Machine Learning and AI via Brain simulations

Andrew Ng

Stanford University & Google

Thanks to:

Stanford:
Adam Coates   Quoc Le   Honglak Lee   Andrew Saxe   Andrew Maas   Chris Manning   Jiquan Ngiam

Google:
Kai Chen   Greg Corrado   Jeff Dean   Matthieu Devin   Rajat Monga   Marc’Aurelio Ranzato   Paul Tucker   Kay Le
This talk

The idea of “deep learning.” Using brain simulations, hope to:
- Make learning algorithms much better and easier to use.
- Make revolutionary advances in machine learning and AI.

Vision is not only mine; shared with many researchers:

E.g., Samy Bengio, Yoshua Bengio, Tom Dean, Jeff Dean, Nando de Freitas, Jeff Hawkins, Geoff Hinton, Quoc Le, Yann LeCun, Honglak Lee, Tommy Poggio, Ruslan Salakhutdinov, Josh Tenenbaum, Kai Yu, Jason Weston, ....

I believe this is our best shot at progress towards real AI.
What do we want computers to do with our data?

Images/video
Label: “Motorcycle”
Suggest tags
Image search
...

Audio
Speech recognition
Music classification
Speaker identification
...

Text
Web search
Anti-spam
Machine translation
...

Andrew Ng
Computer vision is hard!
What do we want computers to do with our data?

Images/video
- Label: “Motorcycle”
- Suggest tags
- Image search
  ...

Audio
- Speech recognition
- Speaker identification
- Music classification
  ...

Text
- Web search
- Anti-spam
- Machine translation
  ...

Machine learning performs well on many of these problems, but is a lot of work. What is it about machine learning that makes it so hard to use?
This talk: Develop ideas using images and audio. Ideas apply to other problems (e.g., text) too.
Why is this hard?

You see this:

But the camera sees this:

<table>
<thead>
<tr>
<th>194</th>
<th>210</th>
<th>201</th>
<th>212</th>
<th>199</th>
<th>213</th>
<th>215</th>
<th>195</th>
<th>178</th>
<th>158</th>
<th>182</th>
<th>209</th>
</tr>
</thead>
<tbody>
<tr>
<td>180</td>
<td>189</td>
<td>190</td>
<td>221</td>
<td>209</td>
<td>205</td>
<td>191</td>
<td>167</td>
<td>147</td>
<td>115</td>
<td>129</td>
<td>163</td>
</tr>
<tr>
<td>114</td>
<td>126</td>
<td>140</td>
<td>188</td>
<td>176</td>
<td>165</td>
<td>152</td>
<td>140</td>
<td>170</td>
<td>106</td>
<td>78</td>
<td>88</td>
</tr>
<tr>
<td>87</td>
<td>103</td>
<td>115</td>
<td>154</td>
<td>143</td>
<td>142</td>
<td>149</td>
<td>153</td>
<td>173</td>
<td>101</td>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>102</td>
<td>112</td>
<td>106</td>
<td>131</td>
<td>122</td>
<td>138</td>
<td>152</td>
<td>147</td>
<td>128</td>
<td>84</td>
<td>58</td>
<td>66</td>
</tr>
<tr>
<td>94</td>
<td>95</td>
<td>79</td>
<td>104</td>
<td>105</td>
<td>124</td>
<td>129</td>
<td>113</td>
<td>107</td>
<td>87</td>
<td>69</td>
<td>67</td>
</tr>
<tr>
<td>68</td>
<td>71</td>
<td>69</td>
<td>98</td>
<td>89</td>
<td>92</td>
<td>98</td>
<td>95</td>
<td>89</td>
<td>88</td>
<td>76</td>
<td>67</td>
</tr>
<tr>
<td>41</td>
<td>56</td>
<td>68</td>
<td>69</td>
<td>99</td>
<td>63</td>
<td>45</td>
<td>60</td>
<td>82</td>
<td>58</td>
<td>76</td>
<td>75</td>
</tr>
<tr>
<td>20</td>
<td>43</td>
<td>69</td>
<td>75</td>
<td>56</td>
<td>41</td>
<td>51</td>
<td>73</td>
<td>55</td>
<td>70</td>
<td>63</td>
<td>44</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>57</td>
<td>69</td>
<td>75</td>
<td>75</td>
<td>73</td>
<td>74</td>
<td>53</td>
<td>68</td>
<td>59</td>
<td>37</td>
</tr>
<tr>
<td>72</td>
<td>59</td>
<td>53</td>
<td>66</td>
<td>84</td>
<td>92</td>
<td>84</td>
<td>74</td>
<td>57</td>
<td>72</td>
<td>63</td>
<td>42</td>
</tr>
<tr>
<td>67</td>
<td>61</td>
<td>58</td>
<td>65</td>
<td>75</td>
<td>78</td>
<td>76</td>
<td>73</td>
<td>59</td>
<td>75</td>
<td>69</td>
<td>50</td>
</tr>
</tbody>
</table>
Machine learning and feature representations

