Autoencoders, denoising autoencoders, and learning deep networks

Pascal Vincent

Part II
joint work with
Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol, Isabelle Lajoie

Laboratoire
d’Informatique
des Systèmes
d’Apprentissage
www.iro.umontreal.ca/~lisa
Denoising Autoencoders for learning Deep Networks

For more details, see:


Also some recent results were produced by Isabelle Lajoie.
Building good predictors on complex domains means learning complicated functions.

These are best represented by multiple levels of non-linear operations i.e. deep architectures.

Deep architectures are an old idea: multi-layer perceptrons.

Learning the parameters of deep architectures proved to be challenging!
Training deep architectures: attempted solutions

- **Solution 1**: initialize at random, and
do gradient descent (Rumelhart, Hinton and Williams, 1986).
  → disappointing performance. Stuck in poor solutions.

- **Solution 2**: Deep Belief Nets (Hinton, Osindero and Teh, 2006): initialize by
  stacking Restricted Boltzmann Machines, fine-tune with Up-Down.
  → impressive performance.

Key seems to be good unsupervised layer-by-layer initialization.

- **Solution 3**: initialize by stacking autoencoders, fine-tune with
  gradient descent. (Bengio et al., 2007; Ranzato et al., 2007)
  → Simple generic procedure, no sampling required.

  Performance almost as good as Solution 2

  ...but not quite. Can we do better?

- **Non-convex optimization**
  → local minima: solution depends on where you start...
Training deep architectures: attempted solutions

- **Solution 1**: initialize *at random*, and do gradient descent (Rumelhart, Hinton and Williams, 1986).
  → disappointing performance. Stuck in poor solutions.

- **Solution 2**: Deep Belief Nets (Hinton, Osindero and Teh, 2006): initialize by stacking Restricted Boltzmann Machines, fine-tune with Up-Down.
  → impressive performance.

Key seems to be good unsupervised layer-by-layer initialization...

- **Solution 3**: initialize by stacking autoencoders, fine-tune with gradient descent. (Bengio et al., 2007; Ranzato et al., 2007)
  → Simple generic procedure, no sampling required.
  Performance almost as good as Solution 2

...but not quite. Can we do better?

- **Non-convex optimization**
  → local minima: solution depends on where you start...
Training deep architectures: attempted solutions

- **Solution 1**: initialize at random, and do gradient descent (Rumelhart, Hinton and Williams, 1986).
  → disappointing performance. Stuck in poor solutions.

- **Solution 2**: Deep Belief Nets (Hinton, Osindero and Teh, 2006): initialize by stacking Restricted Boltzmann Machines, fine-tune with Up-Down.
  → impressive performance.

Key seems to be good unsupervised layer-by-layer initialization...

- **Solution 3**: initialize by stacking autoencoders, fine-tune with gradient descent. (Bengio et al., 2007; Ranzato et al., 2007)
  → Simple generic procedure, no sampling required.
  Performance almost as good as Solution 2

...but not quite. Can we do better?

- Non-convex optimization
  → local minima: solution depends on where you start...
Training deep architectures: attempted solutions

- **Solution 1**: initialize at random, and do gradient descent (Rumelhart, Hinton and Williams, 1986).
  → disappointing performance. Stuck in poor solutions.

- **Solution 2**: Deep Belief Nets (Hinton, Osindero and Teh, 2006): initialize by stacking Restricted Boltzmann Machines, fine-tune with Up-Down.
  → impressive performance.

Key seems to be good unsupervised layer-by-layer initialization.

- **Solution 3**: initialize by stacking autoencoders, fine-tune with gradient descent. (Bengio et al., 2007; Ranzato et al., 2007)
  → Simple generic procedure, no sampling required.
  Performance almost as good as Solution 2

...but not quite. Can we do better?

- Non-convex optimization
  ➔ local minima: solution depends on where you start...
Can we do better?

Open question: what would make a good unsupervised criterion for finding good initial intermediate representations?

- Inspiration: our ability to “fill-in-the-blanks” in sensory input. missing pixels, small occlusions, image from sound, . . .

- Good fill-in-the-blanks performance $\leftrightarrow$ distribution is well captured.

