Deep Learning: introduction, trends & tendencies

Christian Wolf
Université de Lyon, INSA-Lyon
LIRIS UMR CNRS 5205

13 Décembre 2016
We would like to predict a value $t$ from an observed input

$$y = f(x, \theta)$$

Parameters $\theta$ are learned from training data.
Supervised learning

In supervised training, we minimize empirical loss over a training set:

prediction: \( y = f(x, \theta) \)

ground-truth: \( y^* \)

loss function: \( \mathcal{L}(y, y^*) \)

Empirical risk minimization:

\[
\hat{\theta} = \min_{\theta} \sum_{i=1}^{N} \mathcal{L}(f(x_i, \theta), y_i^*)
\]

Example for a classification problem: cross-entropy loss

\[
\mathcal{L}(y_i, y_i^*) = - \sum_{k} y_{i}^{k*} \log y_i^k
\]

output for class \( k \)
Learning by gradient descent

Iterative minimisation through **gradient descent**:

\[
\theta^{[t+1]} = \theta^{[t]} + \nu \nabla \mathcal{L}(f(x, \theta), y^*)
\]

Can be blocked in a local minimum (not that it matters much …)
As discussed in Section 3.1, the bias parameters in (5.2) can be absorbed into the set of weight parameters by defining an additional input variable $x_0$ whose value is clamped at $x_0 = 1$, so that (5.2) takes the form

$$a_j = \sum_{i=0}^{D} w_{ji} x_i.$$ 

(5.8)

We can similarly absorb the second-layer biases into the second-layer weights, so that the overall network function becomes

$$y_k(x, w) = \sigma \left( \sum_{j=0}^{M} w_{kj}^{(2)} h \left( \sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right).$$

(5.9)

As can be seen from Figure 5.1, the neural network model comprises two stages of processing, each of which resembles the perceptron model of Section 4.1.7, and for this reason the neural network is also known as the multilayer perceptron, or MLP. A key difference compared to the perceptron, however, is that the neural network uses continuous sigmoidal nonlinearities in the hidden units, whereas the perceptron uses step-function nonlinearities. This means that the neural network function is differentiable with respect to the network parameters, and this property will play a central role in network training.

If the activation functions of all the hidden units in a network are taken to be linear, then for any such network we can always find an equivalent network without hidden units. This follows from the fact that the composition of successive linear transformations is itself a linear transformation. However, if the number of hidden units is smaller than either the number of input or output units, then the transformations that the network can generate are not the most general possible linear transformations from inputs to outputs because information is lost in the dimensionality reduction at the hidden units. In Section 12.4.2, we show that networks of linear units give rise to principal component analysis. In general, however, there is little interest in multilayer networks of linear units.

The network architecture shown in Figure 5.1 is the most commonly used one in practice. However, it is easily generalized, for instance by considering additional layers of processing each consisting of a weighted linear combination of the form (5.4) followed by an element-wise transformation using a nonlinear activation function. Note that there is some confusion in the literature regarding the terminology for counting the number of layers in such networks. Thus the network in Figure 5.1 may be described as a 3-layer network (which counts the number of layers of units, and treats the inputs as units) or sometimes as a single-hidden-layer network (which counts the number of layers of hidden units). We recommend a terminology in which Figure 5.1 is called a two-layer network, because it is the number of layers of adaptive weights that is important for determining the network properties.

Another generalization of the network architecture is to include skip-layer connections, each of which is associated with a corresponding adaptive parameter.

Network architectures

$$y_k(x, w) = \sigma \left( \sum_{j=0}^{M} w_{kj}^{(2)} h \left( \sum_{i=0}^{D} w_{ji}^{(1)} x_i \right) \right)$$
« LeNet »

[LeCun et al., 1998]
Data augmentation

Add cropped and mirrored versions of each training image
Increases invariance
Unsupervised Learning

« The brain has about $10^{14}$ synapses and we only live for about $10^9$ seconds. So we have more parameters than data. This motivates the idea that we must do a lot of unsupervised learning since the perceptual input is the only place we can get $10^5$ dimensions of constraint per second. »

Geoffrey Hinton
« What is a cat? »
Learning from various sources

Generative Adversarial Networks (GANs)

[Karpathy et al. 2016]

[Lerer, Gross, Fergus, ICML 2016]

[Agrawal et al., NIPS 2016]
Structured Output Prediction

Predicting for multiple inter-dependent variables

Output Classes

Single-label Problem

Multi-label Problem
Learning from heterogeneous data

- Learn from multiple datasets annotated for different tasks
- Each datapoint can have multiple correlated labels

