Image Analysis (FMAN20)
Lecture 7, 2018

MAGNUS OSKARSSON
Image Analysis - Motivation

Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books
Yukun Zhu*, Ryan Kiros*, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, Sanja Fidler
Arxiv, June 2015
Overview – Deep Learning

1. History and Motivation
2. Components of Deep Learning
   1. Convolution
   2. Non-linear
   3. Max pooling
   4. Soft-Max
3. Network Design
4. Training
5. Examples
Computer vision
bridge the gap between pixels and meaning

Images are collections of intensity measurements (or RGB, or …)
Computer vision
bridge the gap between pixels and meaning
Autonomous Precision Livestock Farming
Deep learning
Convolutional Neural Networks

- Slides and material from
- MatConvNet
  - http://www.robots.ox.ac.uk/~vgg/practicals/cnn/
- Gabrielle Flood’s master’s thesis
- Anna Gummeson’s master’s thesis
Components for deep learning

- One neuron
- Example: Logistic regression
- Classification model ($x$ feature vector, ($w,b$) parameters, $s$ smooth thresholding)
  \[ x \in \mathbb{R}^d, w \in \mathbb{R}^d, b \in \mathbb{R}, f(x) = s(w^T x + b) \]
- Logistic regression
  \[ s(z) = \frac{1}{1 + e^{-x}} \]
- ML estimate of parameters ($w,b$) is a convex optimization problem
Single Layer Neural Networks

One Neuron

- One neuron

\[ x \in \mathbb{R}^d, \; w \in \mathbb{R}^d, \; b \in \mathbb{R}, \; f(x) = s(w^T x + b) \]
Single Layer Neural Networks
Several Neurons

- Several parallel neurons

\[ x \in \mathbb{R}^d, \ y \in \mathbb{R}^k, \ B \in \mathbb{R}^d, \ W - k \times d \text{matrix} \]

\[ y = s(Wx + B) \]

- Elementwise smooth thresholding – \( s \)
Artificial Neural Networks
One hidden layer

- Multi-class classification
- One hidden layer
- Trained by back-propagation
- Popular since the 1990s
Deep Neural Networks
Many layers

• However
• Naively implemented would give to many parameters
• Example
  • 1M pixel image
  • 1M hidden layers
  • $10^{12}$ parameters between each pair of layers
Convolutional neural network, CNN
CNN-Blocks - Convolutional layer

Convolution of an image as a filter-operation.

Original

Result

Filter

Flipped filter

Convolution of an image represented as a sparsely connected ANN.
CNN-Blocks - Convolutional layer

- Input: Data block $x$ of size $m \times n \times k_1$

- Output: Data block $y$ of size $m \times n \times k_2$

- Filter: Filter kernel block $w$ of size $m_w \times n_w \times k_1 \times k_2$

- Offsets: Vector $w_o$ of length $k_2$

$$y(i, j, k) = w_o(k) + \sum_u \sum_v \sum_l x(i - u, j - v, l)w(u, v, l, k)$$
CNN-Blocks - Convolutional layer

\[ y(i, j) = \sum_u \sum_v x(i - u, j - v)w(u, v) \]

\[ y(i, j) = w_o + \sum_l \left( \sum_u \sum_v x(i - u, j - v, l)w(u, v, l) \right) \]

\[ y(i, j, k) = w_o(k) + \sum_u \sum_v \sum_l x(i - u, j - v, l)w(u, v, l, k) \]
CNN-Blocks - Max-pooling
CNN-Blocks - RELU

\[ f(x) = \max(0, x) \]

\[ y(i, j, k) = \max(x(i, j, k), 0) \]
CNN-Blocks – Softmax
(convert from ’log probabilities’ dj to ’probabilities’ that sum to 1)

\[ p_j = \frac{e^{d_j}}{\sum_{k=1}^{m} e^{d_k}} \]
Result, Network design
CNN-Blocks – input – output

\[ y = f(x, w) \]

Input: image \( x \) of size \( m \times n \times k \), typically \( k=1 \) (gray-scale) or \( k=3 \) (color)
Output: vector \( y \) of size \( 1 \times 1 \times N \), which we interpret as \( N \) probabilities \( y_j \)
The probability that the image \( x \) is of class \( j \)
Training data \((x_i, c_i)\)

\[
y = f(x, w)
\]

Input: image \(x\) of size \(m \times n \times k\), typically \(k=1\) (grayscale) or \(k=3\) (colour)
Output: vector \(y\) of size \(1 \times 1 \times N\), which we interpret as \(N\) probabilities \(y_j\)
The probability that the image \(x\) is of class \(j\) \(y_{c_i}\)
Training data $T = \{(x_1, c_1), \ldots (x_N, c_N)\}$

- Classification network  
  $$y = f(x, w)$$

- Evaluate one example $(x_k, c_k)$ (like adding another layer)
  $$\sum_{k=1}^{N} - \log y(x_k, w)_{c_k}$$