Input

Raw image

+ Motorbikes
- “Non”-Motorbikes

+ Motorbikes
- “Non”-Motorbikes

Learning algorithm
Machine learning and feature representations

Input

Raw image

pixel 1

pixel 2

Motorbikes
“Non”-Motorbikes

Learning algorithm
Machine learning and feature representations

Input

Raw image

- Motorbikes
- “Non”-Motorbikes

Learning algorithm

pixel 1

pixel 2
What we want

Input

Motorbikes

"Non"-Motorbikes

Feature representation

Learning algorithm

E.g., Does it have Handlebars? Wheels?

Raw image

Features

+ Motorbikes
- “Non”-Motorbikes

Handlebars

Wheels

pixel 1

pixel 2
Computing features in computer vision

But... we don’t have a handlebars detector. So, researchers try to hand-design features to capture various statistical properties of the image.

- Find edges at four orientations
- Sum up edge strength in each quadrant

Final feature vector

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

0.1  
0.7  
0.4  
0.6  
0.5  
0.5  

...
Feature representations

Input

Feature Representation

Learning algorithm
How is computer perception done?

Images/video
- Image
- Vision features
- Detection

Audio
- Audio
- Audio features
- Speaker ID

Text
- Text
- Text features
- Text classification,
  Machine translation,
  Information retrieval,
  ....
Feature representations

Input → Feature Representation → Learning algorithm
Computer vision features

SIFT

Spin image

HoG

RIFT

Textons

GLOH
Audio features

Spectrogram

MFCC

Flux

ZCR

Rolloff
Coming up with features is difficult, time-consuming, requires expert knowledge.

When working applications of learning, we spend a lot of time tuning the features.
Feature representations
The “one learning algorithm” hypothesis

Auditory cortex learns to see

[Roe et al., 1992]
The “one learning algorithm” hypothesis

Somatosensory cortex learns to see

Somatosensory Cortex

[Metin & Frost, 1989]
Sensor representations in the brain

Seeing with your tongue

Human echolocation (sonar)

Haptic belt: Direction sense

Implanting a 3rd eye

[BrainPort; Welsh & Blasch, 1997; Nagel et al., 2005; Constantine-Paton & Law, 2009]
On two approaches to computer perception

The adult visual system computes an incredibly complicated function of the input.

We can try to directly implement most of this incredibly complicated function (hand-engineer features).

Can we learn this function instead?

A trained learning algorithm (e.g., neural network, boosting, decision tree, SVM,…) is very complex. But the learning algorithm itself is usually very simple. The complexity of the trained algorithm comes from the data, not the algorithm.
Learning input representations

Find a better way to represent images than pixels.
Find a better way to represent audio.
Feature learning problem

• Given a 14x14 image patch $x$, can represent it using 196 real numbers.

• Problem: Can we find a better feature vector to represent this?
Self-taught learning (Unsupervised Feature Learning)

Unlabeled images

Motorcycles

Not motorcycles

Testing: What is this?

This uses unlabeled data. One can learn the features from labeled data too.
Self-taught learning (Unsupervised Feature Learning)

Unlabeled images

Motorcycles
Not motorcycles

Testing: What is this?

[This uses unlabeled data. One can learn the features from labeled data too.]
First stage of visual processing: V1

V1 is the first stage of visual processing in the brain. Neurons in V1 typically modeled as edge detectors:

Neuron #1 of visual cortex (model)

Neuron #2 of visual cortex (model)
Sparse coding (Olshausen & Field, 1996). Originally developed to explain early visual processing in the brain (edge detection).