- $\rightarrow$ old notion of associative memory (motivated Hopfield models (Hopfield, 1982))

What we propose: unsupervised initialization by explicit fill-in-the-blanks training.
The denoising autoencoder

- Clean input $x \in [0, 1]^d$ is partially destroyed, yielding corrupted input: $\tilde{x} \sim q_D(\tilde{x}|x)$.
- $\tilde{x}$ is mapped to hidden representation $y = f_\theta(\tilde{x})$.
- From $y$ we reconstruct a $z = g_{\theta'}(y)$.
- Train parameters to minimize the cross-entropy "reconstruction error" $L_H(x, z) = H(B_x || B_z)$, where $B_x$ denotes multivariate Bernoulli distribution with parameter $x$. 
The denoising autoencoder

- Clean input $\mathbf{x} \in [0, 1]^d$ is partially destroyed, yielding corrupted input: $\tilde{\mathbf{x}} \sim q_D(\tilde{\mathbf{x}} | \mathbf{x})$.

- $\tilde{\mathbf{x}}$ is mapped to hidden representation $\mathbf{y} = f_\theta(\tilde{\mathbf{x}})$.

- From $\mathbf{y}$ we reconstruct a $\mathbf{z} = g_{\theta'}(\mathbf{y})$.

- Train parameters to minimize the cross-entropy “reconstruction error” $L_H(\mathbf{x}, \mathbf{z}) = H(B_\mathbf{x} \| B_\mathbf{z})$, where $B_\mathbf{x}$ denotes multivariate Bernoulli distribution with parameter $\mathbf{x}$. 
The denoising autoencoder

- Clean input \( x \in [0, 1]^d \) is partially destroyed, yielding corrupted input: \( \tilde{x} \sim q_D(\tilde{x}|x) \).
- \( \tilde{x} \) is mapped to hidden representation \( y = f_\theta(\tilde{x}) \).

- From \( y \) we reconstruct \( z = g_{\theta'}(y) \).
- Train parameters to minimize the cross-entropy “reconstruction error” \( L_H(x, z) = \text{H}(\mathcal{B}_x \| \mathcal{B}_z) \), where \( \mathcal{B}_x \) denotes multivariate Bernoulli distribution with parameter \( x \).
The denoising autoencoder

- Clean input $x \in [0, 1]^d$ is partially destroyed, yielding corrupted input: $\tilde{x} \sim q_D(\tilde{x} | x)$.
- $\tilde{x}$ is mapped to hidden representation $y = f_\theta(\tilde{x})$.
- From $y$ we reconstruct a $z = g_{\theta'}(y)$.
- Train parameters to minimize the cross-entropy “reconstruction error” $L_H(x, z) = H(B_x || B_z)$, where $B_x$ denotes multivariate Bernoulli distribution with parameter $x$. 

Pascal Vincent

Autoencoders, denoising autoencoders, and learning deep networks
Clean input $x \in [0, 1]^d$ is partially destroyed, yielding corrupted input: $\tilde{x} \sim q_D(\tilde{x} | x)$.

$\tilde{x}$ is mapped to hidden representation $y = f_\theta(\tilde{x})$.

From $y$ we reconstruct a $z = g_{\theta'}(y)$.

Train parameters to minimize the cross-entropy “reconstruction error” $L_H(x, z) = \mathbb{H}(B_x \parallel B_z)$, where $B_x$ denotes multivariate Bernoulli distribution with parameter $x$. 

The denoising autoencoder
Choose a fixed proportion $\nu$ of components of $\mathbf{x}$ at random.

- Reset their values to 0.
- Can be viewed as replacing a component considered missing by a default value.

Other corruption processes are possible.
We use standard sigmoid network layers:

- \( y = f_{\theta}(\tilde{x}) = \text{sigmoid}(W \tilde{x} + b) \)

- \( g_{\theta'}(y) = \text{sigmoid}(W' y + b') \)

and cross-entropy loss.
Denoising is a fundamentally different task

- Think of classical autoencoder in overcomplete case: $d' \geq d$
- Perfect reconstruction is possible without having learnt anything useful!
- Denoising autoencoder learns useful representation in this case.
- Being good at denoising requires capturing structure in the input.

Denoising using classical autoencoders was actually introduced much earlier (LeCun, 1987; Gallinari et al., 1987), as an alternative to Hopfield networks (Hopfield, 1982).
Denoising is a fundamentally different task

- Think of classical autoencoder in overcomplete case: $d' \geq d$

- Perfect reconstruction is possible without having learnt anything useful!

- Denoising autoencoder learns useful representation in this case.

- Being good at denoising requires capturing structure in the input.

