An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio
Université de Montréal

13/06/2007
Started in 2005, for a total of 58 talks + 2 “Best-of NIPS” sessions.

You can find most of the past talk slides and information about upcoming talks on the web page

Bilingual page :
- “McGill-UdeM-MITACS Machine Learning Seminars” in english
- “Séminaires UdeM-McGill-MITACS d’Apprentissage Automatique” in french

Google keywords :
- “mitacs machine learning”
- “active learning in POMDP”
- “echo state network apprentissage”
- “Aaron Courville”
- “aaron courville site :umontreal.ca”
- “indiansumit chopra”
- “an efficient method for gradient based applications SVM models”
Started in 2005, for a total of 58 talks + 2 “Best-of NIPS” sessions.

You can find most of the past talk slides and information about upcoming talks on the web page

Bilingual page:
- “McGill-UdeM-MITACS Machine Learning Seminars” in english
- “Séminaires UdeM-McGill-MITACS d’Apprentissage Automatique” in french

Google keywords:
- “mitacs machine learning”
- “active learning in POMDP”
- “echo state network apprentissage”
- “Aaron Courville”
- “aaron courville site :umontreal.ca”
- “indiansumit chopra”
- “an efficient method for gradient based applications SVM models”
Started in 2005, for a total of 58 talks + 2 “Best-of NIPS” sessions.

You can find most of the past talk slides and information about upcoming talks on the web page.

Bilingual page:
- “McGill-UdeM-MITACS Machine Learning Seminars” in english
- “Séminaires UdeM-McGill-MITACS d’Apprentissage Automatique” in french

Google keywords:
- “mitacs machine learning”
- “active learning in POMDP”
- “echo state network apprentissage”
- “Aaron Courville”
- “aaron courville site :umontreal.ca”
- “indiansumit chopra”
- “an efficient method for gradient based applications SVM models”
Little known facts about the MITACS seminar

- Started in 2005, for a total of 58 talks + 2 “Best-of NIPS” sessions.
- You can find most of the past talk slides and information about upcoming talks on the web page.

Bilingual page:
- “McGill-UdeM-MITACS Machine Learning Seminars” in English
- “Séminaires UdeM-McGill-MITACS d’Apprentissage Automatique” in French

Google keywords:
- “mitacs machine learning”
- “active learning in POMDP”
- “echo state network apprentissage”
- “Aaron Courville”
- “aaron courville site :umontreal.ca”
- “indiansumit chopra”
- “an efficient method for gradient based applications SVM models”
Recently, models relying on deep architectures have been proposed (Deep Belief Networks).

- Their performance compare favorably to state of art models such as Support Vector Machines.
- They have been tested on relatively few and simple problems.
- We propose to evaluate them on problems with many factors of variation.
Recently, models relying on deep architectures have been proposed (Deep Belief Networks)

Their performance compare favorably to state of art models such as Support Vector Machines

They have been tested on relatively few and simple problems

We propose to evaluate them on problems with many factors of variation
Recently, models relying on deep architectures have been proposed (Deep Belief Networks)

Their performance compare favorably to state of art models such as Support Vector Machines

They have been tested on relatively few and simple problems

We propose to evaluate them on problems with many factors of variation
Recently, models relying on deep architectures have been proposed (Deep Belief Networks).

Their performance compare favorably to state of art models such as Support Vector Machines.

They have been tested on relatively few and simple problems.

We propose to evaluate them on problems with many factors of variation.
We focus here on problems were the input distribution has the following structure:

\[ p(x) = \sum_{\phi_1, \ldots, \phi_m} p(x|\phi_1, \ldots, \phi_m)p(\phi_1, \ldots, \phi_m) \]

where \( p(\phi_1, \ldots, \phi_m) \) is high for (exponentially) many combinations of values of the factors of variation \( \phi_i \).

Problems with such a structure:

- digit recognition (vision): \( \phi_i \in \{\)rotation angle, scaling, background, etc.\}\)
- document classification (NLP): \( \phi_i \in \{\)topics, style, etc.\}\)
- speech recognition (signal processing): \( \phi_i \in \{\)speaker gender, background noise, environment echo, etc.\}\)

We will focus on vision problems.

We want to avoid hand-engineered solutions to these problems.
We focus here on problems where the input distribution has the following structure:

\[ p(x) = \sum_{\phi_1, \ldots, \phi_m} p(x | \phi_1, \ldots, \phi_m) p(\phi_1, \ldots, \phi_m) \]

where \( p(\phi_1, \ldots, \phi_m) \) is high for (exponentially) many combinations of values of the factors of variation \( \phi_i \).