Input: Images $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ (each in $\mathbb{R}^{n \times n}$)

Learn: Dictionary of bases $\phi_1, \phi_2, \ldots, \phi_k$ (also $\mathbb{R}^{n \times n}$), so that each input $x$ can be approximately decomposed as:

$$x \approx \sum_{j=1}^{k} a_j \phi_j$$

s.t. $a_j$'s are mostly zero ("sparse")
Sparse coding illustration

Natural Images

Learned bases (\(\phi_1, \ldots, \phi_{64}\)): “Edges”

Test example

\[
\begin{align*}
  x & \approx 0.8 \cdot \phi_{36} + 0.3 \cdot \phi_{42} + 0.5 \cdot \phi_{63} \\
  [a_1, \ldots, a_{64}] & = [0, 0, \ldots, 0, \textbf{0.8}, 0, \ldots, 0, \textbf{0.3}, 0, \ldots, 0, \textbf{0.5}, 0]
\end{align*}
\]

(Feature representation)

More succinct, higher-level, representation.
• Method “invents” edge detection.
• Automatically learns to represent an image in terms of the edges that appear in it. Gives a more succinct, higher-level representation than the raw pixels.
• Quantitatively similar to primary visual cortex (area V1) in brain.
Sparse coding applied to audio

Image shows 20 basis functions learned from unlabeled audio.

[Evan Smith & Mike Lewicki, 2006]
Sparse coding applied to audio

Image shows 20 basis functions learned from unlabeled audio.

[Evan Smith & Mike Lewicki, 2006]
Sparse coding applied to touch data

Collect touch data using a glove, following distribution of grasps used by animals in the wild.

Example learned representations
Learning feature hierarchies

Higher layer
(Combinations of edges; cf. V2)

“Sparse coding”
(edges; cf. V1)

Input image (pixels)

[Technical details: Sparse autoencoder or sparse version of Hinton’s DBN.]

[Lee, Ranganath & Ng, 2007]
Learning feature hierarchies

[Technical details: Sparse autoencoder or sparse version of Hinton’s DBN.]

[Lee, Ranganath & Ng, 2007]
Hierarchical Sparse coding (Sparse DBN): Trained on face images

Training set: Aligned images of faces.

[Image of aligned face images]

- Pixels
- Edges
- Object parts (combination of edges)
- Object models

[Andrew Ng]

[Honglak Lee]
Hierarchical Sparse coding (Sparse DBN)

Features learned from training on different object classes.
Machine learning applications
## Video Activity recognition (Hollywood 2 benchmark)

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hessian + ESURF [Williams et al 2008]</td>
<td>38%</td>
</tr>
<tr>
<td>Harris3D + HOG/HOF [Laptev et al 2003, 2004]</td>
<td>45%</td>
</tr>
<tr>
<td>Cuboids + HOG/HOF [Dollar et al 2005, Laptev 2004]</td>
<td>46%</td>
</tr>
<tr>
<td>Hessian + HOG/HOF [Laptev 2004, Williams et al 2008]</td>
<td>46%</td>
</tr>
<tr>
<td>Dense + HOG / HOF [Laptev 2004]</td>
<td>47%</td>
</tr>
<tr>
<td>Cuboids + HOG3D [Klaser 2008, Dollar et al 2005]</td>
<td>46%</td>
</tr>
<tr>
<td><strong>Unsupervised feature learning (our method)</strong></td>
<td><strong>52%</strong></td>
</tr>
</tbody>
</table>

Unsupervised feature learning significantly improves on the previous state-of-the-art. [Le, Zhou & Ng, 2011]
Sparse coding on audio (speech)

Approximately:

\[ x = 0.9 \phi_{36} + 0.7 \phi_{42} + 0.2 \phi_{63} \]
Many bases seem to correspond to phonemes.
Hierarchical Sparse coding (sparse DBN) for audio

Spectrogram
Hierarchical Sparse coding (sparse DBN) for audio

Spectrogram
Hierarchical Sparse coding (sparse DBN) for audio
### Phoneme Classification (TIMIT benchmark)

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clarkson and Moreno (1999)</td>
<td>77.6%</td>
</tr>
<tr>
<td>Gunawardana et al. (2005)</td>
<td>78.3%</td>
</tr>
<tr>
<td>Sung et al. (2007)</td>
<td>78.5%</td>
</tr>
<tr>
<td>Petrov et al. (2007)</td>
<td>78.6%</td>
</tr>
<tr>
<td>Sha and Saul (2006)</td>
<td>78.9%</td>
</tr>
<tr>
<td>Yu et al. (2006)</td>
<td>79.2%</td>
</tr>
<tr>
<td><strong>Unsupervised feature learning (our method)</strong></td>
<td><strong>80.3%</strong></td>
</tr>
</tbody>
</table>

Unsupervised feature learning significantly improves on the previous state-of-the-art. 

