

# Dynamic Routing Between Capsules

Mine Melodi Caliskan  
CmpE597 - Deep Learning

May 14, 2018

# Outline

- ▶ Introduction
- ▶ CNN vs. CapsNet
- ▶ Neurons vs. Capsules
- ▶ CapsNet Architecture
- ▶ How It Works
- ▶ Inputs and Outputs of Capsules
- ▶ Dynamic Routing Between Capsules
- ▶ Margin Loss Function
- ▶ Performance on MNIST and other datasets
- ▶ Discussion

# Introduction

- ▶ A new type of neural network based on *capsules* that are designed to support *transformation-independent* object recognition.
- ▶ The activities of the neurons within an active capsule represent the various properties of a particular entity that is present in the image.
- ▶ Different types of instantiation parameter, e.g position, size, orientation, velocity, etc.
- ▶ *Dynamic routing between capsules* algorithm allows to train such a network.

# Motivation

- ▶ CNN is good at detecting features, but less effective at exploring the spatial relationships among features (perspective, size, orientation), i.e CNN is transitional invariance, not equivariance.
  - ▶ Max pooling detects whether a feature is present in any region of the image, loses spatial information about the feature.
- ▶ Capsule networks' goal: Transformation-independent object recognition i.e *equivariance*, by capturing part-whole relationships.



**Figure 1:** Example: In a face, each of the two eyes is part of a forehead. [2]

## Neurons in CNN

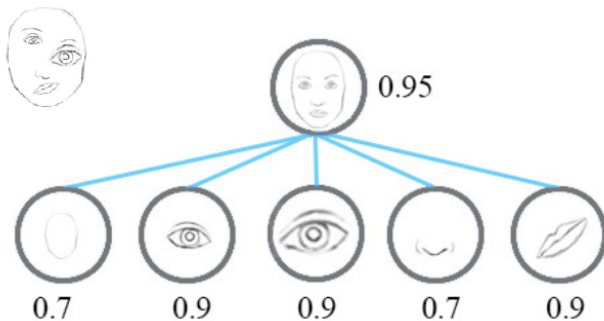


Figure 2: Face detection using a simple CNN. [2]

A simple CNN model can extract the features for nose, eyes and mouth correctly but will wrongly activate the neuron for the face detection. Without realize the mis-match in spatial orientation and size, the activation for the face detection will be too high.

## Neurons in Capsule Network

Each neuron contains the likelihood as well as properties of the features, e.g [likelihood, orientation, size]. This spatial information can detect the in-consistence in the orientation and size. Therefore output a much lower activation for the face detection.

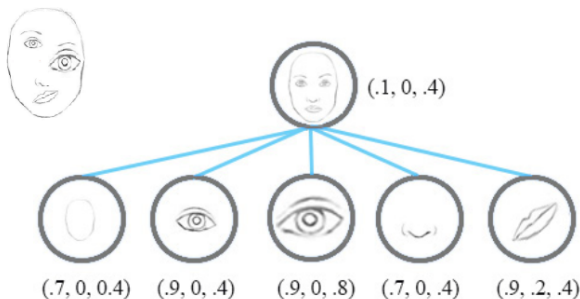
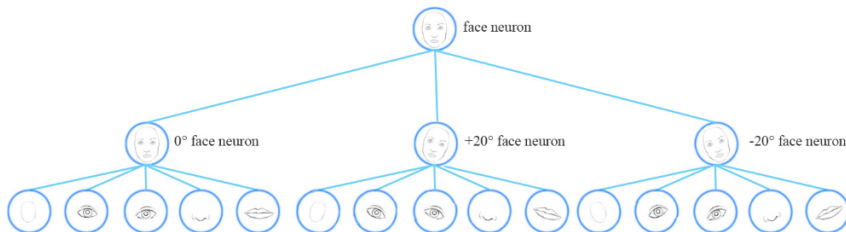


Figure 3: Face detection using capsule networks. [2]

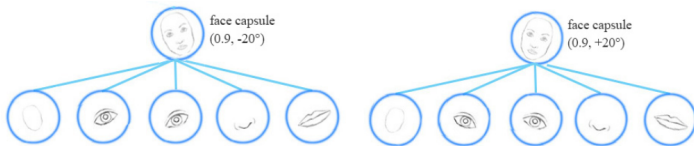
The term capsules indicates that neurons in this architecture output a vector instead of a single scalar value.

