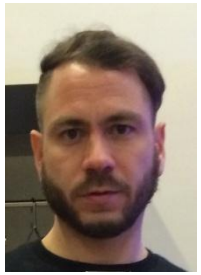
Deep Learning

History, Approaches, Applications



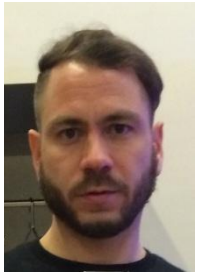
An Introduction by:

Thomas Lidy & Jan Schlüter

First Vienna Deep Learning Meetup

April 7, 2016 @ sektor5, Vienna

Thomas Lidy



Audio Analysis & Machine Learning Aficionado

1998 - 2006	Computer Science, TU Wien
2003 - 2004	Telecommunications & Sound, Spain
2004 - 2012	Research Assistant Music IR, TU Wien
2008 - 2013	Founder & CEO Spectralmind
2014	Data Mining in Oil Industry
2015 - 2016	MusicBricks Project TU Wien
	PhD Cand. Deep Learning for Music
2016	Consultant Music Tech & Machine Learning

Jan Schlüter

PhD Researcher & Deep Learning Practitioner

2005 - 2008	BSc Computer Science, University of Hamburg
2008 - 2011	MSc Computer Science, TU Munich
2009	University of Helsinki
2011 -	Researcher at the Austrian Research Institute for Artificial Intelligence (OFAI), Intelligent Music Processing and Machine Learning Group

Core developer of *Lasagne*: github.com/Lasagne/Lasagne
Current maintainer of *cudamat*: github.com/cudamat/cudamat

Outline

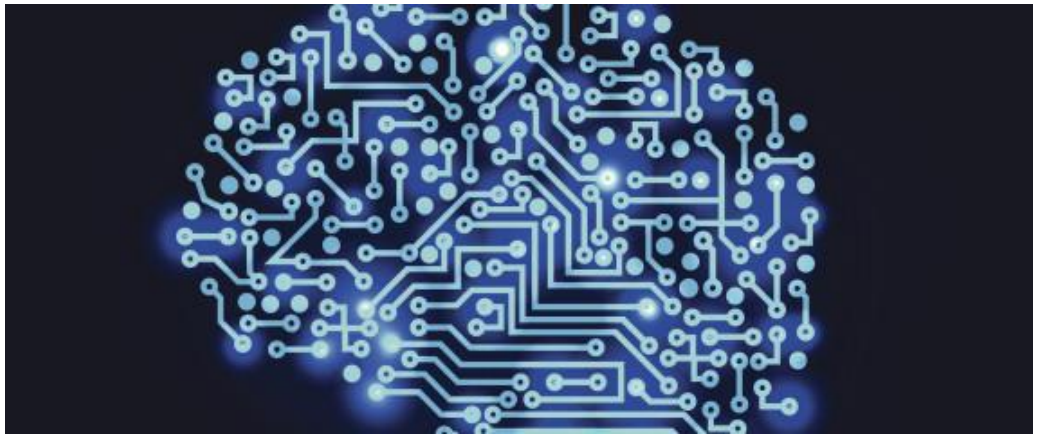
I: Presentation

- A Brief History of Neural Networks
- What is Deep Learning?
- What is it good for? - Application Examples
- Software for Deep Learning
- About this Meetup Group & Next Events

II: Practical Session

- Who is here and why?
- Discussion of hot topics

A Brief History of Neural Networks




Deep Learning

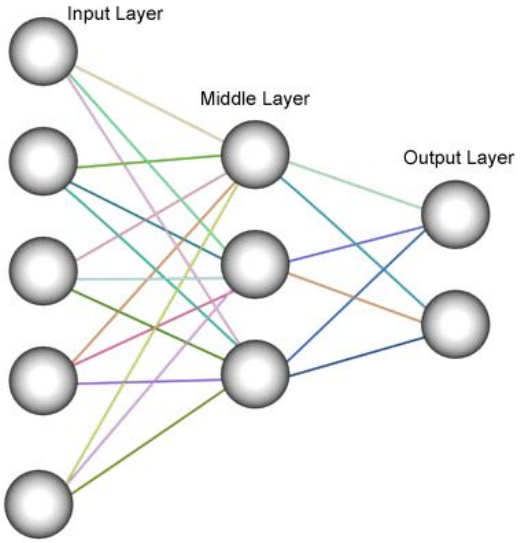
Neural Networks

are loosely inspired by biological neurons
that are interconnected and communicate with each other


Deep Learning

Neural Networks

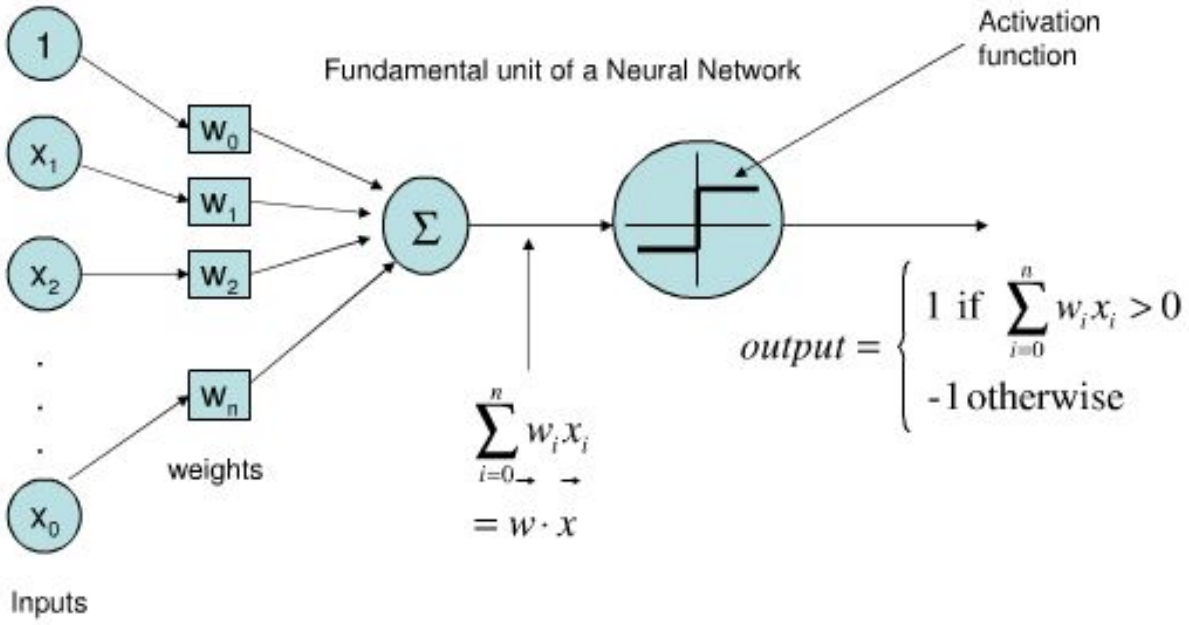
In reality, a neural network is just a **mathematical function**:



- in which the **“neurons”** are **sets of adaptive weights**, i.e. numerical parameters that are **tuned by a learning algorithm**,
- and which has the capability of approximating non-linear functions of their inputs.

Origins of Neural Networks

1958: Rosenblatt: The Perceptron



Linear binary classifier using a step function

For the first time a NN could solve simple classification problems merely from training data


Deep Learning

Ups & Downs of Neural Networks

NNs and AI have experienced several hype cycles, followed by disappointment, criticism, and funding cuts:

1950s - 70s: Golden years of AI (funded by DARPA):
solve algebra, play chess & checkers, reasoning, semantic nets
"within ten years a digital computer will be the world's chess champion"

1969: shown that XOR problem cannot be solved by Perceptron
(led to the invention of multi-layer networks later on)

Mid 1970s: Chain reaction that begins with pessimism in the AI community, followed by pessimism in the press, followed by a severe cutback in funding, followed by the “end” of serious research (“AI winter”)

Ups & Downs of Neural Networks

1980s: Governments (starting in Japan) and industry provide AI with billions of dollars. Boom of “expert systems”.

1986: Backpropagation had been invented in the 1970s, but only 1986 it became popular through a famous paper by David Rumelhart, Geoffrey Hinton, and Ronald Williams. It showed that also complex functions became solvable through NNs by using multiple layers.

Late 1980s: Investors - despite actual progress in research - became disillusioned and withdrew funding again.

