Deep Neural Networks
Applications in Handwriting Recognition
Who am I?

Théodore Bluche  <tb@a2ia.com>

PhD defended at Université Paris-Sud last year

Deep Neural Networks
for Large Vocabulary Handwritten Text Recognition

Now working as a Research Engineer at a2ia in Paris

… automatic document processing (handwriting recognition and more…)
… part of the research team (6 people)
… implementation of new neural networks
… improving the speed and accuracy of production models
… build the models of tomorrow
Handwriting Recognition ...

Goal:

Convert scanned document (image) to text
... is full of challenges
Puzzle

What characters are those?
Answer:

u, (part of) m, en, n! → hard to segment characters, then recognize!
Why is handwriting recognition interesting?

**INPUT**
- 2D
- image → computer vision
- variable-sized

**OUTPUT**
- Sequence
- text → natural language processing
- variable-sized

Suite à mon divorce, je me retrouve dans la
Handwritten Document Processing Pipeline

**Document Layout Analysis**

**Optical Modeling**

**Language Modeling**

**Classification / Keyword Spotting / Named-Entity Extraction / ...**

---

Dear Charlie,

You are cordially invited to the grand opening of my new art gallery titled "The new era of Media Music and paintings" on July 17th, 2012.

P.S: Your presence is obligatory due to your great help of launching my career.

P.S: US presence is obligatory due to your great help of launching my career.

---

Dear Charlie,

You are cordially invited to the grand opening of my new art gallery titled "The new era of Media Music and paintings" on July 17th, 2012.

P.S: Your presence is obligatory due to your great help of launching my career.

P.S: US presence is obligatory due to your great help of launching my career.

---

Dear Charlie,

You are cordially invited to the grand opening of my new art gallery titled "The new era of Media Music and paintings" on July 17th, 2012.

P.S: Your presence is obligatory due to your great help of launching my career.

P.S: US presence is obligatory due to your great help of launching my career.
Outline of this talk

➔ Standard Handwriting Recognition (HWR) System
  ◆ Image processing - Feature Extraction - Optical Model - Hidden Markov Model - Language Model

➔ Deep Neural Networks for HWR -- plugging NNs in the system
  ◆ Neural Nets : Multilayer perceptrons | Recurrent Neural Networks
  ◆ Deep Neural Nets …
  ◆ … automatically learn good features and context

➔ End-to-End HWR -- from pixels to text
  ◆ Multidimensional Recurrent Neural Networks
  ◆ Attention-based methods

➔ Tips and Tricks
Outline of this talk

➔ Standard Handwriting Recognition (HWR) System
   ◆ Image processing - Feature Extraction - Optical Model - Hidden Markov Model - Language Model

➔ Deep Neural Networks for HWR -- plugging NNs in the system
   ◆ Neural Nets : Multilayer perceptrons | Recurrent Neural Networks
   ◆ Deep Neural Nets ...
   ◆ ... automatically learn good features and context

➔ End-to-End HWR -- from pixels to text
   ◆ Multidimensional Recurrent Neural Networks
   ◆ Attention-based methods

➔ Tips and Tricks
Coping with different writing styles

Preprocessing examples:

Slant correction
Contrast enhancement
Height normalization
Modeling ambiguous cursive text

No segmentation
→ model words directly

Explicit segmentation
→ model chars/parts of chars

Delayed segmentation
→ model sequences of observations
The sliding window technique

\[ x_t = [x_{1,t}, \ldots, x_{D,t}] \]

\[ X = X_1 \cdots X_T \]

Recognition

\[ W = w_1 \cdots w_n \]
Features (example)

56 handcrafted features extracted from each frame

→ pixel density measures in the frame and different horizontal regions
→ measures of the center of gravity
→ pixel configuration relative counts
→ pixel density in vertical regions
→ Histogram of Gradients (HoG) in 8 directions
→ ...
A Sequence Modeling problem

Sequence of observations ➔ Optical Model ➔ Sequence of characters
\[ x = x_1 \cdots x_T \]
\[ C = c_1 \cdots c_m \]

Language Model ➔ Sequence of words
\[ W = w_1 \cdots w_n \]

