Dynamic Routing Between Capsules

Sara Sabour, Nicholas Frosst, and Geoffrey Hinton

NIPS 2017

Erfan Loweimi
Peter Bell and Steve Renals

Centre for Speech Technology Research (CSTR)
Listen! 23, 4, 2019
Capsule Papers ...

[PDF] Transforming Auto-encoders - University of Toronto Computer Science
by GE Hinton - Cited by 356 - Related articles
Three capsules of a transforming auto-encoder that models translations. Each capsule in the figure has 3 recognition units and 4 generation units. The weights ...

[PDF] Dynamic Routing Between Capsules - NIPS Proceedings
by S Sabour - 2017 - Cited by 551 - Related articles

[PDF] matrix capsules with em routing - OpenReview
https://openreview.net/pdf?id=HJWLFGRb
by GE Hinton - 2018 - Cited by 101 - Related articles
MATRIX CAPSULES WITH EM ROUTING. GeoffreyHinton, SaraSabour, NicholasFrosst. Google Brain. Toronto, Canada. {geoffhinton, sasabour, frosst}@google.
Dynamic Routing Between Capsules

Sara Sabour  
Nicholas Frosst

Geoffrey E. Hinton  
Google Brain  
Toronto
{sasabour, frosst, geoffhinton}@google.com

[PDF] Dynamic Routing Between Capsules - NIPS Proceedings
by S Sabour - 2017 - Cited by 557 - Related articles
Dynamic Routing Between Capsules

Sara Sabour
Nicholas Frosst

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

[PDF] Dynamic Routing Between Capsules - NIPS Proceedings
by S Sabour - 2017
Cited by 557 - Related articles
Outlines

• Pros/Cons of CNNs
• CapsNet aims to solving two problems ...
• Routing mechanism
• Experimental Results
• Challenges
• Wrap-up
Convolutional Neural Networks (CNN)

- Main components:
  - Feature detectors, interleaved with subsampling layers
- CNNs work best for recognition
  - Weight sharing
  - Sparsity of connections
Convolutional Neural Networks (CNN)

- Main components:
  - Feature detectors, interleaved with subsampling layers
- CNNs work best for recognition
  - Weight sharing
  - Sparsity of connections
- CNNs afford some translation invariance to small changes
  - Replicating the feature detectors (learned knowledge) across image
  - Max-pooling
CNNs Problems (1)

- **Picasso Problem** → Right parts in wrong position
  - Mere existence of parts means whole
CNNs Problems (1)

- Picasso Problem → Right parts in wrong position
  - Mere existence of parts means whole
    - OK-ish for classification, BAD for segmentation/localisation
CNNs Problems (2)

- No built-in mechanism to extrapolate their understanding (internal representation) to radically new viewpoints
CNNs Problems (2)

- No built-in mechanism to extrapolate their understanding (internal representation) to radically new viewpoints
- Only can deal with this through a lot of training data
Max-pooling is the Culprit ... 

- Along with replicating filters (knowledge) leads to some translation/rotation invariance.
Max-pooling is the Culprit ...

- Along with replicating filters (knowledge) leads to some translation/rotation invariance

- Most active neuron are routed to the higher level ...
  - Without considering the higher level activities (hierarchy)

- Discard information about precise position and relative spatial relationships
Max-pooling is the Culprit …

“The pooling operation used in CNNs is a big mistake and the fact that it works so well is disaster.”

“Internal data representation of a CNN does not take into account important spatial hierarchies between simple and complex objects.”
Computer Graphics

- Entities + Instantiation Parameters $\rightarrow$ Synthetic Images
  - **Entities**: basic shapes
  - **Instantiation parameters**: pose (translation, rotation, etc.)
Inverse Computer Graphics

- Image $\rightarrow$ Entities + Instantiation Parameters

\[ \{\text{Entity 1, pose 1}\} \]
\[ \{\text{Entity 2, pose 2}\} \]
\[ \ldots \]
\[ \{\text{Entity N, pose N}\} \]
Inverse Computer Graphics

- Image → Entities + Instantiation Parameters

Hinton claim: Human brain performs some inverse graphics.
Some Definitions

- Invariant
  - A property that does not change after some transformation

- Equivariant
  - A property that changes predictably under transformation

- Image transformations
  - Shift (translation), scale (size), rotation (orientation), reflection (mirror)
Note that …

