What's Wrong With Deep Learning?

Yann LeCun
Facebook AI Research &
Center for Data Science, NYU
yann@cs.nyu.edu
http://yann.lecun.com
The motivation for ConvNets and Deep Learning: end-to-end learning
- Integrating feature extractor, classifier, contextual post-processor

A bit of archeology: ideas that have been around for a while
- Kernels with stride, non-shared local connections, metric learning…
- “fully convolutional” training

What's missing from deep learning?
- 1. Theory
- 2. Reasoning, structured prediction
- 3. Memory, short-term/working/episodic memory
- 4. Unsupervised learning that actually works
Deep Learning = Learning Hierarchical Representations

**Traditional Pattern Recognition:** Fixed/Handcrafted Feature Extractor

**Mainstream Modern Pattern Recognition:** Unsupervised mid-level features

**Deep Learning:** Representations are hierarchical and trained
[Hubel & Wiesel 1962]:

- **simple cells** detect local features
- **complex cells** “pool” the outputs of simple cells within a retinotopic neighborhood.

Cognitron & Neocognitron [Fukushima 1974-1982]
The Mammalian Visual Cortex is Hierarchical

- The ventral (recognition) pathway in the visual cortex has multiple stages
  - Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations

[picture from Simon Thorpe]

[Gallant & Van Essen]
Deep Learning = Learning Hierarchical Representations

It's deep if it has more than one stage of non-linear feature transformation

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]
Early Networks [LeCun 85, 86]

Binary threshold units trained supervised with “target prop”

Hidden units compute a virtual target
Trained with Backprop. 320 examples.

- Convolutions with stride (subsampling)
- No separate pooling layers

### Network Architecture Performance

<table>
<thead>
<tr>
<th>Network Architecture</th>
<th>Links</th>
<th>Weights</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single layer network</td>
<td>2570</td>
<td>2570</td>
<td>80 %</td>
</tr>
<tr>
<td>Two layer network</td>
<td>3240</td>
<td>3240</td>
<td>87 %</td>
</tr>
<tr>
<td>Locally connected</td>
<td>1226</td>
<td>1226</td>
<td>88.5 %</td>
</tr>
<tr>
<td>Constrained network</td>
<td>2266</td>
<td>1132</td>
<td>94 %</td>
</tr>
<tr>
<td>Constrained network 2</td>
<td>5194</td>
<td>1060</td>
<td>98.4 %</td>
</tr>
</tbody>
</table>
Trained with Backprop.

USPS Zipcode digits: 7300 training, 2000 test.

Convolution with stride. No separate pooling.

10 output units

layer H3
30 hidden units

layer H2
12 x 16 = 192 hidden units

layer H1
12 x 64 = 768 hidden units

256 input units

fully connected
~ 300 links

fully connected
~ 6000 links

~ 40,000 links
from 12 kernels
5 x 5 x 8

~ 20,000 links
from 12 kernels
5 x 5
Filters-tanh $\rightarrow$ pooling $\rightarrow$ filters-tanh $\rightarrow$ pooling $\rightarrow$ filters-tanh
LeNet-1 Demo from 1993

Running on a 486 PC with an AT&T DSP32C add-on board (20 Mflops!)
Integrating Segmentation

Multiple Character Recognition
Multiple Character Recognition [Matan et al 1992]

- SDNN: Space Displacement Neural Net
- Also known as “replicated convolutional net”, or just ConvNet
  - (are we going to call this “fully convolutional net” now?)
- There is no such thing as a “fully connected layer”
- They are actually convolutional layers with 1x1 convolution kernels.
Multiple Character Recognition. Integrated Segmentation

- Trained with “semi synthetic” data
  - the individual character positions are known
- Training sample: a character painted with flanking characters or an inter-character space
Multiple Character Recognition. Integrated Segmentation
Word-level training with weak supervision [Matan et al 1992]

- Word-level training
- No labeling of individual characters
- How do we do the training?
- We need a “deformable part model”

ConvNet

5
4
3
2

window width of each classifier

Multiple classifiers
Spoken word recognition with trainable elastic word templates. First example of structured prediction on top of deep learning [Driancourt&Bottou 1991, Bottou 1991, Driancourt 1994]
Word-level training with elastic word models

- Isolated spoken word recognition
- Trainable elastic templates and trainable feature extraction
- Globally trained at the word level
- Elastic matching using dynamic time warping
  - Viterbi algorithm on a trellis.

Trainable Automatic Speech Recognition system with a convolutional net (TDNN) and dynamic time warping (DTW)

The feature extractor and the structured classifier are trained simultaneously in an integrated fashion.

with the LVQ2 Loss:
- Driancourt and Bottou's speech recognizer (1991)
- Bengio's speech recognizer (1992)
- Haffner's speech recognizer (1993)
Making every single module in the system trainable.

Every module is trained simultaneously so as to optimize a global loss function.

Includes the feature extractor, the recognizer, and the contextual post-processor (graphical model).

Problem: back-propagating gradients through the graphical model.
"Shallow" Structured Prediction

Energy function is linear in the parameters

\[ E(X, Y, Z) = \sum_i W_i^T h_i(X, Y, Z) \]

with the NLL Loss:
- Conditional Random Field
  [Lafferty, McCallum, Pereira 2001]

with Hinge Loss:
- Max Margin Markov Nets and Latent SVM
  [Taskar, Altun, Hofmann...]

with Perceptron Loss
- Structured Perceptron
  [Collins...]

Input:
- X

Latent Vars:
- Z1
- Z2
- Z3

Features:
- h(X,Y,Z)

Outputs:
- Y1
- Y2
- Y3
- Y4
Deep Structured Prediction

Energy function is linear in the parameters

\[ E(X, Y, Z) = \sum_i g_i(X, Y, Z, W_i) \]

Graph Transformer Networks
- [LeCun, Bottou, Bengio, Haffner 97, 98]
- NLL loss
- Perceptron loss

ConvNet

Outputs: Y1, Y2, Y3, Y4
Latent Vars: Z1, Z2, Z3
Input: X
Structured Prediction on top of Deep Learning

This example shows the structured perceptron loss.

In practice, we used negative log-likelihood loss.

