Deep Learning
Sigurd Spieckermann
Siemens Corporate Technology
Google rilascia gratis lezioni di deep learning.
Will robots take over your job?
OK, LET’S GET SERIOUS NOW …
Deep Learning vs. Classic Data Modeling

LEARNED FROM DATA

- RULE-BASED SYSTEMS
  - INPUT
  - OUTPUT
- HAND-DESIGNED PROGRAM
  - INPUT
  - HAND-DESIGNED FEATURES
  - MAPPING FROM FEATURES
  - OUTPUT
- CLASSIC MACHINE LEARNING
  - INPUT
  - FEATURES
  - MAPPING FROM FEATURES
  - OUTPUT
- DEEP LEARNING
  - INPUT
  - FEATURES
  - ADDITIONAL LAYERS OF MORE ABSTRACT FEATURES
  - MAPPING FROM FEATURES
  - OUTPUT

SOURCE: http://www.deeplearningbook.org/contents/intro.html
Deep Learning
Hierarchical Feature Extraction

SOURCE: http://www.eidolonspeak.com/Artificial_Intelligence/SOA_P3_Fig4.png
Deep Learning
Hierarchical Feature Extraction

SOURCE:
NEURAL NETWORKS
Neural Networks
Linear Regression

\[ \hat{y} = w^T x + b \]
Neural Networks
Logistic Regression

\[ \hat{y} = \text{logistic}(w^T x + b) \]

\[ \text{logistic}(z) = \frac{1}{1 + e^{-z}} \]
Neural Networks
Fully Connected Feedforward Neural Network

\[ h = \phi(W^{(1)}x + b^{(1)}) \]
\[ \hat{y} = \text{logistic}(W^{(2)}h + b^{(2)}) \]
\[ \phi(z) = \begin{cases} 
\tanh(z) \\
\text{relu}(z) \\
... 
\end{cases} \]
\[ \text{relu}(z) := \max(0, z) = \begin{cases} 
z & z > 0 \\
0 & z \leq 0 
\end{cases} \]
Neural Networks
1D Convolutional Feedforward Neural Network
Neural Networks
1D Convolution

FILTERS
# FEATURES
WINDOW SIZE

TIME

INPUT

OUTPUT

CONV

# FEATURES
WINDOW SIZE
TIME

CONV
Neural Networks
1D Convolution

FILTERS
# FEATURES

WINDOW SIZE

TIME

INPUT

OUTPUT

CONV
Neural Networks
1D Convolution

FILTERS
# FEATURES

INPUT
WINDOW SIZE

TIME

CONV

OUTPUT

# FEATURES
FILTERS
WINDOW SIZE
TIME
INPUT
CONV
OUTPUT
Neural Networks
1D Convolution

FILTERS
# FEATURES
WINDOW SIZE

TIME

INPUT

OUTPUT

CONV
Neural Networks
1D Convolution

FILTERS
# FEATURES

WINDOW SIZE

TIME

INPUT

OUTPUT

CONV
Neural Networks
1D Convolution

FILTERS
# FEATURES

WINDOW SIZE

TIME

INPUT

OUTPUT

CONV
Neural Networks
1D Convolution

FILTERS
# FEATURES

WINDOW SIZE

TIME

INPUT

OUTPUT

CONV
Neural Networks
1D Convolution

<table>
<thead>
<tr>
<th>FILTERS</th>
<th># FEATURES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WINDOW SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONV</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FILTERS</th>
<th>INPUT</th>
<th>OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Neural Networks
1D Convolution

FILTERS
# FEATURES

WINDOW SIZE

TIME

INPUT

CONV

OUTPUT
Neural Networks
1D Convolution

AND SO ON ...
Neural Networks
1D Convolution (mode=HALF)

NOTE THAT THE INPUT AND OUTPUT SEQUENCES HAVE THE SAME LENGTH
Neural Networks
1D Convolution (mode=HALF)

FILTERS
# FEATURES

WINDOW SIZE

ZERO PADDING

TIME

NOTE THAT THE INPUT AND OUTPUT SEQUENCES HAVE THE SAME LENGTH
Neural Networks
1D Convolution (mode=HALF)

NOTE THAT THE INPUT AND OUTPUT SEQUENCES HAVE THE SAME LENGTH
Neural Networks
1D Convolution (mode=HALF)

The diagram illustrates the process of 1D convolution in the HALF mode. The input sequence is padded with zeros on both sides. The convolution operation then applies filters across the sequence, resulting in an output sequence of the same length as the input sequence, without any overlap.