[Lee et al., 2009]
State-of-the-art
Unsupervised feature learning
<table>
<thead>
<tr>
<th>Images</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR Object classification</td>
<td>Accuracy</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Prior art (Ciresan et al., 2011)</td>
<td>80.5%</td>
<td>Prior art (Scherer et al., 2010)</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>82.0%</td>
<td>Stanford Feature learning</td>
</tr>
<tr>
<td>NORB Object classification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prior art (Scherer et al., 2010)</td>
<td>94.4%</td>
<td></td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Video                                       |                      |                      |
| Hollywood2 Classification                   | Accuracy             | Accuracy             |
| Prior art (Laptev et al., 2004)            | 48%                  | Prior art (Liu et al., 2009) | 71.2% |
| Stanford Feature learning                  | 53%                  | Stanford Feature learning | 75.8% |
| KTH                                         | Accuracy             |                      |
| Prior art (Wang et al., 2010)              | 92.1%                | Prior art (Wang et al., 2010) | 85.6% |
| Stanford Feature learning                  | 93.9%                | Stanford Feature learning | 86.5% |
| YouTube                                    |                      |                      |
| Prior art (Liu et al., 2009)               | 71.2%                |                      |
| Stanford Feature learning                  | 75.8%                |                      |
| UCF                                        |                      |                      |
| Prior art (Wang et al., 2010)              | 85.6%                |                      |
| Stanford Feature learning                  | 86.5%                |                      |
| Hollywood2 Classification                  |                      |                      |
| Prior art (Laptev et al., 2004)            | 48%                  | Prior art (Liu et al., 2009) | 71.2% |
| Stanford Feature learning                  | 53%                  | Stanford Feature learning | 75.8% |
| KTH                                        |                      |                      |
| Prior art (Wang et al., 2010)              | 92.1%                | Prior art (Wang et al., 2010) | 85.6% |
| Stanford Feature learning                  | 93.9%                | Stanford Feature learning | 86.5% |
| YouTube                                    |                      |                      |
| Prior art (Liu et al., 2009)               | 71.2%                |                      |
| Stanford Feature learning                  | 75.8%                |                      |
| UCF                                        |                      |                      |
| Prior art (Wang et al., 2010)              | 85.6%                |                      |
| Stanford Feature learning                  | 86.5%                |                      |

| Text/NLP                                    |                      |                      |
| Paraphrase detection                        | Accuracy             | Accuracy             |
| Prior art (Das & Smith, 2009)               | 76.1%                | Prior art (Nakagawa et al., 2010) | 77.3% |
| Stanford Feature learning                  | 76.4%                | Stanford Feature learning | 77.7% |
| Sentiment (MR/MPQA data)                   |                      |                      |
| Prior art (Nakagawa et al., 2010)           | 77.3%                |                      |
| Stanford Feature learning                  | 77.7%                |                      |

| Multimodal (audio/video)                   |                      |                      |
| AVLetters Lip reading                      | Accuracy             |                      |
| Prior art (Zhao et al., 2009)              | 58.9%                |                      |
| Stanford Feature learning                  | 65.8%                |                      |
| Other unsupervised feature learning records: |
| Pedestrian detection (Yann LeCun)          |                      |                      |
| Speech recognition (Geoff Hinton)          |                      |                      |
| PASCAL VOC object classification (Kai Yu)  |                      |                      |
Technical challenge: Scaling up
Supervised Learning

- Choices of learning algorithm:
  - Memory based
  - Winnow
  - Perceptron
  - Naïve Bayes
  - SVM
  - ….

- What matters the most?

“It’s not who has the best algorithm that wins. It’s who has the most data.”
Large numbers of features is critical. The specific learning algorithm is important, but ones that can scale to many features also have a big advantage.
Attemps to scale up

Significant effort spent on algorithmic tricks to get algorithms to run faster.

