Deep Learning: history, models & challenges, with an application in signal processing and mobile authentification

Christian Wolf
INRIA-Chroma, CITI, LIRIS, INSA-Lyon
Learning to predict

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

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

Parameters $\theta$ are learned from training data.
The 3 fundamental problems of ML

1. Expressivity
   – What is the complexity of functions my model can represent?

2. Trainability
   – How easy is it to fit my model to my training data?

3. Generalization
   – Does my model generalize to unseen data?
   – In presence of shifts in distributions?

(After Eric Jang & Jascha Sohl-Dickstein)
The standard toolbox

Deep neural networks

Convolutions introduce an inductive bias for imaging/vision applications.

Recurrent networks allow to model sequences

[LeCun et al., 1998]  [Hocheiter and Schmidhuber, 1997]
Deep neural networks
Learning by gradient descent

Iterative minimisation through gradient descent:

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

Learning rate

Can be blocked in a local minimum (not that it matters much …)

[Figure: C. Bishop, 2006]
Some gentle words on learning theory

\[ L_{\mathcal{D}, f(h)} \overset{\text{def}}{=} \mathbb{P}_{x \sim \mathcal{D}} [h(x) \neq f(x)] \]

Expected error over (unknown) data distribution \( \mathcal{D} \) and (groundtruth) labeling function \( f \)

Model \( h \)

Labeling function (ground truth)

[Shai Shalev-Shwartz et al. 2014]
Empirical Risk Minimization

\[ L_S(h) \overset{\text{def}}{=} \frac{|\{ i \in [m] : h(x_i) \neq y_i \}|}{m}, \]

Empirical error over Samples \( y_i \) from training set \( S \)

How different is the empirical error from the expected error?

[Shai Shalev-Shwartz et al. 2014]
Sources of error

\[ L_{\mathcal{D}, f}(h) \overset{\text{def}}{=} \mathbb{P}_{x \sim \mathcal{D}} [h(x) \neq f(x)] \]

1. Lack of generalization (shift between empirical error and expected error on the target domain)

2. Optimization problem (the solution of the ERM problem is not optimal)
Fitting and generalization

- Example: data generated by a function \( t = \sin(2\pi x) \)
- Objective: predict \( t \) from \( x \) with the function considered unknown
Fitting and generalization

« Fitting » of a polynomial of order M

\[ y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]

Least squares criterion

\[ E(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 \]

Linear derivative -> direct solution

[C. Bishop, Pattern recognition and Machine learning, 2006]
How do we choose model complexity?

Underfitting

- $M = 0$
- $M = 1$

Overfitting

- $M = 3$
- $M = 9$

Illustration: model fitting and generalization

[C. Bishop, Pattern recognition and Machine learning, 2006]
Model selection

Separate the data into two parts:
- Training set
- Validation set / hold-out set

Root Mean Square Error (RMS)

\[ E_{\text{RMS}} = \sqrt{2E(w^*)/N} \]
Regularisation

Additional loss terms which add a prior over the parameters

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

Regularisation parameter

\[ \ln\lambda = -18 \]

\[ \ln\lambda = 0 \]

\[ M=9 \]

[C. Bishop, Pattern recognition and Machine learning, 2006]
Big Data!

Overfitting decreases with increasing amount of data

![Figure 1.6](image)

- Plots of the solutions obtained by minimizing the sum-of-squares error function using the polynomial for N = 15 data points (left plot) and N = 100 data points (right plot).

We see that increasing the size of the data set reduces the over-fitting problem.

- Intuitively, what is happening is that the more flexible polynomials with larger values of M are becoming increasingly tuned to the random noise on the target values.

- Another way to say this is that the larger the data set, the more complex (in other words more flexible) the model that we can afford to fit to the data.

- One rough heuristic that is sometimes advocated is that the number of data points should be no less than some multiple (say 5 or 10) of the number of adaptive parameters in the model. However, as we shall see in Chapter 3, the number of parameters is not necessarily the most appropriate measure of model complexity.

- Also, there is something rather unsatisfying about having to limit the number of parameters in a model according to the size of the available training set. It would seem more reasonable to choose the complexity of the model according to the complexity of the problem being solved.

- We shall see that the least squares approach to finding the model parameters represents a specific case of maximum likelihood (discussed in Section 1.2.5), and that the over-fitting problem can be understood as a general property of maximum likelihood.

- By adopting a Bayesian approach, the over-fitting problem can be avoided. We shall see that there is no difficulty from a Bayesian perspective in employing models for which the number of parameters greatly exceeds the number of data points. Indeed, in a Bayesian model the effective number of parameters adapts automatically to the size of the data set.

- For the moment, however, it is instructive to continue with the current approach and to consider how in practice we can apply it to data sets of limited size where we have

\[ M = 9 \]
The standard toolbox

Deep neural networks

Convolutions introduce an inductive bias for imaging/vision applications.