NOTE THAT THE INPUT AND OUTPUT SEQUENCES HAVE THE SAME LENGTH.
Neural Networks

1D Convolution (mode=HALF)

# FEATURES
WINDOW SIZE
FILTERS

ZERO PADDING
# FEATURES

TIME
CONV

AND SO ON ...

NOTE THAT THE INPUT AND OUTPUT SEQUENCES HAVE THE SAME LENGTH
Neural Networks
1D Pooling (max, sum, average, …)
Neural Networks

1D Pooling (max, sum, average, ...)
Neural Networks
1D Pooling (max, sum, average, …)
Neural Networks

2D Convolutional Feedforward Neural Network

SOURCE: http://deeplearning.net/tutorial/_images/mylenet.png
Neural Networks
Recurrent Neural Network (Elman architecture)

\[ h_t = \tanh(Vx_t + Uh_{t-1} + b) \]
\[ \hat{y}_t = \phi(Wh_t + d) \]
Neural Networks
Recurrent Neural Network (Elman architecture, unfolded)
Neural Networks

Bi-Directional Recurrent Neural Network (Elman architecture, unfolded)
NEURAL NETWORKS HAVE BEEN AROUND FOR DECADES!

SO WHAT’S NEW?
What’s new?

OPTIMIZATION & LEARNING

OPTIMIZATION ALGORITHMS
• AdaGrad
• AdaDelta
• Adam
• RMSProp
• Hessian-Free Optimization
• …

REPARAMETERIZATION
• Batch Normalization
• Weight Normalization
• …

REGULARIZATION
• Dropout
• DropConnect
• …

MODEL ARCHITECTURES

BUILDING BLOCKS
• Spatial/Temporal Pooling
• Attention Mechanism
• Gated Recurrent Units
• Beam-search for sequence generation
• Variable-length sequence modeling
• …

ARCHITECTURES
• Inception (Google)
• VGG (Oxford University)
• Encoder-Decoder Framework
• End-to-end Models
• …

SOFTWARE

• Theano
• Blocks + Fuel
• Keras
• Lasagne
• PyLearn2*
• TensorFlow
• Torch7
• Caffe…

GENERAL

• GPUs
• Data
<table>
<thead>
<tr>
<th>What's new?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OPTIMIZATION &amp; LEARNING</strong></td>
</tr>
<tr>
<td><strong>OPTIMIZATION ALGORITHMS</strong></td>
</tr>
<tr>
<td>• AdaGrad</td>
</tr>
<tr>
<td>• AdaDelta</td>
</tr>
<tr>
<td>• Adam</td>
</tr>
<tr>
<td>• RMSProp</td>
</tr>
<tr>
<td>• Hessian-Free Optimization</td>
</tr>
<tr>
<td>• …</td>
</tr>
<tr>
<td><strong>REPARAMETERIZATION</strong></td>
</tr>
<tr>
<td>• Batch Normalization</td>
</tr>
<tr>
<td>• Weight Normalization</td>
</tr>
<tr>
<td>• …</td>
</tr>
<tr>
<td><strong>REGULARIZATION</strong></td>
</tr>
<tr>
<td>• Dropout</td>
</tr>
<tr>
<td>• DropConnect</td>
</tr>
<tr>
<td>• …</td>
</tr>
<tr>
<td><strong>MODEL ARCHITECTURES</strong></td>
</tr>
<tr>
<td><strong>BUILDING BLOCKS</strong></td>
</tr>
<tr>
<td>• Spatial/Temporal Pooling</td>
</tr>
<tr>
<td>• Attention Mechanism</td>
</tr>
<tr>
<td>• Gated Recurrent Units</td>
</tr>
<tr>
<td>• Beam-search for sequence generation</td>
</tr>
<tr>
<td>• Variable-length sequence modeling</td>
</tr>
<tr>
<td>• …</td>
</tr>
<tr>
<td><strong>ARCHITECTURES</strong></td>
</tr>
<tr>
<td>• Inception (Google)</td>
</tr>
<tr>
<td>• VGG (Oxford University)</td>
</tr>
<tr>
<td>• Encoder-Decoder Framework</td>
</tr>
<tr>
<td>• End-to-end Models</td>
</tr>
<tr>
<td>• …</td>
</tr>
<tr>
<td><strong>SOFTWARE</strong></td>
</tr>
<tr>
<td>• Theano</td>
</tr>
<tr>
<td>• Blocks + Fuel</td>
</tr>
<tr>
<td>• Keras</td>
</tr>
<tr>
<td>• Lasagne</td>
</tr>
<tr>
<td>• PyLearn2*</td>
</tr>
<tr>
<td>• TensorFlow</td>
</tr>
<tr>
<td>• Torch7</td>
</tr>
<tr>
<td>• Caffe…</td>
</tr>
<tr>
<td><strong>GENERAL</strong></td>
</tr>
<tr>
<td>• GPUs</td>
</tr>
<tr>
<td>• Data</td>
</tr>
</tbody>
</table>
What’s new?
Optimization Algorithms