- Evaluation function:
  $$g(T, w) = \sum_{k=1}^{N} - \log y(x_k, w)_{c_k}$$

- Solve
  $$\min_w g(T, w)$$
Example: OCR, classify images as a-z, Network design

```matlab
>> net

net =

    layers: {1x7 cell}
          imageMean: 0.9176775

>> vl_simplenn_display(net)

<table>
<thead>
<tr>
<th>layer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>cnv</td>
<td>mpool</td>
<td>cnv</td>
<td>mpool</td>
<td>cnv</td>
<td>relu</td>
<td>cnv</td>
</tr>
<tr>
<td>support</td>
<td>5x5</td>
<td>2x2</td>
<td>5x5</td>
<td>2x2</td>
<td>4x4</td>
<td>1x1</td>
<td>2x2</td>
</tr>
<tr>
<td>stride</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>pad</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>out dim</td>
<td>20</td>
<td>20</td>
<td>50</td>
<td>50</td>
<td>500</td>
<td>500</td>
<td>26</td>
</tr>
<tr>
<td>filt dim</td>
<td>1</td>
<td>n/a</td>
<td>20</td>
<td>n/a</td>
<td>50</td>
<td>n/a</td>
<td>500</td>
</tr>
<tr>
<td>rec. field</td>
<td>5</td>
<td>6</td>
<td>14</td>
<td>16</td>
<td>28</td>
<td>28</td>
<td>32</td>
</tr>
<tr>
<td>c/g net KB</td>
<td>4/0</td>
<td>0/0</td>
<td>196/0</td>
<td>0/0</td>
<td>3129/0</td>
<td>0/0</td>
<td>406/0</td>
</tr>
</tbody>
</table>

total network CPU/GPU memory: 3.6/0 MB
```
Example: OCR, classify images as a-z
Training data

training chars for 'a'
Example: OCR, classify images as a-z, Training

\[ g(T, w) = \sum_{k=1}^{N_{\text{training}}} - \log y(x_k, w)_{c_k} \]

\[ \#\{c_k = \arg\max_i y(x_k, w)_i\} \]


**Tricks**

- **Stochastic Gradient Descent**
  - Computation of \( g(T, w) = \sum_{k=1}^{N} - \log y(x_k, w)_{c_k} \)
  - Requires going through all examples (all N).
  - If N is large and/or if computing \( y(x_k, w) \) is time-consuming, use stochastic gradient descent, i.e. update parameters using subsets of training data.

- **Jittering** - construct a larger training set by perturbing the examples, jittering, translating images, rotating images, warping, mirroring, adding noise, ...

- **Dropout** – in each computation of \( y(x_k, w) \) let a random subset of the neurons die, i.e. set the output to zero.
Generalisation, Expand data set
Generalisation, Dropout
Generalisation, Weight decay
(Prior on small weights)

\[ E(w) = - \sum_{n=1}^{N} \log \left( \frac{e^{y_d(n)}(n)}{\sum_{i=1}^{k} e^{y_i(n)}} \right) + \frac{\lambda}{2} \sum_{l} w^2_l \]
Thoughts

- Modelling. It takes time to
  - Figure out an appropriate network structure
  - Gather data and ground truth
- The optimization does not always work
  - Parameters explode
  - Nothing happens
- Visualization of the features is important for understanding.
- Feedback in networks
Example: Prostate cancer
Data
Result, Cross-validation
Example: Prostate cancer Training
Example: Prostate cancer
Results: Confusion matrix

\[
\begin{bmatrix}
51 & 0 & 0 & 0 \\
3 & 46 & 3 & 1 \\
0 & 6 & 43 & 0 \\
0 & 3 & 0 & 52
\end{bmatrix}
\]
Visualisation
Examples: Image net, Data

• ImageNet Large Scale Visual Recognition Challenge
• Yearly challenge since 2010
• 2011 - 25% error
• 2012 - 16% error (using CNN). This kicked off the deep learning hype
• 2015 – 4% error
• By 2015, researchers reported that current software exceeds human ability at the narrow ILSVCR tasks.
• However, as one of the challenge’s organisers, Olga Russakovsky, pointed out in 2015, the programs only have to identify images as belonging to one of a thousand categories!
Training Deep Learning

- Data
  - Obtain data,
  - cut-outs of right size,
  - jittering,
  - Data expansion (translation, rotation, scaling, mirroring, adding noise, …)

- Data
  - Obtain ground truth
  - How should the problem be coded
Training Deep Learning

- Hyperparameters
  - How many layers
  - Size of convolution kernels
  - Number of channels
  - Order of layers

- Training parameters
  - Initializing weights
  - Momentum
  - ...
Non-linear function

- Different choices of non-linear functions.
- Faster learning
- Rectifier ...
  \[ f(x) = \max(0, x) \]
- ... currently most popular, faster training
- Arguments

\[
\begin{align*}
  f(x) &= \max(0, x) \\
  f(x) &= \ln(1 + e^x) \\
  f(x) &= \begin{cases} 
    x & \text{if } x > 0 \\
    0.01x & \text{otherwise}
  \end{cases} \\
  f(x) &= (1 + e^{-x})^{-1} \\
  f(x) &= \tanh(x)
\end{align*}
\]
Training Deep Learning

- Once all of these are in place, there are several good systems for optimizing the parameters
  - MatConvNet, TensorFlow, Caffe, Torch7, Theano
  - Can train on single CPU
  - Faster if compiled for GPU
  - Even faster on cluster of computers with multiple GPU (e.g. LUNARC, http://www.lunarc.lu.se)
- More links on home page for PhD course
  - http://www.control.lth.se/Education/DoctorateProgram/deep-learning-study-circle.html
Deep learning - summary

• What is deep learning
• Supervised vs unsupervised learning
• Goal function, energy function $E$
• Choice of non-linearity ReLU
• Optimization Back-propagation, SGD
• Tricks dropout
• Examples from speech and vision
• Software
• References
• To learn more – take course on Machine Learning FMAN45
Master’s Thesis Suggestion
Masters thesis suggestion of the day

- How do we get training data for this?
Masters thesis suggestion of the day

- Turn it around!
- This is easy to get training data for
Masters thesis suggestion:
Action-Based Learning in Brain Circuits and Deep Artificial Neural Networks

Research collaboration with Christian Balkenius, Per Petersson, Jeanette Hellgren Kotaleski

Figure 1. The robot Epi grasping an object.

Figure 4. Neuronal firing rate modulation in cortico-basal ganglia circuits reveals distinct action bracketing in spontaneous grooming behavior (Tamte et al., unpublished).