Recurrent networks allow to model sequences

Input  Hidden  Output

\[ \text{LeCun et al., 1998} \]

\[ \text{Hocheiter and Schmidhuber, 1997} \]
« LeNet »

[LeCun et al., 1998]
A linear and shift invariant operator is equivalent to a convolution with the impulse response of the operator!

\[ [\phi(f)](x, y) = \phi \left( \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) \phi^{m,n,p} \right) (x, y) \]

\[ = \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) \phi^{m,n,S(0,0,p)} (x, y) \]

\[ = \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) S(h) (x, y) \]

\[ = \sum_{m=-M/2}^{M/2} \sum_{n=-N/2}^{N/2} f(m, n) h(x - m, y - n) \]

\[ = \sum_{m'=-M/2}^{M/2} \sum_{n'=-N/2}^{N/2} f(x - m', y - n') h(m', n') \]

\[ (m' = x - m \ n' = y - n) \]
The standard toolbox

Deep neural networks

Convolutions introduce an inductive bias for imaging/vision applications.

Recurrent networks allow to model sequences

[LeCun et al., 1998]  [Hocheiter and Schmidhuber, 1997]
Hidden Markov Models

You might be familiar with Hidden Markov Models (HMMs):
Recurrent neural networks (RNNs)

Output layer

Hidden layer

Input layer

Prediction: feed-forward computation in a DAG. No optimization is needed.
Graphical models vs. Neural networks

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

Neural networks:
- state is deterministic
- complex models with componential hidden states
- higher order interactions are easier to handle
- 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?
From RNNs to LSTMs

Objectives:
- handle long and short term transitions
- Deal with vanishing/exploding gradients

Figures: Chris Olah

RNN

LSTM

[Hocheiter and Schmidhuber, 1997]
Functional mappings & Tensor flows

Inputs, outputs and parameters of functional mappings are usually tensors of multiple dimensions.

Example for input tensors in image processing: 4D tensors [index-in-batch, x-spatial, y-spatial, color channel]
Tensorflow (Google): example

We can calculate derivatives of tensors with respect to other tensors:

```
# The input tensors
x = tf.placeholder(tf.float32, [None, 224, 224, 1])

# The target labels
y_gt = tf.placeholder(tf.float32, [None, NO_CLASSES])

# Layer 1: convolutional
W_conv1 = weight_variable([5, 5, 1, 32])
b_conv1 = bias_variable([32])
h_conv1 = tf.nn.relu(tf.nn.conv2d(x, W_conv1, strides=[1, 1, 1, 1], padding='SAME') + b_conv1)
h_pool1 = tf.nn.max_pool(h_conv1, ksize=[1, 2, 2, 1], strides=[1, 2, 2, 1], padding='SAME')

# Layer 2 (output): fully connected
W_fc2 = weight_variable([NO_CLASSES, 1024])
b_fc2 = bias_variable([1024])
y = tf.matmul(h_pool1, W_fc2) + b_fc2

# The loss function
loss = tf.reduce_sum(tf.nn.softmax_cross_entropy_with_logits(logits=y, labels=tf.stop_gradient(y_gt)))
```

We can define a symbolic (!) computation graph:
PyTorch (Facebook) : example

Calculation is imperative, not symbolic. All calculations are carried out immediately. Computation traces are stored to be able to backpropagate later. Easy debugging: no inference engine needed to show values.

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 64, 5, padding=2)
        self.conv2 = nn.Conv2d(64, 64, 5, padding=2)
        self.conv3 = nn.Conv2d(64, 64, 5, padding=2)
        self.conv4 = nn.Conv2d(64, 64, 5, padding=2)
        self.conv5 = nn.Conv2d(64, 64, 5, padding=2)
        self.pool = nn.MaxPool2d(2, 2)
        self.fc1 = nn.Linear(7*7*64, 1024)
        self.d = nn.Dropout(p=0.5)
        self.fc2 = nn.Linear(1024, 10)

    def forward(self, x):
        # Input images are of size 224x224x1
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = self.pool(F.relu(self.conv4(x)))
        x = self.pool(F.relu(self.conv5(x)))