Why no Deep Learning in the 1980s?

Neural Networks could not become “deep” yet - because:

- Computers were slow. So the neural networks were tiny and could not achieve (the expected) high performance on real problems.
- Datasets were small. There were no large datasets that had enough information to constrain the numerous parameters of (hypothetical) large neural networks.
- Nobody knew how to train deep nets. Today, object recognition networks have > 25 successive layers of convolutions. In the past, everyone was very sure that such deep nets cannot be trained. Therefore, networks were shallow and did not achieve good results.

citing Ilya Sutskever
http://yyue.blogspot.co.at/2015/01/a-brief-overview-of-deep-learning.html

Ups & Downs of Neural Networks

1991: Hornik proved 1 hidden layer network can model any continuous function (universal approximation theorem)

1991/92 Vanishing Gradient: problem in multi-layer networks where training in front layers is slow due to backpropagation diminishing the gradient updates through the layers. Identified by Hochreiter & Schmidhuber who also proposed solutions.

1990s - mid 2000s:

Due to lack of computational power, interest in NNs decreased again and other Machine Learning models, such as Bayesian models, Decision Trees and Support Vector Machines became popular.

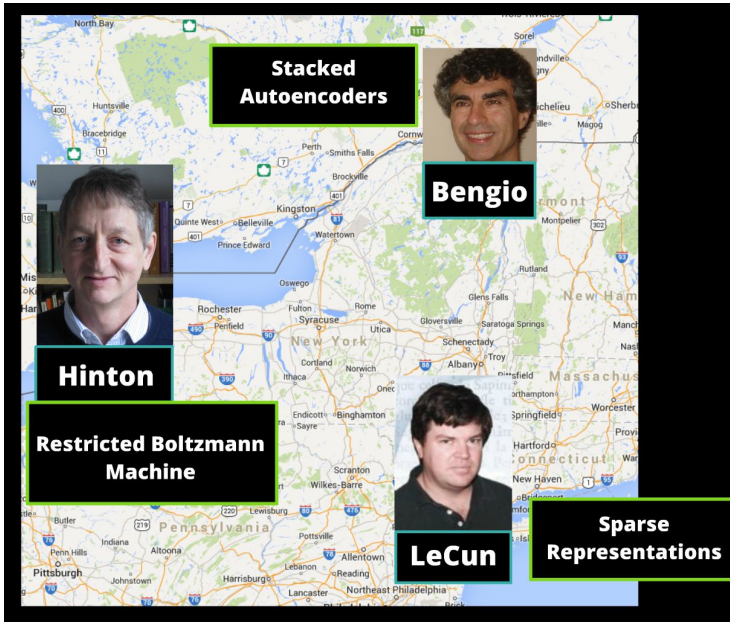

Deep Learning

Resurrection of Deep Learning in the 2000s

2000s: Hinton, Bengio and LeCun (“The fathers of the age of deep learning”) join forces in a project

They overcome some problems that caused deep networks not to learn anything at all

2006: Breakthrough with Layer-wise pre-training by unsupervised learning (using RBMs)



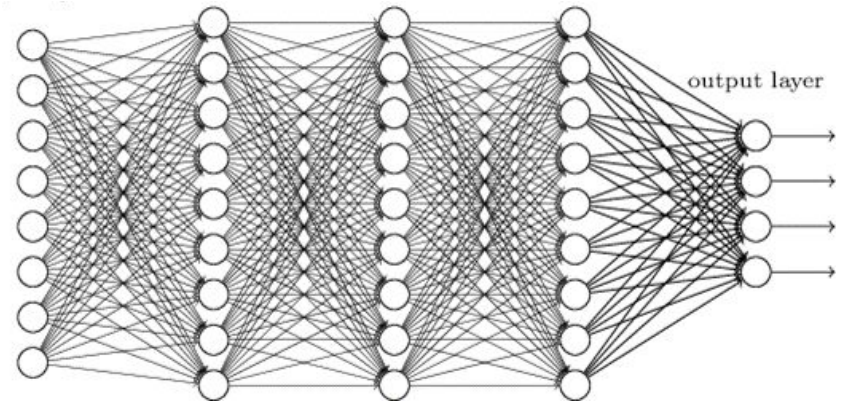
2010s: Important new contributions:

- Simpler initialization (without pre-training)
- Dropout
- Simplier activations: Rectifier Units (ReLUs)
- Batch Normalization

→ not a re-invention of NNs but paved the way for very deep NNs

https://www.datarobot.com/blog/a-primer-on-deep-learning/

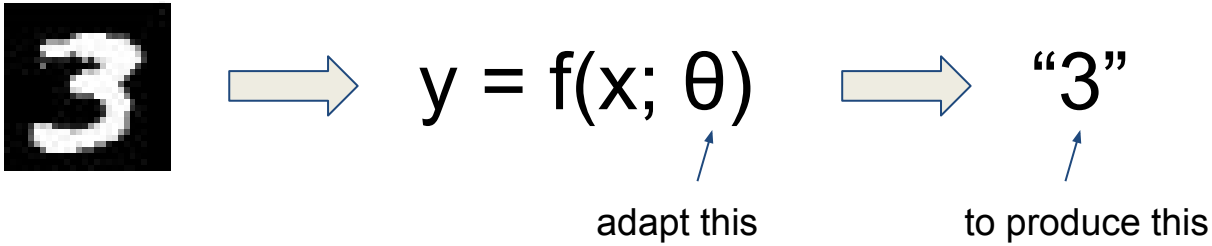
What is Deep Learning?




Deep Learning

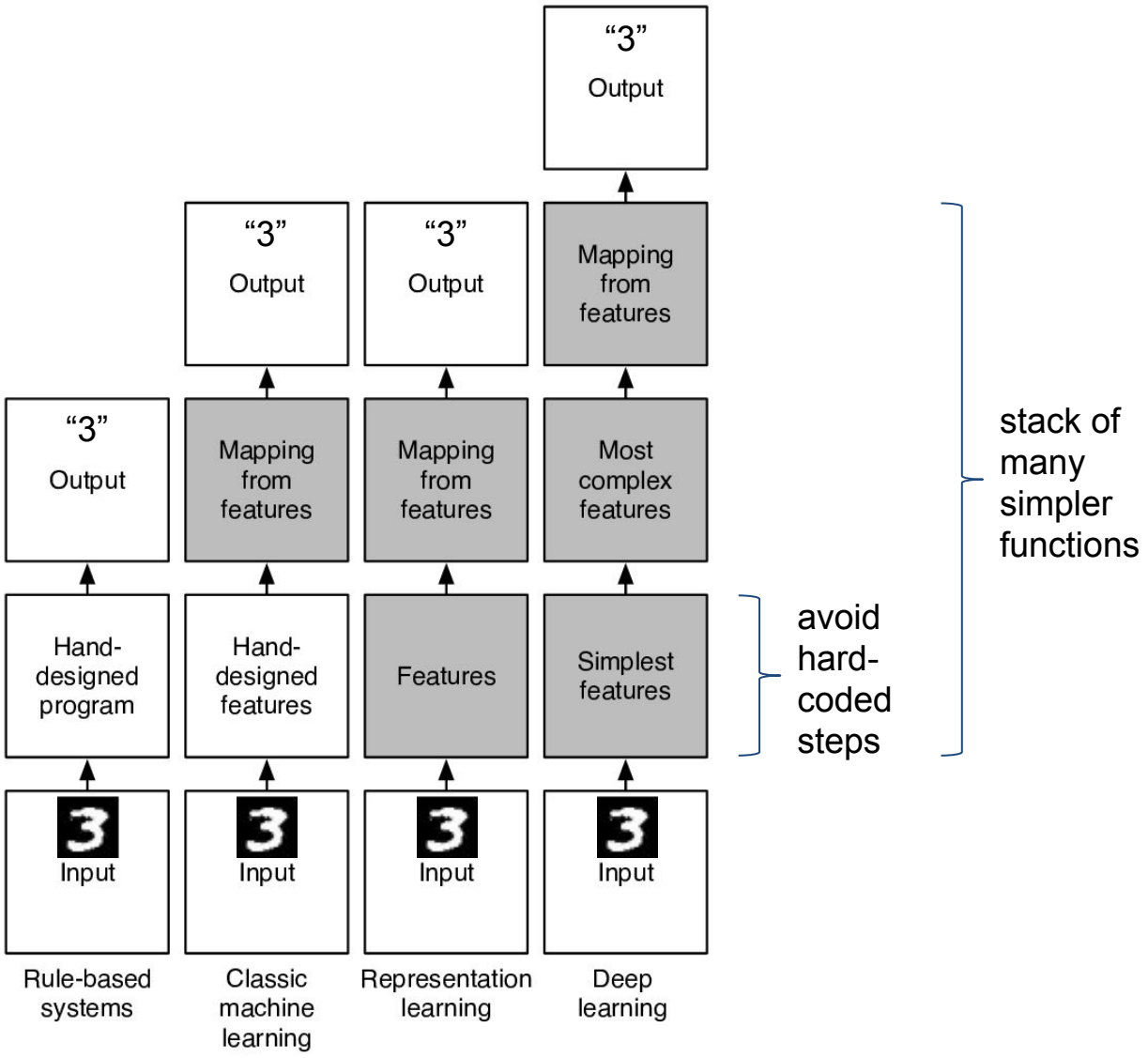
What is Deep Learning?

