Building an Object Recognition System

IDEA: Use data to optimize features for the given task.

```
      FEATURE EXTRACTOR
        ▸
          “CAR”
        ▸
            CLASSIFIER
```

Ranzato
Building an Object Recognition System

What we want: Use parameterized function such that
a) features are computed efficiently
b) features can be trained efficiently
Building an Object Recognition System

- Everything becomes adaptive.
- No distinction between feature extractor and classifier.
- Big non-linear system trained from raw pixels to labels.
Q: How can we build such a highly non-linear system?
Building an Object Recognition System

Q: How can we build such a highly non-linear system?
A: By combining simple building blocks we can make more and more complex systems.
Building A Complicated Function

Simple Functions

\[ \sin(x) \quad \log(x) \quad \cos(x) \quad x^3 \quad \exp(x) \]

One Example of Complicated Function

\[ \log(\cos(\exp(\sin^3(x)))) \]
Building A Complicated Function

Simple Functions

\[ \sin(x), \log(x), \cos(x), x^3, \exp(x) \]

One Example of Complicated Function

\[ \log(\cos(\exp(\sin^3(x)))) \]

- Function composition is at the core of deep learning methods.
- Each “simple function” will have parameters subject to training.
Implementing A Complicated Function

Complicated Function

\[ \log (\cos (\exp (\sin^3 (x)))) \]

Diagram:

1. \( \sin (x) \)
2. \( x^3 \)
3. \( \exp (x) \)
4. \( \cos (x) \)
5. \( \log (x) \)
Intuition Behind Deep Neural Nets

“CAR”
Intuition Behind Deep Neural Nets

**NOTE:** Each black box can have trainable parameters. Their composition makes a highly non-linear system.
Intuition Behind Deep Neural Nets

NOTE: System produces a hierarchy of features.
Intuition Behind Deep Neural Nets

Q: What do the intermediate representations do?
Intuition Behind Deep Neural Nets

Lee et al. “Convolutional DBN’s for scalable unsup. learning…” ICML 2009
Intuition Behind Deep Neural Nets

“CAR”

Lee et al. ICML 2009
Intuition Behind Deep Neural Nets

“CAR”

Lee et al. ICML 2009
KEY IDEAS OF NEURAL NETS

IDEA # 1
Learn features from data

IDEA # 2
Use differentiable functions that produce features efficiently

IDEA # 3
End-to-end learning: no distinction between feature extractor and classifier

IDEA # 4
“Deep” architectures: cascade of simpler non-linear modules
KEY QUESTIONS

- What is the input-output mapping?
- How are parameters trained?
- How computational expensive is it?
- How well does it work?
Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function

- Neural Networks for Vision: Convolutional & Tiled

- Unsupervised Training of Neural Networks

- Extensions:
  - semi-supervised / multi-task / multi-modal

- Comparison to Other Methods
  - boosting & cascade methods
  - probabilistic models

- Large-Scale Learning with Deep Neural Nets
Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function

- Neural Networks for Vision: Convolutional & Tiled

- Unsupervised Training of Neural Networks

- Extensions:
  - semi-supervised / multi-task / multi-modal

- Comparison to Other Methods
  - boosting & cascade methods
  - probabilistic models

- Large-Scale Learning with Deep Neural Nets
Linear Classifier: SVM

Input: $X \in \mathbb{R}^D$

Binary label: $y$

Parameters: $W \in \mathbb{R}^D$

Output prediction: $W^T X$

Loss: $L = \frac{1}{2} \|W\|^2 + \lambda \max[0, 1 - W^T X \cdot y]$

Hinge Loss
Linear Classifier: Logistic Regression

Input: $X ∈ R^D$

Binary label: $y$

Parameters: $W ∈ R^D$

Output prediction: $W^T X$

Loss: $L = \frac{1}{2} \| W \|^2 + \lambda \log(1 + \exp(-W^T X y))$
Logistic Regression: Probabilistic Interpretation

Input: \( X \in \mathbb{R}^D \)

Binary label: \( y \)

Parameters: \( W \in \mathbb{R}^D \)

Output prediction: \( p(y=1|X) = \frac{1}{1 + e^{-W^T X}} \)

Loss: \( L = -\log(p(y|X)) \)

\[ \text{Q: What is the gradient of } L \text{ w.r.t. } W? \]
Logistic Regression: Probabilistic Interpretation