- Invariant
- Equivariant
- Image transformations

- Effect of image transformations on …
  - Labels → invariant
  - Instantiations parameters → equivariant
Hinton: Human visual system imposes some coordinate frames in order to represent shapes (after Irvin Rock)

http://ycpcs.github.io/cs470-fall2014/labs/lab07-2.html
Hinton: Human visual system imposes some coordinate frames in order to represent shapes (after Irvin Rock)

http://ycpcs.github.io/cs470-fall2014/labs/lab07-2.html
Hinton: Human visual system imposes some coordinate frames in order to represent shapes (after Irvin Rock)

http://ycpcs.github.io/cs470-fall2014/labs/lab07-2.html
Tetrahedron Jigsaw Puzzle
Tetrahedron Jigsaw Puzzle

1. Find **intrinsic frame of reference**
   - Imagine the whole

2. Build **part-whole maps** using
   - frame of reference
   - contextual info

E. Loweimi
END DETOUR
Invariance and Equivalence

- Label → invariance
- Pose (Instantiation parameters) → equivalence
- Label → invariance
- Pose (Instantiation parameters) → equivalence

* No built-in disentanglement mechanism in CNN
  - A lot of data is required for dealing with pose change.
Capsule Networks aim at solving two problems ...

- Disentangling invariance (label) and equivariance (pose) learning mechanisms

- Smarter way for information flow from lower layers to the higher layers in the hierarchy
Capsule and CapsNet

- A set of neurons that collectively produce an activity vector
- Each capsule detects/represents an entity
  - Length: probability of presence/existence
  - Orientation: instantiation parameters, state, properties

CapsNets // CNN with two differences
- Scalar feature detectors replaced with vector-output capsules
- Pooling replaced with routing/agreement
Capsule and CapsNet

• A set of neurons that collectively produce an activity vector
• Each capsule detects/represents an entity
  – Length: probability of presence/existence
  – Orientation: instantiation parameters, state, properties

• CapsNets vs CNN with two differences
  – Scalar feature detectors r replaced with vector-output capsules
  – Max-pooling is replaced with routing-by-agreement
CapsNet Approach to Invariance and Equivalence Properties

- Entity’s presence probability: invariance
- Entity’s pose (Instantiation parameters) equivalence
- Built-in separation mechanism

E. Loweimi
Dynamic Routing Mechanism
Routing-by-Agreement – Steps

0) Outputs of capsules in lower layer \((u_i)\) r available

For capsule \(j\) in higher layer, make a prediction \((\hat{u}_j|i)\)

Compare the prediction with actual output \((v_j)\)

Based on \(\hat{u}_j|i\) & \(v_j\) similarity, adjust the connection strength (routing)
Routing-by-Agreement – Steps

0) Outputs of capsules in lower layer ($u_i$) are available

1) For capsule $j$ in higher layer, make a prediction ($\hat{u}_{ji}$)
Routing-by-Agreement – Steps

0) Outputs of capsules in lower layer \((u_i)\) r available

1) For capsule \(j\) in higher layer, make a prediction \((\hat{u}_{ji})\)

2) Compare the prediction with actual output \((v_j)\)
Routing-by-Agreement – Steps

0) Outputs of capsules in lower layer ($u_i$) are available

1) For capsule $j$ in higher layer, make a prediction ($\hat{u}_{ji}$)

2) Compare the prediction with actual output ($v_j$)

3) Based on $\hat{u}_{ji}$ & $v_j$ similarity, adjust the connection strength (routing)
Routing-by-Agreement – Steps

0) Outputs of capsules in lower layer \( (u_i) \) r available

1) For capsule \( j \) in higher layer, make a prediction \( (\hat{u}_{j|i}) \)

2) Compare the prediction with actual output \( (v_j) \)

3) Based on \( \hat{u}_{j|i} \) & \( v_j \) similarity, adjust the connection strength (routing)

4) Go to (2), if not converged
Routing-by-Agreement – Equations

\( \hat{u}_{ji} \): Prediction using \( W_{ij} \)

\( c_{ij} \): coupling coef.