Deployed in 1996 in check reading machines.
Graph transformer network trained to read check amounts. Trained globally with Negative-Log-Likelihood loss. 50% percent correct, 49% reject, 1% error (detectable later in the process. Fielded in 1996, used in many banks in the US and Europe. Processes an estimated 10% to 20% of all the checks written in the US.
Object Detection
Face Detection [Vaillant et al. 93, 94]

- ConvNet applied to large images
- Heatmaps at multiple scales
- Non-maximum suppression for candidates
- 6 second on a Sparcstation for 256x256 image
mid 2000s: state of the art results on face detection

<table>
<thead>
<tr>
<th>Data Set-→</th>
<th>TILTED</th>
<th>PROFILE</th>
<th>MIT+CMU</th>
</tr>
</thead>
<tbody>
<tr>
<td>False positives per image-→</td>
<td>4.42</td>
<td>26.9</td>
<td>0.47</td>
</tr>
<tr>
<td>Our Detector</td>
<td>90%</td>
<td>97%</td>
<td>67%</td>
</tr>
<tr>
<td>Jones &amp; Viola (tilted)</td>
<td>90%</td>
<td>95%</td>
<td>x</td>
</tr>
<tr>
<td>Jones &amp; Viola (profile)</td>
<td>x</td>
<td>70%</td>
<td>83%</td>
</tr>
</tbody>
</table>

[Osadchy et al. 2004] [Osadchy et al, JMLR 2007]
Simultaneous face detection and pose estimation
Semantic Segmentation
**ConvNets for Biological Image Segmentation**

- **Biological Image Segmentation**
  - [Ning et al. IEEE-TIP 2005]

- **Pixel labeling with large context using a convnet**

- **ConvNet takes a window of pixels and produces a label for the central pixel**

- **Cleanup using a kind of conditional random field (CRF)**
  - Similar to a field of expert, but conditional.

- **3D version for connectomics**
  - [Jain et al. 2007]

[Hadsell et al., J. Field Robotics 2009]
Long Range Vision with a Convolutional Net

Pre-processing (125 ms)
- Ground plane estimation
- Horizon leveling
- Conversion to YUV + local contrast normalization
- Scale invariant pyramid of distance-normalized image “bands”
Convolutional Net Architecture

- **100 features per 3x12x25 input window**
- **YUV image band**
  - 20-36 pixels tall,
  - 36-500 pixels wide
- **YUV input**

---

**CONVOLUTIONS (7x6)**

- 20@30x484
- 3@36x484

**MAX SUBSAMPLING (1x4)**

- 20@30x125

**CONVOLUTIONS (6x5)**

- 100@25x121
Scene Parsing/Labeling: Multiscale ConvNet Architecture

- Each output sees a large input context:
  - 46x46 window at full rez; 92x92 at ½ rez; 184x184 at ¼ rez
  - [7x7conv]-->[2x2pool]-->[7x7conv]-->[2x2pool]-->[7x7conv]->
  - Trained supervised on fully-labeled images
Method 1: majority over super-pixel regions

1. Input image
2. Super-pixel boundaries
3. Multi-scale ConvNet
4. Features from Convolutional net (d=768 per pixel)
5. "soft" categories scores
6. Majority Vote Over Superpixels
7. Categories aligned With region boundaries

[Farabet et al. IEEE T. PAMI 2013]
Scene Parsing/Labeling

[Farabet et al. ICML 2012, PAMI 2013]
Scene Parsing/Labeling on RGB+Depth Images

[Couprie, Farabet, Najman, LeCun ICLR 2013, ICIP 2013]
Scene Parsing/Labeling: Performance

Stanford Background Dataset [Gould 1009]: 8 categories

<table>
<thead>
<tr>
<th></th>
<th>Pixel Acc.</th>
<th>Class Acc.</th>
<th>CT (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gould et al. 2009 [14]</td>
<td>76.4%</td>
<td>-</td>
<td>10 to 600s</td>
</tr>
<tr>
<td>Munoz et al. 2010 [32]</td>
<td>76.9%</td>
<td>66.2%</td>
<td>12s</td>
</tr>
<tr>
<td>Tighe et al. 2010 [46]</td>
<td>77.5%</td>
<td>-</td>
<td>10 to 300s</td>
</tr>
<tr>
<td>Socher et al. 2011 [45]</td>
<td>78.1%</td>
<td>-</td>
<td>?</td>
</tr>
<tr>
<td>Kumar et al. 2010 [22]</td>
<td>79.4%</td>
<td>-</td>
<td>&lt; 600s</td>
</tr>
<tr>
<td>Lempitsky et al. 2011 [28]</td>
<td>81.9%</td>
<td>72.4%</td>
<td>&gt; 60s</td>
</tr>
<tr>
<td>singlescale convnet</td>
<td>66.0 %</td>
<td>56.5 %</td>
<td>0.35s</td>
</tr>
<tr>
<td>multiscale convnet</td>
<td>78.8 %</td>
<td>72.4%</td>
<td>0.6s</td>
</tr>
<tr>
<td>multiscale net + superpixels</td>
<td>80.4%</td>
<td>74.56%</td>
<td>0.7s</td>
</tr>
<tr>
<td>multiscale net + gPb + cover</td>
<td>80.4%</td>
<td>75.24%</td>
<td>61s</td>
</tr>
<tr>
<td>multiscale net + CRF on gPb</td>
<td>81.4%</td>
<td>76.0%</td>
<td>60.5s</td>
</tr>
</tbody>
</table>

[Rejected from CVPR 2012]
[Farabet et al. ICML 2012][Farabet et al. IEEE T. PAMI 2013]
# Scene Parsing/Labeling: Performance

<table>
<thead>
<tr>
<th></th>
<th>Pixel Acc.</th>
<th>Class Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Liu <em>et al.</em> 2009 [31]</td>
<td>74.75%</td>
<td>-</td>
</tr>
<tr>
<td>Tighe <em>et al.</em> 2010 [44]</td>
<td>76.9%</td>
<td>29.4%</td>
</tr>
<tr>
<td>raw multiscale net¹</td>
<td>67.9%</td>
<td>45.9%</td>
</tr>
<tr>
<td>multiscale net + superpixels¹</td>
<td>71.9%</td>
<td>50.8%</td>
</tr>
<tr>
<td>multiscale net + cover¹</td>
<td>72.3%</td>
<td>50.8%</td>
</tr>
<tr>
<td>multiscale net + cover²</td>
<td>78.5%</td>
<td>29.6%</td>
</tr>
</tbody>
</table>

- **SIFT Flow Dataset**
- **[Liu 2009]**: 33 categories

<table>
<thead>
<tr>
<th></th>
<th>Pixel Acc.</th>
<th>Class Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tighe <em>et al.</em> 2010 [44]</td>
<td>66.9%</td>
<td>7.6%</td>
</tr>
<tr>
<td>raw multiscale net¹</td>
<td>37.8%</td>
<td>12.1%</td>
</tr>
<tr>
<td>multiscale net + superpixels¹</td>
<td>44.1%</td>
<td>12.4%</td>
</tr>
<tr>
<td>multiscale net + cover¹</td>
<td>46.4%</td>
<td>12.5%</td>
</tr>
<tr>
<td>multiscale net + cover²</td>
<td>67.8%</td>
<td>9.5%</td>
</tr>
</tbody>
</table>

- **Barcelona dataset**
- **[Tighe 2010]**: 170 categories

[Farabet et al. IEEE T. PAMI 2012]
Scene Parsing/Labeling

[Farabet et al. ICML 2012, PAMI 2013]
No post-processing
Frame-by-frame
ConvNet runs at 50ms/frame on Virtex-6 FPGA hardware
But communicating the features over ethernet limits system performance
Then, two things happened...