- **Machine learning:** Express problem as a function, automatically adapt parameters from data



- **Deep learning:** Learnable function is a *stack of many simpler functions* that often have the same form
  - Often, it is an artificial neural network
  - Often, one tries to minimize hard-coded steps

Machine Learning Paradigms



Inspiration for Going “Deep”

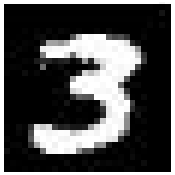
- Humans organize their ideas and concepts **hierarchically**
- Humans **first learn simpler concepts** and then compose them to represent more abstract ones
- Engineers **break-up solutions into multiple levels** of abstraction and processing
- It would be good to **automatically learn** / discover these concepts

Y. Bengio, Deep Learning, MLSS 2015, Austin, Texas, Jan 2014


Deep Learning

Let's Get Concrete

- **Machine learning:** Express problem as a function, automatically adapt parameters from data
- **Deep learning:** Learnable function is a stack of *many simpler functions* that often have the same form



$$y = f(f(f(x;\theta_1);\theta_2);\theta_3)$$

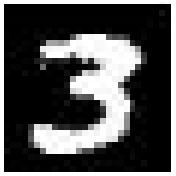
"3"

adapt this

to produce this

- Most commonly used functions:
  - Matrix product: $f(x; \theta) = W^T x$
  - 2D Convolution: $f(x; \theta) = x * W$
  - Subsampling
  - Element-wise nonlinearities (sigmoid, tanh, rectifier)

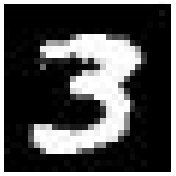
From Formula to "Neural Network"



$$y = f(f(f(x;\theta_1);\theta_2);\theta_3)$$

"3"

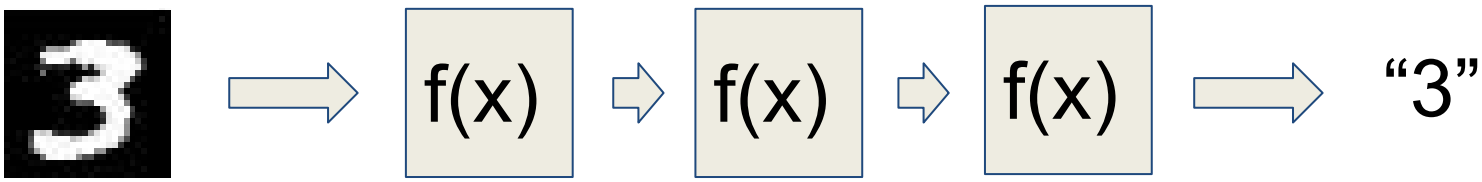
It is convenient to express repeated function application as a flow chart:



f(x) → f(x) → f(x) → "3"

Computation in **sequential steps of parallel processing**, often termed "layers". (Superficially similar to processing in the brain.)

From Formula to "Neural Network"

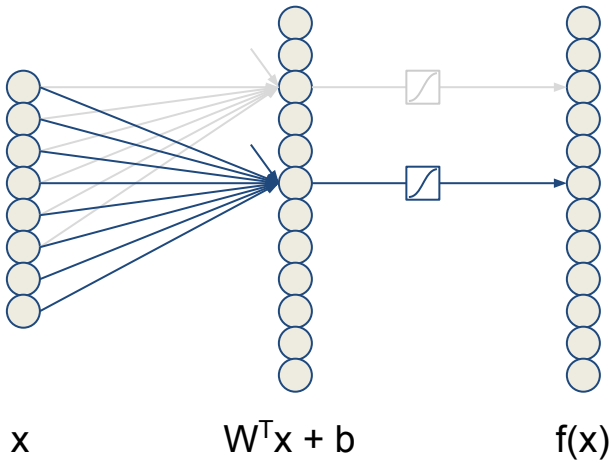


Computation in **sequential steps of parallel processing**, often termed "layers". (Superficially similar to processing in the brain.)

Typical "layer": matrix product followed by biased nonlinearity,

$$f(x) = \phi(W^T x + b)$$

Can be visualized as follows:



A single output value is computed as **a weighted sum of its inputs, followed by a nonlinear function**. The value is termed "neuron", the weighting coefficients are termed "connection weights". (Superficially similar to biological neurons.)

Mathematical Reasons for Going "Deep"

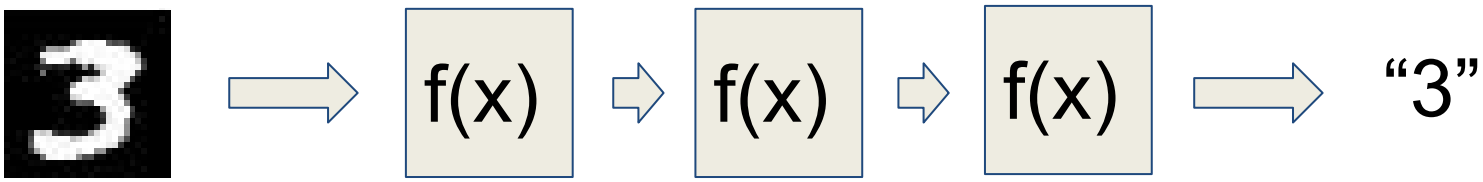
A neural network with a single hidden layer of enough units can approximate any continuous function arbitrarily well. In other words, it can solve whatever problem you're interested in!
(Cybenko 1998, Hornik 1991)

But:

- "Enough units" can be a very large number. There are functions representable with a small, but deep network that would require exponentially many units with a single layer.
  (e.g., Hastad et al. 1986, Bengio & Delalleau 2011)

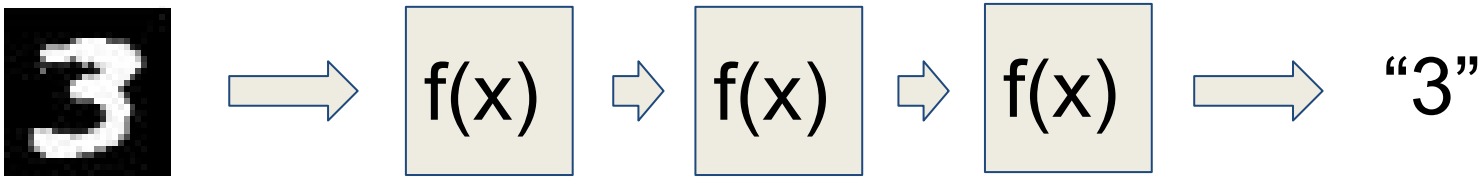
- The proof only says that a shallow network *exists*, it does not say how to find it. Evidence indicates that it is easier to train a deep network to perform well than a shallow one.

Training Neural Networks



- **Machine learning:** Express problem as a function, automatically adapt parameters from data.
- **Neural Networks:** Function consists of “layers”, each has some tunable values (e.g., connection weights).
- **Training**:
  - **Randomly initialize** tunable values
  - Define **cost function** telling how well the network does (on a set of training examples)
  - **Iteratively** adapt tunable values to **minimize value of cost function** (following simple gradient-based rule)
- For each step there exist recipes from the 1980s. Only the first step required better recipes for deep networks.

