Activities report from August 2012 to April 2013

Telmo Amaral

26th April 2013

Project "Reusable Deep Neural Networks: Applications to Biomedical Data"
(PDTC/EIA-EIA/119004/2010)

Instituto de Engenharia Biomédica (INEB)
Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Contents

1 Introduction **3**

2 Introduction to deep learning **3**

3 Visit to LISA lab **3**
3.1 Input from Prof. Bengio 3
3.2 Activities and notes 5

4 Notes on existing software **5**

5 Introduction to convolutional neural networks **7**

6 Notes on UCI biomedical datasets **7**

7 Notes on relevant conferences **8**

A Slides: Introduction to deep learning **10**

B Slides: Introduction to convolutional neural networks **14**

C Spreadsheet: Notes on UCI biomedical data sets **17**

References **19**

1 Introduction

This report describes several activities I have undertaken since the beginning of my contract in August 2012 until the end of the project's first year, at the start of May, 2013. Each of the following sections addresses an individual activity. Activities are ordered more or less chronologically.

Besides the tasks described here, work related with the training of restricted Boltzmann machines (RBMs) and stacked auto-encoders is described in detail in two separate technical reports [1, 2].

2 Introduction to deep learning

My work in the project started in August 2012, with a review of introductory materials about deep learning, which included the first three sections of the 2009 article/book "Learning deep architectures for AI" by Bengio [4], as well as the 2010 review paper "Deep machine learning – a new frontier in artificial intelligence research" by Arel et al. [3].

Those readings allowed me to become acquainted with the concepts of shallow versus deep architectures, the motivations for the use of deep architectures (such as their inspiration from nature and their theoretical advantages in terms of efficiency), and the limitations associated with deep architectures until the breakthrough led by Hinton et al. [8] in 2006.

I prepared an overview of these topics and presented it in the project meeting of October 12, 2012. This presentation included a first list of some publicly available software implementations of deep learning models. It is included in Appendix A of this report.

3 Visit to LISA lab

In January 2013, for a period of four weeks, I visited the LISA machine learning laboratory[1] at the University of Montreal, headed by Prof. Yoshua Bengio. LISA is one of the most important research groups worldwide in the area of deep learning, routinely producing state-of-the-art research.

3.1 Input from Prof. Bengio

Prof. Bengio offered some comments of particular relevance to our project, as well as some feedback on the work carried out to date within the project, regarding the training of RBMs, which I have demonstrated to him. His comments are summarised in the following paragraphs.

Machine learning models in general need large amounts of training data. Often, there is not a lot of data available in biomedical applications, especially

[1] See https://www.iro.umontreal.ca/rubrique.php3?id_rubrique=27&lang=en.

data annotated by experts. In order to train deep learning models aimed at biomedical applications, it would be important to have at least a large amount of non-labelled data, for unsupervised pre-training.

Computer vision is where deep learning excels the most. A drop in object classification error rate from about 30% to less than 16% has been recently achieved using deep convolutional nets, as reported by Krizhevsky et al. [9].

The unsupervised training of auto-encoders could offer an opportunity for the use of alternative cost functions, given that auto-encoders are trained to reconstruct their own inputs. Moreover, auto-encoders are much simpler to train than RBMs. These remarks by Prof. Bengio prompted our own work with auto-encoders, started by Luís Alexandre and continued by myself and Chetak Kandaswamy, as described in a separate technical report [2].

Regarding the unsupervised training of RBMs, division of the training data into mini-batches is important not only to avoid operations with very large matrices, but also in terms of efficiency. Using all the available training data at each training epoch (iteration) would be excessive, because we don't need to update the model's parameters precisely in the right direction at each epoch; a smaller amount of data is sufficient to update the parameters in the right general direction, and much less time-consuming. An analogy can be drawn with a person going from a point A to a point B in a number of steps: the person doesn't need to worry about moving precisely in the direction of point B at each step taken; a number of roughly precise steps will lead to B just the same.

A clarification regarding the sampling mechanism used during the training of RBMs: what we're using is so-called *block* Gibbs sampling, as opposed to Gibbs sampling "one bit at at time". Block Gibbs sampling allows to more efficiently explore the modes of the multi-dimensional data distribution, since all bits are samples in parallel. This is possible in *restricted* Boltzmann machines, but not in unrestricted ones.

