Outline

• Context

• Three periods for the development of neural networks
  • 1960 - Early Days – Birth of Neural Networks
  • 1990 - Non Linear Machines – Statistical Learning Theory
  • 2010 - Deep Learning – Large Size Industrial Applications
Context

• Big Data - Industrial eco-system
  • Launched in the early 2000s by internet companies for search problems (Google, Yahoo, ...)

• Machine Learning - IT actors
  • GAFA (Google, Apple, Facebook, Amazon), BAT (Baidu, Tencent, Alibaba), ..., Startups
  • Research
    • Large groups at Google Brain, Google Deep Mind, Facebook FAIR, Baidu AI lab, Baidu Institute of Deep Learning, etc
      • Lot of them focus on Deep Learning
    • Strong influence on research directions
    • Establish « standards »
      • e.g. Google released Tensor Flow ML library
    • Research in the domain requires access to sophisticated libraries and to computing power e.g. GPU
Context of the presentation: supervised learning paradigm

• Training set
  • couples (input – target) \((x^1, d^1), \ldots, (x^N, d^N)\)

• Objective: learn to associate inputs to targets

• Typical problems: classification, regression, ranking

![](image)
Context of the presentation: inductive learning and ERM framework

- Hyp: data are generated i.i.d. according to a distribution $p(x, y)$
  - Let us define
    - Learning machine: $F \in F$
      - Typically a parametric function
    - Loss for $(x, y)$ and $F$: $c(F(x), y)$
    - Risk:
      - $R = E_{X,Y}[c(F(x), y)] = \int_{X,Y} c(F(x), y) dp(x, y)$
      - Optimal solution: $F^* = \arg\min_F(R)$

- **Empirical Risk Minimization**
  - Empirical risk defined on a finite dataset $D = \{(x^i, y^i)\}_{i=1..N}$:
    - $R_{Emp} = \frac{1}{N} \sum_{i=1}^{N} c(F(x^i), y^i)$
  - **Inductive learning and ERM**
    - Choose $\hat{F} = \arg\min_F R_{Emp}$
      - Train a Learning machine on a finite sample (training set)
      - Evaluate its performance on another sample (test set)
    - ERM has been the main paradigm used up to the 80es
    - ERM is not sufficient for learning (overfitting/ generalization)
1960 – Early days - Birth of Neural Networks
Neural Networks inspired Machine Learning

- Artificial Network Networks are an important paradigm in Statistical Machine learning
- Human brain is used as a source of inspiration and as a metaphor for developing ANN
  - Human brain is a dense network $10^{11}$ of simple computing units, the neurons. Each neuron is connected – in mean- to $10^4$ neurons.
  - Brain as a computation model
    - Distributed computations by simple processing units
    - Information and control are distributed
    - Learning is performed by observing/ analyzing huge quantities of data and also by trials and errors
Formal Model of the Neuron
McCulloch – Pitts 1943

A synchronous assembly of neurons is capable of universal computations (aka equivalent to a Turing machine)

\[
y = \begin{cases} 
1 & \text{if } \sum_i^N w_i x_i - w_0 > 0 \\
0 & \text{otherwise}
\end{cases}
\]
• Learning
  • Hebb rule (1949) (Neuropsychologist D. Hebb) - rephrased
    • If two neurons on either side of a synapse are activated simultaneously (synchronously), the strength of the synapse is increased
    • Most famous rule of self organization

• Neural Networks as computation model
  • Analog processing via simple units
  • Distributed and parallel computing
  • Programming is replaced by learning from examples
Perceptron (1958 Rosenblatt)

- The decision cell is a threshold function
  \[ F(x) = \text{sgn}(\sum_{i=0}^{n} w_i x_i) \]
- This simple perceptron can perform 2 class classification
Perceptron Algorithm (2 classes)

Data

Labeled Dataset \( \{(x^i, y^i), i = 1..N, x \in \mathbb{R}^n, y \in \{-1,1\} \} \)

Output

classifier \( w \in \mathbb{R}^n \), decision \( F(x) = \text{sgn}(\sum_{i=0}^{n} w_i x_i) \)

Initialize \( w(0) \)

Repeat (t)

Choose an example \( (x(t), y(t)) \)

if \( y(t)w(t) \cdot x(t) \leq 0 \) then \( w(t + 1) = w(t) + \epsilon d(t)x(t) \)