**Optical Model**

- core component of the system
- from pixels / features to characters probabilities
- usually one prediction for each frame / window, and then decoding with a sequence model such as HMM to handle different sequence lengths

**Language Model**

- inclusion of prior knowledge / constraints
- e.g. a vocabulary to allow only character sequences that form known words
- + statistics on large text corpora to promote frequent sequences of words
- nb. in practice we have many char. sequence hypotheses, and the LM weights them
Hidden Markov Models (quickly)

Each character is associated with a small HMM = states and transitions

- transition model = probabilities to go from one state to the other
- each state is associated with a distribution of probabilities over features (optical model here) used to match frames to states

Sequences of characters (e.g. words) are modeled by the concatenation of HMMs
Recognition

Handwriting recognition = find the most likely sequence of words given observations

\[
\mathbf{w}^* = \arg \max_{\mathbf{w}} p(\mathbf{w} | \mathbf{x}) = \arg \max_{\mathbf{w}} \frac{p(\mathbf{x} | \mathbf{w}) P(\mathbf{w})}{p(\mathbf{x})} = \arg \max_{\mathbf{w}} p(\mathbf{x} | \mathbf{w}) P(\mathbf{w})
\]

Optical Model

⇒ in the optical model, words are represented by HMMs (i.e. sequences of states)

\[
p(\mathbf{x} | \mathbf{w}) = \sum_{\mathbf{q} \rightarrow \mathbf{w}} p(\mathbf{x} | \mathbf{q}) = \sum_{\mathbf{q} \rightarrow \mathbf{w}} \prod_t p(x_t | q_t) p(q_t | q_{t-1})
\]

(including Markov assumption and computed efficiently with dynamic programming)

Language Model

⇒ Model the sequence of words (chain-rule)

\[
P(\mathbf{w}) = P(w_1, w_2, \ldots w_N)
\]

= \(P(w_1) P(w_2 | w_1) \ldots P(w_N | w_{N-1} \ldots w_1)\)

n-gram assumption (probas derived from counts in a big textual corpus)

\[
P(w_i | w_{i-1} \ldots w_1) \approx P(w_i | w_{i-1} \ldots w_{i-n+1})
\]
More about HMMs and recognition with LM...

For more details, you may read:


...
**Historical System**

- Input image
- Preprocessing
- Sliding window
- Feature extraction
- Hidden Markov Models
  - Emission model = Gaussian mixtures
  - Transition models = states $\rightarrow$ characters
- Vocabulary
- Language model
State-of-the-art

➔ First steps (preproc, features) =
   ◆ normalize and reduce variability
   ◆ possible loss of relevant information

➔ Last steps (HMM, language model) =
   ◆ add constraints to help correct optical model's mistakes
   ◆ cannot recognize out-of-vocab words, may add mistakes

➔ Optical model = core of the system
   ◆ from image (features) to text (characters, or parts of characters)
   ◆ goal: try to avoid design of good preproc / feature extraction / character models and to rely less on language constraints
      (ultimately, if all characters are well recognized, we wouldn't need an LM)

➔ DEEP NEURAL NETWORKS
Historical System $\rightarrow$ Neural Nets

- Input image
- Preprocessing
- Sliding window
- Feature extraction
- Hidden Markov Models
  - Emission model = Gaussian mixtures
  - Transition models = states $\rightarrow$ characters
- Vocabulary
- Language model

(Deep) Neural Network
Outline of this talk

➔ Standard Handwriting Recognition (HWR) System
   ◆ Image processing – Feature Extraction – Optical Model – Hidden Markov Model – Language Model

➔ Deep Neural Networks for HWR -- plugging NNs in the system
   ◆ Neural Nets : Multilayer perceptrons | Recurrent Neural Networks
   ◆ Deep Neural Nets …
   ◆ … automatically learn good features and context

➔ End-to-End HWR -- from pixels to text
   ◆ Multidimensional Recurrent Neural Networks
   ◆ Attention-based methods

➔ Tips and Tricks
Simple Neuron

Multiply each input value by a weight, sum, apply non-linear function, output new value

\[ x_t = [x_{t,1}, x_{t,2}, \ldots, x_{t,D}] \]
A layer computes a simple function $y_t = f(x_t, \theta)$

... for example:

$y_t = \tanh(W x_t + b); \theta = \{W, b\}$
Handling sequential data

Apply the same layer at each timestep
Neurons/Layers for sequential data

A **recurrent** neuron is just a simple neuron with previous output as additional input.