Input: \( X \in \mathbb{R}^D \)

Binary label: \( y \)

Parameters: \( W \in \mathbb{R}^D \)

Output prediction: \( p(y=1|X) = \frac{1}{1 + e^{-W^T X}} \)

Loss: \( L = \log(1 + \exp(-W^T X y)) \)

Q: What is the gradient of \( L \) w.r.t. \( W \)?
Simple Functions

\begin{align*}
\sin(x) & \\
\cos(x) & \\
x^3 & \\
\log(x) & \\
\exp(x) & \\
\end{align*}

Complicated Function

\[-\log\left(\frac{1}{1+e^{-W^TX}}\right)\]
Logistic Regression: Computing Loss

Complicated Function

\[-\log\left(\frac{1}{1 + e^{-W^T X}}\right)\]

\[W^T X \quad u \quad \frac{1}{1 + e^{-u}} \quad p \quad -\log(p) \quad L\]
Given $y(x)$ and $dL/dy$, What is $dL/dx$?
Chain Rule

Given $y(x)$ and $dL/\ dy$, What is $dL/\ dx$?

\[
\frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx}
\]
Chain Rule

\[ \frac{dL}{dx} = \frac{dL}{dy} \cdot \frac{dy}{dx} \]

Given \( y(x) \) and \( dL/dy \),
What is \( dL/dx \)?

All needed information is local!
Logistic Regression: Computing Gradients

\[ X \xrightarrow{W^T X} u \xrightarrow{\frac{1}{1+e^{-u}}} p \xrightarrow{-\log(p)} L \]

\[ \frac{-1}{p} \]
Logistic Regression: Computing Gradients

\[ \frac{dL}{dp} = \frac{-1}{p} \]

\[ p(1-p) \]

\[ \frac{dp}{du} = \frac{1}{1+e^{-u}} \]

\[ L \]

\[ X \]

\[ W^T X \]

\[ u \]

\[ \frac{1}{1+e^{-u}} \]

\[ p \]

\[ -\log(p) \]
Logistic Regression: Computing Gradients

\[
\begin{align*}
X & \rightarrow W^T X & \rightarrow u & = \frac{1}{1 + e^{-u}} & \rightarrow p & = -\log(p) & \rightarrow L \\
\frac{dL}{dW} & = \frac{dL}{dp} \cdot \frac{dp}{du} \cdot \frac{du}{dW} & = (p - 1)X
\end{align*}
\]
What Did We Learn?

- Logistic Regression
- How to compute gradients of complicated functions
Neural Network

Logistic Regression → Logistic Regression
A neural net can be thought of as a stack of logistic regression classifiers. Each input is the output of the previous layer.

NOTE: intermediate units can be thought of as linear classifiers trained with implicit target values.
Key Computations: F-Prop / B-Prop

F-PROP

\[ X \rightarrow \theta \rightarrow Z \]
Key Computations: F-Prop / B-Prop

B-PROP

\[ \frac{\partial L}{\partial X} \]

\[ \{ \frac{\partial Z}{\partial X}, \frac{\partial Z}{\partial \theta} \} \]

\[ \frac{\partial L}{\partial Z} \]

\[ \frac{\partial L}{\partial \theta} \]
Neural Net: Training

A) Compute loss on small mini-batch
Neural Net: Training

A) Compute loss on small mini-batch

F-PROP

Layer 2 -> Layer 3
A) Compute loss on small mini-batch

Neural Net: Training

F-PROP

Layer 1  \rightarrow  Layer 3
Neural Net: Training

A) Compute loss on small mini-batch

F-PROP
Neural Net: Training

A) Compute loss on small mini-batch
B) Compute gradient w.r.t. parameters

B-PROP
Neural Net: Training

A) Compute loss on small mini-batch
B) Compute gradient w.r.t. parameters

B-PROP

Layer 1 -> Layer 2 -> Layer 3
Neural Net: Training

A) Compute loss on small mini-batch
B) Compute gradient w.r.t. parameters

B-PROP
Neural Net: Training

A) Compute loss on small mini-batch
B) Compute gradient w.r.t. parameters
C) Use gradient to update parameters \[ \theta \leftarrow \theta - \eta \frac{dL}{d\theta} \]
\[ h_{j+1} = \sigma(W_{j+1}^T h_j + b_{j+1}) \]

\[ W_j \in \mathbb{R}^{M \times N}, \quad b_j \in \mathbb{R}^N \]

