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
Dept. of Information Management, Tamkang University

http://mail.tku.edu.tw/myday/
2017-05-11
## 課程大綱 (Syllabus)

<table>
<thead>
<tr>
<th>週次 (Week)</th>
<th>日期 (Date)</th>
<th>內容 (Subject/Topics)</th>
</tr>
</thead>
</table>
| 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典型、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) |
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)
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

Source: https://www.i-scoop.eu/artificial-intelligence-cognitive-computing/
Artificial Intelligence (AI)
Intelligent Document Recognition algorithms

Source: https://www.i-scoop.eu/artificial-intelligence-cognitive-computing/
Deep Dream

[Link to GitHub repository](https://github.com/fchollet/keras/blob/master/examples/deep_dream.py)
LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton.

"Deep learning."

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, conventional models were trained on hand-designed features, which were time-consuming and often led to suboptimal performance. Deep learning, on the other hand, automatically learns features from raw data, enabling it to achieve state-of-the-art performance in a wide range of tasks. This capability has led to significant advancements in areas such as computer vision, natural language processing, and speech recognition.

Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding, particularly topic classification, sentiment analysis, question answering and language translation.
Machine Learning Models

- Deep Learning
- Association rules
- Decision tree
- Clustering
- Bayesian
- Kernel
- Ensemble
- Dimensionality reduction
- Regression Analysis
- Instance based

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Neural networks (NN) 1960

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Multilayer Perceptrons (MLP) 1985

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Support Vector Machine (SVM) 1995

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Hinton presents the Deep Belief Network (DBN)
New interests in deep learning and RBM
State of the art MNIST 2005

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Deep Recurrent Neural Network (RNN) 2009

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Convolutional DBN 2010
Max-Pooling CDBN 2011

Source: Sunila Gollapudi (2016), Practical Machine Learning, Packt Publishing
Neural Networks

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

Source: https://www.youtube.com/watch?v=bxet-V8XR&index=1&list=PLiaHhY2lBX9hdHaRtr6b7XevZtZRa1PoU
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.

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

Course Description

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

http://cs224d.stanford.edu/
Recurrent Neural Networks (RNNs)
RNN

\[ h_t \]

\[ X_t \]

\[ h_0 \]

\[ h_1 \]

\[ h_2 \]

\[ h_t \]

\[ X_0 \]

\[ X_1 \]

\[ X_2 \]

\[ X_t \]

\[ \ldots \]
RNN long-term dependencies

I grew up in France… I speak fluent French.

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
RNN LSTM

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Long Short Term Memory (LSTM)

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Gated Recurrent Unit (GRU)

$h_{t-1}$

$h_t$

$x_t$

$x$

$1-$

$\sigma$

$tanh$

$\sigma$

$r_t$

$z_t$

$\tilde{h}_t$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
LSTM vs GRU

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.

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

Source: https://github.com/Vict0rSch/deep_learning/tree/master/keras/recurrent
The Sequence to Sequence model (seq2seq)

ENCODER

Incoming Email

Are you free tomorrow?

DECODER

Reply

Yes, what's up? <END>

<START>
Neural Networks

Source: https://www.youtube.com/watch?v=bx2T-V8XR&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

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

X1  

X2  

Source: https://www.youtube.com/watch?v=bx2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)

Hidden Layers (H)

Output Layer (Y)

Deep Neural Networks
Deep Learning

Source: https://www.youtube.com/watch?v=bxzT-V8XR&index=1&list=PLiaHhY2lBX9hdHar6b7XevZtgZRa1PoU
Neural Networks

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

Source: https://www.youtube.com/watch?v=bxelT-V8XR8&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neuron and Synapse

Source: https://en.wikipedia.org/wiki/Neuron
Neural Networks

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

Score

Hours
Sleep

Hours
Study

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Neural Networks

Input Layer (X)

Hidden Layer (H)

Output Layer (Y)

Source: https://www.youtube.com/watch?v=P2HPcj8lRJE&list=PLjJh1vlSEYgvGod9wWiydumYl8hOXixNu&index=2
Convolutional Neural Networks

(CNNs / ConvNets)

http://cs231n.github.io/convolutional-networks/
A regular 3-layer Neural Network

http://cs231n.github.io/convolutional-networks/
A ConvNet arranges its neurons in three dimensions (width, height, depth)

http://cs231n.github.io/convolutional-networks/
The activations of an example ConvNet architecture.
ConvNets

http://cs231n.github.io/convolutional-networks/
ConvNets

http://cs231n.github.io/convolutional-networks/
ConvNets

http://cs231n.github.io/convolutional-networks/
ConvNets
max pooling

http://cs231n.github.io/convolutional-networks/
Convolutional Neural Networks (CNN) (LeNet)
Sparse Connectivity

Source: http://deeplearning.net/tutorial/lenet.html
Convolutional Neural Networks (CNN) (LeNet)

Shared Weights

Source: http://deeplearning.net/tutorial/lenet.html
Convolutional Neural Networks (CNN) (LeNet)

example of a convolutional layer

Source: http://deeplearning.net/tutorial/lenet.html
Convolutional Neural Networks (CNN) (LeNet)