- The ImageNet dataset [Fei-Fei et al. 2012]
  - 1.2 million training samples
  - 1000 categories

- Fast Graphical Processing Units (GPU)
  - Capable of 1 trillion operations/second
Very Deep ConvNet for Object Recognition

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic Fox (1.0); Eskimo Dog (0.6); White Wolf (0.4); Siberian Husky (0.4)
Layer 1: 3x96 kernels, RGB->96 feature maps, 11x11 Kernels, stride 4
Layer 1: 3x512 kernels, 7x7, 2x2 stride.

1: nn.SpatialConvolutionRing nInputPlane=3 nOutputPlane=512 kW*kH=7*7 dW*dH=2*2
• How the filters in the first layer learn
Deep Face

- [Taigman et al. CVPR 2014]
- Alignment
- ConvNet
- Metric Learning
Siamese Architecture and loss function

Loss function:
- Outputs corresponding to input samples that are neighbors in the neighborhood graph should be nearby.
- Outputs for input samples that are not neighbors should be far away from each other.

![Diagram showing Siamese Architecture and loss function](image)

Make this small

\[ D_{W} \uparrow \]
\[ ||G_{w}(x_{1}) - G_{w}(x_{2})|| \]

Make this large

\[ D_{W} \uparrow \]
\[ ||G_{w}(x_{1}) - G_{w}(x_{2})|| \]

Similar images (neighbors in the neighborhood graph)

Dissimilar images (non-neighbors in the neighborhood graph)
Learning Video Features with C3D

- **C3D Architecture**
  - 8 convolution, 5 pool, 2 fully-connected layers
  - 3x3x3 convolution kernels
  - 2x2x2 pooling kernels

- **Dataset: Sports-1M [Karpathy et al. CVPR’14]**
  - 1.1M videos of 487 different sport categories
  - Train/test splits are provided

---

Du Tran (1,2)  
Lubomir Bourdev (2)  
Rob Fergus (2,3)  
Lorenzo Torresani (1)  
Manohar Paluri (2)

(1) Dartmouth College, (2) Facebook AI Research, (3) New York University
### Sport Classification Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Nets</th>
<th>Clip hit@1</th>
<th>Video hit@1</th>
<th>Video hit@5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Video’s Single-Frame + Multires [19]</td>
<td>3 nets</td>
<td>42.4</td>
<td>60.0</td>
<td>78.5</td>
</tr>
<tr>
<td>Deep Video’s Slow Fusion [19]</td>
<td>1 net</td>
<td>41.9</td>
<td>60.9</td>
<td>80.2</td>
</tr>
<tr>
<td>C3D (trained from scratch)</td>
<td>1 net</td>
<td>44.9</td>
<td>60.0</td>
<td>84.4</td>
</tr>
<tr>
<td>C3D (fine-tuned from I380K pre-trained model)</td>
<td>1 net</td>
<td>46.1</td>
<td>61.1</td>
<td>85.2</td>
</tr>
</tbody>
</table>
Video Classification

- Using a spatio-temporal ConvNet

1. synchronized_skating: 0.99
2. ice_skating: 1.00
Video Classification

- Using a spatio-temporal ConvNet

airplane: 0.93
race: 0.01
explosion: 0.01
truck: 0.01
traveling: 0.01
Video Classification

- Spatio-temporal ConvNet
Now,
What's Wrong with Deep Learning?
Missing Some Theory
Why are ConvNets a good architecture?
- Scattering transform
- Mark Tygert's "complex ConvNet"

How many layers do we really need?
- Really?

How many effective free parameters are there in a large ConvNet
- The weights seem to be awfully redundant

What about Local Minima?
- Turns out almost all the local minima are equivalent
- Local minima are degenerate (very flat in most directions)
- Random matrix / spin glass theory comes to the rescue
- [Choromanska, Henaff, Mathieu, Ben Arous, LeCun AI-stats 2015]
Deep Nets with ReLUs: Objective Function is Piecewise Polynomial

If we use a hinge loss, delta now depends on label Yk:

\[ L(W) = \sum_P C_p(X, Y, W) \left( \prod_{(ij) \in P} W_{ij} \right) \]

Piecewise polynomial in W with random coefficients

A lot is known about the distribution of critical points of polynomials on the sphere with random (Gaussian) coefficients [Ben Arous et al.]

- High-order spherical spin glasses
- Random matrix theory

![Graph showing distribution of critical points](image)
Missing: Reasoning
Deep Learning systems can be assembled into energy models AKA factor graphs.

- Energy function is a sum of factors.
- Factors can embed whole deep learning systems.

- X: observed variables (inputs).
- Z: never observed (latent variables).
- Y: observed on training set (output variables).

Inference is energy minimization (MAP) or free energy minimization (marginalization) over Z and Y given an X.

- \( F(X,Y) = \text{MIN}_z E(X,Y,Z) \)
- \( F(X,Y) = -\log \text{SUM}_z \exp[-E(X,Y,Z)] \)
Energy-Based Learning [LeCun et al. 2006]

Push down on the energy of desired outputs
Push up on everything else

[LeCun et al 2006] “A tutorial on energy-based learning”
Stick a CRF on top of a ConvNet
Pose Estimation and Attribute Recovery with ConvNets

Pose-Aligned Network for Deep Attribute Modeling
[Zhang et al. CVPR 2014] (Facebook AI Research)

(a) Highest scoring results for people wearing glasses.

(b) Highest scoring results for people wearing a hat.

Body pose estimation [Tompson et al. ICLR, 2014]

Real-time hand pose recovery
[Tompson et al. Trans. on Graphics 14]
Person Detection and Pose Estimation

[Tompson, Goroshin, Jain, LeCun, Bregler CVPR 2015]
Person Detection and Pose Estimation

Start with a tree graphical model

**MRF** over spatial locations

Joint Distribution:

\[
P(f, s, e, w) = \frac{1}{Z} \prod_{i,j} \Psi(x_i, x_j) \prod_i \Phi(x_i, \tilde{x}_i)
\]
Start with a tree graphical model

... And approximate it

\[ b(f) = \Phi(f) \prod_i (\Phi(x_i) \ast \Psi(f | x_i) + c(f | x_i)) \]
(1) B. Sapp and B. Taskar. MODEC: Multimodel decomposition models for human pose estimation. CVPR'13
(2) S. Johnson and M. Everingham. Learning Effective Human Pose Estimation for Inaccurate Annotation. CVPR’11
Missing: Memory
Word Embedding in continuous vector spaces

- Bengio 2003
- Collobert & Weston 2010
- Word2Vec [Mikolov 2011]
- Predict a word from previous words and/or following words

Neural net of some kind

```
what are the major languages spoken in greece ?
```
Beijing – China + France = Paris
Embedding sentences into vector spaces

- Using a convolutional net or a recurrent net.

