Neural Networks and Learning Systems
TBMI 26, 2017

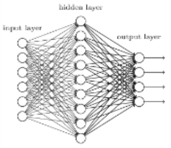
Lecture 5

Deep Learning

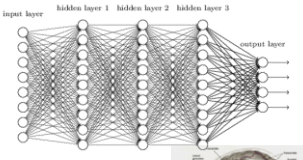
Ola Friman
ola.friman@liu.se

What is Deep Learning?

Not deep…



Deep…



Essentially another name for a large neural network

2

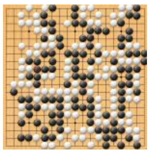
Major applications

- Classifying audio signals
  - Speech recognition
  - Voice recognition
  - Real-time translation
  - Database search
- Classifying image data
  - Image tagging
  - Finding objects in images
  - Autonomous vehicles
  - Photosearch
  - Video recommendations
- Intensively used & researched by data-oriented companies such as Google, Apple, Microsoft, IBM, Facebook, Baidu, Nvidia

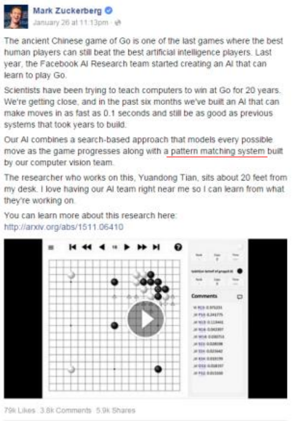
3

From the news

- Volvo buys Nvidia deep learning computer for its autonomous vehicle progam
  - http://www.dn.se/ekonomi/volvo-koper-nvidias-superdator-for-sjalvkorande-bilar/
  - http://nvidianews.nvidia.com/news/nvidia-s-deep-learning-car-computer-selected-by-volvo-on-journey-toward-a-crash-free-future
- Google vs. Facebook, Deep Neural networks for playing Go
- http://googleresearch.blogspot.com.au/2016/01/alphago-mastering-ancient-game-of-go.html
- https://www.facebook.com/zuck/posts/10102619979696481?fref=nf



4


Mark Zuckerberg
January 26 at 11:13pm

The ancient Chinese game of Go is one of the last games where the best human players can still beat the best artificial intelligence players. Last year, the Facebook AI Research team started creating an AI that can learn to play Go.

Scientists have been trying to teach computers to win at Go for 20 years. We're getting close, and in the past six months we've built an AI that can make moves in as fast as 0.1 seconds and still be as good as previous systems that took years to build.

Our AI combines a search-based approach that models every possible move as the game progresses along with a pattern matching system built by our computer vision team.

The researcher who works on this, Yuandong Tian, sits about 20 feet from my desk. I love having our AI team right near me so I can learn from what they're working on.

You can learn more about this research here:
http://arxiv.org/abs/1511.06410

79k Likes 3.8k Comments 5.9k Shares

5

History
(roughly)

- 1960's (and before): Linear methods, perceptron
- 1980's: Nonlinear breakthroughs, neural networks
- 1990's-now:
  - Kernel methods, SVM
  - Ensemble learning, boosting, bagging
- 2010 - now
  - Deep neural networks
    - Massive amounts of data available
    - Computational power (GPU)
    - New optimization tricks

6

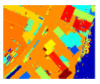
ML history : Feature learning

- The machine learning field has been developed through
  - Increasing amount of computational power
  - Increasing possibilities to generate & collect data -> More training data

ML development

Feature handcrafting

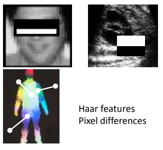
SVM &
Neural Networks



$x = \begin{bmatrix} \text{Mean intensity} \\ \text{Std intensity} \\ \text{Border regularity} \\ \text{Compactness} \end{bmatrix}$

Low-level features

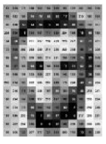
Bagging, Decision Forests
Boosted Decision Trees



Haar features
Pixel differences

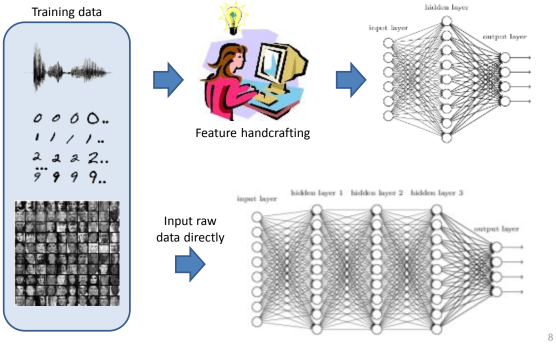
Raw sensor data as features

Deep Neural Network



7

Bigger is better?: Advantages

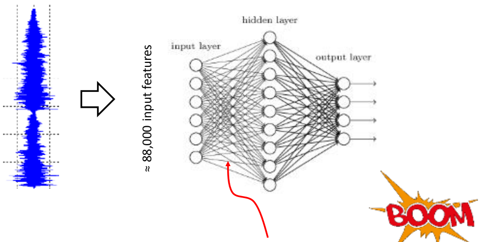


8

Bigger is better?: Problems

2 s @ 44.1 kHz

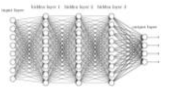
500 hidden nodes (say)



88,000 x 500 = 44 million weights!