\[ h_j \in \mathbb{R}^M, \quad h_{j+1} \in \mathbb{R}^N \]
NEURAL NET: ARCHITECTURE

\[ h_{j+1} = \sigma(W_{j+1}^T h_j + b_{j+1}) \]

\[ \sigma(x) = \frac{1}{1 + e^{-x}} \]
NEURAL NET: ARCHITECTURE

\[ h_{j+1} = \sigma(W_{j+1}^T h_j + b_{j+1}) \]

\[ \sigma(x) = \tanh(x) \]
Graphical Notations

\[ f(X; W) \]

is equivalent to

\[ X \rightarrow h \]

\[ W \]

\( h_k \) is called feature, hidden unit, neuron or code unit
NOTE: Multi-layer neural nets with more than two layers are nowadays called deep nets!!
NOTE: User must specify number of layers, number of hidden units, type of layers and loss function.

NOTE: Multi-layer neural nets with more than two layers are nowadays called deep nets!!
MOST COMMON LOSSES

Square Euclidean Distance (regression):

\[ y, \hat{y} \in \mathbb{R}^N \]

\[ L = \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \]
Cross Entropy (classification):

\[ y, \hat{y} \in [0,1]^N, \quad \sum_{i=1}^{N} y_i = 1, \quad \sum_{i=1}^{N} \hat{y}_i = 1 \]

\[ L = - \sum_{i=1}^{N} y_i \log \hat{y}_i \]
**NOTE:** User specifies loss based on the task.
Algorithm (SGD):
Given a small mini-batch
- FPROP
- BPROP
- PARAMETER UPDATE
\[ \theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta} \]
NOTES
- User chooses optimization algorithm.
- Computational cost of F-PROP & B-PROP is similar.
TOY EXAMPLE: SYNTHETIC DATA

1 input & 1 output
100 hidden units in each layer

Diagram showing curves for 1 hidden layer, 2 hidden layers, and 3 hidden layers.
TOY EXAMPLE: SYNTHETIC DATA

- 1 input & 1 output
- 3 hidden layers

Graph showing the output for different numbers of hidden layers: 10, 100, and 1000.
TOY EXAMPLE: SYNTHETIC DATA

1 input & 1 output
3 hidden layers, 1000 hiddens
Regression of cosine
TOY EXAMPLE: SYNTHETIC DATA

1 input & 1 output
3 hidden layers, 1000 hiddens
Regression of cosine
TOY EXAMPLE: MNIST

Linear Classifier ................................................................. 12.0%
Boosted stumps ................................................................. 7.7%
Product of boosted stumps .................................................. 1.3%
Boosted trees .................................................................... 1.5%
Stumps on Haar features .................................................... 0.9%
K-NN ............................................................................. 5.0%
SVM Gaussian kernel .......................................................... 1.4%
2 layer nnet 800 hiddens .................................................... 1.6%
4 layer nnet (pre-trained) .................................................... 1.0%
Conv. Net (pre-trained) ....................................................... 0.6%

http://yann.lecun.com/exdb/mnist/
Outline

- **Neural Networks for Supervised Training**
  - Architecture
  - Loss function

- **Neural Networks for Vision: Convolutional & Tiled**

- **Unsupervised Training of Neural Networks**

- **Extensions:**
  - semi-supervised / multi-task / multi-modal

- **Comparison to Other Methods**
  - boosting & cascade methods
  - probabilistic models

- **Large-Scale Learning with Deep Neural Nets**
FULLY CONNECTED NEURAL NET
Example: 1000x1000 image
1M hidden units
1B parameters!!!

- Spatial correlation is local
- Better to put resources elsewhere!
LOCALLY CONNECTED NEURAL NET

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters
Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters
LOCALLY CONNECTED NEURAL NET

STATIONARITY? Statistics is similar at different locations

Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters
CONVOLUTIONAL NET

Share the same parameters across different locations:

Convolutions with learned kernels
Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters
NEURAL NETS FOR VISION

A standard neural net applied to images:
- scales quadratically with the size of the input
- does not leverage stationarity

Solution:
- connect each hidden unit to a small patch of the input
- share the weight across hidden units

This is called: convolutional network.

LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998
Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?
By “pooling” (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.
CONV NETS: EXTENSIONS

Over the years, some new modules have proven to be very effective when plugged into conv-nets:

- **L2 Pooling**

\[
h_{i+1, x, y} = \sqrt{\sum_{(j, k) \in \mathcal{N}(x, y)} h_{i, j, k}^2}
\]

- **Local Contrast Normalization**

\[
h_{i+1, x, y} = \frac{h_{i, x, y} - m_{i, \mathcal{N}(x, y)}}{\sigma_{i, \mathcal{N}(x, y)}}
\]

*Jarrett et al. “What is the best multi-stage architecture for object recognition?” ICCV 2009*
Kavukguoglu et al. “Learning invariant features …” CVPR 2009
Kavukguoglu et al. “Learning invariant features …” CVPR 2009
LOCAL CONTRAST NORMALIZATION

\[ h_{i+1, x, y} = \frac{h_{i, x, y} - m_{i, N(x, y)}}{\sigma_{i, N(x, y)}} \]
LOCAL CONTRAST NORMALIZATION

\[ h_{i+1,x,y} = \frac{h_{i,x,y} - m_{i,N}(x,y)}{\sigma_{i,N}(x,y)} \]
CONV NETS: EXTENSIONS

L2 Pooling & Local Contrast Normalization help learning more invariant representations!
CONV NETS: TYPICAL ARCHITECTURE

One stage (zoom)

Input Image

1\textsuperscript{st} stage

2\textsuperscript{nd} stage

3\textsuperscript{rd} stage

Linear Layer

Class Labels

Whole system

Filtering → Pooling → LCN
Since convolutions and sub-sampling are differentiable, we can use standard back-propagation.

Algorithm:
- Given a small mini-batch
  - FPROP
  - BPROP
  - PARAMETER UPDATE
CONV NETS: EXAMPLES

- Object category recognition
  Boureau et al. “Ask the locals: multi-way local pooling for image recognition” ICCV 2011

- Segmentation
  Turaga et al. “Maximin learning of image segmentation” NIPS 2009

- OCR
  Ciresan et al. “MCDNN for Image Classification” CVPR 2012

- Pedestrian detection
  Kavukcuoglu et al. “Learning convolutional feature hierarchies for visual recognition” NIPS 2010

- Robotics
  Sermanet et al. “Mapping and planning..with long range perception” IROS 2008
CONV NETS: LIMITATIONS

- requires lots of labeled data to train

- difficult optimization

- scalability
LIMITATIONS & SOLUTIONS

- requires lots of labeled data to train
+ unsupervised learning

- difficult optimization
+ layer-wise training

- scalability
+ distributed training
Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function
- Neural Networks for Vision: Convolutional & Tiled
- Unsupervised Training of Neural Networks
- Extensions:
  - semi-supervised / multi-task / multi-modal
- Comparison to Other Methods
  - boosting & cascade methods
  - probabilistic models
- Large-Scale Learning with Deep Neural Nets
BACK TO LOGISTIC REGRESSION

input → prediction → Error

target
Unsupervised Learning

input → prediction → Error
Unsupervised Learning

Q: How should we train the input-output mapping if we do not have target values?

A: Code has to retain information from the input but only if this is similar to training samples. By better representing only those inputs that are similar to training samples we hope to extract interesting structure (e.g., structure of manifold where data live).
Unsupervised Learning

Q: How to constrain the model to represent training samples better than other data points?
Unsupervised Learning

- reconstruct the **input** from the **code** & make code **compact** (auto-encoder with bottle-neck).

- reconstruct the input from the code & make code **sparse** (sparse auto-encoders)
  
  *see work in LeCun, Ng, Fergus, Lee, Yu's labs*

- add **noise** to the input or code (denoising auto-encoders)
  
  *see work in Y. Bengio, Lee's lab*

- make sure that the model defines a distribution that **normalizes** to 1 (RBM).
  