# Equivariance



**Figure 4:** A CNN model uses multiple neurons and layers in capturing different feature's variants. [2]

# Equivariance



**Figure 5:** A capsule network share the same capsule to detect multiple variants in a simpler network. [2]

A capsule detects the face is rotated right 20 (or rotated left 20) rather than realizes the face matched a variant that is rotated right 20. By forcing the model to learn the feature variant in a capsule, we may extrapolate possible variants **more effectively** with **less training data**.



# Neuron vs. Capsule

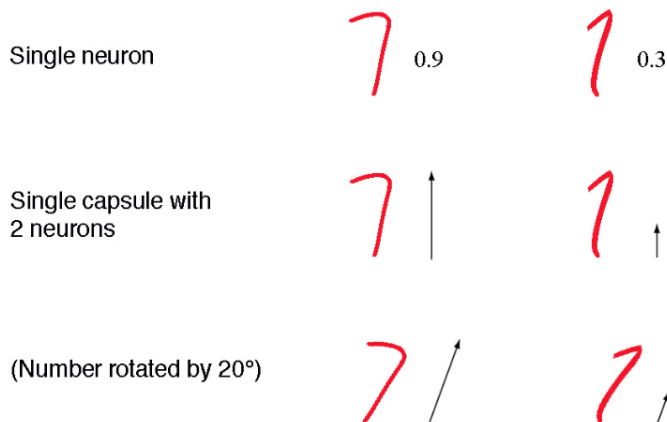
## Neuron

- ▶ The activation of a single neuron  $\sim$  the likelihood of a specific feature in the input image.

## Capsule

- ▶ A group of neurons which not only capture the likelihood of specific features but also the parameters related to the features.
- ▶ The magnitude of activity vector represents the probability of detecting specific features, and its orientation represents its parameters.

# Neuron vs. Capsule



**Figure 6:** This capsule outputs a 2 –  $D$  vector in detecting the number “7”. For the first image in the second row, it outputs a vector  $v = (0, 0.9)$ . [2]

# CapsNet Architecture

The CapsNet has 2 parts: encoder and decoder. The first 3 layers are encoder, and the second 3 are decoder:

Layer 1. Convolutional layer

Layer 2. PrimaryCaps layer

Layer 3. DigitCaps layer

Layer 4. Fully connected #1

Layer 5. Fully connected #2

Layer 6. Fully connected #3

# CapsNet architecture

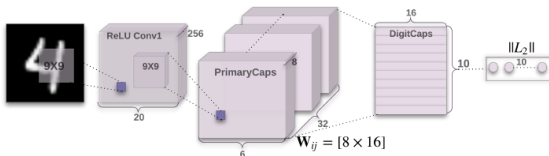


Figure 7: CapsNet with 3 layers.

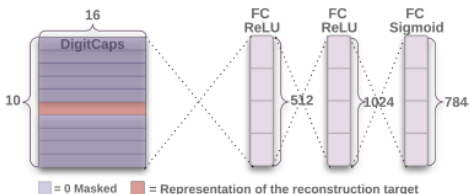
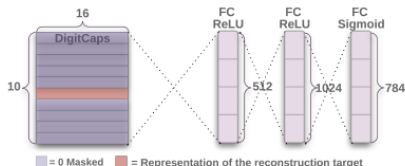


Figure 8: Decoder structure to reconstruct a digit from the DigitCaps layer representation.