Source: [http://deeplearning.net/tutorial/lenet.html](http://deeplearning.net/tutorial/lenet.html)
show flights from Boston to New York today
Recurrent Neural Networks with Word Embeddings
Semantic Parsing / Slot-Filling
(Spoken Language Understanding)

<table>
<thead>
<tr>
<th>Input (words)</th>
<th>show</th>
<th>flights</th>
<th>from</th>
<th>Boston</th>
<th>to</th>
<th>New</th>
<th>York</th>
<th>today</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output (labels)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>B-dept</td>
<td>O</td>
<td>B-arr</td>
<td>I-arr</td>
<td>B-date</td>
</tr>
</tbody>
</table>

Source: http://deeplearning.net/tutorial/rnnslu.html
show flights from Boston to New York today

<table>
<thead>
<tr>
<th>Input (words)</th>
<th>show</th>
<th>flights</th>
<th>from</th>
<th>Boston</th>
<th>to</th>
<th>New</th>
<th>York</th>
<th>today</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output (labels)</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>B-dept</td>
<td>O</td>
<td>B-arr</td>
<td>I-arr</td>
<td>B-date</td>
</tr>
</tbody>
</table>

Source: http://deeplearning.net/tutorial/rnnslu.html
Neural Networks

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

Source: https://www.youtube.com/watch?v=bx8T-V8XRc&index=1&list=PLiaHhY2iBX9hdHaR96b7XevZtgZRa1PoU
<table>
<thead>
<tr>
<th>Hours Sleep</th>
<th>Hours Study</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>75</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>93</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>?</td>
</tr>
</tbody>
</table>

Source: https://www.youtube.com/watch?v=bxz2T-V8XRs&index=1&list=PLhaHhY2iBX9hdHaRt6b7XevZtgZRa1PoU
<table>
<thead>
<tr>
<th></th>
<th>Hours Sleep</th>
<th>Hours Study</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training</td>
<td>5</td>
<td>1</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2</td>
<td>93</td>
</tr>
<tr>
<td>Testing</td>
<td>8</td>
<td>3</td>
<td>?</td>
</tr>
<tr>
<td>X</td>
<td>3</td>
<td>5</td>
<td>75</td>
</tr>
</tbody>
</table>

Source: https://www.youtube.com/watch?v=bxe2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
\[ Y = WX + b \]
Y = WX + b

Output

input

Weights

Trained

bias

Source: https://www.youtube.com/watch?v=G8eNWzOggE
$W X + b = Y$

Scores $\rightarrow$ Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
W X + b = Y

\[ S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \]

Logits → Scores → Probabilities

Source: https://www.youtube.com/watch?v=G8eNWzxOgqE
\[
S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} = \frac{e^{2.0}}{e^{2.0}+e^{1.0}+e^{0.1}} = \frac{2.7182^{2.0}}{2.7182^{2.0}+2.7182^{1.0}+2.7182^{0.1}} = 0.7
\]

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

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

\[W \cdot X + b = Y\]

\[\begin{bmatrix}
2.0 \\
1.0 \\
0.1
\end{bmatrix} \rightarrow S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \rightarrow \begin{bmatrix}
0.7 \\
0.2 \\
0.1
\end{bmatrix}\]

Logits \rightarrow Scores \rightarrow Probabilities
Training a Network

= Minimize the Cost Function

Source: https://www.youtube.com/watch?v=bxetV8XR&s=index=1&list=PLiaHhY2lBX9hdHaRt6b7XevZtgZRa1PoU
Training a Network

= Minimize the **Cost** Function

Minimize the **Loss** Function
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bx2T-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRt6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bxn2T-V8XR&s=1&list=PLiaHhY2X9hdHaRt6b7XevZtgZRa1PoU
Error = Predict Y - Actual Y

Error : Cost : Loss

Source: https://www.youtube.com/watch?v=bx2T-V8XR&index=1&list=PLiaHhY2iBX9hdHaR67XevZtgZRa1PoU
Activation Functions
Activation Functions

**Sigmoid**

**TanH**

**ReLU**

(Rectified Linear Unit)

\[ f(x) = \max(0, x) \]

[0, 1] \quad [-1, 1]
Activation Functions

Sigmoid:
\[ f(x) = \frac{1}{1 + e^{-x}} \]

Tanh:
\[ \tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \]

ReLU:
\[ f(x) = \begin{cases} 
0 & \text{for } x < 0 \\
 x & \text{for } x \geq 0 
\end{cases} \]
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

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
What is Deep Learning?

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

\[ x_1 \rightarrow w_1 \rightarrow y \]
\[ x_2 \rightarrow w_2 \]
\[ \ldots \]
\[ x_n \rightarrow w_n \]

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
The Neuron

\[ y = F \left( \sum_{i} w_i x_i \right) \]

\[ F(x) = \max(0, x) \]
\[ y = \max \left( 0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3 \right) \]
IS THIS A CAT or DOG?