• Efficient sparse coding. [LeCun, Ng, Yu]
• Efficient posterior inference [Bengio, Hinton]
• Convolutional Networks. [Bengio, de Freitas, LeCun, Lee, Ng]
• Tiled Networks. [Hinton, Ng]
• Randomized/fast parameter search. [DiCarlo, Ng]
• Massive data synthesis. [LeCun, Schmidhuber]
• Massive embedding models [Bengio, Collobert, Hinton, Weston]
• Fast decoder algorithms. [LeCun, Lee, Ng, Yu]
• GPU, FPGA and ASIC implementations. [Dean, LeCun, Ng, Olukotun]
### Images

<table>
<thead>
<tr>
<th>CIFAR Object classification</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prior art (Ciresan et al., 2011)</td>
<td>80.5%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>82.0%</td>
</tr>
<tr>
<td>NORB Object classification</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Prior art (Scherer et al., 2010)</td>
<td>94.4%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>95.0%</td>
</tr>
</tbody>
</table>

### Video

<table>
<thead>
<tr>
<th>Hollywood2 Classification</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prior art (Laptev et al., 2004)</td>
<td>48%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>53%</td>
</tr>
<tr>
<td>KTH</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Prior art (Wang et al., 2010)</td>
<td>92.1%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>93.9%</td>
</tr>
<tr>
<td>YouTube</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Prior art (Liu et al., 2009)</td>
<td>71.2%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>75.8%</td>
</tr>
<tr>
<td>UCF</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Prior art (Wang et al., 2010)</td>
<td>85.6%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>86.5%</td>
</tr>
</tbody>
</table>

### Text/NLP

<table>
<thead>
<tr>
<th>Paraphrase detection</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prior art (Das &amp; Smith, 2009)</td>
<td>76.1%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>76.4%</td>
</tr>
<tr>
<td>Sentiment (MR/MPQA data)</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Prior art (Nakagawa et al., 2010)</td>
<td>77.3%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>77.7%</td>
</tr>
</tbody>
</table>

### Multimodal (audio/video)

<table>
<thead>
<tr>
<th>AVLetters Lip reading</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prior art (Zhao et al., 2009)</td>
<td>58.9%</td>
</tr>
<tr>
<td>Stanford Feature learning</td>
<td>65.8%</td>
</tr>
</tbody>
</table>

Other unsupervised feature learning records:
- Pedestrian detection (Yann LeCun)
- Speech recognition (Scott Hinton)
- PASCAL VOC object classification (Kai Yu)
Scaling up: Discovering object classes

[Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Greg Corrado, Matthieu Devin, Kai Chen, Jeff Dean]
Training procedure

What features can we learn if we train a massive model on a massive amount of data. Can we learn a “grandmother cell”?

- Train on 10 million images (YouTube)
- 1000 machines (16,000 cores) for 1 week.
- 1.15 billion parameters
- Test on novel images
Face neuron

Top Stimuli from the test set

Optimal stimulus by numerical optimization

Raina, Madhavan and Ng, 2008
Invariance properties

- **Horizontal shift**: Feature response remains relatively constant with a +15 pixel shift.
- **Vertical shift**: Feature response shows significant change with a +15 pixel shift.
- **3D rotation angle**: Feature response decreases with a 90° rotation.
- **Scale factor**: Feature response increases with a 1.6x scale factor.
Cat neuron

Top Stimuli from the test set

Optimal stimulus by numerical optimization
Cat face neuron

Random distractors

Cat faces
Visualization

Top Stimuli from the test set

Optimal stimulus by numerical optimization
Weaknesses & Criticisms
Weaknesses & Criticisms

• You’re learning everything. It’s better to encode prior knowledge about structure of images (or audio, or text).

A: Wasn’t there a similar machine learning vs. linguists debate in NLP ~20 years ago….

• Unsupervised feature learning cannot currently do X, where X is:
  - Go beyond Gabor (1-layer) features.
  - Work on temporal data (video).
  - Learn hierarchical representations (compositional semantics).
  - Get state-of-the-art in activity recognition.
  - Get state-of-the-art on image classification.
  - Get state-of-the-art on object detection.
  - Learn variable-size representations.

A: Many of these were true, but not anymore (were not fundamental weaknesses). There’s still work to be done though!

• We don’t understand the learned features.

A: True. Though many vision/audio/etc. features also suffer from this (e.g, concatenations/combinations of different features).
Conclusion
• Deep Learning and Self-Taught learning: Lets learn rather than manually design our features.
• Discover the fundamental computational principles that underlie perception?
• Sparse coding and deep versions very successful on vision and audio tasks. Other variants for learning recursive representations.
• To get this to work for yourself, see online tutorial: http://deeplearning.stanford.edu/wiki

Thanks to:
Advanced topics + Research philosophy

Andrew Ng
Stanford University & Google
Learning Recursive Representations
Imagine taking each word, and computing an n-dimensional feature vector for it.