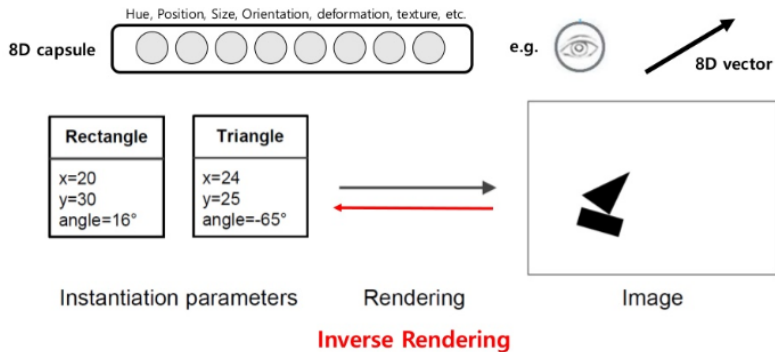
# CapsNet architecture



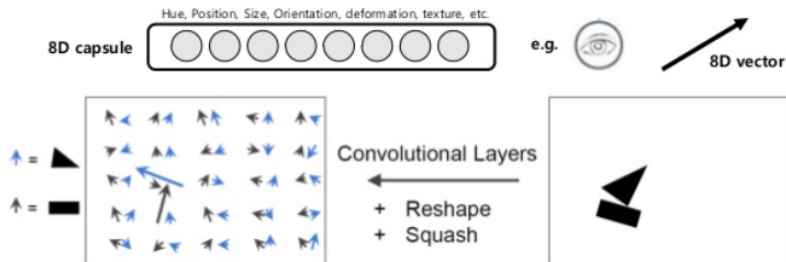
**Figure 9:** Decoder structure to reconstruct a digit from the DigitCaps layer representation.

Decoder is used as a regularizer, it takes the output of the correct DigitCap as input and learns to recreate an 28 by 28 pixels image, with the loss function being Euclidean distance between the reconstructed image and the input image.

# How it works



# How it works



- ▶ Length of the activation vector: Estimated probability of presence.
- ▶ Orientation: Object's estimated pose parameters.



## How it works

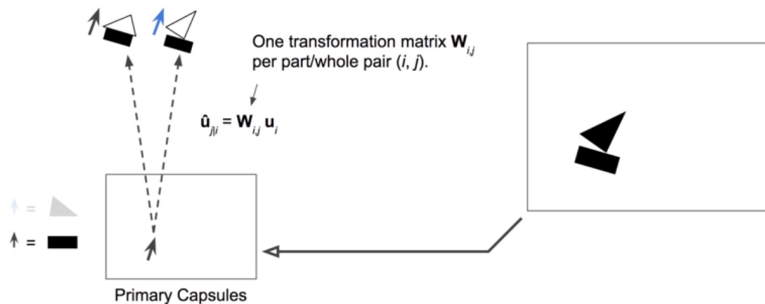


Figure 10: Primary capsules predict next layers output. [1]

## How it works

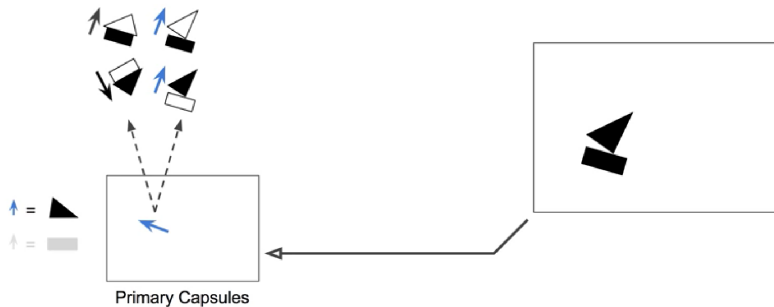
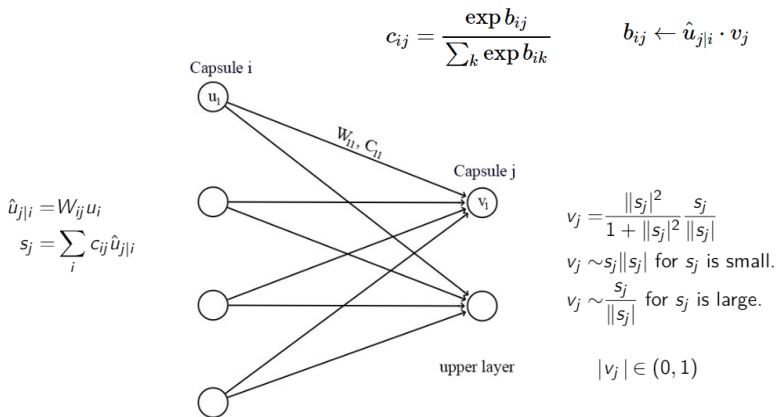


Figure 11: Primary capsules predict next layers output. [1]