Until convergence

- The learning rule is a stochastic gradient algorithm for minimizing the number of wrongly labeled points
• **Convergence theorem** (Novikof, 1962)
  Let \( D = \{(x^1, d^1), \ldots, (x^N, d^N)\} \) a data sample. If
  - \( R = \max_{1 \leq i \leq N} \|x^i\| \)
  - \( \sup_w \min_i d^i(w \cdot x^i) > \rho \)
  - The training sequence is presented a sufficient number of time

  The algorithm will converge after at most \( \left\lceil \frac{R^2}{\rho^2} \right\rceil \) corrections

• **Generalization bound** (Aizerman, 1964)
  If in addition we provide the following stopping rule:
  Perceptron stops if after correction number \( k \), the next \( m_k = \frac{1+2 \ln k - \ln \eta}{-\ln(1-\varepsilon)} \) data are correctly recognized

  Then
  - the perceptron will converge in at most
  \( l \leq \frac{1+4 \ln R/\rho - \ln \eta}{-\ln(1-\varepsilon)} \left\lceil \frac{R^2}{\rho^2} \right\rceil \) steps

  - with probability \( 1 - \eta \), test error is less than \( \varepsilon \)
    - >>>>>>>> Link between training and generalization performance
Adaline (Widrow - Hoff 1959)

- Context
  - Adaptive filtering, equalization, etc.
- «Least Mean Square» algorithm
  - Loss function: euclidean distance (target – computed output)
  - Algorithm: stochastic gradient (Robbins – Monro (1951))
- Workhorse algorithm of adaptive signal processing
  - Simple, robust
Adaline example

- **Adaptive** noise cancelling

![Adaptive noise cancelling concept](image)

*Fig. 1. The adaptive noise cancelling concept.*

![Multiple-reference noise canceller in fetal ECG experiment](image)

*Fig. 15. Multiple-reference noise canceller used in fetal ECG experiment.*
Summary

• Many of the main concepts of statistical Machine Learning are already present in the early days
  • Learning machine as alternative models of computations
  • Adaptive algorithms for optimizing loss functions
  • Applications
    • Pattern recognition
    • Signal processing
  • Performance guarantees assessed by generalization bounds
1990 - Non Linear Machines and Statistical Learning Theory
Multi-layer Perceptron (Hinton – Sejnowski 1986)

- Neurons arranged into layers
- Each neuron is a non linear unit

\[
\hat{y} = F_W(x) = f \circ (W_2 f \circ (W_1 x))
\]

Note: \( \circ \) is a pointwise operator \( f \circ (x_1, x_2) = (f(x_1), f(x_2)) \)
Training

• Similar to Perceptron or Adaline
  • Stochastic Gradient Descent - The algorithm is called Back-Propagation
• Loop over the training set
  • pick one example or a small batch of examples
• Forward pass
  • For each example compute the output
    • \( \hat{y} = F_w(x) = f \odot (W_2f \odot (W_1x)) \)
  • Compute the error
    • \( \delta = c(y, \hat{y}) \)
• Backward pass
  • Back propagate the errors starting from the last weight layer down to the first weight layer
    • \( w_{ij} = w_{ij} + \Delta w_{ij}, \text{ with } \Delta w_{ij} = -\frac{\partial c(y, \hat{y})}{\partial w_{ij}} \)

• Note: large literature on optimization for MLPs and other NNs
Properties

• Universal Approximation
  
  • e.g. Cybenko 89: Let $f$ be a continuous saturating function. The space of functions of the form $g(x) = \sum_{j=1}^{n} v_j f(w_j \cdot x)$ is dense in the space of continuous functions on the unit cube $C(I)$. i.e. $\forall h \in C(I)\exists \epsilon > 0, \exists g : |g(x) - h(x)| < \epsilon$ on $I$.

  • No « constructive » result
    
    • e.g. number of hidden neurons or hidden layers for a given problem.
Generalization and Model Selection

- Exemple regression (Bishop 06)

- ERM is not sufficient
  - The model complexity should be adjusted both to the task and to the information brought by the examples
  - Both the model parameters and the model capacity should be learned
  - Lots of practical method and of theory has been devoted to this problem
Generalization and Model Selection

• Practical solutions
  • Regularization
  • Ensemble models
    • Bagging, Boosting, Forests
  • Algebraic criteria
    • AIC, BIC, LOO, MDL, ...
  • Surrogate losses
  • ...