Denoising using classical autoencoders was actually introduced much earlier (LeCun, 1987; Gallinari et al., 1987), as an alternative to Hopfield networks (Hopfield, 1982).
Denoising is a fundamentally different task

- Think of classical autoencoder in overcomplete case: $d' \geq d$
- Perfect reconstruction is possible without having learnt anything useful!
- Denoising autoencoder learns useful representation in this case.
- Being good at denoising requires capturing structure in the input.

Denoising using classical autoencoders was actually introduced much earlier (LeCun, 1987; Gallinari et al., 1987), as an alternative to Hopfield networks (Hopfield, 1982).
Denoising is a fundamentally different task

- Think of classical autoencoder in overcomplete case: \( d' \geq d \)

- Perfect reconstruction is possible without having learnt anything useful!

- Denoising autoencoder learns useful representation in this case.

- Being good at denoising requires capturing structure in the input.

Denoising using classical autoencoders was actually introduced much earlier (LeCun, 1987; Gallinari et al., 1987), as an alternative to Hopfield networks (Hopfield, 1982).
Denoising is a fundamentally different task

- Think of classical autoencoder in overcomplete case: $d' \geq d$
- Perfect reconstruction is possible without having learnt anything useful!
- Denoising autoencoder learns useful representation in this case.
- Being good at denoising requires capturing structure in the input.

Denoising using classical autoencoders was actually introduced much earlier (LeCun, 1987; Gallinari et al., 1987), as an alternative to Hopfield networks (Hopfield, 1982).
Learn first mapping $f_\theta$ by training as a denoising autoencoder.

- Remove scaffolding. Use $f_\theta$ directly on input yielding higher level representation.
- Learn next level mapping $f^{(2)}_\theta$ by training denoising autoencoder on current level representation.
- Iterate to initialize subsequent layers.
Learn first mapping $f_\theta$ by training as a denoising autoencoder.

Remove scaffolding. Use $f_\theta$ directly on input yielding higher level representation.

Learn next level mapping $f_\theta^{(2)}$ by training denoising autoencoder on current level representation.

Iterate to initialize subsequent layers.
1. Learn first mapping $f_\theta$ by training as a denoising autoencoder.
2. Remove scaffolding. Use $f_\theta$ directly on input yielding higher level representation.
3. Learn next level mapping $f^{(2)}_\theta$ by training denoising autoencoder on current level representation.
4. Iterate to initialize subsequent layers.
Learn first mapping $f_\theta$ by training as a denoising autoencoder.

2. Remove scaffolding. Use $f_\theta$ directly on input yielding higher level representation.

3. Learn next level mapping $f^{(2)}_\theta$ by training denoising autoencoder on current level representation.

4. Iterate to initialize subsequent layers.
Learn first mapping $f_\theta$ by training as a denoising autoencoder.

Remove scaffolding. Use $f_\theta$ directly on input yielding higher level representation.

Learn next level mapping $f_\theta^{(2)}$ by training denoising autoencoder on current level representation.

Iterate to initialize subsequent layers.
Learning deep networks
Layer-wise initialization

1. Learn first mapping $f_\theta$ by training as a denoising autoencoder.
2. Remove scaffolding. Use $f_\theta$ directly on input yielding higher level representation.
3. Learn next level mapping $f_\theta^{(2)}$ by training denoising autoencoder on current level representation.
4. Iterate to initialize subsequent layers.
1. Learn first mapping $f_\theta$ by training as a denoising autoencoder.

2. Remove scaffolding. Use $f_\theta$ directly on input yielding higher level representation.

3. Learn next level mapping $f_\theta^{(2)}$ by training denoising autoencoder on current level representation.

4. Iterate to initialize subsequent layers.
Initial deep mapping was learnt in an **unsupervised** way.

→ initialization for a supervised task.

Output layer gets added.

Global fine tuning by gradient descent on supervised criterion.
Initial deep mapping was learnt in an **unsupervised** way.

→ **initialization** for a **supervised** task.

Output layer gets added.

Global fine tuning by gradient descent on supervised criterion.
Initial deep mapping was learnt in an **unsupervised** way.

→ **initialization** for a **supervised** task.