[Distributional representations, or Bengio et al., 2003, Collobert & Weston, 2008.]

2-d embedding example below, but in practice use ~100-d embeddings.
"Generic" hierarchy on text doesn’t make sense

The cat on the mat.

Node has to represent sentence fragment "cat sat on." Doesn’t make sense.

Feature representation for words
What we want (illustration)

This node’s job is to represent “on the mat.”

The cat on the mat.

The cat sat on the mat.
The cat on the mat.

What we want (illustration)

This node’s job is to represent “on the mat.”
The country of my birth...
The cat on the mat.

Learning recursive representations

This node’s job is to represent “on the mat.”
Learning recursive representations

This node’s job is to represent “on the mat.”

The cat on the mat.
Learning recursive representations

Basic computational unit: Neural Network that inputs two candidate children’s representations, and outputs:
• Whether we should merge the two nodes.
• The semantic representation if the two nodes are merged.

This node’s job is to represent “on the mat.”
The cat on the mat.
Andrew Ng

The cat on the mat.

Parsing a sentence

Neural Network

No

Yes

Neural Network

Neural Network

The cat sat on the mat.
The cat on the mat.

Parsing a sentence
## Finding Similar Sentences

- Each sentence has a feature vector representation.
- Pick a sentence (“center sentence”) and list nearest neighbor sentences.
- Often either semantically or syntactically similar. (Digits all mapped to 2.)

<table>
<thead>
<tr>
<th>Similarities</th>
<th>Center Sentence</th>
<th>Nearest Neighbor Sentences (most similar feature vector)</th>
</tr>
</thead>
</table>
| Bad News                | Both took further hits yesterday                         | 1. We 're in for a lot of turbulence ...  
2. BSN currently has 2.2 million common shares outstanding  
3. This is panic buying  
4. We have a couple or three tough weeks coming |
| Something said          | I had calls all night long from the States, he said      | 1. Our intent is to promote the best alternative, he says  
2. We have sufficient cash flow to handle that, he said  
3. Currently, average pay for machinists is 22.22 an hour, Boeing said  
4. Profit from trading for its own account dropped, the securities firm said |
| Gains and good news     | Fujisawa gained 22 to 2,222                              | 1. Mochida advanced 22 to 2,222  
2. Commerzbank gained 2 to 222.2  
3. Paris loved her at first sight  
4. Profits improved across Hess's businesses |
| Unknown words which are cities | Columbia , S.C                                           | 1. Greenville , Miss  
2. UNK , Md |


# Finding Similar Sentences

<table>
<thead>
<tr>
<th>Similarities</th>
<th>Center Sentence</th>
<th>Nearest Neighbor Sentences (most similar feature vector)</th>
</tr>
</thead>
</table>
| Declining to comment = not disclosing  | Hess declined to comment                                                         | 1. PaineWebber declined to comment  
2. Phoenix declined to comment  
3. Campeau declined to comment  
4. Coastal wouldn't disclose the terms |
| Large changes in sales or revenue      | Sales grew almost 2% to 222.2 million from 222.2 million                         | 1. Sales surged 22% to 222.22 billion yen from 222.22 billion  
2. Revenue fell 2% to 2.22 billion from 2.22 billion  
3. Sales rose more than 2% to 22.2 million from 22.2 million  
4. Volume was 222.2 million shares, more than triple recent levels |
| Negation of different types            | There's nothing unusual about business groups pushing for more government spending| 1. We don't think at this point anything needs to be said  
2. It therefore makes no sense for each market to adopt different circuit breakers  
3. You can't say the same with black and white  
4. I don't think anyone left the place UNK UNK |
| People in bad situations               | We were lucky                                                                     | 1. It was chaotic  
2. We were wrong  
3. People had died |
|                                        |                                                                                  |
Application: Paraphrase Detection

- Task: Decide whether or not two sentences are paraphrases of each other. (MSR Paraphrase Corpus)

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>79.9</td>
</tr>
<tr>
<td>Rus et al., (2008)</td>
<td>80.5</td>
</tr>
<tr>
<td>Mihalcea et al., (2006)</td>
<td>81.3</td>
</tr>
<tr>
<td>Islam et al. (2007)</td>
<td>81.3</td>
</tr>
<tr>
<td>Qiu et al. (2006)</td>
<td>81.6</td>
</tr>
<tr>
<td>Fernando &amp; Stevenson (2008) (WordNet based features)</td>
<td>82.4</td>
</tr>
<tr>
<td>Das et al. (2009)</td>
<td>82.7</td>
</tr>
<tr>
<td>Wan et al (2006) (many features: POS, parsing, BLEU, etc.)</td>
<td>83.0</td>
</tr>
<tr>
<td><strong>Stanford Feature Learning</strong></td>
<td><strong>83.4</strong></td>
</tr>
</tbody>
</table>
A small crowd quietly enters the historic church.