ConvNet or Recurrent Net

[sentence vector]

what are the major languages spoken in greece?
"Who did Clooney marry in 1987?"

Detection of Freebase entity in the question
what are bigos?
   ["stew"]
what are dallas cowboys colors?
   ["navy_blue", "royal_blue", "blue", "white", "silver"]
   ["blue", "navy_blue", "white", "royal_blue", "silver"]
how is egyptian money called?
   ["egyptian_pound"]
   ["egyptian_pound"]
what are fun things to do in sacramento ca?
   ["sacramento_zoo"]
   ["raging_waters_sacramento", "sutter_s_fort", "b_street_theatre", "sacramento_zoo", "california_state_capitol_museum", ...]
how are john terry's children called?
   ["georgie_john_terry", "summer_rose_terry"]
   ["georgie_john_terry", "summer_rose_terry"]
what are the major languages spoken in greece?
   ["greek_language", "albanian_language"]
   ["greek_language", "albanian_language"]
what was laura ingalls wilder famous for?
   ["writer", "author"]
   ["writer", "journalist", "teacher", "author"]
who plays sheldon cooper mother on the big bang theory?
  ["jim_parsons"] ["jim_parsons"]
who does peyton manning play football for?
  ["denver_broncos"] ["indianapolis_colts", "denver_broncos"]
who did vladimir lenin marry?
  ["nadezhda_krupskaya"] ["nadezhda_krupskaya"]
where was teddy roosevelt's house?
  ["new_york_city"] ["manhattan"]
who developed the tcp ip reference model?
  ["vint_cerf", "robert_e._kahn"] ["computer_scientist", "engineer"]
Every object, concept or “thought” can be represented by a vector

-0.2, 0.3, -4.2, 5.1, …... represent the concept “cat”

-0.2, 0.4, -4.0, 5.1, …... represent the concept “dog”

The vectors are similar because cats and dogs have many properties in common.

Reasoning consists in manipulating thought vectors

- Comparing vectors for question answering, information retrieval, content filtering
- Combining and transforming vectors for reasoning, planning, translating languages

Memory stores thought vectors

MemNN (Memory Neural Network) is an example

At FAIR we want to “embed the world” in thought vectors

We call this World2vec
But How can Neural Nets Remember Things?

- **Recurrent networks cannot remember things for very long**
  - The cortex only remember things for 20 seconds

- **We need a “hippocampus” (a separate memory module)**
  - LSTM [Hochreiter 1997], registers
  - **Memory networks** [Weston et al. 2014] (FAIR), associative memory
  - NTM [DeepMind 2014], “tape”.
Add a short-term memory to a network

I: (input feature map) – converts the incoming input to the internal feature representation.

G: (generalization) – updates old memories given the new input.

O: (output feature map) – produces a new output (in the feature representation space), given the new input and the current memory.

R: (response) – converts the output into the response format desired. For example, a textual response or an action.

<table>
<thead>
<tr>
<th>Method</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Fader et al., 2013)</td>
<td>0.54</td>
</tr>
<tr>
<td>(Bordes et al., 2014)</td>
<td>0.73</td>
</tr>
<tr>
<td>MemNN</td>
<td>0.71</td>
</tr>
<tr>
<td>MemNN (with BoW features)</td>
<td>0.79</td>
</tr>
</tbody>
</table>

Results on Question Answering Task

Fig. 2. An example story with questions correctly answered by a MemNN. The MemNN was trained on the simulation described in Section 4.2 and had never seen many of these words before, e.g. Bilbo, Frodo and Gollum.
Missing: Unsupervised Learning
Energy-Based Unsupervised Learning

Push down on the energy of desired outputs
Push up on everything else
Seven Strategies to Shape the Energy Function

1. build the machine so that the volume of low energy stuff is constant
   - PCA, K-means, GMM, square ICA

2. push down of the energy of data points, push up everywhere else
   - Max likelihood (needs tractable partition function)

3. push down of the energy of data points, push up on chosen locations
   - contrastive divergence, Ratio Matching, Noise Contrastive Estimation, Minimum Probability Flow

4. minimize the gradient and maximize the curvature around data points
   - score matching

5. train a dynamical system so that the dynamics goes to the manifold
   - denoising auto-encoder

6. use a regularizer that limits the volume of space that has low energy
   - Sparse coding, sparse auto-encoder, PSD

7. if $E(Y) = \|Y - G(Y)\|^2$, make $G(Y)$ as "constant" as possible.
   - Contracting auto-encoder, saturating auto-encoder
1. build the machine so that the volume of low energy stuff is constant

PCA, K-means, GMM, square ICA...

**PCA**

\[ E(Y) = \| W^T W Y - Y \|^2 \]

**K-Means**, 
Z constrained to 1-of-K code

\[ E(Y) = \min_z \sum_i \| Y - W_i Z_i \|^2 \]
#6. use a regularizer that limits the volume of space that has low energy

- Sparse coding, sparse auto-encoder, Predictive Sparse Decomposition
### Energy Functions of Various Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Encoder</th>
<th>Decoder</th>
<th>Energy</th>
<th>Loss</th>
<th>Pull-up</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PCA</strong></td>
<td>$W'Y$</td>
<td>$WZ$</td>
<td>$|Y-WZ|^2$</td>
<td>$F(Y)$</td>
<td>dimens.</td>
</tr>
<tr>
<td><strong>Autoencoder</strong></td>
<td>$\sigma(WeY)$</td>
<td>$WdZ$</td>
<td>$|Y-WZ|^2$</td>
<td>$F(Y)$</td>
<td>dimens.</td>
</tr>
<tr>
<td><strong>Sparse Coding</strong></td>
<td>$\sigma(WeZ)$</td>
<td>$WdZ$</td>
<td>$|Y-WZ|^2$</td>
<td>$F(Y)$</td>
<td>sparsity</td>
</tr>
<tr>
<td><strong>K-Means</strong></td>
<td>--</td>
<td>$WZ$</td>
<td>$|Y-WZ|^2$</td>
<td>$F(Y)$</td>
<td>1-of-N code</td>
</tr>
</tbody>
</table>

2 dimensional toy dataset: spiral

Visualizing energy surface

(black = low, white = high)
Dictionary Learning With Fast Approximate Inference: Sparse Auto-Encoders
How to Speed Up Inference in a Generative Model?