On the subject of sampling from a trained RBM, in theory, when starting from a random visible vector, it is necessary to perform a large number of Gibbs sampling steps to obtain a good sample, like we did in our implementation. However, once this "burn in" is achieved, each new step, if done without restarting from a random vector, is enough to yield a good sample. In practice, if we start not from a random vector but from a training vector, we can assume that the "burn in" is already done and start drawing samples right away. These techniques can greatly speed up the generation of samples.

The fact that the type of RBM we've implemented is meant for use with binary data doesn't preclude its use with real-valued data scaled between 0 and 1. Our discouraging results with real-valued data, in particular with the Iris data set, could be due to the choice of learning rate. In some cases it is important to adjust the learning rate throughout the training. (Later, in our work with auto-encoders, we have implemented adaptive learning rates.)

In case we pursue experiments with RBMs, we should now focus on using more efficient code, so that we may use data sets with large numbers of features and examples.

3.2 Activities and notes

During my visit, I did a more in-depth reading of some parts of Prof. Bengio's 2009 book [4], in particular the sections on neural networks for deep architectures (Section 4) and on energy-based models and Boltzmann machines (Section 5). Prof. Bengio's lecture notes "Introduction to Gradient-Based Learning"[2] proved to be very useful as an introduction to gradient descent learning methods. The material I read on convolutional neural networks later helped me to prepare the presentation described in Section 5.

I spent some time also with the previously mentioned paper by Krizhevsky et al. [9], which reports recent developments on the use of convolutional neural networks, as well as with the pre-print of a new review on deep learning by Bengio et al. [5], which is now publicly available. While not precluding the reading of the 2009 book, this article contains a lot of information on recent developments and research directions.

In order to demonstrate my Matlab implementation of RBM training algorithms to Prof. Bengio, I spent some time tidying up its experimental scripts and collecting results into a report. This ultimately became the technical report on experiments with RBMs being submitted separately [1].

I read some introductory materials on the Python programming language, the NumPy numerical library, as well as Theano, a Python library that facilitates the development of deep learning models, while giving the option of performing their training on one or more graphical processing units (GPUs). The LISA lab was the ideal place to become acquainted with these programming technologies, as Theano has been developed by researchers at LISA and is extensively used in their work.

Throughout the visit, I collected some notes on various software implementations of deep learning models that are publicly available. Those notes are presented in Section 4.

I had the chance to attend a seminar presented by a leading researcher of the Canadian company D-Wave Systems Inc.[3], on the use of quantum computing for training deep learning models. This company specialises in the development and commercialisation of quantum computers, an emerging field with potential deep learning applications. I attended also a brief "tea talk" about a paper co-authored by Prof. Pedro Domingos, a Portuguese researcher based in the U.S. who works with deep architectures [6].

4 Notes on existing software

The following is a list of some software implementations of deep learning models that are publicly available, collected during my introductory readings and also during my visit the LISA lab. In each case, the group or researcher responsible for the software is given (with a link to the relevant web page), as well as the

[2] See http://www.iro.umontreal.ca/~bengioy/ift6266/H12/html/gradient_en.html.
[3] See http://www.dwavesys.com.

programming language and relevant libraries used, and the models that are implemented.

1. LISA lab
   Deep Learning Tutorials
   http://deeplearning.net/tutorial/
   Python / Numpy / Theano
   - logistic regression
   - multilayer perceptrons
   - deep convolutional networks
   - autoencoders, denoising autoencoders
   - stacked denoising autoencoders
   - restricted boltzmann machines
   - deep belief networks
2. LISA lab
   Wiki, "fundamental research projects"
   http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/
   C++ / PLearn
   - deep belief networks
   - stacked autoassociators
3. Rasmusberg Palm
   DeepLearnToolbox
   https://github.com/rasmusbergpalm/DeepLearnToolbox
   Matlab
   - deep belief networks
   - stacked autoencoders
   - convolutional neural networks
   - convolutional autoencoders
   - vanilla neural networks
4. Ruslan Salakhutdinov, Geoff Hinton
   Training a deep autoencoder or a classifier on MNIST digits [7]
   http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html
   Matlab
   - restricted boltzmann machines
     - binary hidden and binary visible
     - Gaussian hidden and binary visible
   - deep autoencoders
   - deep belief networks ?