        # Feature maps are now of size 7x7x64
        x = x.view(-1, 7*7*64)
        x = F.relu(self.fc1(x))
        x = self.d(x)
        x = F.relu(self.fc2(x))
        return x
```

output = net.forward(inp)
loss = crossentropy(output, lab_gtv)
loss.backward()
optimizer.step()
May the loss go down
Visualization of learned network features

- Select the strongest activations in the feature map
- Backproject them into the lower layers

[Zeiler and Fergus, ECCV 2014]
Visualization of learned network features

Layer 3

[Zeiler and Fergus, ECCV 2014]
Visualization of learned network features

Layer 2

Layer 3

Layer 4

Layer 5

Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activations in a random subset of feature maps across the validation set, projected down to pixel space using our deconvolutional network approach. Our reconstructions are not samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.

[Zeiler and Fergus, ECCV 2014]
[Yosinski, Clune, Bengio, Lipson, "How transferable are features in deep neural networks?", 2014]
5: Transfer + fine-tuning improves generalization

3: Fine-tuning recovers co-adapted interactions

2: Performance drops due to fragile co-adaptation

4: Performance drops due to representation specificity

Numerical/optimization issues (Positive or negative)

[Yosinski, Clune, Bengio, Lipson, "How transferable are features in deep neural networks?", 2014]
Difficult problems

Example: Visual Question Answering

“What is the moustache made of?”

Huge Models
How can we still train all this?

- Regularization, normalization, tricks
- Data augmentation
- Throw large/insane amounts of data at the problem
  - Simulation, rendering
  - Complex and tiring acquisitions
- Increase the amount of information used from the data
  - Weakly supervised learning
  - Semi supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Create « smarter » models (inductive bias)
- Learn to focus on the relevant parts the data
  - Attention mechanisms
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
Attention in language (google translate)

Figures: Stephen Merity

[Wu et al., arxiv 2016]
Recurrent models of visual attention

[Baradel, Wolf, Mille, Taylor, CVPR 2018]
Continuous automatic authentication on smartphones
Entering PINs on smartphones is painful

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

[Neverova, Wolf, Lacey, Fridmann, Chandra, Barbello, Taylor, IEEE Access 2016]
Data & pre-processing

- 6D data: 3D linear acceleration, 3D angular velocity
- Adding additive and multiplicative noise for obfuscation ("device" vs. "user")
- Add various angles and magnitudes calculated from initial data
- Normalization
- 14D feature vector
Embedded processing: constraints

• No cloud-based processing (inference and model adaptation to a given device is done on the device):
  - limited memory and computational resources,
  - no “negative” data available,
  - all data from the given device is seen as “positive”.

  – Continuous authentication with low latency.
Discriminative vs. Generative solution

- A purely discriminative solution was briefly considered:
  - First offline training of deep neural networks on a large amount of data, multiple classes (=devices/users)
  - Training a new binary classifier (user vs. rest) on the device

- The resources on the device are too limited

- Chosen solution:
  - Discriminative pre-training of features (deep neural networks)
  - Generative biometric framework (Gaussian Mixture Models)
GMM based biometrics

A GMM is learned from the features $y$ extracted on data $x$ using EM:

$$
    y = f(\{x^{(t)}\}) \in \mathbb{R}^N
$$

$$
    p(y|\Theta) = \sum_{i=1}^{M} \pi_i \mathcal{N}(y; \mu_i, \Sigma_i)
$$

A universal background model (UBM) is trained from a large number of devices.

Client models are adapted from the UBM, changing only a subset of parameters.

**Scoring**: thresholding log-likelihood ratio

$$
    \Lambda(Y) = \log p(Y|\Theta_{\text{client}}) - \log p(Y|\Theta_{\text{UBM}}).
$$
Learning data representations

Static ConvNet aggregating temporal statistics

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

### Feature extraction

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy, %</th>
<th># parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST Convnet</td>
<td>37.13</td>
<td>6102137</td>
</tr>
<tr>
<td>LT Convnet</td>
<td>56.46</td>
<td>6102137</td>
</tr>
<tr>
<td>Conv-RNN</td>
<td>64.57</td>
<td>1960295</td>
</tr>
<tr>
<td>Conv-CWRNN</td>
<td>68.83</td>
<td>1964254</td>
</tr>
<tr>
<td>Conv-LSTM</td>
<td>68.92</td>
<td>1965403</td>
</tr>
<tr>
<td><strong>Conv-DCWRNN</strong></td>
<td><strong>69.41</strong></td>
<td><strong>1964254</strong></td>
</tr>
</tbody>
</table>

### Biometric setting

<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>

Upper bounds: thresholds optimized per device/session

Sanity check

- Does the model learn the user “style” of performing tasks or a typical sequence of tasks itself?
- We extracted parts of each session where all users interacted with the same application (a popular mail client, a messenger and a social network application).
- We observed that the results were almost identical to the ones previously obtained on the whole dataset.
- Low correlation with a particular activity.
Conclusion

Application
- The way smartphone users interact with their device IS highly correlated with their identity.
- Future work will combine multiple modalities (inertial, touch etc.)

Deep learning
- We propose a new type of recurrent neural networks which combines the advantages of CWRNNs with shift-invariance and training efficiency.
- Other applications of this model are currently under investigation (gesture recognition, activity recognition etc.).

[Neverova, Wolf, Lacey, Fridman, Chandra, Barbello, Taylor, IEEE Access 2016]
General conclusion

- 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
Team LyonTech : Robocup@Home
Learning from interactions

World

- Evaluation
- Simulator

Agent

- Reinforcement Learning
- Perception/ Detection+Recognition

Actions

- Recognition action
- Motion action
- Reward

Generated input image

Ground truth

Features

World

Présentation du parcours de recherche

Les objets à détecter

Rôle du module

Scènes simulées

Les traitements sur les images

Les scènes simulées - façade

Théo Deprelle

INSA Lyon / Télécommunications
Embodied Question Answering

Requires Competences in Machine Learning, Computer Vision, Natural Language Processing, Robotics