\( s_j \): pre-activation

\( v_j \): activation

Squashing Non-linearity

\( b_{ij} \): logit (similarity)

\[
\hat{u}_{ji} = W_{ij} u_i
\]

\[
s_j = \sum_i c_{ij} \hat{u}_{ji}
\]

\[
v_j = \frac{\|s_j\|^2 s_j}{1 + \|s_j\|^2 \|s_j\|} \quad \text{iteration}
\]

\[
b_{ij} = v_j \hat{u}_{ji} \quad \Rightarrow \quad c_{ij} = \frac{\exp(b_{ij})}{\sum_{j'} \exp(b_{ij'})}
\]
Routing-by-Agreement – WorkFlow

Dynamic Routing

\[ \sum_{j} s_j \]

E. Loweimi
Routing-by-Agreement – Algorithm

**Procedure 1 Routing algorithm.**

1: **procedure** ROUTING($\hat{u}_{j|i}$, $r$, $l$)
2: for all capsule $i$ in layer $l$ and capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow 0$.  
3: **for** $r$ iterations **do**
4: for all capsule $i$ in layer $l$: $c_i \leftarrow \text{softmax}(b_i)$  
5: for all capsule $j$ in layer $(l+1)$: $s_j \leftarrow \sum_i c_{ij} \hat{u}_{j|i}$  
6: for all capsule $j$ in layer $(l+1)$: $v_j \leftarrow \text{squash}(s_j)$
7: for all capsule $j$ in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i}.v_j$

**return** $v_j$

- $c_{ij}$ is learned by **dynamic routing** in **forward** path
- $W_{ij}$ is learned by **backprop** in **backward** path

\[
c_{ij} = \frac{\exp(b_{ij})}{\sum_{j'} \exp(b_{ij'})}
\]

\[
v_j = \frac{\|s_j\|^2 s_j}{1 + \|s_j\|^2 \|s_j\|}
\]
## Conventional NN vs CapsNet

<table>
<thead>
<tr>
<th></th>
<th>Neurons</th>
<th>Capsules</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input/Output</strong></td>
<td>Vector/Scalar</td>
<td>Vector/Vector</td>
</tr>
<tr>
<td><strong>Training</strong></td>
<td>Backpropagation</td>
<td>Dynamic Routing &amp; Backpropagation</td>
</tr>
<tr>
<td><strong>Pre-activation</strong></td>
<td>$z_j = \sum_i w_{ij} x_i + b_j$</td>
<td>$s_j = \sum_i c_{ij} W_{ij} u_i$</td>
</tr>
<tr>
<td><strong>Non-linearity</strong></td>
<td>scalar2scaler</td>
<td>vector2vector</td>
</tr>
<tr>
<td></td>
<td>ReLU, Tanh, etc.</td>
<td>$v_j = \frac{</td>
</tr>
</tbody>
</table>
Routing-by-Agreement

Intuitions
Routing-by-Agreement – Intuition (1)

PREDICTION

\[ \hat{u}_{ji} \rightarrow \mathbf{v}_j \]
\[ \hat{u}_{j1} \rightarrow \mathbf{v}_j \]
\[ \hat{u}_{j10} \rightarrow \mathbf{v}_j \]
\[ \hat{u}_{k1} \rightarrow \mathbf{v}_k \]
\[ \hat{u}_{k10} \rightarrow \mathbf{v}_k \]

Vote plane

E. Loweimi
Routing-by-Agreement – Intuition (1)

\[ \begin{align*}
\hat{u}_{j1} & \rightarrow v_j \\
\hat{u}_{j10} & \rightarrow v_j \\
\hat{u}_{k1} & \rightarrow v_k \\
\hat{u}_{k10} & \rightarrow v_k \\
\vdots & \\
\hat{u}_1 & \rightarrow v_1 \\
\vdots & \\
\hat{u}_{10} & \rightarrow v_{10}
\end{align*} \]
Routing-by-Agreement – Intuition (1)

Capsules in dash circle should be routed to larger coupling coefficients, $c_{ij}$, *(part-whole map)*.
Routing-by-Agreement – Intuition (2)

Coincidence filtering: Outliers are filtered out (small $c_{ij}$).
Routing-by-Agreement – Note

Votes distribution in vote plane, i.e. $\hat{u}_{ji}$ and $\hat{u}_{kl}$, are different because $W_{ij}$ and $W_{ik}$ are different.

\[ \hat{u}_{ji} = W_{ij} u_i \]

E. Loweimi
Each higher level capsule has a dynamic routing block. 
- Primary capsule layer (lowest level) has not.

Dynamic Routing
Routing-by-Agreement – Intuition (3)

Computes Logits (agreement or Similarity)

Routing-by-agreement

Top-down feedback ≡ Iteration

E. Loweimi 22/40
Routing-by-agreement to done greedily across layers ...

– When iterations between blue-green finished, move to green-red.
Capsule Network in NIPS 2017