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
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”

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
\[ y = \max (0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3) \]
Next time:

\[
y = \max(0, -0.23 \times x_1 + 0.31 \times x_2 + 0.65 \times x_3)
\]

\[
y = \max(0, -0.21 \times x_1 + 0.3 \times x_2 + 0.7 \times x_3)
\]
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!)

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
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)

Source: http://deeplearning.net/software_links/
Deep Learning with Google TensorFlow
### Deep Learning Libraries: TensorFlow and Keras

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

<table>
<thead>
<tr>
<th>new contributors from 2017-02-11 to 2017-04-12</th>
<th>new forks from 2017-02-11 to 2017-04-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 131 tensorflow/tensorflow</td>
<td>#1 4192 tensorflow/tensorflow</td>
</tr>
<tr>
<td>#2 63 fchollet/keras</td>
<td>#2 991 fchollet/keras</td>
</tr>
<tr>
<td>#3 51 pytorch/pytorch</td>
<td>#3 810 BVLC/caffe</td>
</tr>
<tr>
<td>#4 49 dm1c/mxnet</td>
<td>#4 517 deeplearning4j/deeplearning4j</td>
</tr>
<tr>
<td>#5 18 Theano/Theano</td>
<td>#5 414 dm1c/mxnet</td>
</tr>
<tr>
<td>#6 11 BVLC/caffe</td>
<td>#6 307 pytorch/pytorch</td>
</tr>
<tr>
<td>#7 11 Microsoft/CNTK</td>
<td>#7 244 Microsoft/CNTK</td>
</tr>
<tr>
<td>#8 9 tflearn/tflearn</td>
<td>#8 211 tflearn/tflearn</td>
</tr>
<tr>
<td>#9 9 pfnet/chainer</td>
<td>#9 134 torch/torch7</td>
</tr>
<tr>
<td>#10 8 torch/torch7</td>
<td>#10 131 Theano/Theano</td>
</tr>
<tr>
<td>#11 5 deeplearning4j/deeplearning4j</td>
<td>#11 116 baidu/paddle</td>
</tr>
<tr>
<td>#12 4 NVIDIA/DIGITS</td>
<td>#12 88 NVIDIA/DIGITS</td>
</tr>
<tr>
<td>#13 3 baidu/paddle</td>
<td>#13 55 pfnet/chainer</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>new issues from 2017-02-11 to 2017-04-12</th>
<th>aggregate activity from 2017-02-11 to 2017-04-12</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 1175 tensorflow/tensorflow</td>
<td>#1 36.64 tensorflow/tensorflow</td>
</tr>
<tr>
<td>#2 568 fchollet/keras</td>
<td>#2 12.52 fchollet/keras</td>
</tr>
<tr>
<td>#3 499 dm1c/mxnet</td>
<td>#3 8.53 dm1c/mxnet</td>
</tr>
<tr>
<td>#4 286 pytorch/pytorch</td>
<td>#4 6.09 BVLC/caffe</td>
</tr>
<tr>
<td>#5 257 Microsoft/CNTK</td>
<td>#5 5.92 pytorch/pytorch</td>
</tr>
<tr>
<td>#6 239 deeplearning4j/deeplearning4j</td>
<td>#6 5.12 deeplearning4j/deeplearning4j</td>
</tr>
<tr>
<td>#7 219 baidu/paddle</td>
<td>#7 4.12 Microsoft/CNTK</td>
</tr>
<tr>
<td>#8 173 Theano/Theano</td>
<td>#8 2.93 Theano/Theano</td>
</tr>
<tr>
<td>#9 171 BVLC/caffe</td>
<td>#9 2.86 baidu/paddle</td>
</tr>
<tr>
<td>#10 112 NVIDIA/DIGITS</td>
<td>#10 2.17 tflearn/tflearn</td>
</tr>
<tr>
<td>#11 84 tflearn/tflearn</td>
<td>#11 1.68 NVIDIA/DIGITS</td>
</tr>
<tr>
<td>#12 57 pfnet/chainer</td>
<td>#12 1.38 torch/torch7</td>
</tr>
<tr>
<td>#13 47 torch/torch7</td>
<td>#13 1.12 pfnet/chainer</td>
</tr>
</tbody>
</table>

Source: https://twitter.com/fchollet/status/852194634470223873
Google TensorFlow

TensorFlow is an Open Source Software Library for Machine Intelligence

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

https://www.tensorflow.org/
numerical computation using data flow graphs

https://www.tensorflow.org/
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]

[https://www.tensorflow.org/](https://www.tensorflow.org/)
Nodes: mathematical operations

edges: multidimensional data arrays (tensors)

communicated between nodes
Computation is a Dataflow Graph

Graph of Nodes, also called Operations or ops.

