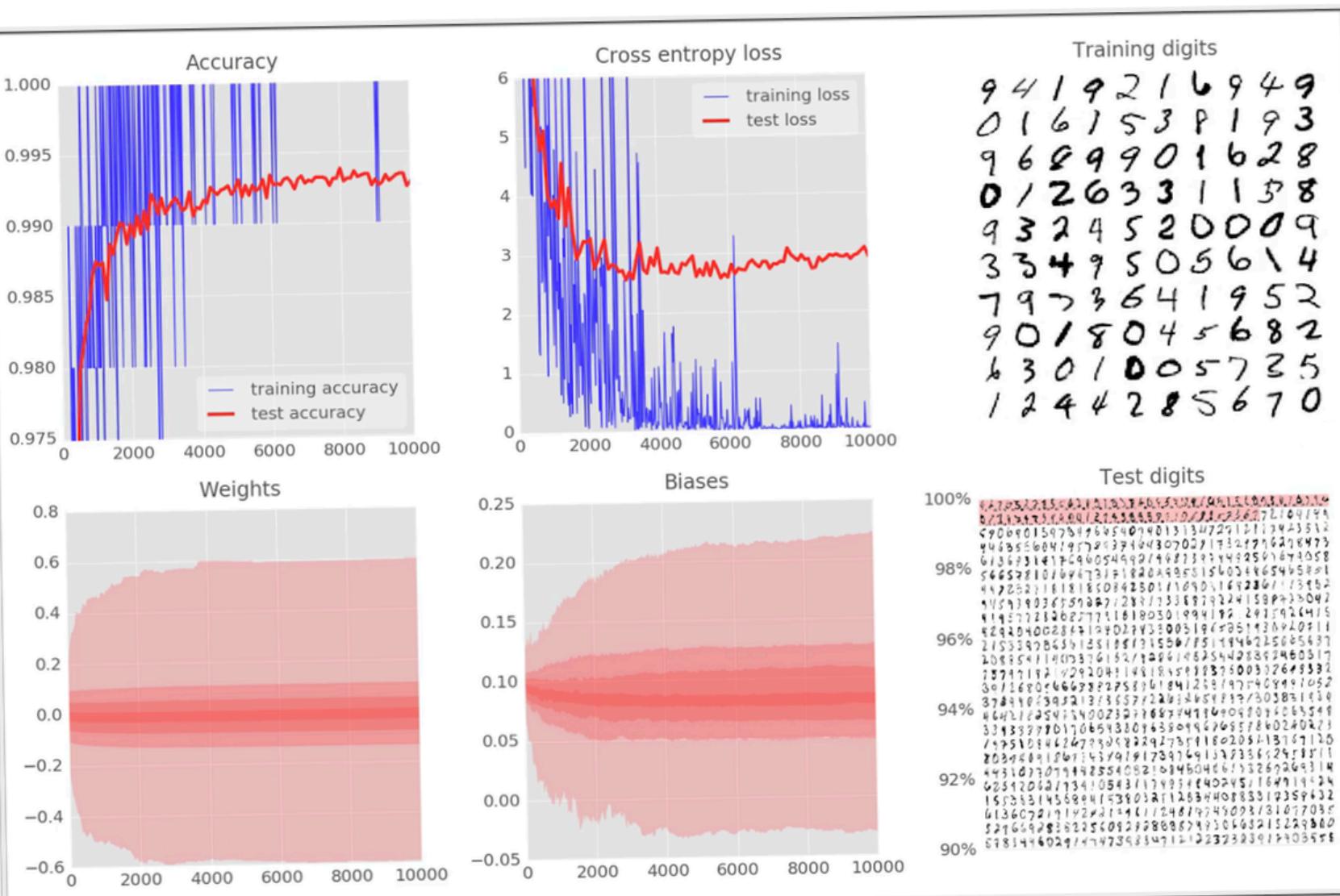
fully connected layer

$W4[7 \times 7 \times 24, 200]$

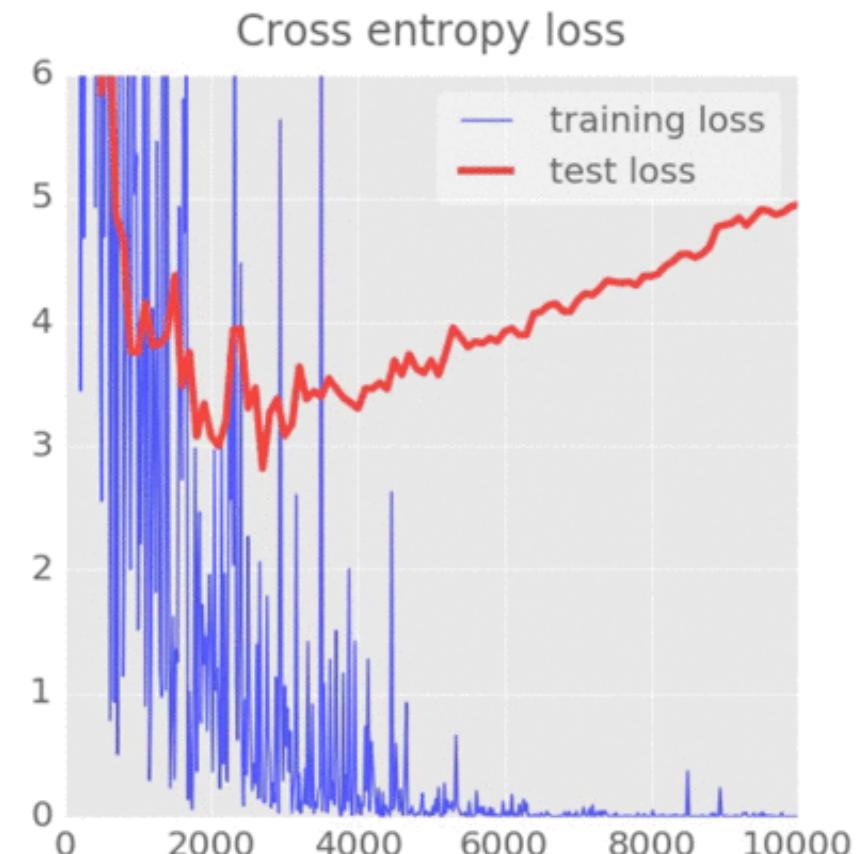
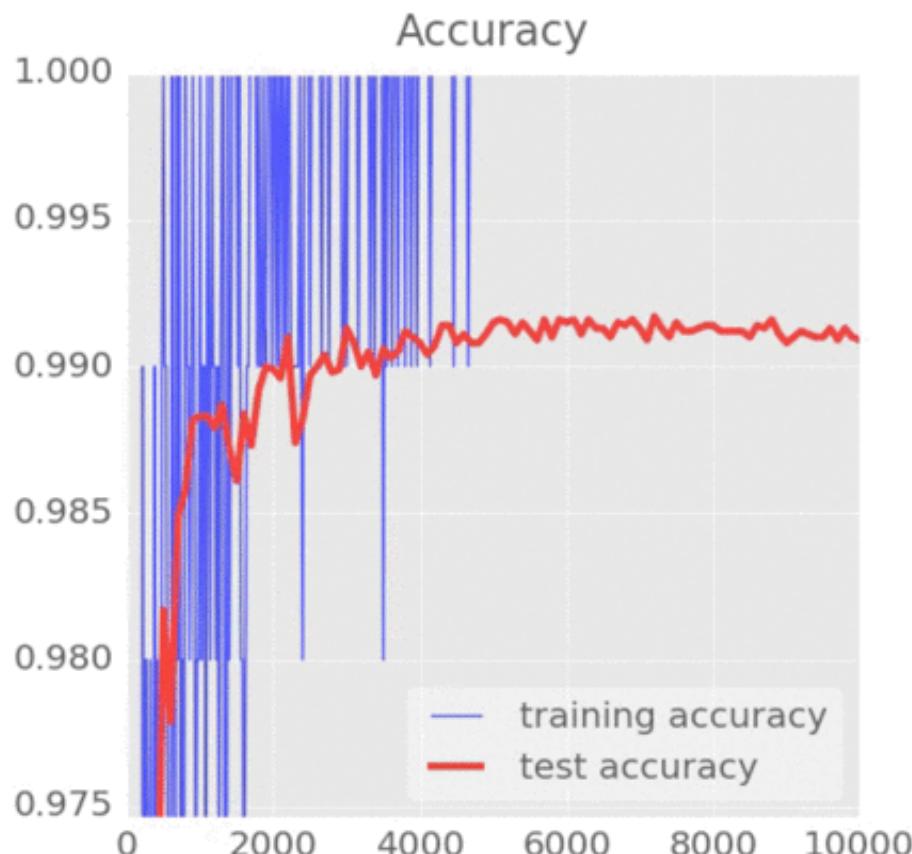
softmax readout layer