9

Bigger is better?: Problems



Lots of parameters (weights)

Overfitting & complicated optimization ”landscape”

- Special training/optimization techniques
- Lots (LOTS!) of training data needed

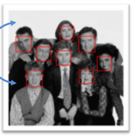
- Long training times (days, weeks, months)
- Getting the training data

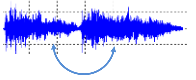
10

Local connectivity

Utilize that patterns in real signals/images are local

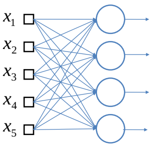
Two pixels far apart
have little to do with each other



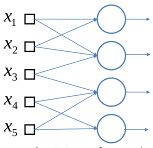


Two sound samples far apart
have little to do with each other

Fully connected



Locally connected



Don’t mix inputs far apart!

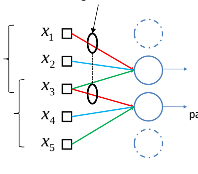
11

Weight sharing

Extract the same features regardless of position in signal/image!

Local connections + weight sharing

Constrain weights to be the same!

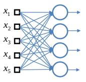


Presence of a pattern
at one position

Presence of the same
pattern at another position

Learn a convolution kernel!

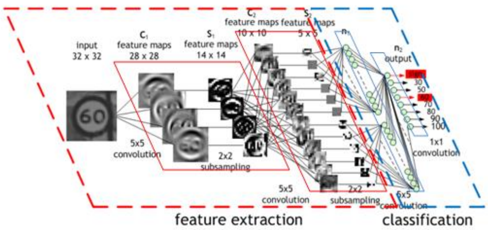
Dramatic reduction of the
number of weights!



12

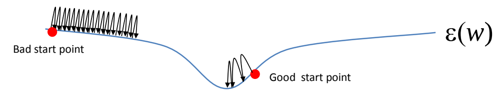
Convolutional Neural Network

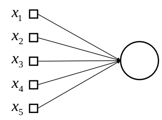
- Local connections + weight sharing architecture
- Initial layers learn convolution kernels (5x5 in example below)
- A ”3D” network structure allows us to learn several different kernels



13

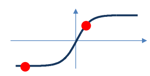
Training tricks : Good initialization (i)





- $s = \mathbf{w}^T\mathbf{x}$ should not saturate the activation function
- Normalize the input features:

$$\hat{x}_i = \frac{x_i - \bar{x}_i}{std(x_i)}$$ Zero mean and equal variation



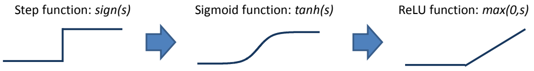
- Init the weights randomly from a uniform dist.:

$$w_i \sim \left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$$ # input features

14

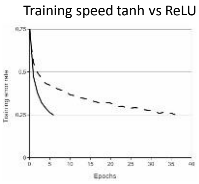
Rectified Linear Unit (ReLU) activation function

Step function: *sign(s)* Sigmoid function: *tanh(s)* ReLU function: *max(0,s)*



No gradient information Saturates = vanishing gradients

Training speed tanh vs ReLU



Krizhevsky et al. ImageNet Classification with Deep Convolutional Neural Networks, 2012

<u>Pros and cons of ReLU functions:</u>

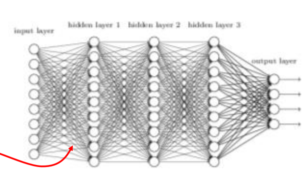
+ No vanishing gradient = faster training
+ Fast to compute

- Knockout problem: A neuron can be driven to a state where it never activates for any input = it is dead and useless

15

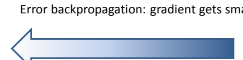
Training tricks: Good initialization (ii)

<u>Vanishing gradient problem</u> when training many layers



Early layers get very small weight updates

Error backpropagation: gradient gets smaller and smaller



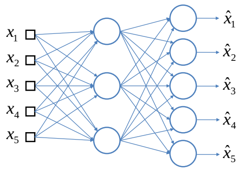
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\partial \varepsilon}{\partial \mathbf{w}}$$

16

Training tricks: Good initialization (ii)

Pre-train to initialize weights in hidden layers

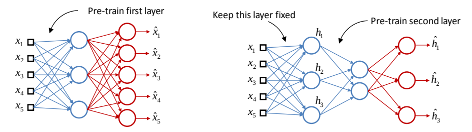
Autoencoder – learn to reproduce the input



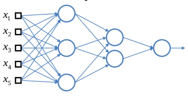
Unsupervised learning – no labels needed!