- bias
- weights
- examples
- labels

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Computation is a Dataflow Graph

Edges are N-dimensional arrays: Tensors

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Logistic Regression as Dataflow Graph

Nodes
- $b$: bias
- $W$: weights
- $X$: inputs
- $Y$: targets

Operations
- Add
- Softmax
- Xent

Edges are N-dimensional arrays: Tensors

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Computation is a Dataflow Graph

with state

‘Biases’ is a variable

Some ops compute gradients

-= updates biases

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
Neural Networks

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

Source: https://www.youtube.com/watch?v=bxet-V8XRs&index=1&list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
Data Flow Graph

Source: https://www.tensorflow.org/
Data Flow Graph

Source: https://www.tensorflow.org/
Data Flow Graph
Python
Download Anaconda

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.

https://www.continuum.io/downloads
# Download Anaconda Python 3.6

## 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 2.7 version

## Download Options

<table>
<thead>
<tr>
<th>Python Version</th>
<th>Graphical Installer</th>
<th>Command-Line Installer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python 3.6</td>
<td>(424M)</td>
<td>(363M)</td>
</tr>
<tr>
<td>Python 2.7</td>
<td>(419M)</td>
<td>(358M)</td>
</tr>
</tbody>
</table>

### Download Link

[https://www.continuum.io/downloads](https://www.continuum.io/downloads)
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

https://www.continuum.io/downloads
OS X Anaconda 3 - 4.3.1
Python 3.6 Installation
Anaconda3-4.3.1-MacOSX-x86_64.pkg
Install Anaconda 3

You will be guided through the steps necessary to install this software.
Install Anaconda 3

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

Go Back  Continue
Install Anaconda 3

Software License Agreement

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.
Install Anaconda 3

Software License Agreement

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.

* 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...
Install Anaconda 3

Standard Install on "Macintosh HD"

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.
Install Anaconda 3

Select a Destination

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

Standard Install on “Macintosh HD”

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.
Install Anaconda 3

Installing Anaconda3

Registering updated applications...

Install time remaining: About a minute
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](https://docs.continuum.io/anaconda/pkg-docs)

178 python packages included.

Supported packages: 453

Source: https://docs.continuum.io/anaconda/pkg-docs
Install Anaconda 3

178 python packages included.
Anaconda-Navigator
Anaconda-Navigator
Jupyter Notebook

**Jupyter Notebook**

- **jupyter notebook**
  - 4.3.1
  - Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

- **qtconsole**
  - 4.2.1
  - PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

- **spyder**
  - 3.1.2
  - Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

- **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

- **glueviz**
  - 0.9.1
  - Multidimensional data visualization across files. Explore relationships within and among related datasets.

- **rstudio**
  - 1.0.136
  - A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Jupyter Notebook
New Python 3
print("hello, world")
from platform import python_version
print("Python Version: ", python_version())

In [1]: print("hello, world")

hello, world

In [2]: from platform import python_version
   print("Python Version: ", python_version())

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

https://conda.io/docs/py2or3.html
Anaconda Create New Python 3.5 Environment (py35)

Source: http://conda.pydata.org/docs/py2or3.html
Anaconda Create New Python 2.7 Environment (py27)

Source: http://conda.pydata.org/docs/py2or3.html
Verify that conda is installed, check current conda version

- `conda --version`

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

http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version
Check current conda version
Check current python version
Check conda environments

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

http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version
```
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
bittarray  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
```

http://conda.pydata.org/docs/using/using.html#verify-that-conda-is-installed-check-current-conda-version
python --version

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

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

source activate py35

source deactivate py35

source deactivate py35

source deactivate py35

source activate py35

source deactivate py35

source deactivate py35

source activate py35
conda create -n py352 python=3.5.2 anaconda

Create a Python 3.5.2 environment

Source: http://conda.pydata.org/docs/py2or3.html
conda create -n py352 python=3.5.2 anaconda

```
# To activate this environment, use:
# > source activate py352
#
# To deactivate this environment, use:
# > source deactivate py352
```

Source: http://conda.pydata.org/docs/py2or3.html
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$ ]
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]
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.

http://playground.tensorflow.org/
TensorBoard

Main Graph

Auxiliary nodes

https://www.tensorflow.org/tensorboard/index.html#graphs
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

```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
>>> 
```

https://github.com/tensorflow/tensorflow
Architecture of TensorFlow

[Diagram]

Core TensorFlow Execution System

- C++ front end
- Python front end

- CPU
- GPU
- Android
- iOS

Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016
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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0
TensorFlow MNIST Tutorial

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

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>mlengine</td>
<td>added example using the Tensorflow high level layers API</td>
<td>26 days ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>small bug fix in batch norm</td>
<td>6 months ago</td>
</tr>
<tr>
<td>CONTRIBUTING.md</td>
<td>initial commit 2</td>
<td>4 months ago</td>
</tr>
<tr>
<td>INSTALL.txt</td>
<td>Update INSTALL.txt</td>
<td>25 days ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Initial commit</td>
<td>a year ago</td>
</tr>
<tr>
<td>README.md</td>
<td>better image URL</td>
<td>3 months ago</td>
</tr>
<tr>
<td>mnist_1.0_softmax.py</td>
<td>global_variables_initializer used everywhere instead of initialize_al...</td>
<td>2 months ago</td>
</tr>
<tr>
<td>mnist_2.0_five_layers_sigmoid.py</td>
<td>Fix spacing in the network structure comment</td>
<td>a month ago</td>
</tr>
<tr>
<td>mnist_2.1_five_layers_relu_lrdecay...</td>
<td>Fix spacing in the network structure comment</td>
<td>a month ago</td>
</tr>
</tbody>
</table>