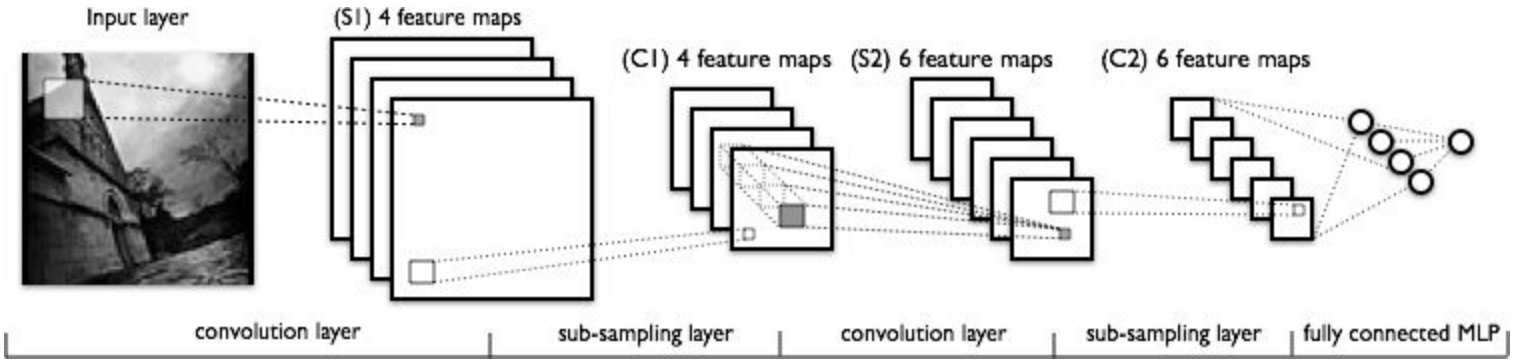
Popular Network Architectures



- Largest design space in employing deep learning: Which functions (or “layers”) to use, and in which combination?

- Two popular architectures:
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)

Convolutional Neural Networks (CNNs)



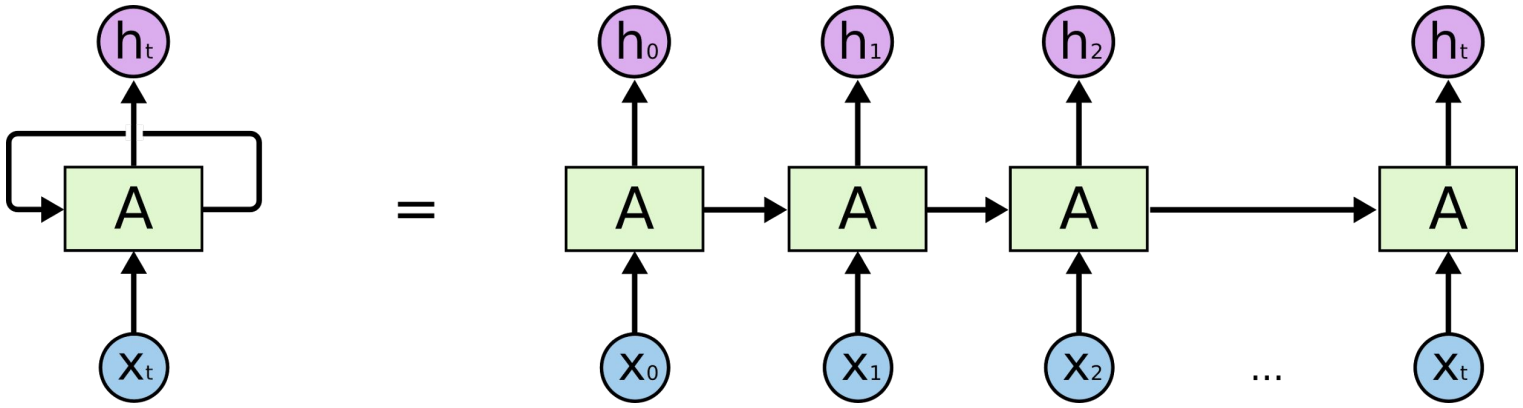
http://deeplearning.net/tutorial/lenet.html

Combines three types of layers:

- **Convolutional layer:** performs 2D convolution of 2D input with multiple learned 2D kernels
- **Subsampling layer:** replaces 2D patches by their maximum ("max-pooling") or average
- **Fully-connected layer:** computes weighted sums of its input with multiple sets of learned coefficients

Applies a nonlinear function after each linear operation (without, a deep network would be linear despite its depth).

Recurrent Neural Networks (RNNs)



- RNNs process **sequences** of same-sized elements
- **Recurrent layers** process an input element together with **their own output** from the previous element
- Useful for time series (text, sound, video)
- Can be seen as a deep network with shared weights, unrolled in time: depth = length of sequence

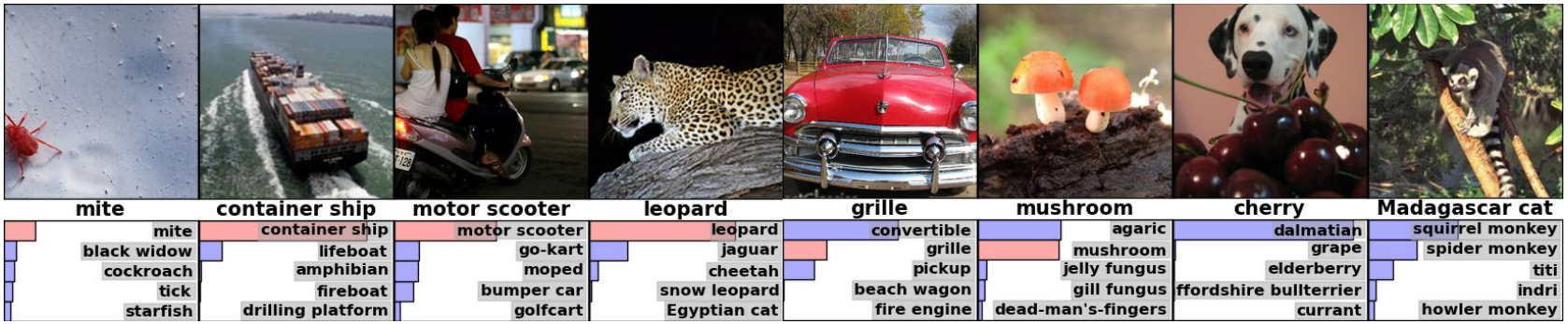
Aug 2015, http://colah.github.io/posts/2015-08-Understanding-LSTMs/

What is it good for?

Application Examples

Object Classification

IMAGENET Large Scale Visual Recognition Challenge:
1.2 million training images of 1000 classes

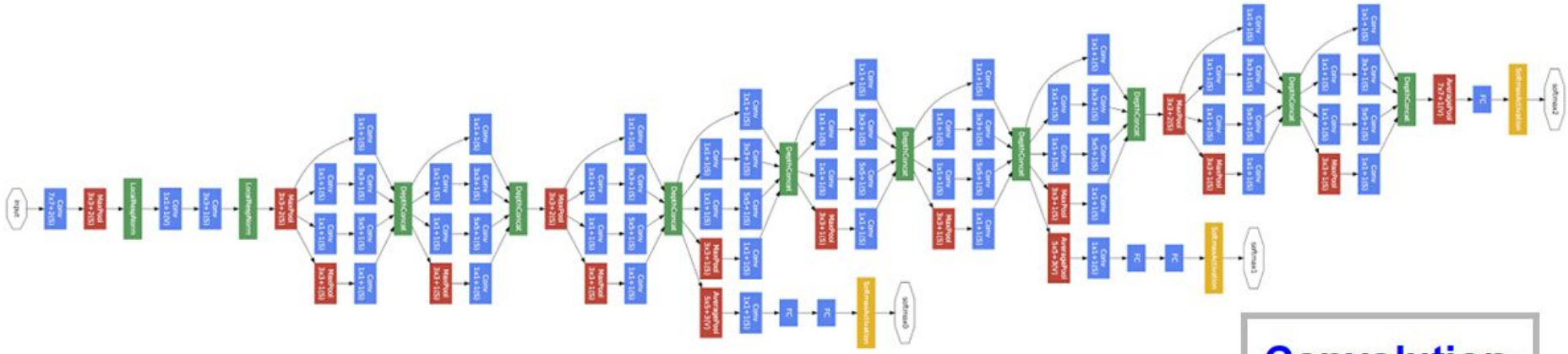


- 2012: AlexNet achieves 16.4% top-5 error (first deep net). Second best entry: 26.2%.
- 2013: Clarifai: 11.7% top-5 error, or 11.2% with extra data (most contestants used deep nets now).
- 2014: GoogLeNet: 6.7%. Andrej Karpathy (a human): 5.1%.
- 2015: ResNets: 3.6%.