$W5[200, 10]$

TensorFlow MNIST Tutorial

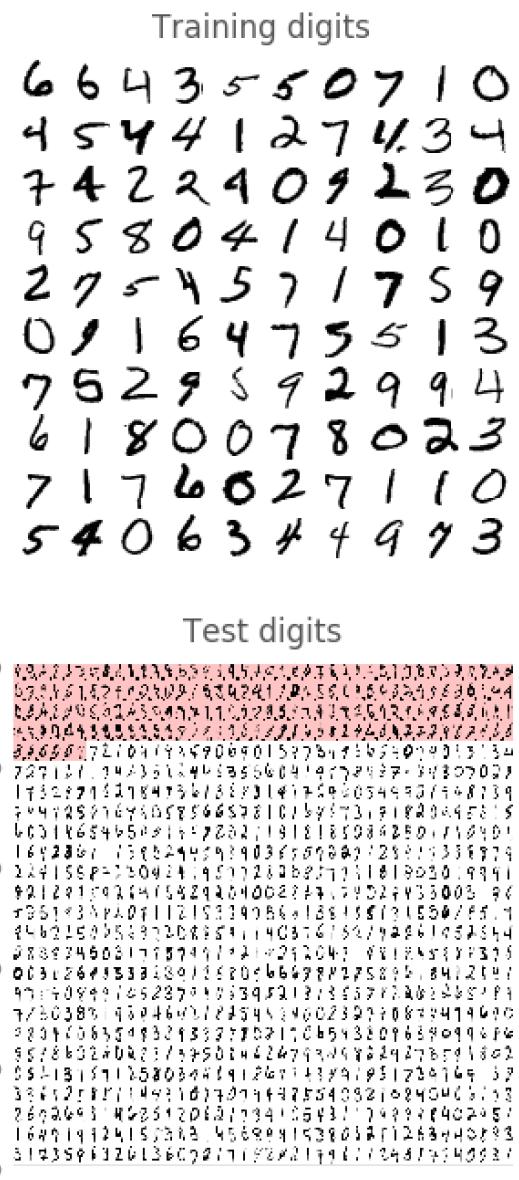
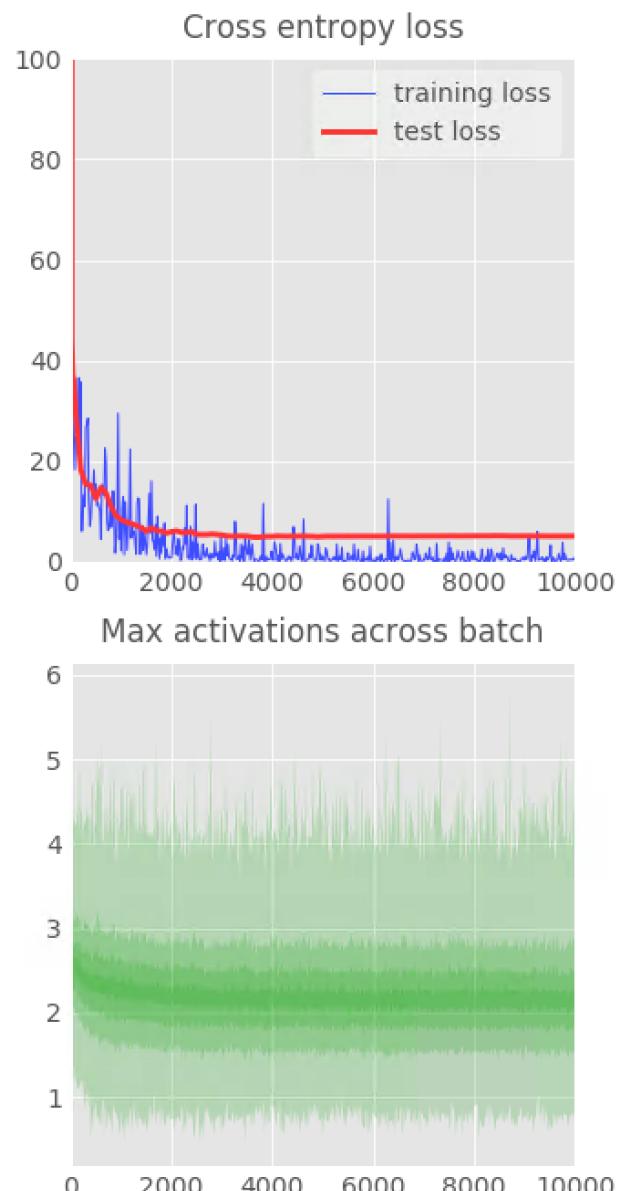
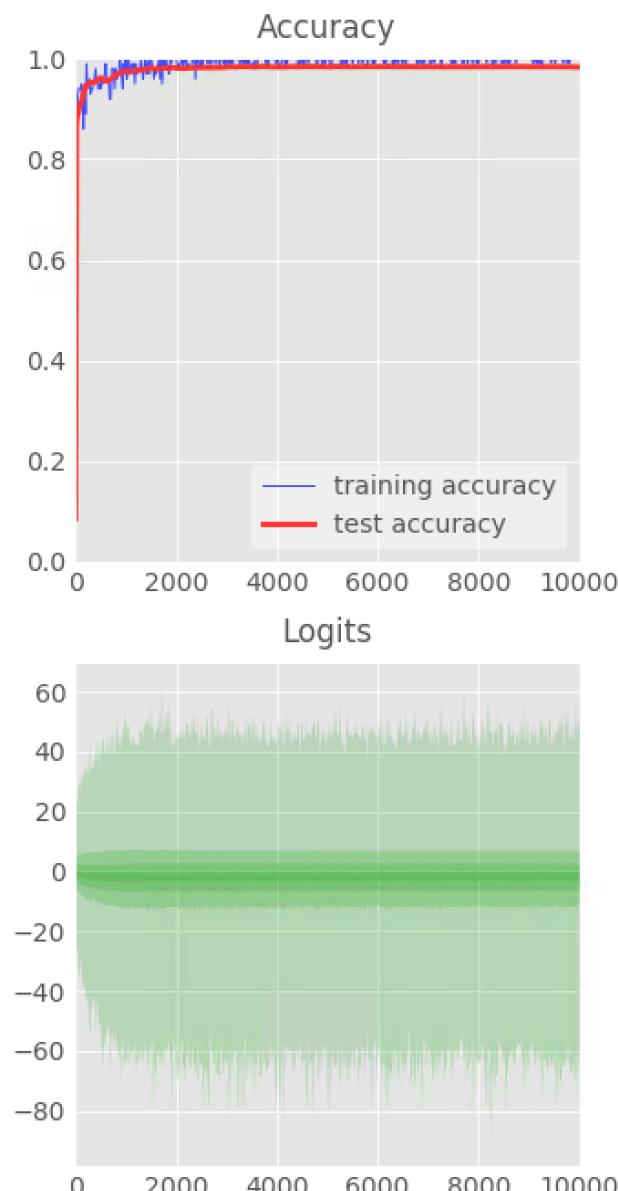


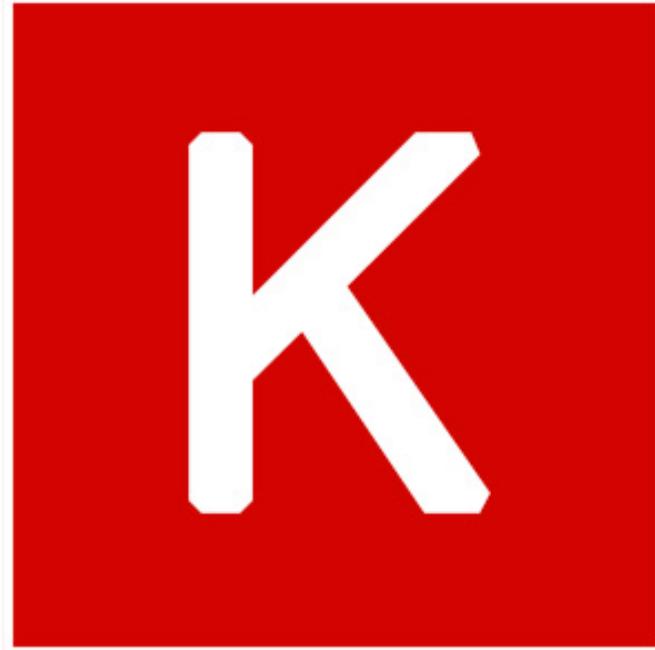
TensorFlow MNIST Tutorial



larger convolutional network

TensorFlow MNIST Tutorial





Keras



Keras

- Keras is a **high-level neural networks API**
- 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.

Keras



Docs » Home

Edit on GitHub

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 TensorFlow to Theano

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 »

Keras: Deep Learning library for Theano and TensorFlow

You have just found Keras.

Keras is a high-level neural networks API, 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 user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at [Keras.io](#).

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

Guiding principles

- **User friendliness.** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **Modularity.** A model is understood as a sequence or a graph of standalone, fully-configurable modules that can

Deep Learning with Keras

Keras Installation

The screenshot shows the Keras Documentation website. The sidebar on the left contains a search bar, a navigation menu with links like Home, Installation, and various API documentation sections, and a footer with links to About Keras models, Sequential, Model (functional API), and Layers.

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 TensorFlow to Theano

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

Installation

Keras uses the following dependencies:

- numpy, scipy
- yaml
- HDF5 and h5py (optional, required if you use model saving/loading functions)
- Optional but recommended if you use CNNs: cuDNN.

When using the TensorFlow backend:

- TensorFlow
 - See installation instructions.

When using the Theano backend:

- Theano
 - See installation instructions.

To install Keras, `cd` to the Keras folder and run the install command:

```
sudo python setup.py install
```

You can also install Keras from PyPI:

```
sudo pip install keras
```

```
conda info --envs
```

```
conda --version
```

```
python --version
```

```
conda list
```

```
conda create -n tensorflow python=3.5
```

```
source activate tensorflow
```

```
activate tensorflow
```

pip install Theano

conda install pydot

sudo pip install keras

pip install keras

pip install tensorflow

pip install ipython[all]