**Output layer** gets added.

**Global fine tuning** by gradient descent on **supervised criterion**.

$$f^\text{sup}_\theta(x)$$

$$f^{(3)}_\theta$$

$$f^{(2)}_\theta$$

$$f^{(1)}_\theta$$

$$x$$

Target
Perspectives on denoising autoencoders
Manifold learning perspective

Denoising autoencoder can be seen as a way to learn a manifold:

- Suppose training data (×) concentrate near a low-dimensional manifold.
- Corrupted examples (●) are obtained by applying corruption process $q_D(\tilde{X}|X)$ and will lie farther from the manifold.
- The model learns with $p(X|\tilde{X})$ to “project them back” onto the manifold.
- Intermediate representation $Y$ can be interpreted as a coordinate system for points on the manifold.
Consider $X \sim q(X)$, $q$ unknown. $\tilde{X} \sim q_D(\tilde{X}|X)$. $Y = f_\theta(\tilde{X})$.

It can be shown that minimizing the expected reconstruction error amounts to maximizing a lower bound on mutual information $I(X; Y)$.

Denoising autoencoder training can thus be justified by the objective that hidden representation $Y$ captures as much information as possible about $X$ even as $Y$ is a function of corrupted input.
Denoising autoencoder training can be shown to be equivalent to maximizing a variational bound on the likelihood of a generative model for the corrupted data.
**basic**: subset of original MNIST digits: 10 000 training samples, 2 000 validation samples, 50 000 test samples.

**rot**: applied random rotation (angle between 0 and $2\pi$ radians)

**bg-rand**: background made of random pixels (value in 0...255)

**bg-img**: background is random patch from one of 20 images

**rot-bg-img**: combination of rotation and background image
Benchmark problems
Shape discrimination

- **rect**: discriminate between tall and wide rectangles on black background.

- **rect-img**: borderless rectangle filled with random image patch. Background is a different image patch.

- **convex**: discriminate between convex and non-convex shapes.
We compared the following algorithms on the benchmark problems:

- **SVM\(_{rbf}\)**: Support Vector Machines with Gaussian Kernel.
- **DBN-3**: Deep Belief Nets with 3 hidden layers (stacked Restricted Boltzmann Machines trained with contrastive divergence).
- **SAA-3**: Stacked Autoassociators with 3 hidden layers (no denoising).
- **SdA-3**: Stacked Denoising Autoassociators with 3 hidden layers.

Hyper-parameters for all algorithms were tuned based on classification performance on validation set. (In particular hidden-layer sizes, and \( \nu \) for SdA-3).
## Performance comparison

### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\text{SVM}_{rbf}$</th>
<th>DBN-3</th>
<th>SAA-3</th>
<th>$\text{SdA-3} (\nu)$</th>
<th>$\text{SVM}_{rbf}(\nu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>3.03 ± 0.15</td>
<td>3.11 ± 0.15</td>
<td>3.46 ± 0.16</td>
<td>2.80 ± 0.14 (10%)</td>
<td>3.07 (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>11.11 ± 0.28</td>
<td>10.30 ± 0.27</td>
<td>10.30 ± 0.27</td>
<td>10.29 ± 0.27 (10%)</td>
<td>11.62 (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>14.58 ± 0.31</td>
<td>6.73 ± 0.22</td>
<td>11.28 ± 0.28</td>
<td>10.38 ± 0.27 (40%)</td>
<td>15.63 (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>22.61 ± 0.37</td>
<td>16.31 ± 0.32</td>
<td>23.00 ± 0.37</td>
<td>16.68 ± 0.33 (25%)</td>
<td>23.15 (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>55.18 ± 0.44</td>
<td>47.39 ± 0.44</td>
<td>51.93 ± 0.44</td>
<td>44.49 ± 0.44 (25%)</td>
<td>54.16 (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>2.15 ± 0.13</td>
<td>2.60 ± 0.14</td>
<td>2.41 ± 0.13</td>
<td>1.99 ± 0.12 (10%)</td>
<td>2.45 (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>24.04 ± 0.37</td>
<td>22.50 ± 0.37</td>
<td>24.05 ± 0.37</td>
<td>21.59 ± 0.36 (25%)</td>
<td>23.00 (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>19.13 ± 0.34</td>
<td>18.63 ± 0.34</td>
<td>18.41 ± 0.34</td>
<td>19.06 ± 0.34 (10%)</td>
<td>24.20 (10%)</td>
</tr>
<tr>
<td>Dataset</td>
<td>SVM&lt;sub&gt;rbf&lt;/sub&gt;</td>
<td>DBN-3</td>
<td>SAA-3</td>
<td>SdA-3 (ν)</td>
<td>SVM&lt;sub&gt;rbf&lt;/sub&gt;(ν)</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------</td>
<td>-------</td>
<td>-------</td>
<td>-----------</td>
<td>----------------------</td>
</tr>
<tr>
<td>basic</td>
<td>3.03 ± 0.15</td>
<td>3.11 ± 0.15</td>
<td>3.46 ± 0.16</td>
<td>2.80 ± 0.14 (10%)</td>
<td>3.07 (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>11.11 ± 0.28</td>
<td>10.30 ± 0.27</td>
<td>10.30 ± 0.27</td>
<td>10.29 ± 0.27 (10%)</td>
<td>11.62 (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>14.58 ± 0.31</td>
<td>6.73 ± 0.22</td>
<td>11.28 ± 0.28</td>
<td>10.38 ± 0.27 (40%)</td>
<td>15.63 (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>22.61 ± 0.37</td>
<td>16.31 ± 0.32</td>
<td>23.00 ± 0.37</td>
<td>16.68 ± 0.33 (25%)</td>
<td>23.15 (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>55.18 ± 0.44</td>
<td>47.39 ± 0.44</td>
<td>51.93 ± 0.44</td>
<td>44.49 ± 0.44 (25%)</td>
<td>54.16 (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>2.15 ± 0.13</td>
<td>2.60 ± 0.14</td>
<td>2.41 ± 0.13</td>
<td>1.99 ± 0.12 (10%)</td>
<td>2.45 (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>24.04 ± 0.37</td>
<td>22.50 ± 0.37</td>
<td>24.05 ± 0.37</td>
<td>21.59 ± 0.36 (25%)</td>
<td>23.00 (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>19.13 ± 0.34</td>
<td>18.63 ± 0.34</td>
<td>18.41 ± 0.34</td>
<td>19.06 ± 0.34 (10%)</td>
<td>24.20 (10%)</td>
</tr>
</tbody>
</table>
## Performance comparison

### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\text{SVM}_{rbf}$</th>
<th>DBN-3</th>
<th>SAA-3</th>
<th>SdA-3 ($\nu$)</th>
<th>$\text{SVM}_{rbf} (\nu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>3.03 ± 0.15</td>
<td>3.11 ± 0.15</td>
<td>3.46 ± 0.16</td>
<td>2.80 ± 0.14 (10%)</td>
<td>3.07 (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>11.11 ± 0.28</td>
<td>10.30 ± 0.27</td>
<td>10.30 ± 0.27</td>
<td>10.29 ± 0.27 (10%)</td>
<td>11.62 (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>14.58 ± 0.31</td>
<td>6.73 ± 0.22</td>
<td>11.28 ± 0.28</td>
<td>10.38 ± 0.27 (40%)</td>
<td>15.63 (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>22.61 ± 0.37</td>
<td>16.31 ± 0.32</td>
<td>23.00 ± 0.37</td>
<td>16.68 ± 0.33 (25%)</td>
<td>23.15 (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>55.18 ± 0.44</td>
<td>47.39 ± 0.44</td>
<td>51.93 ± 0.44</td>
<td>44.49 ± 0.44 (25%)</td>
<td>54.16 (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>2.15 ± 0.13</td>
<td>2.60 ± 0.14</td>
<td>2.41 ± 0.13</td>
<td>1.99 ± 0.12 (10%)</td>
<td>2.45 (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>24.04 ± 0.37</td>
<td>22.50 ± 0.37</td>
<td>24.05 ± 0.37</td>
<td>21.59 ± 0.36 (25%)</td>
<td>23.00 (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>19.13 ± 0.34</td>
<td>18.63 ± 0.34</td>
<td>18.41 ± 0.34</td>
<td>19.06 ± 0.34 (10%)</td>
<td>24.20 (10%)</td>
</tr>
</tbody>
</table>
## Performance comparison
### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\text{SVM}_{rbf}$</th>
<th>DBN-3</th>
<th>SAA-3</th>
<th>SdA-3 ($\nu$)</th>
<th>SVM$_{rbf}$($\nu$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>3.03±0.15</td>
<td>3.11±0.15</td>
<td>3.46±0.16</td>
<td>2.80±0.14 (10%)</td>
<td>3.07 (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>11.11±0.28</td>
<td>10.30±0.27</td>
<td>10.30±0.27</td>
<td>10.29±0.27 (10%)</td>
<td>11.62 (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>14.58±0.31</td>
<td>6.73±0.22</td>
<td>11.28±0.28</td>
<td>10.38±0.27 (40%)</td>
<td>15.63 (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>22.61±0.37</td>
<td>16.31±0.32</td>
<td>23.00±0.37</td>
<td>16.68±0.33 (25%)</td>
<td>23.15 (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>55.18±0.44</td>
<td>47.39±0.34</td>
<td>51.93±0.34</td>
<td>44.49±0.44 (25%)</td>
<td>54.16 (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>2.15±0.13</td>
<td>2.60±0.14</td>
<td>2.41±0.13</td>
<td>1.99±0.12 (10%)</td>
<td>2.45 (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>24.04±0.37</td>
<td>22.50±0.37</td>
<td>24.05±0.37</td>
<td>21.59±0.36 (25%)</td>
<td>23.00 (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>19.13±0.34</td>
<td>18.63±0.34</td>
<td>18.41±0.34</td>
<td>19.06±0.34 (10%)</td>
<td>24.20 (10%)</td>
</tr>
</tbody>
</table>