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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0
git clone https://github.com/martin-gorner/tensorflow-mnist-tutorial.git

cd tensorflow-mnist-tutorial

python3 mnist_1.0_softmax.py

Source: https://github.com/martin-gorner/tensorflow-mnist-tutorial/blob/master/INSTALL.txt
MNIST dataset: 60,000 labeled digits

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#0
cd tensorflow-mnist-tutorial
python3 mnist_1.0_softmax.py

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#1
Train a Neural Network

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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Training digits

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Training digits

accuracy

cross entropy loss

weights

biases

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Test digits

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Cross entropy loss

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

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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#2
Weights and Biases

Accuracy

Cross entropy loss

Weights

Biases

Training digits

Test digits

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

Softmax

Cross-entropy

Mini-batch
Very Simple Model: Softmax Classification

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Very Simple Model: Softmax Classification

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Very Simple Model: Softmax Classification

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

weighted sum of all pixels + bias

neuron outputs

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
In Matrix notation, 100 images at a time

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

$W$: 10 columns

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
X: 100 images, one per line, flattened

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
X: 100 images, one per line, flattened

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
**X**: 100 images, one per line, flattened

<table>
<thead>
<tr>
<th>W₀,₀</th>
<th>W₀,₁</th>
<th>W₀,₂</th>
<th>W₀,₃</th>
<th>...</th>
<th>W₀,₉</th>
</tr>
</thead>
<tbody>
<tr>
<td>W₁,₀</td>
<td>W₁,₁</td>
<td>W₁,₂</td>
<td>W₁,₃</td>
<td>...</td>
<td>W₁,₉</td>
</tr>
<tr>
<td>W₂,₀</td>
<td>W₂,₁</td>
<td>W₂,₂</td>
<td>W₂,₃</td>
<td>...</td>
<td>W₂,₉</td>
</tr>
<tr>
<td>W₃,₀</td>
<td>W₃,₁</td>
<td>W₃,₂</td>
<td>W₃,₃</td>
<td>...</td>
<td>W₃,₉</td>
</tr>
<tr>
<td>W₄,₀</td>
<td>W₄,₁</td>
<td>W₄,₂</td>
<td>W₄,₃</td>
<td>...</td>
<td>W₄,₉</td>
</tr>
<tr>
<td>W₅,₀</td>
<td>W₅,₁</td>
<td>W₅,₂</td>
<td>W₅,₃</td>
<td>...</td>
<td>W₅,₉</td>
</tr>
<tr>
<td>W₆,₀</td>
<td>W₆,₁</td>
<td>W₆,₂</td>
<td>W₆,₃</td>
<td>...</td>
<td>W₆,₉</td>
</tr>
<tr>
<td>W₇,₀</td>
<td>W₇,₁</td>
<td>W₇,₂</td>
<td>W₇,₃</td>
<td>...</td>
<td>W₇,₉</td>
</tr>
<tr>
<td>W₈,₀</td>
<td>W₈,₁</td>
<td>W₈,₂</td>
<td>W₈,₃</td>
<td>...</td>
<td>W₈,₉</td>
</tr>
</tbody>
</table>

... 

| L₀,₀ | L₀,₁ |

**Sources:**
- https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
X: 100 images, one per line, flattened

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
What are "weights" and "biases"?

How is the "cross-entropy" computed?

How exactly does the training algorithm work?
\[ Y = f(X) \]

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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Forecast: $Y[100, 10]$

TensorFlow (Python) Softmax

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

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

matrix multiply

broadcast on all lines

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Cross Entropy

\[\text{Cross entropy: } - \sum Y'_i \cdot \log(Y_i)\]

actual probabilities, "one-hot" encoded

computed probabilities

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Minimizing Cross Entropy (Minimizing Loss)

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
"mini-batches":
100 images and labels
import tensorflow as tf
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()
# 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))
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)
# 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

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#5
Deep Learning

Go deep!
5 fully-connected layers

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Sigmoid

\[
\frac{1}{1 + e^{-x}}
\]

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
5 fully-connected layers

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial
ReLU
ReLU
TensorFlow MNIST Tutorial

This is a graph showing the accuracy over time for training and testing. The accuracy starts high and decreases sharply after a certain point. The graph indicates a significant drop in test accuracy after approximately 1500 iterations.

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Learning Rate

Slow down…

Learning rate decay

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
LR = 0.003

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Accuracy

Cross entropy loss

Weights

Biases

Training digits

Test digits

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Dropout
Overfitting

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Dropout

Training
pkeep = 0.75

Test
pkeep = 1.0

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Accuracy

Cross entropy loss

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

Accuracy

Cross entropy loss

5 layers sigmoid

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Overfitting

Overfitting ?!