Problems with such a structure:
- digit recognition (vision): \( \phi_i \in \{\text{rotation angle, scaling, background, etc.}\} \)
- document classification (NLP): \( \phi_i \in \{\text{topics, style, etc.}\} \)
- speech recognition (signal processing): \( \phi_i \in \{\text{speaker gender, background noise, environment echo, etc.}\} \)

We will focus on vision problems.

We want to avoid hand-engineered solutions to these problems.
Problems with Many Factors of Variation

- We focus here on problems were the input distribution has the following structure

\[ p(x) = \sum_{\phi_1, \ldots, \phi_m} p(x|\phi_1, \ldots, \phi_m)p(\phi_1, \ldots, \phi_m) \]

where \( p(\phi_1, \ldots, \phi_m) \) is high for (exponentially) many combinations of values of the factors of variation \( \phi_i \)

- Problems with such a structure:
  - digit recognition (vision): \( \phi_i \in \{ \text{rotation angle, scaling, background, etc.} \} \)
  - document classification (NLP): \( \phi_i \in \{ \text{topics, style, etc.} \} \)
  - speech recognition (signal processing): \( \phi_i \in \{ \text{speaker gender, background noise, environment echo, etc.} \} \)

- We will focus on vision problems

- We want to avoid hand-engineered solutions to these problems
A shallow model is a model with very few layers of computational units:

To approximate a complex function, such models will need large (exponential) number of computational units.
A shallow model is a model with very few layers of computational units:

- One hidden layer neural network
- Kernel SVM

To approximate a complex function, such models will need large (exponential) number of computational units.
A *deep architecture model* is such that its output is the result of the composition of many computational units.

Many layers potentially yield highly complex functions with a limited number of parameters.

The $d$ dimensional parity function modeled with
- $O(d2^d)$ parameters with Gaussian SVM
- $O(d^2)$ parameters with a $O(\log_2 d)$ hidden layer neural network.
A deep architecture model is such that its output is the result of the composition of many computational units.

Many layers potentially yield highly complex functions with a limited number of parameters.

The $d$ dimensional parity function modeled with:
- $O(d2^d)$ parameters with Gaussian SVM
- $O(d^2)$ parameters with a $O(\log_2 d)$ hidden layer neural network.
(Hinton et al., 2006) introduced the Deep Belief Network (DBN), a deep probabilistic neural network.

The training procedure is first **layer-wise greedy and unsupervised**

Then the output of the model is **fine-tuned** on the supervised data

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(y_i|x_i, \theta)$$
Learning Deep Architecture Models (DBN)

- (Hinton et al., 2006) introduced the Deep Belief Network (DBN), a deep probabilistic neural network

- The training procedure is first layer-wise greedy and unsupervised

- Then the output of the model is fine-tuned on the supervised data

\[
\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(y_i|x_i, \theta)
\]
Greedy Module : Restricted Boltzmann Machines

- RBM is an energy-based generative model:

\[ p(X = x, H = h) \propto e^{-\mathcal{E}(x, h)} = e^{x'b + h'c + h'Wx} \]

- **Inference is easy**: \( p(H|X) \) factorizes \( (H_i \perp H_j|X, \quad i \neq j) \):

\[ p(X^i = 1|H = h) = \text{sigm}(b^i + \sum_i h^i W^{ij}) \]
\[ p(H^i = 1|X = x) = \text{sigm}(c^i + \sum_j W^{ij} x^j). \]

⇒ no explaining away

- **Training is easy**: Contrastive Divergence (Hinton 2002)

\[
\frac{\partial \log p(x)}{\partial \theta} = - \sum_{h'} p(h'|x) \frac{\partial \mathcal{E}(x, h)}{\partial \theta} + \sum_{x', h'} p(x', h') \frac{\partial \mathcal{E}(x', h')}{\partial \theta}
\]

\[ \mathcal{E}(x, h) = e^{x'b + h'c + h'Wx} - \ln(\sum_{x'} e^{x''b + h''c + h''Wx'}) \]

\[ \mathcal{E}(x', h') = \ln(\sum_{x'} e^{x''b + h''c + h''Wx'}) \]
An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation
Shallow and Deep Architecture Models