- Neural networks are composed of differentiable building blocks
- Training a neural network means minimization of some non-convex differentiable cost function using iterative gradient-based optimization methods
• Neural networks are composed of differentiable building blocks
• Training a neural network means minimization of some non-convex differentiable cost function using iterative gradient-based optimization methods
What’s new?
Optimization Algorithms

• Neural networks are composed of differentiable building blocks
• Training a neural network means minimization of some non-convex differentiable cost function using iterative gradient-based optimization methods
• Gradients are computed using backpropagation
• The simplest optimization algorithm is “gradient descent”

\[
\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \nabla_{\theta^{(i)}} f(\theta^{(i)})
\]
What’s new?
Optimization Algorithms

• Neural networks are composed of differentiable building blocks
• Training a neural network means minimization of some non-convex differentiable cost function using iterative gradient-based optimization methods
• Gradients are computed using backpropagation
• The simplest optimization algorithm is “gradient descent”
• … but it has limitations

IMAGE SOURCE:
What’s new?
Optimization Algorithms

- Neural networks are composed of differentiable building blocks
- Training a neural network means minimization of some non-convex differentiable cost function using iterative gradient-based optimization methods
- Gradients are computed using backpropagation
- The simplest optimization algorithm is “gradient descent”
- … but it has limitations
- Information about the local curvature of the cost function helps to adjust the direction and magnitude of the gradient for better progress (along the lines of Newton’s method)
- Exact local curvature is infeasible to compute
- Recent optimization algorithms like AdaGrad, RMSProp, AdaDelta etc. try to approximate local curvature information efficiently
What’s new?
Reparameterization

- First-order gradient-based optimization methods are not invariant to reparameterization of the optimization objective.
What’s new?
Reparameterization

• First-order gradient-based optimization methods are not invariant to reparameterization of the optimization objective
• Instead of using more sophisticated optimization algorithms that are better at dealing with ill-conditioned optimization problems, reparameterize the objective function so that simpler optimization algorithms work better
What’s new?
Reparameterization

• First-order gradient-based optimization methods are not invariant to reparameterization of the optimization objective
• Instead of using more sophisticated optimization algorithms that are better at dealing with ill-conditioned optimization problems, reparameterize the objective function so that simpler optimization algorithms work better
• We typically standardize (approximately decorrelate) real-valued (Gaussian-like) inputs which makes the optimization problem easier
What’s new?
Reparameterization

- First-order gradient-based optimization methods are not invariant to reparameterization of the optimization objective
- Instead of using more sophisticated optimization algorithms that are better at dealing with ill-conditioned optimization problems, reparameterize the objective function so that simpler optimization algorithms work better
- We typically standardize (approximately decorrelate) real-valued (Gaussian-like) inputs which makes the optimization problem easier
- Why not do this in each (hidden) layer as well?
  ⇒ Batch Normalization
What’s new?
Regularization

- Randomly set neurons to zero
- Results in an ensemble with an exponential number of members whose parameters are shared
- Primarily used in fully connected layers because of the large number of parameters
- Rarely used in convolutional layers
- Rarely used in recurrent neural networks (if at all between the hidden state and output)
What’s new?
Regularization

- Randomly set neurons to zero
- Results in an ensemble with an exponential number of members whose parameters are shared
- Primarily used in fully connected layers because of the large number of parameters
- Rarely used in convolutional layers
- Rarely used in recurrent neural networks (if at all between the hidden state and output)
What’s new?
Regularization

- Randomly set neurons to zero
- Results in an ensemble with an exponential number of members whose parameters are shared
- Primarily used in fully connected layers because of the large number of parameters
- Rarely used in convolutional layers
- Rarely used in recurrent neural networks (if at all between the hidden state and output)
What’s new?
Regularization

- Randomly set neurons to zero
- Results in an ensemble with an exponential number of members whose parameters are shared
- Primarily used in fully connected layers because of the large number of parameters
- Rarely used in convolutional layers
- Rarely used in recurrent neural networks (if at all between the hidden state and output)
What’s new?