Too many neurons

BAD Network

Not enough DATA
Convolutional Layer

$W[4, 4, 3]$
Convolutional Layer

\[ W[4, 4, 3] \]

\[ W_2[4, 4, 3] \]

\[ W[4, 4, 3, 2] \]

- **Filter size**
- **Input channels**
- **Output channels**
Convolutional Max-Pool

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
All Convolutional

Hacker’s tip

ALL Convolutional
Bigger Convolutional Neural Network

- Convolutional layer, 4 channels
  \( W_1[5, 5, 1, 4] \) stride 1

- Convolutional layer, 8 channels
  \( W_2[4, 4, 4, 8] \) stride 2

- Convolutional layer, 12 channels
  \( W_3[4, 4, 8, 12] \) stride 2

- Fully connected layer
  \( W_4[7x7x12, 200] \)

- Softmax readout layer
  \( W_5[200, 10] \)

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Bigger Convolutional Neural Network

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
Bigger Convolutional Neural Network + Dropout

28x28x1

28x28x6

14x14x12

7x7x24

+ Dropout $p=0.75$

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
$W4[7x7x24, 200]$ softmax readout layer
$W5[200, 10]$

+ biases on all layers

Source: https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/
TensorFlow MNIST Tutorial

![Graphs showing accuracy and cross entropy loss for different networks.](https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/)

_larger convolutional network_
• 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.