Computing $\hat{p}(y|x)$

\[ \hat{h}_1 = p(h_1|x) = \text{sigmoid}(c_1 + W_1 x) \]
Computing $\hat{p}(y|x)$

$$\hat{h}_2 = p(h_2|h_1) = \text{sigm}(c_2 + W_2 \hat{h}_1)$$
An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation
Shallow and Deep Architecture Models

Computing $\hat{p}(y|x)$

$$\hat{p}(y|x) = \sum_{h_3} p(y, h^3 | h_2)$$

$$\propto e^{c'_y} \prod_k \left( 1 + e^{V_k y + W^k_3 h_2 + c^k} \right)$$
Learning Deep Architecture Models (SAA)

- Instead of stacking RBMs, we can have Stacked Autoassociators (SAA)
- The training procedure is first **layer-wise greedy** and **unsupervised**

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(y_i|x_i, \theta)$$

- Then the output of the model is **fine-tuned** on the supervised data
Learning Deep Architecture Models (SAA)

- Instead of stacking RBMs, we can have Stacked Autoassociators (SAA)
- The training procedure is first \textit{layer-wise greedy} and \textit{unsupervised}

Then the output of the model is \textit{fine-tuned} on the supervised data

\[
\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} \log \hat{p}(y_i|x_i, \theta)
\]
Autoassociator is a neural network trained to reconstruct its input:

$$\hat{x} = \text{sigm}(c + W' \text{sigm}(b + Wx))$$

The reconstruction error is

$$R(x, \hat{x}) = -\sum_i x^i \log \hat{x}^i + (1-x^i) \log(1-\hat{x}^i)$$

The neural network is trained using a gradient descent algorithm.
Experimental setup

- We report results for the following models:
  - Support Vector Machine classifiers with polynomial ($\text{SVM}_{\text{poly}}$) and Gaussian ($\text{SVM}_{\text{rbf}}$) kernels
  - One hidden layer neural network (NNet)
  - Deep Belief Network (DBN-1 and DBN-3) and Stacked Autoassociator (SAA-3) with one or three hidden layers

- The validation set was used to do model selection and early stopping

- $\text{SVM}_{\text{poly}}$ and $\text{SVM}_{\text{rbf}}$ were retrained on the union of the training and validation set
Dataset Characteristics Summary

- Classification datasets on $28 \times 28$ pixel images
- All datasets have a training, validation and test split
- Training set size varies from 1000 to 10000 samples
- Validation set size varies from 200 to 2000 samples
- All datasets have a test set of size 50000.
Variations on Digit Recognition

- MNIST dataset with additional factors of variation
  1. Pick sample \((x, y) \in \mathcal{X}\) from the digit recognition dataset;
  2. Create a perturbed version \(x^*\) of \(x\) according to some factors of variation;
  3. Add \((x^*, y)\) to a new dataset \(\mathcal{X}^*\);
  4. Go back to 1 until enough samples are generated.

We generated the following datasets:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Additional factors of variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>mnist-rot</td>
<td>rotation angle between 0 and (2\pi) radians</td>
</tr>
<tr>
<td>mnist-back-rand</td>
<td>random background pixels between 0 and 255</td>
</tr>
<tr>
<td>mnist-back-image</td>
<td>random patch from 20 black and white images</td>
</tr>
<tr>
<td>mnist-rot-back-image</td>
<td>factors of (mnist-rot) and (mnist-back-image)</td>
</tr>
</tbody>
</table>
An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation

Benchmark Problems

Variations on Digit Recognition (samples)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM$_{rbf}$</th>
<th>SVM$_{poly}$</th>
<th>NNet</th>
<th>DBN-1</th>
<th>SAA-3</th>
<th>DBN-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>mnist-basic</td>
<td>3.03</td>
<td>3.69</td>
<td>4.69</td>
<td>3.94</td>
<td>3.46</td>
<td>3.11</td>
</tr>
<tr>
<td>mnist-rot</td>
<td>10.38</td>
<td>13.61</td>
<td>17.62</td>
<td>12.11</td>
<td>11.43</td>
<td>12.30</td>
</tr>
<tr>
<td>mnist-back-image</td>
<td>22.61</td>
<td>24.01</td>
<td>27.41</td>
<td>16.15</td>
<td>23.00</td>
<td>16.31</td>
</tr>
<tr>
<td>mnist-rot-back-image</td>
<td>32.62</td>
<td>37.59</td>
<td>42.17</td>
<td>31.84</td>
<td>24.09</td>
<td>28.51</td>
</tr>
</tbody>
</table>
An Empirical Evaluation of Deep Architectures on Problems with Many Factors of Variation
Benchmark Problems