Object Classification

IMAGENET Large Scale Visual Recognition Challenge:
1.2 million training images of 1000 classes

GoogLeNet: 22 layers, with auxiliary classifiers

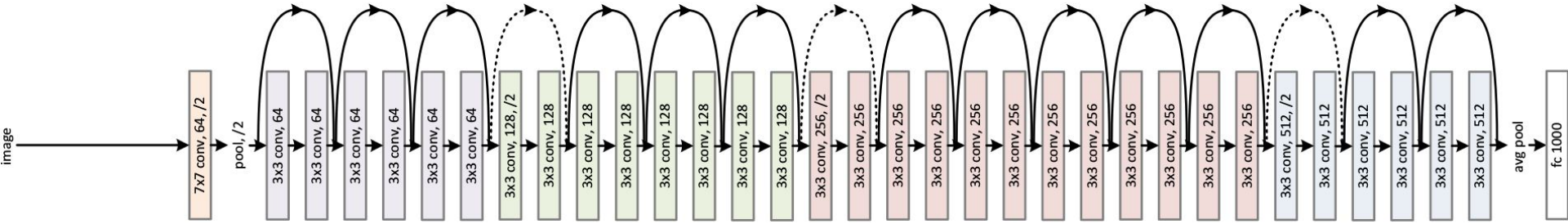


Sep 2014: Going Deeper with Convolutions, http://arxiv.org/abs/1409.4842

Object Classification

IMAGENET Large Scale Visual Recognition Challenge:
1.2 million training images of 1000 classes

ResNet: 152 layers, with shortcut connections (here: 38 layers)



Dec 2015: Deep Residual Learning for Image Recognition, http://arxiv.org/abs/1512.03385
Mar 2016: Identity Mappings in Deep Residual Networks, http://arxiv.org/abs/1603.05027

Image Captioning

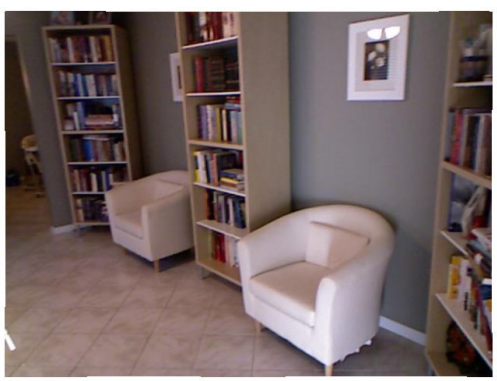
Input: Photograph. **Output:** Textual description.
Method: CNN to analyze image, RNN to output text

Describes without errors	Describes with minor errors	Somewhat related to the image	Unrelated to the image
 A person riding a motorcycle on a dirt road.	 Two dogs play in the grass.	 A skateboarder does a trick on a ramp.	 A dog is jumping to catch a frisbee.
 A group of young people playing a game of frisbee.	 Two hockey players are fighting over the puck.	 A little girl in a pink hat is blowing bubbles.	 A refrigerator filled with lots of food and drinks.

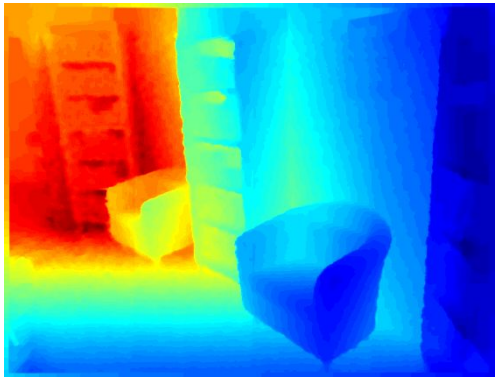
Nov 2014: Show and Tell: A Neural Image Caption Generator, http://arxiv.org/abs/1411.4555

Depth Estimation

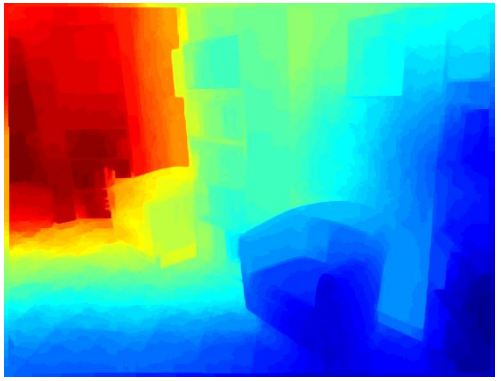
Input: Single photograph
Output: Its depth map (how far is each pixel from the observer)
Method: fully-convolutional network



input



ground truth

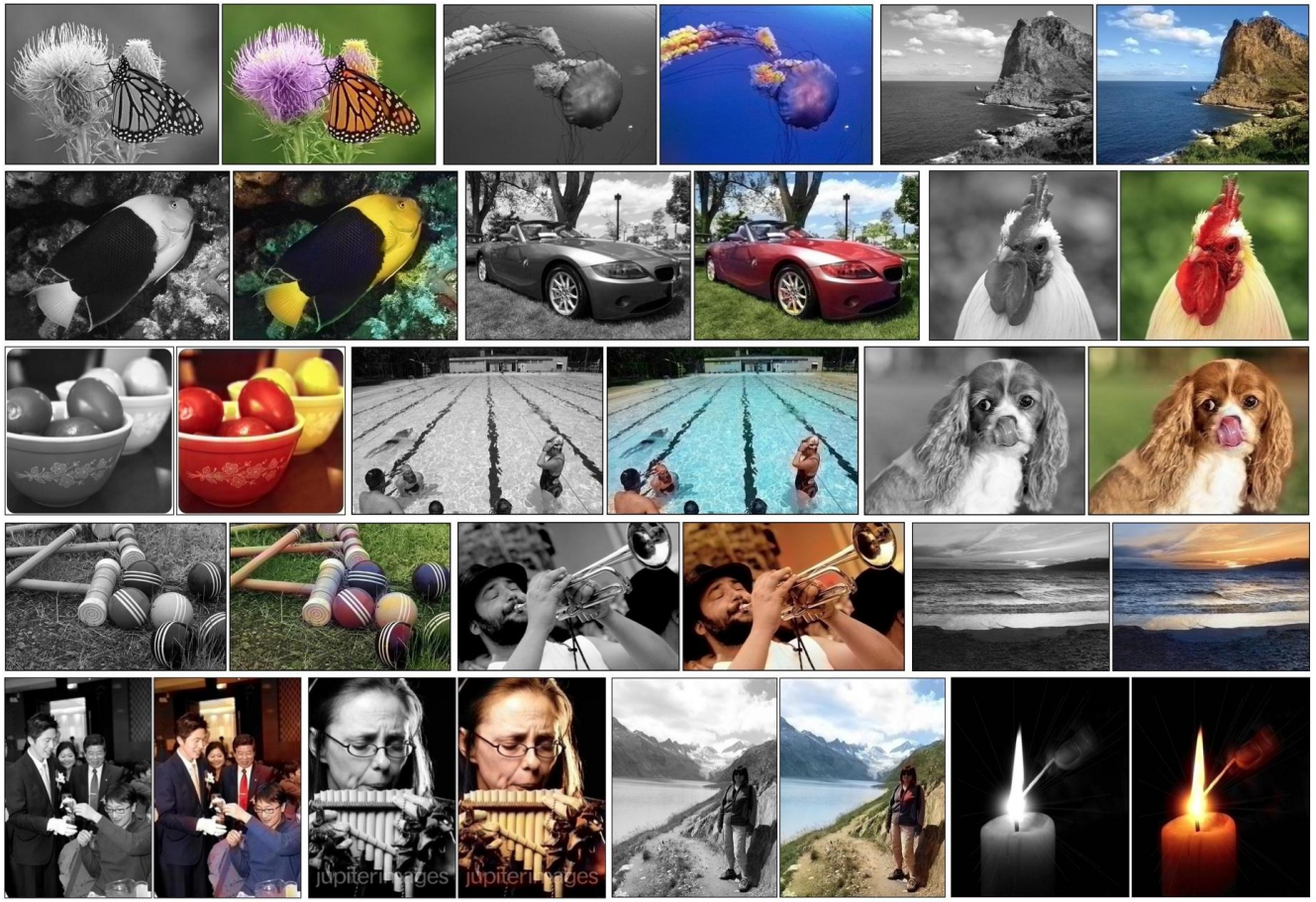


network prediction

Feb 2015: Learning Depth from Single Monocular Images Using Deep Convolutional Neural Fields, http://arxiv.org/abs/1502.07411

