Neural Networks: Deep Networks and Autoencoders

Based on slides from C.J. Taylor, Alex Krizhevsky, Ilya Sutskever, Geoff Hinton, Quoc Le, Lyle Ungar ...
Neural Nets

◆ Non-parametric
  ● Or, technically, semi-parametric
  ● Flexible model form
  ● No priors, no feature selection

◆ Used when there are vast amounts of data
  ● Hence popular (again) now

◆ Deep networks
  ● Idea: representation should have different levels of abstraction
Neural Nets can be

- **Supervised**
  - Generalizes *logistic regression* to a semi-parametric form

- **Unsupervised**
  - Generalizes *PCA* to a semi-parametric form

For image recognition, neural nets often have built in structure
ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
Ilya Sutskever
Geoffrey Hinton

University of Toronto
Canada

Paper with same name to appear in NIPS 2012
Neural networks

- A neuron

\[ x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3) \]

\( x \) is called the total input to the neuron, and \( f(x) \) is its output

- A neural network

A neural network computes a differentiable function of its input. For example, ours computes:

\[ p(\text{label} \mid \text{an input image}) \]
Neurons

Traditional sigmoidal e.g. logistic function

\[ f(x) = \tanh(x) \]

\[ x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3) \]

\( x \) is called the total input to the neuron, and \( f(x) \) is its output

Very bad (slow to train)

But one can use any nonlinear function

\[ f(x) = \max(0, x) \]

Very good (quick to train)
Overview of our model

- **Deep**: 7 hidden “weight” layers
- **Learned**: all feature extractors initialized at white Gaussian noise and learned from the data
- Entirely supervised
- More data = good

**Convolutional layer**: convolves its input with a bank of 3D filters, then applies point-wise non-linearity

**Fully-connected layer**: applies linear filters to its input, then applies point-wise non-linearity
Local receptive fields

layer \( m+1 \)

layer \( m \)

layer \( m-1 \)
**Figure 1:** example of a convolutional layer

Here, we show two layers of a CNN, containing 4 feature maps at layer \((m-1)\) and 2 feature maps \((h^0_l\) and \(h^1_l\)) at layer \(m\). Pixels (neuron outputs) in \(h^0_l\) and \(h^1_l\) (outlined as blue and red squares) are computed from pixels of layer \((m-1)\) which fall within their 2x2 receptive field in the layer below (shown as colored rectangles). Notice how the receptive field spans all four input feature maps. The weights \(W^0\) and \(W^1\) of \(h^0_l\) and \(h^1_l\) are thus 3D weight tensors. The leading dimension indexes the input feature maps, while the other two refer to the pixel coordinates.

Putting it all together, \(W_{ij}^{kl}\) denotes the weight connecting each pixel of the \(k\)-th feature map at layer \(m\), with the pixel at coordinates \((i,j)\) of the \(l\)-th feature map of layer \((m-1)\).
Local pooling

**Max-pooling** partitions the input image into non-overlapping rectangles and outputs the maximum value for each.

Reduces the computational complexity
Provides translation invariance.
Our model

- Max-pooling layers follow first, second, and fifth convolutional layers
- The number of neurons in each layer is given by 253440, 186624, 64896, 64896, 43264, 4096, 4096, 1000
Overview of our model

- Trained with stochastic gradient descent on two NVIDIA GPUs for about a week
- 650,000 neurons
- 60,000,000 parameters
- 630,000,000 connections
- **Final feature layer:** 4096-dimensional

**Convolutional layer:** convolves its input with a bank of 3D filters, then applies point-wise non-linearity

**Fully-connected layer:** applies linear filters to its input, then applies point-wise non-linearity
Training

Using stochastic gradient descent and the backpropagation algorithm (just repeated application of the chain rule)

One output unit per class
\[ x_i = \text{total input to output unit } i \]
\[ f(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{1000} \exp(x_j)} \]

We maximize the log-probability of the correct label, \( \log f(x_t) \)
Data augmentation

• Our neural net has 60M real-valued parameters and 650,000 neurons

• It overfits a lot. Therefore we train on 224x224 patches extracted randomly from 256x256 images, and also their horizontal reflections.
Validation classification

<table>
<thead>
<tr>
<th>mite</th>
<th>container ship</th>
<th>motor scooter</th>
<th>leopard</th>
</tr>
</thead>
<tbody>
<tr>
<td>mite, black widow</td>
<td>container ship, lifeboat</td>
<td>motor scooter, go-kart</td>
<td>leopard, jaguar, cheetah</td>
</tr>
<tr>
<td>cockroach, tick</td>
<td>amphibian, fireboat</td>
<td>moped, bumper car, golfcart</td>
<td></td>
</tr>
<tr>
<td>starfish</td>
<td>drilling platform</td>
<td></td>
<td>snow leopard, Egyptian cat</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>grille</th>
<th>mushroom</th>
<th>cherry</th>
<th>Madagascar cat</th>
</tr>
</thead>
<tbody>
<tr>
<td>convertible, grille</td>
<td>agaric, mushroom</td>
<td>dalmatian, grape, elderberry,</td>
<td>squirrel monkey, spider monkey</td>
</tr>
<tr>
<td>pickup, beach wagon</td>
<td>jelly fungus, gill fungus,</td>
<td>affordshire bullterrier,</td>
<td>titi, indri, howler monkey</td>
</tr>
<tr>
<td>fire engine</td>
<td>dead-man's-fingers</td>
<td>currant</td>
<td></td>
</tr>
</tbody>
</table>
Validation classification
Validation localizations
Retrieval experiments