# Inputs and Outputs of Capsules



$W_{ij}$  : Transformation matrix ( $p \times k$ )  $\Rightarrow$  Encodes part-whole relationships  
 $u_i$ : input vector ( $k \times 1$ )  
 $v_j$ : output vector ( $p \times 1$ )

# Routing by Agreement

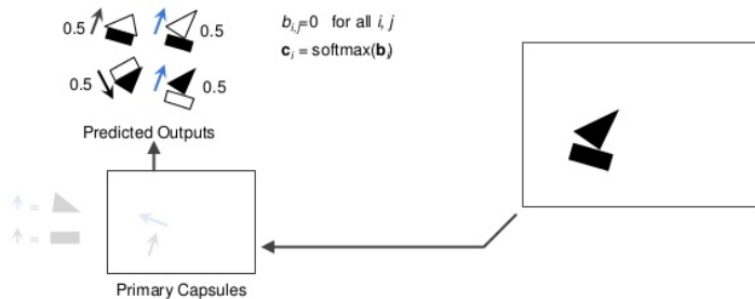


Figure 12: Routing weights. [1]

# Routing by Agreement

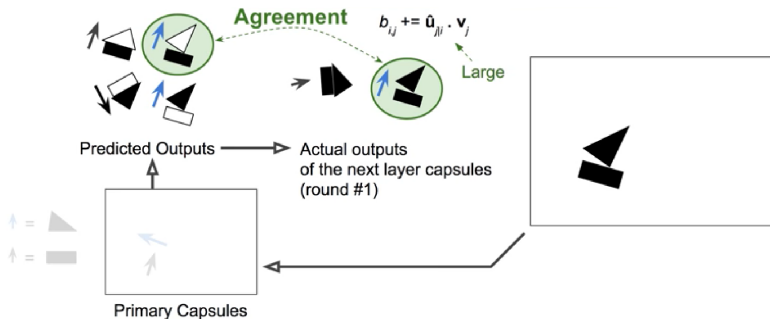


Figure 13: Update routing weights. [1]

# Routing by Agreement

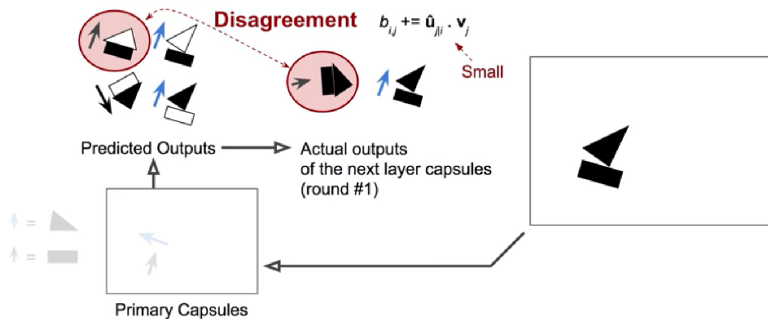
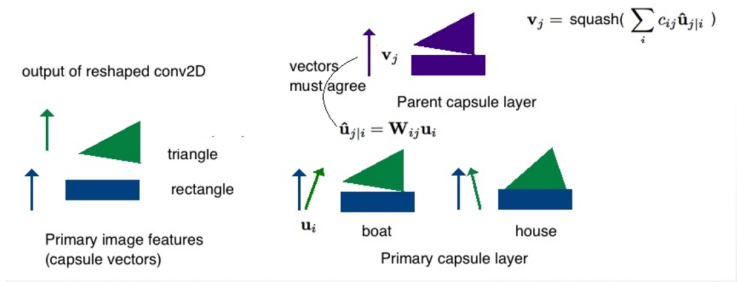


Figure 14: Update routing weights. [1]

# Dynamical Routing between Capsules

The essence of the *dynamic routing algorithm*: Lower level capsule will send its input to the higher level capsule that “agrees” with its input.

- Hierarchy of parts: coupled layers.



# Dynamical Routing between Capsules

In capsule, we use iterative dynamic routing to compute the capsule output by calculating an intermediate value  $c_{ij}$  (coupling coefficient).