• Statistical learning theory
  • Vapnik ERM/ SRM
  • PAC (Probably Approximately Correct) framework
  • Rademacher complexity
  • .....
Practical capacity control: régularisation

- Hadamard: Ill posed problems, Tikhonov: Regularization
- Principle:
  - Control the variance of the solution by constraining functional $F$
  - Optimize $R_{\text{Emp}}^{\text{Reg}} = R_{\text{Emp}} + \lambda R_{\text{Reg}}(F)$
    - $R_{\text{Emp}}$: loss associated to the problem e.g. MSE, Entropy, ...
    - $R_{\text{Reg}}(F)$: constraints on the solution (e.g. weight distribution)
- Example: regularized least square
  \[
  R_{\text{Emp}}^{\text{Reg}} = \frac{1}{N} \sum_{i=1}^{N} (y^i - F(w, x^i))^2 + \frac{\lambda}{2} \sum_{j=1}^{n} |w_j|^q
  \]

**Figure 3.3** Contours of the regularization term in (3.29) for various values of the parameter $q$.

- $q = 2$ regularisation $L_2$, $q = 1$ regularisation $L_1$ also known as « Lasso »
Generalization and Model Selection
Vapnik Statistical Learning Theory

- One of the most influential work on statistical learning theory
- 4 questions for analyzing ERM
  - Consistency of ERM principle
    - Uniform convergence of $R_{\text{emp}}(\theta)$ to $R(\theta)$
  - Non asymptotic theory of the rate of convergence of the learning process
    - How fast the empirical risk converges to the actual risk in terms of training set size
    - Example for binary functions
  - Let $F$ a family of functions taking values in $\{-1,1\}$ with VC-dim $h$. $D$ a dataset of i.i.d. examples with $|D| > h$. Then for any $\delta > 0$, with probability at least $1 - \delta$ the following bound holds for all $f \in F$:
    \[
    R(f) \leq R_{\text{emp}}(f) + \sqrt{\frac{8\log 2eN}{h} + \frac{8\log 4}{h} N}
    \]
    With $N = |D|$ the size of the training set

- Control of the generalization of the learning process
  - Structural Risk minimization
  - Practical construction of learning algorithms

- Rq
  - Independent of the data distribution
  - not practical – e.g. for NNs, VC-Dim is unknown, bounds are too large
Radial Basis Function Networks

- Linear combination of gaussian kernels
- Kernel machines are another important family of learning machines
  - Developed 1995-2005, e.g. Support Vector Machines, Gaussian Processes
  - Mainly rely on convex optimization
  - More amenable to theoretical analysis – e.g. generalization bounds

\[ f(x) = \exp\left(-\frac{\|x - w_i\|^2}{\sigma_i^2}\right) \]

\[ F_k(x) = \sum_{i \text{ hidden unit}} w_{ki} \exp\left(-\frac{\|x - w_i\|^2}{\sigma_i^2}\right) \]
Recurrent networks

- **Recurrent Neural Networks**
  - They are dynamic, non-linear and continuous state models

- **Dynamics**
  - Time limited
    - Stop the dynamic after a # time steps
  - Unlimited
    - Wait for convergence: stable state or limit cycle

- Input and supervision are usually defined as sequences but may take different forms
Recurrent networks

- Usual forms correspond to state space models with local recurrent units

\[
\begin{align*}
\mathbf{x}_t & \rightarrow \mathbf{y}_t \\
\mathbf{h} & \rightarrow \mathbf{y}_t \\
\mathbf{c} & \rightarrow \mathbf{y}_t
\end{align*}
\]

- Approximation property (Lo 1993)
  - Any non linear dynamic system may be approximated by a RNN to any desired degree of accuracy, provided that the network has an adequate number of hidden neurons.

- Computational property (Siegelmann & Sontag, 1991)
  - All Turing machines may be simulated by fully connected recurrent networks built of neurons with sigmoidal activation functions.