- Factor Graph with an asymmetric factor
  - Inference $Z \rightarrow Y$ is easy
    - Run $Z$ through deterministic decoder, and sample $Y$
  - Inference $Y \rightarrow Z$ is hard, particularly if Decoder function is many-to-one
    - MAP: minimize sum of two factors with respect to $Z$
      - $Z^* = \text{argmin}_z \text{ Distance}[\text{Decoder}(Z), Y] + \text{FactorB}(Z)$

Examples: K-Means (1 of K), Sparse Coding (sparse), Factor Analysis
Sparse linear reconstruction

Energy = reconstruction_error + code_prediction_error + code_sparsity

\[ E(Y^i, Z) = \| Y^i - W_d Z \|^2 + \lambda \sum_j |z_j| \]

\( Y \rightarrow \hat{Z} = \arg \min_Z E(Y, Z) \)

Inference is expensive: ISTA/FISTA, CGIHT, coordinate descent....
#6. use a regularizer that limits the volume of space that has low energy

- Sparse coding, sparse auto-encoder, Predictive Sparse Decomposition
Examples: most ICA models, Product of Experts
Train a “simple” feed-forward function to predict the result of a complex optimization on the data points of interest.

1. Find optimal $Z_i$ for all $Y_i$; 2. Train Encoder to predict $Z_i$ from $Y_i$
Learning to Perform Approximate Inference: Predictive Sparse Decomposition Sparse Auto-Encoders
Sparse auto-encoder: Predictive Sparse Decomposition (PSD)

[Kavukcuoglu, Ranzato, LeCun, 2008 → arXiv:1010.3467],

Prediction the optimal code with a trained encoder

Energy = reconstruction_error + code_prediction_error + code_sparsity

\[
E(Y^i, Z) = \|Y^i - W_d Z\|^2 + \|Z - g_e(W_e, Y^i)\|^2 + \lambda \sum_j |z_j|
\]

\[
g_e(W_e, Y^i) = \text{shrinkage}(W_e Y^i)
\]
Regularized Encoder-Decoder Model (auto-Encoder) for Unsupervised Feature Learning

- **Encoder**: computes feature vector $Z$ from input $X$
- **Decoder**: reconstructs input $X$ from feature vector $Z$
- **Feature vector**: high dimensional and regularized (e.g. sparse)
- **Factor graph with energy function $E(X,Z)$ with 3 terms:**
  - Linear decoding function and reconstruction error
  - Non-Linear encoding function and prediction error term
  - Pooling function and regularization term (e.g. sparsity)

$$E(Y,Z) = \| Y - W_d Z \|^2 + \| Z - g_e(W_e, Y) \|^2 + \sum_j \sqrt{\sum_{k \in P_j} Z_k^2}$$

INPUT $Y$

$g_e(W_e, Y^i)$

$\| Y^i - \tilde{Y} \|^2$

$W_d Z$

$Z$

$Z - \tilde{Z}$

$\| Z - \tilde{Z} \|^2$

FEATURES

$\sqrt{\sum Z_k^2}$

$\lambda \sum .$

L2 norm within each pool
Basis functions (and encoder matrix) are digit parts
Training on natural images patches.

- 12x12
- 256 basis functions
Learned Features on natural patches: V1-like receptive fields
Learning to Perform Approximate Inference
LISTA
ISTA/FISTA: iterative algorithm that converges to optimal sparse code

INPUT

\[ Z(t + 1) = \text{Shrinkage}_{\lambda/L} \left[ Z(t) - \frac{1}{L} W_d^T (W_d Z(t) - Y) \right] \]

ISTA/FISTA reparameterized:

\[ Z(t + 1) = \text{Shrinkage}_{\lambda/L} \left[ W_e^T Y + S Z(t) \right] ; \quad W_e = \frac{1}{L} W_d; \quad S = I - \frac{1}{L} W_d^T W_d \]

LISTA (Learned ISTA): learn the \( W_e \) and \( S \) matrices to get fast solutions

[Gregor & LeCun, ICML 2010], [Bronstein et al. ICML 2012], [Rolfe & LeCun ICLR 2013]
Think of the FISTA flow graph as a recurrent neural net where \( W_e \) and \( S \) are trainable parameters.

Time-Unfold the flow graph for \( K \) iterations.

Learn the \( W_e \) and \( S \) matrices with “backprop-through-time”.

Get the best approximate solution within \( K \) iterations.

LISTA: Train \( W_e \) and \( S \) matrices to give a good approximation quickly.
Learning ISTA (LISTA) vs ISTA/FISTA

- **Error** vs **Number of LISTA or FISTA iterations**

Graph showing the reconstruction error for different iterative methods with varying numbers of iterations. The legend includes:
- FISTA (4x)
- FISTA (1x)
- LISTA (4x)
- LISTA (1x)
LISTA with partial mutual inhibition matrix

Proportion of S matrix elements that are non zero

Smallest elements removed
Learning Coordinate Descent (LcoD): faster than LISTA

- CoD (4x)
- CoD (1x)
- LCoD (4x)
- LCoD (1x)
Discriminative Recurrent Sparse Auto-Encoder (DrSAE)

- Rectified linear units
- Classification loss: cross-entropy
- Reconstruction loss: squared error
- Sparsity penalty: L1 norm of last hidden layer
- Rows of $W_d$ and columns of $W_e$ constrained in unit sphere

[Rolfe & LeCun ICLR 2013]
DrSAE Discovers manifold structure of handwritten digits

Image = prototype + sparse sum of “parts” (to move around the manifold)
Convolutional Sparse Coding

Replace the dot products with dictionary element by convolutions.

- Input Y is a full image
- Each code component Z_k is a feature map (an image)
- Each dictionary element is a convolution kernel

Regular sparse coding

\[ E(Y, Z) = \|Y - \sum_k W_k Z_k\|^2 + \alpha \sum_k |Z_k| \]

Convolutional S.C.

\[ E(Y, Z) = \|Y - \sum_k W_k * Z_k\|^2 + \alpha \sum_k |Z_k| \]

“deconvolutional networks” [Zeiler, Taylor, Fergus CVPR 2010]
Convolutional PSD: Encoder with a soft sh() Function

- Convolutional Formulation
  - Extend sparse coding from **PATCH** to **IMAGE**

\[
\mathcal{L}(x, z, D) = \frac{1}{2} \| x - \sum_{k=1}^{K} D_k * z_k \|_2^2 + \sum_{k=1}^{K} \| z_k - f(W^k * x) \|_2^2 + |z|_1
\]

- **PATCH** based learning
- **CONVOLUTIONAL** learning
Filters and Basis Functions obtained with 1, 2, 4, 8, 16, 32, and 64 filters.
Phase 1: train first layer using PSD

\[ \|Y^i - \hat{Y}\|^2 \]

\[ W_d Z \]

\[ \|Z - \hat{Z}\|^2 \]

\[ g_e(W_e, Y^i) \]

\[ |z_j| \]

\[ \lambda \sum \]
Phase 1: train first layer using PSD
Phase 2: use encoder + absolute value as feature extractor
Phase 1: train first layer using PSD