Recall that the prediction vector  $\hat{u}_{j|i}$  is computed as:

$$\hat{u}_{j|i} = W_{ij} u_i \quad (1)$$

and the activity vector  $v_j$  (the capsule  $j$  output) is:

$$s_j = \sum_i c_{ij} \hat{u}_{j|i} \quad (2)$$

$$v_j = \frac{\|s_j\|^2}{1 + \|s_j\|^2} \frac{s_j}{\|s_j\|} \quad (3)$$

Intuitively, prediction vector  $\hat{u}_{j|i}$  is the prediction (vote) from the capsule  $i$  on the output of the capsule  $j$  above.



# Dynamical Routing between Capsules

- ▶ If the activity vector has close similarity with the prediction vector  $\Rightarrow$  Both capsules are highly related.
- ▶ Similarity is measured using the scalar product of the prediction and the activity vector.

$$b_{ij} \leftarrow \hat{u}_{j|i} v_j \quad (4)$$

- ▶ Therefore, the similarity score  $b_{ij}$  takes into account on both likeliness and the feature properties, instead of just likeliness in neurons.

Also,  $b_{ij}$  remains low if the activation  $u_i$  of capsule  $i$  is low since  $\hat{u}_{j|i}$  length is proportional to  $u_i$ . i.e.  $b_{ij}$  should remain low between the mouth capsule and the face capsule if the mouth capsule is not activated.

# Dynamical Routing between Capsules

The *coupling coefficients*  $c_{ij}$  is computed as the softmax of  $b_{ij}$ :

$$c_{ij} = \frac{\exp b_{ij}}{\sum \exp_k b_{ik}} \quad (5)$$

To make  $b_{ij}$  more accurate , it is updated iteratively in multiple iterations (typically in 3 iterations).

$$b_{ij} \leftarrow b_{ij} + \hat{u}_{j|i} v_j \quad (6)$$

# Routing Algorithm

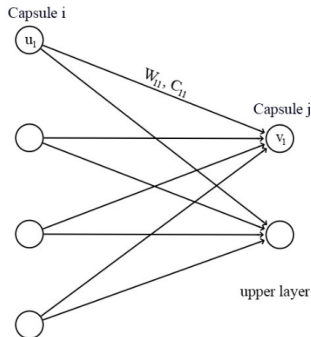
---

**Procedure 1** Routing algorithm.

---

```
1: procedure ROUTING( $\hat{\mathbf{u}}_{j|i}, r, l$ )
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  $l$ :  $\mathbf{c}_i \leftarrow \text{softmax}(\mathbf{b}_i)$ 
5:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{s}_j \leftarrow \sum_i c_{ij} \hat{\mathbf{u}}_{j|i}$ 
6:     for all capsule  $j$  in layer  $(l + 1)$ :  $\mathbf{v}_j \leftarrow \text{squash}(\mathbf{s}_j)$ 
7:     for all capsule  $i$  in layer  $l$  and capsule  $j$  in layer  $(l + 1)$ :  $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i} \cdot \mathbf{v}_j$ 
   return  $\mathbf{v}_j$ 
```

---



# Equivariance of Capsules

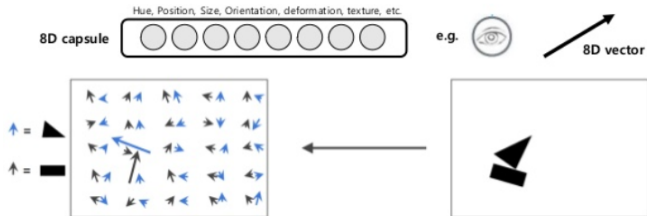
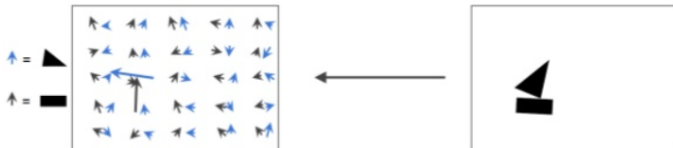


Figure 15: Slight rotations in the image changes the activation vectors slightly. [1]



# Two types of equivariance

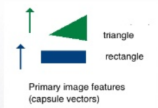
## Place-coding:

convNet w/out pooling

low level features for

small receptive fields

when a part moves, it may  
gets a new capsule



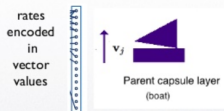
position maps to active  
capsules ( $\mathbf{u}$ ) in primary layer