Dynamic Routing Between Capsules

Sara Sabour  Nicholas Frosst

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com
Capsule Network in NIPS 2017

\[ W_{ij} = [8 \times 16] \]

rescale(-1, capsule_size)

W/out //40-pooling

Primary capsule size

DigitCaps size

E. Loweimi
Capsule Network in NIPS 2017

\[ W_{ij} = [8 \times 16] \]

\[ \text{reshape}(-1, \text{capsule\_size}) \]

\[ \|v_0\|, \|v_2\|, \|v_9\| \]

Primary capsule size

DigitCaps size

E. Loweimi
Reconstruction Network (Decoder)

- Target capsule is kept, rest is 0 masked
- Reconstruction from a vector $\rightarrow \sim$ auto-encoder

E. Loweimi
Unsupervised Reconstruction Loss

Classification (supervised)

Reconstruction (unsupervised)

Encoder

Decoder

\[ \|v_0\|, \|v_2\|, \|v_9\| \]

\[ W_{ij} = [8 \times 16] \]

E. Loweimi
Loss Function

• Loss function = \text{supervised} + \alpha \text{ unsupervised}
Loss Function

- Loss function = supervised + \( \alpha \) unsupervised
- Supervised (encoder) \( \rightarrow \) classification
  - margin loss
- Unsupervised (decoder) \( \rightarrow \) Reconstruction
  - MSE
  - Down-scaled by \( \alpha = 5e-4 \)
  - Adjusting scales + making the supervised part dominant
Supervised Margin Loss

\[ L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda (1 - T_k) \max(0, \|v_k\| - m^-)^2 \]

\[ L = \sum_k L_k \]

\[ T_k = 1 \text{ if (digit of class } k \text{ is present) else 0} \]
Supervised Margin Loss

\[ L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda (1 - T_k) \max(0, \|v_k\| - m^-)^2 \]

\[ L = \sum_k L_k \]

- \( T_k = 1 \) if (digit of class k is present) else 0

\[ Z = T_k X + (1 - T_k) Y \]
- \( T_k \in (0,1) \) → weighted mean
- \( T_k \in \{0,1\} \) → \( Z = X \) if \( T_k = 1 \) else \( Y \)
Supervised Margin Loss

\[ L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda (1 - T_k) \max(0, \|v_k\| - m^-)^2 \]

\[ L = \sum_k L_k \]

* Hinge (max-margin) loss:
  - \( \max(0, m^+-x) \)
    \( \implies \) min loss: \( x > m^+ \)
  - \( \max(0, x-m^-) \)
    \( \implies \) min loss: \( x < m^- \)
  - \( m^+=0.9, m^-=0.1 \)
Supervised Margin Loss

\[ L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda (1 - T_k) \max(0, \|v_k\| - m^-)^2 \]

\[ L = \sum_k L_k \]

For minimum loss
- If \( T_k = 1 \): \( \|v_k\| > m^+ \)
- If \( T_k = 0 \): \( \|v_k\| < m^- \)
Supervised Margin Loss

\[ L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda (1 - T_k) \max(0, \|v_k\| - m^-)^2 \]

\[ L = \sum_k L_k \]

* \( \lambda = 0.5 \)
  – down-weighs \( T_k = 0 \) case
  – Purpose: Numerical stability

E. Loweimi
Experimental Results
## CapsNet Classification Error

**Table 1**

<table>
<thead>
<tr>
<th>Method</th>
<th>Routing</th>
<th>Reconstruction</th>
<th>MNIST (%)</th>
<th>MultiMNIST (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>0.39</td>
<td>8.1</td>
</tr>
<tr>
<td>CapsNet</td>
<td>1</td>
<td>no</td>
<td>0.34 ± 0.032</td>
<td>-</td>
</tr>
<tr>
<td>CapsNet</td>
<td>1</td>
<td>yes</td>
<td>0.29 ± 0.011</td>
<td>7.5</td>
</tr>
<tr>
<td>CapsNet</td>
<td>3</td>
<td>no</td>
<td>0.35 ± 0.036</td>
<td>-</td>
</tr>
<tr>
<td>CapsNet</td>
<td>3</td>
<td>yes</td>
<td><strong>0.25 ± 0.005</strong></td>
<td><strong>5.2</strong></td>
</tr>
</tbody>
</table>