A **Long Short-Term Memory (LSTM)** neuron also has an *internal state* and *gates* to control the flow of information.

Gates are simple neurons and LSTM may be viewed as a mini-neural net.
Long Short-Term Memory

A very good step-by-step tutorial (from which my diagram are inspired) by Christopher Olah
http://colah.github.io/posts/2015-08-Understanding-LSTMs/  (a MUST-READ!)
Long Short-Term Memory

- The inputs are those of a recurrent neuron (input $x(t) +$ previous output $y(t-1)$)
- The internal state is propagated from the previous timestep
- Three gates with sigmoid (= soft 0/1) activation function to control the flow of information (they are kinds of simple neurons)
  - The forget gate ($f$) controls whether the previous internal state is added to the current state
  - The input gate ($i$) controls whether the input, transformed by a simple neuron ($c$), is added to the current state
  - The output gate ($o$) controls whether the internal state leaves the neuron (after a tanh activation)
- The output ($y(t)$) is tanh of the current state, modulated with the output gate
Layers to Neural Networks

→ A layer outputs a new vector from an input vector.

→ It may be viewed as learnt features

→ It can be used as the input of a neural network

= a neural network is obtained by stacking layers of neurons.

(may be purely feed-forward or recurrent)
Gradient Descent Training

Given a measure of error $E$ on a test set $(x,y)$, find the best parameters (minimizing it):

$$\theta^* = \text{argmin } E(\theta, x, y)$$

i.e.

$$\frac{\partial E}{\partial \theta} = 0$$

Gradient descent:

$$\theta \leftarrow \theta - \eta \frac{\partial E}{\partial \theta}$$

Only requirement: the error measure (or cost function) should be differentiable w.r.t the parameters.
Neural Nets training with Backpropagation

**Backpropagation**: exploit layered network structure to do gradient descent efficiently

\[
\frac{\partial E}{\partial x_t} = \frac{\partial E}{\partial y_t} \frac{\partial y_t}{\partial x_t}
\]

Propagation of the error gradient from one layer to the previous one

\[
\frac{\partial E}{\partial \theta} = \frac{\partial E}{\partial y_t} \frac{\partial y_t}{\partial \theta}
\]

Computation of the gradient w.r.t. the parameters of one layer

= you need to know how to compute

- the *gradient of the cost function* (that you'll minimize) *w.r.t. the outputs of the network*
- for each layer: *the gradient of the output w.r.t. the input and the parameters*

... the rest is only multiplications

(http://arunmallya.github.io/writeups/nn/backprop.html: derivatives for simple costs/layers)
Neural Nets training with Backpropagation

1. Propagate the input forward, layer by layer
2. Compute the error from output and target
3. Compute its gradient w.r.t. the output
4. Propagate the error gradient backward, layer by layer, using chain rule, and compute the gradient w.r.t parameters

Recurrent network are "unfolded" in time so they can be seen as feedforward networks (or directed acyclic graphs)
A word about softmax

→ There are *as many outputs of the network as classes* in the classification problem (e.g. HMM states, characters, ...)

→ Each output represents a score for the corresponding class

→ With a simple *softmax* normalization, they can represent a probability for each class:

\[
p(\text{class}_t = k \mid x_t) = \frac{e^{y_{t,k}}}{\sum_n e^{y_{t,n}}}
\]

→ Hence a cost function can be devised so as to maximize the probability of the correct class (and this cost is easy to differentiate w.r.t. the outputs of the network)
Frame classification (MLP style)

➔ Input = one frame = one vector of pixel or feature values
➔ Output = posterior probabilities over HMM states (or sometimes characters)

\[
\begin{pmatrix}
\vdots \\
 m_2 \\
\vdots \\
 g_1 \\
\vdots \\
 \hat{e}_3 \\
\vdots \\
 g_2 \\
\end{pmatrix}, \quad \begin{pmatrix}
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\vdots \\
\end{pmatrix}
\]