Image Colorization

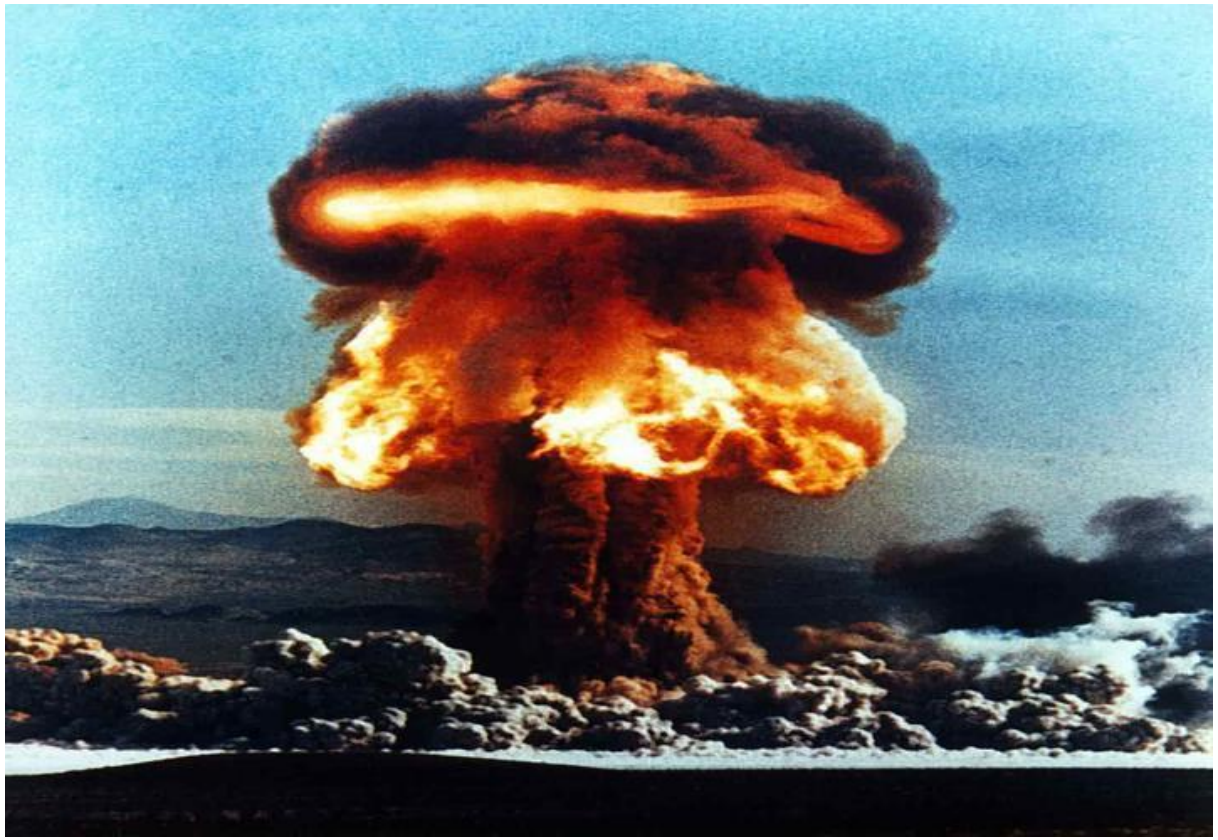
Input: Gray-scale image. **Output:** Colored image.
Method: fully-convolutional network



Mar 2016: Colorful Image Colorization, http://arxiv.org/abs/1603.08511, http://richzhang.github.io/colorization/

Inceptionism / DeepDream

Starting point: ConvNet trained for object classification
Use network in reverse: Modify the input image to maximize classification confidence (for given class, or the most probable)



Jun 2015: Inceptionism: Going Deeper into Neural Networks, http://googleresearch.blogspot.co.at/2015/06/inceptionism-going-deeper-into-neural.html

Inceptionism / DeepDream

Starting point: ConvNet trained for object classification
Use network in reverse: Modify the input image to maximize classification confidence (for given class, or the most probable)

Jun 2015: Inceptionism: Going Deeper into Neural Networks, http://googleresearch.blogspot.co.at/2015/06/inceptionism-going-deeper-into-neural.html

Inceptionism / DeepDream

Starting point: ConvNet trained for object classification
Use network in reverse: Modify the input image to maximize classification confidence (for given class, or the most probable)

Jun 2015: Inceptionism: Going Deeper into Neural Networks, http://googleresearch.blogspot.co.at/2015/06/inceptionism-going-deeper-into-neural.html


Deep Learning

Neural Style Transfer

Starting point: ConvNet trained for object classification
Use network in reverse: Find image that matches high-level representation of photograph and low-level repr. of painting

Aug 2015: A Neural Algorithm of Artistic Style, http://arxiv.org/abs/1508.06576

Image Generation

Generative Adversarial Networks: One network transforms random noise into images, a second one tries to distinguish generated from real images. Both are trained at the same time.

Photographs of bed rooms that do not actually exist

Nov 2015: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, http://arxiv.org/abs/1511.06434, https://github.com/Newmu/dcgan_code

Deep Learning

Image Generation

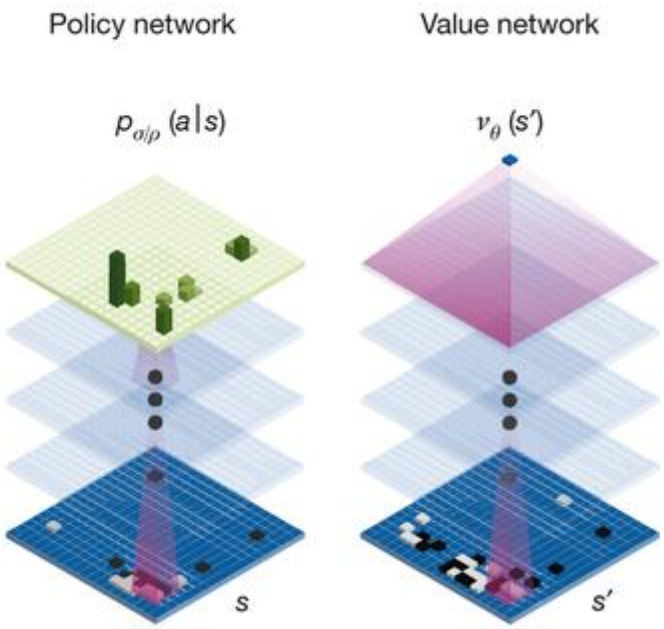
Generative Adversarial Networks: One network transforms random noise into images, a second one tries to distinguish generated from real images. Both are trained at the same time.

Generated album covers

Nov 2015: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, http://arxiv.org/abs/1511.06434, https://github.com/Newmu/dcgan_code

Google AlphaGo

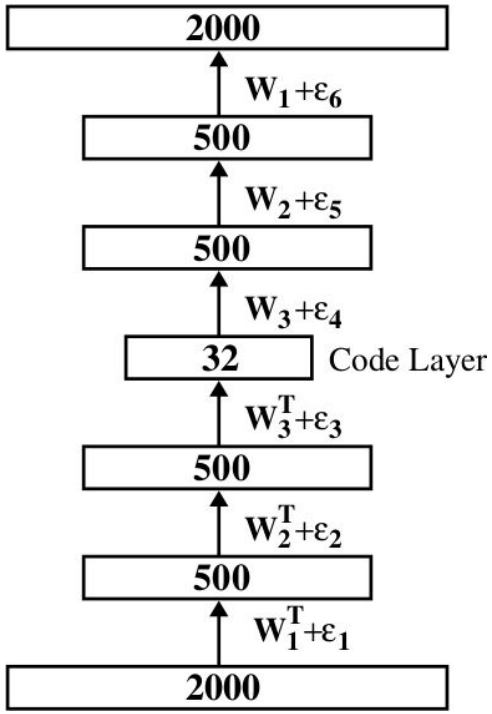
Input: Go board (stone positions and some helping markers)
Output: Position for next move (policy) or likely winner (value)
Method: CNN with supervised and reinforcement learning
Combined with Monte-Carlo Tree Search for playing