Gensim

pip install -U gensim

```
bash-3.2$ pip install -U gensim
Collecting gensim
  Downloading gensim-2.0.0-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x8
  6_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (5.6MB)
    100% |██████████| 5.6MB 126kB/s
Requirement already up-to-date: six>=1.5.0 in ./anaconda/lib/python3.6/site-packages (from
gensim)
Collecting scipy>=0.7.0 (from gensim)
  Downloading scipy-0.19.0-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x8
  6_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (16.2MB)
    100% |██████████| 16.2MB 43kB/s
Collecting smart-open>=1.2.1 (from gensim)
  Downloading smart_open-1.5.2.tar.gz
Collecting numpy>=1.3 (from gensim)
  Downloading numpy-1.12.1-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x8
  6_64.macosx_10_10_intel.macosx_10_10_x86_64.whl (4.4MB)
    100% |██████████| 4.4MB 148kB/s
Collecting boto>=2.32 (from smart-open>=1.2.1->gensim)
  Downloading boto-2.46.1-py2.py3-none-any.whl (1.4MB)
    100% |██████████| 1.4MB 372kB/s
Requirement already up-to-date: bz2file in ./anaconda/lib/python3.6/site-packages (from
smart-open>=1.2.1->gensim)
Collecting requests (from smart-open>=1.2.1->gensim)
  Downloading requests-2.13.0-py2.py3-none-any.whl (584kB)
    100% |██████████| 593kB 632kB/s
Building wheels for collected packages: smart-open
  Running setup.py bdist wheel for smart-open ... done
  Stored in directory: /Users/imyday/Library/Caches/pip/wheels/02/44/43/68e963ce2b45baef
a913a4e558bcd787403458afddffcf45ca
Successfully built smart-open
Installing collected packages: numpy, scipy, boto, requests, smart-open, gensim
  Found existing installation: numpy 1.11.3
    Uninstalling numpy-1.11.3:
      Successfully uninstalled numpy-1.11.3
  Found existing installation: scipy 0.18.1
    Uninstalling scipy-0.18.1:
      Successfully uninstalled scipy-0.18.1
  Found existing installation: boto 2.45.0
    DEPRECATION: Uninstalling a distutils installed project (boto) has been deprecated a
nd will be removed in a future version. This is due to the fact that uninstalling a dist
utils project will only partially uninstall the project.
    Uninstalling boto-2.45.0:
      Successfully uninstalled boto-2.45.0
  Found existing installation: requests 2.12.4
    Uninstalling requests-2.12.4:
      Successfully uninstalled requests-2.12.4
  Found existing installation: smart-open 1.4.0
    Uninstalling smart-open-1.4.0:
      Successfully uninstalled smart-open-1.4.0
  Found existing installation: gensim 1.0.1
    Uninstalling gensim-1.0.1:
      Successfully uninstalled gensim-1.0.1
Successfully installed boto-2.46.1 gensim-2.0.0 numpy-1.12.1 requests-2.13.0 scipy-0.19.
0 smart-open-1.5.2
bash-3.2$
```

Keras

`sudo pip install keras`

```
bash-3.2$ sudo pip install keras
Password:
The directory '/Users/imyday/Library/Caches/pip/http' or its parent directory is not owned by the current user and the cache has been disabled. Please check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
The directory '/Users/imyday/Library/Caches/pip' or its parent directory is not owned by the current user and caching wheels has been disabled. check the permissions and owner of that directory. If executing pip with sudo, you may want sudo's -H flag.
Collecting keras
  Downloading Keras-2.0.3.tar.gz (196kB)
    100% |██████████| 204kB 365kB/s
Collecting theano (from keras)
  Downloading Theano-0.9.0.tar.gz (3.1MB)
    100% |██████████| 3.1MB 148kB/s
Requirement already satisfied: pyyaml in ./anaconda/lib/python3.6/site-packages (from keras)
Requirement already satisfied: six in ./anaconda/lib/python3.6/site-packages (from keras)
Requirement already satisfied: numpy>=1.9.1 in ./anaconda/lib/python3.6/site-packages (from theano->keras)
Requirement already satisfied: scipy>=0.14 in ./anaconda/lib/python3.6/site-packages (from theano->keras)
Installing collected packages: theano, keras
  Running setup.py install for theano ... done
  Running setup.py install for keras ... done
Successfully installed keras-2.0.3 theano-0.9.0
bash-3.2$ █
```

TensorFlow

pip install tensorflow

```
bash-3.2$ pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-1.1.0-cp36-cp36m-macosx_10_11_x86_64.whl (31.3MB)
    100% |██████████| 31.3MB 23kB/s
Requirement already satisfied: wheel>=0.26 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: six>=1.10.0 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Collecting protobuf>=3.2.0 (from tensorflow)
  Downloading protobuf-3.2.0-py2.py3-none-any.whl (360kB)
    100% |██████████| 368kB 453kB/s
Requirement already satisfied: werkzeug>=0.11.10 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: numpy>=1.11.0 in ./anaconda/lib/python3.6/site-packages (from tensorflow)
Requirement already satisfied: setuptools in ./anaconda/lib/python3.6/site-packages/setuptools-27.2.0-py3.6.
egg (from protobuf>=3.2.0->tensorflow)
Installing collected packages: protobuf, tensorflow
Successfully installed protobuf-3.2.0 tensorflow-1.1.0
bash-3.2$
```

Keras Github

 Features Business Explore Pricing This repository Search Sign in or Sign up

 Watch 1,018 Star 14,893 Fork 5,180

Code Issues 2,486 Pull requests 27 Projects 1 Wiki Pulse Graphs

Deep Learning library for Python. Convnets, recurrent neural networks, and more. Runs on TensorFlow or Theano.
<http://keras.io/>

deep-learning tensorflow theano neural-networks machine-learning data-science

3,503 commits 4 branches 28 releases 424 contributors

Branch: master ▾ New pull request Find file Clone or download ▾

Commit	Message	Time
 phipleg	committed with fchollet Added logsumexp to backend. (#6346)	Latest commit 7d52af6 a day ago
 docker	Update docker files to TensorFlow 1, Theano 0.9 (#6116)	20 days ago
 docs	fix stateful RNNs FAQ link (#6336)	3 days ago
 examples	Spelling errors (#6232)	11 days ago
 keras	Added logsumexp to backend. (#6346)	a day ago
 tests	Added logsumexp to backend. (#6346)	a day ago
 .gitignore	Fix FAQ question	a month ago
 .travis.yml	Update Travis config	9 days ago
 CONTRIBUTING.md	Mention requests for contribution in CONTRIBUTING.md	a month ago

<https://github.com/fchollet/keras>

Keras Examples



Features Business Explore Pricing

This repository

Search

Sign in or Sign up

fchollet / keras

Watch

1,018

Star

14,893

Fork

5,181

Code

Issues 2,486

Pull requests 27

Projects 1

Wiki

Pulse

Graphs

Branch: master ▾

keras / examples /

Create new file

Find file

History



Mohanson committed with fchollet Spelling errors (#6232)

Latest commit 5bd3976 11 days ago

..

README.md	Adding mnist_acgan.py example link in README (#4876)	4 months ago
addition_rnn.py	Spelling errors (#6232)	11 days ago
antirectifier.py	Style fix for examples. (#5980)	28 days ago
babimemnn.py	Style fixes in example scripts	a month ago
babirnn.py	Style fixes in example scripts	a month ago
cifar10_cnn.py	fix rmsprop learning rate for convergence (#6182)	17 days ago
conv_filter_visualization.py	Finish updating examples.	a month ago
conv_lstm.py	Update a number of example scripts.	2 months ago
deep_dream.py	Finish updating examples.	a month ago
image_ocr.py	Fixed URL for wordlist.tgz in image_ocr.py (#6136)	20 days ago
imdb_bidirectional_lstm.py	Finish updating examples.	a month ago
imdb_cnn.py	Finish updating examples.	a month ago
imdb_cnn_lstm.py	Style fix for examples. (#5980)	28 days ago

Keras MNIST CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_mnist_cnn.ipynb

jupyter Keras_mnist_cnn Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Help

Code CellToolbar

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

Keras MNIST CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_mnist_cnn.ipynb

jupyter Keras_mnist_cnn Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Python 3

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Using TensorFlow backend.

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
```

Keras MINST CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_mnist_cnn.ipynb

jupyter Keras_mnist_cnn Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Python 3

Using TensorFlow backend.

```
Downloading data from https://s3.amazonaws.com/img-datasets/mnist.npz
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [=====] - 200s - loss: 0.3155 - acc: 0.9028 - val_loss: 0.0756 - val_acc: 0.9761
Epoch 2/12
60000/60000 [=====] - 209s - loss: 0.1106 - acc: 0.9681 - val_loss: 0.0523 - val_acc: 0.9837
Epoch 3/12
60000/60000 [=====] - 220s - loss: 0.0834 - acc: 0.9749 - val_loss: 0.0416 - val_acc: 0.9852
Epoch 4/12
60000/60000 [=====] - 224s - loss: 0.0700 - acc: 0.9795 - val_loss: 0.0392 - val_acc: 0.9879
Epoch 5/12
60000/60000 [=====] - 229s - loss: 0.0614 - acc: 0.9818 - val_loss: 0.0358 - val_acc: 0.9871
Epoch 6/12
60000/60000 [=====] - 227s - loss: 0.0558 - acc: 0.9828 - val_loss: 0.0345 - val_acc: 0.9880
Epoch 7/12
60000/60000 [=====] - 217s - loss: 0.0498 - acc: 0.9850 - val_loss: 0.0337 - val_acc: 0.9883
Epoch 8/12
60000/60000 [=====] - 217s - loss: 0.0473 - acc: 0.9865 - val_loss: 0.0294 - val_acc: 0.9899
Epoch 9/12
60000/60000 [=====] - 217s - loss: 0.0439 - acc: 0.9872 - val_loss: 0.0316 - val_acc: 0.9889
Epoch 10/12
60000/60000 [=====] - 217s - loss: 0.0415 - acc: 0.9871 - val_loss: 0.0319 - val_acc: 0.9897
Epoch 11/12
60000/60000 [=====] - 217s - loss: 0.0380 - acc: 0.9889 - val_loss: 0.0275 - val_acc: 0.9904
Epoch 12/12
60000/60000 [=====] - 215s - loss: 0.0376 - acc: 0.9889 - val_loss: 0.0285 - val_acc: 0.9905
Test loss: 0.0285460013417
Test accuracy: 0.9905
```

Keras MNIST CNN

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Keras MINST CNN

```
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras import backend as K
```

Keras MINST CNN

```
batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)
```

Keras MNIST CNN

```
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Keras MINST CNN

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                activation='relu',
                input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

Keras MINST CNN

```
model.fit(x_train, y_train,
           batch_size=batch_size,
           epochs=epochs,
           verbose=1,
           validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Keras MNIST CNN

```
python mnist_cnn.py
```

Using TensorFlow backend.

Downloading data from <https://s3.amazonaws.com/img-datasets/mnist.npz>

x_train shape: (60000, 28, 28, 1)

60000 train samples

10000 test samples

Train on 60000 samples, validate on 10000 samples

Epoch 1/12

```
60000/60000 [=====] - 108s - loss: 0.3510 - acc: 0.8921 - val_loss: 0.0880 - val_acc: 0.9738
```

Epoch 2/12

```
60000/60000 [=====] - 106s - loss: 0.1200 - acc: 0.9649 - val_loss: 0.0567 - val_acc: 0.9820
```

Epoch 3/12

```
60000/60000 [=====] - 104s - loss: 0.0889 - acc: 0.9735 - val_loss: 0.0438 - val_acc: 0.9856
```

Epoch 4/12