[Fourure, Emonet, Fromont, Muselet, Tremeau, Neverova, Wolf, (Under Review)]
Learning from heterogeneous data

Dataset 1

Dataset 2

Dataset 3

CNN

Dataset-wise soft-max

Label set 1

Label set 2

Label set 3

Selective cross-entropy loss function

[fourure, Emonet, Fromont, Muselet, Tremeau, Neverova, Wolf, (Under Review)]
Structured Output Prediction

Predicting for multiple inter-dependent variables

- Sequences
- Images and other 2D grids
- Kinematic trees

Multi-label Problems
Graphical models

\[ x_i \]

\[ y_i \]

\[
\begin{align*}
A & \perp B \\
A & \not\perp B \\
A & \not\perp C | B
\end{align*}
\]
Graphical models

HMM

\[ p(x, y) \]

CRF

\[ p(y|x) \]
Graphical models

Prediction requires to optimize over the exponential space of all output labels (and eventual latent variables).
In general, intractable.
Models are often simple (of low order).
Recurrent neural networks

Prediction: feed-forward computation in a DAG. No optimization is needed.
GM vs. NN

- GM : state is stochastic
- GM : make it easier for experienced practioneers to model known relationships between data.
- GM : optimization required for prediction
- GM : specific structures allow to obtain global optima with message passing (chains and trees) or graph cuts (submodular potentials) etc.

- NN : state is deterministic
- NN : complex models with componential hidden states
- NN : higher order interactions are easier to handle
- NN : no optimization during prediction

Is it better to get a global min/max of a simple model or a feed-forward prediction for a high-capacity model trained on a large amount of data?
Structured Prediction Energy Networks

Learn a prior on the label set.
Perform gradient descent on the relaxed output labels.

[Belanger and McCallum, ICML 2016]

Figure: Damien Fourure
Some concrete examples …
Entering PINs on smartphones is painful!

AND STARTING TODAY, ALL PASSWORDS MUST CONTAIN LETTERS, NUMBERS, DOODLES, SIGN LANGUAGE AND SQUIRREL NOISES.

© 2005 Scott Adams, Inc. / Dist. by UFS, Inc.

Automatically authenticate smartphone users given their behavior (=interaction style). Shut off phone when theft is detected.
Project "Abacus » (Google)

- 1500 volunteers, 1500 Nexus 5 smartphones
- Several months of natural daily usage, 27.6 TB of data
- Multiple sensors: camera, touchscreen, GPS, bluetooth, wifi, cell antenna, inertial, magnetometer
- **This work**: inertial sensors only, recorded at 200Hz

Work of Natalia Neverova
LIRIS/INSA-Lyon
Now at Facebook AI Research

With Graham W. Taylor,
University of Guelph, Canada

Learning data representations in the biometric framework

- static ConvNet aggregating temporal statistics
- discriminative pretraining
- explicit modeling of temporal transitions by recurrent connections
Vanilla RNN vs Clockwork RNN

[Koutnik et al., 2014]
Clockwork RNN vs Dense Clockwork RNN: update rule
On shift-invariance
Experimental results

<table>
<thead>
<tr>
<th>Model</th>
<th>EER, %</th>
<th>HTER, %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw features</td>
<td>36.21</td>
<td>42.17</td>
</tr>
<tr>
<td>ST Convnet</td>
<td>32.44</td>
<td>34.89</td>
</tr>
<tr>
<td>LT Convnet</td>
<td>28.15</td>
<td>29.01</td>
</tr>
<tr>
<td>Conv-RNN</td>
<td>22.32</td>
<td>22.49</td>
</tr>
<tr>
<td>Conv-CWRNN</td>
<td>21.52</td>
<td>21.92</td>
</tr>
<tr>
<td>Conv-LSTM</td>
<td>21.13</td>
<td>21.41</td>
</tr>
<tr>
<td>Conv-DCWRNN</td>
<td>20.01</td>
<td>20.52</td>
</tr>
<tr>
<td><strong>Conv-DCWRNN, zt-norm</strong></td>
<td><strong>18.17</strong></td>
<td><strong>19.29</strong></td>
</tr>
<tr>
<td><strong>Conv-DCWRNN (per device)</strong></td>
<td><strong>15.84</strong></td>
<td><strong>16.13</strong></td>
</tr>
<tr>
<td><strong>Conv-DCWRNN (per session)</strong></td>
<td><strong>8.82</strong></td>
<td><strong>9.37</strong></td>
</tr>
</tbody>
</table>

Structured Output Prediction

Predicting for multiple inter-dependent variables

- Sequences
- Images and other 2D grids
- Kinematic trees

Multi-label Problems
Layers, scales and spatial resolutions

Reducing spatial resolution assures that the receptive fields of fields increase in higher layers (gain in abstraction).