First column contains query images from ILSVRC-2010 test set, remaining columns contain retrieved images from training set.
Unsupervised Neural Nets

- **Autoencoder**
  - Takes same image as input and output, often adding noise to the input
  - Learns weights to minimize the reconstruction error

- **Generalizes PCA or ICA**
- **Produces new features that can be used in supervised learning**
Independent Components Analysis (ICA)

- Given observations $X$, find $W$ such that components $s_j$ of $S = XW$ are “as independent of each other as possible”
  - E.g. have maximum KL-divergence or low mutual information
  - Alternatively, find directions in $X$ that are most skewed
  - Usually mean center and “whiten” the data first

- Very much like PCA
  - But the loss function is not quadratic
  - And optimization cannot be done by SVD
Independent Components Analysis (ICA)

- Given observations $X$, find $W$ such that components $s_j$ of $s = X W$ are “as independent of each other as possible”
- Reconstruct $X \sim XWW^+ = SW^+$
  - $S$ like the PCs like $W^+$ like loadings

- **Autoencoder** – nonlinear generalization that “encodes” $X$ as $S$ and then “decodes” it
Semi-Supervised Learning

- Hypothesis: $P(c|x)$ can be more accurately computed using shared structure with $P(x)$
Semi-Supervised Learning

- Hypothesis: $P(c|x)$ can be more accurately computed using shared structure with $P(x)$

from Socher and Manning
Tera-scale deep learning

Quoc V. Le
Stanford University and Google
Joint work with

Kai Chen  Greg Corrado  Jeff Dean  Matthieu Devin

Rajat Monga  Andrew Ng  Marc’ Aurelio Ranzato  Paul Tucker  Ke Yang

Additional Thanks: Samy Bengio, Zhenghao Chen, Tom Dean, Pangwei Koh, Mark Mao, Jiquan Ngiam, Patrick Nguyen, Andrew Saxe, Mark Segal, Jon Shlens, Vincent Vanhouke, Xiaoyun Wu, Peng Xe, Serena Yeung, Will Zou
Warning: this $x$ and $W$ are the transpose of what we use

\[ \min_W \sum_j \sum_i h_j(W; x^{(i)}) \]

\[ \text{s.t. } WW^T = I \]

\[ \min_W \frac{\lambda}{m} \sum_{i=1}^{m} \left\| W^T W x^{(i)} - x^{(i)} \right\|_2^2 + \sum_j \sum_i h_j(W; x^{(i)}) \]

**Lemma 3.1** When the input data $\{x^{(i)}\}_{i=1}^{m}$ is whitened, the reconstruction cost $\frac{\lambda}{m} \sum_{i=1}^{m} \left\| W^T W x^{(i)} - x^{(i)} \right\|_2^2$ is equivalent to the orthonormality cost $\lambda \left\| W^T W - I \right\|_F^2$.

**Lemma 3.2** The column orthonormality cost $\lambda \left\| W^T W - I_n \right\|_F^2$ is equivalent to the row orthonormality cost $\lambda \left\| W W^T - I_k \right\|_F^2$ up to an additive constant.

---

- Equivalence between Sparse Coding, Autoencoders, RBMs and ICA
- Build deep architecture by treating the output of one layer as input to another layer

Le, et al., *ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning.* NIPS 2011
Visualization of features learned

Most are local features
Challenges with 1000s of machines
Asynchronous Parallel SGDs

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012
Local receptive field networks

Le, et al., *Tiled Convolutional Neural Networks*. NIPS 2010
10 million 200x200 images
1 billion parameters
Training

Dataset: 10 million 200x200 unlabeled images from YouTube/Web

Train on 2000 machines (16000 cores) for 1 week

1.15 billion parameters
- 100x larger than previously reported
- Small compared to visual cortex

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012
The face neuron

Top stimuli from the test set

Optimal stimulus by numerical optimization

Le, et al., *Building high-level features using large-scale unsupervised learning*. ICML 2012
The cat neuron

Le, et al., Building high-level features using large-scale unsupervised learning. ICML 2012
What you should know

- **Supervised neural nets**
  - Generalize *logistic regression*
  - Often solved by stochastic gradient descent plus chain rule ("backpropagation")

- **Unsupervised neural nets**
  - Generalize *PCA* or *ICA*
  - Often trained recursively as nonlinear autoencoders
  - Generally learn an "overcomplete basis"
  - Used in semi-supervised learning

For image recognition, neural nets often have built in structure – local receptive fields and max-pooling