## Rate-coding:

traditional\_neurological way of coding (1926)

stimulus info encoded in rate of firing  
(as opposed to magnitude, population, timing, ...)

when a part rotates or moves,  
the capsule values change



maps to real-values of capsule output vectors ( $\mathbf{v}$ )

# Margin loss for digit existence

loss term for one DigitCap

$$L_c = \underbrace{T_c \max(0, m^+ - \|\mathbf{v}_c\|)^2}_{\substack{\text{1 when correct} \\ \text{DigitCap,} \\ \text{0 when incorrect}}} + \underbrace{\lambda (1 - T_c) \max(0, \|\mathbf{v}_c\| - m^-)^2}_{\substack{\text{1 when incorrect} \\ \text{DigitCap,} \\ \text{0 when correct}}}$$

calculated for correct DigitCap

calculated for incorrect DigitCaps

L2 norm

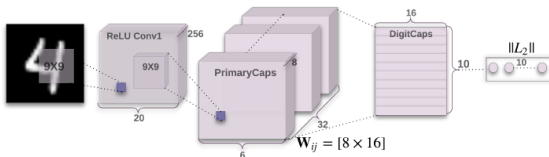
L2 norm

0.5 constant used for numerical stability

zero loss when correct prediction with probability greater than 0.9, non-zero otherwise

zero loss when incorrect prediction with probability less than 0.1, non-zero otherwise

Figure 16: CapsNet Loss Function.



# Reconstruction Loss

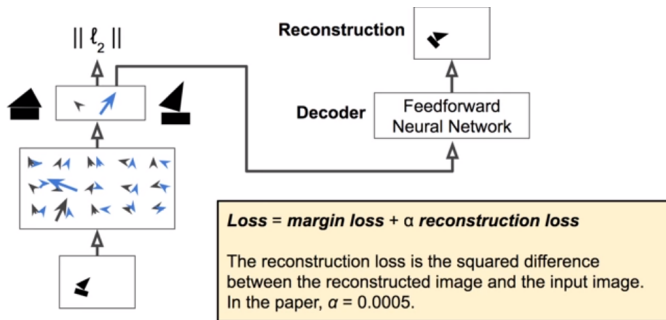


Figure 17: Regularization by reconstruction. [1]

- ▶ Reconstruction loss forces the network to preserve all the information required to reconstruct the image.
- ▶ This constraint acts as a regularizer, reduces the risk of overfitting and helps to generalize to new examples.
- ▶ Reconstruction loss is scaled down to ensure the margin loss dominates the training.

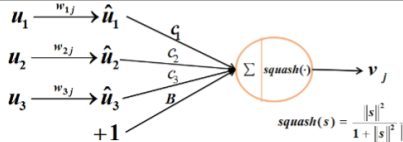
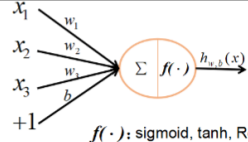
		capsule	VS.	traditional neuron
Input from low-level neurons/capsules		vector( $u_i$ )		scalar( $x_i$ )
Operations	Linear/Affine Transformation	$\hat{u}_{ji} = W_{ij} u_i + B_j$ (Eq. 2)		$a_{ji} = w_{ij} \cdot x_i + b_j$
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{ji}$ (Eq. 2)		$z_j = \sum_{i=1}^3 1 \cdot a_{ji}$
	Summation			
	Non-linearity activation	$v_j = \text{squash}(s_j)$ (Eq. 1)		$h_{w,b}(x) = f(z_j)$
output		vector( $v_j$ )		scalar( $h$ )
				

Figure 18: Comparison between capsules and neurons.[3]



- ▶ Additional reconstruction loss to encourage the digit capsules to encode the instantiation parameters of the input digit. But scale down this reconstruction loss so that it does not dominate the margin loss during training.
- ▶ Mask out all but the activity vector of the correct digit capsule. Then use this activity vector to reconstruct the input image.
- ▶ The output of the digit capsule is fed into a decoder consisting of 3 fully connected layers that model the pixel intensities.
- ▶ Minimize the sum of squared differences between the outputs of the logistic units and the pixel intensities.