```
60000/60000 [=====] - 106s - loss: 0.0744 - acc: 0.9783 - val_loss: 0.0392 - val_acc: 0.9862
```

Epoch 5/12

```
60000/60000 [=====] - 106s - loss: 0.0648 - acc: 0.9807 - val_loss: 0.0363 - val_acc: 0.9873
```

Epoch 6/12

```
60000/60000 [=====] - 109s - loss: 0.0574 - acc: 0.9840 - val_loss: 0.0348 - val_acc: 0.9884
```

Epoch 7/12

```
60000/60000 [=====] - 104s - loss: 0.0522 - acc: 0.9842 - val_loss: 0.0324 - val_acc: 0.9890
```

Epoch 8/12

```
60000/60000 [=====] - 104s - loss: 0.0484 - acc: 0.9856 - val_loss: 0.0315 - val_acc: 0.9894
```

Epoch 9/12

```
60000/60000 [=====] - 104s - loss: 0.0447 - acc: 0.9870 - val_loss: 0.0296 - val_acc: 0.9902
```

Epoch 10/12

```
60000/60000 [=====] - 109s - loss: 0.0419 - acc: 0.9877 - val_loss: 0.0338 - val_acc: 0.9894
```

Epoch 11/12

```
60000/60000 [=====] - 104s - loss: 0.0405 - acc: 0.9879 - val_loss: 0.0301 - val_acc: 0.9896
```

Epoch 12/12

```
60000/60000 [=====] - 127s - loss: 0.0391 - acc: 0.9883 - val_loss: 0.0304 - val_acc: 0.9899
```

Test loss: 0.030424870987

Test accuracy: 0.9899

IMDB

Large Movie Review Dataset

- This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets.
- We provide a set of **25,000** highly polar movie reviews for **training**, and **25,000** for **testing**.
- There is additional unlabeled data for use as well.
- Raw text and already processed bag of words formats are provided.
- [Large Movie Review Dataset v1.0](#)
 - http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

IMDB Dataset (Mass et al., 2011)

Features	PL04	Our Dataset	Subjectivity
Bag of Words (bnc)	85.45	87.80	87.77
Bag of Words ($b\Delta t'c$)	85.80	88.23	85.65
LDA	66.70	67.42	66.65
LSA	84.55	83.96	82.82
Our Semantic Only	87.10	87.30	86.65
Our Full	84.65	87.44	86.19
Our Full, Additional Unlabeled	87.05	87.99	87.22
Our Semantic + Bag of Words (bnc)	88.30	88.28	88.58
Our Full + Bag of Words (bnc)	87.85	88.33	88.45
Our Full, Add'l Unlabeled + Bag of Words (bnc)	88.90	88.89	88.13
Bag of Words SVM (Pang and Lee, 2004)	87.15	N/A	90.00
Contextual Valence Shifters (Kennedy and Inkpen, 2006)	86.20	N/A	N/A
tf. Δ idf Weighting (Martineau and Finin, 2009)	88.10	N/A	N/A
Appraisal Taxonomy (Whitelaw et al., 2005)	90.20	N/A	N/A

Table 2: Classification accuracy on three tasks. From left to right the datasets are: A collection of 2,000 movie reviews often used as a benchmark of sentiment classification (Pang and Lee, 2004), 50,000 reviews we gathered from IMDB, and the sentence subjectivity dataset also released by (Pang and Lee, 2004). All tasks are balanced two-class problems.

Keras Github

 Features Business Explore Pricing This repository Search Sign in or Sign up

 Watch 1,018 Star 14,893 Fork 5,180

 Code Issues 2,486 Pull requests 27 Projects 1 Wiki Pulse Graphs

Deep Learning library for Python. Convnets, recurrent neural networks, and more. Runs on TensorFlow or Theano.
<http://keras.io/>

deep-learning tensorflow theano neural-networks machine-learning data-science

3,503 commits 4 branches 28 releases 424 contributors

Branch: master ▾ New pull request Find file Clone or download ▾

Commit	Message	Date
	phipleg committed with fchollet Added logsumexp to backend. (#6346)	Latest commit 7d52af6 a day ago
	Update docker files to TensorFlow 1, Theano 0.9 (#6116)	20 days ago
	fix stateful RNNs FAQ link (#6336)	3 days ago
	Spelling errors (#6232)	11 days ago
	Added logsumexp to backend. (#6346)	a day ago
	Added logsumexp to backend. (#6346)	a day ago
	Fix FAQ question	a month ago
	Update Travis config	9 days ago
	Mention requests for contribution in CONTRIBUTING.md	a month ago

<https://github.com/fchollet/keras>

240

Keras Examples



Features Business Explore Pricing

This repository

Search

Sign in or Sign up

fchollet / keras

Watch

1,018

Star

14,893

Fork

5,181

Code

Issues 2,486

Pull requests 27

Projects 1

Wiki

Pulse

Graphs

Branch: master ▾

keras / examples /

Create new file

Find file

History



Mohanson committed with fchollet Spelling errors (#6232)

Latest commit 5bd3976 11 days ago

..

README.md	Adding mnist_acgan.py example link in README (#4876)	4 months ago
addition_rnn.py	Spelling errors (#6232)	11 days ago
antirectifier.py	Style fix for examples. (#5980)	28 days ago
babimemnn.py	Style fixes in example scripts	a month ago
babirnn.py	Style fixes in example scripts	a month ago
cifar10_cnn.py	fix rmsprop learning rate for convergence (#6182)	17 days ago
conv_filter_visualization.py	Finish updating examples.	a month ago
conv_lstm.py	Update a number of example scripts.	2 months ago
deep_dream.py	Finish updating examples.	a month ago
image_ocr.py	Fixed URL for wordlist.tgz in image_ocr.py (#6136)	20 days ago
imdb_bidirectional_lstm.py	Finish updating examples.	a month ago
imdb_cnn.py	Finish updating examples.	a month ago
imdb_cnn_lstm.py	Style fix for examples. (#5980)	28 days ago

Keras Examples

- [imdb_bidirectional_lstm.py](#) Trains a Bidirectional LSTM on the IMDB sentiment classification task.
- [imdb_cnn.py](#) Demonstrates the use of Convolution1D for text classification.
- [imdb_cnn_lstm.py](#) Trains a convolutional stack followed by a recurrent stack network on the IMDB sentiment classification task.
- [imdb_fasttext.py](#) Trains a FastText model on the IMDB sentiment classification task.
- [imdb_lstm.py](#) Trains a LSTM on the IMDB sentiment classification task.
- [lstm_benchmark.py](#) Compares different LSTM implementations on the IMDB sentiment classification task.
- [lstm_text_generation.py](#) Generates text from Nietzsche's writings.

Keras IMDB Movie reviews

sentiment classification

- Dataset of 25,000 movies reviews from IMDB, labeled by sentiment (positive/negative).
- Reviews have been preprocessed, and each review is encoded as a sequence of word indexes (integers).
- For convenience, words are indexed by overall frequency in the dataset, so that for instance the integer "3" encodes the 3rd most frequent word in the data.
- This allows for quick filtering operations such as: "only consider the top 10,000 most common words, but eliminate the top 20 most common words".
- As a convention, "0" does not stand for a specific word, but instead is used to encode any unknown word.

Keras IMDB load_data

```
def load_data(path='imdb.npz',
              num_words=None,
              skip_top=0,
              maxlen=None,
              seed=113,
              start_char=1,
              oov_char=2,
              index_from=3):
    path = get_file(
        path, origin='https://s3.amazonaws.com/text-datasets/imdb.npz')
    f = np.load(path)
    x_train = f['x_train']
    labels_train = f['y_train']
    x_test = f['x_test']
    labels_test = f['y_test']
    f.close()
```

Keras IMDB get_word_index

```
def get_word_index(path='imdb_word_index.json'):
    path = get_file(
        path,
        origin='https://s3.amazonaws.com/text-datasets/imdb_word_index.json')
    f = open(path)
    data = json.load(f)
    f.close()
    return data
```

Keras IMDB CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_imdb_cnn.ipynb

jupyter Keras_imdb_cnn Last Checkpoint: 15 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Python 3

```
from __future__ import print_function
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers import Conv1D, GlobalMaxPooling1D
from keras.datasets import imdb

# set parameters:
max_features = 5000
 maxlen = 400
batch_size = 32
embedding_dims = 50
filters = 250
kernel_size = 3
hidden_dims = 250
epochs = 2

print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
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()

# we start off with an efficient embedding layer which maps
# our vocab indices into embedding_dims dimensions
model.add(Embedding(max_features,
                    embedding_dims,
```

Keras IMDB CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_imdb_cnn.ipynb

jupyter Keras_imdb_cnn Last Checkpoint: 19 minutes ago (autosaved)



File Edit View Insert Cell Kernel Widgets Help

Python 3

```
model.add(Embedding(max_features,
                     embedding_dims,
                     input_length=maxlen))
model.add(Dropout(0.2))

# we add a Convolution1D, which will learn filters
# word group filters of size filter_length:
model.add(Conv1D(filters,
                 kernel_size,
                 padding='valid',
                 activation='relu',
                 strides=1))
# we use max pooling:
model.add(GlobalMaxPooling1D())

# We add a vanilla hidden layer:
model.add(Dense(hidden_dims))
model.add(Dropout(0.2))
model.add(Activation('relu'))

# We project onto a single unit output layer, and squash it with a sigmoid:
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test))
```

Using TensorFlow backend.

Loading data...

Downloading data from <https://s3.amazonaws.com/text-datasets/imdb.npz>
25000 train sequences

Keras IMDB CNN

localhost:8888/notebooks/Documents/SCDBA/DL/Keras_imdb_cnn.ipynb

jupyter Keras_imdb_cnn Last Checkpoint: 13 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help

Code CellToolbar

```
# we use max pooling:  
model.add(GlobalMaxPooling1D())  
  
# We add a vanilla hidden layer:  
model.add(Dense(hidden_dims))  
model.add(Dropout(0.2))  
model.add(Activation('relu'))  
  
# We project onto a single unit output layer, and squash it with a sigmoid:  
model.add(Dense(1))  
model.add(Activation('sigmoid'))  
  
model.compile(loss='binary_crossentropy',  
              optimizer='adam',  
              metrics=['accuracy'])  
model.fit(x_train, y_train,  
          batch_size=batch_size,  
          epochs=epochs,  
          validation_data=(x_test, y_test))
```

Using TensorFlow backend.