Training:

➔ Collect a dataset of \((x_t, q_t)\) = frames with correct HMM state
➔ Minimize \(- \log p( q_t | x_t )\)
➔ Measure the Frame Error Rate (\% of frames with wrong HMM state prediction)
### Sequence classification (RNN style)

**Option 1**

- Same as MLP except hidden layers depend on the values at (t-1) or (t+1)
- i.e. HMM states or characters are predicted potentially taking into account larger context
- Can follow the same training method for each t
Sequence classification (RNN style)

Option 2 : CTC

➔ To train the network **directly with** frame sequences and **character sequences**
➔ i.e. no need to label each frame with an HMM state

Minimize:

\[-\log p (c_1, c_2, \ldots, c_N | x = x_1, x_2, \ldots, x_T)\]

➔ Measure the **Character Error Rate** (% of character substitutions, deletions or insertions)

Sequence sizes are not equal !!!
Connectionist Temporal Classification (CTC)

→ The network **outputs characters** (not HMM states)

→ **Problem** \( T \) items in the output sequence, \( N \) items in the target char sequence

→ Make sure that \( T > N \) and **define a simple mapping** of sequences that removes duplicates:

\[
\begin{align*}
\text{AAABBCCCC} & \rightarrow \text{ABC} \\
\text{ABBBBBCCCC} & \rightarrow \text{ABC} \\
& \cdots \\
\text{AAAAABCCCC} & \rightarrow \text{ABC}
\end{align*}
\]

\[
p(c_1 \ldots c_N | \mathbf{x}) = \sum_{y_1 \ldots y_T \rightarrow c_1 \ldots c_N} p(y_1 \ldots y_T | \mathbf{x}) \\
= \sum_{y_1 \ldots y_T \rightarrow c_1 \ldots c_N} \prod_t p(y_t | \mathbf{x})
\]

→ Computed efficiently with **dynamic programing**

→ **Problem** how to output \( \text{ABB} \) ( \( \text{AAABBBBBBB} \rightarrow \text{AB} \) )?
Connectionist Temporal Classification (CTC)

→ **Problem** how to output $ABB$ ($AAABBBBBB \rightarrow AB$) ?

→ The network outputs characters + a special **NULL** (or blank or non-char) symbol –

→ The mapping **removes** duplicates, and then **NULLs**

<table>
<thead>
<tr>
<th>String</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AAABBBBBBB$</td>
<td>$AB$</td>
</tr>
<tr>
<td>$AAABBBBBCC$</td>
<td>$ABC$</td>
</tr>
<tr>
<td>$AA-BB--C-$</td>
<td>$A-B-C-$</td>
</tr>
<tr>
<td>$-A--B--C-$</td>
<td>$A-B-C-$</td>
</tr>
<tr>
<td>$AAAABBBBBB$</td>
<td>$AB$</td>
</tr>
<tr>
<td>$AA-BB--B-$</td>
<td>$A-B-B-$</td>
</tr>
<tr>
<td>$-A--B--B-$</td>
<td>$A-B-B-$</td>
</tr>
<tr>
<td>$AAAABBBBBB$</td>
<td>$AB$</td>
</tr>
<tr>
<td>$AA-BB--B-$</td>
<td>$A-B-B-$</td>
</tr>
<tr>
<td>$-A--B--B-$</td>
<td>$A-B-B-$</td>
</tr>
</tbody>
</table>
**Historical System → Neural Nets**

- Input image
- Preprocessing
- Sliding window
- Feature extraction
- Hidden Markov Models
  - Emission model = Gaussian mixtures
  - Transition models = states → characters
- Vocabulary
- Language model

(Deep) Neural Network
Standard GMM → Neural Net → Deep Neural Net

→ Big improvement by using neural nets instead of GMMs

→ Similar big improvement by using deep neural nets instead of shallow neural nets
Impact of the net's depth

MLP

RNN
Impact of the net’s depth

→ At constant number of free parameters in the models, deeper nets give better results
Features vs. Pixels