Source: https://keras.io/
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

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
```

[https://keras.io/#installation](https://keras.io/#installation)
conda info --envs
conda --version
python --version
conda list
conda create -n tensorflow python=3.5
source activate tensorflow
activate tensorflow

Source: http://deeplearning.net/software/theano/install.html#anaconda
pip install Theano
conda install pydot
sudo pip install keras
pip install keras
pip install tensorflow
pip install ipython[all]

Source: http://deeplearning.net/software/theano/install.html#anaconda
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_x86_64.whl (5.6MB)
    100% (5.6/5.6MB) 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_x86_64.whl (16.2MB)
    100% (16.2/16.2MB) 16.2MB 43KB/s
Requirement already up-to-date: numpy>=1.3 (from gensim)
  Downloading numpy-1.12.1-cp36-cp36m-macosx_10_6_intel.macosx_10_9_intel.macosx_10_9_x86_64.whl (4.4MB)
    100% (4.4/4.4MB) 4.4MB 148KB/s
Collecting requests (from smart-open>=1.2.1->gensim)
  Downloading requests-2.13.0-py2.py3-none-any.whl (584KB)
    100% (584/584KB) 593kB 632KB/s
Building wheels for collected packages: smart-open
  Running setup.py bdist_wheel for smart-open ... done
  Stored in directory: /Users/lym/Library/Caches/pip/wheels/02/4a/43/68e963ce2b45baef89f6ec5458f2d0d88eafdfc45ca
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
      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
Keras
sudo pip install keras

Password:
The directory '/Users/myday/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/myday/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% 196kB 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
TensorFlow
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$
http://keras.io/

https://github.com/fchollet/keras
# Keras Examples

[GitHub Repository](https://github.com/fchollet/keras/tree/master/examples)

Mohanson committed with fchollet: Spelling errors (#6232)

Forked from master branch

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>README.md</td>
<td>Adding mnist_ae.gan.py example link in README (#4876)</td>
<td>4 months ago</td>
</tr>
<tr>
<td>addition_rnn.py</td>
<td>Spelling errors (#6232)</td>
<td>11 days ago</td>
</tr>
<tr>
<td>antirectifier.py</td>
<td>Style fix for examples. (#5980)</td>
<td>28 days ago</td>
</tr>
<tr>
<td>babi_memnn.py</td>
<td>Style fixes in example scripts</td>
<td>a month ago</td>
</tr>
<tr>
<td>babi_rnn.py</td>
<td>Style fixes in example scripts</td>
<td>a month ago</td>
</tr>
<tr>
<td>cifar10_cnn.py</td>
<td>Fix rmsprop learning rate for convergence (#6182)</td>
<td>17 days ago</td>
</tr>
<tr>
<td>conv_filter_visualization.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>conv_lstm.py</td>
<td>Update a number of example scripts.</td>
<td>2 months ago</td>
</tr>
<tr>
<td>deep_dream.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>image_ocr.py</td>
<td>Fixed URL for wordlist.tgz in image_ocr.py (#6136)</td>
<td>20 days ago</td>
</tr>
<tr>
<td>imdb_bidirectional_lstm.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>imdb_cnn.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>imdb_cnn_lstm.py</td>
<td>Style fix for examples. (#5980)</td>
<td>28 days ago</td>
</tr>
</tbody>
</table>
```python
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, 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')
```

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py
# 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.Adamax(),
              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
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

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py
from __future__ import print_function
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, 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])

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py
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])

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_cnn.py
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

Source: http://ai.stanford.edu/~amaas/data/sentiment/
### IMDB Dataset (Mass et al., 2011)

<table>
<thead>
<tr>
<th>Features</th>
<th>PL04</th>
<th>Our Dataset</th>
<th>Subjectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bag of Words (bnc)</td>
<td>85.45</td>
<td>87.80</td>
<td>87.77</td>
</tr>
<tr>
<td>Bag of Words (bΔt’c)</td>
<td>85.80</td>
<td>88.23</td>
<td>85.65</td>
</tr>
<tr>
<td>LDA</td>
<td>66.70</td>
<td>67.42</td>
<td>66.65</td>
</tr>
<tr>
<td>LSA</td>
<td>84.55</td>
<td>83.96</td>
<td>82.82</td>
</tr>
<tr>
<td>Our Semantic Only</td>
<td>87.10</td>
<td>87.30</td>
<td>86.65</td>
</tr>
<tr>
<td>Our Full</td>
<td>84.65</td>
<td>87.44</td>
<td>86.19</td>
</tr>
<tr>
<td>Our Full, Additional Unlabeled</td>
<td>87.05</td>
<td>87.99</td>
<td>87.22</td>
</tr>
<tr>
<td>Our Semantic + Bag of Words (bnc)</td>
<td>88.30</td>
<td>88.28</td>
<td>88.58</td>
</tr>
<tr>
<td>Our Full + Bag of Words (bnc)</td>
<td>87.85</td>
<td>88.33</td>
<td>88.45</td>
</tr>
<tr>
<td>Our Full, Add’l Unlabeled + Bag of Words (bnc)</td>
<td>88.90</td>
<td>88.89</td>
<td>88.13</td>
</tr>
<tr>
<td>Bag of Words SVM (Pang and Lee, 2004)</td>
<td>87.15</td>
<td>N/A</td>
<td>90.00</td>
</tr>
<tr>
<td>Contextual Valence Shifters (Kennedy and Inkpen, 2006)</td>
<td>86.20</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>tf.idf Weighting (Martineau and Finin, 2009)</td>
<td>88.10</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Appraisal Taxonomy (Whitelaw et al., 2005)</td>
<td>90.20</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

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


[http://keras.io/](http://keras.io/)

<table>
<thead>
<tr>
<th>tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>deep-learning</td>
</tr>
<tr>
<td>tensorflow</td>
</tr>
<tr>
<td>theano</td>
</tr>
<tr>
<td>neural-networks</td>
</tr>
<tr>
<td>machine-learning</td>
</tr>
<tr>
<td>data-science</td>
</tr>
</tbody>
</table>

- **3,503 commits**
- **4 branches**
- **28 releases**
- **424 contributors**

**Recent commits**

<table>
<thead>
<tr>
<th>branch</th>
<th>message</th>
<th>author</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>phipleg committed with fchollet</strong> Added logsumexp to backend. (#6346)</td>
<td>phipleg</td>
<td>7d52af6 a day ago</td>
</tr>
<tr>
<td>docker</td>
<td>Update docker files to TensorFlow 1, Theano 0.9 (#6116)</td>
<td></td>
<td>20 days ago</td>
</tr>
<tr>
<td>docs</td>
<td>fix stateful RNNs FAQ link (#6336)</td>
<td></td>
<td>3 days ago</td>
</tr>
<tr>
<td>examples</td>
<td>Spelling errors (#6232)</td>
<td></td>
<td>11 days ago</td>
</tr>
<tr>
<td>keras</td>
<td>Added logsumexp to backend. (#6346)</td>
<td></td>
<td>a day ago</td>
</tr>
<tr>
<td>tests</td>
<td>Added logsumexp to backend. (#6346)</td>
<td></td>
<td>a day ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Fix FAQ question</td>
<td></td>
<td>a month ago</td>
</tr>
<tr>
<td>.travis.yml</td>
<td>Update Travis config</td>
<td></td>
<td>9 days ago</td>
</tr>
<tr>
<td>CONTRIBUTING.md</td>
<td>Mention requests for contribution in CONTRIBUTING.md</td>
<td></td>
<td>a month ago</td>
</tr>
</tbody>
</table>

[https://github.com/fchollet/keras](https://github.com/fchollet/keras)
# Keras Examples

[keras](https://github.com/fchollet/keras/tree/master/examples)

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>README.md</td>
<td>Adding mnist_ae.py example link in README (#4876)</td>
<td>4 months ago</td>
</tr>
<tr>
<td>addition_rnn.py</td>
<td>Spelling errors (#6232)</td>
<td>11 days ago</td>
</tr>
<tr>
<td>antirectifier.py</td>
<td>Style fix for examples. (#5980)</td>
<td>28 days ago</td>
</tr>
<tr>
<td>babi_memn.py</td>
<td>Style fixes in example scripts</td>
<td>a month ago</td>
</tr>
<tr>
<td>babi_rnn.py</td>
<td>Style fixes in example scripts</td>
<td>a month ago</td>
</tr>
<tr>
<td>cifar10_cnn.py</td>
<td>fix rmsprop learning rate for convergence (#6182)</td>
<td>17 days ago</td>
</tr>
<tr>
<td>conv_filter_visualization.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>conv_lstm.py</td>
<td>Update a number of example scripts.</td>
<td>2 months ago</td>
</tr>
<tr>
<td>deep_dream.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>image_ocr.py</td>
<td>Fixed URL for wordlist.tgz in image_ocr.py (#6136)</td>
<td>20 days ago</td>
</tr>
<tr>
<td>imdb_bidirectional_lstm.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>imdb_cnn.py</td>
<td>Finish updating examples.</td>
<td>a month ago</td>
</tr>
<tr>
<td>imdb_cnn_lstm.py</td>
<td>Style fix for examples. (#5980)</td>
<td>28 days ago</td>
</tr>
</tbody>
</table>
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.

Source: [https://github.com/fchollet/keras/tree/master/examples](https://github.com/fchollet/keras/tree/master/examples)
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.

Source: https://keras.io/datasets/
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()
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

Source: keras/datasets/imdb.py
```python
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,
...}
```
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
# 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>
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'

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_cnn.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

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)
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

Source: https://github.com/fchollet/keras/blob/master/examples/imdb_lstm.py
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)

Source: https://github.com/fchollet/keras/blob/master/examples/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: 0.8284
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
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'
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

Source: https://github.com/fchollet/keras/blob/master/examples/lstm_benchmark.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)
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"))
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...')
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)
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))
# Deep Learning Training Visualization