Phase 2: use encoder + absolute value as feature extractor

Phase 3: train the second layer using PSD
Phase 1: train first layer using PSD
Phase 2: use encoder + absolute value as feature extractor
Phase 3: train the second layer using PSD
Phase 4: use encoder + absolute value as 2nd feature extractor
Using PSD to Train a Hierarchy of Features

- **Phase 1**: train first layer using PSD
- **Phase 2**: use encoder + absolute value as feature extractor
- **Phase 3**: train the second layer using PSD
- **Phase 4**: use encoder + absolute value as 2\textsuperscript{nd} feature extractor
- **Phase 5**: train a supervised classifier on top
- **Phase 6 (optional)**: train the entire system with supervised back-propagation

\[ Y \quad \left| z_j \right| \quad g_e(W_e, Y^i) \quad \left| z_j \right| \quad g_e(W_e, Y^i) \quad \text{classifier} \]
Pedestrian Detection: INRIA Dataset. Miss rate vs false positives

[Kavukcuoglu et al. NIPS 2010] [Sermanet et al. ArXiv 2012]
Unsupervised Learning: Invariant Features
Unsupervised PSD ignores the spatial pooling step.

Could we devise a similar method that learns the pooling layer as well?

Idea [Hyvarinen & Hoyer 2001]: group sparsity on pools of features

- Minimum number of pools must be non-zero
- Number of features that are on within a pool doesn't matter
- Pools tend to regroup similar features

\[
E(Y, Z) = \| Y - W_d Z \|_2^2 + \| Z - g_e(W_e, Y) \|_2^2 + \sum_j \sqrt{\sum_{k \in P_j} Z_k^2}
\]
Learning Invariant Features with L2 Group Sparsity

- Idea: features are pooled in group.
  - Sparsity: sum over groups of L2 norm of activity in group.

- [Hyvärinen Hoyer 2001]: “subspace ICA”
  - decoder only, square

- [Welling, Hinton, Osindero NIPS 2002]: pooled product of experts
  - encoder only, overcomplete, log student-T penalty on L2 pooling

- [Kavukcuoglu, Ranzato, Fergus LeCun, CVPR 2010]: Invariant PSD
  - encoder-decoder (like PSD), overcomplete, L2 pooling

- [Le et al. NIPS 2011]: Reconstruction ICA
  - Same as [Kavukcuoglu 2010] with linear encoder and tied decoder

  - Locally-connect non shared (tiled) encoder-decoder

INPUT

Encoder only (PoE, ICA), Decoder Only or Encoder-Decoder (iPSD, RICA)

Y

SIMPLE FEATURES

Z

L2 norm within each pool

\[ \sqrt{\left( \sum Z^2_k \right)} \]

\[ \lambda \sum \]
Groups are local in a 2D Topographic Map

- The filters arrange themselves spontaneously so that similar filters enter the same pool.
- The pooling units can be seen as complex cells.
- Outputs of pooling units are invariant to local transformations of the input.
  - For some it's translations, for others rotations, or other transformations.
Training on 115x115 images. Kernels are 15x15 (not shared across space!)

- [Gregor & LeCun 2010]
- Local receptive fields
- No shared weights
- 4x overcomplete
- L2 pooling
- Group sparsity over pools

Decoder

Reconstructed Input

(Inferred) Code

Predicted Code

Input

Encoder
Image-level training, local filters but no weight sharing

Training on 115x115 images. Kernels are 15x15 (not shared across space!)
Topographic Maps

119x119 Image Input
100x100 Code
20x20 Receptive field size
sigma=5

K. Obermayer and GG Blasdel. Journal of Neuroscience, Vol 13, 4114-4129 (Monkey)

Michael C. Crair, et. al. The Journal of Neurophysiology Vol. 77 No. 6 June 1997, pp. 3381-3385 (Cat)
Image-level training, local filters but no weight sharing

- Color indicates orientation (by fitting Gabors)
- Replace the L1 sparsity term by a lateral inhibition matrix
- Easy way to impose some structure on the sparsity

\[
\min_{W, Z} \sum_{x \in X} \|Wz - x\|^2 + |z^T S z|
\]

[Gregor, Szlam, LeCun NIPS 2011]
Each edge in the tree indicates a zero in the S matrix (no mutual inhibition)
Sij is larger if two neurons are far away in the tree
Non-zero values in S form a ring in a 2D topology

Input patches are high-pass filtered
Sparse Auto-Encoder with “Slow Feature” Penalty

Supervised filters CIFAR10

spare conv. auto-encoder

slow & sparse conv. auto-encoder
Trained on YouTube videos

[Goroshin et al. Arxiv:1412.6056]
Object is cross-product of object type and instantiation parameters

Mapping units [Hinton 1981], capsules [Hinton 2011]

[Karol Gregor et al.]
What-Where Auto-Encoder Architecture

Decoder

\[
\begin{array}{cccc}
S^t & S^{t-1} & S^{t-2} & \text{Predicted input} \\
W^1 & W^1 & W^1 & W^2 \\
C_1^t & C_1^{t-1} & C_1^{t-2} & C_2^t \\
\end{array}
\]

Inferred code

Predicted code

Encoder

\[
\begin{array}{cccc}
S^t & S^{t-1} & S^{t-2} & \text{Input} \\
W^1 & W^1 & W^1 & W^2 \\
C_1^t & C_1^{t-1} & C_1^{t-2} & C_2^t \\
\end{array}
\]

\[
\begin{array}{ccc}
C_1^t & C_1^{t-1} & C_1^{t-2} \\
f \circ \tilde{W}^1 & f \circ \tilde{W}^1 & f \circ \tilde{W}^1 \\
\end{array}
\]

\[
\begin{array}{c}
C_2^t \\
f \\
\end{array}
\]
Low-Level Filters Connected to Each Complex Cell

C1
(where)

C2
(what)
Integrated Supervised & Unsupervised Learning

[Zhao, Mathieu, LeCun arXiv:1506.02351]

Stacked What-Where Auto-Encoder

<table>
<thead>
<tr>
<th>model</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional Kernel Networks [18]</td>
<td>62.32%</td>
</tr>
<tr>
<td>HMP [1]</td>
<td>64.5%</td>
</tr>
<tr>
<td>NOMP [17]</td>
<td>67.9%</td>
</tr>
<tr>
<td>Multi-task Bayesian Optimization [31]</td>
<td>70.1%</td>
</tr>
<tr>
<td>Zero-bias ConvNets + ADCU [22]</td>
<td>70.2%</td>
</tr>
<tr>
<td>Exemplar ConvNets [2]</td>
<td>72.8%</td>
</tr>
<tr>
<td>WWAE-4layer</td>
<td>174.80%</td>
</tr>
</tbody>
</table>
The End
The bAbI Tasks

Questions that an AI system ought to be able to answer
Our first task consists of questions where a single supporting fact, previously given, provides the answer. We test simplest cases of this, by asking for the location of a person. A small sample of the task is thus:

John is in the playground.
Bob is in the office.
Where is John? A: playgroun

This kind of synthetic data was already used with MemNNs. It can be considered the simplest case of some real world QA datasets such as in Fader et al., ‘13.
A harder task is to answer questions where two supporting statements have to be chained to answer the question:

John is in the playground.
Bob is in the office.
John picked up the football.
Bob went to the kitchen.
Where is the football? A: playground
Where was Bob before the kitchen? A: office

E.g. to answer the first question Where is the football? both John picked up the football and John is in the playground are supporting facts.