```
Loading data...  
Downloading data from https://s3.amazonaws.com/text-datasets/imdb.npz  
25000 train sequences  
25000 test sequences  
Pad sequences (samples x time)  
x_train shape: (25000, 400)  
x_test shape: (25000, 400)  
Build model...  
Train on 25000 samples, validate on 25000 samples  
Epoch 1/2  
25000/25000 [=====] - 266s - loss: 0.4110 - acc: 0.8012 - val_loss: 0.2965 - val_acc: 0.8739  
Epoch 2/2  
25000/25000 [=====] - 286s - loss: 0.2429 - acc: 0.9020 - val_loss: 0.2726 - val_acc: 0.8862
```

Out[1]: <keras.callbacks.History at 0x11dc37b00>

Keras IMDB CNN

```
python imdb_cnn.py
Using TensorFlow backend.
Loading data...
Downloading data from https://s3.amazonaws.com/text-datasets/imdb.npz
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 400)
x_test shape: (25000, 400)
Build model...
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
25000/25000 [=====] - 157s - loss: 0.4050 - acc: 0.8065 - val_loss: 0.2924 - val_acc: 0.8750
Epoch 2/2
25000/25000 [=====] - 128s - loss: 0.2433 - acc: 0.9040 - val_loss: 0.2701 - val_acc: 0.8865
Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.python.client.session.Session object at 0x0000019F153C2A20>>
Traceback (most recent call last):
  File "C:\Program Files\Anaconda3\lib\site-packages\tensorflow\python\client\session.py", line 587, in __del__
AttributeError: 'NoneType' object has no attribute 'TF_NewStatus'
```

Keras IMDB LSTM

```
from __future__ import print_function
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
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(num_words=max_features)
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))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

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

print('Train...')
model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=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 IMDB LSTM

```
from __future__ import print_function
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb
```

Keras IMDB LSTM

```
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(num_words=max_features)
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 IMDB LSTM

```
print('Build model...')

model = Sequential()
model.add(Embedding(max_features, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

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

Keras IMDB LSTM

```
print('Train...')
model.fit(x_train, y_train,
           batch_size=batch_size,
           epochs=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)
```

```
python imdb_lstm.py
Using TensorFlow backend.
Loading data...
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 80)
x_test shape: (25000, 80)
Build model...
Train...
Train on 25000 samples, validate on 25000 samples
Epoch 1/15
25000/25000 [=====] - 111s - loss: 0.4561 - acc: 0.7837 - val_loss: 0.3892 - val_acc: 0.8275
Epoch 2/15
25000/25000 [=====] - 112s - loss: 0.2947 - acc: 0.8792 - val_loss: 0.4266 - val_acc: 0.8353
Epoch 3/15
25000/25000 [=====] - 111s - loss: 0.2122 - acc: 0.9178 - val_loss: 0.4133 - val_acc: 0.8284
Epoch 4/15
25000/25000 [=====] - 112s - loss: 0.1461 - acc: 0.9450 - val_loss: 0.4670 - val_acc: 0.8260
Epoch 5/15
25000/25000 [=====] - 113s - loss: 0.1038 - acc: 0.9633 - val_loss: 0.5580 - val_acc: 0.8203
Epoch 6/15
25000/25000 [=====] - 113s - loss: 0.0739 - acc: 0.9749 - val_loss: 0.6738 - val_acc: 0.8174
Epoch 7/15
25000/25000 [=====] - 113s - loss: 0.0542 - acc: 0.9810 - val_loss: 0.7463 - val_acc: 0.8154
Epoch 8/15
25000/25000 [=====] - 113s - loss: 0.0428 - acc: 0.9856 - val_loss: 0.8131 - val_acc: 0.8157
Epoch 9/15
25000/25000 [=====] - 115s - loss: 0.0334 - acc: 0.9889 - val_loss: 0.8566 - val_acc: 0.8165
Epoch 10/15
25000/25000 [=====] - 114s - loss: 0.0248 - acc: 0.9920 - val_loss: 0.9186 - val_acc: 0.8165
Epoch 11/15
25000/25000 [=====] - 116s - loss: 0.0156 - acc: 0.9955 - val_loss: 0.9016 - val_acc: 0.8082
Epoch 12/15
25000/25000 [=====] - 117s - loss: 0.0196 - acc: 0.9942 - val_loss: 0.9720 - val_acc: 0.8124
Epoch 13/15
25000/25000 [=====] - 120s - loss: 0.0152 - acc: 0.9957 - val_loss: 1.0064 - val_acc: 0.8148
Epoch 14/15
25000/25000 [=====] - 121s - loss: 0.0128 - acc: 0.9961 - val_loss: 1.1103 - val_acc: 0.8121
Epoch 15/15
25000/25000 [=====] - 114s - loss: 0.0110 - acc: 0.9970 - val_loss: 1.0173 - val_acc: 0.8132
25000/25000 [=====] - 23s
Test score: 1.01734088922
Test accuracy: 0.8132
```

Keras IMDB LSTM

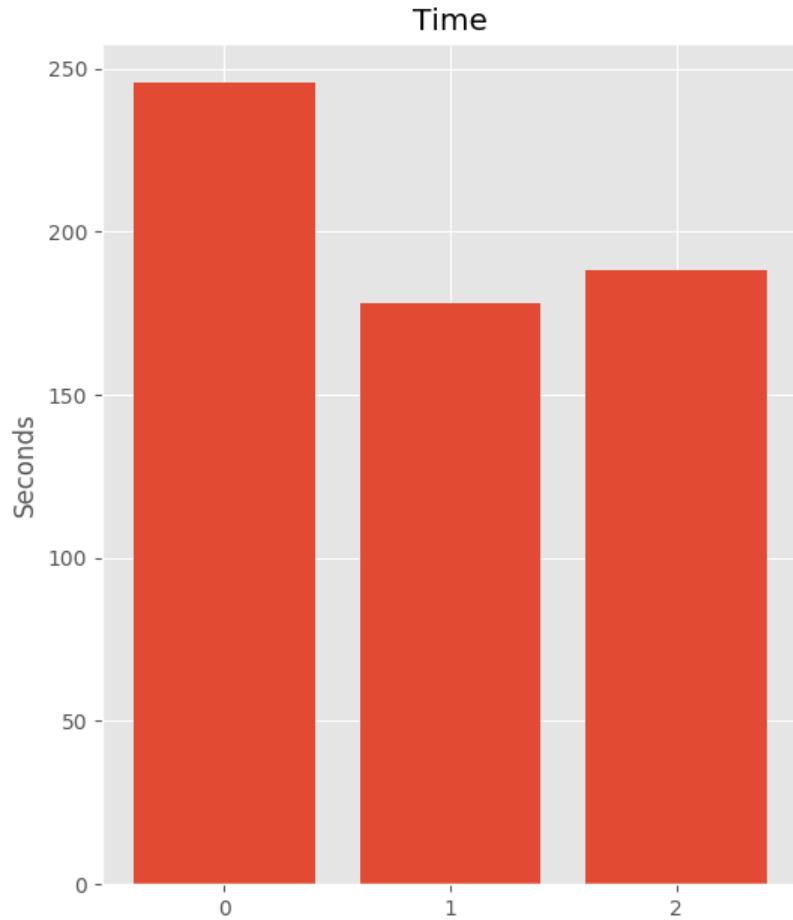
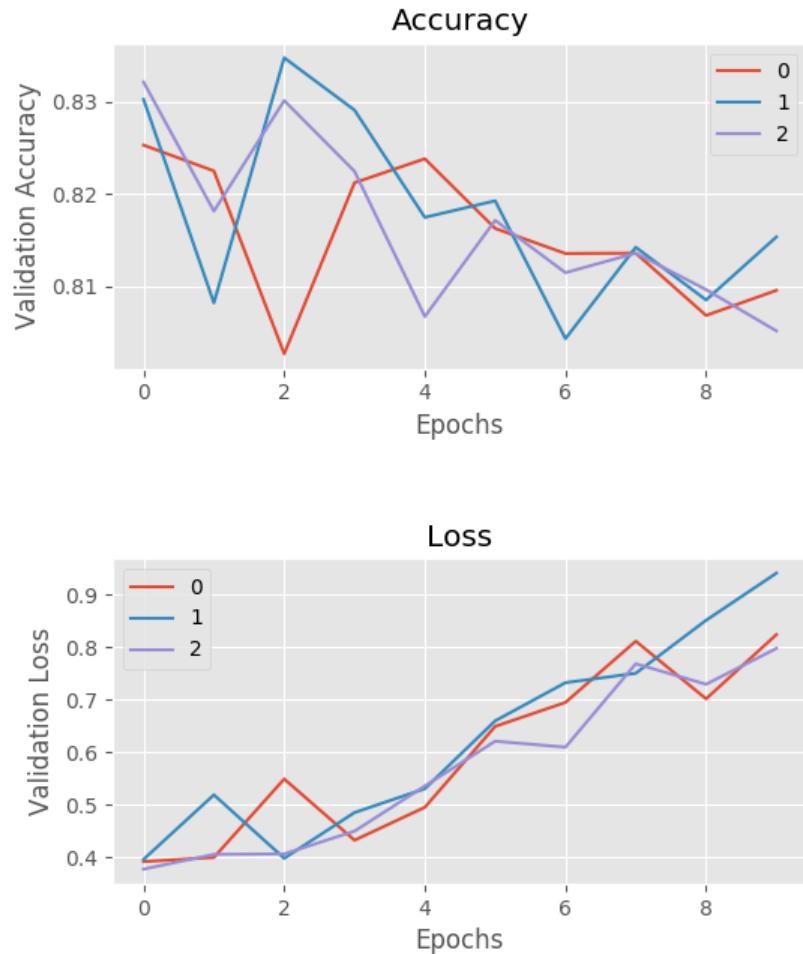
Keras IMDB FastText

```
python imdb_fasttext.py
Using TensorFlow backend.
Loading data...
25000 train sequences
25000 test sequences
Average train sequence length: 238
Average test sequence length: 230
Pad sequences (samples x time)
x_train shape: (25000, 400)
x_test shape: (25000, 400)
Build model...
Train on 25000 samples, validate on 25000 samples
Epoch 1/5
25000/25000 [=====] - 14s - loss: 0.6102 - acc: 0.7397 - val_loss: 0.5034 - val_acc: 0.8105
Epoch 2/5
25000/25000 [=====] - 14s - loss: 0.4019 - acc: 0.8656 - val_loss: 0.3697 - val_acc: 0.8654
Epoch 3/5
25000/25000 [=====] - 14s - loss: 0.3025 - acc: 0.8959 - val_loss: 0.3199 - val_acc: 0.8791
Epoch 4/5
25000/25000 [=====] - 14s - loss: 0.2521 - acc: 0.9113 - val_loss: 0.2971 - val_acc: 0.8848
Epoch 5/5
25000/25000 [=====] - 14s - loss: 0.2181 - acc: 0.9249 - val_loss: 0.2899 - val_acc: 0.8855
Exception ignored in: <bound method BaseSession.__del__ of <tensorflow.python.client.session.Session object at
0x000001E3257DB438>
Traceback (most recent call last):
  File "C:\Program Files\Anaconda3\lib\site-packages\tensorflow\python\client\session.py", line 587, in __del__
AttributeError: 'NoneType' object has no attribute 'TF_NewStatus'
```