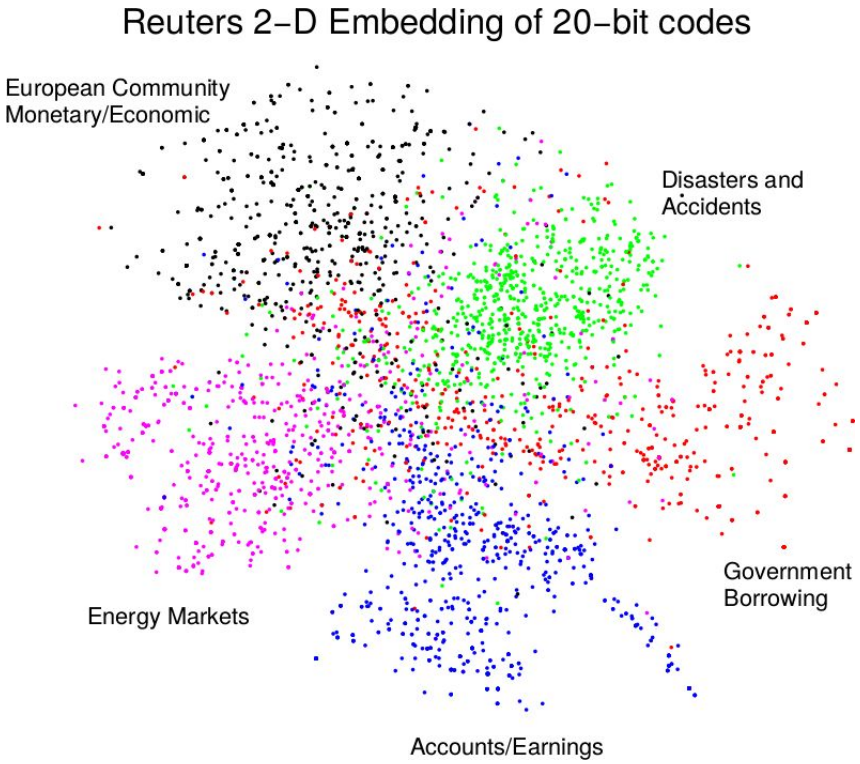


Nature 526, Jan 2016: Mastering the game of Go with deep neural networks and tree search

Semantic Hashing

Input and **Output:** Document word count vectors
Method: MLP auto-encoder
Learns to transform input to short bit codes and back.
Bit codes useable for very fast retrieval of similar documents.

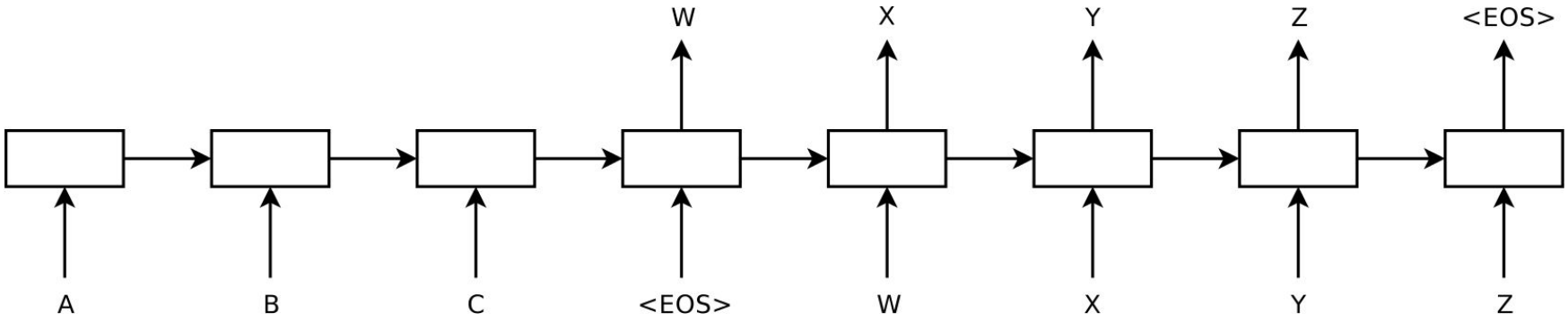




2007: Semantic Hashing, http://www.cs.utoronto.ca/~rsalakhu/papers/semantic_final.pdf

Machine Translation

Input: Sequence of English words
Output: Sequence of French words
Method: RNN (encode to vector, decode from vector)



Sep 2014: Sequence to Sequence Learning with Neural Networks, http://arxiv.org/abs/1409.3215

Text Generation

Input: a sequence of characters
Output: the most likely next character
Method: RNN (LSTM)

After training, can generate text from scratch (or from a seeding text), character by character. Must learn to combine characters to words, words to sentences, sentences to documents: can even learn document structure.

May 2015: The Unreasonable Effectiveness of Recurrent Neural Networks, http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Text Generation

Trained on Shakespeare

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

May 2015: The Unreasonable Effectiveness of Recurrent Neural Networks, http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Text Generation

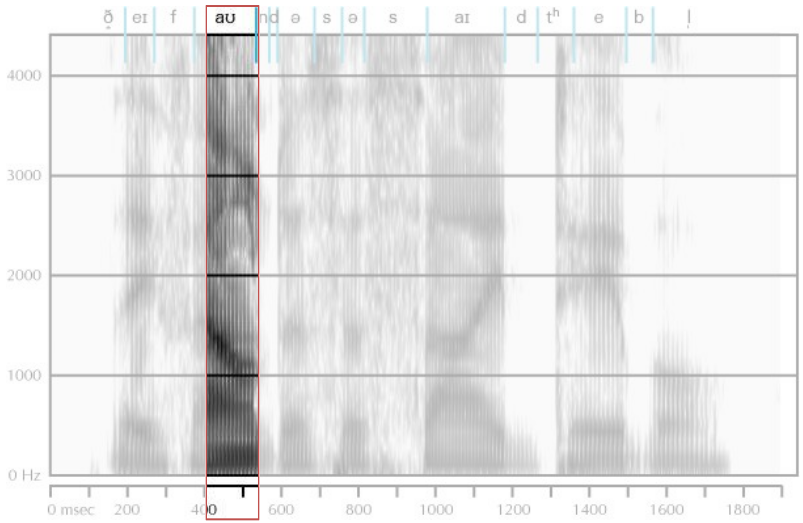
Trained on Linux source code

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

The Unreasonable Effectiveness of Recurrent Neural Networks, http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Speech Recognition

First successful networks in speech recognition:
Input: short spectrogram excerpt
Output: detected phoneme
Method: CNN
Has to be combined with word and language model, requires carefully-aligned training data.

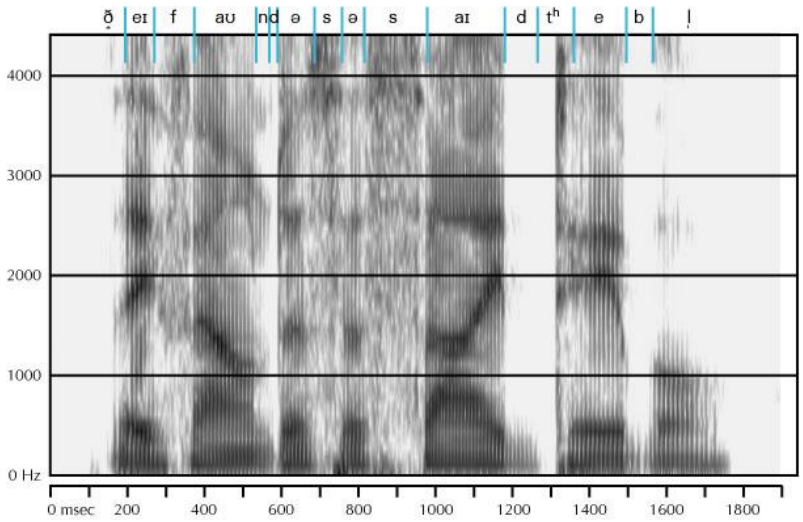


aʊ

Dec 2015: Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, http://arxiv.org/abs/1512.02595

Speech Recognition

State of the art truly learns from end to end:
Input: spectrogram of recorded sentence
Output: transcribed sentence as a sequence of characters
Method: CNN for initial processing, RNN for temporal context
Works with coarsely aligned training data, learns word and language model on its own, learns English and Mandarin with the same architecture.