Again, this kind of task was already used with MemNNs.
Note that, to show the difficulty of these tasks for a learning machine with no other knowledge we can shuffle the letters of the alphabet and produce equivalent datasets:

Sbdm ip im vdu yonrckblms.
Abf ip im vdu bhhigu.
Sbdm yigaus ly vdu hbbvfnoo.
Abf zumv vb vdu aivgdum.
Mduku ip vdu hbbvfnoo?
A:yonrckblms
Mduku znp Abf fuhbkv vdu aivgdum?
A:bhhigu
Similarly, one can make a task with three supporting facts:

John picked up the apple. 
John went to the office. 
John went to the kitchen. 
John dropped the apple. 

Where was the apple before the kitchen? 
A: office

The first three statements are all required to answer this.
To answer questions the ability to differentiate and recognize subjects and objects is crucial.

We consider the extreme case: sentences feature re-ordered words:

- The office is north of the bedroom.
- The bedroom is north of the bathroom.

What is north of the bedroom? A: office
What is the bedroom north of? A: bathroom

Note that the two questions above have exactly the same words, but in a different order, and different answers.

So a bag-of-words will not work.
Similarly, sometimes one needs to differentiate three separate arguments, such as in the following task:

Mary gave the cake to Fred.
Fred gave the cake to Bill.
Jeff was given the milk by Bill.
Who gave the cake to Fred? A: Mary
Who did Fred give the cake to? A: Bill
What did Jeff receive? A: milk
Who gave the milk? A: Bill

The last question is potentially the hardest for a learner as the first two can be answered by providing the actor that is not mentioned in the question.
(6) Yes/No Questions

This task tests, in the simplest case possible (with a single supporting fact) the ability of a model to answer true/false type questions:

- John is in the playground.
- Daniel picks up the milk.
- Is John in the classroom? A: no
- Does Daniel have the milk? A: yes
(7) Counting

This task tests the ability of the QA system to perform simple counting operations, by asking about the number of objects with a certain property:

Daniel picked up the football.  
Daniel dropped the football.  
Daniel got the milk.  
Daniel took the apple.  
How many objects is Daniel holding?  
A: two
(8) Lists/Sets

While many of our tasks are designed to have single word answers for simplicity, this task tests the ability to produce a set of single word answers in the form of a list:

- Daniel picks up the football.
- Daniel drops the newspaper.
- Daniel picks up the milk.

What is Daniel holding? A: milk, football

The task above can be seen as a QA task related to database search. Note that we could also consider the following question types:

**Intersection:** Who is in the park carrying food?

**Union:** Who has milk or cookies?

**Set difference:** Who is in the park apart from Bill?

However, we leave those for future work.
We test one of the simplest types of negation, that of supporting facts that imply a statement is false:

- Sandra travelled to the office.
- Fred is no longer in the office.
- Is Fred in the office? A: no
- Is Sandra in the office? A: yes

The Yes/No task (6) is a prerequisite.

Slightly harder: we could add things like “Is Fred with Sandra?”
This task tests if we can model statements that describe possibilities rather than certainties:

- John is either in the classroom or the playground.
- Sandra is in the garden.
- Is John in the classroom? A: maybe
- Is John in the office? A: no

The Yes/No task (6) is a prerequisite.

Slightly harder: we could add things like “Is John with Sandra?”
(11) Basic Coreference

- This task tests the simplest type of coreference, that of detecting the nearest referent, for example:

  Daniel was in the kitchen.
  Then he went to the studio.
  Sandra was in the office.
  Where is Daniel? A: studio

Next level of difficulty: flip order of last two statements, and it has to learn the difference between ‘he’ and ‘she’.

**Much harder difficulty:** adapt a real coref dataset into a question answer format.
This task tests referring to multiple subjects in a single statement, for example:

Mary and Jeff went to the kitchen. Then Jeff went to the park. Where is Mary? A:kitchen
This task tests coreference in the case where the pronoun can refer to multiple actors:

Daniel and Sandra journeyed to the office.
Then they went to the garden.
Sandra and John travelled to the kitchen.
After that they moved to the hallway.
Where is Daniel? A: garden
(14) Time manipulation

- While our tasks so far have included time implicitly in the order of the statements, this task tests understanding the use of time expressions within the statements:

> In the afternoon Julie went to the park. Yesterday Julie was at school. Julie went to the cinema this evening. Where did Julie go after the park? A: cinema

**Much harder difficulty:** adapt a real time expression labeling dataset into a question answer format, e.g. Uzzaman et al., ‘12.
This task tests basic deduction via inheritance of properties:

- Sheep are afraid of wolves.
- Cats are afraid of dogs.
- Mice are afraid of cats.
- Gertrude is a sheep.

What is Gertrude afraid of? A: wolves

Deduction should prove difficult for MemNNs because it effectively involves search, although our setup might be simple enough for it.
(16) Basic Induction

- This task tests basic induction via inheritance of properties:

Lily is a swan.
Lily is white.
Greg is a swan.
What color is Greg? A: white

Induction should prove difficult for MemNNs because it effectively involves search, although our setup might be simple enough for it.
This task tests spatial reasoning, one of many components of the classical SHRDLU system:

The triangle is to the right of the blue square. The red square is on top of the blue square. The red sphere is to the right of the blue square.
Is the red sphere to the right of the blue square? A:yes
Is the red square to the left of the triangle? A:yes

The Yes/No task (6) is a prerequisite.
(18) Reasoning about size

This task requires reasoning about relative size of objects and is inspired by the commonsense reasoning examples in the Winograd schema challenge:

- The football fits in the suitcase.
- The suitcase fits in the cupboard.
- The box of chocolates is smaller than the football.

Will the box of chocolates fit in the suitcase?