Pre-training to init hidden layers

- Pre-train hidden layers one at the time

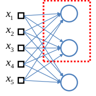


- Final training of all layers with backprop



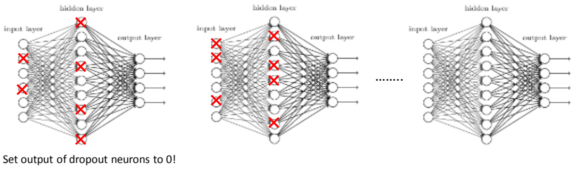
H. Larochelle et al. "Exploring Strategies for Training Deep Neural Networks", Journal of Machine Learning Research (2009)

Training tricks: Dropout



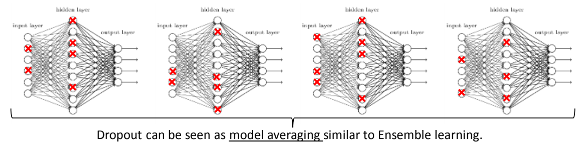
- How do we prevent weights for two different neurons to converge to similar values?
- Each neuron should focus on a different aspect of the input data, so-called 'co-adaptation' should be avoided.
- Dropout: remove neurons randomly during training. Forces neurons to adapt to uncertain input, prevents co-adaption.

Epoch 1 Epoch 2 Use all neurons when ready!

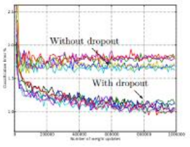


Set output of dropout neurons to 0!

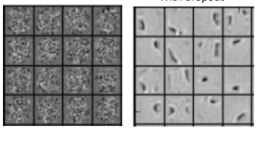
Dropout vs. Ensemble learning



Dropout can be seen as model averaging similar to Ensemble learning.
Avoids overfitting!



Without dropout With dropout



Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", 2014

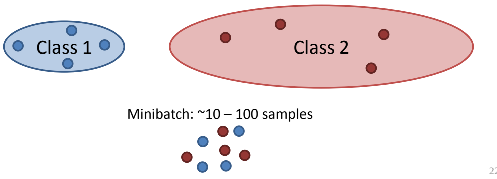
Training tricks: Balanced training data

$$\varepsilon(\mathbf{w})=\sum_{i=1}^{N}\left(f(\mathbf{x}_i)-y_i\right)^2=\underbrace{\sum_{j=1}^{N_1}\left(f(\mathbf{x}_j)-y_j\right)^2}_{\text{Class 1 error}}+\underbrace{\sum_{i=1}^{N_2}\left(f(\mathbf{x}_i)-y_i\right)^2}_{\text{Class 2 error}}$$

- If $N_1 >> N_2$, the classifier will focus on classifying class 1 correctly.
- Remedies:
  - Duplicate the samples of class 2 so that $N_1 \approx N_2$.
  - Reduce the number of samples from class 1 so that $N_1 \approx N_2$ (possibly use different subsets for each optimzation step).
  - Introduce different weights for class 1 and class 2 in the error function that balances their respective influence on the cost function.

Training tricks: Minibatches

- Online training (stochastic gradient descent): One epoch consists of only one training data sample
- Batch training: One epoch uses all samples. Costly for LARGE data sets (memory, computation)
- Minibatches: Use a balanced random subset of the training data for each epoch.



Training tricks – Summary

- Normalize input
- ReLU activation function
- Pre-training of hidden layers
- Dropout
- Balance the training data
- Minibatches

Normalize input, ReLU activation function, Pre-training of hidden layers: Combat vanishing gradients

Dropout, Balance the training data: Combat overtraining

Minibatches: Training efficiency and practical necessity

Hardware-accelerated training

- 10 million training images, say 10 ms to feed an image through the network
  - 10^6*0.01 seconds ≈ 27 hours for 1 epoch
- Parallelization opportunities:
  - In batch training, each training sample can be propagated in parallel.
  - The output from neurons in each layer can be computed in parallel (compactly written as matrix multiplications + activation function!)