```python
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_accuacy_loss_" + py_filename + "_" + str(epochs) + ".png", dpi= 300)
```
# Log File for Deep Learning Summary Analysis

```python
def log_file_utf8(logfile, py_filename, epochs, score, acc, timer_end, timer_start, history):
    log_file_utf8("logfile.txt", 'DL_Summary:\tpy_filename\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 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
#Log File for Deep Learning Summary Analysis

```
log_file_utf8("logfile.txt", 'DL_Summary:\tpy_filename\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))
```

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

*Note: The table above represents the summary of various models' performance measured in terms of epochs, score, accuracy, and timer (in seconds). The best accuracy values are highlighted in bold.*
imdb_cnn_lstm_2.py

Model accuracy

- Train accuracy
- Test/val accuracy

Model loss

- Train loss
- Test/val loss
model accuracy

accuracy

model loss

loss

epoch
The diagram shows the model accuracy and loss over epochs for a deep learning model applied to the IMDB fasttext dataset. Two plots are presented:

1. **Model Accuracy**:
   - The plot tracks the training accuracy (blue line) and test/val accuracy (orange line) over epochs.
   - As the number of epochs increases, the training accuracy approaches 1.0, indicating high performance.
   - The test/val accuracy also increases initially but stabilizes at around 0.85, showing good generalization.

2. **Model Loss**:
   - The plot displays the training loss (blue line) and test/val loss (orange line) over epochs.
   - The training loss decreases significantly during the early epochs, indicating effective learning.
   - The test/val loss increases gradually, suggesting some overfitting as the model learns the training data too well.

These plots are typically used to evaluate the performance and generalization capabilities of a machine learning model, allowing for informed decisions on model tuning and validation.
Deep Learning with CPU vs. GPU

Timings:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Backend</th>
<th>Time / Epoch</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>TF</td>
<td>3 hrs</td>
</tr>
<tr>
<td>Titan X (maxwell)</td>
<td>TF</td>
<td>4 min</td>
</tr>
<tr>
<td>Titan X (maxwell)</td>
<td>TH</td>
<td>7 min</td>
</tr>
</tbody>
</table>

Source: https://github.com/fchollet/keras/blob/master/examples/mnist_ae.py
Deep Learning Studio
Cloud platform for designing Deep Learning AI without programming

http://deepcognition.ai/
Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming

CIFAR-10 - Object Recognition in Images

http://deepcognition.ai/
## Deep Learning Studio

Cloud platform for designing Deep Learning AI without programming

---

### MNIST Handwritten Digits Classifier

<table>
<thead>
<tr>
<th>Model</th>
<th>HyperParameters</th>
<th>Training</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Dataset Source: Testing

#### Training Run: Run0

- **Dataset Source:** Testing
- **Training Run:** Run0
- **Download Trained Model**

<table>
<thead>
<tr>
<th>Digit Label</th>
<th>Image</th>
<th>predictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
References

- Martin Gorner (2017), TensorFlow and Deep Learning without a PhD, Part 2 (Google Cloud Next '17), https://www.youtube.com/watch?v=fTUwdUXFFfl8
- Deep Learning Basics: Neural Networks Demystified, https://www.youtube.com/playlist?list=PLiaHhY2iBX9hdHaRr6b7XevZtgZRa1PoU
- Deep Learning SIMPLIFIED, https://www.youtube.com/playlist?list=PLjJh1vlSEYgvGod9wWiydumYI8hOXixNu
- TensorFlow: https://www.tensorflow.org/
- Theano: http://deeplearning.net/software/theano/
- Keras: http://keras.io/
- 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
- https://github.com/leriomaggio/deep-learning-keras-tensorflow