5. Ruslan Salakhutdinov
   Learning Deep Boltzmann Machines
   http://www.utstat.toronto.edu/~rsalakhu/DBM.html
   Matlab

   - deep boltzmann machines

6. Hugo Larochelle
   Efficient Learning of Deep Boltzmann Machines [10]
   an enhancement of 5
   http://www.dmi.usherb.ca/~larocheh/code/dbm_recnet.tar.gz
   Matlab

   - deep boltzmann machines

7. Andrej Karpathy
   matrbm
   a simplified version of 4
   http://code.google.com/p/matrbm/
   Matlab

   - restricted boltzmann machines
   - deep belief networks (of stacked RBMs)

5 Introduction to convolutional neural networks

Following up on my visit to the LISA lab, I prepared an introduction to convolutional neural networks, focusing on the motivation for their use, the main differences from traditional networks, the concepts of shared weights and multiple feature maps, down-sampling layers, and a description of the LeNet-5 model. This introduction was presented in the project meeting of February 22, 1013 and is included in Appendix B of this report.

6 Notes on UCI biomedical datasets

In order to form an idea of the type of biomedical data that we could use in our experiments while relying only on CPU power (as opposed to using GPUs), I searched the Machine Learning Repository maintained by the University of California - Irvine (UCI) [4] for data sets that fulfilled a number of characteristics, namely: being life sciences related; being frequently and recently used; being appropriate for classification learning; and being neither too small (less than 1000 examples) nor too big (say below 10000 examples).

A spreadsheet gathering the collected information on 62 available life sciences data sets is included in Appendix C. The data sets are ordered by decreasing number of papers citing them. This number of citations, as well as the range of

[4] See http://archive.ics.uci.edu/ml/index.html.

dates of those citations, provides a measure of how popular and recently used each data set is.

Taking into account the characteristics we desired, the five data sets highlighted in yellow seemed to be the most interesting ones, namely: Mushroom data; Splice-junction gene sequence data; Abalone data; Thyroid disease data; and Yeast data. These findings were discussed in the project meeting of March 15, 2013. We have subsequently used two of these data sets in our experiments with auto-encoders.

7 Notes on relevant conferences

I gathered in the table shown below information on conferences that are potentially interesting for publication of our work. Conferences are categorised into machine learning (ML), computer vision (CV), and biomedical engineering (BME). This list was discussed in the project meeting of April 19, 2013, but it should be kept up to date.

Deadline	Conference			ML	CV	BME	Notes
2012-11-10	ISBI	2013				x	
2012-11-15	AISTATS	2013		x			(1)
2012-11-15	CVPR	2013		x	x		IEEE conference.
2012-12-07	ESANN	2013		x			
2013-01-15	ICLR	2013		x			(1)
2013-02-04	EMBC	2013				x	
2012-10-01 2012-12-15 2013-02-15	ICML	2013		23 - x			(2)
2013-03-01	IJCNN	2013		1 - x			
2013-04-07	ICANN	2013		6 - x			
2013-04-12	ICCV	2013			x		
2013-04-19	GCPR	2013	Germany	8 - x			
2013-04-22	ECML/PKDD	2013	Czech Republic	7 - x			(3)
2013-04-24	BMVC	2013	UK		x		
2013-04-30	PReMI	2013	India	2 - x			
2013-05-11	ALT	2013	Singapore	x			
2013-05-13	UKCI	2013	UK	x			
2013-05-16	ICNC	2013	China	4 - x			IEEE conference.
2013-05-24	IDEAL	2013	China	x			
2013-05-31	AI*IA	2013	Italy	x			
2013-05-31	NIPS	2013	USA	17 - x			
2013-06-01	ICETET	2013	India	7 - x			
2013-06-15	CIARP	2013	Cuba	4 - x			
2013-06-15	ICONIP	2013	South Korea	5 - x			Bengio as plenary speaker.
2013-06-15	PRIA	2013	Russia	x	x		
2013-09-25	ICPRAM	2014	France	x			
2013-12-20	ICPR	2014	Sweden	11 - x			

Notes:

Numbers next to x's are Field Ratings from Microsoft Academic Search

(1) AISTATS may not be very relevant, because its former Learning Workshop now became a separate conference, ICLR.

(2) ICML 2013 had three submission / reviewing cycles. Next year there may be even more. The idea seems to be to make the conference work also a journal.