## Reconstruction as a regularization method













$(l, p, r)$	$(2, 2, 2)$	$(5, 5, 5)$	$(8, 8, 8)$	$(9, 9, 9)$	$(5, 3, 5)$	$(5, 3, 3)$
Input						
Output						

Figure 19: Sample MNIST test reconstructions of a CapsNet with 3 routing iterations.

# Capsules on MNIST

- Images that have been shifted by up to pixels in each direction with zero padding 2

Method	Routing	Reconstruction	MNIST (%)	MultiMNIST (%)
Baseline	-	-	0.39	8.1
CapsNet	1	no	$0.34_{\pm 0.032}$	-
CapsNet	1	yes	$0.29_{\pm 0.011}$	7.5
CapsNet	3	no	$0.35_{\pm 0.036}$	-
CapsNet	3	yes	<b><math>0.25_{\pm 0.005}</math></b>	<b>5.2</b>

Figure 20: CapsNet classification test accuracy. The MNIST average and standard deviation results are reported from 3 trials.

## The individual dimensions of a capsule




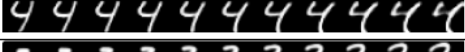


Scale and thickness	
Localized part	
Stroke thickness	
Localized skew	
Width and translation	
Localized part	

Figure 21: Reconstruction of images after dimension perturbations of the top layer outputs.

- ▶ After computing, they the activity vector for the correct digit capsule feed a perturbed version of this activity vector to the decoder network and see how the perturbation affects the reconstruction.

# Robustness to Affine Transformations

- ▶ affNIST data set: Each example is an MNIST digit with a random small affine transformation.
- ▶ Capsnets were never trained with affine transformations other than translation and any natural transformation seen in the standard MNIST.
- ▶ An under-trained CapsNet with early stopping which achieved 99.23% accuracy on the expanded MNIST test set achieved 79% accuracy on the affNIST test set.
- ▶ A traditional convolutional model with a similar number of parameters which achieved similar accuracy (99.22%) on the expanded MNIST test set only achieved 66% on the affNIST test set.

# Segmenting highly overlapping digits

## **MultiMNIST dataset**

- ▶ Generated by overlaying a digit on top of another digit from the same set (training or test) but different class.
- ▶ Each digit is shifted up to 4 pixels in each direction resulting in a 3636 image.
- ▶ Considering a digit in a 2828 image is bounded in a 2020 box, two digits bounding boxes on average have 80% overlap.

## MultiMNIST results



Figure 22: Sample reconstructions of a CapsNet with 3 routing iterations on MultiMNIST test dataset.

## Other datasets

- ▶ CIFAR10 and achieved 10.6% error
- ▶ "None-of-the-above" category for the routing softmaxes
- ▶ The exact same architecture that is used for MNIST on smallNORB and achieved 2.7% test error rate, which is on-par with the state-of-the-art

### *original NORB DATASET*





# Discussion

## Pros

- ▶ Requires less training data.
- ▶ Position and pose information are preserved.
- ▶ Routing by agreement works good for overlapping objects.
- ▶ Offers robustness to affine transformations.
- ▶ Activation functions are easier to interpret, e.g rotation, thickness.

## Cons

- ▶ Slow training due to routing by agreement algorithm.
- ▶ Cannot see two very identical objects.
- ▶ Not tested yet on larger images, e.g ImageNet.

# References



Geron, A. *Capsule Networks - Tutorial*.

<https://www.youtube.com/watch?v=pPN8d0E3900>



Hui, J. *Understanding Dynamic Routing between Capsules (Capsule Networks)* <https://jhui.github.io/2017/11/03/Dynamic-Routing-Between-Capsules/>



Liao, H. *CapsNet-Tensorflow*.

<https://github.com/naturomics/CapsNet-Tensorflow>



Pechyonkin, M. *Capsnet Series*. [https:](https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition)

[//medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition](https://medium.com/ai%C2%B3-theory-practice-business/understanding-hintons-capsule-networks-part-i-intuition)



Sabour, S., Frosst, N., & Hinton, G. E. *Dynamic Routing between Capsules*. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, arXiv:1710.09829v2 7 Nov 2017.