Input vector = all the raw pixel values in the window flattened as a single vector of WxH dimensions
Historical System $\rightarrow$ Neural Nets

- Input image
- Preprocessing
- Sliding window
- Feature extraction
- Hidden Markov Models
  - Emission model = Gaussian mixtures
  - Transition models $\rightarrow$ states $\rightarrow$ characters
- Vocabulary
- Language model

(Deep) Neural Network
Outline of this talk

➔ Standard Handwriting Recognition (HWR) System
  ◆ Image processing - Feature Extraction - Optical Model - Hidden Markov Model - Language Model

➔ Deep Neural Networks for HWR -- plugging NNs in the system
  ◆ Neural Nets : Multilayer perceptrons | Recurrent Neural Networks
  ◆ Deep Neural Nets …
  ◆ … automatically learn good features and context

➔ End-to-End HWR -- from pixels to text
  ◆ Multidimensional Recurrent Neural Networks
  ◆ Attention-based methods

➔ Tips and Tricks
Neural Networks for Images (pixel level)

→ Instead of a feature vector, the **input is only one pixel value** (or a vector of 3 RGB values for color images)

→ The network is **replicated** at each position in the image
Feature Maps

→ The outputs of one hidden layer for a pixel may be viewed as new "pixel" values, defining new channels

→ Since the network is replicated, each output have a similar meaning across all pixels (but different values)

→ So a given output across the whole image defines a new (kind of) image: a feature map

in the end, it's just a way of representing or interpreting the net...
**e.g. Convolutional Neural Network**

→ We can include spatial (structured) context:

instead of giving 1 pixel value at the current position, we give the values of all pixels in a given neighborhood

→ This is still replicated at all positions = convolution, with kernel defined by the weights

→ You can reduce the size of the feature maps by replicating the net every $N$ positions (output will be $N$ times smaller)

(nb: also possible to have convolution in sequential nets... )
e.g. Multi-Dimensional Recurrent Neural Networks

→ As for sequences, you can make the network recurrent

the input at a given position includes the outputs of the same layer at neighbors
Multidimensional RNN

→ **MD Recurrent + Convolutional** layers

→ applied directly to the pixel of the raw text line image

→ A special **Collapse** layer on top to get sequential representation

→ Trained with CTC to output character sequences

**Current State-of-the-art!**
**Historical System → Neural Nets**

- Input image
- Preprocessing
- Sliding window
- Feature extraction

- Hidden Markov Models
  - Emission model = Gaussian mixtures
  - Transition models = states → characters

- Vocabulary
- Language model

(Deep) Neural Network
What happens in the net? (bottom)

MDLSTM (4 directions)

Convolutions

Sum + tanh

MDLSTM (4 directions)

Simple features (like oriented edges, ...)

Image (269 x 61)

[...]
What happens in the net? (middle)

Convolutions

MDLSTM (4 directions)

Complex features (like loops, ascenders, vertical strokes, …)
What happens in the net? (top)

More abstract features (combination of features, closer to character level...)

MDLSTM (4 directions)

Collapse

Softmax
Historical System → Neural Nets

- Input image
- Preprocessing
- Sliding window
- Feature extraction
- Hidden Markov Models
  - Emission model = Gaussian mixtures
  - Transition models = states → characters
- Vocabulary
- Language model

In several setups, we even see that a vocabulary does not help (because of high out-of-vocabulary words rate) → right now, hybrid word/character language models are best...
Attention Neural Network

→ An **Attention Neural Network** predicts where to look next

→ = a distribution of probability over positions in the feature maps

→ the sum of the feature vectors across all positions in the maps, **weighted by the attention network output** is fed to a neural network which recognize the next character

= attention net + decoder is applied $N$ times
A guard reported that at East Croydon he had seen what was accepted as the same couple sitting close together in a first-class compartment of the train from London Bridge of which he was in charge. The two could have joined this train by taking one from Victoria and changing at East Croydon. He also believed that they had still been together at South Croydon, and he remembered

J'ai hérité d'une somme de 3000 euros la semaine dernière et j'ai décidé de procéder à une commande d'actions boursière pour un montant de 1500 euros. Étant donné que vous êtes mon banquier depuis 10 ans maintenant je vous fais confiance quant au choix du placement.