**STOA: 0.21%**

Trials for STD: 3
CapsNet Classification Error

Table 1

<table>
<thead>
<tr>
<th>Method</th>
<th>Routing</th>
<th>Reconstruction</th>
<th>MNIST (%)</th>
<th>MultiMNIST (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>0.39</td>
<td>8.1</td>
</tr>
<tr>
<td>CapsNet</td>
<td>1</td>
<td>no</td>
<td>0.34±0.032</td>
<td>-</td>
</tr>
<tr>
<td>CapsNet</td>
<td>1</td>
<td>yes</td>
<td>0.29±0.011</td>
<td>7.5</td>
</tr>
<tr>
<td>CapsNet</td>
<td>3</td>
<td>no</td>
<td>0.35±0.036</td>
<td>-</td>
</tr>
<tr>
<td>CapsNet</td>
<td>3</td>
<td>yes</td>
<td>0.25±0.005</td>
<td>5.2</td>
</tr>
</tbody>
</table>

* Routing iterations: 3 to 5 is enough ← computational overhead
* Adding reconstruction term to loss is useful.
CapsNet Logit Change (MNIST)

After 500 epochs, average change in logit ($b_{ij}$) is stabilised.

3 iterations of routing is enough.
CapsNet Training Loss (CIFAR 10)

3 iteration of routing optimises the loss faster and converges to a lower loss at the end.

More routing iterations increases the network capacity → overfitting
CapsNet Error on CIFAR 10

- CapsNet: 10.6%
  - About what standard CNNs achieved when first tried
    - Zeiler and Fergus 2013 → 19.4, 15.1%
  - State-of-the-art: 3.47% (Graham 2015)
CapsNet Error on Small NORB

- CapsNet error: 2.7%
  - Best task for CapsNet (Appendix B)
  - On-par with state-of-the-art (2.56%)
    - Ciresen et al., 2011
  - New CapsNet with EM routing, ICLR 2018 → 1.4%
(l, r, p) = (target label, prediction, reconstruction target)

Correct Classification

Reconstruction from the target capsule (~ auto-encoder)
CapsNet Reconstruction

\[(l, r, p) = (\text{target label, prediction, reconstruction target})\]

<table>
<thead>
<tr>
<th>((l, p, r))</th>
<th>(2, 2, 2)</th>
<th>(5, 5, 5)</th>
<th>(8, 8, 8)</th>
<th>(9, 9, 9)</th>
<th>(5, 3, 5)</th>
<th>(5, 3, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input</strong></td>
<td><img src="image1" alt="Input 2" /></td>
<td><img src="image2" alt="Input 5" /></td>
<td><img src="image3" alt="Input 8" /></td>
<td><img src="image4" alt="Input 9" /></td>
<td><img src="image5" alt="Input 3" /></td>
<td><img src="image6" alt="Input 3" /></td>
</tr>
<tr>
<td><strong>Output</strong></td>
<td><img src="image7" alt="Output 2" /></td>
<td><img src="image8" alt="Output 5" /></td>
<td><img src="image9" alt="Output 8" /></td>
<td><img src="image10" alt="Output 9" /></td>
<td><img src="image11" alt="Output 3" /></td>
<td><img src="image12" alt="Output 3" /></td>
</tr>
</tbody>
</table>

Correct Classification

Misclassification

The model preserves many of the details while smoothing the noise.

E. Loweimi
### Effect of Dimension Perturbation on Recon.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale and thickness</td>
<td>-0.25</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>-0.20</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0.20</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>6</td>
</tr>
<tr>
<td>Localized part</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Stroke thickness</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Localized skew</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Width and translation</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Localized part</td>
<td></td>
<td>2</td>
</tr>
</tbody>
</table>

Tweak value: 37/40

E. Loweimi
Effect of Dimension Perturbation on Recon.

<table>
<thead>
<tr>
<th>Tweak value</th>
<th>-0.25</th>
<th>-0.20</th>
<th>0.00</th>
<th>0.20</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale and thickness</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
</tr>
<tr>
<td>Localized part</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
<td>6 6 6 6</td>
</tr>
<tr>
<td>Stroke thickness</td>
<td>5 5 5 5</td>
<td>5 5 5 5</td>
<td>5 5 5 5</td>
<td>5 5 5 5</td>
<td>5 5 5 5</td>
</tr>
<tr>
<td>Localized skew</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
<td>4 4 4 4</td>
</tr>
<tr>
<td>Width and translation</td>
<td>3 3 3 3</td>
<td>3 3 3 3</td>
<td>3 3 3 3</td>
<td>3 3 3 3</td>
<td>3 3 3 3</td>
</tr>
<tr>
<td>Localized part</td>
<td>2 2 2 2</td>
<td>2 2 2 2</td>
<td>2 2 2 2</td>
<td>2 2 2 2</td>
<td>2 2 2 2</td>
</tr>
</tbody>
</table>

Each dimension of capsule learns to span the space of variation of an instantiation parameter, e.g. scale, translation, thickness.
Effect of Dimension Perturbation on Recon.