Challenge in cases where decisions need to made in high resolution (segmentation, super-resolution etc.)
Layers, scales and spatial resolutions

Reducing spatial resolution assures that the receptive fields of fields increase in higher layers (gain in abstraction).

Challenge in cases where decisions need to made in high resolution (segmentation, super-resolution etc.)
Object detection through regression

[Redmon, Divvala, Girshick, Farhadi, CVPR 2016]
Detection with spatial context layers

[Moysset, Kermorvant, Wolf, (under review+arxiv 2016)]
Spatial context layers

[Moysset, Kermorvant, Wolf, (under review+arxiv 2016)]
Results

### Text Recognition Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>Precision</th>
<th>F-Meas.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shi et al. [28]</td>
<td>35.1%</td>
<td>38.4%</td>
<td>36.7%</td>
</tr>
<tr>
<td>Nicolaou et al. [19]</td>
<td>46.7%</td>
<td>39.6%</td>
<td>42.9%</td>
</tr>
<tr>
<td>Multibox [6]</td>
<td>4.2%</td>
<td>10.0%</td>
<td>6.0%</td>
</tr>
<tr>
<td>Multibox [6] (optimized)</td>
<td>28.8%</td>
<td>52.3%</td>
<td>31.1%</td>
</tr>
<tr>
<td>Ours, no LSTMs</td>
<td>28.6%</td>
<td>52.4%</td>
<td>31.1%</td>
</tr>
<tr>
<td>Ours</td>
<td>51.2%</td>
<td>61.4%</td>
<td>55.9%</td>
</tr>
</tbody>
</table>

### Detection and Recognition Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>French (507 pages)</th>
<th>English (265 pages)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shi et al. [28]</td>
<td>48.6%</td>
<td>30.4%</td>
</tr>
<tr>
<td>Nicolaou et al. [19]</td>
<td>65.3%</td>
<td>50.0%</td>
</tr>
<tr>
<td>Multibox [6]</td>
<td>27.2%</td>
<td>14.8%</td>
</tr>
<tr>
<td>Multibox [6] (optimized)</td>
<td>32.4%</td>
<td>36.2%</td>
</tr>
<tr>
<td>Ours, no LSTMs</td>
<td>57.8%</td>
<td>56.9%</td>
</tr>
<tr>
<td>Ours</td>
<td>71.2%</td>
<td>71.1%</td>
</tr>
</tbody>
</table>

[Moysset, Kermorvant, Wolf, (under review+arxiv 2016)]
Gesture recognition

Project **Interabot**
Awabot / LIRIS / LIG / Voxler
Work of Natalia Neverova
LIRIS/INSA-Lyon
Now at Facebook AI Research

With Graham W. Taylor,
University of Guelph, Canada
Multi-modal input

Skeleton descriptor

Depth and intensity images

Audio stream
Single-scale deep architecture. Individual classifiers are pre-trained for each data modality and then fused using a 2-layer fully connected network initialized in a specific way.
Results

[Neverova, Wolf, Taylor, Nebout, IEEE-T-PAMI 2016]
ECCV 2014 Challenge on Looking at People (gesture recognition track)

<table>
<thead>
<tr>
<th></th>
<th>Team</th>
<th>Score</th>
<th></th>
<th>Team</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ours [47]</td>
<td>0.850</td>
<td>7</td>
<td>Camgoz et al. [61]</td>
<td>0.747</td>
</tr>
<tr>
<td>2</td>
<td>Monnier et al. [29]</td>
<td>0.834</td>
<td>8</td>
<td>Evangelidis et al. [62]</td>
<td>0.745</td>
</tr>
<tr>
<td>3</td>
<td>Chang [30]</td>
<td>0.827</td>
<td>9</td>
<td>Undisclosed authors</td>
<td>0.689</td>
</tr>
<tr>
<td>4</td>
<td>Peng et al. [63]</td>
<td>0.792</td>
<td>10</td>
<td>Chen et al. [64]</td>
<td>0.649</td>
</tr>
<tr>
<td>5</td>
<td>Pigou et al. [36]</td>
<td>0.789</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Wu [37]</td>
<td>0.787</td>
<td>17</td>
<td>Undisclosed authors</td>
<td>0.271</td>
</tr>
</tbody>
</table>