  *see work in Y. Bengio, Hinton, Lee, Salakthudinov's lab*
AUTO-ENCODERS NEURAL NETS

- input higher dimensional than code
- error: $\|\text{prediction} - \text{input}\|^2$
- training: back-propagation
SPARSE AUTO-ENCODERS

- sparsity penalty: $\|\text{code}\|_1$
- error: $\|\text{prediction} - \text{input}\|_2$
- loss: sum of square reconstruction error and sparsity
- training: back-propagation
SPARSE AUTO-ENCODERS

Le et al. "ICA with reconstruction cost." NIPS 2011
How To Use Unsupervised Learning

1) *Given unlabeled data, learn features*
How To Use Unsupervised Learning

1) Given unlabeled data, learn features
2) Use encoder to produce features and train another layer on the top
How To Use Unsupervised Learning

1) Given unlabeled data, learn features
2) Use encoder to produce features and train another layer on the top

Layer-wise training of a feature hierarchy
How To Use Unsupervised Learning

1) Given unlabeled data, learn features
2) Use encoder to produce features and train another layer on the top
3) Feed features to classifier & train just the classifier

Reduced overfitting since features are learned in unsupervised way!
How To Use Unsupervised Learning

1) Given unlabeled data, learn features
2) Use encoder to produce features and train another layer on the top
3) feed features to classifier & jointly train the whole system

Given enough data, this usually yields the best results: end-to-end learning!
Visualizing Learned Features

Q: can we interpret the learned features?
Visualizing Learned Features

Q: how are these images computed?

1\textsuperscript{st} layer features

2\textsuperscript{nd} layer features

Ranzato et al. "Sparse feature learning for DBNs" NIPS 2007
Visualizing Learned Features

reconstruction: $W h = W_1 h_1 + W_2 h_2 + \ldots$

<table>
<thead>
<tr>
<th>$8$</th>
<th>$\approx$</th>
<th>$8$</th>
</tr>
</thead>
</table>

| 0.9 | 0.7 | 0.5 | 1.0 |

$W$ shows what each code unit represents.
1st layer features

Ranzato et al. “Sparse feature learning for DBNs” NIPS 2007
Q: How about the second layer features?

A: Similarly, each second layer code unit can be visualized by taking its bases and then projecting those bases in image space through the first layer decoder.
Visualizing Learned Features

Q: How about the second layer features?

A: Similarly, each second layer code unit can be visualized by taking its bases and then projecting those bases in image space through the first layer decoder.

Missing edges have 0 weight.
Light gray nodes have zero value.
Example of Feature Learning

1\textsuperscript{st} layer features

2\textsuperscript{nd} layer features

Ranzato et al. “Sparse feature learning for DBNs” NIPS 2007
Outline

- **Neural Networks for Supervised Training**
  - Architecture
  - Loss function

- **Neural Networks for Vision: Convolutional & Tiled**

- **Unsupervised Training of Neural Networks**

- **Extensions:**
  - semi-supervised / multi-task / multi-modal

- **Comparison to Other Methods**
  - boosting & cascade methods
  - probabilistic models

- **Large-Scale Learning with Deep Neural Nets**
Semi-Supervised Learning
Semi-Supervised Learning

LOTS & LOTS OF UNLABELED DATA!!!
Semi-Supervised Learning

\[ \text{Loss} = \text{supervised\_error} + \text{unsupervised\_error} \]
Face detection is hard because of lighting, pose, but also occluding goggles.

Face detection could made be easier by face identification.

The identification task may help the detection task.
Multi-Task Learning

- Easy to add many error terms to loss function.
- Joint learning of related tasks yields better representations.

Example of architecture:

Collobert et al. “NLP (almost) from scratch” JMLR 2011
Audio and Video streams are often complimentary to each other.

E.g., audio can provide important clues to improve visual recognition, and vice versa.
Multi-Modal Learning

- Weak assumptions on input distribution
- Fully adaptive to data

Example of architecture:

Ngiam et al. "Multi-modal deep learning" ICML 2011
Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function
- Neural Networks for Vision: Convolutional & Tiled
- Unsupervised Training of Neural Networks
- Extensions:
  - semi-supervised / multi-task / multi-modal
- Comparison to Other Methods
  - boosting & cascade methods
  - probabilistic models
- Large-Scale Learning with Deep Neural Nets
Boosting & Forests

**Deep Nets:**
- single highly non-linear system
- “deep” stack of simpler modules
- all parameters are subject to learning

**Boosting & Forests:**
- sequence of “weak” (simple) classifiers that are linearly combined to produce a powerful classifier
- subsequent classifiers do not exploit representations of earlier classifiers, it’s a “shallow” linear mixture
- typically features are not learned
<table>
<thead>
<tr>
<th>Properties</th>
<th>Deep Nets</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive features</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Hierarchical features</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>End-to-end learning</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td>Leverage unlab. data</td>
<td>✔️</td>
<td>🧐</td>
</tr>
<tr>
<td>Easy to parallelize</td>
<td>🧐</td>
<td>✔️</td>
</tr>
<tr>
<td>Fast training</td>
<td>🧐</td>
<td>✔️</td>
</tr>
<tr>
<td>Fast at test time</td>
<td>🧐</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Deep Neural-Nets VS Probabilistic Models

**Deep Neural Nets:**
- mean-field approximations of intractable probabilistic models
- usually more efficient
- typically more unconstrained (partition function has to be replaced by other constraints, e.g. sparsity).