(3) ECML/PKDD 2013 accepted submissions also for a new "journal track": papers could be submitted to one of two journals and, if accepted, would receive a presentation slot at the conference.

A Slides: Introduction to deep learning

Introduction to deep architectures and planning of work directions

Telmo Amaral

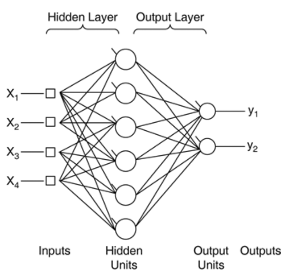
12th October 2012

Outline

1. Shallow vs. deep architectures
2. Motivation for deep architectures
   - Efficiency
   - Inspiration from nature
3. Limitations and recent breakthrough
4. Work directions

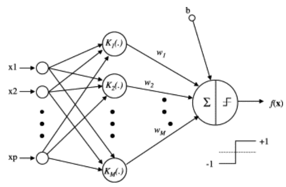
Depth of a flow graph

- Learning architectures can be represented as flow graphs.
- Depth: length of longest path from an input to an output.
- E.g. multi-layer perceptrons (MLPs) have depth 2.



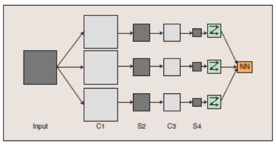
Another shallow example

- Support vector machines (SVMs) have depth 2 as well.

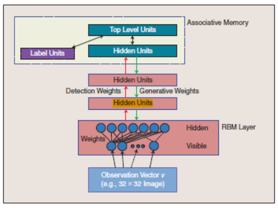


Two deeper architectures

- A convolutional neural network (CNN):

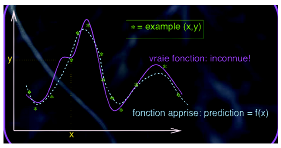


- A deep belief network (DBN):



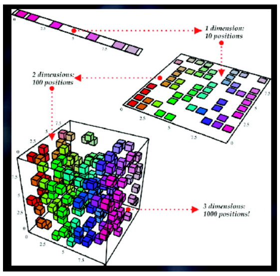
Local generalisation

- Principle exploited by majority of learning algorithms.
- Locally capture the variations: if two inputs are close, the corresponding outputs should also be close.
- But there should be at least as many training examples as ups and downs in the target function.



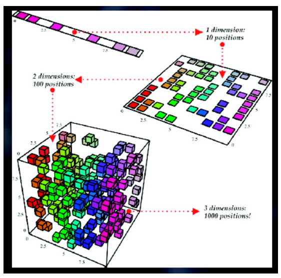
Local generalisation

- Curse of dimensionality: number of variations of interest in the target function can easily be exponential to the number of input dimensions.
- An alternative is needed...



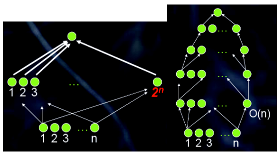
Non-local generalisation – distributed representations

- Trivial example: number 9 represented as
  - 0000000001000000 vs.
  - 1
    0
    0
    1
- Potential to capture exponentially more variations for the same number of free parameters – better generalisation.



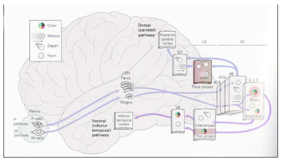
Non-local generalisation – distributed representations

- Some families of functions with n inputs can be compactly represented with $O(n)$ nodes when depth is d, but require $O(2^n)$ nodes when depth is $d-1$.
- The existence of a compact, deep representation suggests an underlying structure in the function; a shallow, unstructured representation won't be able to generalise well.



The brain has a deep architecture

- E.g. in the visual cortex signals flow through a hierarchy of areas.
- Each area represents information at a higher level of abstraction.
- Representations are *sparse*, but still exponentially efficient: 1% of simultaneously active neurons is a very large number, and inputs are represented by the activation of features that are *not mutually exclusive*.

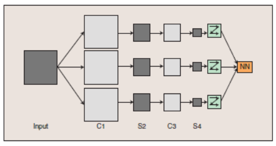


Cognitive processes seem deep

- Cognitive processes involve the hierarchical organisation of ideas.
- Humans first learn simpler concepts, then compose them into more abstract ones.
- Engineers typically solve complex problems by breaking them up into simpler ones.