Training on GPU clusters

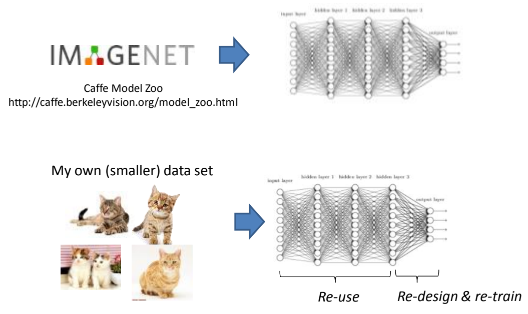
Software for Deep Learning

- Caffe
  - http://caffe.berkeleyvision.org/
- Theano
  - http://deeplearning.net/software/theano/
- Torch
  - http://torch.ch/
- TensorFlow (from Google Brain Team)
  - https://www.tensorflow.org/
- Computational Network Toolkit (Microsoft)
  - https://github.com/Microsoft/CNTK
- NVIDIA cuDNN
  - https://developer.nvidia.com/cudnn

25

Transfer learning

Re-use already trained deep networks!



Yosinki et al. "How transferable are features in deep neural networks?", 2014

26

Deep Neural networks: Disadvantages & critic

- Lots, lots, lots of training data
  - Costly to get the data
  - Long training times
- "Dark art" to train deep neural networks
  - Lots of architecture, optimization, data-preconditioning issues
- Lack of mathematical underpinnings
  - Still only gradient descent
  - Compare with Support Vector Machines theory
  - Empirical & heurisitc results – "this worked"

27

SUPERVISED LEARNING SUMMARY

28

Classifier summary

- **k-Nearest Neighbors**
  - Simple non-linear classifier for small data sets. Don't underestimate.
- **Support Vector Machines**
  - Overall favourite classifier world-wide?
  - Works well with little training data due to maximum-margin cost function.
  - Somewhat difficult to set meta-parameters *C* and kernel (see lecture 7).
- **Neural Network**
  - Allround but tricky to train.
  - Deep neural network currently the hottest classifier technique. Currently best results on important applications. Requires lots of data.
- **Decision trees**
  - Easy-to-use non-linear classifier.
  - Successful together with Bagging (Random Forest) and Boosting for real-time classification using cascaded classifers.
- **Fisher Linear Discriminant (will be introduced in Lecture 6)**
  - Easy-to-train and intuitive linear classifier.
  - Useful as benchmark to compare non-linear classifiers against.

29

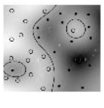
Better data beats better algorithms

- Feature engineering
  - Clever features facilitate learning greatly.
  - Trend towards lower-level features
    - Boosting
    - Deep Learning
- Training data must represent new data well
  - Overtraining/overfitting a fundamental problem.
- Lots of training data
  - Labeling/annotation requires manual work.
  - Experts may be required.

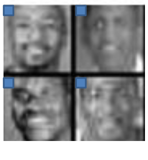
30

Generalization enemy 1 – Model overfitting

- Overfitting because model has too many parameters (in relation to the number of training data samples)



- Neural network with too many nodes.
- Decision tree which is too deep.

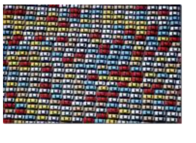


Memorization:
Face if pixel (1,1) has intensity value 189, 195, 196 or 201

31

Generalization enemy 2 – Insufficiently representative training data

vs.



Bertrand Russell's inductive turkey (Russel, 1912):

This turkey found that, on the first morning in the turkey farm, he was fed at 9.00 a.m. However, being a good inductivist, he did not jump to conclusions. Patiently, he waited many days, observing that he was fed every day at 9.00 a.m., whether the sun shone or it rained, in windy weather and in calm. Eventually, his careful list of observation statements led him to conclude that 'I am always fed at 9.00 a.m.' On Christmas Eve his inductive inference with true premises was shown to have led him to a patently false conclusion, for on that day at 9.00 a.m., instead of being fed, his throat was cut.

Overfitting to unrepresentative data:
Learning for the course exam by memorizing the answers to the previous year's exam

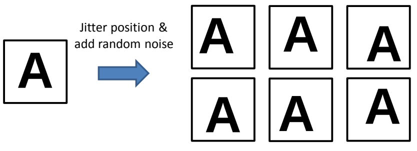
32

Reducing labeling effort: Multiply training data by perturbation

(a.k.a data augmentation, bootstrapping)

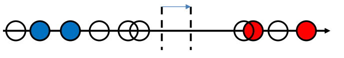
OCR

Master's thesis by O. Fridolfsson



33

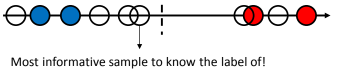
Reducing labeling effort: Semi-supervised learning



- Use unlabeled information for training too!
- Can go wrong!
- Evidence that humans use semi-supervised learning:
  - Zhu et al. (2007) "Humans Perform Semi-Supervised Classification Too"
  - Gibson et al. (2013) "Human semi-supervised learning"

34

Reducing labeling effort: Active learning



Most informative sample to know the label of!

1. Start by training on a small labeled subset.
2. Label the most informative unlabeled samples as queried by the learning machine.
3. Retrain and goto step 2.

35