**Hierarchical Probabilistic Models (DBN, DBM, etc.):**
- in the most interesting cases, they are intractable
- they better deal with uncertainty
- they can be easily combined
Example: Auto-Encoder

**Neural Net:**

\[ Z = \sigma \left( W_e^T X + b_e \right) \]

reconstruction \[ \hat{X} = W_d Z + b_d \]

**Probabilistic Model** (Gaussian RBM):

\[ E[Z | X] = \sigma \left( W^T X + b_e \right) \]

\[ E[X | Z] = W Z + b_d \]
<table>
<thead>
<tr>
<th>Properties</th>
<th>Deep Nets</th>
<th>Probab. Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive features</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Hierarchical features</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>End-to-end learning</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Leverage unlab. data</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Models uncertainty</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Fast training</td>
<td>😊</td>
<td>✗</td>
</tr>
<tr>
<td>Fast at test time</td>
<td>😊</td>
<td>😞</td>
</tr>
</tbody>
</table>
Outline

- Neural Networks for Supervised Training
  - Architecture
  - Loss function

- Neural Networks for Vision: Convolutional & Tiled

- Unsupervised Training of Neural Networks

- Extensions:
  - semi-supervised / multi-task / multi-modal

- Comparison to Other Methods
  - boosting & cascade methods
  - probabilistic models

- Large-Scale Learning with Deep Neural Nets
Observation #1: more features always improve performance unless data is scarce.

Observation #2: deep learning methods have higher capacity and have the potential to model data better.

Q #1: Given lots of data and lots of machines, can we scale up deep learning methods?

Q #2: Will deep learning methods perform much better?
The Challenge

A Large Scale problem has:
- lots of training samples (>10M)
- lots of classes (>10K) and
- lots of input dimensions (>10K).

- best optimizer in practice is on-line SGD which is naturally sequential, hard to parallelize.
- layers cannot be trained independently and in parallel, hard to distribute
- model can have lots of parameters that may clog the network, hard to distribute across machines
Our Solution

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012

Ranzato
Our Solution

1\textsuperscript{st} machine

2\textsuperscript{nd} machine

3\textsuperscript{rd} machine

1\textsuperscript{st} layer

2\textsuperscript{nd} layer

input

Ranzato
Our Solution

1\textsuperscript{st} machine

2\textsuperscript{nd} machine

3\textsuperscript{rd} machine

MODEL PARALLELISM
Distributed Deep Nets

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012
Distributed Deep Nets

MODEL PARALLELISM + DATA PARALLELISM

input #1
input #2
input #3

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012
Asynchronous SGD

PARAMETER SERVER

1\textsuperscript{st} replica

2\textsuperscript{nd} replica

3\textsuperscript{rd} replica
Asynchronous SGD

PARAMETER SERVER

\[ \frac{\partial L}{\partial \theta_1} \]

1st replica

2nd replica

3rd replica
Asynchronous SGD

PARAMETER SERVER

$\theta_1$

$1^{st}$ replica

$2^{nd}$ replica

$3^{rd}$ replica
Asynchronous SGD

PARAMETER SERVER
(update parameters)
Asynchronous SGD

PARAMETER SERVER

\[
\frac{\partial L}{\partial \theta_2}
\]

1\textsuperscript{st} replica

2\textsuperscript{nd} replica

3\textsuperscript{rd} replica

Ranzato
Asynchronous SGD

PARAMETER SERVER

$\theta_2$

1$^{st}$ replica

2$^{nd}$ replica

3$^{rd}$ replica

Ranzato
Asynchronous SGD

PARAMETER SERVER
(update parameters)
Unsupervised Learning With 1B Parameters

**DATA:** 10M youtube (unlabeled) frames of size 200x200.

*Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012*
Deep Net:
- 3 stages
- each stage consists of local filtering, L2 pooling, LCN
  - 18x18 filters
  - 8 filters at each location
  - L2 pooling and LCN over 5x5 neighborhoods
- training jointly the three layers by:
  - reconstructing the input of each layer
  - sparsity on the code

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012
Unsupervised Learning With 1B Parameters

Deep Net:
- 3 stages
- each stage consists of local filtering, L2 pooling, LCN
  - 18x18 filters
  - 8 filters at each location
  - L2 pooling and LCN over 5x5 neighborhoods
- training jointly the three layers by:
  - reconstructing the input of each layer
  - sparsity on the code

1B parameters!!!
Validating Unsupervised Learning

The network has seen lots of objects during training, but without any label.