- In practice, RNN did not make their way to applications in the 90es
  - They remained a curiosity until recently

- Limitations
  - Long term dependencies, gradient vanishing

State space equations

\[
\begin{align*}
c_{t+1} &= f(c_t, x_t) \\
y_t &= g(c_t, x_t)
\end{align*}
\]
Example: trajectory learning

(Pearlmutter, 1995, IEEE Trans. on Neural Networks – nice review paper on RNN)

- Globally recurrent net: 2 output units
Summary

• Non linear machines
• Fundations for modern statistical machine learning
• Fundations for statistical learning theory
• First real world applications
2010 Deep Learning
Basic idea of deep learning

• Learn high level/abstract representations from raw data
  • Key idea: stack layers of neurons to build deep architectures
  • Find a way to train them

Figure from Lee H. et al. 2011
Learnig High Level Representations in Videos –
Google (Le et al. 2012)

- **Objective**
  - Learn high level representations without teacher
    - 10 millions images 200x200 from YouTube videos
    - **Auto-encoder** $10^9$ connexions
  - « High level » detectors
    - Show test images to the network
      - E.g. faces
    - Look for neurons with maximum response
  - Some neurons respond to high level characteristics
    - Faces, cats, silhouettes

*Figure 3. Top: Top 48 stimuli of the best neuron from the test set. Bottom: The optimal stimulus according to numerical constraint optimization.*
Convolutional nets

- ConvNet architecture (Y. LeCun since 1988)
  - Deployed e.g. at Bell Labs in 1989-90 for zip code recognition
  - Character recognition
  - Convolution with learned filters: non linear embedding in high dimension
  - Pooling: average, max
• In Convnet
  • The first hidden layer consists in 64 different convolution kernels over the initial input, resulting in 64 different mapping of the input
  • The second hidden layer is a sub-sampling layer with 1 pooling transformation is applied to each matrix representation of the first hidden layer
  • etc
  • Last layer is a classification layer
Convolutional nets - visualization

• Hand writing recognition (Y. LeCun Bell labs 1989)
Convolutional nets (Krizhevsky et al. 2012)

- A landmark in object recognition - AlexNet
- Imagenet competition
  - Large Scale Visual Recognition Challenge
  - 1000 categories, 1.5 Million labeled training samples
  - Method: large convolutional net
  - 650K neurons, 630M synapses, 60M parameters
  - Trained with backprop on GPU
Convolutional nets

- **Imagenet 2012 classification challenge**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Error rate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U. Toronto</td>
<td>0.15315</td>
<td>Deep learning</td>
</tr>
<tr>
<td>2</td>
<td>U. Tokyo</td>
<td>0.26172</td>
<td>Hand-crafted features and learning models. Bottleneck.</td>
</tr>
<tr>
<td>3</td>
<td>U. Oxford</td>
<td>0.26979</td>
<td>Hand-crafted features and learning models. Bottleneck.</td>
</tr>
<tr>
<td>4</td>
<td>Xerox/INRIA</td>
<td>0.27058</td>
<td>Hand-crafted features and learning models. Bottleneck.</td>
</tr>
</tbody>
</table>

Object recognition over 1,000,000 images and 1,000 categories (2 GPU)

- **ImageNet 2013 – image classification challenge**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Error rate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NYU</td>
<td>0.11397</td>
<td>Deep learning</td>
</tr>
<tr>
<td>2</td>
<td>NUS</td>
<td>0.12535</td>
<td>Deep learning</td>
</tr>
<tr>
<td>3</td>
<td>Oxford</td>
<td>0.13555</td>
<td>Deep learning</td>
</tr>
</tbody>
</table>

MSRA, IBM, Adobe, NEC, Clarilai, Berkley, U. Tokyo, UCLA, ULIC, Toronto ... Top 20 groups all used deep learning

- **ImageNet 2013 – object detection challenge**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Mean Average Precision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>UvA-Evision</td>
<td>0.22581</td>
<td>Hand-crafted features</td>
</tr>
<tr>
<td>2</td>
<td>NEC-MU</td>
<td>0.20895</td>
<td>Hand-crafted features</td>
</tr>
<tr>
<td>3</td>
<td>NYU</td>
<td>0.19400</td>
<td>Deep learning</td>
</tr>
</tbody>
</table>

- **ImageNet 2014 – image classification challenge**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Error rate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Google</td>
<td>0.06656</td>
<td>Deep learning</td>
</tr>
<tr>
<td>2</td>
<td>Oxford</td>
<td>0.07325</td>
<td>Deep learning</td>
</tr>
<tr>
<td>3</td>
<td>MSRA</td>
<td>0.08062</td>
<td>Deep learning</td>
</tr>
</tbody>
</table>