Je vous prie d'agréer Monsieur, l'expression de mes sentiments distingués.

with that of the suitability of round for spg congestion... due to the light pressure is inevitably...
Historical System → Neural Nets

- Input line
- Preprocessing
- Sliding window
- Feature extraction
- Hidden Markov Models
  - Emission model = Gaussian mixtures
  - Transition models = states → characters
- Vocabulary
- Language model

(Deep) Neural Network
Outline of this talk

➔ Standard Handwriting Recognition (HWR) System
   ◆ Image processing – Feature Extraction – Optical Model – Hidden Markov Model – Language Model

➔ Deep Neural Networks for HWR -- plugging NNs in the system
   ◆ Neural Nets: Multilayer perceptrons | Recurrent Neural Networks
   ◆ Deep Neural Nets ...
   ◆ ... automatically learn good features and context

➔ End-to-End HWR -- from pixels to text
   ◆ Multidimensional Recurrent Neural Networks
   ◆ Attention-based methods

➔ Tips and Tricks
Tips & Tricks

→ The theory is pretty simple (linear algebra, simple derivatives, basic probabilities, standard numerical optimization)

→ The practice is a mess ...

- optimization theory is solid for convex problems, deep learning is highly non-convex
- gradient descent is theoretically sound when applied to the whole training set at once, but we do it example-wise (stochastic GD), and want to minimize the cost while preserving generalization = non standard optimization
- The learning rate is probably the most important parameter to tune
Tips & Tricks – Optimization

→ The first step is to define a good cost function = what you want to minimize = should represent your problem

→ The dynamics of training is quite important (even when it should work, it does not always). The non-linearities and values of the weights will play a role

- a good initialization of the weights is often crucial (a simple random init. is rarely sufficient, there are rules of thumb for good initialization; when possible, initialize the weights with those of a net already trained for another task)
- Regularization (weight decay, dropout, …) is especially important with deep neural nets with a lot of parameters
- Plain SGD can be improved (e.g. look for momentum, ADAGRAD, etc.)
Tips & Tricks – Training

→ Deep learning solves complicated problems, but with *complicated models*

- Check first if a simple model is not sufficient
- Complicated model are complicated to train: think *curriculum* = start simple and increase complexity
- Most methods are gradient-based. Everybody makes mistakes. When implementing neural nets, always *check your gradients* are right (remember the definition of a derivative)
- The *devil is in the details*: when you try to implement something you read in a paper, pay attention to every details (of the net, data, optimization, etc.) and remember that author do not always tell them... (not as easy as it seems)
- The answer to “is it a good model for my problem?” is often “try!”
Tips & Tricks – Deep Learning

It is not magic!! (although it often looks like it)

→ It is a lot of parameters = a lot of data needed to adjust them + a good implementation to do it fast + good initialization / formulation of the problem / optimization method.
   (if not enough data, don't expect miracle and spend time for preproc/feature extraction OR data augmentation)

→ It is maths + a lot of “cooking”: knowing about pastas, tomatoes and beef is not enough to make a good bolognese, you should also learn the good recipes!

→ A lot of intuition, understanding and good ideas will come with experience (and vice-versa). Play with models and problems, you'll end up having a sense of want could work and what won't…

→ … but question what/why you are doing, don't just download ML libraries to run experiments
To conclude...

→ Historically, HWR was: preprocessing, feature extraction, Gaussian HMMs, Language model
→ With deep neural networks, we can recognize character sequences from the raw image

- with enough data, the preproc is less useful and we avoid loss of information
- same with features, which are learnt and task-specific
- same with character modeling: we can output character sequences directly

→ NB: data/preproc/feature engineering disapear, but now: model engineering
→ NB: you need a lot of data

→ Historically, HWR was: recognition of chars, then words, then lines: moving toward recognition of paragraphs and full pages

→ Note: also, deep neural nets for layout analysis, language models, ...
Thanks for your attention

Théodore Bluche
tb@a2ia.com
(do not hesitate to reach me if you have questions)

A few refs...