Q.: how can we validate unsupervised learning?

Q.: Did the network form any high-level representation? E.g., does it have any neuron responding for faces?

- build validation set with 50% faces, 50% random images
- study properties of neurons

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012
Validating Unsupervised Learning

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012
Top Images For Best Face Neuron
Best Input For Face Neuron

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012
Unsupervised + Supervised (ImageNet)

Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012
Object Recognition on ImageNet

<table>
<thead>
<tr>
<th>METHOD</th>
<th>ACCURACY %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weston &amp; Bengio 2011</td>
<td>9.3</td>
</tr>
<tr>
<td>Linear Classifier on deep features</td>
<td>13.1</td>
</tr>
<tr>
<td>Deep Net (from random)</td>
<td>13.6</td>
</tr>
<tr>
<td>Deep Net (from unsup.)</td>
<td><strong>15.8</strong></td>
</tr>
</tbody>
</table>

**IMAGENET v.2011 (16M images, 20K categories)**

*Le et al. “Building high-level features using large-scale unsupervised learning” ICML 2012*
Top Inputs After Supervision
Experiments: and many more...

- automatic speech recognition
- natural language processing
- biomed applications
- finance

Generic learning algorithm!!
References

Tutorials & Background Material


Convolutional Nets

Unsupervised Learning

- ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning. Le, Karpenko, Ngiam, Ng. In NIPS*2011
- Rifai, Vincent, Muller, Glorot, Bengio, Contracting Auto-Encoders: Explicit invariance during feature extraction, in: Proceedings of the Twenty-eight International Conference on Machine Learning (ICML'11), 2011

Multi-modal Learning

Locally Connected Nets
- Gregor, LeCun “Emergence of complex-like cells in a temporal product network with local receptive fields” Arxiv. 2009
- Ranzato, Mnih, Hinton “Generating more realistic images using gated MRF's” NIPS 2010
- Le, Ngiam, Chen, Chia, Koh, Ng “Tiled convolutional neural networks” NIPS 2010

Distributed Learning

Papers on Scene Parsing

Papers on Segmentation
Papers on Object Recognition

- Sermanet, LeCun: Traffic Sign Recognition with Multi-Scale Convolutional Networks, Proceedings of International Joint Conference on Neural Networks (IJCNN'11)
- Ciresan, Meier, Gambardella, Schmidhuber. Convolutional Neural Network Committees For Handwritten Character Classification. 11th International Conference on Document Analysis and Recognition (ICDAR 2011), Beijing, China.

Papers on Action Recognition

- Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis, Le, Zou, Yeung, Ng. In Computer Vision and Pattern Recognition (CVPR), 2011
Papers on Vision for Robotics


Deep Convex Nets & Deconv-Nets

- Zeiler, Taylor, Fergus "Adaptive Deconvolutional Networks for Mid and High Level Feature Learning." ICCV. 2011

Papers on Biological Inspired Vision

- Pinto, Doukhan, DiCarlo, Cox "A high-throughput screening approach to discovering good forms of biologically inspired visual representation." {PLoS} Computational Biology. 2009
Software & Links

Deep Learning website
- http://deeplearning.net/

C++ code for ConvNets
- http://eblearn.sourceforge.net/

Matlab code for R-ICA unsupervised algorithm

Python-based learning library
- http://deeplearning.net/software/theano/

Lush learning library which includes ConvNets
- http://lush.sourceforge.net/

Code used to generate demo for this tutorial
- http://cs.nyu.edu/~fergus/tutorials/deep_learning_cvpr12/
Acknowledgements

Quoc Le, Andrew Ng

Jeff Dean, Kai Chen, Greg Corrado, Matthieu Devin, Mark Mao, Rajat Monga, Paul Tucker, Samy Bengio

Yann LeCun, Pierre Sermanet, Clement Farabet