OPTIMIZATION & LEARNING

OPTIMIZATION ALGORITHMS
- AdaGrad
- AdaDelta
- Adam
- RMSProp
- Hessian-Free Optimization
- …

REPARAMETERIZATION
- Batch Normalization
- Weight Normalization
- …

REGULARIZATION
- Dropout
- DropConnect
- …

MODEL ARCHITECTURES

BUILDING BLOCKS
- Spatial/Temporal Pooling
- Attention Mechanism
- Gated Recurrent Units
- Beam-search for sequence generation
- Variable-length sequence modeling
- …

ARCHITECTURES
- Inception
- VGG
- Encoder-Decoder Framework
- End-to-end Models
- …

SOFTWARE
- Theano
- Blocks + Fuel
- Keras
- Lasagne
- PyLearn2*
- TensorFlow
- Torch7
- Caffe…

GENERAL
- GPUs
- Data

* deprecated
What’s new?
Attention Mechanism in Image Caption Generation

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.

SOURCE:
Economic growth has slowed down in recent years.

Das Wirtschaftswachstum hat sich in den letzten Jahren verlangsamt.

Economic growth has slowed down in recent years.

La croissance économique s'est ralentie ces dernières années.
What’s new?
Inception Architecture

Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax
What’s new?
VGG-16 Architecture

- Filter size 3×3
- 2+ successive convolutions with before pooling instead of the common CONV → POOL chain
- Convolution mode “half”
- More layers ⇒ larger capacity
- Parameter-efficient due to small filters
What’s new?
Encoder-Decoder Framework

SOURCE: https://www.tensorflow.org/versions/r0.8/tutorials/seq2seq/index.html#sequence-to-sequence-models
What’s new?
End-to-end model (object recognition)

SOURCE:
What's new?
End-to-end model (object recognition)

SOURCE:
What's new?
End-to-end model (question answering)
Data Flow Graphs / Computation Graphs
What’s new?

OPTIMIZATION & LEARNING

OPTIMIZATION ALGORITHMS
• AdaGrad
• AdaDelta
• Adam
• RMSProp
• Hessian-Free Optimization
• …

REPARAMETERIZATION
• Batch Normalization
• Weight Normalization
• …

REGULARIZATION
• Dropout
• DropConnect
• …

MODEL ARCHITECTURES

BUILDING BLOCKS
• Spatial/TemporalPooling
• AttentionMechanism
• GatedRecurrentUnits
• Beam-search for sequence generation
• Variable-length sequence modeling
• …

ARCHITECTURES
• Inception (Google)
• VGG (Oxford University)
• Encoder-Decoder Framework
• End-to-end Models
• …

SOFTWARE

• Theano
  • Blocks + Fuel
  • Keras
  • Lasagne
  • PyLearn2*
• TensorFlow
• Torch7
• Caffe…

GENERAL

• GPUs
• Data
Computation Graphs
Matrix-Vector Multiplication

\[ y \leftarrow Ax \]
\[ A \in \mathbb{R}^{m \times n} \]
\[ x \in \mathbb{R}^n \]
\[ y \in \mathbb{R}^m \]
Computation Graphs
Indexing

 SYMBOL TYPE data type
 symbolic variable
 operator

 MATRIX float

 Symbolic variable

 A
 MATRIX float

 i
 VECTOR int

 Indexing

 B

 A

 i

 B

 2 5 0
Computation Graphs
Graph Optimization

\[ z \leftarrow \frac{xy}{y} \]

\[ z = \frac{x \cdot y}{y} \]

Optimization

Diagram:
- Node `div` with input `x` and `y`,
- Node `mul` with input `x` and `y`,
- Node `z` with input from `div`.
- Labels: `x`, `y`, `z` as SCALAR float.
Computation Graphs
Automatic Differentiation

\[ y \leftarrow x^2 \]
Computation Graphs
Automatic Differentiation

\[ y \leftarrow x^2 \]

GRAD(y, x)
Computation Graphs
Automatic Differentiation

\[ y \leftarrow x^2 \]

\[ \text{square} \]

\[ x \quad \text{SCALAR float} \]

\[ y \quad \text{SCALAR float} \]

\[ \text{mul} \]

\[ 2 \quad \text{SCALAR float} \]

\[ \frac{\partial y}{\partial x} \leftarrow 2x \]
AUTOMATIC DIFFERENTIATION IS AN EXTREMELY POWERFUL FEATURE FOR DEVELOPING MODELS WITH DIFFERENTIABLE OPTIMIZATION OBJECTIVES
What’s new?