Problems before 2006

- Deep NNs tended to yield worse training and test errors than shallow NNs (up to 2 hidden layers).
- Exception: CNNs, though tailored for very specific applications.

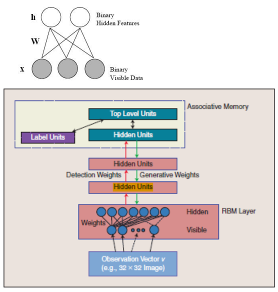


Key principles that constituted a breakthrough

- Pre-training of layers through unsupervised learning: representation learned at each layer used as input to the next layer, allowing the extraction of gradually higher abstractions.
- Fine-tuning of all layers through supervised learning (including additional top layers that produce predictions).

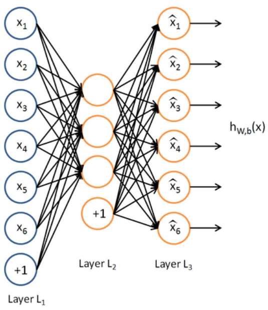
Seminal papers

- Hinton, Osindero, and Teh, *A fast learning algorithm for deep belief nets*, 2006.
- Pre-training achieved by treating layers as restricted Boltzmann machines (RBMs).



Seminal papers

- Bengio, Lamblin, Popovici, and Larochelle, *Greedy layer-wise training of deep networks*, 2007.
- RBMs vs. auto-encoders.
- Ranzato, Poultney, Chopra, and LeCun, *Efficient learning of sparse representations with an energy-based model*, 2007.
- Sparse auto-encoders within a convolutional architecture.



Performance of deep NNs

- Better than shallow NNs in
  - vision tasks; and
  - natural language processing (NLP) tasks.
- Better than SVMs in vision tasks.
- Allow dataset sizes untreatable by SVMs in NLP tasks.

Some available software

- Rasmus Berg Palm's DeepLearnToolbox, a Matlab deep learning toolbox, including DBNs; stacked and convolutional auto-encoders; CNNs and vanilla NNs.
- Hugo Larochelle's MLPython, a Python machine learning library, which can be used for deep learning research, featuring RBMs and auto-encoders.
- Ruslan Salakhutdinov's and Geoff Hinton's Matlab code for training a deep auto-encoder made of stacked RBMs. Andrej Karpathy's matrbm, simpler Matlab code for the same purpose.
- Ruslan Salakhutdinov's Matlab code for learning Deep Boltzmann Machines (DBMs, an alternative to DBNs). Hugo Larochelle's Matlab code for efficient learning of DBMs (apparently a continuation of the same work).

Shallow vs. deep architectures
Motivation for deep architectures
Limitations and recent breakthrough
Work directions

Main directions of work

- Use existing toolboxes such as MLPython and DeepLearnToolbox to test DNNs with different types of building blocks (e.g. RBMs and auto-associators) and the more traditional risk functionals (e.g. MSE and cross-entropy).
- Develop our own implementation of DNNs, in order to experiment with risk functionals based on more sophisticated principles (e.g. error density, MEE, Z-EDM, and EXP).

Telmo Amaral Deep architectures and work directions

B Slides: Introduction to convolutional neural networks

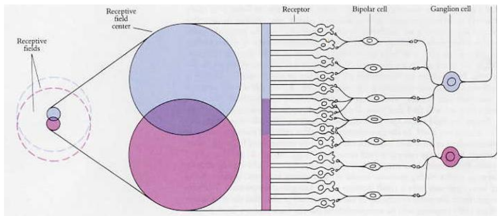
Introduction to convolutional neural networks (CNNs)

Telmo Amaral

22nd February, 2013

Motivation

- CNNs were the only exception to the difficulty in training deep neural networks before the use of unsupervised pre-training.
- CNNs are variants of MLPs inspired from biology.
- Cells within the visual cortex are sensitive to small sub-regions of the visual field, called receptive fields.
- Receptive fields are tiled to cover the whole visual field.

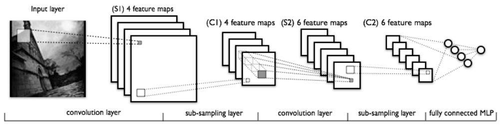


Motivation

- Many neurally inspired models emulate the behaviour of the visual system, such as:
- NeoCognitron. Fukushima, K. *Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position*. Biological Cybernetics, 36, 193–202. (1980).
- LeNet-5. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278–2324. (1998).
- HMAX. Serre, T., Wolf, L., Bileschi, S., and Riesenhuber, M. *Robust object recognition with cortex-like mechanisms*. IEEE Trans. Pattern Anal. Mach. Intell., 29(3), 411–426. (2007).