Discrimination between Tall and Wide Rectangles

- rectangles: the pixels corresponding to the border of the rectangle has a value of 255, 0 otherwise

- rectangles-image: the border and inside of the rectangles correspond to an image patch. A background patch is also sampled

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM(_{\text{rbf}})</th>
<th>SVM(_{\text{poly}})</th>
<th>NNet</th>
<th>DBN-1</th>
<th>SAA-3</th>
<th>DBN-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>rectangles</td>
<td>2.15</td>
<td>2.15</td>
<td>7.16</td>
<td>4.71</td>
<td>2.41</td>
<td>2.60</td>
</tr>
<tr>
<td>rectangles-image</td>
<td>24.04</td>
<td>24.05</td>
<td>33.20</td>
<td>23.69</td>
<td>24.05</td>
<td>22.50</td>
</tr>
</tbody>
</table>
Recognition of Convex Sets

- *convex* contains images corresponding to convex and non convex sets of pixels

The convex sets are intersections of random half-planes

The non convex sets correspond to the union of a random number of convex sets, failing a convexity test

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM\textsubscript{rbf}</th>
<th>SVM\textsubscript{poly}</th>
<th>NNet</th>
<th>DBN-1</th>
<th>SAA-3</th>
<th>DBN-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>convex</td>
<td>19.13</td>
<td>19.82</td>
<td>32.25</td>
<td>19.92</td>
<td><strong>18.41</strong></td>
<td><strong>18.63</strong></td>
</tr>
</tbody>
</table>
## Results Summary

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM(_{rbf})</th>
<th>SVM(_{poly})</th>
<th>NNet</th>
<th>DBN-1</th>
<th>SAA-3</th>
<th>DBN-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>mnist-basic</td>
<td>3.03</td>
<td>3.69</td>
<td>4.69</td>
<td>3.94</td>
<td>3.46</td>
<td>3.11</td>
</tr>
<tr>
<td>mnist-rot</td>
<td>10.38</td>
<td>13.61</td>
<td>17.62</td>
<td>12.11</td>
<td>11.43</td>
<td>12.30</td>
</tr>
<tr>
<td>mnist-back-rand</td>
<td>14.58</td>
<td>16.62</td>
<td>20.04</td>
<td>9.80</td>
<td>11.28</td>
<td><strong>6.73</strong></td>
</tr>
<tr>
<td>mnist-back-image</td>
<td>22.61</td>
<td>24.01</td>
<td>27.41</td>
<td><strong>16.15</strong></td>
<td>23.00</td>
<td><strong>16.31</strong></td>
</tr>
<tr>
<td>mnist-rot-back-image</td>
<td>32.62</td>
<td>37.59</td>
<td>42.17</td>
<td>31.84</td>
<td><strong>24.09</strong></td>
<td>28.51</td>
</tr>
<tr>
<td>rectangles</td>
<td><strong>2.15</strong></td>
<td><strong>2.15</strong></td>
<td>7.16</td>
<td>4.71</td>
<td><strong>2.41</strong></td>
<td>2.60</td>
</tr>
<tr>
<td>rectangles-image</td>
<td>24.04</td>
<td>24.05</td>
<td>33.20</td>
<td>23.69</td>
<td>24.05</td>
<td><strong>22.50</strong></td>
</tr>
<tr>
<td>convex</td>
<td>19.13</td>
<td>19.82</td>
<td>32.25</td>
<td>19.92</td>
<td><strong>18.41</strong></td>
<td><strong>18.63</strong></td>
</tr>
</tbody>
</table>

*TAB.:* Results on the benchmark for problems with factors of variation (in percentages). The best performance as well as those with overlapping confidence intervals are marked in bold.
Investigation of the “background effect”

**Figure:** Classification error of SVM$_{rbf}$, SAA-3 and DBN-3 on MNIST examples with progressively less pixel correlation in the background.
Conlusion

- In general, models with deep architectures either perform as well or outperform other models.

- There are still challenges in scaling the current algorithms to problems with very complex input distribution.

- Datasets and experimental details can be found on our public wiki page:
  
  http://www.iro.umontreal.ca/~lisa/ptwiki/
Future Work

- Address the “focus problem” of greedy layer-wise unsupervised training
- Develop learning algorithms that make better use of the capacity of the model or models appropriate for more complex input distributions
- Develop models and algorithms with less hyper-parameters
THANK YOU!