A: yes

Tasks 3 (three supporting facts) and 6 (Yes/No) are prerequisites.
In this task the goal is to find the path between locations:

The kitchen is north of the hallway.
The den is east of the hallway.
How do you go from den to kitchen?
A: west, north

This is going to prove difficult for MemNNs because it effectively involves search.
(The original MemNN can also output only one word \( \downarrow \))
(20) Reasoning about Agent’s Motivations

- This task tries to ask *why* an agent performs a certain action.
- It addresses the case of actors being in a given state (hungry, thirsty, tired, ...) and the actions they then take:

| John is hungry.  
| John goes to the kitchen.  
| John eats the apple.  
| Daniel is hungry.  
| Where does Daniel go? A:kitchen  
| Why did John go to the kitchen? A:hungry  


One way of solving these tasks: Memory Networks!!

MemNNs have four component networks (which may or may not have shared parameters):

- **I**: (input feature map) this converts incoming data to the internal feature representation.
- **G**: (generalization) this updates memories given new input.
- **O**: this produces new output (in feature representation space) given the memories.
- **R**: (response) converts output O into a response seen by the outside world.
Experiments

- **Protocol**: 1000 training QA pairs, 1000 for test.

“Weakly supervised” methods:

- Ngram baseline, uses bag of Ngram features from sentences that share a word with the question.
- LSTM

**Fully supervised methods** (for train data, have supporting facts labeled):

- Original MemNNs, *and all our variants*. 
<table>
<thead>
<tr>
<th>TASK</th>
<th>N-grams</th>
<th>LSTM</th>
<th>MemNN (Weston et al., 2014)</th>
<th>MemNN + AM</th>
<th>MemNN + AM + NG</th>
<th>MemNN + AM + NL</th>
<th>MemNN + AM + NL + ML</th>
<th>No. of ex. req. ≥ 95</th>
<th>MultiTask Training</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 - Single Supporting Fact</td>
<td>36</td>
<td>50</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>250 ex.</td>
<td>100</td>
</tr>
<tr>
<td>3.2 - Two Supporting Facts</td>
<td>2</td>
<td>20</td>
<td>100</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>500 ex.</td>
<td>100</td>
</tr>
<tr>
<td>3.3 - Three Supporting Facts</td>
<td>7</td>
<td>20</td>
<td>20</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>500 ex.</td>
<td>98</td>
</tr>
<tr>
<td>3.4 - Two Arg. Relations</td>
<td>50</td>
<td>61</td>
<td>71</td>
<td>69</td>
<td>73</td>
<td>73</td>
<td>100</td>
<td>500 ex.</td>
<td>80</td>
</tr>
<tr>
<td>3.5 - Three Arg. Relations</td>
<td>20</td>
<td>70</td>
<td>83</td>
<td>83</td>
<td>86</td>
<td>86</td>
<td>98</td>
<td>1000 ex.</td>
<td>99</td>
</tr>
<tr>
<td>3.6 - Yes/No Questions</td>
<td>49</td>
<td>48</td>
<td>47</td>
<td>52</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3.7 - Counting</td>
<td>52</td>
<td>49</td>
<td>68</td>
<td>86</td>
<td>83</td>
<td>83</td>
<td>90</td>
<td>FAIL</td>
<td>86</td>
</tr>
<tr>
<td>3.8 - Lists/Sets</td>
<td>40</td>
<td>45</td>
<td>77</td>
<td>88</td>
<td>94</td>
<td>94</td>
<td>91</td>
<td>FAIL</td>
<td>93</td>
</tr>
<tr>
<td>3.9 - Simple Negation</td>
<td>62</td>
<td>64</td>
<td>65</td>
<td>71</td>
<td>63</td>
<td>63</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3.10 - Indefinite Knowledge</td>
<td>45</td>
<td>44</td>
<td>59</td>
<td>57</td>
<td>97</td>
<td>97</td>
<td>98</td>
<td>1000 ex.</td>
<td>98</td>
</tr>
<tr>
<td>3.11 - Basic Coreference</td>
<td>29</td>
<td>72</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>250 ex.</td>
<td>100</td>
</tr>
<tr>
<td>3.12 - Conjunction</td>
<td>9</td>
<td>74</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>250 ex.</td>
<td>100</td>
</tr>
<tr>
<td>3.13 - Compound Coreference</td>
<td>26</td>
<td>94</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>250 ex.</td>
<td>100</td>
</tr>
<tr>
<td>3.15 - Basic Deduction</td>
<td>20</td>
<td>21</td>
<td>74</td>
<td>73</td>
<td>100</td>
<td>77</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3.16 - Basic Induction</td>
<td>43</td>
<td>23</td>
<td>27</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3.17 - Positional Reasoning</td>
<td>46</td>
<td>51</td>
<td>54</td>
<td>46</td>
<td>49</td>
<td>57</td>
<td>60</td>
<td>FAIL</td>
<td>72</td>
</tr>
<tr>
<td>3.18 - Size Reasoning</td>
<td>52</td>
<td>52</td>
<td>57</td>
<td>74</td>
<td>54</td>
<td>15</td>
<td>36</td>
<td>FAIL</td>
<td>19</td>
</tr>
<tr>
<td>3.19 - Path Finding</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>15</td>
<td>36</td>
<td>100</td>
<td>93</td>
</tr>
<tr>
<td>3.20 - Agent’s Motivations</td>
<td>76</td>
<td>91</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>250 ex.</td>
<td>100</td>
</tr>
<tr>
<td>Mean Performance</td>
<td>34</td>
<td>49</td>
<td>75</td>
<td>79</td>
<td>83</td>
<td>87</td>
<td>93</td>
<td>93</td>
<td>92</td>
</tr>
</tbody>
</table>
## Action Recognition Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baselines</strong></td>
<td></td>
</tr>
<tr>
<td>Imagenet</td>
<td>68.8</td>
</tr>
<tr>
<td>iDT</td>
<td>76.2</td>
</tr>
<tr>
<td><strong>Use raw pixel inputs</strong></td>
<td></td>
</tr>
<tr>
<td>Deep networks [19]</td>
<td>65.4</td>
</tr>
<tr>
<td>Spatial stream network [36]</td>
<td>72.6</td>
</tr>
<tr>
<td>LRCN [7]</td>
<td>71.1</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>75.8</td>
</tr>
<tr>
<td>C3D (1 net)</td>
<td>82.3</td>
</tr>
<tr>
<td>C3D (3 nets)</td>
<td><strong>85.2</strong></td>
</tr>
<tr>
<td>iDT with Fisher vector [31]</td>
<td>87.9</td>
</tr>
<tr>
<td>Temporal stream network [36]</td>
<td>83.7</td>
</tr>
<tr>
<td>Two-stream networks [36]</td>
<td>88.0</td>
</tr>
<tr>
<td>LRCN [7]</td>
<td>82.9</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>84.3</td>
</tr>
<tr>
<td>Multi-skip feature stacking [26]</td>
<td>89.1</td>
</tr>
<tr>
<td><strong>Use optical flows</strong></td>
<td></td>
</tr>
<tr>
<td>C3D (3 nets) + iDT</td>
<td><strong>90.4</strong></td>
</tr>
</tbody>
</table>