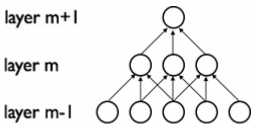
Main differences from traditional MLPs

- Local connectivity
- Shared weights
- Multiple feature maps
- Down-sampling layers

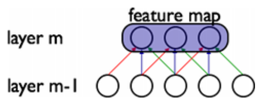


Local / sparse connectivity pattern

- Each hidden unit is connected to a local subset of units on the layer below.
- So, spatially local filters are learnt.
- Stacking layers leads to (non-linear) filters that are increasingly global in relation to the bottom layer.

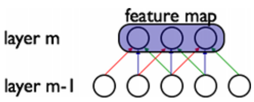


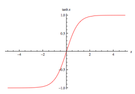
Shared weights



- Certain weights are constrained to be identical: the same filter is replicated across each layer of weights.
- A layer of replicated filters determines the values of a layer of hidden units, to form a feature map.

Shared weights





- A feature map h is obtained by convolving filter W with image x on the layer below (x can itself be a feature map), then adding a scalar bias b, and applying a non-linearity such as tanh:

  $h = \tanh((W * x) + b)$
- Recalling 1D an 2D convolution:

  $o[n] = f[n] * g[n] = \sum_{u=-\infty}^{\infty} f[u]g[u-n] = \sum_{u=-\infty}^{\infty} f[n-u]g[u]$

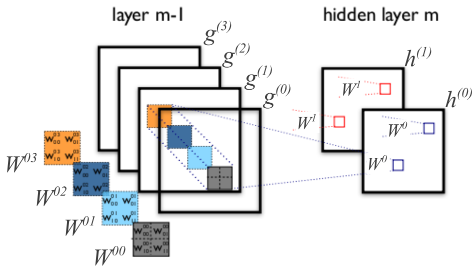
  $o[m,n] = f[m,n] * g[m,n] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v]g[u-m, v-n]$

Shared weights

- Why interesting?
- Invariance to translation: filter replication allows for features to be detected regardless of their position in the visual field.
- Weight sharing greatly reduces the number of free parameters to learn.
- Gradient descent can still be used to learn shared parameters, with a small adaptation:

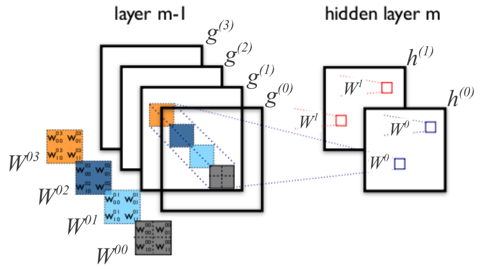
1. compute partial derivatives of loss function with respect to each *connection*, as in a conventional MLP with no sharing;
2. add partial derivatives of connections that share each weight, to form the derivative with respect to that weight.

Multiple feature maps



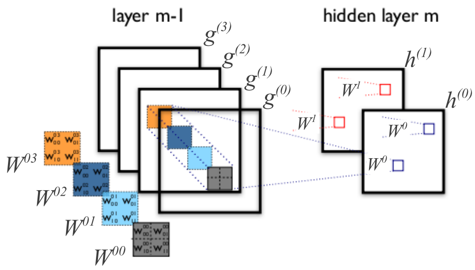
- For a richer representation, each hidden layer is composed of *a set* of feature maps, $\{h^{(k)}, k = 0..K\}$.

Multiple feature maps



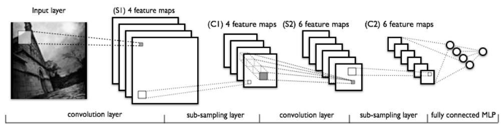
- W_{ij}^{kl} is the weight connecting any pixel on feature map k of a given layer to pixel ij (in filter coordinates) on feature map l of the layer below.