Keras IMDB CNN LSTM

```
python imdb_cnn_lstm_2.py
Using TensorFlow backend.
Loading data...
25000 train sequences
25000 test sequences
Pad sequences (samples x time)
x_train shape: (25000, 100)
x_test shape: (25000, 100)
Build model...
Train...
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
25000/25000 [=====] - 64s - loss: 0.3824 - acc: 0.8238 - val_loss: 0.3591 - val_acc: 0.8467
Epoch 2/2
25000/25000 [=====] - 63s - loss: 0.1953 - acc: 0.9261 - val_loss: 0.3827 - val_acc: 0.8488
24990/25000 [=====>.] - ETA: 0s
Test score: 0.382728585386
Test accuracy: 0.848799994493
```

Keras LSTM Benchmark



imdb_lstm_2.py

```
from __future__ import print_function

from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Embedding
from keras.layers import LSTM
from keras.datasets import imdb

py_filename = 'imdb_lstm_2.py'
max_features = 20000
 maxlen = 80 # cut texts after this number of words (among top max_features
most common words)
batch_size = 32
epochs = 20 #60

#%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

import codecs
import datetime
import timeit
timer_start = timeit.default_timer()
#timer_end = timeit.default_timer()
#print('timer_end - timer_start', timer_end - timer_start)
```

imdb_lstm_2.py

```
def getDateTimeNow():
    strnow = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
    return strnow

def read_file_utf8(filename):
    with codecs.open(filename, 'r', encoding='utf-8') as f:
        text = f.read()
    return text

def write_file_utf8(filename, text):
    with codecs.open(filename, 'w', encoding='utf-8') as f:
        f.write(text)
    f.close()

def log_file_utf8(filename, text):
    with codecs.open(filename, 'a', encoding='utf-8') as f:
        #append file
        f.write(text + '\n')
    f.close()

log_file_utf8("logfile.txt", '***** ' + py_filename + ' *****')
log_file_utf8("logfile.txt", '***** Start DateTime: ' + getDateTimeNow())

print('Start: ', datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))
```

imdb_lstm_2.py

```
print('Loading data...')
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=max_features)
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))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

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

imdb_lstm_2.py

```
print('Train...')  
print('model.fit: ', datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))  
history = model.fit(x_train, y_train,  
                      batch_size = batch_size,  
                      epochs = epochs,  
                      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)
```

imdb_lstm_2.py

```
timer_end = timeit.default_timer()
print('Timer: ', str(round(timer_end - timer_start, 2)), 's')
print('DateTime: ', datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))
log_file_utf8("logfile.txt", 'Timer: ' + str(round(timer_end - timer_start, 2))
+ ' s')
log_file_utf8("logfile.txt", '***** End Datetime: ' +
datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))

# summarize history for accuracy
#http://machinelearningmastery.com/display-deep-learning-model-training-history-in-keras/
print('history.history.keys():', history.history.keys())
print('history.history:', history.history)
log_file_utf8("logfile.txt", 'history.history:' + str(history.history))
```

imdb_lstm_2.py

```
# Deep Learning Training Visualization
plt.figure(figsize=(10, 8))  # make separate figure
ax1 = plt.subplot(2, 1, 1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
ax1.xaxis.set_major_locator(plt.NullLocator())
# plt.xlabel('epoch')
plt.legend(['train acc', 'test val_acc'], loc='upper left')
# plt.show()
ax2 = plt.subplot(2, 1, 2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train loss', 'test val_loss'], loc='upper left')
plt.savefig("training_accuracy_loss_" + py_filename + "_" + str(epochs) +
".png", dpi= 300)
```

imdb_lstm_2.py

```
#Log File for Deep Learning Summary Analysis
log_file_utf8("logfile.txt", 'DL_Summary:\t' + py_filename +
    '\tepochs\t' + str(epochs) +
    '\tscore\t' + str(score) +
    '\taccuracy\t' + str(acc) +
    '\tTimer\t' + str(round(timer_end - timer_start, 2)) +
    '\thistory\t' + str(history.history))
plt.show()
```

python filename.py