Each node in the hierarchy has a “feature vector” representation.
Nearest neighbor examples for image patches

- Each node (e.g., set of merged superpixels) in the hierarchy has a feature vector.
- Select a node (“center patch”) and list nearest neighbor nodes.
- I.e., what image patches/superpixels get mapped to similar features?

Selected patch

Nearest Neighbors
**Multi-class segmentation (Stanford background dataset)**

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixel CRF (Gould et al., ICCV 2009)</td>
<td>74.3</td>
</tr>
<tr>
<td>Classifier on superpixel features</td>
<td>75.9</td>
</tr>
<tr>
<td>Region-based energy (Gould et al., ICCV 2009)</td>
<td>76.4</td>
</tr>
<tr>
<td>Local labelling (Tighe &amp; Lazebnik, ECCV 2010)</td>
<td>76.9</td>
</tr>
<tr>
<td>Superpixel MRF (Tighe &amp; Lazebnik, ECCV 2010)</td>
<td>77.5</td>
</tr>
<tr>
<td>Simultaneous MRF (Tighe &amp; Lazebnik, ECCV 2010)</td>
<td>77.5</td>
</tr>
<tr>
<td><strong>Stanford Feature learning (our method)</strong></td>
<td>78.1</td>
</tr>
</tbody>
</table>
### Multi-class Segmentation MSRC dataset: 21 Classes

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextonBoost (Shotton et al., ECCV 2006)</td>
<td>72.2</td>
</tr>
<tr>
<td>Framework over mean-shift patches (Yang et al., CVPR 2007)</td>
<td>75.1</td>
</tr>
<tr>
<td>Pixel CRF (Gould et al., ICCV 2009)</td>
<td>75.3</td>
</tr>
<tr>
<td>Region-based energy (Gould et al., IJCV 2008)</td>
<td>76.5</td>
</tr>
<tr>
<td><strong>Stanford Feature learning (out method)</strong></td>
<td><strong>76.7</strong></td>
</tr>
</tbody>
</table>
Analysis of feature learning algorithms

Andrew Coates  Honglak Lee
Supervised Learning

- Choices of learning algorithm:
  - Memory based
  - Winnow
  - Perceptron
  - Naïve Bayes
  - SVM
  - ….

- What matters the most?

“It’s not who has the best algorithm that wins. It’s who has the most data.”

[Banko & Brill, 2001]
Unsupervised Feature Learning

• Many choices in feature learning algorithms;
  – Sparse coding, RBM, autoencoder, etc.
  – Pre-processing steps (whitening)
  – Number of features learned
  – Various hyperparameters.

• What matters the most?
Unsupervised feature learning

Most algorithms learn Gabor-like edge detectors.

Sparse auto-encoder
Unsupervised feature learning

Weights learned with and without whitening.

- **Sparse auto-encoder**
  - With whitening
  - Without whitening

- **Sparse RBM**
  - With whitening
  - Without whitening

- **K-means**
  - With whitening
  - Without whitening

- **Gaussian mixture model**
  - With whitening
  - Without whitening
Scaling and classification accuracy (CIFAR-10)
Results on CIFAR-10 and NORB (old result)

- K-means achieves state-of-the-art
  - Scalable, fast and almost parameter-free, K-means does surprisingly well.