Multiple feature maps



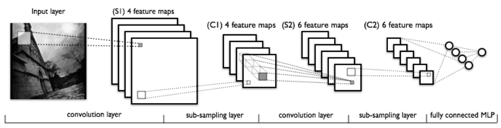
- If a given layer has K feature maps, the layer below has L feature maps, and all filters cover $I \times J$ pixels, then there are $K \times L \times I \times J$ different weights between the two layers.

Down-sampling layers



- Max-pooling: input image is partitioned into a set of non-overlapping rectangles and, for each rectangle, the maximum value is output.
- Reduces computational complexity for upper layers.
- Provides a form of invariance to small translations.

A full LeNet-5 model



- Lower layers alternate between convolution and down-sampling.
- Upper layers are fully connected and equivalent to a traditional MLP.
- (With other types of network it could be difficult to learn a reduced representation starting from 10k or 20k pixels, but with the combination of convolution down-sampling that type of learning becomes easier.)

(Image sources)

- http://deeplearning.net/tutorial/lenet.html
- http://hubel.med.harvard.edu/book/b10.htm
- http://mathworld.wolfram.com/HyperbolicTangent.html

C Spreadsheet: Notes on UCI biomedical data sets

Name	Data Types	Default Task	Attribute Types	# Instanc	# Attribu	Year	# citations		between	and	reported classification accuracies
Iris	Multivariate	Classification	Real	150	4	1988	55	+	1989	2005	
Breast Cancer	Multivariate	Classification	Categorical	286	9	1988	52	+	1995	2005	
Heart Disease	Multivariate	Classification	Categorical, Integer, Real	303	75	1988	36	+	1989	2004	
Soybean (Large)	Multivariate	Classification	Categorical	307	35	1988	30	+	1993	2004	
Soybean (Small)	Multivariate	Classification	Categorical	47	35	1987	30	+	1993	2004	
Breast Cancer Wisconsin (Diagnostic)	Multivariate	Classification	Real	569	32	1995	25	+	1995	2004	
Breast Cancer Wisconsin (Original)	Multivariate	Classification	Integer	699	10	1992	25	+	1995	2004	
Breast Cancer Wisconsin (Prognostic)	Multivariate	Classification, Regression	Real	198	34	1995	25	+	1995	2004	
Mushroom	Multivariate	Classification	Categorical	8124	22	1987	25	+	1995	2005	99.89% [Visakh, 2012]; 94.66% [Dai, 2012]
Molecular Biology (Promoter Gene Sequ	Sequential, Domain-Theory	Classification	Categorical	106	58	1990	22	+	1995	2005	
Molecular Biology (Splice-junction Gene	Sequential, Domain-Theory	Classification	Categorical	3190	61	1992	22	+	1993	2003	
Abalone	Multivariate	Classification	Categorical, Integer, Real	4177	8	1995	18	+	1997	2004	58.00% [Fouad, 2013]; 51.38% [Fathi, 2013]
Hepatitis	Multivariate	Classification	Categorical, Integer, Real	155	19	1988	17	+	1989	2004	
Liver Disorders	Multivariate		Categorical, Integer, Real	345	7	1990	16	+	1989	2004	
Pima Indians Diabetes	Multivariate	Classification	Integer, Real	768	8	1990	16	+	1995	2004	
Thyroid Disease	Multivariate, Domain-Theory	Classification	Categorical, Real	7200	21	1987	15	+	1994	2005	99.08% [Kabir, 2013] 97.50% [Dai, 2012]
Zoo	Multivariate	Classification	Categorical, Integer	101	17	1990	12	+	1995	2004	
Audiology (Original)	Multivariate	Classification	Categorical	226		1987	11	+	1995	2004	
Audiology (Standardized)	Multivariate	Classification	Categorical	226	69	1992	11	+	1995	2004	
Lymphography	Multivariate	Classification	Categorical	148	18	1988	10	+	1995	2004	
Ecoli	Multivariate	Classification	Real	336	8	1996	9	+	1997	2004	
Horse Colic	Multivariate	Classification	Categorical, Integer, Real	368	27	1989	9	+	1989	2003	
Yeast	Multivariate	Classification	Real	1484	8	1996	8	+	1997	2004	
Primary Tumor	Multivariate	Classification	Categorical	339	17	1988	6	+	1995	2002	
Molecular Biology (Protein Secondary St	Sequential	Classification	Categorical	128			5	+	1996	2003	
Arrhythmia	Multivariate	Classification	Categorical, Integer, Real	452	279	1998	3		2002	2005	
SPECT Heart	Multivariate	Classification	Categorical	267	22	2001	3	+	2002	2004	