```
python imdb_fasttext_2.py
```

```
python imdb_cnn_2.py
```

```
python imdb_lstm_2.py
```

```
python imdb_cnn_lstm_2.py
```

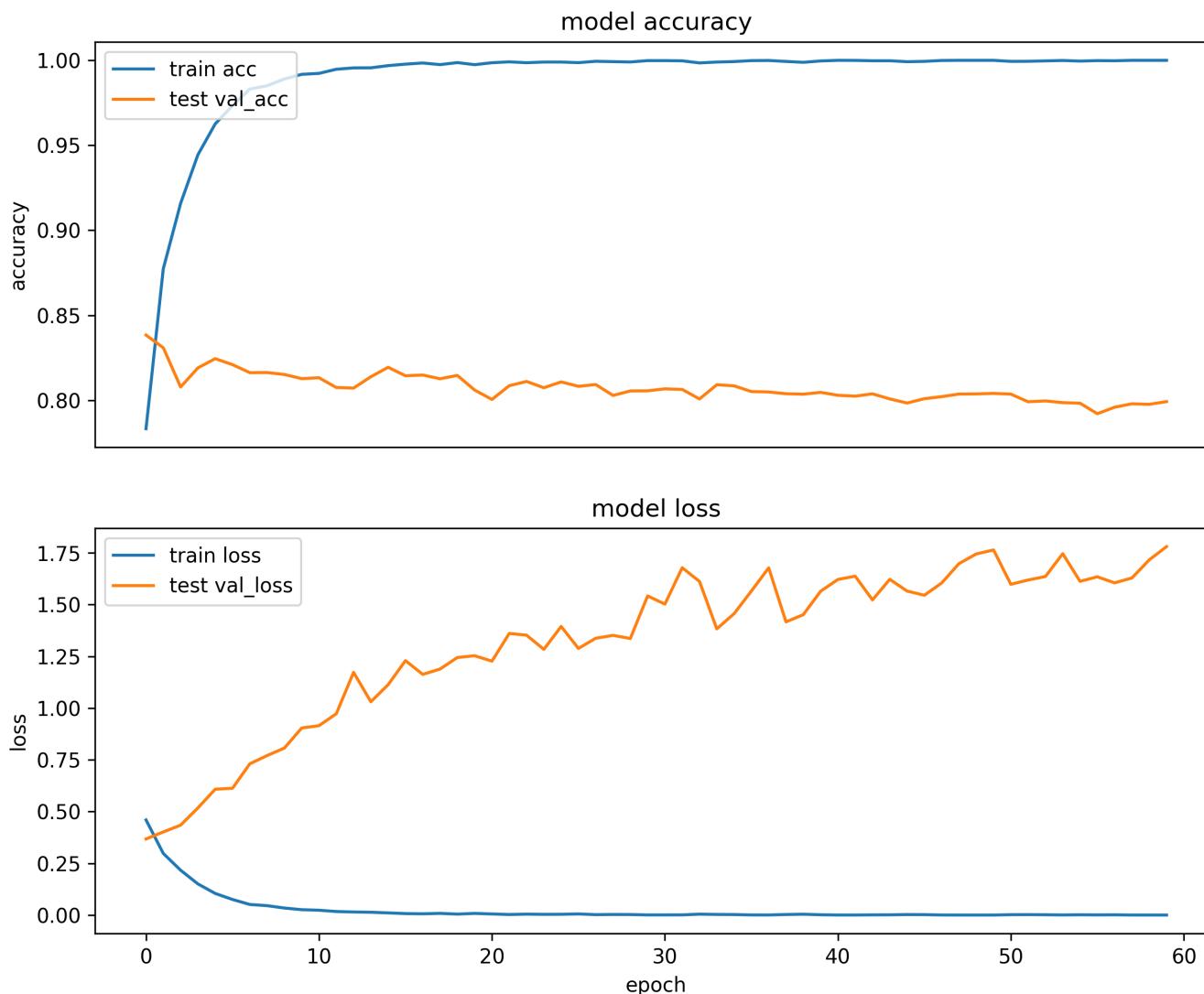
```
python imdb_bidirectional_lstm_2.py
```

Deep Learning Summary

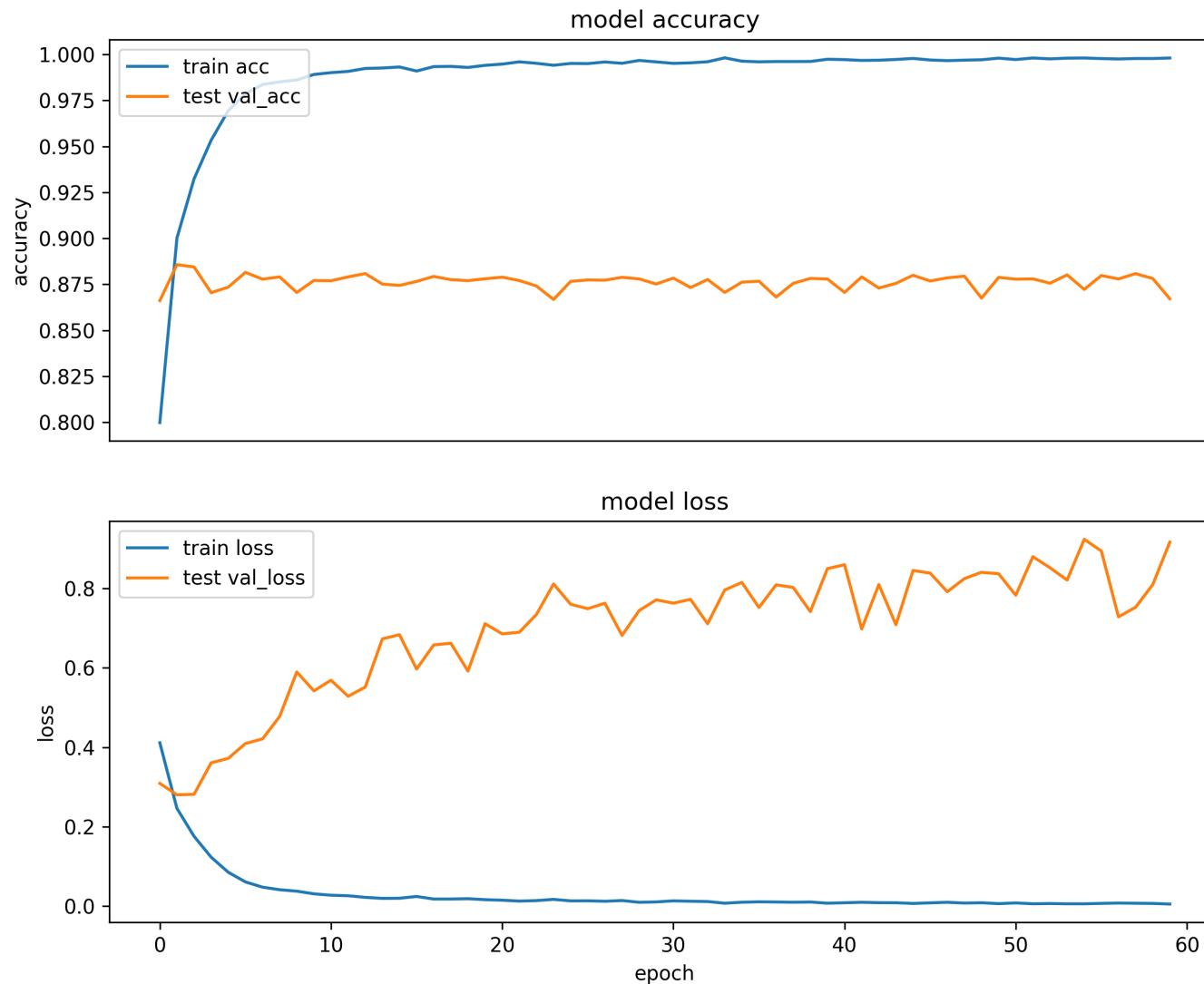
```
#Log File for Deep Learning Summary Analysis
log_file_utf8("logfile.txt", 'DL_Summary:\t' + py_filename +
    '\tepochs\t' + str(epochs) +
    '\tscore\t' + str(score) +
    '\taccuracy\t' + str(acc) +
    '\tTimer\t' + str(round(timer_end - timer_start, 2)) +
    '\thistory\t' + str(history.history))
```

Model	epochs	Score	Accuracy	Timer (s)
imdb_lstm_2.py	30	0.6440	0.8540	682.57
imdb_cnn_2.py	30	0.7186	0.8775	4320.38
imdb_lstm_2.py	30	1.5716	0.8052	3958.93
imdb_cnn_lstm_2.py	30	1.3105	0.8240	2471.65
imdb_bidirectional_lstm_2.py	30	1.4083	0.8255	4344.36
imdb_fasttext_2.py	30	0.6439	0.8540	1117.78
imdb_fasttext_2.py	60	1.2335	0.8407	1297.02
imdb_cnn_2.py	60	0.9170	0.8672	8507.48
imdb_lstm_2.py	60	1.7803	0.7992	8039.67
imdb_cnn_lstm_2.py	60	1.4623	0.8137	4912.25
imdb_bidirectional_lstm_2.py	60	1.8975	0.8138	8589.17

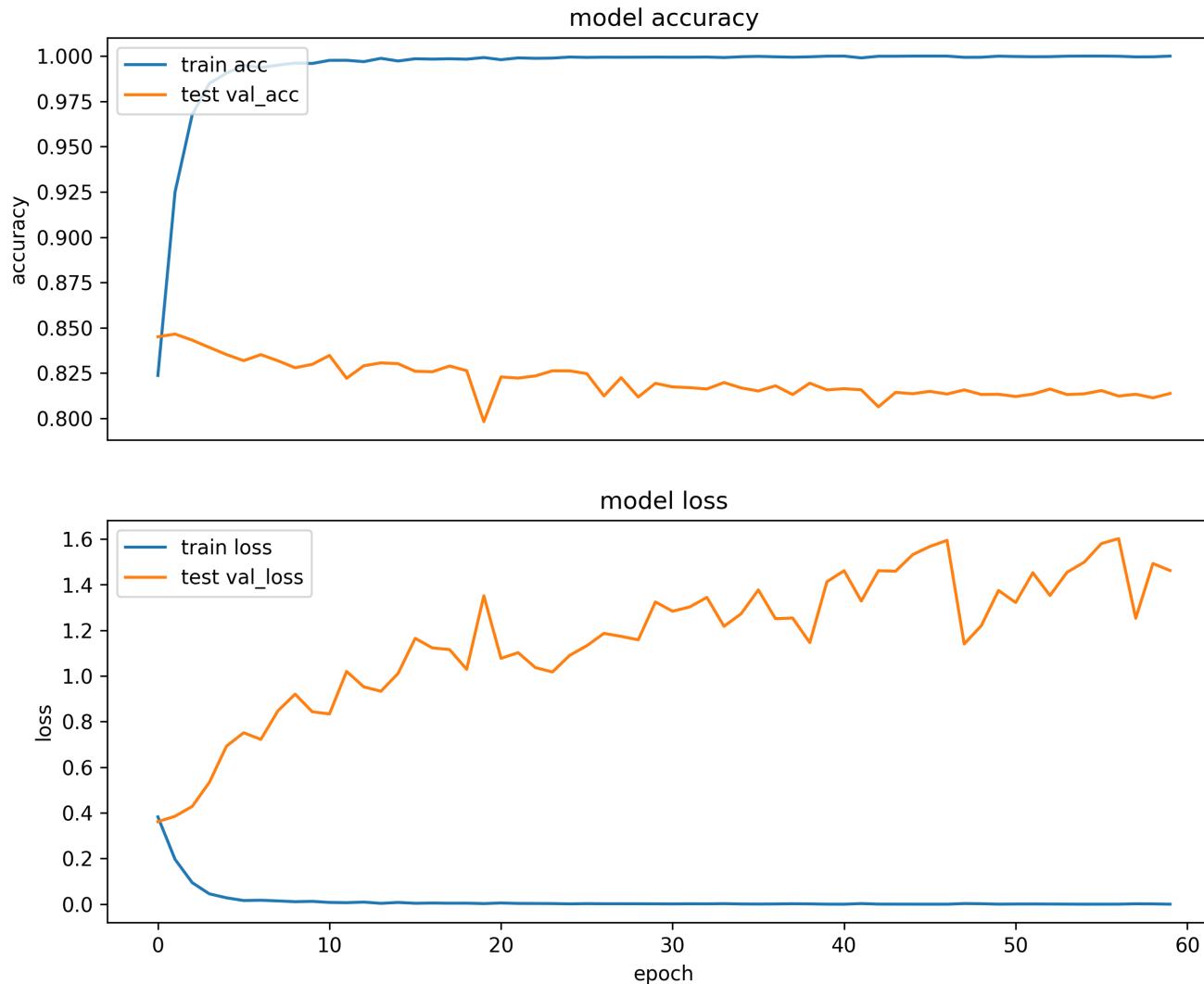
imdb_lstm_2.py



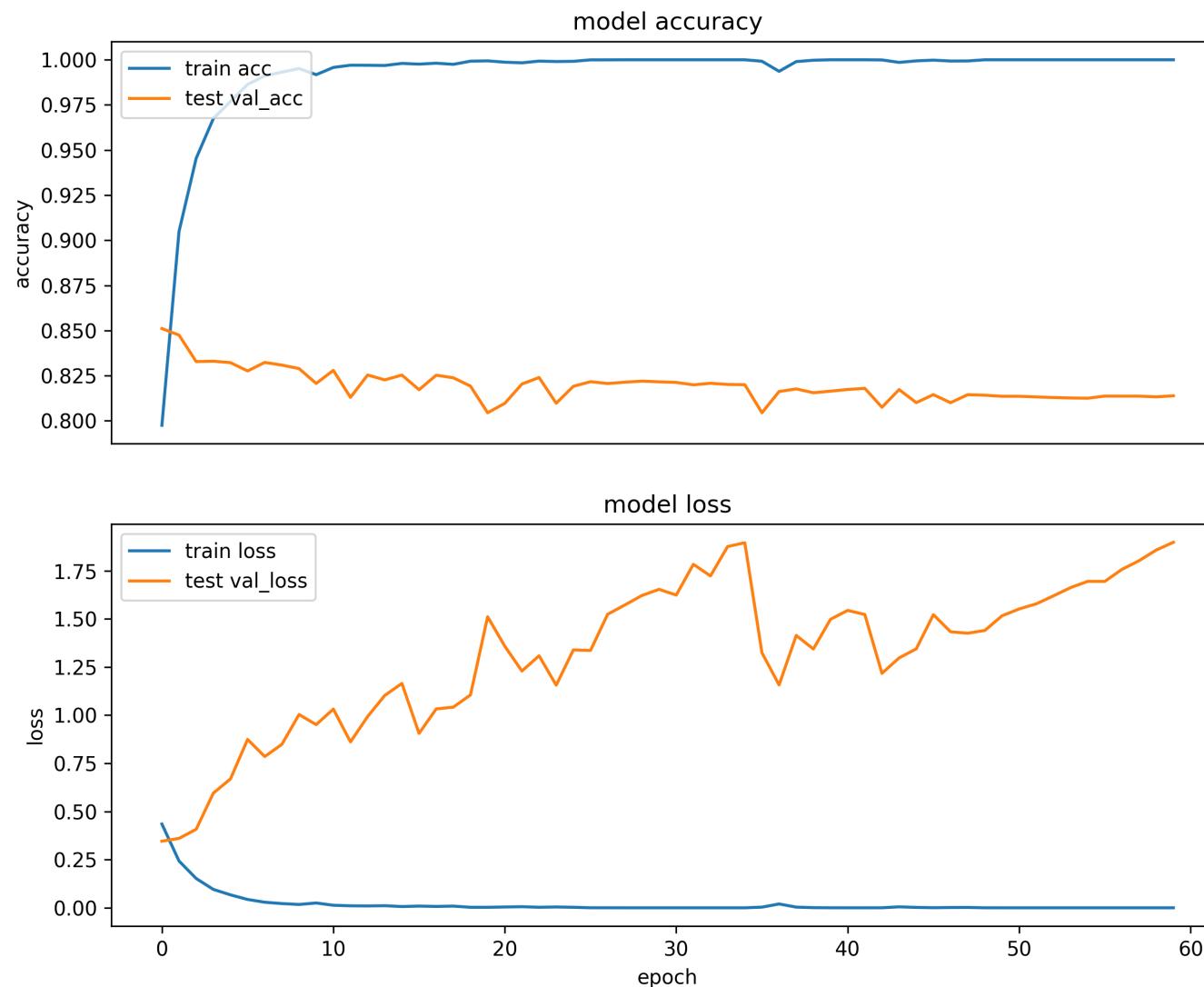
imdb_cnn_2.py



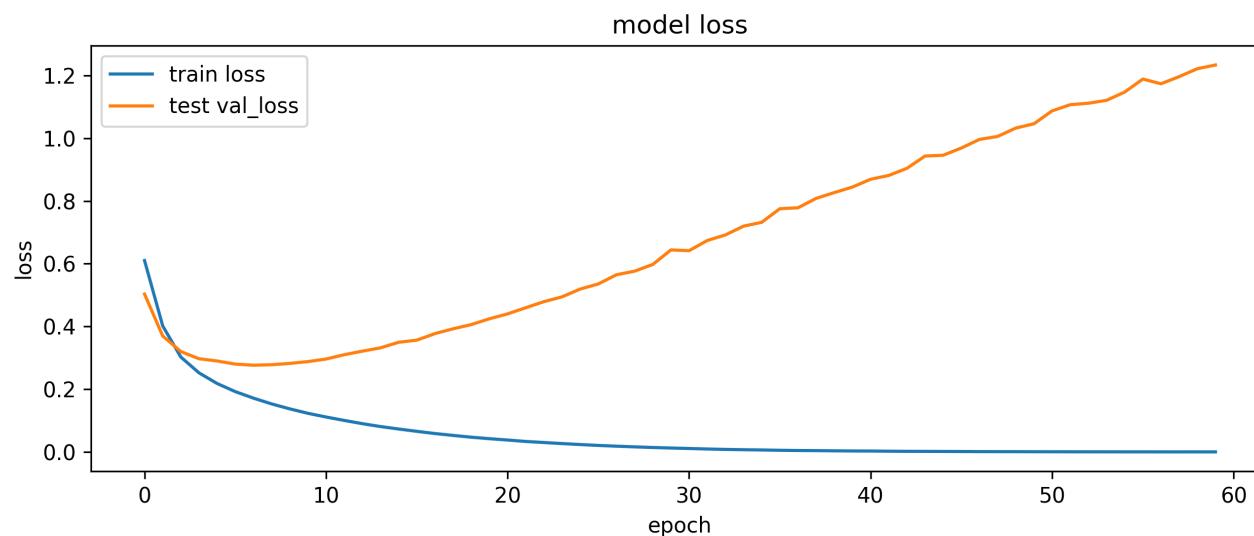
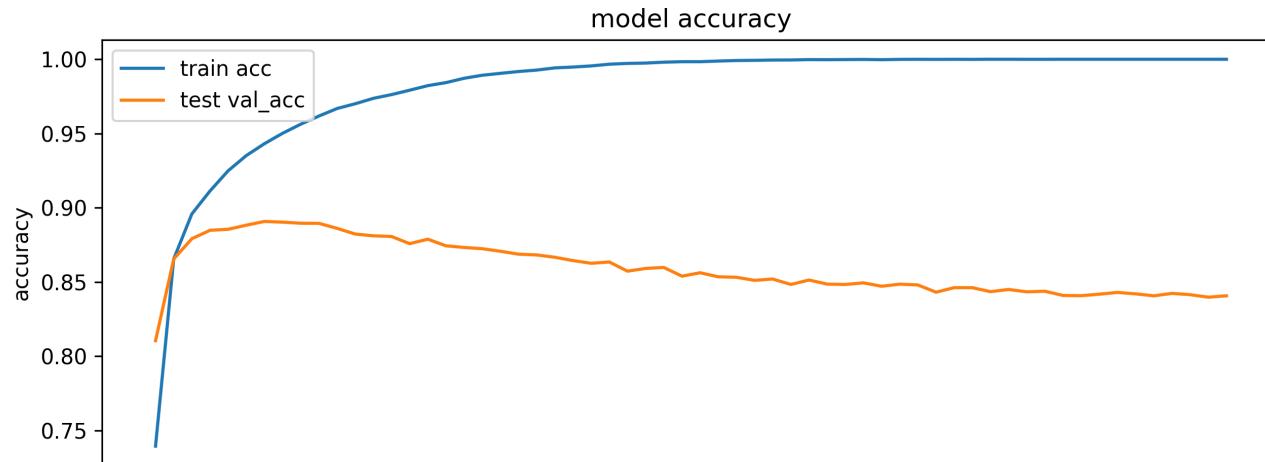
imdb_cnn_lstm_2.py



imdb_bidirectional_lstm_2.py



imdb_fasttext_2.py



Deep Learning with CPU vs. GPU

Timings:

Hardware	Backend	Time / Epoch

CPU	TF	3 hrs
Titan X (maxwell)	TF	4 min
Titan X (maxwell)	TH	7 min

Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming

[HOME](#)[FEATURES](#)[PRICING](#)[CONTACTS](#)[VIDEOS](#)[SUPPORT](#)[REGISTER](#)[LOGIN](#)

Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming

[Register Free Account](#)[LAUNCH APP](#)

Deep Learning Studio β - beta

CIFAR-10 - Object Recognition in Images



mandeep2



Data

Model

HyperParameters

Training

Results

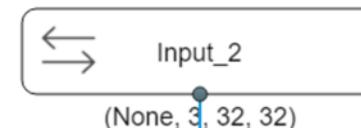
Inference

 Show Advance Layers[Pre-Trained Models](#)

InceptionV3



VGG19



Instantiate the vgg19 architecture, [more...](#)

Include Top Fully Connect...

false

trainable

10

<http://deepcognition.ai/>

274

Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming

Deep Cognition Deep Learning Studio β - beta

CIFAR-10 - Object Recognition in Images

Data Model HyperParameters Training Results Inference

Show Advance Layers

Pre-Trained Models

- InceptionV3
- VGG19
- VGG16
- ResNet50

Special Functions

Convolutional Layers

Core Layers

Pooling Layers

Recurrent Layers

Advanced Activations Layers

Convolutional Recurrent Layers

Noise Layers