Pascal Vincent  
Autoencoders, denoising autoencoders, and learning deep networks
## Performance comparison

### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM&lt;sub&gt;rbf&lt;/sub&gt;</th>
<th>DBN-3</th>
<th>SAA-3</th>
<th>SdA-3 (ν)</th>
<th>SVM&lt;sub&gt;rbf&lt;/sub&gt;(ν)</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>3.03±0.15</td>
<td>3.11±0.15</td>
<td>3.46±0.16</td>
<td>2.80±0.16 (10%)</td>
<td>3.07 (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>11.11±0.28</td>
<td>10.30±0.27</td>
<td>10.30±0.27</td>
<td>10.29±0.27 (10%)</td>
<td>11.62 (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>14.58±0.31</td>
<td>6.73±0.22</td>
<td>11.28±0.28</td>
<td>10.38±0.27 (40%)</td>
<td>15.63 (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>22.61±0.37</td>
<td>16.31±0.32</td>
<td>23.00±0.37</td>
<td>16.68±0.33 (25%)</td>
<td>23.15 (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>55.18±0.44</td>
<td>47.39±0.44</td>
<td>51.93±0.34</td>
<td>44.49±0.44 (25%)</td>
<td>54.16 (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>2.15±0.13</td>
<td>2.60±0.14</td>
<td>2.41±0.13</td>
<td>1.99±0.12 (10%)</td>
<td>2.45 (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>24.04±0.37</td>
<td>22.50±0.37</td>
<td>24.05±0.37</td>
<td>21.59±0.36 (25%)</td>
<td>23.00 (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>19.13±0.34</td>
<td>18.63±0.34</td>
<td>18.41±0.34</td>
<td>19.06±0.34 (10%)</td>
<td>24.20 (10%)</td>
</tr>
</tbody>
</table>
## Performance comparison

### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\text{SVM}_{rbf}$</th>
<th>DBN-3</th>
<th>SAA-3</th>
<th>$\text{SdA-3 (}\nu\text{)}$</th>
<th>$\text{SVM}_{rbf(}\nu\text{)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>$3.03 \pm 0.15$</td>
<td>$3.11 \pm 0.15$</td>
<td>$3.46 \pm 0.16$</td>
<td>$2.80 \pm 0.16$ (10%)</td>
<td>$3.07$ (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>$11.11 \pm 0.28$</td>
<td>$10.30 \pm 0.27$</td>
<td>$10.30 \pm 0.27$</td>
<td>$10.29 \pm 0.27$ (10%)</td>
<td>$11.62$ (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>$14.58 \pm 0.31$</td>
<td>$6.73 \pm 0.22$</td>
<td>$11.28 \pm 0.28$</td>
<td>$10.38 \pm 0.27$ (40%)</td>
<td>$15.63$ (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>$22.61 \pm 0.37$</td>
<td>$16.31 \pm 0.32$</td>
<td>$23.00 \pm 0.37$</td>
<td>$16.68 \pm 0.33$ (25%)</td>
<td>$23.15$ (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>$55.18 \pm 0.44$</td>
<td>$47.39 \pm 0.44$</td>
<td>$51.93 \pm 0.44$</td>
<td>$44.49 \pm 0.44$ (25%)</td>
<td>$54.16$ (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>$2.15 \pm 0.13$</td>
<td>$2.60 \pm 0.14$</td>
<td>$2.41 \pm 0.13$</td>
<td>$1.99 \pm 0.12$ (10%)</td>
<td>$2.45$ (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>$24.04 \pm 0.37$</td>
<td>$22.50 \pm 0.37$</td>
<td>$24.05 \pm 0.37$</td>
<td>$21.59 \pm 0.36$ (25%)</td>
<td>$23.00$ (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>$19.13 \pm 0.34$</td>
<td>$18.63 \pm 0.34$</td>
<td>$18.41 \pm 0.34$</td>
<td>$19.06 \pm 0.34$ (10%)</td>
<td>$24.20$ (10%)</td>
</tr>
</tbody>
</table>
## Performance comparison

### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\text{SVM}_{rbf}$</th>
<th>DBN-3</th>
<th>SAA-3</th>
<th>$\text{SdA-3 (}\nu\text{)}$</th>
<th>$\text{SVM}_{rbf(}\nu\text{)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>$3.03 \pm 0.15$</td>
<td>$3.11 \pm 0.15$</td>
<td>$3.46 \pm 0.16$</td>
<td>$2.80 \pm 0.14$ (10%)</td>
<td>$3.07$ (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>$11.11 \pm 0.28$</td>
<td>$10.30 \pm 0.27$</td>
<td>$10.30 \pm 0.27$</td>
<td>$10.29 \pm 0.27$ (10%)</td>
<td>$11.62$ (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>$14.58 \pm 0.31$</td>
<td>$6.73 \pm 0.22$</td>
<td>$11.28 \pm 0.28$</td>
<td>$10.38 \pm 0.27$ (40%)</td>
<td>$15.63$ (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>$22.61 \pm 0.37$</td>
<td>$16.31 \pm 0.32$</td>
<td>$23.00 \pm 0.37$</td>
<td>$16.68 \pm 0.33$ (25%)</td>
<td>$23.15$ (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>$55.18 \pm 0.44$</td>
<td>$47.39 \pm 0.44$</td>
<td>$51.93 \pm 0.44$</td>
<td>$44.49 \pm 0.44$ (25%)</td>
<td>$54.16$ (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>$2.15 \pm 0.13$</td>
<td>$2.60 \pm 0.14$</td>
<td>$2.41 \pm 0.13$</td>
<td>$1.99 \pm 0.12$ (10%)</td>
<td>$2.45$ (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>$24.04 \pm 0.37$</td>
<td>$22.50 \pm 0.37$</td>
<td>$24.05 \pm 0.37$</td>
<td>$21.59 \pm 0.36$ (25%)</td>
<td>$23.00$ (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>$19.13 \pm 0.34$</td>
<td>$18.63 \pm 0.34$</td>
<td>$18.41 \pm 0.34$</td>
<td>$19.06 \pm 0.34$ (10%)</td>
<td>$24.20$ (10%)</td>
</tr>
</tbody>
</table>
## Performance Comparison

### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$\text{SVM}_{rbf}$</th>
<th>$\text{DBN-3}$</th>
<th>$\text{SAA-3}$</th>
<th>$\text{SdA-3} (\nu)$</th>
<th>$\text{SVM}_{rbf} (\nu)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>3.03 ± 0.15</td>
<td>3.11 ± 0.15</td>
<td>3.46 ± 0.16</td>
<td>2.80 ± 0.14 (10%)</td>
<td>3.07 (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>11.11 ± 0.28</td>
<td>10.30 ± 0.27</td>
<td>10.30 ± 0.27</td>
<td>10.29 ± 0.27 (10%)</td>
<td>11.62 (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>14.58 ± 0.31</td>
<td>6.73 ± 0.22</td>
<td>11.28 ± 0.28</td>
<td>10.38 ± 0.27 (40%)</td>
<td>15.63 (25%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>22.61 ± 0.37</td>
<td>16.31 ± 0.32</td>
<td>23.00 ± 0.37</td>
<td>16.68 ± 0.33 (25%)</td>
<td>23.15 (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>55.18 ± 0.44</td>
<td>47.39 ± 0.44</td>
<td>51.93 ± 0.44</td>
<td>44.49 ± 0.44 (25%)</td>
<td>54.16 (10%)</td>
</tr>
<tr>
<td>rect</td>
<td>2.15 ± 0.13</td>
<td>2.60 ± 0.14</td>
<td>2.41 ± 0.13</td>
<td>1.99 ± 0.12 (10%)</td>
<td>2.45 (25%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>24.04 ± 0.37</td>
<td>22.50 ± 0.37</td>
<td>24.05 ± 0.37</td>
<td>21.59 ± 0.36 (25%)</td>
<td>23.00 (10%)</td>
</tr>
<tr>
<td>convex</td>
<td>19.13 ± 0.34</td>
<td>18.63 ± 0.34</td>
<td>18.41 ± 0.34</td>
<td>19.06 ± 0.34 (10%)</td>
<td>24.20 (10%)</td>
</tr>
</tbody>
</table>
Learnt filters
0 % destroyed
Learnt filters
10 % destroyed
Learnt filters
25 % destroyed