Dermatology	Multivariate	Classification	Categorical, Integer	366	33	1998	2	+	2002	2004	
Echocardiogram	Multivariate	Classification	Categorical, Integer, Real	132	12	1989	2	+	1989	2002	
Haberman's Survival	Multivariate	Classification	Integer	306	3	1999	2	+	2002	2003	
Lung Cancer	Multivariate	Classification	Integer	32	56	1992	2	+	1998	2003	
Post-Operative Patient	Multivariate	Classification	Categorical, Integer	90	8	1993	2	+	1999	2000	
Statlog (Heart)	Multivariate	Classification	Categorical, Real	270	13		2	+	1997	2004	
Contraceptive Method Choice	Multivariate	Classification	Categorical, Integer	1473	9	1997	1	+	2002		
SPECTF Heart	Multivariate	Classification	Integer	267	44	2001	1		2004		
Acute Inflammations	Multivariate	Classification	Categorical, Integer	120	6	2009	0				
Arcene	Multivariate	Classification	Real	900	10000	2008	0				
Breast Tissue	Multivariate	Classification	Real	106	10	2010	0				
Cardiotocography	Multivariate	Classification	Real	2126	23	2010	0				
Demospongiae	Multivariate	Classification	Integer	503		2010	0				
Dorothea	Multivariate	Classification	Integer	1950	100000	2008	0				
ILPD (Indian Liver Patient Dataset)	Multivariate	Classification	Integer, Real	583	10	2012	0				
KEGG Metabolic Reaction Network (Und	Multivariate, Univariate, Text	Classification, Regression, Clustering	Integer, Real	65554	29	2011	0				
KEGG Metabolic Relation Network (Direc	Multivariate, Univariate, Text	Classification, Regression, Clustering	Integer, Real	53414	24	2011	0				
Mammographic Mass	Multivariate	Classification	Integer	961	6	2007	0				
One-hundred plant species leaves data		Classification	Real	1600	64	2012	0				
p53 Mutants	Multivariate	Classification	Real	16772	5409	2010	0				
Parkinsons	Multivariate	Classification	Real	197	23	2008	0				
Parkinsons Telemonitoring	Multivariate	Regression	Integer, Real	5875	26	2009	0				
Plants	Multivariate	Clustering	Categorical	22632	70	2008	0				
PubChem Bioassay Data	Multivariate	Classification	Integer, Real			2011	0				
Quadruped Mammals	Multivariate, Data-Generator	Classification	Real		72	1992	0				
seeds	Multivariate	Classification, Clustering	Real	210	7	2012	0				
Sponge	Multivariate	Clustering	Categorical, Integer	76	45		0				
Abscisic Acid Signaling Network	Multivariate	Causal-Discovery	Integer	300	43	2008					
Covertype	Multivariate	Classification	Categorical, Integer	581012	54	1998					
Diabetes	Multivariate, Time-Series		Categorical, Integer		20						
E. Coli Genes	Relational					2001					
EEG Database	Multivariate, Time-Series		Categorical, Integer, Real	122	4	1999					
ICU	Multivariate, Time-Series		Real								
Localization Data for Person Activity	Univariate, Sequential, Time-Series	Classification	Real	164860	8	2010					
M. Tuberculosis Genes	Relational					2001					

References

[1] T. Amaral. Experiments with a restricted Boltzmann machine. Technical Report 1/2012, Instituto de Engenharia Biomédica / NNIG, December 2012. 3, 5

[2] T. Amaral. Using different cost functions when pre-training stacked autoencoders. Technical Report 1/2013, Instituto de Engenharia Biomédica / NNIG, April 2013. 3, 4

[3] I. Arel, D. Rose, and T. Karnowski. Deep machine learning - a new frontier in artificial intelligence research. *IEEE Computational Intelligence Magazine*, 5(4):13–18, 2010. 3

[4] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. 3, 5

[5] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *arXiv preprint*, arXiv:1206.5538, 2012. 5

[6] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 3248–3256, 2012. 5

[7] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 6

[8] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 3

[9] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012. 4, 5

[10] R. Salakhutdinov and H. Larochelle. Efficient learning of deep Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, 2010. 7