- **ImageNet 2014 – object detection challenge**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Mean Average Precision</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Google</td>
<td>0.43933</td>
<td>Deep learning</td>
</tr>
<tr>
<td>2</td>
<td>CUHK</td>
<td>0.40656</td>
<td>Deep learning</td>
</tr>
<tr>
<td>3</td>
<td>Deepinsight</td>
<td>0.40452</td>
<td>Deep learning</td>
</tr>
<tr>
<td>4</td>
<td>UvA-Evision</td>
<td>0.35421</td>
<td>Deep learning</td>
</tr>
<tr>
<td>5</td>
<td>Berkeley Vision</td>
<td>0.34521</td>
<td>Deep learning</td>
</tr>
</tbody>
</table>

CNN examples
CNN example: Image labeling at Google

DeViSE: A Deep Visual-Semantic Embedding Model

(Frome et al, NIPS 2013)

- Key idea
  - represent labels (words) and images in the same space so that labeling can be performed via k-nearest neighbors

![Diagram](Figure from Frome et al, NIPS 2013)

Figure 1: (a) Left: a traditional visual object categorization network with a softmax output layer. Right: a skip-gram language model, which learns word representations that allow the prediction of nearby words in a document. Center: our joint model, which is initialized with parameters pre-trained at the lower layers of the other two models. (b) t-SNE visualization [16] of the skip-gram language model embedding space across a subset of image recognition labels.

\[
\text{loss}(\text{image}, \text{label}) = \sum_{j \neq \text{label}} \max[0, \text{margin} - \tilde{t}_{\text{label}} M \tilde{v}(\text{image}) + \tilde{t}_j M \tilde{v}(\text{image})]
\]
Image labeling at Google
DeViSE: A Deep Visual-Semantic Embedding Model (Frome et al, NIPS 2013)

Our model  Softmax over ImageNet 1K

eyepiece, ocular  typewriter keyboard
Polaroid  tape player
compound lens  reflex camera
telephoto lens, zoom lens  CD player
rangefinder, range finder  space bar

A
reel
punching bag, punch bag, ...

B
whistle
bassoon
bassoon
letter opener, paper knife, ...

English horn, cor anglais
hook and eye
hand

G
patas, bushar monkey, ...
proboscis monkey, Nasalis ... 
macaque
maraquine

titmouse, tit
 titi, titi monkey
bovetbird, catbird
quenca, quenon monkey

D
pineapple, ananas
coral fungus
artichoke, globe artichoke
sea anemone, anemone
sweet orange, orange tree, ...
cardoon

fresu

E
comestible, edible, ...
pot, sugarpot
dressing, salad dressing
cauliflower

Sicilian pizza

vegetable, veggie, veg

fruit

F
cucumber, cuke
broccoli

dune buggy, beach buggy
warplane, military plane
missle

searcher, search, quester
projectile, missile

Tregaralpbus eurycerus, ...
sports car, sport car

bongo, bongo drum
submarine, pigboat, sub, ...

Figure 2: Zero-shot predictions of the joint semantic visual model and a vision model trained on ImageNet 2012 1K. Predictions ordered by decreasing score. Correct predictions labeled in green. Ground truth: (a) telephoto lens, zoom lens; (b) English horn, cor anglais; (c) babbler, cockler; (d) pineapple, pineapple plant, Ananas comosus; (e) salad bar; (f) spacecraft, ballistic capsule, space vehicle.
CNN example: A neural algorithm of Artistic Style (Gatys et al. 2015)

Generate images by combining content and style

Makes use of a discriminatively trained CNN

Image generation
  • inverse problem on the CNN

https://deepart.io
• Idea (simplified)
  • $c$ input a content image and $F_c$ a filter representation of $c$
  • $a$ input art image and $G_a$ a filter correlation representation of $a$
  • $x$ a white noise image, $F_x$ and $G_x$ the corresponding filter and filter correlation representations
  • loss:
    • $L = \|F_c - F_x\|^2 + \alpha \|G_a - G_x\|^2$
• Generated image
  • Solve an inverse problem
    • $\hat{x} = \text{argmin}_x(L)$
    • Solved by gradient
Encoder-Decoder paradigm: example of neural translation — (Cho et al. 2014, Sutskever et al. 2014)