<table>
<thead>
<tr>
<th>Scale and thickness</th>
<th>-0.25</th>
<th>-0.20</th>
<th>0.00</th>
<th>0.20</th>
<th>0.25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Localized part</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Stroke thickness</td>
<td>5.5</td>
<td>5.5</td>
<td>5.5</td>
<td>5.5</td>
<td>5.5</td>
</tr>
<tr>
<td>Localized skew</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Width and translation</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Localized part</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Tweak value

Higher interpretability & controllability
MultiMNIST Reconstruction

- **MultiMNIST**
  - Each X has two labels \((l_1, l_2)\)
- **L**: \((l_1, l_2)\)
  - Target classification labels
- **R**: \((r_1, r_2)\)
  - Target reconstruction label
- **P**: predicted label
- **\(*\)**: reconstruction from a digit that is neither the label nor the prediction.

Red and Green are reconstructed digits (yellow: overlap)

---

E. Loweimi
MultiMNIST Reconstruction

CapsNet successfully deals with overlapping objects.

Red and Green are reconstructed digits (yellow: overlap)
Challenges Ahead CapsNet

- Not state-of-the-art in tasks like CIFAR 10 (good start!)
- Not tested yet on larger images due to technical issues
  - Slow training $\rightarrow$ Routing iterations
  - Memory problem
- A CapsNet cannot see two very close identical objects
  - “crowding” $\leftrightarrow$ similar to human vision system
Wrap-up (1)

- Each capsule is a group of neurons
  - Expand artificial scalar neuron to vector

- Capsule represents an entity through a vector (inverse graphics)
  - Magnitude → probability of the entity presence → invariant
  - Phase → state of the entity → equivariant

- Dynamic routing: how capsules of two layers should communicate

- Parameters & Learning
  - Coupling coefficients ($c_{ij}$) → routing-by-agreement
  - Affine transformations ($W_{ij}$) → backpropagation
Wrap-up (2)

• Advantages:
  – built-in disentanglement between entity’s pose (instantiation parameters) and presence probability
  – Dynamic hierarchical modelling, smarter than static max-pooling
  – requires less data, higher robustness (viewpoint), interpretability

• Challenges:
  – Technical difficulties in scaling up (e.g. memory problem)
  – Performance is still not in the state-of-the-art level
    • e.g. CIFAR 10 (Error: 10.6% vs 3.47%)
That’s it!

• Thanks for Your Attention

• Q/A

• Appendices
  - Appendix A: MNIST Database & its variants
  - Appendix B: (Small) NORB Database
MNIST Database

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

LeNet-5 Architecture

E. Loweimi
MNIST Database

- Yan LeCun et al., 1998
- Handwritten digits
  - 28 x 28
  - Training: 60 k
  - Test: 10 k
- Variants
  - affMNIST
  - MultiMNIST
  - EMNIST: letters+digits
    - train: 240k, test: 40k
Learning Methods for Generic Object Recognition
with Invariance to Pose and Lighting

Yann LeCun, Fu Jie Huang,
The Courant Institute, New York University
715 Broadway, New York, NY 10003, USA
http://yann.lecun.com

Léon Bottou
NEC Labs America,
4 Independence Way, Princeton, NJ 08540
http://leon.bottou.org

Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’04)
1063-6919/04 $20.00 © 2004 IEEE
(Small) NORB Database

- Y. LeCun et al., 2004
- 3D object recognition task
  - 96 x 96 images of 50 toys, 5-generic categories
    - Animal, human, airplane, car, truck
- Objects where imaged by 2 cameras under ...
  - 6 Lighting conditions, 9 elevations, 18 azimuths
- Download
  - NORB → 29160 images
  - Small NORB → 24300 images
    - Normalised object sizes and uniform background

E. Loweimi