OPTIMIZATION & LEARNING

OPTIMIZATION ALGORITHMS
• AdaGrad
• AdaDelta
• Adam
• RMSProp
• Hessian-Free Optimization
• ...

REPARAMETERIZATION
• Batch Normalization
• Weight Normalization
• ...

REGULARIZATION
• Dropout
• DropConnect
• ...

MODEL ARCHITECTURES

BUILDING BLOCKS
• Spatial/Temporal Pooling
• Attention Mechanism
• Gated Recurrent Units
• Beam-search for sequence generation
• Variable-length sequence modeling
• ...

ARCHITECTURES
• Inception (Google)
• VGG (Oxford University)
• Encoder-Decoder Framework
• End-to-end Models
• ...

SOFTWARE

• Theano
• Blocks + Fuel
• Keras
• Lasagne
• PyLearn2*
• TensorFlow
• Torch7
• Caffe...

GENERAL

• GPUs
• Data
What’s new?

GPUs

NVIDIA®
What’s new?
DEEP LEARNING is NOT only meant literally, but more importantly it is about learning solutions to problems in a **fully automated** way.
# Recommended Material

## Module 1: Neural Networks

- **Image Classification**: Data-driven Approach, k-Nearest Neighbor, train/val/test splits
  - L1/L2 distances, hyperparameter search, cross-validation
- **Linear classification**: Support Vector Machine, Softmax
  - parametric approach, bias trick, hinge loss, cross-entropy loss, L2 regularization, web demo
- **Optimization**: Stochastic Gradient Descent
  - optimization landscapes, local search, learning rate, analytic/numerical gradient
- **Backpropagation, Intuitions**
  - chain rule interpretation, real-valued circuits, patterns in gradient flow
- **Neural Networks Part 1**: Setting up the Architecture
  - model of a biological neuron, activation functions, neural net architecture, representational power
- **Neural Networks Part 2**: Setting up the Data and the Loss
  - preprocessing, weight initialization, batch normalization, regularization (L2/dropout), loss functions
- **Neural Networks Part 3**: Learning and Evaluation
  - gradient checks, sanity checks, babysitting the learning process, momentum (+nesterov), second-order methods, Adagrad/RMSprop, hyperparameter optimization, model ensembles

## Module 2: Convolutional Neural Networks

- **Convolutional Neural Networks**: Architectures, Convolution / Pooling Layers
  - layers, spatial arrangement, layer patterns, layer sizing patterns, AlexNet/ZFNet/VGGNet case studies, computational considerations
- **Understanding and Visualizing Convolutional Neural Networks**
  - ISNE embeddings, deconvnets, data gradients, fooling ConvNets, human comparisons
- **Transfer Learning and Fine-tuning Convolutional Neural Networks**

---

## Course Instructors

- Fei-Fei Li
- Andrej Karpathy
- Justin Johnson

[http://cs231n.stanford.edu/](http://cs231n.stanford.edu/)
[http://cs231n.github.io/](http://cs231n.github.io/)
Recommended Material

CS244d: Deep Learning for Natural Language Processing

Course Instructor
Richard Socher

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, window-based neural networks, recurrent neural networks, long-short-term-memory models, recursive neural networks, convolutional neural networks as well as some very novel models involving a memory component. Through lectures and programming assignments students will learn the necessary engineering tricks for making neural networks work on practical problems.

http://cs244d.stanford.edu/
Recommended Material

INTRODUCTION

• Tutorial on Neural Networks (Deep Learning and Unsupervised Feature Learning): http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial
Recommended Material

PARAMETER INITIALIZATION


BATCH NORMALIZATION


DROPOUT


Recommended Material

OPTIMIZATION & TRAINING

Recommended Material

COMPUTER VISION


Recommended Material

NATURAL LANGUAGE PROCESSING