• Translation
  • Given a sentence in language A: $x_1, ..., x_T$
  • Translate it into a sentence in language B: $y_1, ..., y_T$,
  • training:
    • Learn conditional distribution $P(y_1, ..., y_T \mid x_1, ..., x_T)$

• Neural Machine Translation
  • Use NNs to learn this conditional distribution
  • Example with recurrent neural networks
    • Encode an input sentence $x_1, ..., x_T$ in a fixed vector $c$
    • Decode the output sentence from vector $c$
    • Training criterion: conditional likelihood:
      \[ E_{sequences} (x,y) \left[ P(y_1, ..., y_T \mid x_1, ..., x_T) \right] \]
Encoder-Decoder paradigm: example of neural translation — (Cho et al. 2014, Sutskever et al. 2014)

- Improved versions of recurrent NN are today effective models for sequence processing
  - E.g. state of the art for speech recognition, language modeling, etc
  - Ongoing developments for translation

- Basic Scheme

Recurrent NN

Unfolded recurrent NN for translation
Neural image caption generator (Vinyals et al. 2015)

• Objective
  • Learn a textual description of an image
    • i.e. using an image as input, generate a sentence that describes the objects and their relation!

• Model
  • Inspired by a translation approach but the input is an image
    • Use a RNN to generate the textual description, word by word, provided a learned description of an image via a deep CNN

Figure 1. NIC, our model, is based end-to-end on a neural network consisting of a vision CNN followed by a language generating RNN. It generates complete sentences in natural language from an input image, as shown on the example above.
Neural image caption generator (Vinyals et al. 2015)

• Loss criterion
  • \( \max_{\theta} \sum_{I,S} \log p(S|I; \theta) \)
    • Where \((I,S)\) is an associated couple (Image, Sentence)
  • \( \log p(S|I; \theta) = \sum_{t=1}^{N} \log p(S_t|I, S_0, \ldots, S_{t-1}) \)
  • \( p(S_t|I, S_0, \ldots, S_{t-1}) \) is modeled with a RNN with \( S_0, \ldots, S_{t-1} \) encoded into the hidden state \( h_t \) of the RNN
  • Here \( h_{t+1} = f(h_t, x_t) \) is modelled using a RNN with LSTM
  • For encoding the image, a CNN is used
Neural image caption generator (Vinyals et al. 2015)

Figure 5. A selection of evaluation results, grouped by human rating.
Summary

• Unprecedented developments in ML in general
  • Conjonction of several factors
    • Data deluge, Computing power, Free software ML libraries by major actors
      • e.g. Tensor Flow (Google), Torch7 (Facebook)
    • Implication of big players and Fast prototype to industrial deployment

• NNs are today at the heart of this development
  • Powerful models
  • Modularity allows to build complex systems, trainable end to end
  • State of the art in many domains
  • Theory still to be developed!
References and links

• Videos used in the talk
  • Interview of B. Widrow on the Adaline
    https://www.youtube.com/watch?v=IEFRtz68m-8
  • The neocognitron of K. Fukushima, an ancestor of convolutional neural networks but without BackProp. Not used in the talk but interesting to look at.
    https://www.youtube.com/watch?v=Qil4kmvm2Sw
  • Demo of LeNet – Early Convolutional Neural Network
    http://yann.lecun.com/exdb/lenet/index.html
  • NYU Semantic Segmentation with a Convolutional Network (33 categories)
    https://www.youtube.com/watch?v=ZJMtDRbqH40&feature=youtu.be
  • NYU Pedestrian Detection
    https://www.youtube.com/watch?v=MnZNSZGNyGc
    https://www.youtube.com/watch?v=UPVvd8WNUks
  • Hand gesture Recognition
    https://www.youtube.com/watch?v=GhqOMJIHD8A
References

The 1960s - Early days of Neural Networks

The 1990s – many books were published at that time, two of my favorites are:
Hertz J.A., Krogh A.S., Palmer R.G. Introduction To The Theory Of Neural Computation (Santa Fe Institute Series), 1991, introduces a variety of NN models developed in the 80es
Bishop C.M., Neural Networks for Pattern Recognition, Oxford University Press, 1995, more oriented towards statistical machine learning

The 2010s

Papers used as illustrations for the presentation