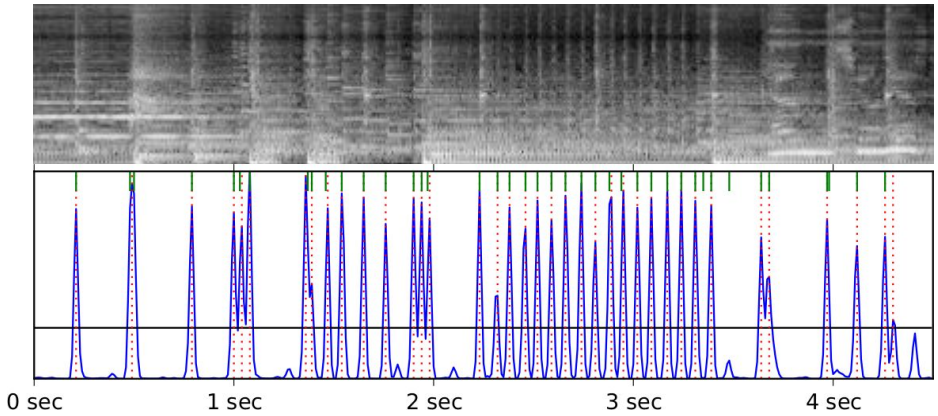


"They found us a side table."

Dec 2015: Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, http://arxiv.org/abs/1512.02595

(Very Basic) Music Perception

Input: log-frequency spectrogram (or excerpt)
Method: MLP or CNN

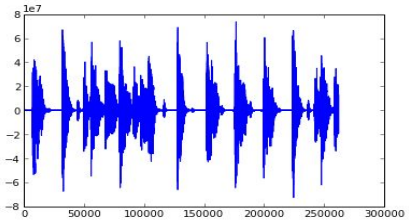


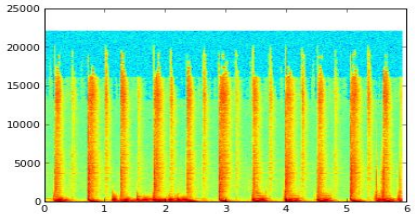
(Own work only:)
2012: Detect music, speech or both in radio broadcasts
2013: Use Semantic Hashing for fast music similarity search
2014: Detect onset times of all musical notes
2014: Find segments within music pieces (chorus, verse)
2015: Detect singing voice, with music-specific data augmentation

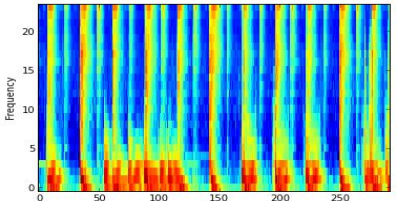
Music / Speech Discrimination

Tom Lidy: winner MIREX 2015 music/speech classification (99.73%)

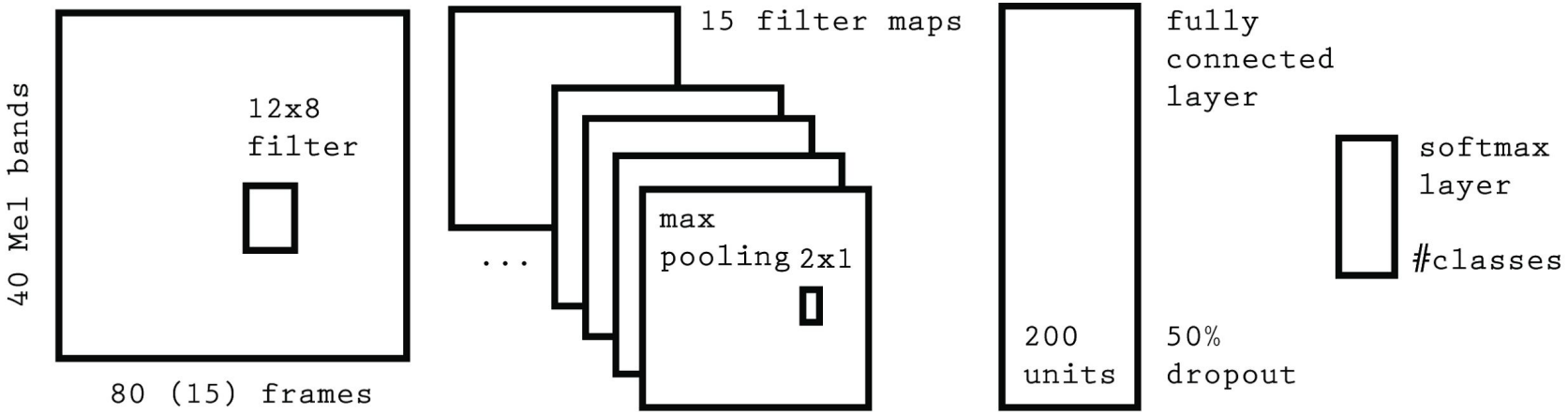
1. Pre-Processing: Waveform → Spectrogram → 40 Mel bands → Log scale







2. CNN with 1 layer, 15 filter maps + 1 full layer (input: 15 40x80 frames per file)



Software Frameworks

Python:

“low”-level:

- Theano
- TensorFlow (Google)

high-level:

- Lasagne
- Keras
- Blocks
- Theanets
- Pretty Tensor (Google)
- ...

C++ / others*:

- Caffe
- Torch
- cuda-convnet2
- Chainer
- MXNet
- CNTK (Microsoft)
- Leaf (AutumnAI)
- ...

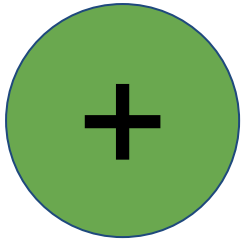
* (with bindings for other languages)

Hardware: GPU recommended! Most libraries support GPUs.

Some Conclusions

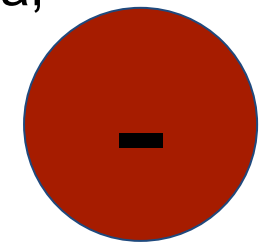
Neural Networks

- benefitted a lot from recent progress in hardware
- are applicable to a wide range of tasks
- tend to require less domain knowledge
- often outperform “hand-crafting” or feature design



Potential drawbacks:

- while reducing the need for feature design, they open up new design spaces (network architecture, training data, training method) which may be less well understood
- large networks usually require large data sets
- … and lots of computing power (and electricity)
- a lot of experimenting & parameter tweaking needed
- operation of a trained network is often hard to analyze (“black box”)




Deep Learning

Recommended Links

A Brief Overview of Deep Learning
http://yyue.blogspot.co.at/2015/01/a-brief-overview-of-deep-learning.html

A Primer on Deep Learning
https://www.datarobot.com/blog/a-primer-on-deep-learning/

Online Books/Courses with Implementation Examples
http://neuralnetworksanddeeplearning.com/
http://cs231n.stanford.edu/
http://deeplearning.net/tutorial

Press: Deep Learning And Machine Intelligence Will Eat The World
http://www.forbes.com/sites/anthonykosner/2014/12/29/tech-2015-deep-learning-and-machine-intelligence-will-eat-the-world/#1658f375282c

More Readings: http://deeplearning.net/reading-list/

More Software Links: http://deeplearning.net/software_links/

Questions?

Questions to the audience

- Who has worked with Neural Networks before?
- Who currently uses Deep Learning?
  - … in research?
  - … in industry?
- If so, which software do you use?
  - Caffe? Torch? Theano? Others?
- Who can talk about a topic in this or a future meetup?

Practical Session

What are your interests in Deep Learning? (& this meetup :-)

1. Write your questions or topics of interests onto yellow Post-It
2. Assign them to:

Research	Software	Applications

For experienced Deep Learners:

- add topics and/or applications that you work or worked on in a Post-it color other than yellow!

Let's then discuss! :-)


Deep Learning

Next Meetups

- **May 9:**
  Alex Champandard:
  “Neural Networks for Image Synthesis“

- **June 6:**
  Gregor Miba: “Recurrent Neural Networks and TensorFlow”
  Jan Schlüter:
  “Open-source Deep Learning with Theano and Lasagne”

Want to hold a talk as well? Get in touch!

Also check out the Discussion Board:
http://www.meetup.com/Vienna-Deep-Learning-Meetup/messages/boards/


Deep Learning

IoT Meetup

IoT meetup is **looking for speakers** for the upcoming events:

- **May 4: IoT & Big Data**
- **June 1: IoT & AI / Machine Learning**
  http://www.iot-vienna.at

IoT VIENNA

Thanks a lot to Sektor5
for hosting us here!

sektor5
coworking spaces wien