```
graph TD; Input_2[Input_2  
None, 3, 32, 32] --> VGG19_1[VGG19_1  
None, 512, 1, 1]; VGG19_1 --> Flatten_2[Flatten_2  
None, 512]; Flatten_2 --> Dense_5[Dense_5  
None, 100]; Dense_5 --> Dropout_1[Dropout_1  
None, 100]; Dropout_1 --> Dense_3[Dense_3  
None, 10]; Dense_3 --> Output_2[Output_2  
None, 10]
```

Instantiate the vgg19 architecture, more...
Include Top Fully Connect...
false
trainable
10
Show Advance Options

Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming

eta

MNIST Handwritten Digits Classifier

Model HyperParameters Training Results

Dataset Source: Testing

Training Run: Run0 Start Inference or Download Trained Model

Digit Label	Image	predictions
• 9	9	• 9
• 1	1	• 1
• 1	1	• 1
• 5	5	• 3
• 0	0	• 0
• 5	5	• 5
• 1	1	• 1
• 2	2	• 6
• 2	2	• 2
• 3	3	• 3

« Previous 1 2 3 4 5 ... 351 Next »

Download Results

References

- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 1 (Google Cloud Next '17),
<https://www.youtube.com/watch?v=u4alGiomYP4>
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17),
<https://www.youtube.com/watch?v=fTUwdXUFFI8>
- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD,
<https://goo.gl/pHeXe7>, <https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist>
- Deep Learning Basics: Neural Networks Demystified,
<https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU>
- Deep Learning SIMPLIFIED,
<https://www.youtube.com/playlist?list=PLjJh1vISEYgvGod9wWiydumYI8hOXixNu>
- TensorFlow: <https://www.tensorflow.org/>
- Theano: <http://deeplearning.net/software/theano/>
- Keras: <http://keras.io/>
- Deep Learning Studio: Cloud platform for designing Deep Learning AI without programming,
<http://deepcognition.ai/>
- Natural Language Processing with Deep Learning (Winter 2017),
https://www.youtube.com/playlist?list=PL3FW7Lu3i5Jsnh1rnUwq_TcylNr7EkRe6
- Udacity, Deep Learning,
https://www.youtube.com/playlist?list=PLAwxtw4SYaPn_OWPFT9ulXLuQrlmzHfOV
- <http://p.migdal.pl/2017/04/30/teaching-deep-learning.html>
- <https://github.com/leriomaggio/deep-learning-keras-tensorflow>