Pascal Vincent
Autoencoders, denoising autoencoders, and learning deep networks
Learnt filters
50 % destroyed
Conclusion

- Unsupervised initialization of layers with an explicit denoising criterion appears to help capture interesting structure in the input distribution.

- This leads to intermediate representations much better suited for subsequent learning tasks such as supervised classification.

- Resulting algorithm for learning deep networks is simple and improves on state-of-the-art on benchmark problems.

- Although our experimental focus was supervised classification, SdA is directly usable in a semi-supervised setting.

- We are currently investigating the effect of different types of corruption process, and applying the technique to recurrent nets.
THANK YOU!
## Performance comparison

<table>
<thead>
<tr>
<th>Dataset</th>
<th>SVM\textsubscript{\textit{rbf}}</th>
<th>SVM\textsubscript{poly}</th>
<th>DBN-1</th>
<th>DBN-3</th>
<th>SAA-3</th>
<th>SdA-3 ((\nu))</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic</td>
<td>3.03 ± 0.15</td>
<td>3.69 ± 0.17</td>
<td>3.94 ± 0.17</td>
<td>3.11 ± 0.15</td>
<td>3.46 ± 0.16</td>
<td>2.80 ± 0.14 (10%)</td>
</tr>
<tr>
<td>rot</td>
<td>11.11 ± 0.28</td>
<td>15.42 ± 0.32</td>
<td>14.69 ± 0.31</td>
<td>10.30 ± 0.27</td>
<td>10.30 ± 0.27</td>
<td>10.29 ± 0.27 (10%)</td>
</tr>
<tr>
<td>bg-rand</td>
<td>14.58 ± 0.31</td>
<td>16.62 ± 0.33</td>
<td>9.80 ± 0.26</td>
<td>6.73 ± 0.22</td>
<td>11.28 ± 0.28</td>
<td>10.38 ± 0.27 (40%)</td>
</tr>
<tr>
<td>bg-img</td>
<td>22.61 ± 0.37</td>
<td>24.01 ± 0.37</td>
<td>16.15 ± 0.32</td>
<td>16.31 ± 0.32</td>
<td>23.00 ± 0.37</td>
<td>16.68 ± 0.33 (25%)</td>
</tr>
<tr>
<td>rot-bg-img</td>
<td>55.18 ± 0.44</td>
<td>56.41 ± 0.43</td>
<td>52.21 ± 0.44</td>
<td>47.39 ± 0.44</td>
<td>51.93 ± 0.44</td>
<td>44.49 ± 0.44 (25%)</td>
</tr>
<tr>
<td>rect</td>
<td>2.15 ± 0.13</td>
<td>2.15 ± 0.13</td>
<td>4.71 ± 0.19</td>
<td>2.60 ± 0.14</td>
<td>2.41 ± 0.13</td>
<td>1.99 ± 0.12 (10%)</td>
</tr>
<tr>
<td>rect-img</td>
<td>24.04 ± 0.37</td>
<td>24.05 ± 0.37</td>
<td>23.69 ± 0.37</td>
<td>22.50 ± 0.37</td>
<td>24.05 ± 0.37</td>
<td>21.59 ± 0.36 (25%)</td>
</tr>
<tr>
<td>convex</td>
<td>19.13 ± 0.34</td>
<td>19.82 ± 0.35</td>
<td>19.92 ± 0.35</td>
<td>18.63 ± 0.34</td>
<td>18.41 ± 0.34</td>
<td>19.06 ± 0.34 (10%)</td>
</tr>
</tbody>
</table>

\*red when confidence intervals overlap.*
References

In *NIPS 19*.

In *Proceedings of COGNITIVA 87*, Paris, La Villette.


*Proceedings of the National Academy of Sciences, USA*, 79.

PhD thesis, Université de Paris VI.
Efficient learning of sparse representations with an energy-based model. 

Learning representations by back-propagating errors. 