<table>
<thead>
<tr>
<th>CIFAR-10 Test accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw pixels</td>
</tr>
<tr>
<td>RBM with back-propagation</td>
</tr>
<tr>
<td>3-Way Factored RBM (3 layers)</td>
</tr>
<tr>
<td>Mean-covariance RBM (3 layers)</td>
</tr>
<tr>
<td>Improved Local Coordinate Coding</td>
</tr>
<tr>
<td>Convolutional RBM</td>
</tr>
<tr>
<td>Sparse auto-encoder</td>
</tr>
<tr>
<td>Sparse RBM</td>
</tr>
<tr>
<td>K-means (Hard)</td>
</tr>
<tr>
<td>K-means (Triangle, 1600 features)</td>
</tr>
<tr>
<td>K-means (Triangle, 4000 features)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NORB Test accuracy (error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional Neural Networks</td>
</tr>
<tr>
<td>Deep Boltzmann Machines</td>
</tr>
<tr>
<td>Deep Belief Networks</td>
</tr>
<tr>
<td>Jarrett et al., 2009</td>
</tr>
<tr>
<td>Sparse auto-encoder</td>
</tr>
<tr>
<td>Sparse RBM</td>
</tr>
<tr>
<td>K-means (Hard)</td>
</tr>
<tr>
<td>K-means (Triangle)</td>
</tr>
</tbody>
</table>
Tiled Convolution
Neural Networks
Learning Invariances

• We want to learn invariant features.

• Convolutional networks uses weight tying to:
  – Reduce number of weights that need to be learned. → Allows scaling to larger images/models.
  – Hard code translation invariance. Makes it harder to learn more complex types of invariances.

• Goal: Preserve computational scaling advantage of convolutional nets, but learn more complex invariances.
Fully Connected Topographic ICA

Pooling Units (Sqrt)

Simple Units (Square)

Input

Doesn’t scale to large images.
Fully Connected Topographic ICA

Pooling Units (Sqrt)

Simple Units (Square)

Orthogonalize

Input

Doesn’t scale to large images.
Local Receptive Fields

Pooling Units (Sqrt)

Simple Units (Square)

Input
Convolution Neural Networks (Weight Tying)

Pooling Units (Sqrt)

Simple Units (Square)

Input
Local pooling can capture complex invariances (not just translation); but total number of parameters is small.
Tiled Networks (Partial Weight Tying)

Pooling Units (Sqrt)

Tile Size \((k) = 2\)

Simple Units (Square)

Input
Tiled Networks (Partial Weight Tying)

Number of Maps ($l$) = 3

Tile Size ($k$) = 2

Pooling Units (Sqrt)

Simple Units (Square)

Input
Tiled Networks (Partial Weight Tying)

- **Input**: Input layer
- **Simple Units (Square)**
  - **Number of Maps (l)** = 3
  - **Tile Size (k)** = 2
- **Pooling Units (Sqrt)**
- **Local Orthogonalization**
## NORB and CIFAR-10 results

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>NORB Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Tiled CNNs [this work]</td>
<td><strong>96.1%</strong></td>
</tr>
<tr>
<td>CNNs [Huang &amp; LeCun, 2006]</td>
<td>94.1%</td>
</tr>
<tr>
<td>3D Deep Belief Networks [Nair &amp; Hinton, 2009]</td>
<td>93.5%</td>
</tr>
<tr>
<td>Deep Boltzmann Machines [Salakhutdinov &amp; Hinton, 2009]</td>
<td>92.8%</td>
</tr>
<tr>
<td>TICA [Hyvarinen et al., 2001]</td>
<td>89.6%</td>
</tr>
<tr>
<td>SVMs</td>
<td>88.4%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>CIFAR-10 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved LCC [Yu et al., 2010]</td>
<td>74.5%</td>
</tr>
<tr>
<td><strong>Deep Tiled CNNs [this work]</strong></td>
<td><strong>73.1%</strong></td>
</tr>
<tr>
<td>LCC [Yu et al., 2010]</td>
<td>72.3%</td>
</tr>
<tr>
<td>mcRBMs [Ranzato &amp; Hinton, 2010]</td>
<td>71.0%</td>
</tr>
<tr>
<td>Best of all RBMs [Krizhevsky, 2009]</td>
<td>64.8%</td>
</tr>
<tr>
<td>TICA [Hyvarinen et al., 2001]</td>
<td>56.1%</td>
</tr>
</tbody>
</table>
Summary/Big ideas
Summary/Big ideas

• Large scale brain simulations as revisiting of the big “AI dream.”

• “Deep learning” has had two big ideas:
  – Learning multiple layers of representation
  – Learning features from unlabeled data

• Has worked well so far in two regimes (confusing to outsiders):
  – Lots of labeled data. “Train the heck out of the network.”
  – Unsupervised Feature Learning/Self-Taught learning

• Scalability is important.

• Detailed tutorial: http://deeplearning.stanford.edu