Ours, improved results after the competition 0.870

[Neverova, Wolf, Taylor, Nebout, IEEE-T-PAMI 2016]
Learning deep architectures

Path V1:
- depth video, right hand
- intensity video, right hand

Path V2:
- depth video, left hand
- intensity video, left hand

Path M:
- mocap stream
- pose feature extractor

Path A:
- audio stream
- mel frequency histograms

Single-scale deep architecture. Individual classifiers are pre-trained for each data modality and then fused using a 2-layer fully connected network balanced in a specific way.
« Modout » : stochastic regularisation

(a) Dropout
(b) DropConnect
(c) Blockout
(d) Modout

[Li, Taylor, Neverova, Wolf, (under review)]
Learning multimodal architectures

Classification accuracy(%) of different stochastic regularization methods on Montalbano.

<table>
<thead>
<tr>
<th></th>
<th>BackProp</th>
<th>Dropout</th>
<th>Blockout</th>
<th>ModDrop</th>
<th>Modout</th>
<th>Modout +Dropout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation accuracy</td>
<td>91.1</td>
<td>91.5</td>
<td>91.7</td>
<td>92.1</td>
<td>91.6</td>
<td><strong>92.9</strong></td>
</tr>
<tr>
<td>Test accuracy</td>
<td>92.0</td>
<td>92.5</td>
<td>92.6</td>
<td>92.4</td>
<td>93.6</td>
<td><strong>93.8</strong></td>
</tr>
</tbody>
</table>

Fan Li
1986 - 2016
Some peculiar properties of deep models
Model fitting and generalization

How do we chose model complexity?

![Polynomial Curve Fitting Example](image)

1.1. Example: Polynomial Curve Fitting

Figure 1.4

Plots of polynomials having various orders $M$, shown as red curves, fitted to the data set shown in Figure 1.2.

The $\text{RMS}$ error defined by

$$E_{\text{RMS}} = \sqrt{\frac{2}{E(w^\star)}}/N$$  \hspace{1cm} (1.3)

in which the division by $N$ allows us to compare different sizes of data sets on an equal footing, and the square root ensures that $E_{\text{RMS}}$ is measured on the same scale (and in the same units) as the target variable $t$.

Graphs of the training and test set RMS errors are shown, for various values of $M$, in Figure 1.5. The test set error is a measure of how well we are doing in predicting the values of $t$ for new data observations of $x$.

We note from Figure 1.5 that small values of $M$ give relatively large values of the test set error, and this can be attributed to the fact that the corresponding polynomials are rather inflexible and are incapable of capturing the oscillations in the function $\sin(2\pi x)$.

Values of $M$ in the range $3 \leq M \leq 8$ give small values for the test set error, and these also give reasonable representations of the generating function $\sin(2\pi x)$, as can be seen, for the case of $M = 3$, from Figure 1.4.

[C. Bishop, Pattern recognition and Machine learning, 2006]
Overfitting decreases with increasing amount of data

\[ M = 9 \]
Regularisation

Restrict the expressive power of the model.

\[ \tilde{E}(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ y(x_n, \mathbf{w}) - t_n \right\}^2 + \frac{\lambda}{2} \| \mathbf{w} \|^2 \]

Regularisation parameter

\( w_0 \) is often omitted

\( M=9 \)

[\text{C. Bishop, Pattern recognition and Machine learning, 2006}]
Learning and generalization

Can we measure the complexity of a network architecture (a hypothesis class)?

Rademacher complexity of a class $\mathcal{H}$ with respect to samples $x$ and a distribution of random labels $\sigma_i$:

$$\mathcal{R}(\mathcal{H}) = \mathbb{E}_{\sigma} \left[ \sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i} \mathcal{L}(f(x_i, \theta), \sigma_i) \right]$$
However …

- Classical theory seems to fail with DL
- Deep models generalize very well, indicating model complexity fitted to the problem
- Recent work: deep models can learn random labels (!) on very large problems (ImageNet!) with zero error
- « The effective capacity of a neural network is large enough for brute-force memorization of the entire data set »

[Zhang, S Bengio, Hardt, Recht, Vinyals, arxiv 10/2016]
Failure cases

An airplane is parked on the tarmac at an airport.

A man riding a motorcycle on a beach.

[Antoine Bordes, Facebook AI Research, 2016]
Costs of DL
Conclusion: current tendencies

- Learning where to look (attention mechanisms)

- Learning what to store where (e.g. external memories, Neural Turing machines)

- Learning how to generate data (e.g. adversarial networks) and through it, representations.

- Learning to go deeper (e.g. residual networks, highway networks)

- Combine Vision and Language

- Deep Reinforcement Learning